QCB Scientific Programming 2019

Release 0.1.0

Luca Bianco

Nov 05, 2019

Contents

1 General Info **1**
- 1.1 Timetable and lecture rooms . . . 1
- 1.2 Tutoring . . . 1
- 1.3 Slides . . . 1
- 1.4 Acknowledgements . . . 1

2 Practical 1 **3**
- 2.1 Slides . . . 3
- 2.2 Setting up the environment . . . 3
- 2.3 The console . . . 7
- 2.4 Visual Studio Code . . . 9
- 2.5 The debugger . . . 15
- 2.6 Exercises . . . 19

3 Practical 2 **23**
- 3.1 Slides . . . 23
- 3.2 Modules . . . 23
- 3.3 Objects . . . 24
- 3.4 Variables . . . 25
- 3.5 Numeric types . . . 26
- 3.6 Strings . . . 29
- 3.7 Exercises . . . 34

4 Practical 3 **43**
- 4.1 Slides . . . 43
- 4.2 Lists . . . 43
- 4.3 Tuples . . . 52
- 4.4 Exercises . . . 54

5 Practical 4 **63**
- 5.1 Slides . . . 63
- 5.2 Execution flow . . . 63
- 5.3 Conditionals . . . 63
- 5.4 Loops . . . 66
- 5.5 Exercises . . . 70

6 Practical 5 **79**

6.1 Slides . . . 79
6.2 More on loops . . . 79
6.3 Dictionaries . . . 84
6.4 Exercises . . . 88

7 Practical 6 **101**
7.1 Slides . . . 101
7.2 Functions . . . 101
7.3 Namespace and variable scope . . . 104
7.4 Argument passing . . . 105
7.5 File input and output . . . 109
7.6 Exercises . . . 114

8 Practical 7 **133**
8.1 Slides . . . 133
8.2 Functions . . . 133
8.3 Getting input from the command line . . . 133
8.4 Argparse . . . 135
8.5 Exercises . . . 139

9 Practical 8 **147**
9.1 Slides . . . 147
9.2 Libraries installation . . . 147
9.3 Pandas . . . 148
9.4 Series . . . 148
9.5 Plotting data . . . 159
9.6 Pandas DataFrames . . . 163
9.7 Exercises . . . 182

10 Practical 9 **207**
10.1 Slides . . . 207
10.2 Numpy . . . 207
10.3 Numpy ndarray . . . 207
10.4 Random arrays . . . 211
10.5 Numpy to and from pandas . . . 214
10.6 Reshaping . . . 215
10.7 Iterating over arrays and Indexing . . . 216
10.8 Broadcasting and arithmetic functions . . . 219
10.9 Filtering . . . 222
10.10 Linear algebra . . . 225
10.11 Matplotlib . . . 227
10.12 Exercises . . . 229

11 Practical 10 **293**
11.1 Slides . . . 293
11.2 Biopython . . . 293
11.3 Installation of Biopython . . . 293
11.4 General overview . . . 294
11.5 Sequence objects . . . 295
11.6 Looping, slicing and concatenation . . . 295
11.7 Sequence annotation . . . 302
11.8 Sequence I/O . . . 303
11.9 Multiple sequence alignment . . . 309
11.10 Exercises . . . 319

12 Practical 11 **329**
12.1 Slides . . . 329
12.2 Biopython . . . 329
12.3 BLAST . . . 329
12.4 Getting data from NCBI . . . 352
12.5 Getting data from ExPASy . . . 355
12.6 3D structure and PDB . . . 356
12.7 Exercises . . . 425

13 Midterm test simulation **467**
13.1 Before you start . . . 467
13.2 Problem 1 . . . 467
13.3 Problem 2 . . . 468

14 **475**
14.1 Index . . . 475

CHAPTER 1

General Info

The contacts to reach me can be found at this page[1].

1.1 Timetable and lecture rooms

Practicals will take place on Tuesdays from 15:30 to 17:30 (in lecture room A107) and on Thursdays from 15:30 to 17:30 (in lecture room A107). This first part of the course will tentatively run from 23/09/2019 to 05/11/2019.

1.2 Tutoring

Samuel Valentini (samuel.valentini) *at* unitn.it is the tutor for this part of the course. You can contact him and go to his office on Wednesdays from 10:00 to 11:00.

1.3 Slides

Slides of the practicals will be available on the top part of each practical page.

1.4 Acknowledgements

I would like to thank Dr. David Leoni for all his help and for sharing Jupman with me. I would also like to thank Dr. Stefano Teso for allowing us to use some of his material of a previous course.

```
In [ ]:
```

[1] http://www.fmach.it/CRI/info-generali/organizzazione/Biologia-computazionale/BIANCO-LUCA

CHAPTER 2

Practical 1

In this practical we will set up a working Python3 development environment and will start familiarizing a bit with Python.

2.1 Slides

The slides shown in the introduction can be found here: Intro

2.2 Setting up the environment

We will need to install several pieces of software to get a working programming environment suitable for this practical. In this section we will install everything that we are going to need in the next few weeks.

Python3 is available for Windows, Linux and Mac.

2.2.1 Our toolbox

If you work on Windows or Mac, you can safely skip the following and go straight to the section "Windows/Mac installation". Note that, regardless your operating system, a useful source of information on how to install python can be found here[2].

Linux on windows

If your computer has Windows installed you have several options to get it to run Linux:

1. This video tutorial (only in Italian) shows you how to set up a usb stick to run Linux from it: https://youtu.be/8_SK8iEMyJk

[2] http://docs.python-guide.org/en/latest/

2. You can install a virtualization software like vmware player[3] and download the .iso file of a linux distribution like ubuntu.[4] and install/run it from vmware player. For more information you can look at this tutorial.[5] Another option is to install virtual box[6].

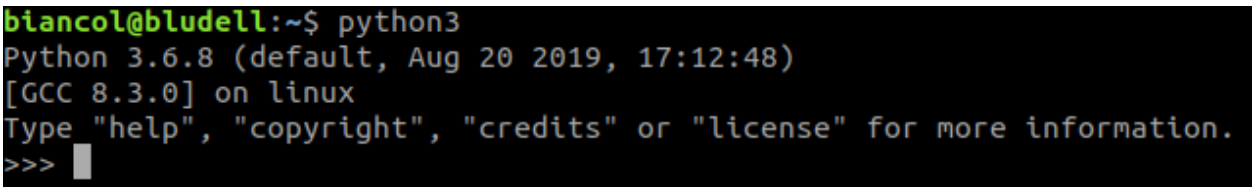
Here[7] you can find some VDI images that you can load in virtual box or in vmware player with several different operating systems. Please refer to this guide[8] for more information (for information on vmware please click on **VM IMAGES –> VMware IMAGES** in the menu of the page).

Linux installation

1. The Python interpreter. In this course we will use python version 3.x. A lot of information on python can be found on the python web page[9]. Open a terminal and try typing in:

   ```
   python3
   ```

   if you get an error like "python3 command not found" you need to install it, while if you get something like this (note that the version might be different):



   you are already sorted, just press Ctrl-D to exit.

   Installation on a debian-like linux distribution (e.g. Ubuntu) can be done by typing the following commands on a terminal:

   ```
   sudo apt-get update
   ```

   ```
   sudo apt-get install python3
   ```

   While **if you are using Fedora** you can use:

   ```
   sudo dnf install python3
   ```

2. Install now the package manager pip, which is a very convenient tool to install python packages, with the following command (**on Fedora, the command above should have already installed it**):

   ```
   sudo apt-get install python3-pip
   ```

   Note:

   If pip is already installed in your system you will get a message like: python3-pip is already the newest version (3.x.y)

[3] https://my.vmware.com/en/web/vmware/free#desktop_end_user_computing/vmware_workstation_player/15_0%7CPLAYER-1550%7Cproduct_downloads
[4] https://ubuntu.com/#download
[5] https://www.youtube.com/watch?v=lIhRkea8LpA
[6] https://www.virtualbox.org/wiki/Downloads
[7] https://www.osboxes.org/virtualbox-images/
[8] https://www.osboxes.org/guide/
[9] https://www.python.org/

3. Finally, install the Integrated Development Environment (IDE) that we will be using. This is called Visual Studio Code and is available for all platforms. You can read about it here[10]. Downloads for all platforms can be found here[11]. On a debian-like distribution go to the folder where you downloaded the .deb package and type:

   ```
   sudo dpkg -i code*.deb
   ```

   While **if you are using Fedora** you can use:

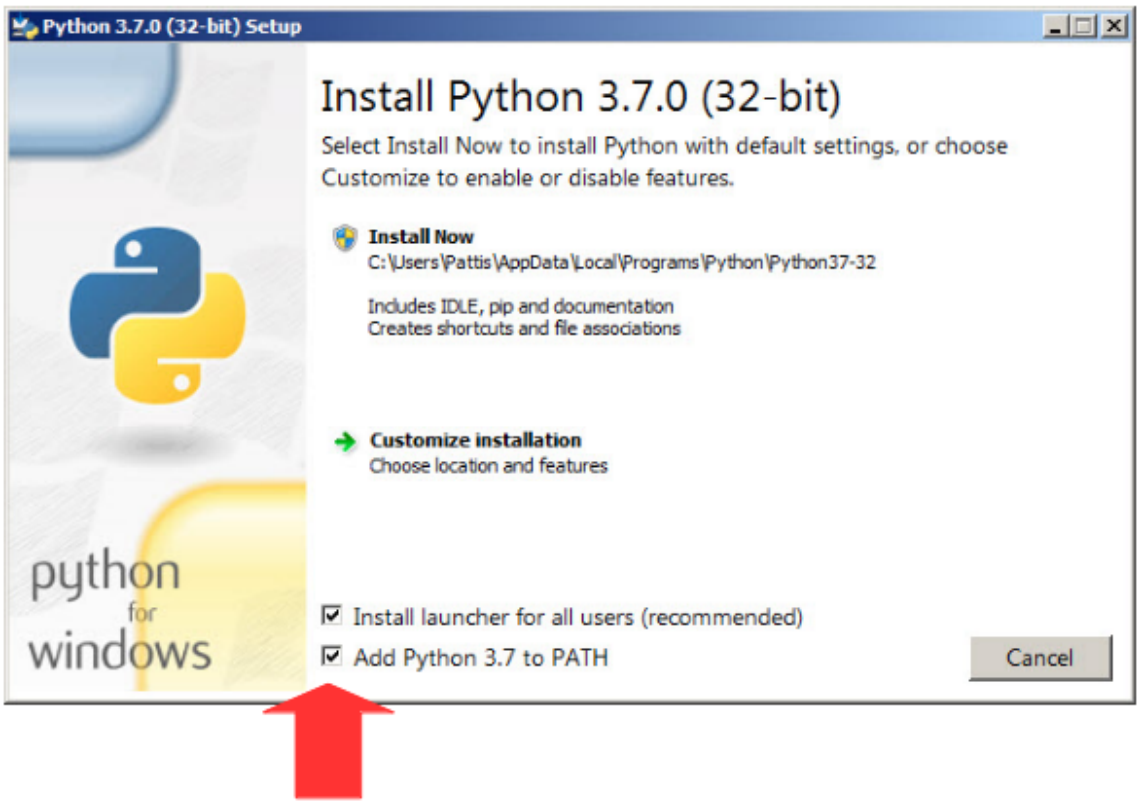
   ```
   sudo dnf install code*.rpm
   ```

Windows/Mac installation

OPTION 1:

1. The python interpreter. In this course we will use python version 3.x. A lot of information on python can be found on the python web page[12]. Installers for Windows and Mac can be downloaded from this page[13]. Click on Download Python 3.7.x. **Please refrain from double-clicking on the installer and read below first.**

   > **Attention! Important note**
   >
   > When executing the installer, please remember to tick the flag "Add Python 3.7.x to PATH" and then click on Install now (see picture below).



[10] https://code.visualstudio.com/
[11] https://code.visualstudio.com/Download
[12] https://www.python.org/
[13] https://www.python.org/downloads/

2. Install now the Integrated Development Environment (IDE) that we will be using. This is called Visual Studio Code and is available for all platforms. You can read about it here[14]. Downloads for all platforms can be found here[15].

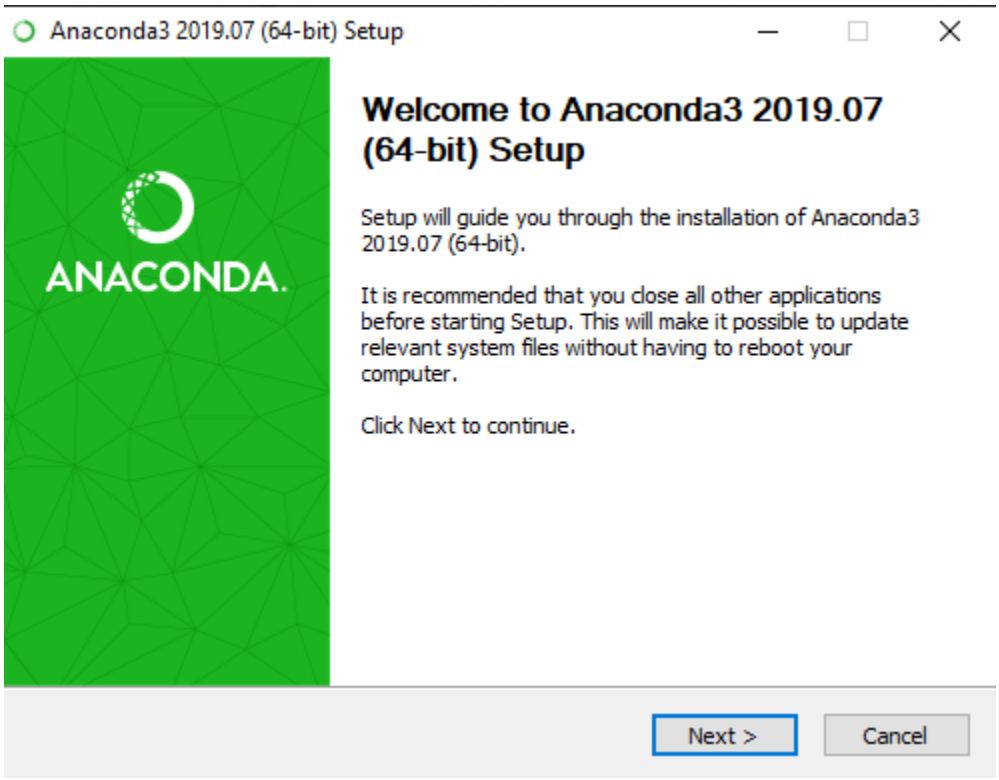
OPTION 2 (easier):

Additional Information:

It is also possible to install python through the Anaconda package manager. You can install Visual Studio Code together with Anaconda(the Anaconda installer will ask if you want it, just say yes!).

Anaconda is available here[16]

Upon launching the installer you should be prompted something like:



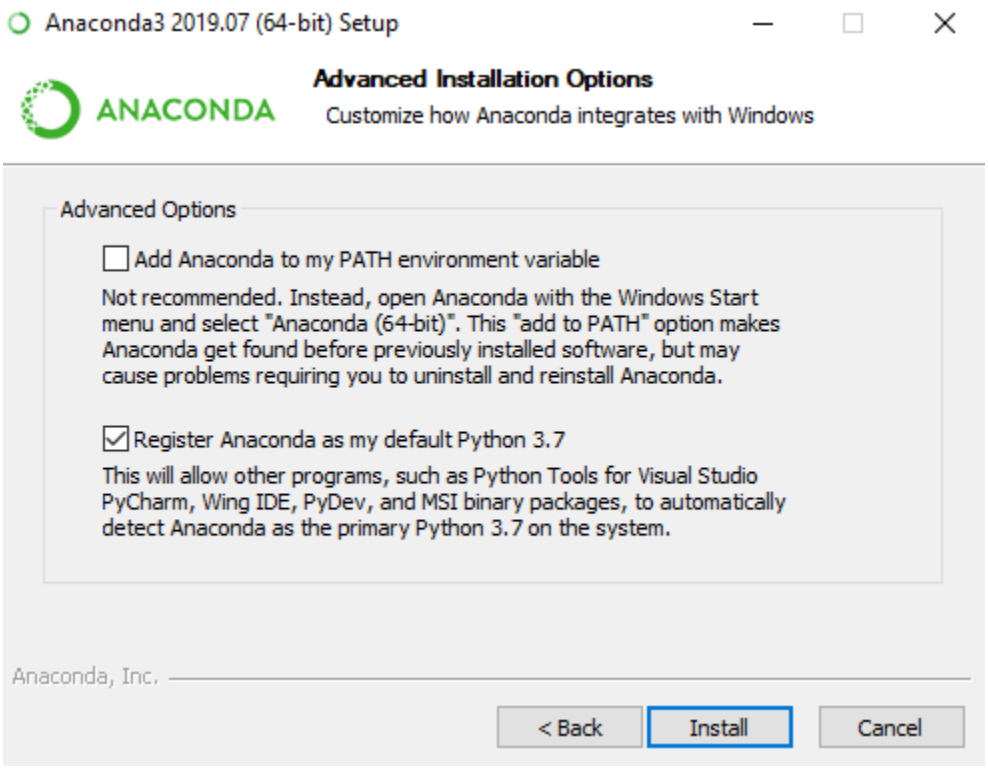
at the next step flag the correct items as in the figure below:

When installation is complete, start anaconda through the **Anaconda Navigator** in the windows menu. When the navigator starts, you should see a screen like:

from which you can install Visual Studio Code as IDE.

For more information please have a look here[17].

[14] https://code.visualstudio.com/
[15] https://code.visualstudio.com/Download
[16] https://www.anaconda.com/distribution/
[17] https://docs.anaconda.com/anaconda/user-guide/getting-started/#open-nav-win



2.3 The console

To access the console on Linux just open a terminal and type:

`python3`

while in Windows you have to look for "Python" and run "Python 3.7". The console should look like this:

Now we are all set to start interacting with the Python interpreter. In the console, type the following instructions:

```
In [1]: 5 + 3
```

```
Out[1]: 8
```

All as expected. The "In [1]" line is the input, while the "Out [1]" reports the output of the interpreter. Let's challenge python with some other operations:

```
In [2]: 12 / 5
```

```
Out[2]: 2.4
```

```
In [3]: 1/133
```

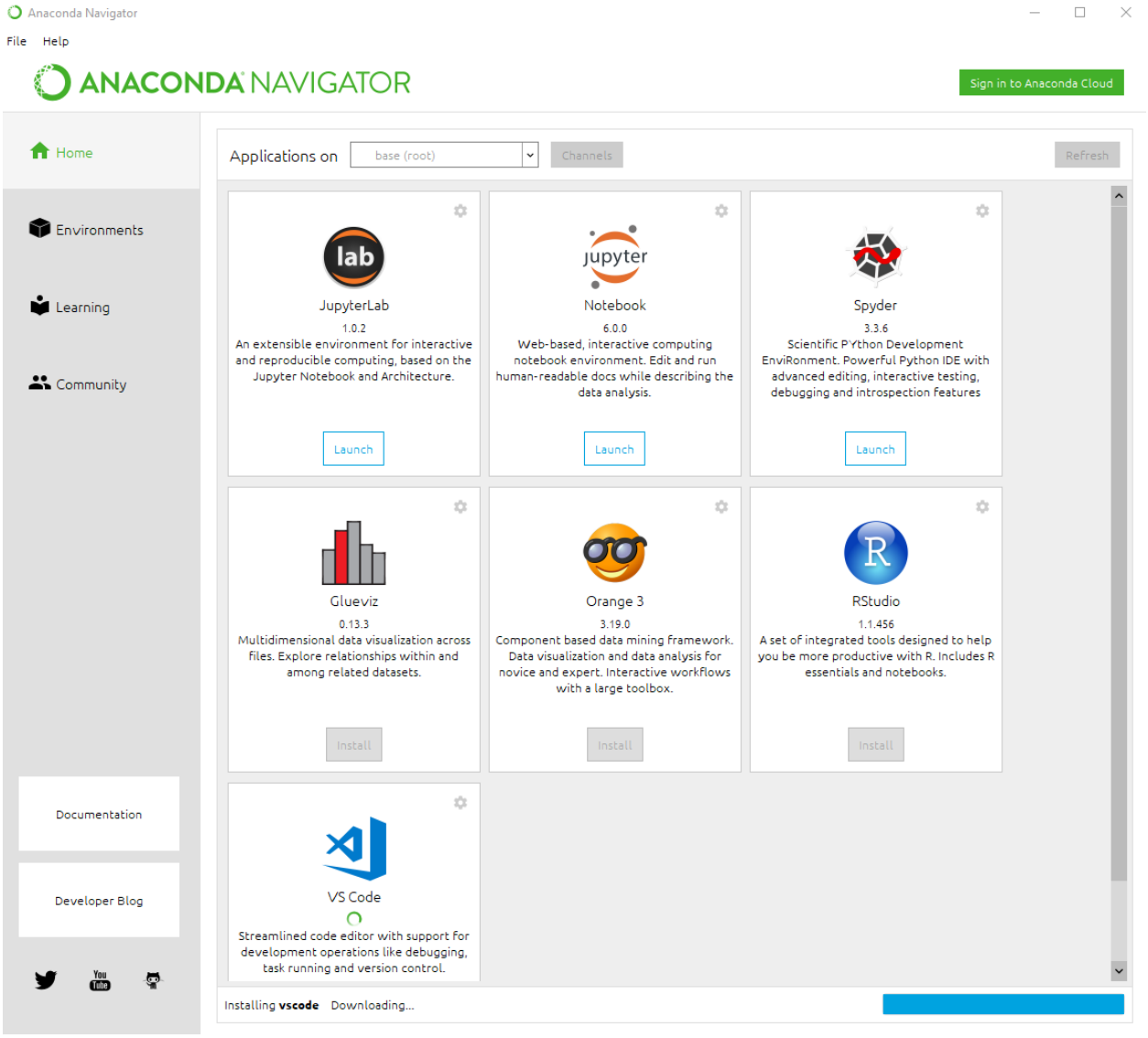
```
Out[3]: 0.007518796992481203
```

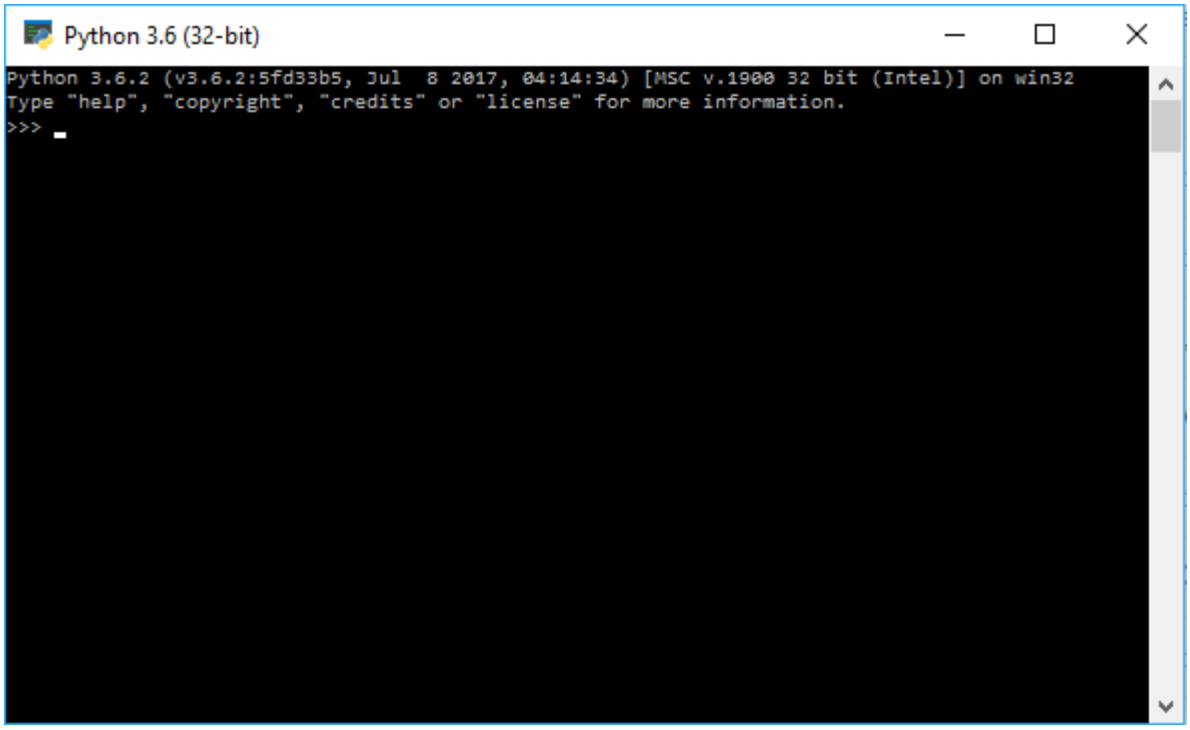
```
In [4]: 2**1000
```

```
Out[4]: 10715086071862673209484250490600018105614048117055336074437503883703510511249361224931983788
```

And some assignments:

```
In [5]: a = 10
        b = 7
        s = a + b
        d = a / b
```


Anaconda Navigator
File Help
ANACONDA NAVIGATOR
Sign in to Anaconda Cloud
Home
Environments
Learning
Community
Documentation
Developer Blog
Applications on
base (root)
Channels
Refresh
JupyterLab
1.0.2
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.
Launch
Notebook
6.0.0
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.
Launch
Spyder
3.3.6
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features
Launch
Glueviz
0.13.3
Multidimensional data visualization across files. Explore relationships within and among related datasets.
Install
Orange 3
3.19.0
Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.
Install
RStudio
1.1.456
A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.
Install
VS Code
Streamlined code editor with support for development operations like debugging, task running and version control.
Installing vscode Downloading...




```
        print("sum is:",s, " division is:",d)
```

```
sum is: 17  division is: 1.4285714285714286
```

In the first four lines, values have been assigned to variables through the = operator. In the last line, the print function is used to display the output. For the time being, we will skip all the details and just notice that the print function somehow managed to get text and variables in input and coherently merged them in an output text. Although quite useful in some occasions, the console is quite limited therefore you can close it for now. To exit press Ctrl-D or exit().

2.4 Visual Studio Code

Once you open the IDE Visual Studio Code you will see the welcome screen:

You can find useful information on this tool here[18]. Please spend some time having a look at that page.

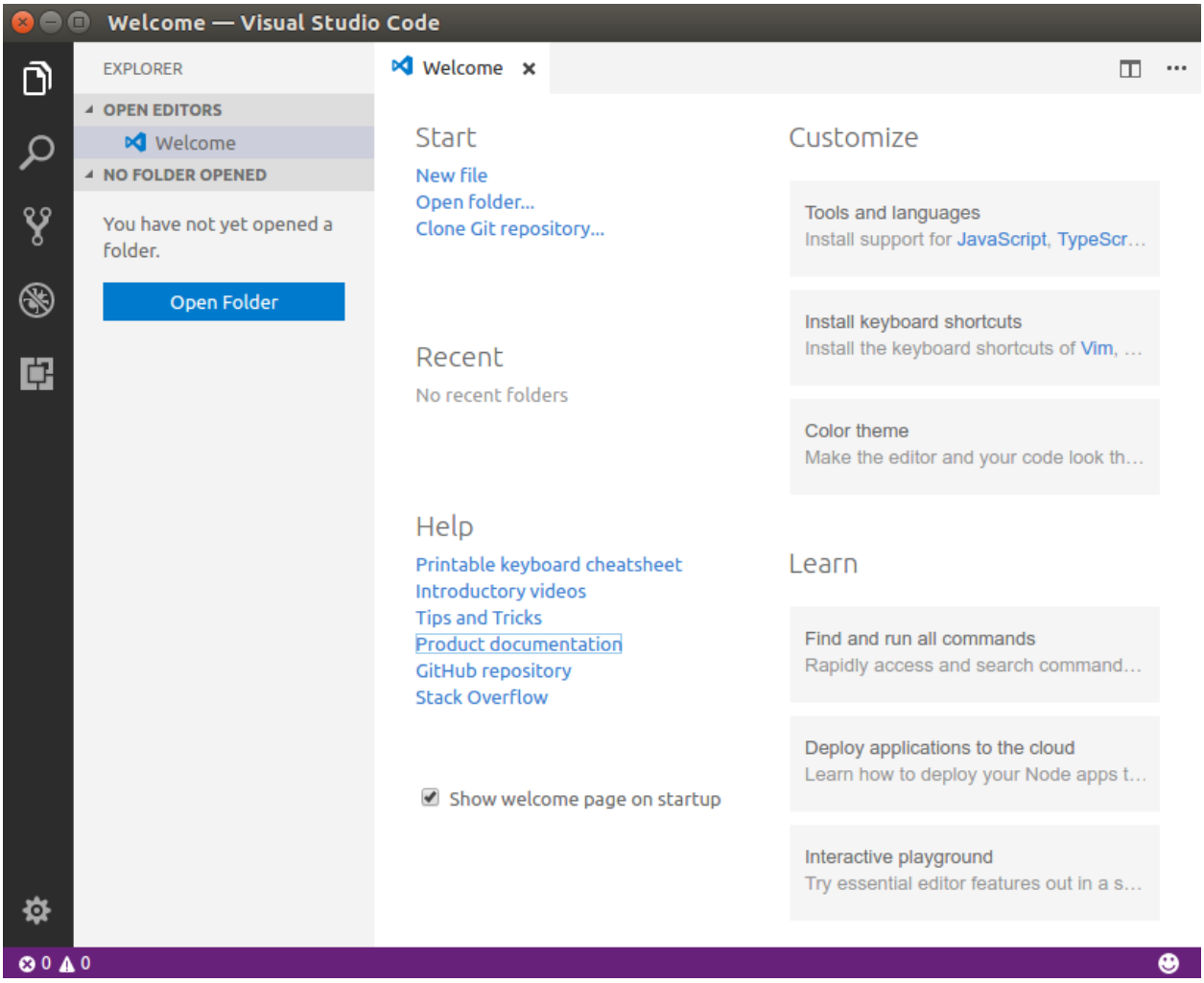
Once you are done with it you can close this window pressing on the "x". First thing to do is to set the python interpreter to use. Click on View –> Command Palette and type "Python" in the text search space. Select **Python: Select Workspace Interpreter** as shown in the picture below.

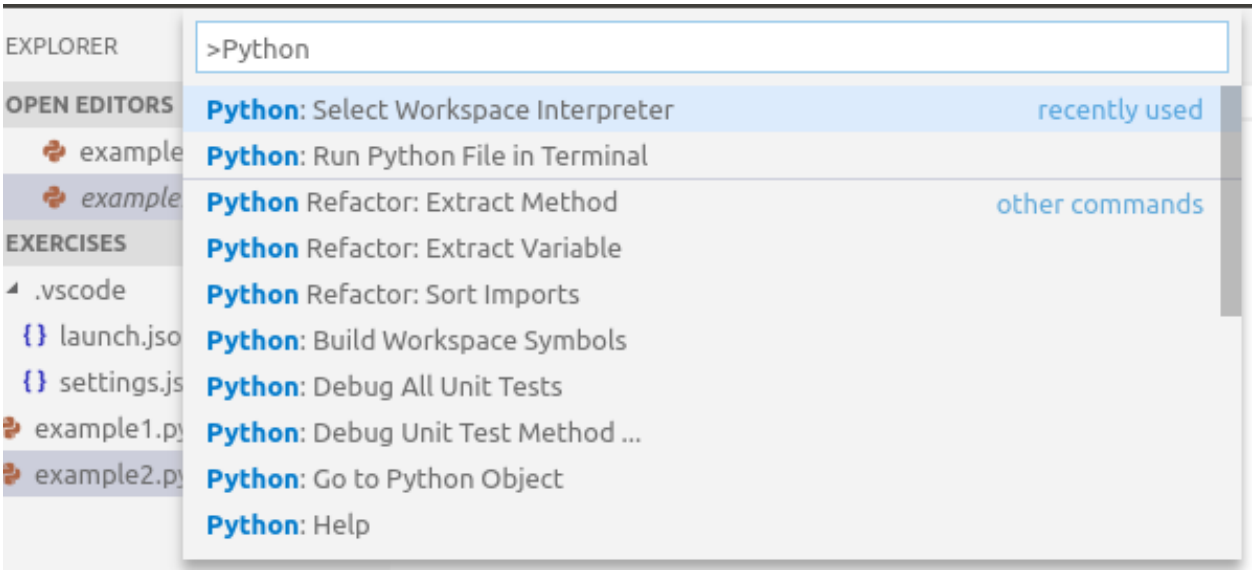
Finally, select the python version you want to use (e.g. Python3).

Now you can click on **Open Folder** to create a new folder to place all the scripts you are going to create. You can call it something like "exercises". Next you can create a new file, *example1.py* (*.py* extension stands for python).

Visual Studio Code will understand that you are writing Python code and will help you writing valid syntax in your scripts.

[18] https://code.visualstudio.com/docs#vscode


Welcome — Visual Studio Code
EXPLORER
OPEN EDITORS
Welcome
NO FOLDER OPENED
You have not yet opened a folder.
Open Folder
Welcome
Start
New file
Open folder...
Clone Git repository...
Recent
No recent folders
Help
Printable keyboard cheatsheet
Introductory videos
Tips and Tricks
Product documentation
GitHub repository
Stack Overflow
Show welcome page on startup
Customize
Tools and languages
Install support for JavaScript, TypeScr...
Install keyboard shortcuts
Install the keyboard shortcuts of Vim, ...
Color theme
Make the editor and your code look th...
Learn
Find and run all commands
Rapidly access and search command...
Deploy applications to the cloud
Learn how to deploy your Node apps t...
Interactive playground
Try essential editor features out in a s...
0 0


EXPLORER
OPEN EDITORS
example
example
EXERCISES
.vscode
launch.jso
settings.js
example1.py
example2.py
>Python
Python: Select Workspace Interpreter
recently used
Python: Run Python File in Terminal
Python Refactor: Extract Method
other commands
Python Refactor: Extract Variable
Python Refactor: Sort Imports
Python: Build Workspace Symbols
Python: Debug All Unit Tests
Python: Debug Unit Test Method ...
Python: Go to Python Object
Python: Help

Warning:

If you get the following error message:

click on **Install Pylint** which is a useful tool to help your coding experience.

Add the following text to your *example1.py* file.

```
In [6]: """
        This is the first example of Python script.
        """
        a = 10 # variable a
        b = 33 # variable b
        c = a / b # variable c holds the ratio

        # Let's print the result to screen.
        print("a:", a, " b:", b, " a/b=", c)
```

```
a: 10  b: 33  a/b= 0.30303030303030304
```

A couple of things worth nothing. The first three lines opened and closed by """ are some text describing the content of the script. Moreover, comments are proceeded by the hash key (#) and they are just ignored by the python interpreter.

Note

Good *Pythonic* code follows some syntactic rules on how to write things, naming conventions etc. The IDE will help you writing pythonic code even though we will not enforce this too much in this course. If you are interested in getting more details on this, you can have a look at the PEP8 Python Style Guide[19] (Python Enanchement Proposals - index 8).

Warning

Please remember to comment your code, as it helps readability and will make your life easier when you have to modify or just understand the code you wrote some time in the past.**

Please notice that Visual Studio Code will help you writing your Python scripts. For example, when you start writing the **print** line it will complete the code for you (**if the Pylint extension mentioned above is installed**), suggesting the functions that match the letters written. This useful feature is called *code completion* and, alongside suggesting possible matches, it also visualizes a description of the function and parameters it needs. Here is an example:

Save the file (Ctrl+S as shortcut). It is convenient to ask the IDE to highlight potential *syntactic* problems found in the code. You can toggle this function on/off by clicking on **View –> Problems**. The *Problems* panel should look like this

Visual Studio Code is warning us that the variable names *a,b,c* at lines 4,5,6 do not follow Python naming conventions for constants. This is because they have been defined at the top level (there is no structure to our script yet) and therefore are interpreted as constants. The naming convention for constants states that they should be in capital letters. To amend the code, you can just replace all the names with the corresponding capitalized name (i.e. A,B,C). If you do that, and you save the file again (Ctrl+S), you will see all these problems disappearing as well as the green underlining of the variable names. If your code does not have an empty line before the end, you might get another warning "*Final new line missing*".

Info

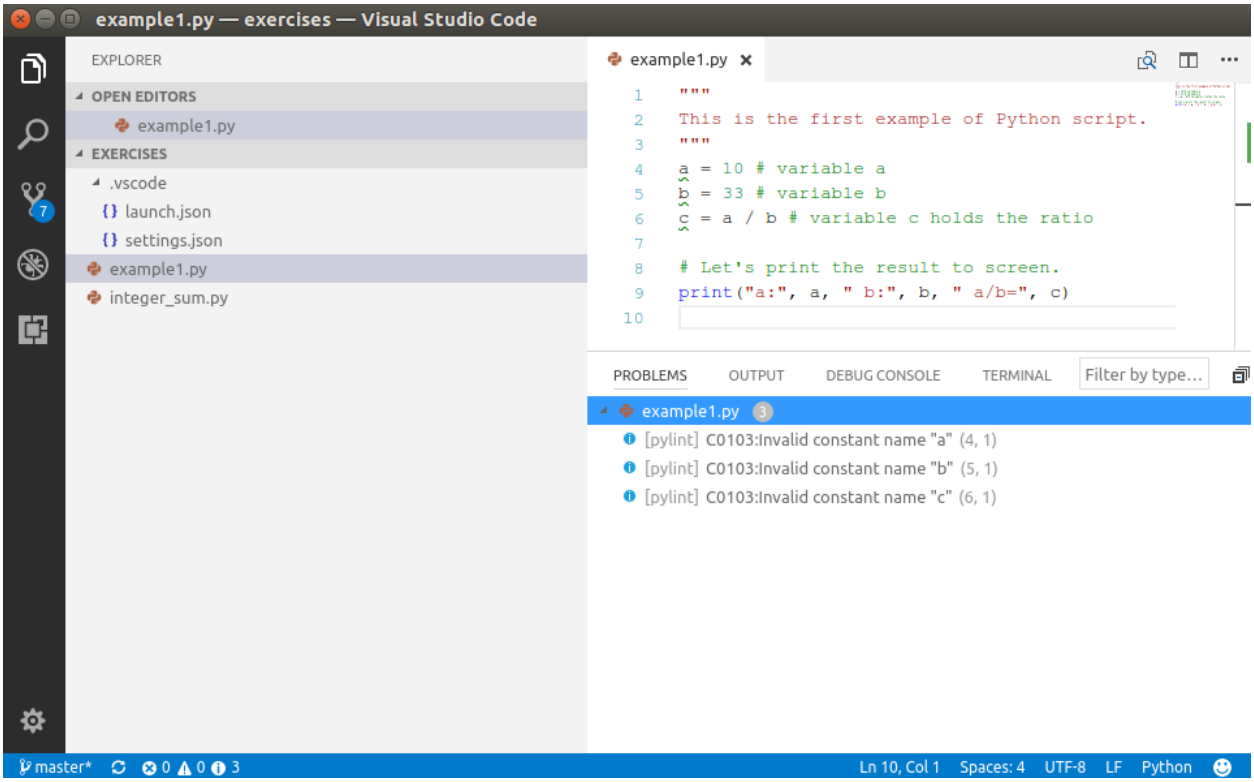
[19] https://www.python.org/dev/peps/pep-0008/

```
pri
```

print

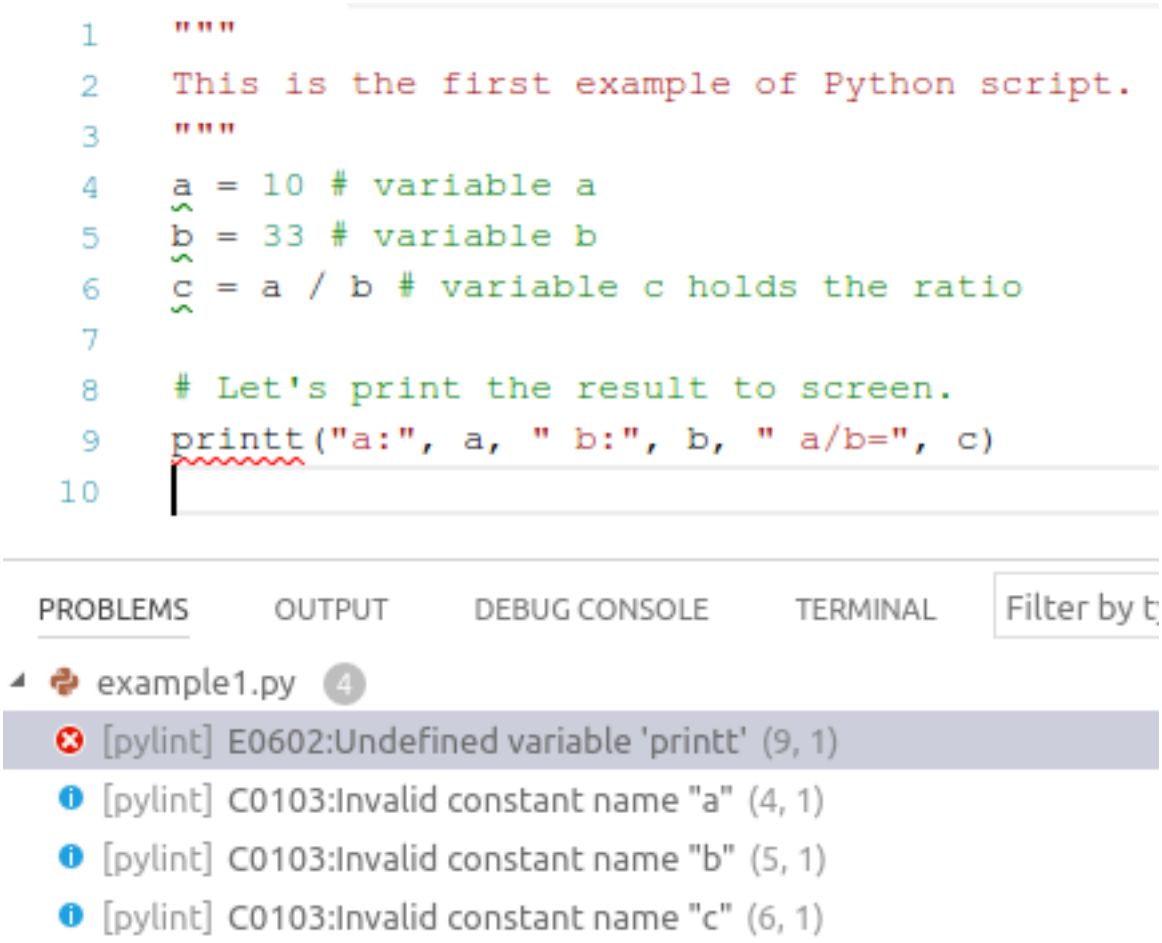
```
def print(value, ..., sep=' ', end
='\n', file=sys.stdout, flush=Fals
e)
```

Prints the values to a stream, or to sys.stdout by default.
Optional keyword arguments:
file: a file-like object (stream); defaults to the current sys.stdout.
sep: string inserted between values, default a space.
end: string appended after the last value, default



Note that these were just warnings and the interpreter **in this case** will happily and correctly execute the code anyway, but it is always good practice to understand what the warnings are telling us before deciding to ignore them!

Had we by mistake mispelled the *print* function name (something that should not happen with the code completion tool that suggests functions names!) writing *printt* (note the double t), upon saving the file, the IDE would have underlined in red the function name and flagged it up as a problem.



This is because the builtin function *printt* does not exist and the python interpreter does not know what to do when it reads it. Note that *printt* is actually underlined in red, meaning that there is an error which will cause the interpreter to stop the execution with a failure. **Please remember that before running any piece of code all errors must be fixed**.

Now it is time to execute the code. By right-clicking in the code panel and selecting **Run Python File in Terminal** (see picture below) you can execute the code you have just written.

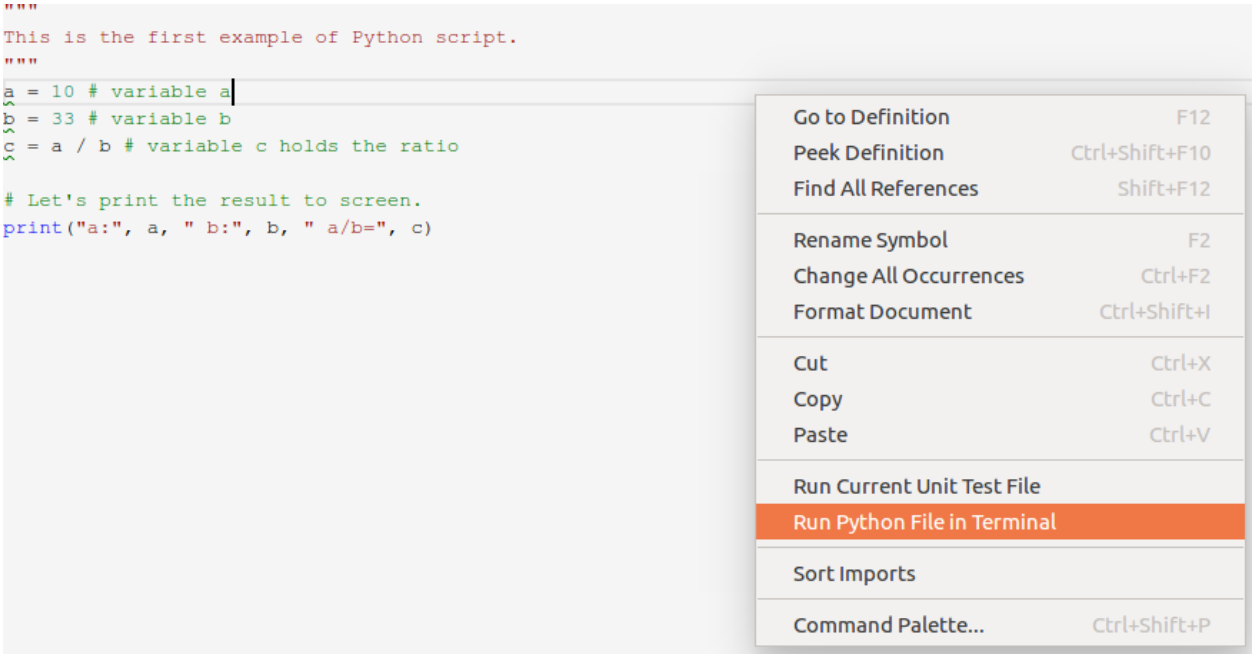
Upon clicking on *Run Python File in Terminal* a terminal panel should pop up in the lower section of the coding panel and the result shown above should be reported.

Saving script files like the *example1.py* above is also handy because they can be invoked several times (later on we will learn how to get inputs from the command line to make them more useful...). To do so, you just need to call the python intepreter passing the script file as parameter. From the folder containing the *example1.py* script:

```
python3 example1.py
```

will in fact return:

a: 10 b: 33 a/b= 0.30303030303030304



Before ending this section, let me add another note on errors. The IDE will diligently point you out *syntactic* warnings and errors (i.e. errors/warnings concerning the structure of the written code like name of functions, number and type of parameters, etc.) but it will not detect *semantic* or *runtime* errors (i.e. connected to the meaning of your code or to the value of your variables). These sort of errors will most probably make your code crash or may result in unexpected results/behaviours. In the next section we will introduce the debugger, which is a useful tool to help detecting these errors.

Before getting into that, consider the following lines of code (do not focus on the *import* line, this is only to load the mathematics module and use its method *sqrt*):

```
In [7]: """
        Runtime error example, compute square root of numbers
        """
        import math

        A = 16
        B = math.sqrt(A)
        C = 5*B
        print("A:", A, " B:", B, " C:", C)

        D = math.sqrt(A-C) # whoops, A-C is now -4!!!
        print(D)
A: 16  B: 4.0  C: 20.0

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-7-5d4ed1b10924> in <module>
      9 print("A:", A, " B:", B, " C:", C)
     10
---> 11 D = math.sqrt(A-C) # whoops, A-C is now -4!!!
     12 print(D)

ValueError: math domain error
```

If you add that code to a python file (e.g. sqrt_example.py), you save it and you try to execute it, you should get an

error message as reported above. You can see that the interpreter has happily printed off the vaule of A,B and C but then stumbled into an error at line 9 (math domain error) when trying to compute $\sqrt{A - C} = \sqrt{-4}$, because the sqrt method of the math module cannot be applied to negative values (i.e. it works in the domain of real numbers).

Please take some time to familiarize with Visual Studio Code (creating files, saving files etc.) as in the next practicals we will take this ability for granted.

2.5 The debugger

Another important feature of advanced Integrated Development Environments (IDEs) is their debugging capabilities. Visual Studio Code comes with a debugging tool that can help you trace the execution of your code and understand where possible errors hide.

Write the following code on a new file (let's call it *integer_sum.py*) and execute it to get the result.

```
In [8]: """ integer_sum.py is a script to
         compute the sum of the first 1200 integers. """

        S = 0
        for i in range(0, 1201):
            S = S + i

        print("The sum of the first 1200 integers is: ", S)
```

```
The sum of the first 1200 integers is:  720600
```

Without getting into too many details, the code you just wrote starts initializing a variable S to zero, and then loops from 0 to 1200 assigning each time the value to a variable i, accumulating the sum of S + i in the variable S.

A final thing to notice is indentation.

Info

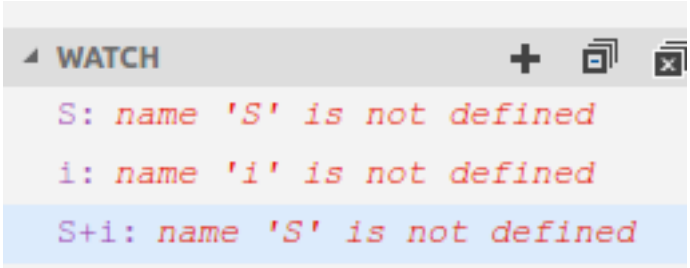
In Python it is important to indent the code properly as this provides the right scope for variables (e.g. see that the line *S = S + 1* starts more to the right than the previous and following line – this is because it is inside the for loop). You do not have to worry about this for the time being, we will get to this in a later practical...

How does this code work? How does the value of S and i change as the code is executed? These are questions that can be answered by the debugger.

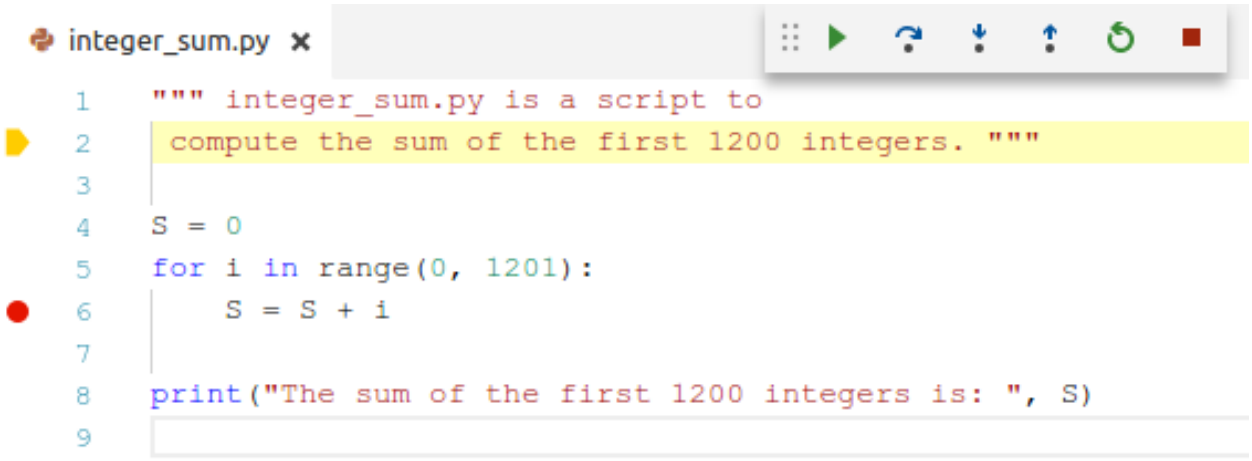
To start the debugger, click on **Debug –> Start Debugging** (shortcut F5). The following small panel should pop up:

We will use it shortly, but before that, let's focus on what we want to track. On the left hand side of the main panel, a *Watch* panel appeared. This is where we need to add the things we want to monitor as the execution of the program goes. With respect to the code written above, we are interested in keeping an eye on the variables S, i and also of the expression S+i (that will give us the value of S of the next iteration). Add these three expressions in the watch panel (click on **+** to add new expressions). The watch panel should look like this:

do not worry about the message "*name X is not defined*", this is normal as no execution has taken place yet and the interpreter still does not know the value of these expressions.



The final thing before starting to debug is to set some breakpoints, places where the execution will stop so that we can check the value of the watched expressions. This can be done by hovering with the mouse on the left of the line number. A small reddish dot should appear, place the mouse over the correct line (e.g. the line corresponding to $S = S + 1$ and click to add the breakpoint (a red dot should appear once you click).



Now we are ready to start debugging the code. Click on the green triangle on the small debug panel and you will see that the yellow arrow moved to the breakpoint and that the watch panel updated the value of all our expressions.

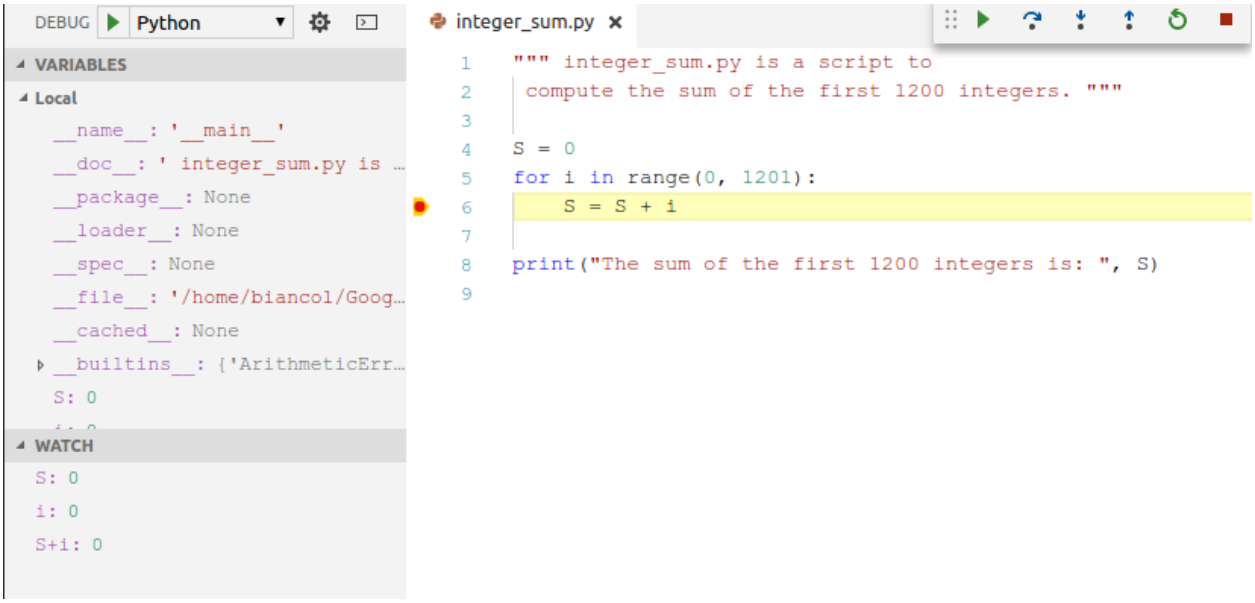
The value of all expressions is zero because the debugger stopped **before** executing the code specified at the breakpoint line (recall that S is initialized to 0 and that i will range from 0 to 1200). If you click again on the green arrow, execution will continue until the next breakpoint (we are in a for loop, so this will be again the same line - trust me for the time being).

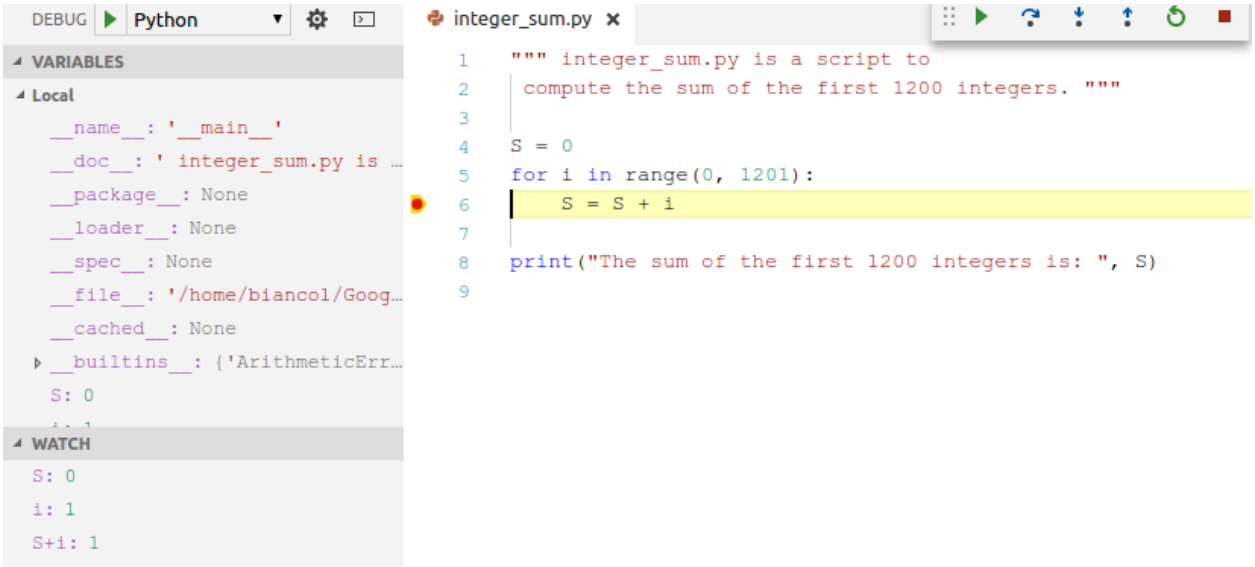
Now i has been increased to 1, S is still 0 (remember that the execution stopped **before** executing the code at the breakpoint) and therefore S + i is now 1. Click one more time on the green arrow and values should update accordingly (i.e. S to 1, i to 2 and S + i to 3), another round of execution should update S to 3, i to 3 and S + i to 6. Got how this works? Variable i is increased by one each time, while S increases by i. You can go on for a few more iterations and see if this makes any sense to you, once you are done with debugging you can stop the execution by pressing the red square on the small debug panel.

Other editors are available, if you already have your favourite one you can stick to it. Some examples are:

- Spyder[20]
- PyCharm Community Edition[21]
- Jupyter Notebook[22]. Note: we might use it later on in the course.

[20] https://pythonhosted.org/spyder/installation.html
[21] https://www.jetbrains.com/pycharm/
[22] http://jupyter.org/


DEBUG
Python
VARIABLES
Local
__name__: '__main__'
__doc__: ' integer_sum.py is …
__package__: None
__loader__: None
__spec__: None
__file__: '/home/biancol/Goog…
__cached__: None
__builtins__: {'ArithmeticErr…
S: 0
WATCH
S: 0
i: 0
S+i: 0
integer_sum.py
""" integer_sum.py is a script to
compute the sum of the first 1200 integers. """
S = 0
for i in range(0, 1201):
S = S + i
print("The sum of the first 1200 integers is: ", S)


DEBUG
Python
VARIABLES
Local
__name__: '__main__'
__doc__: ' integer_sum.py is …
__package__: None
__loader__: None
__spec__: None
__file__: '/home/biancol/Goog…
__cached__: None
__builtins__: {'ArithmeticErr…
S: 0
WATCH
S: 0
i: 1
S+i: 1
integer_sum.py
""" integer_sum.py is a script to
compute the sum of the first 1200 integers. """
S = 0
for i in range(0, 1201):
S = S + i
print("The sum of the first 1200 integers is: ", S)

2.5.1 A quick Jupyter primer (just for your information)

Jupyter allows to write notebooks organized in cells (these can be saved in files with .ipynb extension). Notebooks contain both the **code**, some **text describing the code** and the **output of the code execution**, they are quite useful to produce some quick reports on data analysis. where there is both code, output of running that code and text. The code by default is Python, but can also support other languages like R). The text is formatted using the **Markdown language**[23] - see cheatsheet[24] for its details. *Jupyter is becoming the de-facto standard for writing technical documentation*.

Installation

To install it (if you have not installed python with Anaconda otherwise you should have it already):

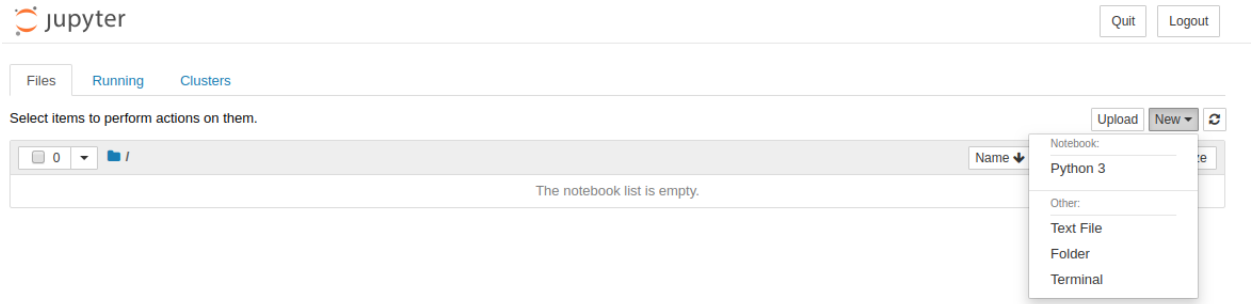
```
python3 -m pip install jupyter
```

you can find more information here[25]

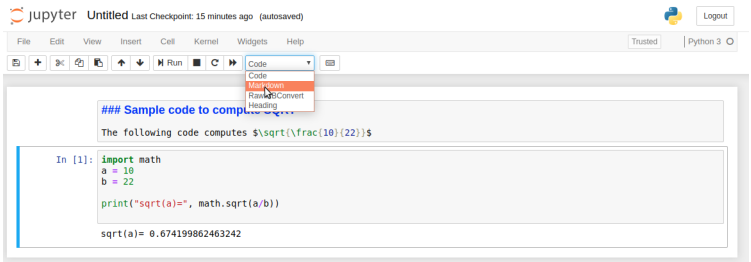
Upon successful installation, you can run it with:

```
jupyter-notebook
```

This should fire up a browser on a page where you can start creating your notebooks or modifying existing ones. To create a new notebook you simply click on **New**:

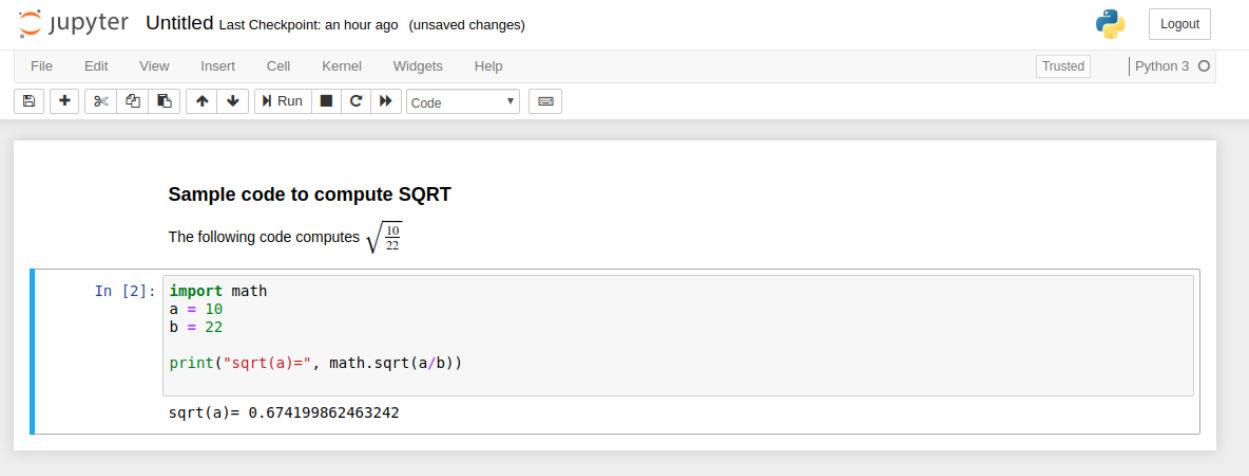


and then you can start adding cells (i.e. containers of code and text). The type of each cell is specified by selecting the cell and selecting the right type in the dropdown list:



Cells can be executed by clicking on the **Run** button. This will get the code to execute (and output to be written) and text to be processed to provide the final page layout. To go back to the edit mode, just double click on an executed cell.

[23] https://en.wikipedia.org/wiki/Markdown
[24] https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet
[25] https://jupyter.org/install



Please take some more time to familiarize with Visual Studio Code (creating files, saving files, interacting with the debugger etc.) as in the next practicals we will take this ability for granted. Once you are done you can move on and do the following exercises.

2.6 Exercises

1. Compute the area of a triangle having base 120 units (B) and height 33 (H). Assign the result to a variable named area and print it.

Show/Hide Solution

```
In [9]: B = 120
        H = 33
        Area = B*H/2
        print("Triangle area is:", Area)
```

```
Triangle area is: 1980.0
```

2. Compute the area of a square having side (S) equal to 145 units. Assign the result to a variable named area and print it.

Show/Hide Solution

```
In [20]: S = 145
         Area = S*S
         print("Square area is:",Area)
```

```
Square area is: 21025
```

3. Modify the program at point 2. to acquire the side S from the user at runtime. (Hint: *use the input function and remember to covert the acquired value into an int*).

Show/Hide Solution

```
In [24]: S_str = input("Insert size: ")
         print(type(S_str))
         print(S_str)
         S = int(S_str)
         print(type(S))
         print(S)
         Area = S**2
         print("Square area is:",Area)
```

```
Insert size: 33
<class 'str'>
33
<class 'int'>
33
Square area is: 1089
```

4. If you have not done so already, put the two previous scripts in two separate files (e.g. triangle_area.py and square_area.py and execute them from the terminal).

5. Write a small script (trapezoid.py) that computes the area of a trapezoid having major base (MB) equal to 30 units, minor base (mb) equal to 12 and height (H) equal to 17. Print the resulting area. Try executing the script from inside Visual Studio Code and from the terminal.

Show/Hide Solution

```
In [12]: """trapezoid.py"""
         MB = 30
         mb = 12
         H = 17
         Area = (MB + mb)*H/2
         print("Trapezoid area is: ", Area)
```

```
Trapezoid area is:  357.0
```

6. Rewrite the example of the sum of the first 1200 integers by using the following equation: $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$.

Show/Hide Solution

```
In [13]: N = 1200

         print("Sum of first 1200 integers: ", N*(N+1)/2)
```

```
Sum of first 1200 integers:  720600.0
```

7. Modify the program at point 6. to make it acquire the number of integers to sum N from the user at runtime.

Show/Hide Solution

```
In [14]: print("Input number N:")
         N = int(input())
         print("Sum of first ", N, " integers: ", N*(N+1)/2)
```

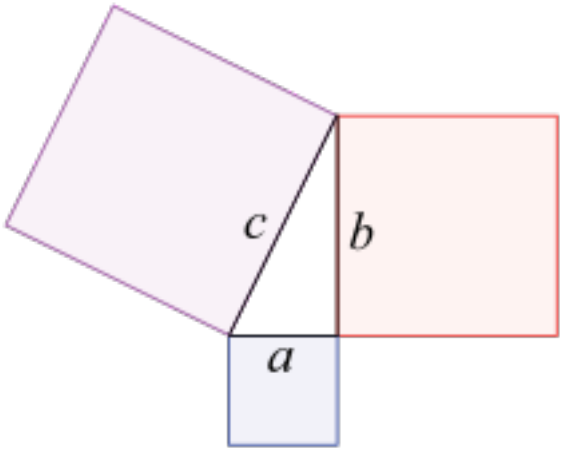
```
Input number N:
77
Sum of first  77  integers:  3003.0
```

8. Write a small script to compute the length of the hypotenuse (c) of a right triangle having sides a=133 and b=72 units (see picture below). Hint: *remember the Pythagorean theorem and use math.sqrt*).

Show/Hide Solution

```
In [15]: import math

         a = 133
         b = 72

         c = math.sqrt(a**2 + b**2)

         print("Hypotenuse: ", c)
```

```
Hypotenuse:  151.23822268196622
```



9. Rewrite the trapezoid script making it compute the area of the trapezoid starting from the major base (MB), minor base (mb) and height (H) taken in input. (Hint: *use the input function and remember to convert the acquired value into an int*).

Show/Hide Solution

```
In [16]: """trapezoidV2.py"""
         MB = int(input("Input the major base (MB):"))
         mb = int(input("Input the minor base (mb):"))
         H = int(input("Input the height (H):"))
         Area = (MB + mb)*H/2
         print("Given MB:", str(MB) , " mb:", str(mb) , " and H:", H)
         print("The trapezoid area is: ", Area)
```

```
Input the major base (MB):33
Input the minor base (mb):22
Input the height (H):10
Given MB: 33  mb: 22  and H: 10
The trapezoid area is:  275.0
```

10. Write a script that reads the side of an heagon in input and computes its perimeter and area printing them to the screen. *Hint:* $Area = \frac{3*\sqrt{3}*side^2}{2}$

Show/Hide Solution

```
In [17]: import math

         side = int(input("Please insert the side of the hexagon: "))

         P = 6*side
         A = (3*math.sqrt(3)*side**2)/2
         print("Perimeter: ", P, " Area: ", A)
```

```
Please insert the side of the hexagon: 9
Perimeter:  54  Area:  210.4441731196186
```

CHAPTER 3

Practical 2

In this practical we will start interacting more with Python, practicing on how to handle data, functions and methods. We will see several built-in data types and then dive deeper into strings.

3.1 Slides

The slides of the introduction can be found here: Intro

3.2 Modules

Python modules are simply text files having the extension **.py** (e.g. `exercise.py`). When you were writing the code in the IDE in the previous practical, you were in fact implementing the corresponding module.

As said in the previous practical, once you implemented and saved the code of the module, you can execute it by typing

```
python3 exercise1.py
```

or, in Visual Studio Code, by right clicking on the code panel and selecting **Run Python File in Terminal**.

A Module A can be loaded from another module B so that B can use the functions defined in A. Remember when we used the `sqrt` function? It is defined in the module math. To import it and use it we indeed wrote something like:

```
In [1]: import math

        A = math.sqrt(4)
        print(A)
```

```
2.0
```

Note

When importing modules we do not need to specify the extension ".py" of the file.

3.3 Objects

Python understands very well objects, and in fact everything is an object in Python.

Objects have **properties** (characteristic features) and **methods** (things they can do). For example, an object *car* caould be defined to have the **properties** *model, make, color, number of doors, position* etc., and the **methods** *steer right, steer left, accelerate, break, stop, change gear, repaint,...* whose application might affect the state of the object.

According to Python's official documentation:

"Objects are Python's abstraction for data. All data in a Python program is represented by objects or by relations between objects."

All you need to know for now is that in Python objects have an **identifier (ID)** (i.e. their name), a **type** (numbers, text, collections,...) and a **value** (the actual data represented by the objects). Once an object has been created the *identifier* and the *type* never change, while its *value* can either change (**mutable objects**) or stay constant (**immutable objects**).

Python provides the following built-in data types:

Type	Meaning	Domain	Mutable?
`bool`	Condition	`True`, `False`	No
`int`	Integer	$\{-2^{-63}, \ldots, 2^{63}-1\} \subset \mathbb{Z}$	No
~~`long`~~	~~Integer~~	~~$\mathbb{Z}$~~	~~No~~
`long`	Integer	$\mathbb{Z}$	No
`float`	Rational	$\mathbb{Q}$ (more or less)	No
`str`	Text	Text	No
`list`	Sequence	Collections of things	Yes
`tuple`	Sequence	Collections of things	No
`dict`	Map	Maps between things	Yes

We will stick with the simplest ones for now, but later on we will dive deeper into the all of them.

3.4 Variables

Variables are just references to objects, in other words they are the **name** given to an object. Variables can be **assigned** to objects by using the assignment operator =.

The instruction

```
In [2]: sides = 4
```

might represent the number of sides of a square. What happens when we execute it in Python? An object is created, it is given an identifier, its type is set to "int" (an integer number), it value to 4 and a name *sides* is placed in the current namespace to point to that object, so that after that instruction we can access that object through its name. The type of an object can be accessed with the function **type()** and the identifier with the function **id()**:

```
In [3]: sides = 4
        print( type(sides) )
        print( id(sides) )
```

```
<class 'int'>
10919520
```

Consider now the following code:

```
In [4]: sides = 4 #a square
        print ("value:", sides, " type:", type(sides), " id:", id(sides))
        sides = 5 #a pentagon
        print ("value:", sides, " type:", type(sides), " id:", id(sides))
```

```
value: 4  type: <class 'int'>  id: 10919520
value: 5  type: <class 'int'>  id: 10919552
```

The value of the variable sides has been changed from 4 to 5, but as stated in the table above, the type `int` is **immutable**. Luckily, this did not prevent us to change the value of sides from 4 to 5. What happened behind the scenes when we executed the instruction sides = 5 is that a new object has been created of type int (5 is still an integer) and it has been made accessible with the same name *sides*, but since it is a different object (i.e. the integer 5) you can see that the identifier is actually different.

Note: You do not have to really worry about what happens behind the scenes, as the Python interpreter will take care of these aspects for you, but it is nice to know what it does.

You can even change the type of a variable during execution but that is normally a **bad idea** as it makes understanding the code more complicated.

You can do (**but, please, refrain!**):

```
In [5]: sides = 4 #a square
        print ("value:", sides, " type:", type(sides), " id:", id(sides))
        sides = "four" #the sides in text format
        print ("value:", sides, " type:", type(sides), " id:", id(sides))
```

```
value: 4  type: <class 'int'>  id: 10919520
value: four  type: <class 'str'>  id: 139909167957192
```

IMPORTANT NOTE: You can chose the name that you like for your variables (I advise to pick something reminding their meaning), but you need to adhere to some simple rules.

1. Names can only contain upper/lower case digits (`A-Z`, `a-z`), numbers (`0-9`) or underscores _;
2. Names cannot start with a number;

3. Names cannot be equal to reserved keywords:

3.5 Numeric types

We already mentioned that numbers are **immutable objects**. Python provides different numeric types: integers, booleans, reals (floats) and even complex numbers and fractions (but we will not get into those).

3.5.1 Integers

Their range of values is limited only by the memory available. As we have already seen, python provides also a set of standard operators to work with numbers:

```
In [6]: a = 7
        b = 4

        a + b # 11
        a - b # 3
        a // b # integer division: 1
        a * b # 28
        a ** b # power: 2401
        a / b # division 0.8333333333333334
        type(a / b)
```

```
Out[6]: float
```

Note that in the latter case the result is no more an integer, but a float (we will get to that later).

3.5.2 Booleans

These objects are used for the boolean algebra. Truth values are represented with the keywords `True` and `False` in Python. A boolean object can only have value `True` or `False`. We can convert booleans into integers with the builtin function `int`. Any integer can be converted into a boolean (and vice-versa) with:

```
In [7]: a = bool(1)
        b = bool(0)
        c = bool(72)
        d = bool(-5)
        t = int(True)
        f = int(False)

        print("a: ", a, " b: ", b, " c: ", c, " d: ", d , " t: ", t, " f: ", f)
```

```
a:  True  b:  False  c:  True  d:  True  t:  1  f:  0
```

any integer is evaluated to true, except 0. Note that, the truth values `True` and `False` respectively behave like the integers 1 and 0.

We can operate on boolean values with the boolean operators `and`, `or`, `not`. Recall boolean algebra for their use:

```
In [8]: T = True
        F = False

        print ("T: ", T, " F:", F)

        print ("T and F: ", T and F) #False
```

```
        print ("T and T: ", T and T) #True
        print ("F and F: ", F and F) #False
        print ("not T: ", not T) # False
        print ("not F: ", not F) # True
        print ("T or F: ", T or F) # True
        print ("T or T: ", T or T) # True
        print ("F or F: ", F or F) # False
```

```
T:  True  F: False
T and F:  False
T and T:  True
F and F:  False
not T:  False
not F:  True
T or F:  True
T or T:  True
F or F:  False
```

Numeric comparators are operators that return a boolean value. Here are some examples (from the lecture):

`a == b`	True if and only if $a = b$
`a != b`	True if and only if $a \neq b$
`a < b`	True if and only if $a < b$
`a > b`	True if and only if $a > b$
`a <= b`	True if and only if $a \leq b$
`a >= b`	True if and only if $a \geq b$

Example: Given a variable a = 10 and a variable b = 77, let's swap their values (i.e. at the end a will be equal to 77 and b to 10). Let's also check the values at the beginning and at the end.

```
In [9]: a = 10
        b =  77
        print("a: ", a, " b:", b)
        print("is a equal to 10?", a == 10)
        print("is b equal to 77?", b == 77)

        TMP = b  #we need to store the value of b safely
        b = a    #ok, the old value of b is gone... is it?
        a = TMP  #a gets the old value of b... :-)

        print("a: ", a, " b:", b)
        print("is a equal to 10?", a == 10)
        print("is a equal to 77?", a == 77)
        print("is b equal to 10?", b == 10)
        print("is b equal to 77?", b == 77)
```

```
a:  10  b: 77
```

```
is a equal to 10? True
is b equal to 77? True
a:  77  b: 10
is a equal to 10? False
is a equal to 77? True
is b equal to 10? True
is b equal to 77? False
```

3.5.3 Real numbers

Python stores real numbers (floating point numbers) in 64 bits of information divided in sign, exponent and mantissa.

Example: Let's calculate the area of the center circle of a football pitch (radius = 9.15m) recalling that $area = Pi*R^2$:

```
In [10]: R = 9.15
         Pi = 3.1415926536
         Area = Pi*(R**2)
         print (Area)
```

```
263.02199094102605
```

Note that the parenthesis around the `R**2` are not necessary as operator `**` has the precedence, but I personally think it helps readability.

Here is a reminder of the precedence of operators:

**	Power (Highest precedence)
+,-	Unary plus and minus
* / // %	Multiply, divide, floor division, modulo
+ -	Addition and subtraction
<= < > >=	Comparison operators
== !=	Equality operators
not or and	Logical operators (Lowest precedence)

Example: Let's compute the GC content of a DNA sequence 33 base pairs long, having 12 As, 9 Ts, 5 Cs and 7Gs. The GC content can be expressed by the formula: $gc = \frac{G+C}{A+T+C+G}$ where A,T,C,G represent the number of nucleotides of each kind. What is the AT content? Is the GC content higher than the AT content?

```
In [11]: A = 12
         T = 9
         C = 5
         G = 7

         gc = (G+C)/(A+T+C+G)

         print("The GC content is: ", gc)

         at = 1 - gc

         print (gc > at)
```

```
The GC content is:  0.36363636363636365
False
```

3.6 Strings

Strings are **immutable objects** (note the actual type is **str**) used by python to handle text data. Strings are sequences of *unicode code points* that can represent characters, but also formatting information (e.g. '\n' for new line). Unlike other programming languages, python does not have the data type character, which is represented as a string of length 1.

There are several ways to define a string:

```
In [12]: S = "my first string, in double quotes"

         S1 = 'my second string, in single quotes'

         S2 = '''my third string is
         in triple quotes
         therefore it can span several lines'''

         S3 = """my fourth string, in triple double-quotes
         can also span
         several lines"""

         print(S, '\n') #let's add a new line at the end of the string with \n
         print(S1,'\n')
         print(S2, '\n')
         print(S3, '\n')
```

```
my first string, in double quotes

my second string, in single quotes

my third string is
in triple quotes
therefore it can span several lines

my fourth string, in triple double-quotes
can also span
several lines
```

To put special characters like ',' and so on you need to "escape them" (i.e. write them following a back-slash).

\\	Backslash
\n	ASCII linefeed (also known as newline)
\t	ASCII tab character
\'	Single quote
\"	Double quote
\xxxx	Unicode character xxxx (hexadecimal)

Example: Let's print a string containing a quote and double quote (i.e. ' and ").

```
In [13]: myString = "This is how I \'quote\' and \"double quote\" things in strings"
         print(myString)
```

```
This is how I 'quote' and "double quote" things in strings
```

Strings can be converted to and from numbers with the functions `str()`, `int()` or `float()`.

Example: Let's define a string *myString* with the value "47001" and convert it into an int. Try adding one and print the result.

```
In [14]: myString = "47001"
         print(myString, " has type ", type(myString))

         myInt = int(myString)

         print(myInt, " has type ", type(myInt))

         myInt = myInt + 1    #adds one

         myString = myString + "1" #cannot add 1 (we need to use a string).
                                   #This will append 1 at the end of the string

         print(myInt)
         print(myString)
```

```
47001  has type  <class 'str'>
47001  has type  <class 'int'>
47002
470011
```

Python defines some operators to work with strings. Recall the slides shown during the lecture:

Result	Operator	Meaning
int	len(str)	Return the length of the string
str	str + str	Concatenate two strings
str	str * int	Replicate the string
bool	str in str	Check if a string is present in another string
str	str[int]	Read the character at specified index
str	str[int:int]	Extract a sub-string

Example A tandem repeat is a short sequence of DNA that is repeated several times in a row. Let's create a string representing the tandem repeat of the motif "ATTCG" repeated 5 times. What is the length of the whole repetitive region? Is the motif "TCGAT" (m1) present in the region? The motif "TCCT" (m2)? Let's give an orientation to the tandem repeat by adding the string "5'-" (5' end) on the left and "-3'" (3' end) to the right.

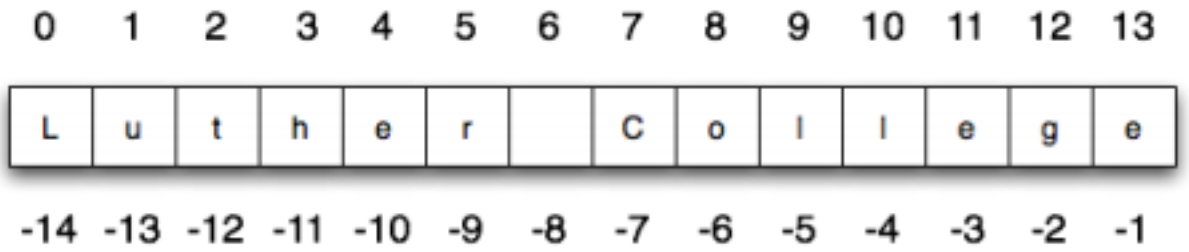
```
In [15]: motif = "ATTCG"

         tandem_repeat = motif * 5

         print(motif)
         print(tandem_repeat, " has length", len(tandem_repeat))
         m1 = "TCGAT"
         m2 = "TCCT"

         print("Is ", m1, " in ", tandem_repeat, " ? ", m1 in tandem_repeat )
```

```
        print("Is ", m2, " in ", tandem_repeat, " ? ", m2 in tandem_repeat )
        oriented_tr = "5\'-" + tandem_repeat + "-3\'"
        print(oriented_tr)
```

```
ATTCG
ATTCGATTCGATTCGATTCGATTCG  has length 25
Is  TCGAT  in  ATTCGATTCGATTCGATTCGATTCG  ?  True
Is  TCCT  in  ATTCGATTCGATTCGATTCGATTCG  ?  False
5'-ATTCGATTCGATTCGATTCGATTCG-3'
```

We can access strings at specific positions (indexing) or get a substring starting from a position S to a position E. The only thing to remember is that numbering starts from 0. The`i`-th character of a string can be accessed as `str[i-1]`. Substrings can be accessed as `str[S:E]`, optionally a third parameter can be specified to set the step (i.e. `str[S:E:STEP]`).

Important note. Remember that when you do str[S:E], **S is inclusive, while E is exclusive** (see S[0:6] below).



Let's see these aspects in action with an example:

```
In [16]: S = "Luther College"
```

```
        print(S) #print the whole string
        print(S == S[:]) #a fancy way of making a copy of the original string
        print(S[0]) #first character
        print(S[3]) #fourth character
        print(S[-1]) #last character
        print(S[0:6]) #first six characters
        print(S[-7:]) #final seven characters
        print(S[0:len(S):2]) #every other character starting from the first
        print(S[1:len(S):2]) #every other character starting from the second
```

```
Luther College
True
L
h
e
Luther
College
Lte olg
uhrClee
```

3.6.1 Methods for the str object

The object `str` has some methods that can be applied to it (remember methods are things you can do on objects). Recall from the lecture that the main methods are:

Result	Method	Meaning
str	str.upper()	Return the string in upper case
str	str.lower()	Return the string in lower case
str	str.strip(str)	Remove strings from the sides
str	str.lstrip(str)	Remove strings from the left
str	str.rstrip(str)	Remove strings from the right
str	str.replace(str, str)	Replace substrings
bool	str.startswith(str)	Check if the string starts with another
bool	str.endswith(str)	Check if the string ends with another
int	str.find(str)	Return the first position of a substring starting from the left
int	str.rfind(str)	Return the position of a substring starting from the right
int	str.count(str)	Count the number of occurrences of a substring

IMPORTANT NOTE: Since Strings are immutable, every operation that changes the string actually produces a new *str* object having the modified string as value.

Recall that **strings are immutable**. For this reason we cannot directly change them with an assignment operator.

Example: Since the genetic code is degenerate, there are many codons encoding for the same aminoacid. Consider Proline, it can be encoded by the following codons: CCU, CCA,CCG, CCC. Let's create a string proline and assign it to its possible codons one after the other.

```
In [17]: """
         Wrong solution. We cannot directly replace the value of a string
         """

         proline = "CCU"
         print("Proline can be encoded by: ", proline)
         proline[2]="A"
         print(".. or by: ", proline)


Proline can be encoded by:  CCU
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-17-9750dcfa1cbd> in <module>()
      5 proline = "CCU"
      6 print("Proline can be encoded by: ", proline)
----> 7 proline[2]="A"
      8 print(".. or by: ", proline)
      9

TypeError: 'str' object does not support item assignment

In [18]: """
```

```
        Correct solution. Using str.replace
        """
        proline = "CCU"
        print("Proline can be encoded by: ", proline)
        proline = proline.replace("U","A")
        print(".. or by: ", proline)
        proline = proline.replace("A","G")
        print(".. or by: ", proline)
        proline = proline.replace("G","C")
        print(".. or by: ", proline)
```

```
Proline can be encoded by:  CCU
.. or by:  CCA
.. or by:  CCG
.. or by:  CCC
```

```
In [19]: """
        Another correct solution. Using string slicing and catenation.
        """
        proline = "CCU"
        print("Proline can be encoded by: ", proline)
        proline = proline[:-1]+"A" #equal to proline[0:-1] or proline[0:2]
        print(".. or by: ", proline)
        proline = proline[:-1]+"G"
        print(".. or by: ", proline)
        proline = proline[:-1]+"C"
        print(".. or by: ", proline)
```

```
Proline can be encoded by:  CCU
.. or by:  CCA
.. or by:  CCG
.. or by:  CCC
```

Example: Given the DNA sequence S = ” aTATGCCCATatcgctAAATTGCTGCCATTACA “. Print its length (removing any blank spaces at either sides), the number of adenines, cytosines, guanines and thymines present. Is the sequence “ATCG” present in S? Print how many times the substring “TGCC” appears in S and all the corresponding indexes.

```
In [20]: S = "   aTATGCCCATatcgctAAATTGCTGCCATTACA    "

        print(S)
        S = S.strip(" ")
        print(S)

        print(len(S))
        tmpS = S.upper() #for simplicity to count only 4 different nucleotides
        print("A count: ", tmpS.count("A"))
        print("C count: ", tmpS.count("C"))
        print("T count: ", tmpS.count("T"))
        print("G count: ", tmpS.count("G"))
        print("Is ATCG in ", tmpS, "? ", tmpS.find("ATCG") != -1) #or tmpS.count("ATCG") > 0
        print("TGCC is present ", tmpS.count("TGCC"), " times in ", tmpS)
        print("TGCC is present at pos ", tmpS.find("TGCC"))
        print("TGCC is present at pos ", tmpS.rfind("TGCC")) #or tmpS.find("TGCC",4)
```

```
   aTATGCCCATatcgctAAATTGCTGCCATTACA
aTATGCCCATatcgctAAATTGCTGCCATTACA
33
A count:  10
C count:  9
```

```
T count:  10
G count:  4
Is ATCG in  ATATGCCCATATCGCTAAATTGCTGCCATTACA ?  True
TGCC is present  2  times in  ATATGCCCATATCGCTAAATTGCTGCCATTACA
TGCC is present at pos  3
TGCC is present at pos  23
```

3.7 Exercises

1. Given the following string on two lines:

```
text = """Nobody said it was easy
No one ever said it would be this hard"""
```

write some python code that a)prints the whole string; b) prints the first and last character; c) prints the first 10 characters; d) prints from the 19th character to the 31st; e) prints the string all in capital letters.

Show/Hide Solution

```
In [19]: text = """Nobody said it was easy
         No one ever said it would be this hard"""
         # a) prints the whole text
         print(text)

         #some empty space...
         print("")

         # b) 1st and last character
         print("1st char: ", text[0], " last char: ", text[-1])

         #c) the 1st 10 characters:
         print("1st 10 chars:", text[0:10])

         #d) from the 20th to the 31st char
         print("\nCharacters from 19 to 31:")
         print(text[18:31])
         print("\nAll upper case:")
         print(text.upper())
```

```
Nobody said it was easy
No one ever said it would be this hard

1st char:  N  last char:  d
1st 10 chars: Nobody sai

Characters from 19 to 31:
easy
No one e

All upper case:
NOBODY SAID IT WAS EASY
NO ONE EVER SAID IT WOULD BE THIS HARD
```

2. An exon of a gene starts from position 12030 on a genome and ends at position 12174. Does an A/T SNP present at position 12111 affect this exon (i.e. is it inside the exon)? And what about a SNP present at position 12188? *Hint: test a suitable boolean expression to answer this question.*

Show/Hide Solution

```
In [3]: E_start = 12030
        E_end = 12174
        SNP1_pos = 12111
        SNP2_pos = 12188

        Test1 = (SNP1_pos >= E_start and SNP1_pos <=E_end)
        Test2 = (SNP2_pos >= E_start and SNP2_pos <=E_end)
        print ("SNP1 (", SNP1_pos,") in [", E_start, ",", E_end, "]?", Test1)
        print ("SNP2 (", SNP2_pos,") in [", E_start, ",", E_end, "]?", Test2)
```

```
SNP1 ( 12111 ) in [ 12030 , 12174 ]? True
SNP2 ( 12188 ) in [ 12030 , 12174 ]? False
```

3. SNP FB_AFFY_0000024 of the Apple 480K SNP chip has 5' flanking region (i.e. the forward probe) CATTATTTTCACTTGGGTCGAGGCCAGATTCCATC and 3' flanking region (i.e. the reverse probe) GGATTGCCCGAAATCAGAGAAAAGTCG. The SNP is a G/A transversion. Answer the following questions:

   (a) What is the length of the 5' flanking region? And that of the 3' flanking region?

   (b) The IUPAC code of the G/A transversion is R. What is the sequence of the whole region using the "[G/A]" notation for the SNP (hint: concatenate it in a new string called *region*) and the iupac notation R (region_iupac)?

   (c) Retrive and print only the SNP from *region* and *iupac_region*

Show/Hide Solution

```
In [8]: SNP_5prime =  "CATTATTTTCACTTGGGTCGAGGCCAGATTCCATC"
        SNP_3prime = "GGATTGCCCGAAATCAGAGAAAAGTCG"

        SNPseq = "G/A"
        SNPiupac = "R"

        print("Length of 5' end: ", len(SNP_5prime))
        print("Length of 3' end: ", len(SNP_3prime))
        region = SNP_5prime + "[" + SNPseq + "]" + SNP_3prime
        region_iupac = SNP_5prime + SNPiupac + SNP_3prime
        print(region)
        print(region_iupac)

        #string slicing and indexing!

        snp_from_region = region[ len(SNP_5prime) + 1 : len(SNP_5prime) + 4 ]
        snp_from_iupac = region_iupac[ len(SNP_5prime) ]

        print("SNP from region: ", snp_from_region)
        print("SNP from iupac region: ", snp_from_iupac)

        # Another way:
        #L_ind = region.find("[")
        #R_ind = region.find("]")
        #print(L_ind)
        #print(R_ind)

        #print(region[L_ind + 1 : R_ind])
```

```
Length of 5' end:  35
Length of 3' end:  27
CATTATTTTCACTTGGGTCGAGGCCAGATTCCATC[G/A]GGATTGCCCGAAATCAGAGAAAAGTCG
CATTATTTTCACTTGGGTCGAGGCCAGATTCCATCRGGATTGCCCGAAATCAGAGAAAAGTCG
SNP from region:  G/A
```

```
SNP from iupac region:  R
35
39
G/A
```

4. Compute the melting temperature T_m of the primer with sequence "TTAGCACACGTGAGCCAATGGAGCAAACGGGTAATT". The melting temperature T_m (in degrees Celtius) can be computed as: $T_m = 64.9 + 41(GC - 16.4)/N$, where GC is the total number of G and C in the primer and N is its length.

Show/Hide Solution

```
In [23]: primer = "TTAGCACACGTGAGCCAATGGAGCAAACGGGTAATT"
         N = len(primer)

         gc = (primer.count("G") + primer.count("C"))

         Tm = 64.9 + 41 * (gc - 16.4)/N

         print("The melting T for primer ", primer, " is: ", Tm, "°C")
```

```
The melting T for primer  TTAGCACACGTGAGCCAATGGAGCAAACGGGTAATT  is:  65.58333333333334 °C
```

5. Convert the following extract of the PalB2[26] gene into mRNA (i.e. replace thymine with uracile): CTGTCTCCCTCACTGTATGTAAATTGCATCTAGAATAGCA TCTGGAGCACTAATTGACACATAGTGGGTATCAATTATTA TTCCAGGTACTAGAGATACCTGGACCATTAACGGATAAAT AGAAGATTCATTTGTTGAGTGACTGAGGATGGCAGTTCCT GCTACCTTCAAGGATCTGGATGATGGGGAGAAACAGAGAA CATAGTGTGAGAATACTGTGGTAAGGAAAGTACAGAGGAC TGGTAGAGTGTCTAACCTAGATTTGGAGAAGGACCTAGAA GTCTATCCCAGGGAAATAAAAATCTAAGCTAAGGTTTGAG GAATCAGTAGGAATTGGCAAAGGAAGGACATGTTCCAGAT GATAGGAACAGGTTATGCAAAGATCCTGAAATGGTCAGAG CTTGGTGCTTTTTGAGAACCAAAAGTAGATTGTTATGGAC CAGTGCTACTCCCTGCCTCTTGCCAAGGGACCCCGCCAAG CAGTATTGTTGGTGT

and print the number of uracils present and the total length of the sequence (**remember to remove newlines**).

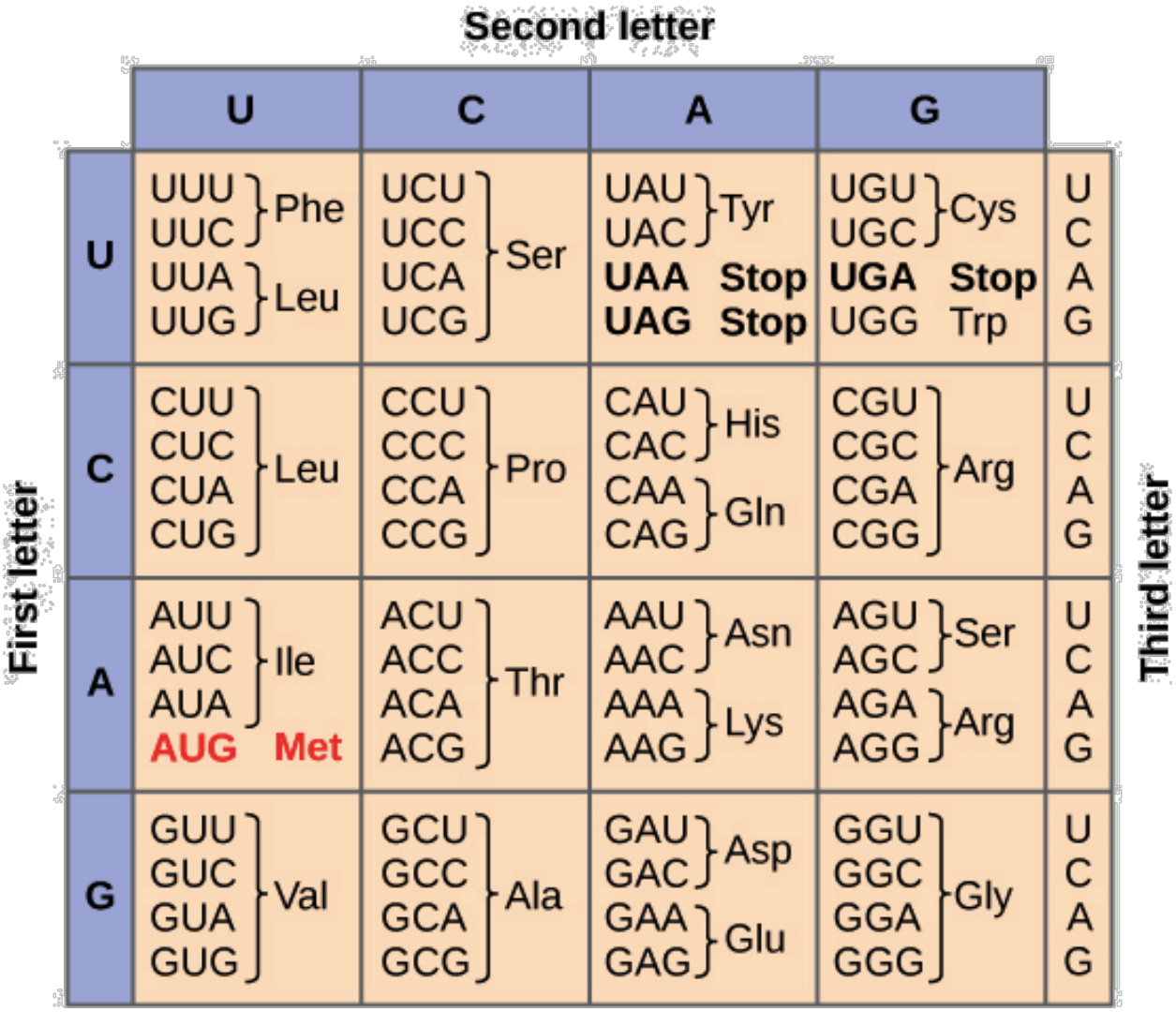
Considering the genetic code and all the possible open reading frames, answer the following questions:

1. How many stop codons are present in the sequence?
2. How many Glycines (Gly)?
3. Is Tryptophane (Trp) present?
4. What is the position of the leftmost Trp? Print the codon to double check correctness (hint: slicing).
5. What is the position of the rightmost Trp? Print the codon to double check correctness (hint: slicing).

Show/Hide Solution

```
In [24]: seq ="""CTGTCTCCCTCACTGTATGTAAATTGCATCTAGAATAGCA
         TCTGGAGCACTAATTGACACATAGTGGGTATCAATTATTA
         TTCCAGGTACTAGAGATACCTGGACCATTAACGGATAAAT
         AGAAGATTCATTTGTTGAGTGACTGAGGATGGCAGTTCCT
         GCTACCTTCAAGGATCTGGATGATGGGGAGAAACAGAGAA
         CATAGTGTGAGAATACTGTGGTAAGGAAAGTACAGAGGAC
         TGGTAGAGTGTCTAACCTAGATTTGGAGAAGGACCTAGAA
         GTCTATCCCAGGGAAATAAAAATCTAAGCTAAGGTTTGAG
         GAATCAGTAGGAATTGGCAAAGGAAGGACATGTTCCAGAT
         GATAGGAACAGGTTATGCAAAGATCCTGAAATGGTCAGAG
         CTTGGTGCTTTTTGAGAACCAAAAGTAGATTGTTATGGAC
         CAGTGCTACTCCCTGCCTCTTGCCAAGGGACCCCGCCAAG
```

[26] http://www.ensembl.org/Homo_sapiens/Gene/Summary?g=ENSG00000083093;r=16:23603160-23641310


Second letter
First letter
Third letter
U
C
A
G
UUU UUC Phe
UUA UUG Leu
UCU UCC UCA UCG Ser
UAU UAC Tyr
UAA Stop
UAG Stop
UGU UGC Cys
UGA Stop
UGG Trp
CUU CUC CUA CUG Leu
CCU CCC CCA CCG Pro
CAU CAC His
CAA CAG Gln
CGU CGC CGA CGG Arg
AUU AUC AUA Ile
AUG Met
ACU ACC ACA ACG Thr
AAU AAC Asn
AAA AAG Lys
AGU AGC Ser
AGA AGG Arg
GUU GUC GUA GUG Val
GCU GCC GCA GCG Ala
GAU GAC Asp
GAA GAG Glu
GGU GGC GGA GGG Gly

```
CACTGCATCCCTTCCCTCTGACTCCACCTTTCCACTTGCC
CAGTATTGTTGGTGT"""

seq = seq.replace("\n","")
mRNA = seq.replace("T","U")

print("Number of uracils: ", mRNA.count("U"))
print("Total length of the sequence: ", len(seq))
stopc = mRNA.count("UAA") + mRNA.count("UGA") + mRNA.count("UAG")
print("Number of stop codons: ", stopc)
gly = mRNA.count("GGU") + mRNA.count("GGC") + mRNA.count("GGA") + mRNA.count("GGG")
print("Number of glycines: ", gly)
print("Is Trp present? ", mRNA.find("UGG")> 0)
rmTrp = mRNA.find("UGG")
print("Leftmost Trp at pos:", rmTrp, " Codon: ", mRNA[rmTrp : rmTrp + 3])
lmTrp = mRNA.rfind("UGG")
print("Rightmost Trp at pos:", mRNA.rfind("UGG"), " Codon: ", mRNA[lmTrp:lmTrp+3])
```

```
Number of uracils:  140
Total length of the sequence:  535
Number of stop codons:  32
Number of glycines:  34
Is Trp present?  True
Leftmost Trp at pos: 42  Codon:  UGG
Rightmost Trp at pos: 529  Codon:  UGG
```

6. Consider the following Illumina HiSeq 4000 read: “”“AATGATACGGCGACCACCGAGATCTACACGCCTCCCTCGCGC CATCAGAGAGTCTGGGTCTCAGGTACCGCAGTTGTATCTTGCGCGACTATA ATCCACGGCTCTTATTCTAGCGTGCGCGTACGGCGGTGGGCGTCGTTACGCTATATT”“”.

Answer the following questions:

```
1. How long is the read (beware of newlines)?
2. What is the GC content of the read (remember $gc = \frac{G+C}{A+T+C+G}$)?
3. A Nextera adapter is "AATGATACGGCGACCACCGAGATCTACACGCCTCCCTCGCGCCATCAG".
Is it present in the read? How long is it?
4. Remove the Nextera adapter from the read and recompute the GC content.
Has GC content increased after adapter trimmming?
```

Show/Hide Solution

```
In [25]: read = """AATGATACGGCGACCACCGAGATCTACACGCCTCCCTCGCGC
         CATCAGAGAGTCTGGGTCTCAGGTACCGCAGTTGTATCTTGCGCGACTATA
         ATCCACGGCTCTTATTCTAGCGTGCGCGTACGGCGGTGGGCGTCGTTACGCTATATT"""

         read = read.replace("\n","")
         print("Read length is: ", len(read), " base pairs")
         g = read.count("G")
         c = read.count("C")
         t = read.count("T")
         a = read.count("A")

         gc = (g + c) / (a + t + c + g)

         print("GC content of read: ", gc)

         adapter = "AATGATACGGCGACCACCGAGATCTACACGCCTCCCTCGCGCCATCAG"

         print("Is the adapter present? ", adapter in read)
```

```
        print("Adapter length: ", len(adapter))
        print("The adapter starts at: ", read.find(adapter))

        trimmed_read = read.replace(adapter,"")

        tr_g = trimmed_read.count("G")
        tr_c = trimmed_read.count("C")
        tr_t = trimmed_read.count("T")
        tr_a = trimmed_read.count("A")

        tr_gc = (tr_g + tr_c) / (tr_a + tr_t + tr_c + tr_g)
        print("GC content of trimmed read: ", tr_gc)
        print("GC content has increased after trimming: ", tr_gc > gc)
```

```
Read length is:  150  base pairs
GC content of read:  0.56
Is the adapter present?  True
Adapter length:  48
The adapter starts at:  0
GC content of trimmed read:  0.5392156862745098
GC content has increased after trimming:  False
```

7. Given *geneA* starting at position 1000 and ending at position 3400, and *geneB* starting at position 3700 and ending at position 6000. Randomly select a position (*pos*) from 1 to 5202 and check the following: a. is pos in geneA? b. is pos in geneB? c. is pos in between the two genes? d. is pos within one of the two genes? e. is pos outside both genes? f. is pos within 100 bases before the start of geneA? To pick a random number you can import the random module and use the random.randint(start,end) function:

   ```
   import random

   pos = random.randint(1,6000)
   ```

Show/Hide Solution

```
In [26]: import random

        geneA_start = 1000
        geneA_end = 3400
        geneB_start = 3700
        geneB_end = 5201

        pos = random.randint(1,6000)
        print("Random position is: ", pos)

        answerA = (pos >= geneA_start and pos <= geneA_end)
        answerB = (pos >= geneB_start and pos <= geneB_end)
        answerC = (pos >geneA_end and pos <geneB_start)
        answerD = (answerA or answerB)
        answerE = (pos < geneA_start or (pos > geneA_end and pos < geneB_start) or (pos > geneB_end)
        answerF = (pos >= geneA_start - 100 ) and (pos < geneA_start)
        print("Is ", pos, " in geneA [", geneA_start, ",", geneA_end, "]? ", answerA)
        print("Is ", pos, " in geneB [", geneB_start, ",", geneB_end, "]? ", answerB)
        print("Is ", pos, " between the two genes? ", answerC)
        print("Is ", pos, " in one of the two genes? ", answerD)
        print("Is ", pos, " outside of both genes? ", answerE)
        print("Is ", pos, " within 100 bases from the start of geneA? ", answerF)
```

```
Random position is:  3729
Is  3729  in geneA [ 1000 , 3400 ]?  False
Is  3729  in geneB [ 3700 , 5201 ]?  True
Is  3729  between the two genes?  False
```

```
Is  3729  in one of the two genes?  True
Is  3729  outside of both genes?  False
Is  3729  within 100 bases from the start of geneA?  False
```

8. The DNA-binding domain of the Tumor Suppressor Protein TP53 can be represented by the string:

chain_a = “”“SSSVPSQKTYQGSYGFRLGFLHSGTAKSVTCTYSPALNKM FCQLAKTCPVQLWVD-STPPPGTRVRAMAIYKQSQHMTEVV RRCPHHERCSDSDGLAPPQHLIRVEGNLRVEYLDDRNTFR HSVVVPYEPPEVGSDCTTIHYNYMCNSSCMGGMNRRPILT IITLEDSSGNLLGRNSFEVRVCACPGRDR-RTEEENLRKKG EPHHELPPGSTKRALPNNT”“”

Answer the following questions:

```
1. How many lines is the sequence written on?
2. How long is the sequence (remove newlines)?
3. Create a new sequence with all new lines removed
4. How many cysteines "C" and histidines "H" are there in the sequence?
5. Does the chain contain the sub-sequence "NLRVEYLDDRN"? Where?
6. Extract the first line of the sequence (Hint: use find and string slicing).
```

Show/Hide Solution

```
In [27]: chain_a = """SSSVPSQKTYQGSYGFRLGFLHSGTAKSVTCTYSPALNKM
         FCQLAKTCPVQLWVDSTPPPGTRVRAMAIYKQSQHMTEVV
         RRCPHHERCSDSDGLAPPQHLIRVEGNLRVEYLDDRNTFR
         HSVVVPYEPPEVGSDCTTIHYNYMCNSSCMGGMNRRPILT
         IITLEDSSGNLLGRNSFEVRVCACPGRDRRTEEENLRKKG
         EPHHELPPGSTKRALPNNT"""

         print(chain_a)
         lines = chain_a.count('\n') + 1
         print("The sequence is in ", lines, " lines")

         sequence = chain_a.replace("\n","")
         print("The sequence has ", len(sequence), " aminoacids")
         print("The sequence counts ", sequence.count('C'), " cysteins")
         print("The sequence counts ", sequence.count('H'), " histidines")
         subseq = "NLRVEYLDDRN"
         print("Does the sequence contain ", subseq, "?", subseq in sequence )
         pos = sequence.find(subseq)
         getS = sequence[pos:pos+len(subseq)]
         print(subseq, " is present at pos: ", pos , "[check:", getS , "]")

         end_first_line = chain_a.find('\n')
         print("The first line is: ", chain_a[0:end_first_line])
```

```
SSSVPSQKTYQGSYGFRLGFLHSGTAKSVTCTYSPALNKM
FCQLAKTCPVQLWVDSTPPPGTRVRAMAIYKQSQHMTEVV
RRCPHHERCSDSDGLAPPQHLIRVEGNLRVEYLDDRNTFR
HSVVVPYEPPEVGSDCTTIHYNYMCNSSCMGGMNRRPILT
IITLEDSSGNLLGRNSFEVRVCACPGRDRRTEEENLRKKG
EPHHELPPGSTKRALPNNT
The sequence is in  6  lines
The sequence has  219  aminoacids
The sequence counts  10  cysteins
The sequence counts  9  histidines
Does the sequence contain  NLRVEYLDDRN ? True
NLRVEYLDDRN  is present at pos:  106 [check: NLRVEYLDDRN ]
The first line is:  SSSVPSQKTYQGSYGFRLGFLHSGTAKSVTCTYSPALNKM
```

9. Calculate the zeros of the equation $ax^2 - b = 0$ where a = 10 and b = 1. Hint: use math.sqrt or ** 0.5. Finally check that substituting the obtained value of x in the equation gives zero.

Show/Hide Solution

```
In [28]: import math

         A = 10
         B = 1

         X = math.sqrt(B/A)

         print("10X**2 - 1 = 0 for X:", X)
         print(10*X**2 -1 == 0)
```

```
10X**2 - 1 = 0 for X: 0.31622776601683794
True
```

CHAPTER 4

Practical 3

In this practical we will work with lists and tuples.

4.1 Slides

The slides of the introduction can be found here: Intro

4.2 Lists

Python lists are **ordered** collections of (homogeneous) objects, but they can hold also non-homogeneous data. List are **mutable objects**. Elements of the collection are specified within two square brackets **[]** and are comma separated.

We can use the function print to print the content of lists. Some examples of list definitions follow:

```
In [1]: my_first_list = [1,2,3]
        print("first:" , my_first_list)

        my_second_list = [1,2,3,1,3] #elements can appear several times
        print("second: ", my_second_list)

        fruits = ["apple", "pear", "peach", "strawberry", "cherry"] #elements can be strings
        print("fruits:", fruits)

        an_empty_list = []
        print("empty:" , an_empty_list)

        another_empty_list = list()
        print("another empty:", another_empty_list)

        a_list_containing_other_lists = [[1,2], [3,4,5,6]] #elements can be other lists
        print("list of lists:", a_list_containing_other_lists)
```

```
        my_final_example = [my_first_list, a_list_containing_other_lists]
        print("a list of lists of lists:", my_final_example)
```

```
first: [1, 2, 3]
second:  [1, 2, 3, 1, 3]
fruits: ['apple', 'pear', 'peach', 'strawberry', 'cherry']
empty: []
another empty: []
list of lists: [[1, 2], [3, 4, 5, 6]]
a list of lists of lists: [[1, 2, 3], [[1, 2], [3, 4, 5, 6]]]
```

4.2.1 Operators for lists

Python provides several operators to handle lists. The following behave like on strings (**remember that as in strings, the first position is 0!**):

Result	Operator	Meaning
bool	=, !=	Check if two lists are equal or different
int	len(list)	Return the length of the list
list	list + list	Concatenate two lists (returns a new list)
list	list * int	Replicate the list (returns a new list)
list	list[int:int]	Extract a sub-list

While this requires that the whole tested obj is present in the list

Result	Operator	Meaning
bool	obj in list	Check if an element is present in a list

and

Result	Operator	Meaning
obj	list[int]	Read/write an element at a specified index

can also change the corresponding value of the list (**lists are mutable objects**).

Some examples follow.

```
In [2]: A = [1, 2, 3 ]
        B = [1, 2, 3, 1, 2]

        print("A is a ", type(A))

        print(A, " has length: ", len(A))
        print("A[0]: ", A[0], " A[1]:", A[1], " A[-1]:", A[-1])
```

```
        print(B, " has length: ", len(B))
        print("Is A equal to B?", A == B)

        C = A + [1, 2]
        print(C)
        print("Is C equal to B?", B == C)
        D = [1, 2, 3]*8
        print(D)

        E = D[12:18] #slicing
        print(E)
        print("Is A*2 equal to E?", A*2 == E)
```

```
A is a  <class 'list'>
[1, 2, 3]  has length:  3
A[0]:  1  A[1]: 2  A[-1]: 3
[1, 2, 3, 1, 2]  has length:  5
Is A equal to B? False
[1, 2, 3, 1, 2]
Is C equal to B? True
[1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3]
[1, 2, 3, 1, 2, 3]
Is A*2 equal to E? True
```

```
In [3]: A = [1, 2, 3, 4, 5, 6]
        B = [1, 3, 5]
        print("A:", A)
        print("B:", B)

        print("Is B in A?", B in A)
        print("A\'s ID:", id(A))
        A[5] = [1,3,5] #we can add elements
        print(A)
        print("A\'s ID:", id(A))
        print("A has length:", len(A))
        print("Is now B in A?", B in A)
```

```
A: [1, 2, 3, 4, 5, 6]
B: [1, 3, 5]
Is B in A? False
A's ID: 140546986337864
[1, 2, 3, 4, 5, [1, 3, 5]]
A's ID: 140546986337864
A has length: 6
Is now B in A? True
```

Note: When slicing do not exceed the list boundaries (or you will be prompted a `list index out of range` error).

Consider the following example:

```
In [4]: A = [1, 2, 3, 4, 5, 6]
        print("A has length:", len(A))

        print("First element:", A[0])
        print("7th-element: ", A[6])
```

```
A has length: 6
```

```
First element: 1

---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-4-699e5f04cae0> in <module>()
      3
      4 print("First element:", A[0])
----> 5 print("7th-element: ", A[6])

IndexError: list index out of range
```

Example: Consider the matrix $M = \begin{bmatrix} 1 & 2 & 3 \\ 1 & 2 & 1 \\ 1 & 1 & 3 \end{bmatrix}$ and the vector $v = [10, 5, 10]^T$. What is the matrix-vector product $M * v$?

$$\begin{bmatrix} 1 & 2 & 3 \\ 1 & 2 & 1 \\ 1 & 1 & 3 \end{bmatrix} * [10, 5, 10]^T = [50, 30, 45]^T$$

```
In [5]: M = [[1, 2, 3], [1, 2, 1], [1, 1, 3]]
        v = [10, 5, 10]
        prod = [0, 0 ,0] #at the beginning the product is the null vector

        prod[0]=M[0][0]*v[0] + M[0][1]*v[1] + M[0][2]*v[2]
        prod[1]=M[1][0]*v[0] + M[1][1]*v[1] + M[1][2]*v[2]
        prod[2]=M[2][0]*v[0] + M[2][1]*v[1] + M[2][2]*v[2]

        print("M: ", M)
        print("v: ", v)
        print("M*v: ", prod)


M:  [[1, 2, 3], [1, 2, 1], [1, 1, 3]]
v:  [10, 5, 10]
M*v:  [50, 30, 45]
```

4.2.2 Methods of the class list

The class list has some methods to operate on it. Recall from the lecture the following methods:

Note: Lists are **mutable objects** and therefore virtually all the previous methods (except *count*) do not have an output value, but they **modify** the list.

Some usage examples follow:

```
In [6]: #A numeric list
        A = [1, 2, 3]
        print(A)
        print("A has id:", id(A))
        A.append(72) #appends one and only one object
        print(A)
        print("A has id:", id(A))
        A.extend([1, 5, 124, 99]) #adds all these objects, one after the other.
        print(A)
        A.reverse()
        print(A)
```

Return	Method	Meaning
None	list.append(obj)	Add a new element at the end of the list
None	list.extend(list)	Add several new elements at the end of the list
None	list.insert(int,obj)	Add a new element at some given position
None	list.remove(obj)	Remove the first occurrence of an element
None	list.reverse()	Invert the order of the elements
None	list.sort()	Sort the elements
int	list.count(obj)	Count the occurrences of an element

```
A.sort()
print(A)
print("Min value: ", A[0]) # In this simple case, could have used min(A)
print("Max value: ", A[-1]) #In this simple case, could have used max(A)
print("Number 1 appears:", A.count(1), " times")
print("While number 837: ", A.count(837))

print("\nDone with numbers, let's go strings...\n")
#A string list
fruits = ["apple", "banana", "pineapple", "cherry","pear", "almond", "orange"]
#Let's get a reverse lexicographic order:
print(fruits)
fruits.sort()
fruits.reverse()
print(fruits)
fruits.remove("banana")
print(fruits)
fruits.insert(5, "wild apple") #put wild apple after apple.
print(fruits)
print("\nSorted fruits:")
fruits.sort() # does not return anything. Modifies list!
print(fruits)
```

```
[1, 2, 3]
A has id: 140547045876232
[1, 2, 3, 72]
A has id: 140547045876232
[1, 2, 3, 72, 1, 5, 124, 99]
[99, 124, 5, 1, 72, 3, 2, 1]
[1, 1, 2, 3, 5, 72, 99, 124]
Min value:  1
Max value:  124
Number 1 appears: 2  times
While number 837:  0
```

```
Done with numbers, let's go strings...

['apple', 'banana', 'pineapple', 'cherry', 'pear', 'almond', 'orange']
['pineapple', 'pear', 'orange', 'cherry', 'banana', 'apple', 'almond']
['pineapple', 'pear', 'orange', 'cherry', 'apple', 'almond']
['pineapple', 'pear', 'orange', 'cherry', 'apple', 'wild apple', 'almond']

Sorted fruits:
['almond', 'apple', 'cherry', 'orange', 'pear', 'pineapple', 'wild apple']
```

Some things to remember

1. append and extend work quite differently:

```
In [7]: A = [1, 2, 3]

        A.extend([4, 5])
        print(A)
        B = [1, 2, 3]
        B.append([4,5])
        print(B)
```

```
[1, 2, 3, 4, 5]
[1, 2, 3, [4, 5]]
```

2. To remove an object it must exist:

```
In [8]: A = [1,2,3, [[4],[5,6]], 8]
        print(A)
        A.remove(2)
        print(A)
        A.remove([[4],[5,6]])
        print(A)
        A.remove(7)
```

```
[1, 2, 3, [[4], [5, 6]], 8]
[1, 3, [[4], [5, 6]], 8]
[1, 3, 8]

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-8-cef337401de9> in <module>()
      5 A.remove([[4],[5,6]])
      6 print(A)
----> 7 A.remove(7)

ValueError: list.remove(x): x not in list
```

3. To sort a list, its elements must be sortable (i.e. homogeneous)!

```
In [1]: A = [4,3, 1,7, 2]
        print(A)
        A.sort()
        print(A)
        A.append("banana")
        print(A)
        A.sort()
        print(A)
```

```
[4, 3, 1, 7, 2]
```

```
[1, 2, 3, 4, 7]
[1, 2, 3, 4, 7, 'banana']

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-1-91b77adb823f> in <module>
      5 A.append("banana")
      6 print(A)
----> 7 A.sort()
      8 print(A)

TypeError: '<' not supported between instances of 'str' and 'int'
```

Important to remember:

Lists are **mutable objects** and this has some consequences! Since lists are mutable objects, they hold references to objects rather than objects.

Take a look at the following examples:

```
In [10]: l1 = [1, 2]
         l2 = [4, 3]
         LL = [l1, l2]
         print("l1:", l1)
         print("l2:",l2)
         print("LL:", LL)
         l1.append(7)
         print("\nAppending 7 to l1...")
         print("l1:", l1)
         print("LL now: ", LL)

         LL[0][1] = -1
         print("\nSetting LL[0][1]=-1...")
         print("LL now:" , LL)
         print("l1 now", l1)
         #but the list can point also to a different object,
         #without affecting the original list.
         LL[0] = 100
         print("\nSetting LL[0] = 100")
         print("LL now:", LL)
         print("l1 now", l1)


l1: [1, 2]
l2: [4, 3]
LL: [[1, 2], [4, 3]]

Appending 7 to l1...
l1: [1, 2, 7]
LL now:  [[1, 2, 7], [4, 3]]

Setting LL[0][1]=-1...
LL now: [[1, -1, 7], [4, 3]]
l1 now [1, -1, 7]

Setting LL[0] = 100
LL now: [100, [4, 3]]
l1 now [1, -1, 7]
```

Making copies

```
In [11]: A = ["hi", "there"]
         B = A
         print("A:", A)
         print("B:", B)
         A.extend(["from", "python"])
         print("A now: ", A)
         print("B now: ", B)

         print("\n---- copy example -------")
         #Let's make a distinct copy of A.
         C = A[:] #all the elements of A have been copied in C
         print("C:", C)
         A[3] = "java"
         print("A now:", A)
         print("C now:", C)

         print("\n---- be careful though -------")
         #Watch out though that...
         D = [A, A]
         E = D[:]
         print("D:", D)
         print("E:", E)

         D[0][0] = "hello"
         print("\nD now:", D)
         print("E now:", E)
         print("A now:", A)
```

```
A: ['hi', 'there']
B: ['hi', 'there']
A now:  ['hi', 'there', 'from', 'python']
B now:  ['hi', 'there', 'from', 'python']

---- copy example -------
C: ['hi', 'there', 'from', 'python']
A now: ['hi', 'there', 'from', 'java']
C now: ['hi', 'there', 'from', 'python']

---- be careful though -------
D: [['hi', 'there', 'from', 'java'], ['hi', 'there', 'from', 'java']]
E: [['hi', 'there', 'from', 'java'], ['hi', 'there', 'from', 'java']]

D now: [['hello', 'there', 'from', 'java'], ['hello', 'there', 'from', 'java']]
E now: [['hello', 'there', 'from', 'java'], ['hello', 'there', 'from', 'java']]
A now: ['hello', 'there', 'from', 'java']
```

Equality and identity

```
In [12]: A = [1, 2, 3]
         B = A
         C = [1, 2, 3]
         print("Is A equal to B?", A == B)
         print("Is A actually B?", A is B)
         print("Is A equal to C?", A == C)
         print("Is A actually C?", A is C)
         #in fact:
         print("\nA's id:", id(A))
         print("B's id:", id(B))
```

```
        print("C's id:", id(C))
        #just to confirm that:
        A.append(4)
        B.append(5)
        print("\nA now: ", A)
        print("B now: ", A)
```

```
Is A equal to B? True
Is A actually B? True
Is A equal to C? True
Is A actually C? False

A's id: 140546986339208
B's id: 140546986339208
C's id: 140546977413576

A now:  [1, 2, 3, 4, 5]
B now:  [1, 2, 3, 4, 5]
```

4.2.3 From strings to lists, the `split` method

Strings have a method *split* that can literally split the string at specific characters.

Example Recall the protein seen in the previous practical:

chain_a = “”“SSSVPSQKTYQGSYGFRLGFLHSGTAKSVTCTYSPALNKM FCQLAKTCPVQLWVD-STPPPGTRVRAMAIYKQSQHMTEVV RRCPHHERCSDSDGLAPPQHLIRVEGNLRVEYLDDRNTFR HSVVVPYEPPEVGSDCTTIHYNYMCNSSCMGGMNRRPILT IITLEDSSGNLLGRNSFEVRVCACPGRDR-RTEEENLRKKG EPHHELPPGSTKRALPNNT”“”

how can we split it into several lines?

```
In [9]: chain_a = """SSSVPSQKTYQGSYGFRLGFLHSGTAKSVTCTYSPALNKM
        FCQLAKTCPVQLWVDSTPPPGTRVRAMAIYKQSQHMTEVV
        RRCPHHERCSDSDGLAPPQHLIRVEGNLRVEYLDDRNTFR
        HSVVVPYEPPEVGSDCTTIHYNYMCNSSCMGGMNRRPILT
        IITLEDSSGNLLGRNSFEVRVCACPGRDRRTEEENLRKKG
        EPHHELPPGSTKRALPNNT"""

        lines = chain_a.split('\n')
        print("Original sequence:")
        print( chain_a, "\n") #some spacing to keep things clear
        print("line by line:")
        print("1st line:" ,lines[0])
        print("2nd line:" ,lines[1])
        print("3rd line:" ,lines[2])
        print("4th line:" ,lines[3])
        print("5th line:" ,lines[4])
        print("6th line:" ,lines[5])

        print("\nSplit the 1st line in correspondence to FRL:\n",lines[0].split("FRL"))
```

```
Original sequence:
SSSVPSQKTYQGSYGFRLGFLHSGTAKSVTCTYSPALNKM
FCQLAKTCPVQLWVDSTPPPGTRVRAMAIYKQSQHMTEVV
RRCPHHERCSDSDGLAPPQHLIRVEGNLRVEYLDDRNTFR
HSVVVPYEPPEVGSDCTTIHYNYMCNSSCMGGMNRRPILT
IITLEDSSGNLLGRNSFEVRVCACPGRDRRTEEENLRKKG
EPHHELPPGSTKRALPNNT
```

```
line by line:
1st line: SSSVPSQKTYQGSYGFRLGFLHSGTAKSVTCTYSPALNKM
2nd line: FCQLAKTCPVQLWVDSTPPPGTRVRAMAIYKQSQHMTEVV
3rd line: RRCPHHERCSDSDGLAPPQHLIRVEGNLRVEYLDDRNTFR
4th line: HSVVVPYEPPEVGSDCTTIHYNYMCNSSCMGGMNRRPILT
5th line: IITLEDSSGNLLGRNSFEVRVCACPGRDRRTEEENLRKKG
6th line: EPHHELPPGSTKRALPNNT

Split the 1st line in correspondence to FRL:
 ['SSSVPSQKTYQGSYG', 'GFLHSGTAKSVTCTYSPALNKM']
```

Note that in the last instruction, the substring FRL is disappeared (as happened to the newline).

4.2.4 And back to strings with the `join` method

Given a list, one can join the elements of the list together into a string by using the `join` method of the class string. The syntax is the following: *str.join(list)* which joins together all the elements in the list in a string separating them with the string *str*.

Example Given the list ['Oct', '5', '2018', '15:30'], let's combine all its elements in a string joining the elements with a dash ("-") and print them. Let's finally join them with a tab ("\t") and print them.

```
In [3]: vals = ['Oct', '5th', '2018', '15:30']
        print(vals)
        myStr = "-".join(vals)
        print("\n" + myStr)
        myStr = "\t".join(vals)
        print("\n" + myStr)
```

```
['Oct', '5th', '2018', '15:30']

Oct-5th-2018-15:30

Oct     5th     2018    15:30
```

4.3 Tuples

Tuples are the **immutable** version of lists (i.e. it is not possible to change their content without actually changing the object). They are sequential collections of objects, and elements of tuples are assumed to be in a particular order. They can hold heterogeneous information. They are defined with the brackets (). Some examples:

```
In [14]: first_tuple = (1,2,3)
         print(first_tuple)

         second_tuple = (1,) #this contains one element only, but we need the comma!
         var = (1) #This is not a tuple!!!
         print(second_tuple, " type:", type(second_tuple))
         print(var, " type:", type(var))
         empty_tuple = () #fairly useless
         print(empty_tuple, "\n")
         third_tuple = ("January", 1 ,2007) #heterogeneous info
         print(third_tuple)

         days = (third_tuple,("February",2,1998), ("March",2,1978),("June",12,1978))
         print(days, "\n")
```

```
        #Remember tuples are immutable objects...
        print("Days has id: ", id(days))
        days = ("Mon","Tue","Wed","Thu","Fri","Sat","Sun")
        #...hence reassignment creates a new object
        print("Days now has id: ", id(days))
```

```
(1, 2, 3)
(1,)  type: <class 'tuple'>
1  type: <class 'int'>
()

('January', 1, 2007)
(('January', 1, 2007), ('February', 2, 1998), ('March', 2, 1978), ('June', 12, 1978))

Days has id:  140546977398040
Days now has id:  140546995078232
```

The following operators work on tuples and they behave exactly as on lists:

Result	Operator	Meaning
bool	=, !=	Check if two tuples are equal or different
int	len(tuple)	Return the length of the tuple
tuple	tuple + tuple	Concatenate two tuples (returns a new tuple)
tuple	tuple * int	Replicate the tuple (returns a tuple)
tuple	tuple[int]	Read an element of the tuple
tuple	tuple[int:int]	Extract a sub-tuple

```
In [15]: practical1 = ("Friday", "28/09/2018")
         practical2 = ("Tuesday", "02/10/2018")
         practical3 = ("Friday", "05/10/2018")

         #A tuple containing 3 tuples
         lectures = (practical1, practical2, practical3)
         #One tuple only
         mergedLectures = practical1 + practical2 + practical3

         print("The first three lectures:\n", lectures, "\n")
         print("mergedLectures:\n", mergedLectures)

         #This returns the whole tuple
         print("1st lecture was on: ", lectures[0], "\n")
         #2 elements from the same tuple
         print("1st lecture was on ", mergedLectures[0], ", ", mergedLectures[1], "\n")
         # Return type is tuple!
         print("3rd lecture was on: ", lectures[2])
         #2 elements from the same tuple returned in tuple
         print("3rd lecture was on ", mergedLectures[4:], "\n")
```

```
The first three lectures:
 (('Friday', '28/09/2018'), ('Tuesday', '02/10/2018'), ('Friday', '05/10/2018'))
```

```
mergedLectures:
 ('Friday', '28/09/2018', 'Tuesday', '02/10/2018', 'Friday', '05/10/2018')
1st lecture was on:  ('Friday', '28/09/2018')

1st lecture was on  Friday ,  28/09/2018

3rd lecture was on:  ('Friday', '05/10/2018')
3rd lecture was on  ('Friday', '05/10/2018')
```

The following methods are available for tuples:

Return	Method	Meaning
int	list.count(obj)	Count the occurrences of an element
int	list.index(obj)	Return the index of the first occurrence of an object

```
In [16]: practical1 = ("Friday", "28/09/2018")
         practical2 = ("Tuesday", "02/10/2018")
         practical3 = ("Friday", "05/10/2018")


         mergedLectures = practical1 + practical2 + practical3  #One tuple only
         print(mergedLectures.count("Friday"), " lectures were on Friday")
         print(mergedLectures.count("Tuesday"), " lecture was on Tuesday")

         print("Index:", practical2.index("Tuesday"))
         print("Index:", practical2.index("Wednesday"))


2  lectures were on Friday
1  lecture was on Tuesday
Index: 0

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-16-fc543d476575> in <module>()
      9
     10 print("Index:", practical2.index("Tuesday"))
---> 11 print("Index:", practical2.index("Wednesday"))

ValueError: tuple.index(x): x not in tuple
```

4.4 Exercises

1. Given the following text string:

```
"""this is a text
string on
several lines that does not say anything."""
```

1. print it; b) print how many lines, words and characters it contains. Finally, c)sort the words alphabetically and print the first and the last in lexicographic order.

Show/Hide Solution

```
In [14]: T = """this is a text
         string on
         several lines that does not say anything."""

         #a) print it
         print(T)
         print("")
         #b) print the lines, words and characters
         lines = T.split('\n')

         #NOTE: words are split by a space or a newline!

         words = lines[0].split(' ') + lines[1].split(' ') + lines[2].split(' ')

         num_chars = len(T)
         print("Lines:", len(lines), "words:", len(words), "chars:", num_chars)
         # alternative way for number of characters:
         print("")
         characters = list(T)
         num_chars2 = len(characters)
         print(characters)
         print(num_chars2)

         words.sort(reverse = True) #NOTE: it does not return ANYTHING!!!
         print(words)
         print("")
         print("First word: ", words[-1])
         print("Last word:", words[0])


this is a text
string on
several lines that does not say anything.

Lines: 3 words: 13 chars: 66

['t', 'h', 'i', 's', ' ', 'i', 's', ' ', 'a', ' ', 't', 'e', 'x', 't', '\n', 's', 't', 'r', 'i', 'n',
66
['this', 'that', 'text', 'string', 'several', 'say', 'on', 'not', 'lines', 'is', 'does', 'anything.',

First word:  a
Last word: this
```

2. The variant calling format (VCF[27]) is a format to represent structural variants of genomes (i.e. SNPs, insertions, deletions) . Each line of this format represents a variant, every piece of information within a line is separated by a tab (\t in python). The first 5 fields of this format report the chromosome (chr), the position (pos), the name of the variant (name), the reference allele (REF) and the alternative allele (ALT). Assuming to have a variable VCF defined containing the following three lines (representing three SNPs):

VCF = “”“MDC000001.124\t7112\tFB_AFFY_0000024\tG\tA MDC000002.328\t941\tFB_AFFY_0000144\tC\tT MDC000004.272\t2015\tFB_AFFY_0000222\tG\tA”“”

[27] http://www.internationalgenome.org/wiki/Analysis/vcf4.0/

```
1. Store these three variants as a list of lists, where each one of the fields is␣
↪kept separate
(e.g. the list should be similar to:
[[chr1,pos1,name1,ref1,alt1], [chr2, pos2, name2, ref2, alt2], ...] where all the␣
↪elements
are as specified in the string VCF (note that "..." means that the list is not␣
↪complete).

2. Print each variant changing its format in:  "name|chr|pos|REF/ALT".
```

Show/Hide Solution

```
In [4]: VCF="""MDC000001.124\t7112\tFB_AFFY_0000024\tG\tA
        MDC000002.328\t941\tFB_AFFY_0000144\tC\tT
        MDC000004.272\t2015\tFB_AFFY_0000222\tG\tA"""

        variants = VCF.split('\n')

        variants[0] = variants[0].split('\t')
        variants[1] = variants[1].split('\t')
        variants[2] = variants[2].split('\t')

        print(variants)

        print(variants[0])
        print(variants[1])
        print(variants[2], "\n")

        info = variants[0]
        print(info[2] + "|" + info[0] + "|" + info[1] + "|" + info[3] +"/" + info[4])
        info = variants[1]
        print(info[2] + "|" + info[0] + "|" + info[1] + "|" + info[3] +"/" + info[4])
        info = variants[2]
        print(info[2] + "|" + info[0] + "|" + info[1] + "|" + info[3] +"/" + info[4])
```

```
[['MDC000001.124', '7112', 'FB_AFFY_0000024', 'G', 'A'], ['MDC000002.328', '941', 'FB_AFFY_0000144',
['MDC000001.124', '7112', 'FB_AFFY_0000024', 'G', 'A']
['MDC000002.328', '941', 'FB_AFFY_0000144', 'C', 'T']
['MDC000004.272', '2015', 'FB_AFFY_0000222', 'G', 'A']

FB_AFFY_0000024|MDC000001.124|7112|G/A
FB_AFFY_0000144|MDC000002.328|941|C/T
FB_AFFY_0000222|MDC000004.272|2015|G/A
FB_AFFY_0000024|MDC000001.124|7112|G/A
FB_AFFY_0000144|MDC000002.328|941|C/T
FB_AFFY_0000222|MDC000004.272|2015|G/A
```

3. Given the list L = ["walnut", "eggplant", "lemon", "lime", "date", "onion", "nectarine", "endive"]:

```
1. Create another list (called newList) containing the first letter of each␣
↪element of L
(e.g newList =["w", "e", ...]).
2. Add a space to newList at position 4 and append an exclamation mark (!) at the␣
↪end.
3. Print the list.
4. Print the content of the list joining all the elements with an empty space
(i.e. use the method join: "".join(newList) )
```

Show/Hide Solution

```
In [18]: L = ["walnut", "eggplant", "lemon", "lime", "date", "onion", "nectarine", "endive" ]

         newList = []
         newList.append(L[0][0])
         newList.append(L[1][0])
         newList.append(L[2][0])
         newList.append(L[3][0])
         newList.append(L[4][0])
         newList.append(L[5][0])
         newList.append(L[6][0])
         newList.append(L[7][0])

         newList.insert(4," ")
         newList.append("!")

         print(newList)
         print("\n", "".join(newList))
['w', 'e', 'l', 'l', ' ', 'd', 'o', 'n', 'e', '!']

 well done!
```

4. Fastq is a standard format for storing sequences and quality information. More information on the format can be found here[28]. This format can be used to store sequencing information coming from the sequencer. Each entry represents a read (i.e. a sequence) and carries four different pieces of information. A sample entry is the following:

@HWI-ST1296:75:C3F7CACXX:1:1101:19142:14904 CCAACAACTTTGACGCTAAGGATAGCTCCATG-GCAGCATATCTGGCACAA + FHIIJIJJGIJJJJJIHHHFFFFFEE:;CIDDDDDDDDDDDDEDDDDDDB”“”

where:

- 1. the first line is the read identifier starts with a “@” and carries several types of information regarding the instrument used for sequencing (the reported example is an illumina read - if you are interested, you can find more info on the format of the ID here[29].
- (ii). the second information is the sequence of the read (note that it can span several lines, not in our simple example though);
- 3. a “+” sign to mark the end of the sequence information (optionally the read identifier can be repeated);
- 4. the phred quality score of the read encoded as a text string. It must contain the same number of elements that are in the sequence. To decode each character into a the corresponding phred score, one needs to convert it into the unicode integer representation of it - 33. For example, the conversion of a character “I” in python can be done using the *ord* built in function in the following way: ord(“I”) - 33 = 40. Finally the phred score can be converted into probability of the base to be wrong with the following formula: $P = 10^{-Q/10}$, where Q is the phred quality score.

Given the entry above:

```
1. Check that the ID starts with a @
2. Store the sequence as a string and as list where each element is one single base
(e.g. sequence =['T', 'A',...])
3. Store the quality as a list where each element is one single quality character
(e.g. qualChar = ['C', 'C', ...])
4. Check that the length of the sequence and quality are the same
(hint: use the sequence string, not the list)
5. Count how many times the sequence "TCCA" appears in the read
```

(continues on next page)

[28] https://en.wikipedia.org/wiki/FASTQ_format
[29] https://support.illumina.com/help/BaseSpace_OLH_009008/Content/Source/Informatics/BS/NamingConvention_FASTQ-files-swBS.htm

(continued from previous page)

```
6. Retrieve the sub-list containing the quality values corresponding to the "TCCA"␣
↪string
7. Convert each value in the list at point 6 in the corresponding probability of the␣
↪base
being wrong
```

Show/Hide Solution

```
In [19]: entry = """@HWI-ST1296:75:C3F7CACXX:1:1101:19142:14904
         CCAACAACTTTGACGCTAAGGATAGCTCCATGGCAGCATATCTGGCACAA
         +
         FHIIJIJJGIJJJJJIHHHFFFFFEE:;CIDDDDDDDDDDDDEDDDDDDB
         """

         print("Does header start with @?", entry.startswith("@"))
         entryList = entry.split('\n')

         sequence = list(entryList[1])
         qualChar = list(entryList[3])


         print("Sequence and quality have the same length:", len(sequence) == len(qualChar))
         print("Sequence \"ACAA\" is present ", entryList[1].count("TCCA"), " times" )

         pos = entryList[1].find("TCCA")

         qVals = qualChar[pos:pos+4]
         print("Sequence: ", entryList[1][pos:pos+4])
         print("Quality: ", qVals)

         phredS = []

         phredS.append(ord(qVals[0]) - 33)
         phredS.append(ord(qVals[1]) - 33)
         phredS.append(ord(qVals[2]) - 33)
         phredS.append(ord(qVals[3]) - 33)

         print("Phred score:", phredS)

         probVals = []
         probVals.append(10**(0-phredS[0]/10))
         probVals.append(10**(0-phredS[1]/10))
         probVals.append(10**(0-phredS[2]/10))
         probVals.append(10**(0-phredS[3]/10))

         print("Error Probabilities:\n", probVals)


Does header start with @? True
Sequence and quality have the same length: True
Sequence "ACAA" is present  1  times
Sequence:  TCCA
Quality:  [':', ';', 'C', 'I']
Phred score: [25, 26, 34, 40]
Error Probabilities:
 [0.0031622776601683794, 0.0025118864315095794, 0.00039810717055349735, 0.0001]
```

5. Given the list L = [10, 60, 72, 118, 11, 71, 56, 89, 120, 175] find the min, max and median value (hint: sort it

and extract the right values). Create a list with only the elements at even indexes (i.e. [10, 72, 11, ..], note that the ".." means that the list is not complete) and re-compute min, max and median values. Finally, re-do the same for the elements located at odd indexes (i.e. [60, 118,..]).

Show/Hide Solution

```
In [20]: L = [10, 60, 72, 118, 11, 71, 56, 89, 120, 175]
         Lodd = L[1::2] #get only odd-indexed elements
         Leven = L[0::2] #get only even-indexed elements

         print("original:" , L)
         print("Lodd:", Lodd)
         print("Leven:", Leven)
         L.sort()
         Lodd.sort()
         Leven.sort()


         print("sorted:  " , L)
         print("sorted Lodd:  " , Lodd)
         print("sorted Leven:  " , Leven)

         print("L: Min: ", L[0], " Max." , L[-1], " Median: ", L[len(L) // 2])
         print("Lodd: Min: ", Lodd[0], " Max." , Lodd[-1], " Median: ", Lodd[len(Lodd) // 2])
         print("Leven: Min: ", Leven[0], " Max." , Leven[-1], " Median: ", Leven[len(Leven) // 2])
```

```
original: [10, 60, 72, 118, 11, 71, 56, 89, 120, 175]
Lodd: [60, 118, 71, 89, 175]
Leven: [10, 72, 11, 56, 120]
sorted:   [10, 11, 56, 60, 71, 72, 89, 118, 120, 175]
sorted Lodd:   [60, 71, 89, 118, 175]
sorted Leven:   [10, 11, 56, 72, 120]
L: Min:  10  Max. 175  Median:  72
Lodd: Min:  60  Max. 175  Median:  89
Leven: Min:  10  Max. 120  Median:  56
```

6. Given the string pets = "siamese cat,dog,songbird,guinea pig,rabbit,hampster" convert it into a list. Create then a tuple of tuples where each tuple has two information: the name of the pet and the length of the name. E.g. (("dog",3), ("hampster",8)). Print the tuple.

Show/Hide Solution

```
In [21]: pets = "cat,dog,bird,guinea pig,rabbit,hampster"

         pet_list = pets.split(',')

         print(pet_list)

         pet_tuples = ((pet_list[0], len(pet_list[0])),
                       (pet_list[1], len(pet_list[1])),
                       (pet_list[2], len(pet_list[2])),
                       (pet_list[3], len(pet_list[3])),
                       (pet_list[4], len(pet_list[4])),
                       (pet_list[5], len(pet_list[5])))

         print(pet_tuples)
```

```
['cat', 'dog', 'bird', 'guinea pig', 'rabbit', 'hampster']
(('cat', 3), ('dog', 3), ('bird', 4), ('guinea pig', 10), ('rabbit', 6), ('hampster', 8))
```

7. Given the string S="apple|pear|apple|cherry|pear|apple|pear|pear|cherry|pear|strawberry". Store the elements separated by the "|" in a list.

(a) How many elements does the list have?

(b) Knowing that the list created at the previous point has only four distinct elements (i.e. "apple","pear","cherry" and "strawberry"), create another list where each element is a tuple containing the name of the fruit and its multiplicity (that is how many times it appears in the original list). Ex. list_of_tuples = [("apple", 3), ("pear", "5"),...]

(c) Print the content of each tuple in a separate line (ex. first line: apple is present 3 times)

Show/Hide Solution

```
In [22]: S="apple|pear|apple|cherry|pear|apple|pear|pear|cherry|pear|strawberry"

         Slist = S.split("|")
         print(Slist)

         appleT = ("apple", Slist.count("apple"))
         pearT = ("pear", Slist.count("pear"))
         cherryT = ("cherry", Slist.count("cherry"))
         strawberryT = ("strawberry", Slist.count("strawberry"))
         list_of_tuples =[appleT, pearT, cherryT, strawberryT]

         print(list_of_tuples, "\n") #adding newline to separate elements

         print(appleT[0], " is present ", appleT[1], " times")
         print(pearT[0], " is present ", pearT[1], " times")
         print(cherryT[0], " is present ", cherryT[1], " times")
         print(strawberryT[0], " is present ", strawberryT[1], " times")
```

```
['apple', 'pear', 'apple', 'cherry', 'pear', 'apple', 'pear', 'pear', 'cherry', 'pear', 'strawberry']
[('apple', 3), ('pear', 5), ('cherry', 2), ('strawberry', 1)]

apple  is present  3  times
pear  is present  5  times
cherry  is present  2  times
strawberry  is present  1  times
```

8. Define three tuples representing points in the 3D space: A = (10,20,30), B = (1,72, 100) and C = (4, 9, 20).

(a) Compute the Euclidean distance between A and B (let's call it AB), A and C (AC), B and C (BC) and print them. Remember that the distance d between two points $X_1 = (x_1, y_1, z_1)$ and $X_2 = (x_2, y_2, z_2)$ is $d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$. Hint: remember to import math to use the *sqrt* method.

(b) Create a point D multiplying every element of C by 10. (Hint: do not use C*10, as this will repeat C 10 times). And compute the distance AD, BD, CD.

(c) Answer to the following questions (writing a suitable boolean expression and printing the result):

i. Is A closer to B than to C?

ii. Is A closer to C than to B or to D?

iii. Is B closer to A than to C to D?

Show/Hide Solution

```
In [23]: import math

         A = (10,20,30)
         B = (1,72, 100)
         C = (4,9,20)
         D = (C[0]*10, C[1]*10, C[2]*10)
```

```
        AB = math.sqrt( (B[0]-A[0])**2 + (B[1] - A[1])**2 + (B[2] - A[2])**2)
        AC = math.sqrt( (C[0]-A[0])**2 + (C[1] - A[1])**2 + (C[2] - A[2])**2)
        BC = math.sqrt( (C[0]-B[0])**2 + (C[1] - B[1])**2 + (C[2] - B[2])**2)
        AD = math.sqrt( (D[0]-A[0])**2 + (D[1] - A[1])**2 + (D[2] - A[2])**2)
        BD = math.sqrt( (D[0]-B[0])**2 + (D[1] - B[1])**2 + (D[2] - B[2])**2)
        CD = math.sqrt( (D[0]-C[0])**2 + (D[1] - C[1])**2 + (D[2] - C[2])**2)

        print("Distance AB: ", AB)
        print("Distance AC: ", AC)
        print("Distance AD: ", AD)
        print("Distance BC: ", BC)
        print("Distance BD: ", BD)
        print("Distance CD: ", CD)

        print("A is closer to B than to C: ", AB < AC)
        print("A is closer to C than to B or D:", AC < AB and AC < AD)
        print("B is closer to A than C to D:", AB < CD)
```

```
Distance AB:  87.66413177577246
Distance AC:  16.0312195418814
Distance AD:  186.27936010197158
Distance BC:  101.87246929372037
Distance BD:  108.83473710171766
Distance CD:  200.64147128647159
A is closer to B than to C:  False
A is closer to C than to B or D: True
B is closer to A than C to D: True
```

9. Given the matrix $M = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$ compute $M^2 = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \times \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} = \begin{bmatrix} 30 & 36 & 42 \\ 66 & 81 & 96 \\ 102 & 126 & 150 \end{bmatrix}$

Show/Hide Solution

```
In [24]: M = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

         Msq = [[0, 0, 0], [0, 0,0], [0, 0, 0]]# the output is initialized to the zero matrix
         #Msq[i,j] is represented as Msq[i][j]
         Msq[0][0] = M[0][0]*M[0][0] + M[0][1]*M[1][0] + M[0][2] * M[2][0]
         Msq[0][1] = M[0][0]*M[0][1] + M[0][1]*M[1][1] + M[0][2] * M[2][1]
         Msq[0][2] = M[0][0]*M[0][2] + M[0][1]*M[1][2] + M[0][2] * M[2][2]
         Msq[1][0] = M[1][0]*M[0][0] + M[1][1]*M[1][0] + M[1][2] * M[2][0]
         Msq[1][1] = M[1][0]*M[0][1] + M[1][1]*M[1][1] + M[1][2] * M[2][1]
         Msq[1][2] = M[1][0]*M[0][2] + M[1][1]*M[1][2] + M[1][2] * M[2][2]
         Msq[2][0] = M[2][0]*M[0][0] + M[2][1]*M[1][0] + M[2][2] * M[2][0]
         Msq[2][1] = M[2][0]*M[0][1] + M[2][1]*M[1][1] + M[2][2] * M[2][1]
         Msq[2][2] = M[2][0]*M[0][2] + M[2][1]*M[1][2] + M[2][2] * M[2][2]

         print(Msq)
```

```
[[30, 36, 42], [66, 81, 96], [102, 126, 150]]
```

CHAPTER 5

Practical 4

In this practical we will work with conditionals (branching) and loops.

5.1 Slides

The slides of the introduction can be found here: Intro

5.2 Execution flow

Recall from the lecture that there are at least three types of execution flows. Our statements can be simple and structured **sequentially**, when one instruction is executed right after the previous one, but some more complex flows involve **conditional** branching (when the portion of the code to be executed depends on the value of some condition), or **loops** when a portion of the code is executed multiple times until a certain condition becomes False.

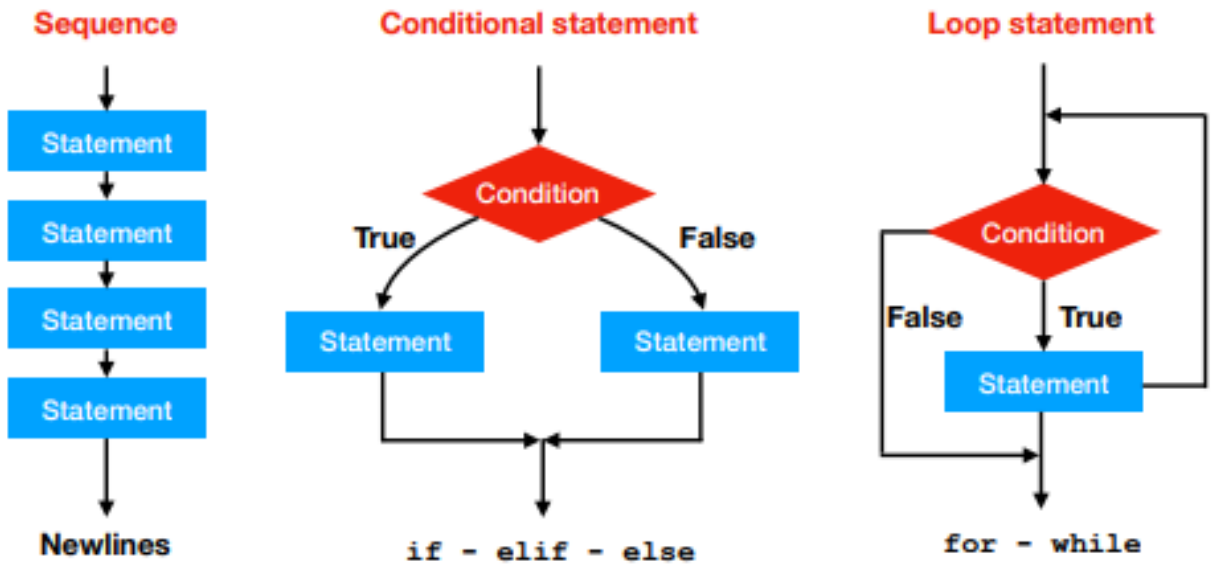
These portions of code are generally called **blocks** and Python, unlike most of the programming languages, **uses indentation (and some keywords like else, ':', 'next', etc.) to define blocks**.

5.3 Conditionals

We can use conditionals any time a decision needs to be made depending on the value of some condition. A block of code will be executed if the condition is evaluated to the boolean **True** and another one if the condition is evaluated to **False**.

5.3.1 The basic *if - else* statement

The basic syntax of conditionals is an if statement like:




```
if condition :

    #This is the True branch
    #do something

else:

    #This is the False branch (or else branch)
    #do something else
```

where condition is a boolean expression that tells the interpreter which of the two blocks should be executed. **If and only if** the condition is **True** the first branch is executed, otherwise execution goes to the second branch (i.e. the else branch). Note that the **condition is followed by a ":"** character and that **the two branches are indented**. This is the way Python uses to identify the block of instructions that belong to the same branch. The **else keyword is followed by ":"** and **is not indented** (i.e. it is at the same level of the *if* statement. There is no keyword at the end of the "else branch", but **indentation tells when the block of code is finished**.

Example: Let's get an integer from the user and test if it is even or odd, printing the result to the screen.

```
In [1]: print("Dear user give me an integer:")
        num = int(input())
        res = ""
        if num % 2 == 0:
            #The number is even
            res = "even"
        else:
            #The number is odd
            res = "odd"

        print("Number ", num, " is ", res)


Dear user give me an integer:
121
Number  121  is  odd
```

Note that the execution is sequential until the *if* keyword, then it branches until the indentation goes back to the same

level of the if (i.e. the two branches rejoin at the *print* statement in the final line). **Remember that the else branch is optional.**

5.3.2 The *if - elif - else* statement

If statements can be chained in such a way that there are more than two possible branches to be followed. Chaining them with the **if - elif - else** statement will make execution follow only one of the possible paths.

The syntax is the following:

```
if condition :

    #This is branch 1
    #do something

elif condition1 :

    #This is branch 2
    #do something

elif condition2 :

    #This is branch 3
    #do something

else:

    #else branch. Executed if all other conditions are false
    #do something else
```

Note that **branch 1** is executed if condition is **True**, **branch 2** if and only if **condition is False and condition1 is True**, **branch 3** if condition is **False, condition 1 is False and condition2 is True**. **If all conditions are False the else branch is executed**.

Example: The tax rate of a salary depends on the income. If the income is < 10000 euros, no tax is due, if the income is between 10000 euros and 20000 the tax rate is 25%, if between 20000 and 45000 it is 35% otherwise it is 40%. What is the tax due by a person earning 35000 euros per year?

```
In [2]: income = 35000
        rate = 0.0

        if income < 10000:
            rate = 0
        elif income < 20000:
            rate = 0.2
        elif income < 45000:
            rate = 0.35
        else:
            rate = 0.4

        tax = income*rate

        print("The tax due is ", tax, " euros (i.e ", rate*100, "%)")
```

```
The tax due is  12250.0  euros (i.e  35.0 %)
```

Note the difference in the two following cases:

```
In [3]: #Example 1

        val = 10

        if val > 5:
            print("Value >5")
        elif val > 5:
            print("I said value is >5!")
        else:
            print("Value is <= 5")

        #Example 2

        val = 10

        if(val > 5):
            print("\n\nValue is >5")

        if(val > 5):
            print("I said Value is >5!!!")
```

```
Value >5


Value is >5
I said Value is >5!!!
```

5.4 Loops

Looping is the ability of repeating a specific block of code several times (i.e. until a specific condition is True or there are no more elements to process).

5.4.1 For loop

The *for* loop is used to loop over a collection of objects (e.g. a string, list, tuple, . . .). The basic syntax of the for loop is the following:

```
for elem in collection :
    #OK, do something with elem
    # instruction 1
    # instruction 2
```

the variable *elem* will get the value of each one of the elements present in *collection* one after the other. The end of the block of code to be executed for each element in the collection is again defined by indentation.

Depending on the type of the collection elem will get different values. Recall from the lecture that:

`str`	`for` iterates over the characters
`list`	`for` iterates over the elements
`tuple`	`for` iterates over the elements
`dict`	`for` iterates over the keys

Let's see this in action:

```
In [4]: S = "Hi there from python"
        Slist = S.split(" ")
        Stuple = ("Hi","there","from","python")
        print("String:", S)
        print("List:", Slist)
        print("Tuple:", Stuple)

        #for loop on string
        print("On strings:")
        for c in S:
            print(c)

        print("\nOn lists:")
        #for loop on list
        for item in Slist:
            print(item)

        print("\nOn tuples:")
        #for loop on list
        for item in Stuple:
            print(item)
```

```
String: Hi there from python
List: ['Hi', 'there', 'from', 'python']
Tuple: ('Hi', 'there', 'from', 'python')
On strings:
H
i

t
h
e
r
e

f
r
o
m

p
y
t
h
o
n

On lists:
Hi
there
from
python

On tuples:
Hi
there
from
python
```

5.4.2 Looping over a range

It is possible to loop over a range of values with the pythom built-in function *range*. The *range* function accepts either two or three parameters (all of them are **integers**). Similarly to the slicing operator, it needs the **starting point**, **end point** and **an optional step**. Three distinct syntaxes are available:

```
range(E)         # ranges from 0 to E-1
range(S,E)       # ranges from S to E-1
range(S,E,step) # ranges from S to E-1 with +step jumps
```

Remember that *S* is **included** while *E* is **excluded**. Let's see some examples.

Example: Given a list of integers, return a list with all the even numbers.

```
In [5]: myList = [1, 7, 9, 121, 77, 82]
        onlyEven = []

        for i in range(0, len(myList)):  #this is equivalent to range(len(myList)):
            if( myList[i] % 2 == 0 ):
                onlyEven.append(myList[i])

        print("original list:", myList)
        print("only even numbers:", onlyEven)


original list: [1, 7, 9, 121, 77, 82]
only even numbers: [82]
```

Example: Store in a list the multiples of 19 between 1 and 100.

```
In [6]: multiples = []

        for i in range(19,101,19):
            multiples.append(i)

        print("multiples of 19: ", multiples)

        #alternative way:
        multiples = []
        for i in range(1, (100//19) + 1):
            multiples.append(i*19)
        print("multiples of 19:", multiples)


multiples of 19:  [19, 38, 57, 76, 95]
multiples of 19: [19, 38, 57, 76, 95]
```

Note: range works differently in Python 2.x and 3.x

In Python 3 the *range* function returns an iterator rather storing the entire list.

```
In [7]: #Check out the difference:
        print(range(0,10))

        print(list(range(0,10)))

range(0, 10)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Example: Let's consider the two DNA strings s1 = "ATACATATAGGGCCAATTATTATAAGTCAC" and s2 = "CGCCACTTAAGCGCCCTGTATTAAAGTCGC" that have the same length. Let's create a third string out such that $out[i]$

is "|" if $s1[i] == s2[i]$, " " otherwise.

```
In [8]: s1 = "ATACATATAGGGCCAATTATTATAAGTCAC"
        s2 = "CGCCACTTAAGCGCCCTGTATTAAAGTCGC"

        outSTR = ""
        for i in range(len(s1)):
            if(s1[i] == s2[i]):
                outSTR = outSTR + "|"
            else:
                outSTR = outSTR + " "

        print(s1)
        print(outSTR)
        print(s2)
```

```
ATACATATAGGGCCAATTATTATAAGTCAC
   ||  || |  |  |   |  ||||| |
CGCCACTTAAGCGCCCTGTATTAAAGTCGC
```

5.4.3 Nested for loops

In some occasions it is useful to nest one (or more) for loops into another one. The basic syntax is:

```
for i in collection:
    for j in another_collection:
        #do some stuff with i and j
```

Example:

Given the matrix $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$ stored as a list of lists (i.e. matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]].

Print it out as: $\begin{matrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{matrix}$

```
In [9]: matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

        for i in range(len(matrix)):
            line = ""
            for j in range(len(matrix[i])):
                line = line + str(matrix[i][j]) + " " #note int --> str conversion!
            print(line)
```

```
1 2 3
4 5 6
7 8 9
```

5.4.4 While loops

The for loop is great when we have to iterate over a finite sequence of elements. But when one needs to loop until a specific condition holds true, another construct must be used: the *while* statement. The loop will end when the condition becomes false.

The basic syntax is the following:

```
while condition:

    #do something
    #update the value of condition
```

An example follows:

```
In [10]: i = 0
         while (i < 5):
             print("i now is:", i)
             i = i + 1 #THIS IS VERY IMPORTANT!
```

```
i now is: 0
i now is: 1
i now is: 2
i now is: 3
i now is: 4
```

Note that if *condition* is false at the beginning the block of code is never executed.

Note: The loop will continue until *condition* holds true and the only code executed is the block defined through the indentation. This block of code must update the value of condition otherwise the interpreter will get stuck in the loop and will never exit.

We can combine *for* loops and *while* loops one into the code block of the other:

```
In [11]: for i in range(1,10):                          # or without string output
             j = 1                                      # for i in range(1,10):
             output = ""                                #     j = 1
             while(j<= i):                              #     while(j<=i):
                 output = str(j) + " " + output         #         print(j, end = " ")
                 j = j + 1                              #         j = j + 1
             print(output)                              #     print("")
```

```
1
2 1
3 2 1
4 3 2 1
5 4 3 2 1
6 5 4 3 2 1
7 6 5 4 3 2 1
8 7 6 5 4 3 2 1
9 8 7 6 5 4 3 2 1
```

5.5 Exercises

1. Given the integer 134479170, print if it is divisible for the numbers from 2 to 16. Hint: use for and if.

Show/Hide Solution

```
In [21]: val = 134479170
         flag = False

         for divisor in range(2,17):
             if val % divisor == 0:
                 print(val, " can be divided by ", divisor)
                 print("\t", divisor, "*", val//divisor, "=", val )
```

```
                flag = True

        if flag == False:
            print("Sorry, could not find divisors")
```

```
134479170  can be divided by  2
         2 * 67239585 = 134479170
134479170  can be divided by  3
         3 * 44826390 = 134479170
134479170  can be divided by  5
         5 * 26895834 = 134479170
134479170  can be divided by  6
         6 * 22413195 = 134479170
134479170  can be divided by  7
         7 * 19211310 = 134479170
134479170  can be divided by  9
         9 * 14942130 = 134479170
134479170  can be divided by  10
         10 * 13447917 = 134479170
134479170  can be divided by  14
         14 * 9605655 = 134479170
134479170  can be divided by  15
         15 * 8965278 = 134479170
```

2. Given the DNA string "GATTACATATATCAGTACAGATATATACGCGCGGGCTTACTATTAAAAACCCC", write a Python script that reverse-complements it. To reverse-complement a string of DNA, one needs to replace and A with T, T with A, C with G and G with C, while any other character is complemented in N. Finally, the sequence has to be reversed (e.g. the first base becomes the last). For example, ATCG becomes CGAT.

Show/Hide Solution

```
In [22]: DNA = "GATTACATATATCAGTACAGATATATACGCGCGGGCTTACTATTAAAAACCCC"

         revComp = ""
         for base in DNA:
             if base == "T":
                 revComp = "A"+ revComp
             elif base == "A":
                 revComp = "T"+ revComp
             elif base == "C":
                 revComp = "G"+ revComp
             elif base == "G":
                 revComp = "C" + revComp
             else:
                 revComp = "N" + revComp

         print("5'-", DNA, "-3'")
         print("3'-", revComp, "-5'")
```

```
5'- GATTACATATATCAGTACAGATATATACGCGCGGGCTTACTATTAAAAACCCC -3'
3'- GGGGTTTTTAATAGTAAGCCCGCGCGTATATATCTGTACTGATATATGTAATC -5'
```

```
In [29]: """ Another solution"""

         DNA = "GATTACATATATCAGTACAGATATATACGCGCGGGCTTACTATTAAAAACCCC"


         dna_list = list(DNA)
         print(dna_list)
         compl = []
         for el in dna_list:
```

```
            if el == 'A':
                compl.append('T')
            elif el == 'T':
                compl.append('A')
            elif el == 'C':
                compl.append('G')
            elif el == 'G':
                compl.append('C')
            else:
                compl.append('N')
        print(compl)
        compl.reverse()
        print(compl)
        print("".join(compl))
```

```
['G', 'A', 'T', 'T', 'A', 'C', 'A', 'T', 'A', 'T', 'A', 'T', 'C', 'A', 'G', 'T', 'A', 'C', 'A', 'G',
['C', 'T', 'A', 'A', 'T', 'G', 'T', 'A', 'T', 'A', 'T', 'A', 'G', 'T', 'C', 'A', 'T', 'G', 'T', 'C',
['G', 'G', 'G', 'G', 'T', 'T', 'T', 'T', 'T', 'A', 'A', 'T', 'A', 'G', 'T', 'A', 'A', 'G', 'C', 'C',
GGGGTTTTTAATAGTAAGCCCGCGCGTATATATCTGTACTGATATATGTAATC
```

3. Write a python script that creates the following pattern:

```
+
++
+++
++++
+++++
++++++
+++++++ <-- 7
++++++
+++++
++++
+++
++
+
```

Show/Hide Solution

```
In [14]: outStr = ""
         for i in range(0,7):
             outStr = ""
             for j in range(0,i+1):
                 outStr = outStr + "+"
             if(i == 6):
                 outStr = outStr + " <-- 7"

             print(outStr)

         for i in range(1,7):
             outStr = ""
             for j in range(0, 7-i):
                 outStr = outStr + "+"
             print(outStr)
```

```
+
++
+++
++++
+++++
++++++
```

```
+++++++ <-- 7
++++++
+++++
++++
+++
++
+
```

4. Count how many of the first 100 integers are divisible by 2, 3, 5, 7 but not by 10 and print these counts. Be aware that a number can be divisible by more than one of these numbers (e.g. 6) and therefore it must be counted as divisible by all of them (e.g. 6 must be counted as divisible by 2 and 3).

Show/Hide Solution

```
In [15]: cnts = [0,0,0,0] #cnts[0] counts for 2, cnts[1] counts for 3...
         vals = [2,3,5,7]
         for i in range(1,101):
             if ( i % 2 == 0 and i % 10 != 0 ):
                 cnts[0] = cnts[0] + 1

             if ( i % 3 == 0 and i % 10 != 0):
                 cnts[1] = cnts[1] + 1

             if ( i % 5 == 0 and i % 10 != 0):
                 cnts[2] = cnts[2] + 1

             if ( i % 7 == 0 and i % 10 != 0):
                 cnts[3] = cnts[3] + 1

         for i in range(0, len(cnts)):
             print(cnts[i], " numbers are divisible by ", vals[i], "(but not 10) in the first 100")
```

```
40  numbers are divisible by  2 (but not 10) in the first 100
30  numbers are divisible by  3 (but not 10) in the first 100
10  numbers are divisible by  5 (but not 10) in the first 100
13  numbers are divisible by  7 (but not 10) in the first 100
```

5. Given the following fastq entry:

```
@HWI-ST1296:75:C3F7CACXX:1:1101:19142:14904
CCAACAACTTTGACGCTAAGGATAGCTCCATGGCAGCATATCTGGCACAA
+
FHIIJIJJGIJJJJJ1HHHFFFFFEE:;CIDDDDDDDDDDDDEDDD-./0
```

Store the sequence and the quality in two strings. Create a list with all the quality phred scores (given a quality character "X" the phred score is: ord("X") -33. Finally print all the bases that have quality lower than 25, reporting the base, its position, quality character and phred score. **Output example:** base: C index: 14 qual:1 phred: 16).

Show/Hide Solution

```
In [16]: entry = """@HWI-ST1296:75:C3F7CACXX:1:1101:19142:14904
         CCAACAACTTTGACGCTAAGGATAGCTCCATGGCAGCATATCTGGCACAA
         +
         FHIIJIJJGIJJJJJ1HHHFFFFFEE:;CIDDDDDDDDDDDDEDDD-./0"""

         lines = entry.split("\n")
         seq = lines[1]
         qual = lines[3]

         phredScores = []
         for i in range(len(qual)):
```

```
            phredScores.append(ord(qual[i]) - 33)


        for i in range(len(seq)):
            if(phredScores[i] <25):
                print("base:",seq[i],"index:",i,"qual:",qual[i],"phredScore:",phredScores[i])
```

```
base: C index: 15 qual: 1 phredScore: 16
base: A index: 46 qual: - phredScore: 12
base: C index: 47 qual: . phredScore: 13
base: A index: 48 qual: / phredScore: 14
base: A index: 49 qual: 0 phredScore: 15
```

6. Given the following sequence:

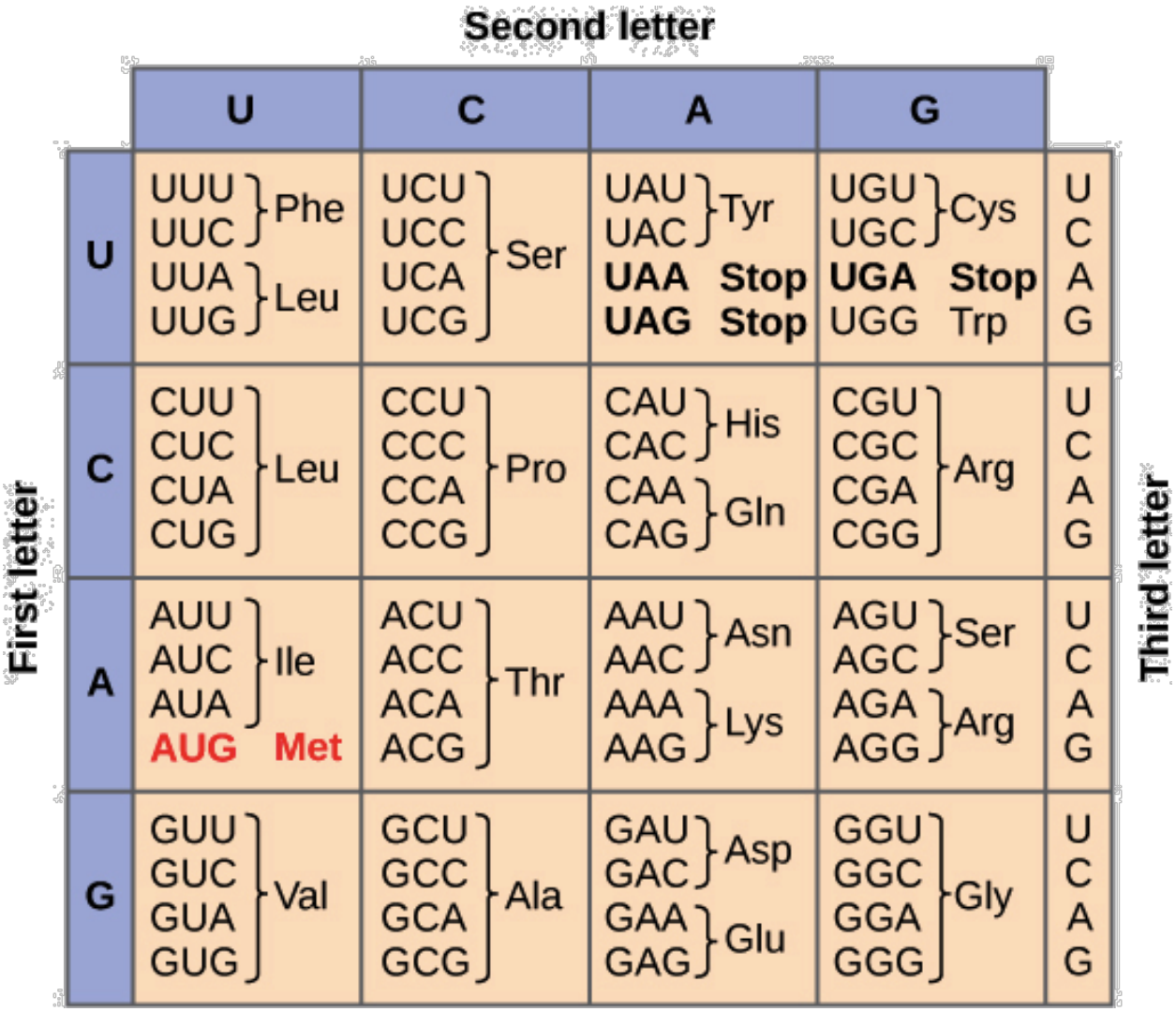
```
AUGCUGUCUCCCUCACUGUAUGUAAAUUGCAUCUAGAAUAGCA
UCUGGAGCACUAAUUGACACAUAGUGGGUAUCAAUUAUUA
UUCCAGGUACUAGAGAUACCUGGACCAUUAACGGAUAAAU
AGAAGAUUCAUUUGUUGAGUGACUGAGGAUGGCAGUUCCU
GCUACCUUCAAGGAUCUGGAUGAUGGGGAGAAACAGAGAA
CAUAGUGUGAGAAUACUGUGGUAAGGAAAGUACAGAGGAC
UGGUAGAGUGUCUAACCUAGAUUUGGAGAAGGACCUAGAA
GUCUAUCCCAGGGAAAUAAAAAUCUAAGCUAAGGUUUGAG
GAAUCAGUAGGAAUUGGCAAAGGAAGGACAUGUUCCAGAU
GAUAGGAACAGGUUAUGCAAAGAUCCUGAAAUGGUCAGAG
CUUGGUGCUUUUUGAGAACCAAAAGUAGAUUGUUAUGGAC
CAGUGCUACUCCCUGCCUCUUGCCAAGGGACCCCGCCAAG
CACUGCAUCCCUUCCCUCUGACUCCACCUUUCCACUUGCC
CAGUAUUGUUGGUGU
```

Considering the genetic code and the first forward open reading frame (i.e. the string as it is **remembering to remove newlines**).

1. How many start codons are present in the whole sequence (i.e. AUG)?
2. How many stop codons (i.e. UAA,UAG, UGA)
3. Create another string in which any codon with except the start and stop codons are substituted with "—" and print the resulting string.

Show/Hide Solution

```
In [17]: seq ="""AUGCUGUCUCCCUCACUGUAUGUAAAUUGCAUCUAGAAUAGCA
         UCUGGAGCACUAAUUGACACAUAGUGGGUAUCAAUUAUUA
         UUCCAGGUACUAGAGAUACCUGGACCAUUAACGGAUAAAU
         AGAAGAUUCAUUUGUUGAGUGACUGAGGAUGGCAGUUCCU
         GCUACCUUCAAGGAUCUGGAUGAUGGGGAGAAACAGAGAA
         CAUAGUGUGAGAAUACUGUGGUAAGGAAAGUACAGAGGAC
         UGGUAGAGUGUCUAACCUAGAUUUGGAGAAGGACCUAGAA
         GUCUAUCCCAGGGAAAUAAAAAUCUAAGCUAAGGUUUGAG
         GAAUCAGUAGGAAUUGGCAAAGGAAGGACAUGUUCCAGAU
         GAUAGGAACAGGUUAUGCAAAGAUCCUGAAAUGGUCAGAG
         CUUGGUGCUUUUUGAGAACCAAAAGUAGAUUGUUAUGGAC
         CAGUGCUACUCCCUGCCUCUUGCCAAGGGACCCCGCCAAG
         CACUGCAUCCCUUCCCUCUGACUCCACCUUUCCACUUGCC
         CAGUAUUGUUGGUG"""

         seq = seq.replace("\n","")


         countStart = 0
```


Second letter
First letter
Third letter
U
C
A
G
UUU UUC Phe
UUA UUG Leu
UCU UCC UCA UCG Ser
UAU UAC Tyr
UAA Stop
UAG Stop
UGU UGC Cys
UGA Stop
UGG Trp
CUU CUC CUA CUG Leu
CCU CCC CCA CCG Pro
CAU CAC His
CAA CAG Gln
CGU CGC CGA CGG Arg
AUU AUC AUA Ile
AUG Met
ACU ACC ACA ACG Thr
AAU AAC Asn
AAA AAG Lys
AGU AGC Ser
AGA AGG Arg
GUU GUC GUA GUG Val
GCU GCC GCA GCG Ala
GAU GAC Asp
GAA GAG Glu
GGU GGC GGA GGG Gly

```
        countEnd = 0

        newSeq = ""
        i = 0
        while i < len(seq):
            codon = seq[i:i+3]
            if(codon == "AUG"):
                    countStart = countStart + 1

            elif (codon == "UAA" or codon == "UAG" or codon == "UGA"):
                countEnd = countEnd + 1
            else:
                codon = "---"

            newSeq = newSeq + codon
            i = i + 3


        print("\nNumber of start codons:", countStart)
        print("Number of end codons:", countEnd)
        print("\n")
        print(seq)
        print("\n")
        print(newSeq)
```

```
Number of start codons: 2
Number of end codons: 12


AUGCUGUCUCCCUCACUGUAUGUAAAUUGCAUCUAGAAUAGCAUCUGGAGCACUAAUUGACACAUAGUGGGUAUCAAUUAUUAUUCCAGGUACUAGAGAUA


AUG----------------------------UAG--------------------UGA--------------------------------UAG-----
```

7. Playing time! Write a python scripts that:

   (a) Picks a random number from 1 to 10, with: import random myInt = random.randint(1,10)

   (b) Asks the user to guess a number and checks if the user has guessed the right one

   (c) If the guess is right the program will stop with a congratulation message

   (d) If the guess is wrong the program will continue asking a number, reporting the numbers already guessed (hint: store them in a list and print it).

   (e) Modify the program to notify the user if he/she inputs the same number more than once.

Show/Hide Solution

```
In [30]: import random
         myInt = random.randint(1,10)
         guessedNumbers = []

         found = False
         while (found == False):
             userInt = int(input("Guess a number from 1 to 10: "))
             if(userInt == myInt):
                 print("Congratulations. The number I guessed was ", myInt)
                 found = True
             else:
                 if(userInt in guessedNumbers):
```

```
                print("You already guessed ", userInt)
            else:
                guessedNumbers.append(userInt)
            guessedNumbers.sort()
            print("Nope, try again. So far you guessed: ", guessedNumbers)
```

```
Guess a number from 1 to 10: 7
Nope, try again. So far you guessed:  [7]
Guess a number from 1 to 10: 5
Nope, try again. So far you guessed:  [5, 7]
Guess a number from 1 to 10: 2
Nope, try again. So far you guessed:  [2, 5, 7]
Guess a number from 1 to 10: 3
Congratulations. The number I guessed was  3
```

CHAPTER 6

Practical 5

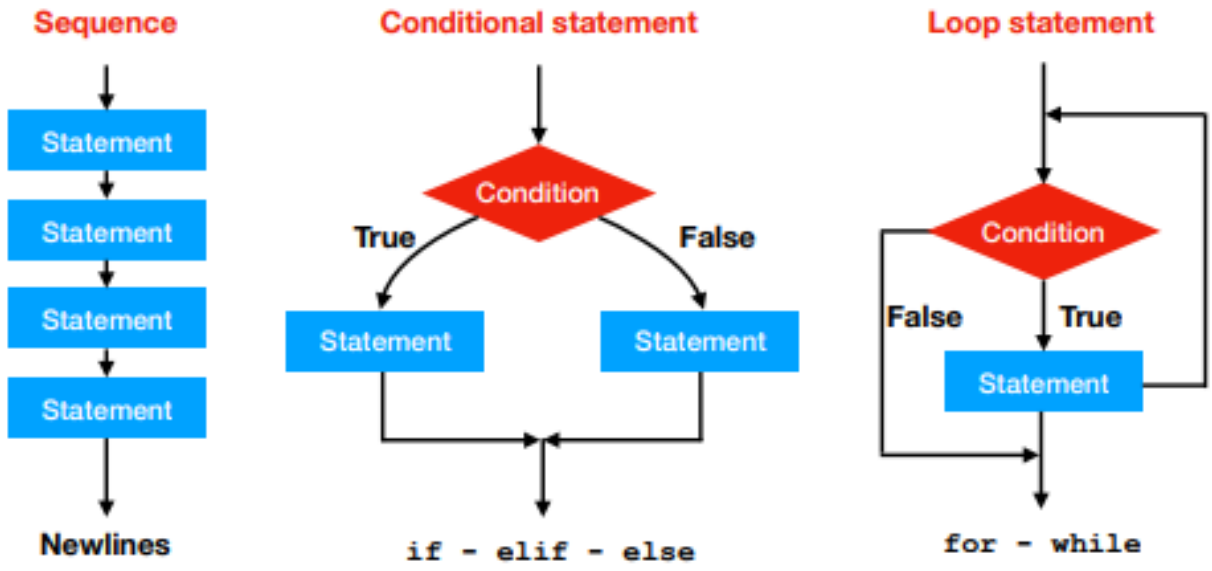
In this practical we will add some information on loops and introduce dictionaries.

6.1 Slides

The slides of the introduction can be found here: Intro

6.2 More on loops

As seen in the previous practical and in the lecture, there are three different ways of execution flow:



We have already seen the *if*, *for* and *while* loops and their variants. The code block of each of these statements is defined by the *indentation* (i.e. spacing).

6.2.1 Ternary operator

In some cases it is handy to be able to initialize a variable depending on the value of another one.

Example: The discount rate applied to a purchase depends on the amount of the sale. Create a variable *discount* setting its value to 0 if the variable *amount* is lower than 100 euros, to 10% if it is higher.

```
In [7]: amount = 110
        discount = 0

        if(amount >100):
            discount = 0.1
        else:
            discount = 0 # not necessary

        print("Total amount:", amount, "discount:", discount)
```

```
Total amount: 110 discount: 0.1
```

The previous code can be written more coincisely as:

```
In [8]: amount = 110
        discount = 0.1 if amount > 100 else 0
        print("Total amount:", amount, "discount:", discount)
```

```
Total amount: 110 discount: 0.1
```

The basic syntax of the ternary operator is:

```
variable = value if condition else other_value
```

meaning that the *variable* is initialized to *value* if the *condition* holds, otherwise to *other_value*.

Python also allows in line operations separated by a ";"

```
In [9]: a = 10; b = a + 1; c = b +2
        print(a,b,c)
```

```
10 11 13
```

> **Note:** Although the ternary operator and in line operations are sometimes useful and less verbose than the explicit definition, they are considered "non-pythonic" and advised against.

6.2.2 Break and continue

Sometimes it is useful to skip an entire iteration of a loop or end the loop before its supposed end. This can be achieved with two different statements: **continue** and **break**.

Continue statement

Within a **for** or **while** loop, **continue** makes the interpreter skip that iteration and move to the next.

Example: Print all the odd numbers from 1 to 20.

```
In [10]: #Two equivalent ways
         #1. Testing remainder == 1
         for i in range(21):
             if(i % 2 == 1):
                 print(i, end = " ")

         print("")

         #2. Skipping if remainder == 0 in for
         for i in range(21):
             if(i % 2 == 0):
                 continue
             print(i, end = " ")
```

```
1 3 5 7 9 11 13 15 17 19
1 3 5 7 9 11 13 15 17 19
```

Continue can be used also within while loops but we need to be careful to update the value of the variable before reaching the continue statement or we will get stuck in never-ending loops. **Example:** Print all the odd numbers from 1 to 20.

```
In [ ]: #Wrong code:
        i = 0
        while (i < 21):
            if(i % 2 == 0):
                continue
            print(i, end = " ")
            i = i + 1 # NEVER EXECUTED IF i % 2 == 0!!!!
```

a possible correct solution using while:

```
In [11]: i = -1
         while( i< 20):        #i is incremented in the loop, so 20!!!
             i = i + 1         #the variable is updated no matter what
             if(i % 2 == 0 ):
                 continue
             print(i, end = " ")
```

```
1 3 5 7 9 11 13 15 17 19
```

Break statement

Within a **for** or **while** loop, **break** makes the interpreter exit the loop and continue with the sequential execution. Sometimes it is useful to get out of the loop if to complete our task we do not need to get to the end of the loop.

Example: Given the following list of integers [1,5,6,4,7,1,2,3,7] print them until a number already printed is found.

```
In [12]: L = [1,5,6,4,7,1,2,3,7]
         found = []
         for i in L:
             if(i in found):
                 break

             found.append(i)
             print(i, end = " ")
```

```
1 5 6 4 7
```

Example: Pick a random number from 1 and 50 and count how many times it takes to randomly choose number 27. Limit the number of random picks to 40 (i.e. if more than 40 picks have been done and 27 has not been found exit anyway with a message).

```
In [13]: import random

         iterations = 1
         picks = []
         while(iterations <= 40):
             pick = random.randint(1,50)
             picks.append(pick)

             if(pick == 27):
                 break
             iterations += 1

         if(iterations == 41):
             print("Sorry number 27 was never found!")
         else:
             print("27 found in ", iterations, "iterations")

         print(picks)
```

```
27 found in  7 iterations
[30, 29, 16, 3, 42, 1, 27]
```

An alternative way without using the break statement makes use of a *flag* variable (that when changes value will make the loop end):

```
In [14]: import random
         found = False # This is called flag
         iterations = 1
         picks = []
         while iterations <= 40 and found == False: #the flag is used to exit
             pick = random.randint(1,50)
             picks.append(pick)
             if pick == 27:
                 found = True     #update the flag, will exit at next iteration
             iterations += 1

         if iterations == 41 and not found:
             print("Sorry number 27 was never found!")
         else:
             print("27 found in ", iterations -1, "iterations")

         print(picks)
```

```
27 found in  20 iterations
[29, 13, 17, 14, 45, 21, 39, 26, 34, 17, 23, 10, 49, 13, 39, 6, 4, 2, 22, 27]
```

6.2.3 List comprehension

List comprehension is a quick way of creating a list. The resulting list is normally obtained by applying a function or a method to the elements of another list that **remains unchanged**.

The basic syntax is:

```
new_list = [ some_function (x) for x in start_list]
```

or

```
new_list = [ x.some_method() for x in start_list]
```

List comprehension can also be used to filter elements of a list and produce another list as sublist of the first one (**remember that the original list is not changed**).

In this case the syntax is:

```
new_list = [ some_function (x) for x in start_list if condition]
```

or

```
new_list = [ x.some_method() for x in start_list if condition]
```

where the element x in start_list becomes part of new_list if and only if the condition holds True.

Let's see some examples:

Example: Given a list of strings ["hi", "there", "from", "python"] create a list with the length of the corresponding element (i.e. the one with the same index).

```
In [15]: elems = ["hi", "there", "from", "python"]

         newList = [len(x) for x in elems]

         for i in range(0,len(elems)):
             print(elems[i], " has length ", newList[i])
```

```
hi  has length  2
there  has length  5
from  has length  4
python  has length  6
```

Example: Given a list of strings ["dog", "cat", "rabbit", "guinea pig", "hamster", "canary", "goldfish"] create a list with the elements starting with a "c" or "g".

```
In [16]: pets = ["dog", "cat", "rabbit", "guinea pig", "hamster", "canary", "goldfish"]

         cg_pets = [x for x in pets if x.startswith("c") or x.startswith("g")]

         print("Original:")
         print(pets)
         print("Filtered:")
         print(cg_pets)
```

```
Original:
['dog', 'cat', 'rabbit', 'guinea pig', 'hamster', 'canary', 'goldfish']
Filtered:
['cat', 'guinea pig', 'canary', 'goldfish']
```

Example: Create a list with all the numbers divisible by 17 from 1 to 200.

```
In [17]: values = [ x for x in range(1,200) if x % 17 == 0]
         print(values)
```

```
[17, 34, 51, 68, 85, 102, 119, 136, 153, 170, 187]
```

Example: Transpose the matrix $\begin{bmatrix} 1 & 10 \\ 2 & 20 \\ 3 & 30 \\ 4 & 40 \end{bmatrix}$ stored as a list of lists (i.e. matrix = [[1, 10], [2,20], [3,30], [4,40]]). The output matrix should be: $\begin{bmatrix} 1 & 2 & 3 & 4 \\ 10 & 20 & 30 & 40 \end{bmatrix}$, represented as [[1, 2, 3, 4], [10, 20, 30, 40]]

```
In [18]: matrix = [[1, 10], [2,20], [3,30], [4,40]]
         print(matrix)
```

```
        transpose = [[row[i] for row in matrix] for i in range(2)]
        print (transpose)
```

```
[[1, 10], [2, 20], [3, 30], [4, 40]]
[[1, 2, 3, 4], [10, 20, 30, 40]]
```

Example: Given the list: ["Hotel", "Icon"," Bus","Train", "Hotel", "Eye", "Rain", "Elephant"] create a list with all the first letters.

```
In [19]: myList = ["Hotel", "Icon"," Bus","Train", "Hotel", "Eye", "Rain", "Elephant"]
         initials = [x[0] for x in myList]

         print(myList)
         print(initials)
         print("".join(initials))
```

```
['Hotel', 'Icon', ' Bus', 'Train', 'Hotel', 'Eye', 'Rain', 'Elephant']
['H', 'I', ' ', 'T', 'H', 'E', 'R', 'E']
HI THERE
```

6.3 Dictionaries

A **dictionary** is a map between one object, the **key**, and another object, the **value**. Dictionaries are **mutable objects** and contain sequences of mappings *key –> object* but there is not specific ordering among them. Dictionaries are defined using the curly braces **{key1 : value1, key2 : value2}** and **:** to separate keys from values.

Some examples on how to define dictionaries follow:

```
In [20]: first_dict = {"one" : 1, "two": 2, "three" : 3, "four" : 4}
         print("First:", first_dict)

         empty_dict = dict()
         print("Empty:",empty_dict)

         second_dict = {1 : "one", 2 : "two", "three" :3 } #BAD IDEA BUT POSSIBLE!!!
         print(second_dict)

         third_dict = dict(zip(["one","two","three","four"],[1,2,3,4]))
         print(third_dict)
         print(first_dict == third_dict)
```

```
First: {'one': 1, 'two': 2, 'three': 3, 'four': 4}
Empty: {}
{1: 'one', 2: 'two', 'three': 3}
{'one': 1, 'two': 2, 'three': 3, 'four': 4}
True
```

Note that there is no ordering of the keys, and that the order in which they have been inserted is not preserved. Moreover, keys and values can be dishomogeneous (e.g. keys strings and values integers). An interesting case is *third_dict* where the function *zip* followed by *dict* is used to map the keys of the first list into the values present in the second.

Note that keys can be **dishomogeneous**, even though this is a bad idea normally. The only requirement for the **keys** is that they **must be immutable objects**. Trying to use a mutable object as a key will make the interpreter crash with the error: **unhashable type**. Finally, keys must be unique. We cannot associate more than one value to the same key.

```
In [21]: a = (1,2,3) #a,b are tuples: hence immutable
         b = (1,3,5)

         my_dict = {a : 6, b : 9 }
```

```
print(my_dict)

c = [1,2,3] #c,d are lists: hence mutable
d = [1,3,5]

dict2 = {c : 6, d : 9}
print(dict2)
```

```
{(1, 2, 3): 6, (1, 3, 5): 9}
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-21-0fe98c7f5acd> in <module>
      8 d = [1,3,5]
      9
---> 10 dict2 = {c : 6, d : 9}
     11 print(dict2)

TypeError: unhashable type: 'list'
```

6.3.1 Functions working on dictionaries

As for the other data types, python provides several operators that can be applied to dictionaries. The following operators are available and they basically work as in lists. The only exception being that the operator **in** checks whether the specified object is present among the **keys**.

Result	Operator	Meaning
bool	obj in dict	Return True if a key is present in the dictionary
int	len(dict)	Return the number of elements in the dictionary
obj	dict[obj]	Read the value associate with a key
	dict[obj] = obj	Add or modify the value associated with a key

Some usage examples follow:

```
In [22]: myDict = {"one" : 1, "two" : 2, "twentyfive" : 25}

print(myDict)
myDict["ten"] = 10
myDict["twenty"] = 20
print(myDict)
myDict["ten"] = "10-again"
print(myDict)
print("The dictionary has ", len(myDict), " elements")
print("The value of \"ten\" is:", myDict["ten"])
print("The value of \"two\" is:", myDict["two"])

print("Is \"twentyfive\" in dictionary?", "twentyfive" in myDict)
print("Is \"seven\" in dictionary?", "seven" in myDict)
```

```
{'one': 1, 'two': 2, 'twentyfive': 25}
{'one': 1, 'two': 2, 'twentyfive': 25, 'ten': 10, 'twenty': 20}
{'one': 1, 'two': 2, 'twentyfive': 25, 'ten': '10-again', 'twenty': 20}
The dictionary has  5  elements
The value of "ten" is: 10-again
The value of "two" is: 2
Is "twentyfive" in dictionary? True
Is "seven" in dictionary? False
```

6.3.2 Dictionary methods

Recall what seen in the lecture, the following methods are available for dictionaries:

Return	Method	Meaning
`list`	`dict.keys()`	Returns the list of the keys that are present in the dictionary
`list`	`dict.values()`	Returns the list of the values that are present in the dictionary
`list of tuples`	`dict.items()`	Returns the list of pairs (key, value) that are present in the dictionary

These methods are new to dictionaries and can be used to loop through the elements in them.

ERRATUM: `dict.keys()` returns a `dict_keys` object not a list. To cast it to list, we need to call `list(dict.keys())`. The same applies to `dict.values()` that returns a `dict_values` object that needs conversion to a list with `list(dict.values())`.

Example Given the protein sequence below, store in a dictionary all the aminoacids present and count how many times they appear. Finally print out the stats (e.g. how many amino-acids are present, the most frequent, the least frequent and the frequency of all of them **in alphabetical order**).

```
>sp|P00517|KAPCA_BOVIN cAMP-dependent protein kinase catalytic subunit alpha
MGNAAAAKKGSEQESVKEFLAKAKEDFLKKWENPAQNTAHLDQFERIKTLGTGSFGRVML
VKHMETGNHYAMKILDKQKVVKLKQIEHTLNEKRILQAVNFPFLVKLEFSFKDNSNLYMV
MEYVPGGEMFSHLRRIGRFSEPHARFYAAQIVLTFEYLHSLDLIYRDLKPENLLIDQQGY
IQVTDFGFAKRVKGRTWTLCGTPEYLAPEIILSKGYNKAVDWWALGVLIYEMAAGYPPFF
ADQPIQIYEKIVSGKVRFPSHFSSDLKDLLRNLLQVDLTKRFGNLKNGVNDIKNHKWFAT
TDWIAIYQRKVEAPFIPKFKGPGDTSNFDDYEEEEIRVSINEKCGKEFSEF
```

```
In [23]: protein = """MGNAAAAKKGSEQESVKEFLAKAKEDFLKKWENPAQNTAHLDQFERIKTLGTGSFGRVML
         VKHMETGNHYAMKILDKQKVVKLKQIEHTLNEKRILQAVNFPFLVKLEFSFKDNSNLYMV
         MEYVPGGEMFSHLRRIGRFSEPHARFYAAQIVLTFEYLHSLDLIYRDLKPENLLIDQQGY
         IQVTDFGFAKRVKGRTWTLCGTPEYLAPEIILSKGYNKAVDWWALGVLIYEMAAGYPPFF
         ADQPIQIYEKIVSGKVRFPSHFSSDLKDLLRNLLQVDLTKRFGNLKNGVNDIKNHKWFAT
         TDWIAIYQRKVEAPFIPKFKGPGDTSNFDDYEEEEIRVSINEKCGKEFSEF"""

         protein = protein.replace("\n","")
```

```
print(protein)

amino_acids = dict()

for a in protein:
    if( a in amino_acids):
        amino_acids[a] = amino_acids[a] + 1
    else:
        amino_acids[a] = 1


num_aminos = len(amino_acids)

print("The number of amino-acids present is ", num_aminos)
#let's get all aminoacids
#and sort them alphabetically
a_keys = list(amino_acids.keys())

a_keys.sort()

mostF = {"frequency" : 0, "aminoacid" : "-"}
leastF = {"frequency" : len(protein), "aminoacid" : "-"}

for a in a_keys:
    freq = amino_acids[a]
    if(mostF["frequency"] < freq):
        mostF["frequency"] = freq
        mostF["aminoacid"] = a

    if(leastF["frequency"] > freq):
        leastF["frequency"] = freq
        leastF["aminoacid"] = a
    print(a, " is present", freq, "times")

print("Amino", leastF["aminoacid"], "has the lowest freq. (",leastF["frequency"],")")
print("Amino", mostF["aminoacid"], "has the highest freq. (",mostF["frequency"],")")
```

```
MGNAAAAKKGSEQESVKEFLAKAKEDFLKKWENPAQNTAHLDQFERIKTLGTGSFGRVMLVKHMETGNHYAMKILDKQKVVKLKQIEHTLNEKRILQAVNF
The number of amino-acids present is  20
A  is present 23 times
C  is present 2 times
D  is present 18 times
E  is present 27 times
F  is present 25 times
G  is present 22 times
H  is present 9 times
I  is present 21 times
K  is present 34 times
L  is present 32 times
M  is present 8 times
N  is present 17 times
P  is present 14 times
Q  is present 14 times
R  is present 15 times
S  is present 16 times
T  is present 14 times
V  is present 20 times
W  is present 6 times
Y  is present 14 times
```

```
Amino C has the lowest freq. ( 2 )
Amino K has the highest freq. ( 34 )
```

Important NOTE. Accessing a value through the key of a dictionary requires that the pair key-value one searches for is present in the dictionary. If the searched key is not present the interpreter crashes out throwing a KeyError as follows:

```
In [24]: myDict = {"one" : 1, "two" : 2, "three" : 3}

         print(myDict["one"])
         print(myDict["seven"])


1

---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-24-a05b31e54a02> in <module>
      2
      3 print(myDict["one"])
----> 4 print(myDict["seven"])

KeyError: 'seven'
```

This could be avoided by checking that the key is present in the dictionary beforehand:

```
In [25]: myDict = {"one" : 1, "two" : 2, "three" : 3}

         search_keys = ["one", "seven"]

         for s in search_keys:
             if s in myDict:
                 print("key:", s, "value:", myDict[s])
             else:
                 print("key", s, "not found in dictionary")
```

```
key: one value: 1
key seven not found in dictionary
```

or by using the dictionary method **get** that has two inputs: the key and a default value to return in case key is not present in the dictionary:

```
In [26]: myDict = {"one" : 1, "two" : 2, "three" : 3}

         search_keys = ["one", "seven"]

         for s in search_keys:
             print("key:", s, "value:", myDict.get(s, "not found"))
```

```
key: one value: 1
key: seven value: not found
```

6.4 Exercises

1. Given the following two lists of integers: [1, 13, 22, 7, 43, 81, 77, 12, 15,21, 84,100] and [44,32,7, 100, 81, 13, 1, 21, 71]:
   (a) Sort the two lists
   (b) Create a third list as intersection of the two lists (i.e. an element is in the intersection if it is present in both lists).

(c) Print the three lists.

Show/Hide Solution

```
In [27]: L1 = [1, 13, 22, 7, 43, 81, 77, 12, 15,21, 84,100]
         L2 = [44,32,7, 100, 81, 13, 1, 21, 71]

         L1.sort()
         L2.sort()
         intersection = [x for x in L1 if x in L2]

         print("L1:    ", L1)
         print("L2:    ", L2)
         print("inters:", intersection)
```

```
L1:     [1, 7, 12, 13, 15, 21, 22, 43, 77, 81, 84, 100]
L2:     [1, 7, 13, 21, 32, 44, 71, 81, 100]
inters: [1, 7, 13, 21, 81, 100]
```

2. Given the string "nOBody Said iT was eAsy, No oNe Ever saId it WoulD be tHis hArd..."

   (a) Create a list with all the letters that are capitalized (use str.isupper)

   (b) Print the list

   (c) Use the string method **join** to concatenate all the letters in a string, using "*" as separator. The syntax of join is str.join(list) and it outputs a string with all the elements in list joined with the character in str (es. "+".join([1,2,3]) returns "1+2+3").

The expected output:

```
['O', 'B', 'S', 'T', 'A', 'N', 'N', 'E', 'I', 'W', 'D', 'H', 'A']
O*B*S*T*A*N*N*E*I*W*D*H*A
```

Show/Hide Solution

```
In [28]: text = "nOBody Said iT was eAsy, No oNe Ever saId it WoulD be tHis hArd..."
         initials = [x for x in text if x.isupper()]

         print(initials)
         print("*".join(initials))
```

```
['O', 'B', 'S', 'T', 'A', 'N', 'N', 'E', 'I', 'W', 'D', 'H', 'A']
O*B*S*T*A*N*N*E*I*W*D*H*A
```

3. Given the following sequence:

```
AUGCUGUCUCCCUCACUGUAUGUAAAUUGCAUCUAGAAUAGCA
UCUGGAGCACUAAUUGACACAUAGUGGGUAUCAAUUAUUA
UUCCAGGUACUAGAGAUACCUGGACCAUUAACGGAUAAAU
AGAAGAUUCAUUUGUUGAGUGACUGAGGAUGGCAGUUCCU
GCUACCUUCAAGGAUCUGGAUGAUGGGGAGAAACAGAGAA
CAUAGUGUGAGAAUACUGUGGUAAGGAAAGUACAGAGGAC
UGGUAGAGUGUCUAACCUAGAUUUGGAGAAGGACCUAGAA
GUCUAUCCCAGGGAAAUAAAAAUCUAAGCUAAGGUUUGAG
GAAUCAGUAGGAAUUGGCAAAGGAAGGACAUGUUCCAGAU
GAUAGGAACAGGUUAUGCAAAGAUCCUGAAAUGGUCAGAG
CUUGGUGCUUUUUGAGAACCAAAAGUAGAUUGUUAUGGAC
CAGUGCUACUCCCUGCCUCUUGCCAAGGGACCCCGCCAAG
CACUGCAUCCCUUCCCUCUGACUCCACCUUUCCACUUGCC
CAGUAUUGUUGGUG
```

considering only first forward open reading frame (i.e. the string as it is **remembering to remove newlines**): 1. create a list of all the codons and print it.

2. create then a dictionary in which codons are keys and values are how many times that codon is present in the first forward open reading frame and print it; 3. retrieve from the dictionary and print the number of times codons: "AUG", "AGA", "GGG", "AUA" are present. If a codon is not present the script should print "NOT present" rather than crash.

Show/Hide Solution

```
In [38]: seq= """AUGCUGUCUCCCUCACUGUAUGUAAAUUGCAUCUAGAAUAGCA
         UCUGGAGCACUAAUUGACACAUAGUGGGUAUCAAUUAUUA
         UUCCAGGUACUAGAGAUACCUGGACCAUUAACGGAUAAAU
         AGAAGAUUCAUUUGUUGAGUGACUGAGGAUGGCAGUUCCU
         GCUACCUUCAAGGAUCUGGAUGAUGGGGAGAAACAGAGAA
         CAUAGUGUGAGAAUACUGUGGUAAGGAAAGUACAGAGGAC
         UGGUAGAGUGUCUAACCUAGAUUUGGAGAAGGACCUAGAA
         GUCUAUCCCAGGGAAAUAAAAAUCUAAGCUAAGGUUUGAG
         GAAUCAGUAGGAAUUGGCAAAGGAAGGACAUGUUCCAGAU
         GAUAGGAACAGGUUAUGCAAAGAUCCUGAAAUGGUCAGAG
         CUUGGUGCUUUUUGAGAACCAAAAGUAGAUUGUUAUGGAC
         CAGUGCUACUCCCUGCCUCUUGCCAAGGGACCCCGCCAAG
         CACUGCAUCCCUUCCCUCUGACUCCACCUUUCCACUUGCC
         CAGUAUUGUUGGUG"""


         seq = seq.replace("\n","")

         codons = [seq[i:i+3] for i in range(0,len(seq),3)]

         count_dict = dict()

         for el in codons:
             if el not in count_dict:
                 count_dict[el] = codons.count(el)

         print("1st fwd ORF codons:")
         print(codons)
         print("Codon counts:")
         print(count_dict)
         print("\n")
         to_check = ["AUG", "AGA", "GGG", "AUA"]
         for c in to_check:
             if c in count_dict:
                 print(c , "is present", count_dict[c], "times")
             else:
                 print(c, "is NOT present")
```

```
1st fwd ORF codons:
['AUG', 'CUG', 'UCU', 'CCC', 'UCA', 'CUG', 'UAU', 'GUA', 'AAU', 'UGC', 'AUC', 'UAG', 'AAU', 'AGC', 'A
Codon counts:
{'AUG': 2, 'CUG': 3, 'UCU': 2, 'CCC': 3, 'UCA': 1, 'UAU': 3, 'GUA': 2, 'AAU': 8, 'UGC': 7, 'AUC': 4,


AUG is present 2 times
AGA is present 8 times
GGG is NOT present
AUA is present 3 times
```

4. Given the following list of gene correlations:

```
geneCorr = [["G1C2W9", "G1C2Q7", 0.2], ["G1C2W9", "G1C2Q4", 0.9],
["Q6NMS1", "G1C2W9", 0.8],["G1C2W9", "Q6NMS1",0.4], ["G1C2Q7", "G1C2Q4",0.76]]
```

where each sublist ["gene1", "gene2", corr] represents a correlation between *gene1* and *gene2* with correlation *corr*, create another list containing only the elements having an high correlation (i.e. > 0.75). Print this list.

Expected result:

```
[['G1C2W9', 'G1C2Q4', 0.9], ['Q6NMS1', 'G1C2W9', 0.8], ['G1C2Q7', 'G1C2Q4', 0.76]]
```

Show/Hide Solution

```
In [30]: geneCorr = [["G1C2W9", "G1C2Q7", 0.2], ["G1C2W9", "G1C2Q4", 0.9], ["Q6NMS1", "G1C2W9", 0.8],
                     ["G1C2W9", "Q6NMS1",0.4], ["G1C2Q7", "G1C2Q4",0.76]]

         highlyCorr = [x for x in geneCorr if x[2] > 0.75]

         print(geneCorr, "\n")
         print(highlyCorr)
```

```
[['G1C2W9', 'G1C2Q7', 0.2], ['G1C2W9', 'G1C2Q4', 0.9], ['Q6NMS1', 'G1C2W9', 0.8], ['G1C2W9', 'Q6NMS1

[['G1C2W9', 'G1C2Q4', 0.9], ['Q6NMS1', 'G1C2W9', 0.8], ['G1C2Q7', 'G1C2Q4', 0.76]]
```

5. Given the following sequence of DNA:

DNA = "GATTACATATATCAGTACAGATATATACGCGCGGGCTTACTATTAAAAACCCC"

```
1. Create a dictionary reporting the frequency of each base (i.e. key is the
base and value is the frequency).
2. Create a dictionary representing an index of all possible dimers (i.e. 2
bases, 16 dimers in total): AA, AT, AC, AG, TA, TT, TC, TG, ... . In this case,
keys of the dictionary are dimers and values are lists with all possible starting
positions of the dimer.
3. Print the DNA string.
4. Print for each base its frequency
4. Print all positions of the dimer "AT"
```

The expected result is:

```
sequence: GATTACATATATCAGTACAGATATATACGCGCGGGCTTACTATTAAAAACCCC
G has frequency: 0.1509433962264151
C has frequency: 0.22641509433962265
A has frequency: 0.3584905660377358
T has frequency: 0.2641509433962264
{'GG': [32, 33], 'TC': [11], 'GT': [14], 'CA': [5, 12, 17], 'TT': [2, 36, 42],
'CG': [27, 29, 31], 'TA': [3, 7, 9, 15, 21, 23, 25, 37, 40, 43], 'AG': [13, 18],
'GA': [0, 19], 'CT': [35, 39], 'GC': [28, 30, 34], 'AT': [1, 6, 8, 10, 20, 22, 24,␣
↪41],
'CC': [49, 50, 51], 'AA': [44, 45, 46, 47], 'AC': [4, 16, 26, 38, 48]}

Dimer AT is found at: [1, 6, 8, 10, 20, 22, 24, 41]
```

Show/Hide Solution

```
In [31]: DNA = "GATTACATATATCAGTACAGATATATACGCGCGGGCTTACTATTAAAAACCCC"

         n = len(DNA)
```

```
        baseFreq = {"A" : DNA.count("A")/n, "T" : DNA.count("T")/n,
                    "C": DNA.count("C")/n, "G" : DNA.count("G")/n }

        dimersDict ={}

        print("sequence:", DNA)

        for base in baseFreq:
            print(base, "has frequency:", baseFreq[base])

        for ind in range(len(DNA) -1 ): #need -1 because at each iteration I get the dimer [ind:ind+
            dimer = DNA[ind:ind+2]
            if(dimer in dimersDict):
                dimersDict[dimer].append(ind)
            else:
                dimersDict[dimer] = [ind]
        print(dimersDict, "\n")
        print("Dimer AT is found at:", dimersDict["AT"])
```

```
sequence: GATTACATATATCAGTACAGATATATACGCGCGGGCTTACTATTAAAAACCCC
A has frequency: 0.3584905660377358
T has frequency: 0.2641509433962264
C has frequency: 0.22641509433962265
G has frequency: 0.1509433962264151
{'GA': [0, 19], 'AT': [1, 6, 8, 10, 20, 22, 24, 41], 'TT': [2, 36, 42], 'TA': [3, 7, 9, 15, 21, 23, 2

Dimer AT is found at: [1, 6, 8, 10, 20, 22, 24, 41]
```

6. Given the following table, reporting molecular weights for each amino acid, store them in a dictionary where the key is the one letter code and the value is the molecular weight (e.g. {"A" : 89, "R":174"}).

Write a python script to answer the following questions:

```
1. What is the average molecular weight of an amino acid?
2. What is the total molecular weight and number of aminoacids
of the P53 peptide GSRAHSSHLKSKKGQSTSRHK?
3. What is the total molecular weight and number of aminoacids
of the peptide YTSLIHSLIEESQNQQEKNEQELLELDKWASLWNWF?
```

Show/Hide Solution

```
In [41]: mws = {"A" : 89, "R" : 174, "N" : 132, "D" : 133, "B": 133, "C": 121, "Q": 146,
                "E": 147, "Z": 147, "G": 75, "H": 155, "I": 131, "L" : 131,
               "K" : 146, "M" : 149, "F" : 165, "P" : 115, "S" : 105, "T" : 119,
                "W" : 204, "Y": 181, "V" : 117
               }
         avgW = 0
         for amino in mws:
             avgW = avgW + mws[amino]

         avgW = avgW/len(mws)
         print("The average molecular weight of amino acids is:", avgW)

         P53amino = "GSRAHSSHLKSKKGQSTSRHK"

         totW = 0
         for amino in P53amino:
             totW += mws[amino]
         print("Peptide ", P53amino, "has",len(P53amino), "amino acids and a total mw of",totW, "Da")
```

Amino Acid	Three-Letter Abbreviation	One-Letter Symbol	Molecular Weight
Alanine	Ala	A	89Da
Arginine	Arg	R	174Da
Asparagine	Asn	N	132Da
Aspartic acid	Asp	D	133Da
Asparagine or aspartic acid	Asx	B	133Da
Cysteine	Cys	C	121Da
Glutamine	Gln	Q	146Da
Glutamic acid	Glu	E	147Da
Glutamine or glutamic acid	Glx	Z	147Da
Glycine	Gly	G	75Da
Histidine	His	H	155Da
Isoleucine	Ile	I	131Da
Leucine	Leu	L	131Da
Lysine	Lys	K	146Da
Methionine	Met	M	149Da
Phenylalanine	Phe	F	165Da
Proline	Pro	P	115Da
Serine	Ser	S	105Da
Threonine	Thr	T	119Da
Tryptophan	Trp	W	204Da
Tyrosine	Tyr	Y	181Da
Valine	Val	V	117Da

```
totW = 0 #reusing same variable name
peptide = "YTSLIHSLIEESQNQQEKNEQELLELDKWASLWNWF"
for amino in peptide:
    totW += mws[amino]

print("Peptide ", peptide, "has",len(peptide), "amino acids and a total mw of",totW, "Da")
```

```
The average molecular weight of amino acids is: 137.04545454545453
Peptide  GSRAHSSHLKSKKGQSTSRHK has 21 amino acids and a total mw of 2662 Da
Peptide  YTSLIHSLIEESQNQQEKNEQELLELDKWASLWNWF has 36 amino acids and a total mw of 5076 Da
```

7. The following string is an extract of a blast[30] alignment with compacted textual output. This is a comma (",") separated text file where the columns report the following info: the first column is the query_id, the second the subject_id (i.e. the reference on which we aligned the query), the third is the percentage of identity and then we have the alignment length, number of mismatches, gap opens, start point of the alignment on the query, end point of the alignment on the query, start point of the alignment on the subject, end point of the alignment on the subject and evalue of the alignment.

```
#Fields:q.id,s.id,% ident,align len,mismatches,gap opens,q.start,q.end,s.start,s.end,
↪evalue
ab1_400a,scaffold16155,98.698,384,4,1,12,394,6700,7083,0.0
ab1_400b,scaffold14620,98.698,384,4,1,12,394,1240,857,0.0
92A2_SP6_344a,scaffold14394,95.575,113,5,0,97,209,250760,250648,2.92e-44
92A2_SP6_344b,scaffold10682,97.849,93,2,0,18,110,898,990,3.81e-38
92A2_T7_558a,scaffold277,88.746,311,31,3,21,330,26630,26937,5.81e-103
92A2_T7_558b,scaffold277,89.545,220,21,2,27,246,27167,26950,6.06e-73
92A2_T7_558c,scaffold1125,88.125,320,31,5,30,346,231532,231847,7.51e-102
ab1_675a,scaffold4896,100.000,661,0,0,15,675,79051,78391,0.0
ab1_676b,scaffold4896,99.552,670,0,3,7,673,78421,79090,0.0
```

1. For each alignment, store the subject id, the percentage of identity and evalue, subject start and end in a dictionary using the query id as key. All this information can be stored in a dictionary having subject id as key and a dictionary with all the information as value:

```
alignments["ab1_400"] = {"subjectid" : scaffold16155, "perc_id" : 98.698, "evalue
↪" : 0.0}
```

2. Print the whole dictionary
3. Print only the alignments having percentage of identity > 90%

Note: skip the first comment line (i.e. skip line if starts with "#"). Note1: when storing the percentage of identity remember to convert the string into a float.

The expected output is:

```
{'92A2_T7_558a': {'perc_id': 88.746, 'subjectid': 'scaffold277', 'evalue': '5.81e-103
↪'},
'92A2_T7_558b': {'perc_id': 89.545, 'subjectid': 'scaffold277', 'evalue': '6.06e-73'},
'92A2_SP6_344b': {'perc_id': 97.849, 'subjectid': 'scaffold10682', 'evalue': '3.81e-38
↪'},
'92A2_T7_558c': {'perc_id': 88.125, 'subjectid': 'scaffold1125', 'evalue': '7.51e-102
↪'},
'ab1_400a': {'perc_id': 98.698, 'subjectid': 'scaffold16155', 'evalue': '0.0'},
'ab1_675a': {'perc_id': 100.0, 'subjectid': 'scaffold4896', 'evalue': '0.0'},
'ab1_400b': {'perc_id': 98.698, 'subjectid': 'scaffold14620', 'evalue': '0.0'},
'92A2_SP6_344a': {'perc_id': 95.575, 'subjectid': 'scaffold14394', 'evalue': '2.92e-44
↪'},
```

(continues on next page)

[30] https://www.ncbi.nlm.nih.gov/pubmed/2231712

(continued from previous page)

```
'ab1_676b': {'perc_id': 99.552, 'subjectid': 'scaffold4896', 'evalue': '0.0'}}


Alignments with identity > 90%:

Query id    Subject id  % ident evalue
92A2_SP6_344b    scaffold10682   97.849      3.81e-38
ab1_400a     scaffold16155   98.698      0.0
ab1_675a     scaffold4896    100.0   0.0
ab1_400b     scaffold14620   98.698      0.0
92A2_SP6_344a    scaffold14394   95.575      2.92e-44
ab1_676b     scaffold4896    99.552      0.0
```

Show/Hide Solution

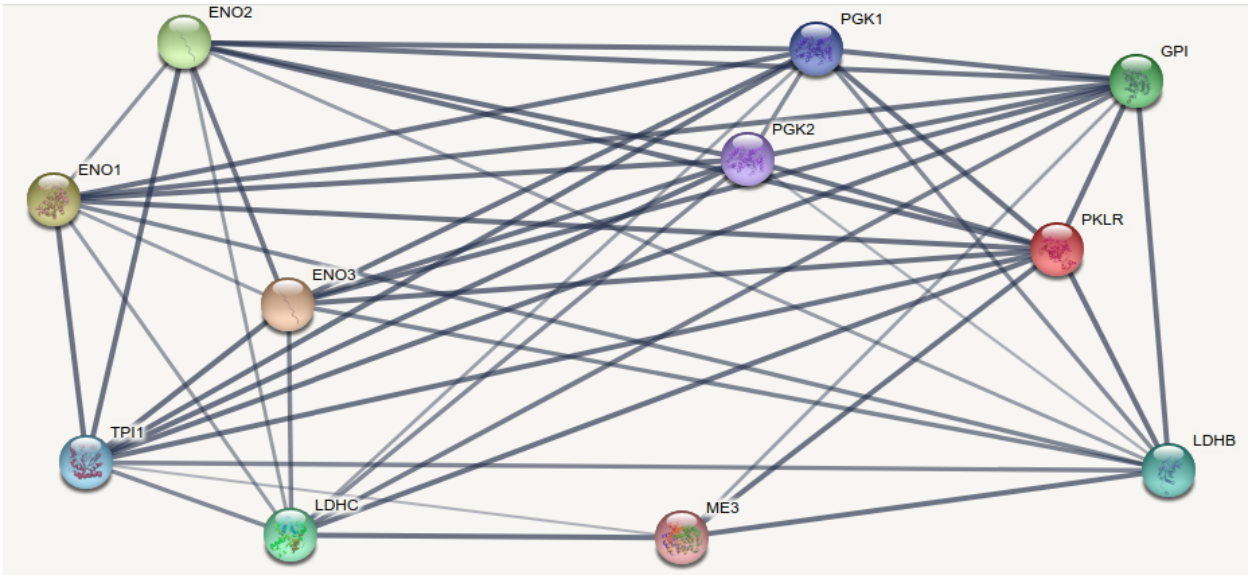
```
In [33]: blast_out = """#Fields:q.id,s.id,% ident,align len,mismatches,gap opens,q.start,q.end,s.sta
         ab1_400a,scaffold16155,98.698,384,4,1,12,394,6700,7083,0.0
         ab1_400b,scaffold14620,98.698,384,4,1,12,394,1240,857,0.0
         92A2_SP6_344a,scaffold14394,95.575,113,5,0,97,209,250760,250648,2.92e-44
         92A2_SP6_344b,scaffold10682,97.849,93,2,0,18,110,898,990,3.81e-38
         92A2_T7_558a,scaffold277,88.746,311,31,3,21,330,26630,26937,5.81e-103
         92A2_T7_558b,scaffold277,89.545,220,21,2,27,246,27167,26950,6.06e-73
         92A2_T7_558c,scaffold1125,88.125,320,31,5,30,346,231532,231847,7.51e-102
         ab1_675a,scaffold4896,100.000,661,0,0,15,675,79051,78391,0.0
         ab1_676b,scaffold4896,99.552,670,0,3,7,673,78421,79090,0.0"""



         alignments = dict()

         for align in blast_out.split("\n"):
             if(align.startswith("#") == False):
                 align_info = align.split(",")
                 alignments[align_info[0]] = {"subjectid" : align_info[1],
                                              "perc_id" : float(align_info[2]),
                                             "evalue" : align_info[-1]
                                             }

         print(alignments, "\n\n")
         print("Alignments with identity > 90%:\n")
         print("Query id\tSubject id\t% ident\tevalue")
         for a in alignments:
             if(alignments[a]["perc_id"]>90):
                 print(a,"\t", alignments[a]["subjectid"],"\t", alignments[a]["perc_id"],"\t", alignm

{'ab1_400a': {'subjectid': 'scaffold16155', 'perc_id': 98.698, 'evalue': '0.0'}, 'ab1_400b': {'subjec


Alignments with identity > 90%:

Query id        Subject id      % ident         evalue
ab1_400a        scaffold16155   98.698           0.0
ab1_400b        scaffold14620   98.698           0.0
92A2_SP6_344a           scaffold14394   95.575          2.92e-44
92A2_SP6_344b           scaffold10682   97.849          3.81e-38
ab1_675a        scaffold4896    100.0   0.0
ab1_676b        scaffold4896    99.552           0.0
```

8. The following text (separated with a comma ",") is an extract of protein-protein interactions network stored in the database STRING[31] involving PKLR[32] (Pyruvate kinase, liver and RBC) that plays a key role in glycolysis:

```
#node1,node2,node1_ext_id,node2_ext_id,
ENO1,TPI1,ENSP00000234590,ENSP00000229270
PKLR,ENO1,ENSP00000339933,ENSP00000234590
PKLR,ENO3,ENSP00000339933,ENSP00000324105
PGK1,ENO1,ENSP00000362413,ENSP00000234590
PGK1,TPI1,ENSP00000362413,ENSP00000229270
GPI,TPI1,ENSP00000405573,ENSP00000229270
PKLR,ENO2,ENSP00000339933,ENSP00000229277
PGK1,ENO3,ENSP00000362413,ENSP00000324105
PGK1,ENO2,ENSP00000362413,ENSP00000229277
GPI,PKLR,ENSP00000405573,ENSP00000339933
ENO2,TPI1,ENSP00000229277,ENSP00000229270
PGK2,ENO1,ENSP00000305995,ENSP00000234590
ENO3,PGK2,ENSP00000324105,ENSP00000305995
PGK2,TPI1,ENSP00000305995,ENSP00000229270
ENO3,TPI1,ENSP00000324105,ENSP00000229270
PGK2,ENO2,ENSP00000305995,ENSP00000229277
GPI,ENO3,ENSP00000405573,ENSP00000324105
PKLR,LDHB,ENSP00000339933,ENSP00000229319
PKLR,LDHC,ENSP00000339933,ENSP00000280704
PKLR,TPI1,ENSP00000339933,ENSP00000229270
PGK1,PKLR,ENSP00000362413,ENSP00000339933
GPI,ENO2,ENSP00000405573,ENSP00000229277
PKLR,PGK2,ENSP00000339933,ENSP00000305995
GPI,PGK1,ENSP00000405573,ENSP00000362413
ME3,PKLR,ENSP00000352657,ENSP00000339933
ME3,LDHB,ENSP00000352657,ENSP00000229319
ME3,LDHC,ENSP00000352657,ENSP00000280704
GPI,PGK2,ENSP00000405573,ENSP00000305995
GPI,ENO1,ENSP00000405573,ENSP00000234590
GPI,LDHB,ENSP00000405573,ENSP00000229319
ENO3,ENO2,ENSP00000324105,ENSP00000229277
GPI,LDHC,ENSP00000405573,ENSP00000280704
ENO3,LDHB,ENSP00000324105,ENSP00000229319
ENO3,LDHC,ENSP00000324105,ENSP00000280704
ENO1,LDHB,ENSP00000234590,ENSP00000229319
LDHB,TPI1,ENSP00000229319,ENSP00000229270
LDHC,TPI1,ENSP00000280704,ENSP00000229270
PGK2,LDHC,ENSP00000305995,ENSP00000280704
PGK1,LDHB,ENSP00000362413,ENSP00000229319
PGK1,PGK2,ENSP00000362413,ENSP00000305995
ENO1,ENO2,ENSP00000234590,ENSP00000229277
LDHC,ENO1,ENSP00000280704,ENSP00000234590
LDHB,ENO2,ENSP00000229319,ENSP00000229277
LDHC,ENO2,ENSP00000280704,ENSP00000229277
ENO3,ENO1,ENSP00000324105,ENSP00000234590
PGK1,LDHC,ENSP00000362413,ENSP00000280704
GPI,ME3,ENSP00000405573,ENSP00000352657
PGK2,LDHB,ENSP00000305995,ENSP00000229319
ME3,TPI1,ENSP00000352657,ENSP00000229270
```

Here is a graphic representation of the protein-protein interactions:

[31] https://string-db.org/
[32] https://www.ncbi.nlm.nih.gov/gene/5313



Edges:

Edges represent protein-protein associations

associations are meant to be specific and meaningful, i.e. proteins jointly contribute to a shared function; this does not necessarily mean they are physically binding each other.

Edge Confidence

low (0.150)
medium (0.400)
high (0.700)
highest (0.900)

Nodes:

Network nodes represent proteins

splice isoforms or post-translational modifications are collapsed, i.e. each node represents all the proteins produced by a single, protein-coding gene locus.

Node Color

colored nodes:
query proteins and first shell of interactors

white nodes:
second shell of interactors

Node Content

empty nodes:
proteins of unknown 3D structure

filled nodes:
some 3D structure is known or predicted

```
Note: we can assume that relations between nodes are transitive. node1 --> node2
implies node2 --> node1.

1. Store the network information in a dictionary having node1 as key and
the list of all nodes2 associated to it as value (remember to skip the first
line that is the header). Remember transitivity, therefore add also node2 -->node1
2. Find all first neighbours of "PKLR" (i.e. the nodes that are directly connected
to "PKLR") and print them
3. Find all first neighbours of "ME3" (i.e. the nodes that are directly connected
to "PKLR") and print them
4. Find all the second neighbours of "ME3" (i.e. the nodes that are connected to nodes
directly connected to "ME3", but not directly to "ME3").
```

Show/Hide Solution

```
In [34]: networkStr = """#node1,node2,node1_ext_id,node2_ext_id,
         ENO1,TPI1,ENSP00000234590,ENSP00000229270
         PKLR,ENO1,ENSP00000339933,ENSP00000234590
         PKLR,ENO3,ENSP00000339933,ENSP00000324105
         PGK1,ENO1,ENSP00000362413,ENSP00000234590
         PGK1,TPI1,ENSP00000362413,ENSP00000229270
         GPI,TPI1,ENSP00000405573,ENSP00000229270
         PKLR,ENO2,ENSP00000339933,ENSP00000229277
         PGK1,ENO3,ENSP00000362413,ENSP00000324105
         PGK1,ENO2,ENSP00000362413,ENSP00000229277
         GPI,PKLR,ENSP00000405573,ENSP00000339933
         ENO2,TPI1,ENSP00000229277,ENSP00000229270
         PGK2,ENO1,ENSP00000305995,ENSP00000234590
         ENO3,PGK2,ENSP00000324105,ENSP00000305995
         PGK2,TPI1,ENSP00000305995,ENSP00000229270
         ENO3,TPI1,ENSP00000324105,ENSP00000229270
         PGK2,ENO2,ENSP00000305995,ENSP00000229277
         GPI,ENO3,ENSP00000405573,ENSP00000324105
         PKLR,LDHB,ENSP00000339933,ENSP00000229319
         PKLR,LDHC,ENSP00000339933,ENSP00000280704
         PKLR,TPI1,ENSP00000339933,ENSP00000229270
         PGK1,PKLR,ENSP00000362413,ENSP00000339933
         GPI,ENO2,ENSP00000405573,ENSP00000229277
         PKLR,PGK2,ENSP00000339933,ENSP00000305995
         GPI,PGK1,ENSP00000405573,ENSP00000362413
         ME3,PKLR,ENSP00000352657,ENSP00000339933
         ME3,LDHB,ENSP00000352657,ENSP00000229319
         ME3,LDHC,ENSP00000352657,ENSP00000280704
         GPI,PGK2,ENSP00000405573,ENSP00000305995
         GPI,ENO1,ENSP00000405573,ENSP00000234590
         GPI,LDHB,ENSP00000405573,ENSP00000229319
         ENO3,ENO2,ENSP00000324105,ENSP00000229277
         GPI,LDHC,ENSP00000405573,ENSP00000280704
         ENO3,LDHB,ENSP00000324105,ENSP00000229319
         ENO3,LDHC,ENSP00000324105,ENSP00000280704
         ENO1,LDHB,ENSP00000234590,ENSP00000229319
         LDHB,TPI1,ENSP00000229319,ENSP00000229270
         LDHC,TPI1,ENSP00000280704,ENSP00000229270
         PGK2,LDHC,ENSP00000305995,ENSP00000280704
         PGK1,LDHB,ENSP00000362413,ENSP00000229319
         PGK1,PGK2,ENSP00000362413,ENSP00000305995
         ENO1,ENO2,ENSP00000234590,ENSP00000229277
         LDHC,ENO1,ENSP00000280704,ENSP00000234590
```

```
        LDHB,ENO2,ENSP00000229319,ENSP00000229277
        LDHC,ENO2,ENSP00000280704,ENSP00000229277
        ENO3,ENO1,ENSP00000324105,ENSP00000234590
        PGK1,LDHC,ENSP00000362413,ENSP00000280704
        GPI,ME3,ENSP00000405573,ENSP00000352657
        PGK2,LDHB,ENSP00000305995,ENSP00000229319
        ME3,TPI1,ENSP00000352657,ENSP00000229270"""

        netData = dict()

        for line in networkStr.split("\n"):

            if(not line.startswith("#") ):
                info = line.split(",")
                #insert node1
                if(info[0] in netData):
                    netData[info[0]].append(info[1])
                else:
                    netData[info[0]] = [info[1]]
                #insert node2
                if(info[1] in netData):
                    netData[info[1]].append(info[0])
                else:
                    netData[info[1]] = [info[0]]


        print(netData)

        print("\nPKLR is directly connected to:", netData["PKLR"] )

        firstNeigh = netData["ME3"]
        print("\nME3 is directly connected to:", firstNeigh  )

        secondNeigh = []
        for node in firstNeigh:
            if(node in netData): #important, the node might not have connections
                neighbours = netData[node]
                for n in neighbours:
                    if(n not in secondNeigh and  n not in firstNeigh and n != "ME3"):
                        secondNeigh.append(n)

        print("\nME3's second neighbours:", secondNeigh  )
```

```
{'ENO1': ['TPI1', 'PKLR', 'PGK1', 'PGK2', 'GPI', 'LDHB', 'ENO2', 'LDHC', 'ENO3'], 'TPI1': ['ENO1', 'P

PKLR is directly connected to: ['ENO1', 'ENO3', 'ENO2', 'GPI', 'LDHB', 'LDHC', 'TPI1', 'PGK1', 'PGK2

ME3 is directly connected to: ['PKLR', 'LDHB', 'LDHC', 'GPI', 'TPI1']

ME3's second neighbours: ['ENO1', 'ENO3', 'ENO2', 'PGK1', 'PGK2']
```

CHAPTER 7

Practical 6

In this practical we will see how to define functions to reuse code, we will talk about the scope of variables and finally will see how to deal with files in Python.

7.1 Slides

The slides of the introduction can be found here: Intro

7.2 Functions

A function is a block of code that has a name and that performs a task. A function can be thought of as a box that gets an input and returns an output.

Why should we use functions? For a lot of reasons including:

1. *Reduce code duplication*: put in functions parts of code that are needed several times in the whole program so that you don't need to repeat the same code over and over again;
2. *Decompose a complex task*: make the code easier to write and understand by splitting the whole program in several easier functions;

both things improve code readability and make your code easier to understand.

The basic definition of a function is:

```
def function_name(input) :
    #code implementing the function
    ...
    ...
    return return_value
```

Functions are defined with the **def** keyword that proceeds the *function_name* and then a list of parameters is passed in the brackets. A colon **:** is used to end the line holding the definition of the function. The code implementing the

function is specified by using indentation. A function **might** or **might not** return a value. In the first case a **return** statement is used.

Example: Define a function that implements the sum of two integer lists (note that there is no check that the two lists actually contain integers and that they have the same size).

```
In [1]: def int_list_sum(l1,l2):
            """implements the sum of two lists of integers having the same size
            """
            ret =[]
            for i in range(len(l1)):
                ret.append(l1[i] + l2[i])
            return ret

        L1 = list(range(1,10))
        L2 = list(range(20,30))
        print("L1:", L1)
        print("L2:", L2)

        res = int_list_sum(L1,L2)

        print("L1+L2:", res)

        res = int_list_sum(L1,L1)

        print("L1+L1", res)
```

```
L1: [1, 2, 3, 4, 5, 6, 7, 8, 9]
L2: [20, 21, 22, 23, 24, 25, 26, 27, 28, 29]
L1+L2: [21, 23, 25, 27, 29, 31, 33, 35, 37]
L1+L1 [2, 4, 6, 8, 10, 12, 14, 16, 18]
```

Note that once the function has been defined, it can be called as many times as wanted with different input parameters. Moreover, **a function does not do anything until it is actually called.** A function can return **0** (in this case the return value would be "None"), **1** or **more** results. Notice also that collecting the results of a function is **not mandatory**.

Example: Let's write a function that, given a list of elements, prints only the even-placed ones without returning anything.

```
In [2]: def get_even_placed(myList):
            """returns the even placed elements of myList"""
            ret = [myList[i] for i in range(len(myList)) if i % 2 == 0]
            print(ret)

        L1 = ["hi", "there", "from","python","!"]
        L2 = list(range(13))

        print("L1:", L1)
        print("L2:", L2)

        print("even L1:")
        get_even_placed(L1)
        print("even L2:")
        get_even_placed(L2)
```

```
L1: ['hi', 'there', 'from', 'python', '!']
L2: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
even L1:
['hi', 'from', '!']
even L2:
[0, 2, 4, 6, 8, 10, 12]
```

Note that the function above is polymorphic (i.e. it works on several data types, provided that we can iterate through them).

Example: Let's write a function that, given a list of integers, returns the number of elements, the maximum and minimum.

```
In [3]: def get_info(myList):
            """returns len of myList, min and max value (assumes elements are integers)"""
            tmp = myList[:] #copy the input list
            tmp.sort()
            return len(tmp), tmp[0], tmp[-1] #return type is a tuple

        A = [7, 1, 125, 4, -1, 0]

        print("Original A:", A, "\n")
        result = get_info(A)
        print("Len:", result[0], "Min:", result[1], "Max:",result[2], "\n" )

        print("A now:", A)
```

```
Original A: [7, 1, 125, 4, -1, 0]

Len: 6 Min: -1 Max: 125

A now: [7, 1, 125, 4, -1, 0]
```

```
In [2]: def my_sum(myList):
            ret = 0
            for el in myList:
                ret += el # == ret = ret + el
            return ret

        A = [1,2,3,4,5,6]
        B = [7, 9, 4]
        s = my_sum(A)

        print("List A:", A)
        print("Sum:", s)
        s = my_sum(B)
        print("List B:", B)
        print("Sum:", s)
```

```
List A: [1, 2, 3, 4, 5, 6]
Sum: 21
List B: [7, 9, 4]
Sum: 20
```

Please note that the return value above is actually a tuple. Importantly enough, a function needs to be defined (i.e. its code has to be written) before it can actually be used.

```
In [5]: A = [1,2,3]
        my_sum(A)

        def my_sum(myList):
            ret = 0
            for el in myList:
                ret += el
            return ret
```

7.3 Namespace and variable scope

Namespaces are mappings from *names* to objects, or in other words places where names are associated to objects. Namespaces can be considered as the context. According to Python's reference a **scope** is a *textual region of a Python program, where a namespace is directly accessible*, which means that Python will look into that *namespace* to find the object associated to a name. Four **namespaces** are made available by Python:

```
1. **Local**: the innermost that contains local names (inside a function or a class);

2. **Enclosing**: the scope of the enclosing function,
it does not contain local nor global names (nested functions) ;

3. **Global**: contains the global names;

4. **Built-in**: contains all built in names
(e.g. print, if, while, for,...)
```

When one refers to a name, Python tries to find it in the current namespace, if it is not found it continues looking in the namespace that contains it until the built-in namespace is reached. If the name is not found there either, the Python interpreter will throw a **NameError** exception, meaning it cannot find the name. The order in which namespaces are considered is: Local, Enclosing, Global and Built-in (LEGB).

Consider the following example:

```
In [6]: def my_function():
            var = 1  #local variable
            print("Local:", var)
            b = "my string"
            print("Local:", b)

        var = 7 #global variable
        my_function()
        print("Global:", var)
        print(b)
```

```
Local: 1
Local: my string
Global: 7

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-6-da11a41b34ff> in <module>()
      8 my_function()
      9 print("Global:", var)
---> 10 print(b)
     11

NameError: name 'b' is not defined
```

Variables defined within a function can only be seen within the function. That is why variable b is defined only within the function. Variables defined outside all functions are **global** to the whole program. The namespace of the local variable is within the function my_function, while outside it the variable will have its global value.

And the following:

```
In [7]: def outer_function():
            var = 1 #outer

            def inner_function():
                var = 2 #inner
```

```
            print("Inner:", var)
            print("Inner:", B)

        inner_function()
        print("Outer:", var)


    var = 3 #global
    B = "This is B"
    outer_function()
    print("Global:", var)
    print("Global:", B)
```

```
Inner: 2
Inner: This is B
Outer: 1
Global: 3
Global: This is B
```

Note in particular that the variable B is global, therefore it is accessible everywhere and also inside the inner_function. On the contrary, the value of var defined within the inner_function is accessible only in the namespace defined by it, outside it will assume different values as shown in the example.

In a nutshell, remember the three simple rules seen in the lecture. Within a **def**:

```
1. Name assignments create local names by default;
2. Name references search the following four scopes in the order:
local, enclosing functions (if any), then global and finally built-in (LEGB)
3. Names declared in global and nonlocal statements map assigned names to
enclosing module and function scopes.
```

7.4 Argument passing

Arguments are the parameters and data we pass to functions. When passing arguments, there are three important things to bear in mind are:

1. Passing an argument is actually assigning an object to a local variable name;
2. Assigning an object to a variable name within a function **does not affect the caller**;
3. Changing a **mutable** object variable name within a function **affects the caller**

Consider the following examples:

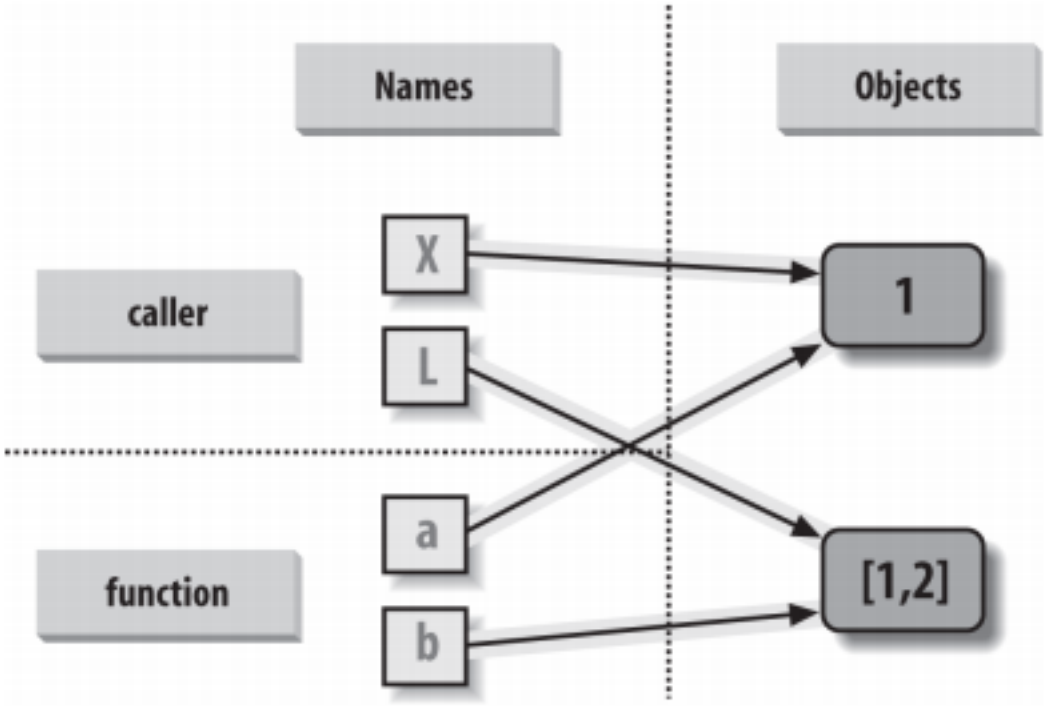
```
In [8]: """Assigning the argument does not affect the caller"""

    def my_f(x):
        x = "local value" #local
        print("Local: ", x)

    x = "global value" #global
    my_f(x)
    print("Global:", x)
    my_f(x)
```

```
Local:  local value
Global: global value
Local:  local value
```

```
In [9]: """Changing a mutable affects the caller"""

        def my_f(myList):
            myList[1] = "new value1"
            myList[3] = "new value2"
            print("Local: ", myList)

        myList = ["old value"]*4
        print("Global:", myList)
        my_f(myList)
        print("Global now: ", myList)
```

```
Global: ['old value', 'old value', 'old value', 'old value']
Local:  ['old value', 'new value1', 'old value', 'new value2']
Global now:  ['old value', 'new value1', 'old value', 'new value2']
```

Recall what seen in the lecture:



The behaviour above is because **immutable objects** are passed **by value** (therefore it is like making a copy), while **mutable objects** are passed **by reference** (therefore changing them effectively changes the original object).

To avoid making changes to a **mutable object** passed as parameter one needs to **explicitely make a copy** of it.

Consider the example seen before. **Example:** Let's write a function that, given a list of integers, returns the number of elements, the maximum and minimum.

```
In [10]: def get_info(myList):
             """returns len of myList, min and max value (assumes elements are integers)"""
             myList.sort()
             return len(myList), myList[0], myList[-1] #return type is a tuple


         def get_info_copy(myList):
             """returns len of myList, min and max value (assumes elements are integers)"""
```

```
            tmp = myList[:] #copy the input list!!!!
            tmp.sort()
            return len(tmp), tmp[0], tmp[-1] #return type is a tuple

        A = [7, 1, 125, 4, -1, 0]
        B = [70, 10, 1250, 40, -10, 0, 10]

        print("A:", A)
        result = get_info(A)
        print("Len:", result[0], "Min:", result[1], "Max:",result[2] )

        print("A now:", A) #whoops A is changed!!!

        print("\n###### With copy now ########")

        print("\nB:", B)
        result = get_info_copy(B)
        print("Len:", result[0], "Min:", result[1], "Max:",result[2] )

        print("B now:", B) #B is not changed!!!
```

```
A: [7, 1, 125, 4, -1, 0]
Len: 6 Min: -1 Max: 125
A now: [-1, 0, 1, 4, 7, 125]

###### With copy now ########

B: [70, 10, 1250, 40, -10, 0, 10]
Len: 7 Min: -10 Max: 1250
B now: [70, 10, 1250, 40, -10, 0, 10]
```

7.4.1 Positional arguments

Arguments can be passed to functions following the order in which they appear in the function definition.

Consider the following example:

```
In [11]: def print_parameters(a,b,c,d):
             print("1st param:", a)
             print("2nd param:", b)
             print("3rd param:", c)
             print("4th param:", d)


         print_parameters("A", "B", "C", "D")
```

```
1st param: A
2nd param: B
3rd param: C
4th param: D
```

7.4.2 Passing arguments by keyword

Given the name of an argument as specified in the definition of the function, parameters can be passed using the **name = value** syntax.

For example:

```
In [12]: def print_parameters(a,b,c,d):
             print("1st param:", a)
             print("2nd param:", b)
             print("3rd param:", c)
             print("4th param:", d)


         print_parameters(a = 1, c=3, d=4, b=2)
         print("\n###############\n")
         print_parameters("first","second",d="fourth",c="third")
         print("\n###############\n")


1st param: 1
2nd param: 2
3rd param: 3
4th param: 4

###############

1st param: first
2nd param: second
3rd param: third
4th param: fourth

###############
```

Arguments passed positionally and by name can be used at the same time, but parameters passed by name must always be to the left of those passed by name. The following code in fact is not accepted by the Python interpreter:

```
In [13]: def print_parameters(a,b,c,d):
             print("1st param:", a)
             print("2nd param:", b)
             print("3rd param:", c)
             print("4th param:", d)

         print_parameters(d="fourth",c="third", "first","second")
  File "<ipython-input-13-4991b2c31842>", line 7
    print_parameters(d="fourth",c="third", "first","second")
                                          ^
SyntaxError: positional argument follows keyword argument
```

7.4.3 Specifying default values

During the definition of a function it is possible to specify default values. The syntax is the following:

```
def my_function(par1 = val1, par2 = val2, par3 = val3):
```

Consider the following example:

```
In [1]: def print_parameters(a="defaultA", b="defaultB",c="defaultC"):
            print("a:",a)
            print("b:",b)
            print("c:",c)

        print_parameters("param_A")
        print("\n#################\n")
```

```
        print_parameters(b="PARAMETER_B")
        print("\n##################\n")
        print_parameters()
        print("\n##################\n")
        print_parameters(c="PARAMETER_C", b="PAR_B")
```

```
a: param_A
b: defaultB
c: defaultC

##################

a: defaultA
b: PARAMETER_B
c: defaultC

##################

a: defaultA
b: defaultB
c: defaultC

##################

a: defaultA
b: PAR_B
c: PARAMETER_C
```

7.5 File input and output

Files are non volatile data stored at specific locations in memory. To read or write from/to a file, one first needs to create a reference to the file. This is called the **handle** to the file.

Recall from the lecture that Python provides programmers with the following functions and methods:

Result	Built-in function	Meaning
`file`	`open(str, [str])`	Get a handle to a file

Result	Method	Meaning
`str`	`file.read()`	Read all the file as a single string
`list of str`	`file.readlines()`	Read all lines of the file as a list of strings
`str`	`file.readline()`	Read one line of the file as a string
`None`	`file.write(str)`	Write one string to the file
`None`	`file.close()`	`Close the file` (i.e. flushes changes to disk)

To read or write to a file one first needs to open it with the python's built-in function **open** that returns a handler to the file. The handler can then be used to read or write the content of the file.

Example: Download the file textFile.txt to your favourite location and read it from python, printing out all its content. **NOTE:** the path of your file might be different from mine.

```
In [15]: fh = open("file_samples/textFile.txt", "r") #read-only mode

         print("--- File content ---")

         for line in fh:
             print(line, end ="") #no need to add a new line

         fh.close()

         print("\n--- File closed ---")
```

```
--- File content ---
Hi everybody,
This is my first file
and it contains a total of
four lines!
--- File closed ---
```

The usual interaction with files is done in three steps: **open the file**, **read or write its content**, **close the file**.

7.5.1 Opening a file

As said, a file needs to be opened before working with it. This can be done by using the function **open**.

The standard syntax is:

```
file_handle = open("file_name", "file_mode")
```

where `file_name` is a string representing the path to reach the file (either absolute or relative to the .py module we are writing) and `file_mode` is the string representing the opening mode of the file. `file_mode` can be:

1. `"r"` which is the default and opens the file in read only mode;
2. `"w"` write only mode. **Note that this would overwrite an existing file**;
3. `"a"` append mode (i.e. add information at the end of an existing file or create a new file)
4. `"t"` text mode. The file will be a text file (this is the default).
5. `"b"` binary mode. The file will not be a textual file.
6. `"x"` exclusive creation. This will fail if the file already exists.
7. `"+"` update mode. The file will be opened with read and write permissions.

Opening modes can be combined (ex. `"ra"` would read and append, `"wb"` would write (and overwrite) a binary file, `"r+b"` read and write a binary file).

7.5.2 Closing a file

A opened file always needs to be closed. The basic syntax is:

```
file_handle.close()
```

this will free up resources and will not allow any more operations on the same file.

The following code will in fact crash, because the file has been closed and therefore no more operations can be performed on it.

```
In [16]: fh = open("file_samples/textFile.txt", "r") #read-only mode

         print("--- File content ---")

         for line in fh:
             print(line, end ="") #no need to add a new line

         fh.close()

         print("\n--- File closed ---")

         print(fh.readline())
```

```
--- File content ---
Hi everybody,
This is my first file
and it contains a total of
four lines!
--- File closed ---
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-16-0367c74d60af> in <module>()
     10 print("\n--- File closed ---")
     11
---> 12 print(fh.readline())

ValueError: I/O operation on closed file.
```

An alternative way to open and implicitly close the file is by using the **with** keyword. The basic syntax is:

```
with open("file") as f:
    #do some stuff
    #until done with it
```

The previous example would become:

```
In [17]: print("--- File content ---")

         with open("file_samples/textFile.txt", "r") as f:
             for line in f:
                 print(line, end ="") #no need to add a new line

         print("\n--- File closed ---")
```

```
--- File content ---
Hi everybody,
This is my first file
and it contains a total of
four lines!
--- File closed ---
```

7.5.3 Reading from a file

There are several ways for reading a file that has been opened in *read* mode having file handle `fh`:

1. `content = fh.read()` reads the whole file in the content string. Good for small and not structured files.
2. `line = fh.readline()` reads the file one line at a time storing it in the string `line`
3. `lines = fh.readlines()` reads all the lines of the file storing them as a list `lines`

4. using the iterator:

```
for line in f:
#process the information
```

   which is the most convenient way for big files.

Let's see some examples. Note that to restart reading a file from the beginning we need to close it and open it gain.

```
In [18]: fh = open("file_samples/textFile.txt", "r") #read-only mode

         content = fh.read()
         print("--- Mode1 (the whole file in a string) ---")
         print(content)
         fh.close()
         print("")
         print("--- Mode2 (line by line) ---")
         with open("file_samples/textFile.txt","r") as f:
             print("Line1: ", f.readline(), end = "")
             print("Line2: ", f.readline(), end = "")

         print("")
         print("--- Mode3 (all lines as a list) ---")
         with open("file_samples/textFile.txt","r") as f:
             print(f.readlines())

         print("")
         print("--- Mode4 (as a stream) ---")
         with open("file_samples/textFile.txt","r") as f:
             for line in f:
                 print(line, end = "")
```

```
--- Mode1 (the whole file in a string) ---
Hi everybody,
This is my first file
and it contains a total of
four lines!

--- Mode2 (line by line) ---
Line1:  Hi everybody,
Line2:  This is my first file

--- Mode3 (all lines as a list) ---
['Hi everybody,\n', 'This is my first file\n', 'and it contains a total of\n', 'four lines!']

--- Mode4 (as a stream) ---
Hi everybody,
This is my first file
and it contains a total of
four lines!
```

7.5.4 Writing to a file

To write some data to a file it must first be opened in write mode (i.e. "w") or append mode ("a", "+"). The actual writing can be performed by the method **write** that has the following basic syntax:

```
file_handle.write(data_to_be_written)
```

To make sure all data is written to the file, we must remember to close the file after writing on it.

Example: Given the matrix represented as list of lists, M = [[1,2,3], [4,5,6], [7,8,9]] let's write it on a file *my_matrix.txt*

$$\text{as } \begin{matrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{matrix}$$

```
In [19]: M =  [[1,2,3], [4,5,6], [7,8,9]]

         with open("file_samples/my_matrix.txt", "w") as f:
             for line in M:
                 str_line = [str(x) for x in line] #to make this "joinable"
                 f.write(" ".join(str_line))
                 f.write("\n")

         #no need to put the close because we used with
```

7.5.5 String formatting with format

Strings can be formatted with the method **format**. A lot of information on the use of format can be found here[33].

Format can be used to add values to a string in specific placeholders (normally defined with the syntax **{}**) or to format values according to the user specifications (e.g. number of decimal places for floating point numbers).

In the following we will see some simple examples.

```
In [20]: #simple empty placeholders
         print("I like {} more than {}.\n".format("python", "java"))

         #indexed placeholders, note order
         print("I like {0} more than {1} or {2}.\n".format("python", "java", "C++"))
         print("I like {2} more than {1} or {0}.\n".format("python", "java", "C++"))

         #indexed and named placeholders
         print("I like {1} more than {c} or {0}.\n".format("python", "java", c="C++"))

         #with type specification
         import math
         print("The square root of {0} is {1:f}.\n".format(2, math.sqrt(2)))

         #with type and format specification (NOTE: {.2f})
         print("The square root of {0} is {1:.2f}.\n".format(2, math.sqrt(2)))

         #spacing data properly
         print("{:2s}|{:5}|{:6}".format("N","sqrt","square"))
         for i in range(0,20):
             print("{:2d}|{:5.3f}|{:6d}".format(i,math.sqrt(i),i*i))
```

```
I like python more than java.

I like python more than java or C++.

I like C++ more than java or python.

I like java more than C++ or python.

The square root of 2 is 1.414214.

The square root of 2 is 1.41.
```

[33] https://docs.python.org/3/library/string.html#format-string-syntax

```
N |sqrt |square
 0|0.000|     0
 1|1.000|     1
 2|1.414|     4
 3|1.732|     9
 4|2.000|    16
 5|2.236|    25
 6|2.449|    36
 7|2.646|    49
 8|2.828|    64
 9|3.000|    81
10|3.162|   100
11|3.317|   121
12|3.464|   144
13|3.606|   169
14|3.742|   196
15|3.873|   225
16|4.000|   256
17|4.123|   289
18|4.243|   324
19|4.359|   361
```

7.6 Exercises

1. Implement a function that takes in input a string representing a DNA sequence and computes its reverse-complement. Take care to reverse complement any character other than (A,T,C,G,a,t,c,g) to N. The function should preserve the case of each letter (i.e. A becomes T, but a becomes t). For simplicity all bases that do not represent nucleotides are converted to a capital N. **Hint: create a dictionary revDict with bases as keys and their complements as values**. Ex. revDict = {"A" : "T" , "a" : "t", ... }.

   (a) Apply the function to the DNA sequence "ATTACATATCATACTATCGCNTTCTAAATA"

   (b) Apply the function to the DNA sequence "acaTTACAtagataATACTaccataGCNTTCTAAATA"

   (c) Apply the function to the DNA sequence "TTTTACCKKKAKTUUUITTTARRRRRAIUTYYA"

   (d) Check that the reverse complement of the reverse complement of the sequence in 1. is exactly as the original sequence.

Show/Hide Solution

```
In [5]: def reverse_complement(DNA):
            """the function reverse complements a string of DNA"""
            revDict = {"A" : "T", "a" : "t",
                       "C" : "G", "c" : "g",
                       "G" : "C", "g" : "c",
                       "T" : "A", "t" : "a"
                      }
            result = ""
            for base in DNA:
                ### dict.get(val, default) : returns default if val not in
                result = revDict.get(base, "N") + result

            return result

        # Second version (equivalent)
        def reverse_complement2(DNA):
```

```
        """the function reverse complements a string of DNA"""
        revDict = {"A" : "T", "a" : "t",
                   "C" : "G", "c" : "g",
                   "G" : "C", "g" : "c",
                   "T" : "A", "t" : "a"
                  }

        result = ""
        DNAlist = list(DNA)
        DNAlist.reverse()
        revComp =[revDict.get(x, "N") for x in DNAlist]
        result = "".join(revComp)
        return result

    # Third version (equivalent)
    def reverse_complement3(DNA):
        """the function reverse complements a string of DNA"""
        k = "AaTtCcGg"
        v = "TtAaGgCc"
        revDict  = dict()
        for i in range(0,len(k)):
            revDict[k[i]] = v[i]
        result = ""
        DNAlist = list(DNA)
        DNAlist.reverse()
        revComp =[revDict.get(x, "N") for x in DNAlist]
        result = "".join(revComp)
        return result


    S1 = "ATTACATATCATACTATCGCNTTCTAAATA"
    S2 = "acaTTACAtagataATACTaccataGCNTTCTAAATA"
    S3 = "TTTTACCKKKAKTUUUITTTARRRRRAIUTYYA"

    print("S1 {}".format(S1) )
    print("rc S1", reverse_complement(S1))

    print("\nS2   ", S2)
    print("rc S2", reverse_complement(S2))

    print("\nS3   ", S3)
    print("rc S3", reverse_complement(S3))

    print("\nrev_comp(rev_comp(S1)): {}".format(reverse_complement(reverse_complement(S1))))
    print("\nrev_comp(rev_comp(S1) == S1?)", reverse_complement(reverse_complement(S1)) == S1)
```

```
S1 ATTACATATCATACTATCGCNTTCTAAATA
rc S1 TATTTAGAANGCGATAGTATGATATGTAAT

S2    acaTTACAtagataATACTaccataGCNTTCTAAATA
rc S2 TATTTAGAANGCtatggtAGTATtatctaTGTAAtgt

S3    TTTTACCKKKAKTUUUITTTARRRRRAIUTYYA
rc S3 TNNANNTNNNNNTAAANNNNANTNNNGGTAAAA

rev_comp(rev_comp(S1)): ATTACATATCATACTATCGCNTTCTAAATA

rev_comp(rev_comp(S1) == S1?) True
```

2. Write the following python functions and test them with some parameters of your choice:

(a) *getDivisors*: the function has a positive integer as parameter and returns a list of all the positive divisors of the integer in input (excluding the number itself). Example: `getDivisors(6) --> [1,2,3]`

(b) *checkSum*: the function has a list and an integer as parameters and returns True if the sum of all elements in the list equals the integer, False otherwise. Example: `checkSum([1,2,3], 6) --> True`, `checkSum([1,2,3],1) --> False`.

(c) *checkPerfect*: the function gets an integer as parameter and returns True if the integer is a perfect number[34], False otherwise. A number is perfect if all its divisors (excluding itself) sum to its value. Example: `checkPerfect(6) --> True` because 1+2+3 = 6. Hint: use the functions implemented before.

Use the three implemented functions to write a fourth function:

getFirstNperfects: the function gets an integer N as parameter and returns a dictionary with the first N perfect numbers. The key of the dictionary is the perfect number, while the value of the dictionary is the list of its divisors. Example: `getFirstNperfects(1) --> {6 : [1,2,3]}`

Get and print the first 4 perfect numbers and finally test if 33550336 is a perfect number.

WARNING: do not try to find more than 4 perfect numbers as it might take a while!!!

Show/Hide Solution

```
In [8]: def getDivisors(intVal):
            """returns the integer divisors of intVal"""
            ret = [x for x in range(1,intVal//2 + 1) if intVal % x == 0]
            #OR:
            #for i in range(1,intVal//2+1):
            #    if(intVal % i == 0):
            #        ret.append(i)
            return ret

        def checkSum(intList, intVal):
            """checks if the sum of elements in intList equals intVal"""
            s = 0
            for x in intList:
                s += x
            return s == intVal

        def checkPerfect(intVal):
            """checks if intVal is a perfect number"""
            divisors = getDivisors(intVal)
            return checkSum(divisors,intVal)

        def getFirstNPerfects(N):
            """Finds the first N perfect numbers"""
            i = 0
            val = 2
            ret = {}
            while(i<N):
                if(checkPerfect(val)):
                    i+=1
                    ret[val] = getDivisors(val)
                    val += 1
                else:
                    val += 1


            return ret
```

[34] https://en.wikipedia.org/wiki/Perfect_number

```
        perfects = getFirstNPerfects(4) #do not go higher than 4, or it might take a while
        perKeys = list(perfects.keys())
        perKeys.sort()
        for p in perKeys:
            print(p, " = ", "+".join([str(x) for x in perfects[p]]))

        print("Is 33550336 a perfect number?", checkPerfect(33550336))
```

```
6  =  1+2+3
28  =  1+2+4+7+14
496  =  1+2+4+8+16+31+62+124+248
8128  =  1+2+4+8+16+32+64+127+254+508+1016+2032+4064
Is 33550336 a perfect number? True
```

3. Blast[35] is a well known tool to perform sequence alignment between a pool of query sequences and a pool of subject sequences. Among the other formats, it can produce an text output that is tab separated (`\t`) capturing user specified output. Comments in the file are written in lines starting with an hash key ("#"). A sample blast output file is blast_sample.tsv, please download it and spend some time to look at it. The meaning of all columns is specified in the file header:

```
# Fields: query id, subject id, query length, % identity, alignment length,
identical, gap opens, q. start, q. end, s. start, s. end, evalue
```

   Write a python program with:

   (a) A function (*readBlast*) that reads in the blast .tsv file (storing each line in a list) ignoring comment lines;

   (b) A function (*filterBlast*) that gets a string in input representing a blast alignment and filters it according to the user specified parameters below. It should return True if all filters are passed, false otherwise:

      i. % identity > identity_threshold (default 0%)

      ii. evalue < evalue_thrshold (default 0.5)

      iii. alignment length / query length > align_threshold (default 0)

The program should report how many entries out of the total passed the filter.

Test several combinations of filters like:

```
1. identity_theshold = 97, all others default
2. evalue_threshold = 0.5, all others default
3. align_threshold = 0.9, all others default
4. align_threshold = 0.9, identity_threshold = 97
```

Show/Hide Solution

In [22]:

```
        def filterBlast(seq, id_thr = 0, eval_thr = 1, al_thr = 0):
            """filters an alignment represented as a string. Returns True if filters OK, False othe
            infos = seq.split("\t")
            return float(infos[3]) > id_thr and float(infos[-1]) < eval_thr and float(infos[4])/int

        def readBlast(inFile):
            """Reads the blast .tsv file and returns a list"""
            fh = open(inFile, "r")
            alignments = []
```

[35] https://www.ncbi.nlm.nih.gov/pubmed/2231712

```
            for line in fh:
                if not line.startswith("#"):
                    alignments.append(line)
            return alignments

        def filterAndCount(alignList, id_thr = 0, eval_thr = 1, al_thr = 0):
            """filters alignments and counts the ones passing the filter"""
            passed = [x for x in alignList if filterBlast(x, id_thr, eval_thr, al_thr)]
            print("{} out of {} aligns passed filter (id_thr:{},eval_thr:{},al_thr:{})".format(
                len(passed), len(alignList), id_thr,eval_thr,al_thr))

        blastF = "file_samples/blast_sample.tsv"

        myAligns = readBlast(blastF)

        filterAndCount(myAligns, id_thr=97)
        filterAndCount(myAligns, eval_thr=1)
        filterAndCount(myAligns, al_thr=0.9)
        filterAndCount(myAligns, al_thr=0.9, id_thr=97)
```

```
8390 out of 90790 aligns passed filter (id_thr:97,eval_thr:1,al_thr:0)
90780 out of 90790 aligns passed filter (id_thr:0,eval_thr:1,al_thr:0)
27711 out of 90790 aligns passed filter (id_thr:0,eval_thr:1,al_thr:0.9)
804 out of 90790 aligns passed filter (id_thr:97,eval_thr:1,al_thr:0.9)
```

4. Fasta is a format[36] representing nucleotide or peptide sequences. Each entry of a fasta file starts with a ">" followed by the identifier of the entry (the header of the sequence). All the lines following a header represent the sequence belonging to that identifier. Here you can find an example of fasta file: contigs82.fasta. Please download it and have a look at its content.

Write a python program that reads a fasta file and prints to screen the identifier of the sequence and the frequency of all the characters in the sequence (note that sequences might contain all IUPAC codes in case of SNPs). Hint: use a dictionary.

Sample output:

```
MDC020656.85 :
     N has freq 0.052
     T has freq 0.273
     A has freq 0.308
     G has freq 0.203
     C has freq 0.163
MDC001115.177 :
     N has freq 0.018
     G has freq 0.157
     A has freq 0.302
     T has freq 0.316
     C has freq 0.207
     ...
```

Show/Hide Solution

```
In [23]: def countFrequency(seq):
             """gets a sequence in input and returns
             a dictionary with bases as keys and frequence as value"""
             bases = {}
             for b in seq:
```

[36] https://en.wikipedia.org/wiki/FASTA_format

```
                if b not in bases:
                    bases[b] = 1
                else:
                    bases[b] += 1

            for b in bases:
                bases[b] = bases[b]/len(seq)
            return bases

        def printData(ident, freqDict):
            """get the identifier and a dictionary with all frequencies
            and prints both information on the screen"""
            print(ident,":")
            for f in freqDict:
                print("\t {} has freq {:.3f}".format(f,freqDict[f]))

        def processFasta(file):
            header = ""
            seq = ""
            with open(file, "r") as f:
                for line in f:
                    line = line.strip()
                    if(line.startswith(">")):
                        if(len(header) == 0 ):
                            #first entry:
                            header = line[1:]
                        else:
                            #this is a new entry
                            freqData = countFrequency(seq)
                            printData(header,freqData)
                            seq = ""
                            header = line[1:]
                    else:
                        seq +=line
            #processing the final entry
            freqData = countFrequency(seq)
            printData(header,freqData)


        inFasta = "file_samples/contigs82.fasta"
        processFasta(inFasta)
```

```
MDC020656.85 :
        C has freq 0.163
        T has freq 0.273
        A has freq 0.308
        N has freq 0.052
        G has freq 0.203
MDC001115.177 :
        C has freq 0.207
        T has freq 0.316
        A has freq 0.302
        N has freq 0.018
        G has freq 0.157
MDC013284.379 :
        C has freq 0.171
        T has freq 0.321
        A has freq 0.309
        N has freq 0.005
```

```
        G has freq 0.194
MDC018185.243 :
        S has freq 0.000
        W has freq 0.000
        K has freq 0.000
        G has freq 0.181
        N has freq 0.001
        M has freq 0.000
        R has freq 0.001
        C has freq 0.204
        T has freq 0.304
        A has freq 0.307
        Y has freq 0.001
MDC018185.241 :
        S has freq 0.000
        W has freq 0.000
        K has freq 0.001
        G has freq 0.183
        N has freq 0.006
        M has freq 0.001
        R has freq 0.001
        C has freq 0.168
        T has freq 0.304
        A has freq 0.335
        Y has freq 0.002
MDC004527.213 :
        C has freq 0.273
        T has freq 0.332
        A has freq 0.227
        N has freq 0.013
        G has freq 0.155
MDC003661.174 :
        C has freq 0.213
        T has freq 0.301
        A has freq 0.246
        N has freq 0.007
        G has freq 0.232
MDC012176.157 :
        C has freq 0.240
        T has freq 0.253
        A has freq 0.343
        N has freq 0.001
        G has freq 0.163
MDC001204.812 :
        C has freq 0.221
        T has freq 0.298
        A has freq 0.293
        N has freq 0.000
        G has freq 0.187
MDC001204.810 :
        C has freq 0.182
        T has freq 0.340
        A has freq 0.297
        N has freq 0.017
        G has freq 0.164
MDC004389.256 :
        C has freq 0.277
        T has freq 0.203
        A has freq 0.285
```

```
         N has freq 0.028
         G has freq 0.207
MDC024257.15 :
         C has freq 0.144
         T has freq 0.331
         A has freq 0.319
         N has freq 0.022
         G has freq 0.184
MDC018297.229 :
         C has freq 0.181
         T has freq 0.305
         A has freq 0.275
         N has freq 0.031
         G has freq 0.209
MDC001802.364 :
         C has freq 0.203
         T has freq 0.328
         A has freq 0.252
         N has freq 0.037
         G has freq 0.180
MDC016621.241 :
         C has freq 0.206
         T has freq 0.262
         A has freq 0.302
         N has freq 0.063
         G has freq 0.168
MDC014057.243 :
         C has freq 0.263
         T has freq 0.212
         A has freq 0.291
         N has freq 0.016
         G has freq 0.217
MDC021015.302 :
         C has freq 0.143
         T has freq 0.223
         A has freq 0.302
         N has freq 0.081
         G has freq 0.251
MDC018185.242 :
         S has freq 0.000
         W has freq 0.000
         K has freq 0.000
         G has freq 0.189
         N has freq 0.000
         M has freq 0.001
         R has freq 0.002
         C has freq 0.200
         T has freq 0.289
         A has freq 0.316
         Y has freq 0.001
MDC051782.000 :
         N has freq 0.044
         W has freq 0.001
         K has freq 0.001
         S has freq 0.001
         G has freq 0.177
         M has freq 0.000
         R has freq 0.002
         C has freq 0.180
```

```
         T has freq 0.306
         A has freq 0.285
         Y has freq 0.002
MDC017187.314 :
         C has freq 0.206
         T has freq 0.276
         A has freq 0.289
         N has freq 0.003
         G has freq 0.225
MDC017187.311 :
         S has freq 0.000
         W has freq 0.001
         K has freq 0.000
         G has freq 0.185
         N has freq 0.068
         M has freq 0.000
         R has freq 0.002
         C has freq 0.179
         T has freq 0.315
         A has freq 0.248
         Y has freq 0.001
MDC012865.410 :
         C has freq 0.162
         T has freq 0.328
         A has freq 0.289
         N has freq 0.003
         G has freq 0.218
MDC000427.83 :
         C has freq 0.165
         T has freq 0.274
         A has freq 0.392
         N has freq 0.010
         G has freq 0.159
MDC017187.319 :
         C has freq 0.181
         T has freq 0.331
         A has freq 0.282
         N has freq 0.006
         G has freq 0.200
MDC017187.318 :
         K has freq 0.000
         N has freq 0.001
         G has freq 0.232
         C has freq 0.187
         T has freq 0.305
         A has freq 0.275
MDC004081.319 :
         C has freq 0.106
         T has freq 0.197
         A has freq 0.172
         G has freq 0.098
         N has freq 0.428
MDC021913.275 :
         C has freq 0.155
         T has freq 0.169
         A has freq 0.199
         N has freq 0.334
         G has freq 0.143
MDC015147.205 :
```

```
        C has freq 0.154
        T has freq 0.350
        A has freq 0.298
        N has freq 0.002
        G has freq 0.196
MDC000038.355 :
        S has freq 0.000
        W has freq 0.000
        K has freq 0.000
        G has freq 0.158
        N has freq 0.025
        M has freq 0.000
        R has freq 0.000
        C has freq 0.170
        T has freq 0.315
        A has freq 0.331
        Y has freq 0.000
MDC016032.95 :
        S has freq 0.001
        W has freq 0.001
        K has freq 0.002
        G has freq 0.222
        N has freq 0.003
        R has freq 0.002
        C has freq 0.142
        T has freq 0.286
        A has freq 0.342
        Y has freq 0.001
MDC052568.000 :
        N has freq 0.008
        W has freq 0.001
        K has freq 0.000
        S has freq 0.001
        G has freq 0.178
        M has freq 0.001
        R has freq 0.002
        C has freq 0.207
        T has freq 0.302
        A has freq 0.300
        Y has freq 0.001
MDC008119.414 :
        C has freq 0.245
        T has freq 0.316
        A has freq 0.209
        N has freq 0.001
        G has freq 0.229
MDC026201.7 :
        C has freq 0.247
        T has freq 0.250
        A has freq 0.313
        G has freq 0.191
MDC003995.601 :
        S has freq 0.000
        W has freq 0.001
        K has freq 0.000
        G has freq 0.167
        N has freq 0.006
        M has freq 0.001
        R has freq 0.001
```

```
        C has freq 0.166
        T has freq 0.334
        A has freq 0.323
        Y has freq 0.001
MDC009211.561 :
        C has freq 0.237
        T has freq 0.286
        A has freq 0.306
        N has freq 0.001
        G has freq 0.171
MDC009211.567 :
        C has freq 0.167
        T has freq 0.327
        A has freq 0.318
        N has freq 0.006
        G has freq 0.181
MDC054294.001 :
        W has freq 0.001
        K has freq 0.001
        S has freq 0.000
        G has freq 0.186
        M has freq 0.001
        R has freq 0.002
        C has freq 0.186
        T has freq 0.318
        A has freq 0.303
        Y has freq 0.002
MDC004364.265 :
        S has freq 0.000
        W has freq 0.000
        K has freq 0.000
        G has freq 0.198
        N has freq 0.007
        R has freq 0.001
        C has freq 0.170
        T has freq 0.302
        A has freq 0.323
        Y has freq 0.001
MDC002360.219 :
        C has freq 0.204
        T has freq 0.303
        A has freq 0.316
        N has freq 0.003
        G has freq 0.173
MDC003408.117 :
        C has freq 0.215
        T has freq 0.247
        A has freq 0.329
        N has freq 0.017
        G has freq 0.191
MDC015155.172 :
        C has freq 0.202
        T has freq 0.277
        A has freq 0.272
        N has freq 0.003
        G has freq 0.247
MDC053310.000 :
        W has freq 0.001
        K has freq 0.001
```

```
        S has freq 0.000
        G has freq 0.185
        M has freq 0.001
        R has freq 0.003
        C has freq 0.167
        T has freq 0.310
        A has freq 0.330
        Y has freq 0.002
MDC019140.398 :
        C has freq 0.163
        T has freq 0.302
        A has freq 0.352
        N has freq 0.005
        G has freq 0.178
MDC019140.399 :
        C has freq 0.165
        T has freq 0.346
        A has freq 0.301
        N has freq 0.001
        G has freq 0.187
MDC011390.337 :
        G has freq 0.177
        N has freq 0.046
        R has freq 0.000
        C has freq 0.228
        T has freq 0.298
        A has freq 0.251
        Y has freq 0.000
MDC007154.375 :
        C has freq 0.197
        T has freq 0.292
        A has freq 0.307
        N has freq 0.003
        G has freq 0.201
MDC006346.716 :
        C has freq 0.199
        T has freq 0.290
        A has freq 0.296
        N has freq 0.024
        G has freq 0.191
MDC010588.505 :
        C has freq 0.257
        T has freq 0.237
        A has freq 0.305
        N has freq 0.006
        G has freq 0.195
MDC002519.240 :
        N has freq 0.001
        G has freq 0.188
        M has freq 0.000
        R has freq 0.000
        C has freq 0.178
        T has freq 0.321
        A has freq 0.312
        Y has freq 0.000
MDC031322.5 :
        S has freq 0.000
        W has freq 0.000
        K has freq 0.000
```

```
        G has freq 0.202
        N has freq 0.002
        M has freq 0.001
        R has freq 0.002
        C has freq 0.149
        T has freq 0.299
        A has freq 0.342
        Y has freq 0.002
MDC010588.502 :
        C has freq 0.184
        T has freq 0.311
        A has freq 0.282
        N has freq 0.009
        G has freq 0.215
MDC006346.711 :
        G has freq 0.187
        N has freq 0.006
        M has freq 0.000
        C has freq 0.191
        T has freq 0.295
        A has freq 0.321
MDC011551.182 :
        R has freq 0.001
        W has freq 0.000
        K has freq 0.000
        N has freq 0.007
        G has freq 0.190
        M has freq 0.000
        Y has freq 0.001
        C has freq 0.189
        T has freq 0.297
        A has freq 0.313
        S has freq 0.000
MDC002717.156 :
        C has freq 0.219
        T has freq 0.284
        A has freq 0.298
        N has freq 0.003
        G has freq 0.196
MDC006346.719 :
        C has freq 0.183
        T has freq 0.306
        A has freq 0.286
        N has freq 0.006
        G has freq 0.219
MDC007838.447 :
        C has freq 0.200
        T has freq 0.299
        A has freq 0.326
        N has freq 0.009
        G has freq 0.166
MDC007018.186 :
        C has freq 0.160
        T has freq 0.262
        A has freq 0.226
        N has freq 0.151
        G has freq 0.202
MDC017873.233 :
        C has freq 0.133
```

```
        T has freq 0.231
        A has freq 0.262
        N has freq 0.235
        G has freq 0.139
MDC016296.138 :
        C has freq 0.172
        T has freq 0.346
        A has freq 0.277
        N has freq 0.016
        G has freq 0.188
MDC019067.226 :
        C has freq 0.144
        T has freq 0.376
        A has freq 0.293
        N has freq 0.007
        G has freq 0.181
MDC036568.1 :
        C has freq 0.154
        T has freq 0.348
        A has freq 0.331
        N has freq 0.006
        G has freq 0.161
MDC014019.318 :
        C has freq 0.249
        T has freq 0.319
        A has freq 0.221
        N has freq 0.002
        G has freq 0.209
MDC007995.528 :
        C has freq 0.166
        T has freq 0.340
        A has freq 0.305
        N has freq 0.009
        G has freq 0.181
MDC026961.60 :
        C has freq 0.201
        T has freq 0.288
        A has freq 0.300
        N has freq 0.005
        G has freq 0.206
MDC013443.168 :
        C has freq 0.203
        T has freq 0.330
        A has freq 0.289
        N has freq 0.005
        G has freq 0.173
MDC022800.298 :
        C has freq 0.131
        T has freq 0.334
        A has freq 0.323
        N has freq 0.008
        G has freq 0.203
MDC021558.159 :
        C has freq 0.176
        T has freq 0.334
        A has freq 0.298
        N has freq 0.002
        G has freq 0.190
MDC002479.192 :
```

```
        C has freq 0.151
        T has freq 0.304
        A has freq 0.305
        N has freq 0.009
        G has freq 0.232
MDC010751.301 :
        C has freq 0.172
        T has freq 0.291
        A has freq 0.318
        N has freq 0.006
        G has freq 0.214
MDC000219.190 :
        S has freq 0.000
        G has freq 0.168
        N has freq 0.003
        M has freq 0.000
        R has freq 0.000
        C has freq 0.160
        T has freq 0.335
        A has freq 0.332
        Y has freq 0.001
MDC020963.161 :
        C has freq 0.248
        T has freq 0.315
        A has freq 0.287
        N has freq 0.002
        G has freq 0.148
MDC020963.162 :
        W has freq 0.000
        G has freq 0.207
        N has freq 0.005
        R has freq 0.000
        C has freq 0.214
        T has freq 0.265
        A has freq 0.309
        S has freq 0.000
MDC051637.000 :
        N has freq 0.010
        K has freq 0.001
        Y has freq 0.005
        G has freq 0.169
        M has freq 0.001
        R has freq 0.003
        C has freq 0.175
        T has freq 0.328
        A has freq 0.307
        W has freq 0.001
        S has freq 0.000
MDC005174.220 :
        C has freq 0.198
        T has freq 0.283
        A has freq 0.311
        N has freq 0.022
        G has freq 0.186
MDC040033.7 :
        C has freq 0.195
        T has freq 0.301
        A has freq 0.297
        N has freq 0.003
```

```
         G has freq 0.204
MDC019674.147 :
         S has freq 0.000
         W has freq 0.000
         G has freq 0.167
         N has freq 0.004
         R has freq 0.000
         C has freq 0.137
         T has freq 0.366
         A has freq 0.324
         Y has freq 0.001
MDC010450.877 :
         C has freq 0.190
         T has freq 0.285
         A has freq 0.293
         N has freq 0.058
         G has freq 0.174
MDC007097.457 :
         C has freq 0.126
         T has freq 0.299
         A has freq 0.289
         N has freq 0.070
         G has freq 0.217
MDC016278.70 :
         C has freq 0.183
         T has freq 0.303
         A has freq 0.277
         N has freq 0.084
         G has freq 0.153
MDC013723.254 :
         C has freq 0.212
         T has freq 0.329
         A has freq 0.279
         N has freq 0.001
         G has freq 0.179
MDC002838.179 :
         C has freq 0.163
         T has freq 0.297
         A has freq 0.359
         N has freq 0.013
         G has freq 0.167
MDC009771.217 :
         C has freq 0.185
         T has freq 0.335
         A has freq 0.315
         N has freq 0.001
         G has freq 0.163
```

5. Write a python program that reads two files. The first is a one column text file (contig_ids.txt) with the identifiers of some contigs that are present in the second file, which is a fasta formatted file (contigs82.fasta). The program will write on a third, fasta formatted file (e.g. filtered_contigs.fasta) only those entries in *contigs82.fasta* having identifier in *contig_ids.txt*.

Show/Hide Solution

```
In [24]: def readIDS(f):
             """reads a one column file in and stores
             the ids in a dictionary that is returned at the end"""
             ret = dict()
             with open(f, "r") as file:
```

```
                for line in file:
                    line = line.strip()
                    if(line not in ret):
                        ret[line] = 1
            return ret

        def filterFasta(inF, outF, ids2keep):
            oF = open(outF, "w")

            outputME = False
            with open(inF, "r") as file:
                for line in file:
                    line = line.strip()
                    if(line.startswith(">")):
                        #this is the header
                        if(line[1:] in ids2keep):
                            oF.write(line +"\n")
                            outputME = True
                            print("Writing contig ", line[1:])
                        else:
                            outputME = False
                    else:
                        if(outputME):
                            oF.write(line +"\n")

            oF.close()


        idsFile = "file_samples/contig_ids.txt"
        inFasta = "file_samples/contigs82.fasta"
        outFasta = "file_samples/filtered_contigs.fasta"

        ids = readIDS(idsFile)
        filterFasta(inFasta,outFasta, ids)
```

```
Writing contig  MDC001115.177
Writing contig  MDC013284.379
Writing contig  MDC018185.243
Writing contig  MDC018185.241
Writing contig  MDC004527.213
Writing contig  MDC012176.157
Writing contig  MDC001204.810
Writing contig  MDC004389.256
Writing contig  MDC018297.229
Writing contig  MDC001802.364
Writing contig  MDC014057.243
Writing contig  MDC021015.302
Writing contig  MDC017187.314
Writing contig  MDC012865.410
Writing contig  MDC000427.83
Writing contig  MDC017187.319
Writing contig  MDC004364.265
Writing contig  MDC002360.219
Writing contig  MDC015155.172
Writing contig  MDC019140.398
Writing contig  MDC019140.399
Writing contig  MDC011390.337
Writing contig  MDC007154.375
Writing contig  MDC010588.505
```

```
Writing contig  MDC002519.240
Writing contig  MDC006346.711
Writing contig  MDC011551.182
Writing contig  MDC002717.156
Writing contig  MDC006346.719
Writing contig  MDC007838.447
Writing contig  MDC007018.186
Writing contig  MDC017873.233
Writing contig  MDC016296.138
Writing contig  MDC019067.226
Writing contig  MDC014019.318
Writing contig  MDC007995.528
Writing contig  MDC026961.60
Writing contig  MDC020963.161
Writing contig  MDC005174.220
Writing contig  MDC040033.7
Writing contig  MDC019674.147
Writing contig  MDC010450.877
Writing contig  MDC007097.457
Writing contig  MDC016278.70
```

6. Write a python program that:

   (a) reads the text file sample_text.txt and stores in a dictionary how many times each word appears (hint: the key is the word and the count is the value);

   (b) prints to screen how many lines the file has and how many distinct words are in the file;

   (c) writes to a text file (scientist_histo.csv) the histogram of the words in comma separated value format (i.e. word,count). Words must be sorted alphabetically;

   (d) Finally, write a function that prints to screen (alphabetically) all the words that have a count higher than a threshold N and apply it with N = 5.

Show/Hide Solution

```
In [25]: def wordHisto(myText):
             """this function returns a dictionary
             with the count of each word in myText (separated by " " or "\n")
             """
             myDict = dict()
             tmp = myText.replace("\n"," ")

             for word in tmp.split(" "):
                 if(word not in myDict):
                     myDict[word] = 1
                 else:
                     myDict[word] += 1
             return myDict

         def writeWordHisto(outF, data):
             """this function writes to outFile
             the word histogram data contained in the dictionary data
             the output format is comma separated
             """
             fh = open(outF, "w")
             dictKeys = list(data.keys())
             dictKeys.sort()
             fh.write("#word,count\n")
             for k in dictKeys:
                 curVal = data[k]
```

```
            myStr = k + "," + str(curVal)+"\n" #string to write in file
            fh.write(myStr)
        fh.close() #remember to close the file

    def printWords(N, data):
        """prints the words in data that have a count higher than N"""
        dictKeys = list(data.keys())
        dictKeys.sort()
        for w in dictKeys:
            cnt = data[w]
            if(cnt > N):
                print("Word \"{}\" is present {} times".format( w, cnt))


    file = "file_samples/sample_text.txt"
    outFile = "file_samples/sample_text_histo.csv"
    wholeText = ""
    fh = open(file,"r")
    wholeText = fh.read()
    wholeText = wholeText.strip() #to remove the final newline character

    fh.close()
    print("The file {} has {} lines".format(file, wholeText.count("\n") +1 )) #n lines have n-1

    #Let's do the job:
    wordD = wordHisto(wholeText)
    writeWordHisto(outFile, wordD)
    print("The total number of distinct words is ", len(wordD))
    printWords(5,wordD)
```

```
The file file_samples/sample_text.txt has 28 lines
The total number of distinct words is  308
Word "challenge" is present 6 times
Word "future" is present 22 times
Word "reply" is present 7 times
Word "shop" is present 6 times
Word "thunder" is present 11 times
Word "umbrella" is present 10 times
```

CHAPTER 8

Practical 7

In this practical we will keep practicing with functions and will see how to get input from the command line.

8.1 Slides

The slides of the introduction can be found here: Intro

8.2 Functions

Reminder. The basic definition of a function is:

```
def function_name(input) :
    #code implementing the function
    ...
    ...
    return return_value
```

Functions are defined with the **def** keyword that proceeds the *function_name* and then a list of parameters is passed in the brackets. A colon **:** is used to end the line holding the definition of the function. The code implementing the function is specified by using indentation. A function **might** or **might not** return a value. In the first case a **return** statement is used.

8.3 Getting input from the command line

To call a program `my_python_program.py` from command line, you just have to open a terminal (in Linux) or the command prompt (in Windows) and, assuming that python is present in the path, you can `cd` into the folder containing your python program, (eg. `cd C:\python\my_exercises\`) and just type in `python3`

`my_python_program.py` or `python my_python_program.py` In case of arguments to be passed by command line, one has to put them after the specification of the program name (eg. `python my_python_program.py parm1 param2 param3`).

Python provides the module **sys** to interact with the interpreter. In particular, **sys.argv** is a list representing all the arguments passed to the python script from the command line.

Consider the following code:

```
In [2]: import sys
        """Test input from command line in systest.py"""

        if(len(sys.argv) != 4):
            print("Dear user, I was expecting 3 params. You gave me ",len(sys.argv)-1)
            exit(1)
        else:
            for i in range(0,len(sys.argv)):
                print("Param {}:{} ({})".format(i,sys.argv[i],type(sys.argv[i])))
```

```
Dear user, I was expecting 3 params. You gave me  2
```

Invoking the `systest.py` script from command line with the command `python3 exercises/systest.py 1st_param 2nd 3` will return:

```
Param 0: exercises/systest.py (<class 'str'>)
Param 1: 1st_param (<class 'str'>)
Param 2: 2nd (<class 'str'>)
Param 3: 3 (<class 'str'>)
```

Invoking the `systest.py` script from command line with the command `python3 exercises/systest.py 1st_param` will return:

```
Dear user, I was expecting three parameters. You gave me  1
```

Note that the parameter at index 0, `sys.argv[0]` holds the name of the script, and that all parameters are actually **strings** (and therefore need to be cast to numbers if we want to do mathematical operations on them).

Example: Write a script that takes two integers in input, i1 and i2, and computes the sum, difference, multiplication and division on them.

```
In [ ]: import sys
        """Maths example with input from command line"""

        if(len(sys.argv) != 3):
            print("Dear user, I was expecting 2 params. You gave me ",len(sys.argv)-1)
            exit(1)
        else:
            i1 = int(sys.argv[1])
            i2 = int(sys.argv[2])
            print("{} + {} = {}".format(i1,i2, i1 + i2))
            print("{} - {} = {}".format(i1,i2, i1 - i2))
            print("{} * {} = {}".format(i1,i2, i1 * i2))
            if(i2 != 0):
                print("{} / {} = {}".format(i1,i2, i1 / i2))
            else:
                print("{} / {} = Infinite".format(i1,i2))
```

Which, depending on user input, should give something like:

note that we need to check if the values given in input are actually numbers, otherwise the execution will crash (third

```
biancol@bluhp:~/Google Drive/work/scripts$ python3.6 /tmp/test.py
Dear user, I was expecting 2 params. You gave me  0
biancol@bluhp:~/Google Drive/work/scripts$ python3.6 /tmp/test.py 75 32
75 + 32 = 107
75 - 32 = 43
75 * 32 = 2400
75 / 32 = 2.34375
biancol@bluhp:~/Google Drive/work/scripts$ python3.6 /tmp/test.py 75 0
75 + 0 = 75
75 - 0 = 75
75 * 0 = 0
75 / 0 = Infinite
biancol@bluhp:~/Google Drive/work/scripts$ python3.6 /tmp/test.py 75 t
Traceback (most recent call last):
  File "/tmp/test.py", line 9, in <module>
    i2 = int(sys.argv[2])
ValueError: invalid literal for int() with base 10: 't'
```

example). This is easy in case of integers (`str.isdigit()`) but in case of floats it is more complex and might require Exception handling.

A more flexible and powerful way of getting input from command line makes use of the `Argparse` module[37].

8.4 Argparse

Argparse is a command line parsing module which deals with user specified parameters (positional arguments) and optional arguments.

Very briefly, the basic syntax of the `Argparse module` (for more information check the official documentation[38]) is the following.

1. Import the module:

```
import argparse
```

2. Define a argparse object:

```
parser = argparse.ArgumentParser(description="This is the description of the program")
```

note the parameter *description* that is a string to describe the program;

3. Add positional arguments:

```
parser.add_argument("arg_name", type = obj,
                    help = "Description of the parameter)
```

where `arg_name` is the name of the argument (which will be used to retrieve its value). The argument has type `obj` (the type will be automatically checked for us) and a description specified in the `help`string.

4. Add optional arguments:

[37] https://docs.python.org/3/howto/argparse.html
[38] https://docs.python.org/3/howto/argparse.html

```
parser.add_argument("-p", "--optional_arg", type = obj, default = def_val,
                    help = "Description of the parameter)
```

where `-p` is a short form of the parameter (and it is optional), `--optional_arg` is the extended name and it requires a value after it is specified, `default` is optional and gives a default value to the parameter. If not specified and no argument is passed, the argument will get the value "None". `Help` is again the description string.

5. Parse the arguments:

```
args = parser.parse_args()
```

the parser checks the arguments and stores their values in the `argparse` object that we called `args`.

6. Retrieve and process arguments:

```
myArgName = args.arg_name
myOptArg = args.optional_arg
```

now variables contain the values specified by the user and we can use them.

Example: Let's write a program that gets a string (S) and an integer (N) in input and prints the string repeated N times. Three optional parameters are specified: verbosity (-v) to make the software print a more descriptive output, separator (-s) to separate each copy of the string (defaults to " ") and trailpoints (-p) to add several "." at the end of the string (defaults to 1).

```
In [ ]: import argparse
        parser = argparse.ArgumentParser(description="""This script gets a string
                                         and an integer and repeats the string N times""")
        parser.add_argument("string", type=str,
                            help="The string to be repeated")
        parser.add_argument("N", type=int,
                            help="The number of times to repeat the string")

        parser.add_argument("-v", "--verbose", action="store_true",
                            help="increase output verbosity")

        parser.add_argument("-p", "--trailpoints", type = int, default = 1,
                            help="Adds these many trailing points")
        parser.add_argument("-s", "--separator", type = str, default = " ",
                            help="The separator between repeated strings")

        args = parser.parse_args()

        mySTR = args.string+args.separator
        trailP = "." * args.trailpoints
        answer = mySTR * args.N

        answer = answer[:-len(args.separator)] + trailP #to remove the last separator

        if args.verbose:
            print("the string {} repeated {} is:".format(args.str, args.N, answer))
        else:
            print(answer)
```

Executing the program from command line without parameters gives the message:

Calling it with the `-h` flag:

```
biancol@bluhp:~/Google Drive/work$ python3.6 /tmp/testargparse.py
usage: testargparse.py [-h] [-v] [-p TRAILPOINTS] [-s SEPARATOR] string N
testargparse.py: error: the following arguments are required: string, N
biancol@bluhp:~/Google Drive/work$
```

```
biancol@bluhp:~/Google Drive/work$ python3.6 /tmp/testargparse.py -h
usage: testargparse.py [-h] [-v] [-p TRAILPOINTS] [-s SEPARATOR] string N

This script gets a string and an integer and repeats the string N times

positional arguments:
  string                The string to be repeated
  N                     The number of time to repeated the string

optional arguments:
  -h, --help            show this help message and exit
  -v, --verbose         increase output verbosity
  -p TRAILPOINTS, --trailpoints TRAILPOINTS
                        Adds these many trailing points
  -s SEPARATOR, --separator SEPARATOR
                        The separator between repeated strings
biancol@bluhp:~/Google Drive/work$
```

With the positional arguments `"ciao a tutti"` and `3`:

```
biancol@bluhp:~/Google Drive/work$ python3.6 /tmp/testargparse.py "ciao a tutti" 3
ciao a tutti ciao a tutti ciao a tutti.
biancol@bluhp:~/Google Drive/work$
```

With the positional arguments `"ciao a tutti"` and `3`, and with the optional parameters `-s "___" -p 3 -v`

Example: Let's write a program that reads and prints to screen a text file specified by the user. Optionally, the file might be compressed with gzip to save space. The user should be able to read also gzipped files. Hint: use the module gzip which is very similar to the standard file management method (more info here[39]). You can find a text file here textFile.txt and its gzipped version here text.gz:

```
In [ ]: import argparse
        import gzip

        parser = argparse.ArgumentParser(description="""Reads and prints a text file""")
        parser.add_argument("filename", type=str, help="The file name")
        parser.add_argument("-z", "--gzipped", action="store_true",
                            help="If set, input file is assumed gzipped")

        args = parser.parse_args()
        inputFile = args.filename
        fh = ""
        if(args.gzipped):
            fh = gzip.open(inputFile, "rt")
        else:
            fh = open(inputFile, "r")

        for line in fh:
```

[39] https://docs.python.org/3/library/gzip.html?highlight=gzip#module-gzip

```
biancol@bluhp:~/Google Drive/work$ python3.6 /tmp/testargparse.py "ciao a tutti" 3 -s "___" -p 3 -v
the string ciao a tutti repeated 3 times is: ciao a tutti___ciao a tutti___ciao a tutti...
biancol@bluhp:~/Google Drive/work$
```

```
        line = line.strip("\n")
        print(line)

    fh.close()
```

The output:

```
biancol@bluhp:~/Google Drive/work/courses/QCBsciprolab$ python3.6 examples/readFile_gz.py file_samples/textFile.txt
Hi everybody,
This is my first file
and it contains a total of
four lines!
biancol@bluhp:~/Google Drive/work/courses/QCBsciprolab$ python3.6 examples/readFile_gz.py file_samples/textFile.gz -z
Hi everybody,
This is my first file
and it contains a total of
four lines!
```

Example: Let's write a program that reads the content of a file and prints to screen some stats like the number of lines, the number of characters and maximum number of characters in one line. Optionally (if flag -v is set) it should print the content of the file. You can find a text file here textFile.txt:

```
In [ ]: import argparse


    def readText(f):
        """reads the file and returns a list with
        each line as separate element"""
        myF = open(f, "r")
        ret = myF.readlines()
        return ret


    def computeStats(fileList):
        """returns a tuple (num.lines, num.characters,max_char.line)"""
        num_lines = len(fileList)
        lines_len = [len(x.replace("\n", "")) for x in fileList]
        num_char = sum(lines_len)
        max_char = max(lines_len)
        return (num_lines, num_char, max_char)


    parser = argparse.ArgumentParser(description="Computes file stats")
    parser.add_argument("inputFile", type=str, help="The input file")
    parser.add_argument(
        "-v", "--verbose", action="store_true", help="if set, prints the file content")

    args = parser.parse_args()

    inFile = args.inputFile
    lines = readText(inFile)
    stats = computeStats(lines)
    if args.verbose:
        print("File content:\n{}\n".format("".join(lines)))
    print(
        "Stats:\nN.lines:{}\nN.chars:{}\nMax. char in line:{}".format(
```

```
                stats[0], stats[1], stats[2]))
```

Output with -v flag:

```
biancol@bluhp:~/Google Drive/work/courses/QCBsciprolab$ python3 fileStats.py file_samples/textFile.txt -v
File content:
Hi everybody,
This is my first file
and it contains a total of
four lines!

Stats:
N.lines:4
N.chars:71
Max. char in line:26
```

Output without -v flag:

```
biancol@bluhp:~/Google Drive/work/courses/QCBsciprolab$ python3 fileStats.py file_samples/textFile.txt
Stats:
N.lines:4
N.chars:71
Max. char in line:26
```

8.5 Exercises

1. Modify the program of Exercise 5 of Practical 6 in order to allow users to specify the input and output files from command line. Then test it with the provided files. The text of the exercise follows:

Write a python program that reads two files. The first is a one column text file (contig_ids.txt) with the identifiers of some contigs that are present in the second file, which is a fasta formatted file (contigs82.fasta). The program will write on a third, fasta formatted file (e.g. filtered_contigs.fasta) only those entries in *contigs82.fasta* having identifier in *contig_ids.txt*.

Show/Hide Solution

```
In [ ]: """exercises/filterFasta.py"""

        import argparse

        def readIDS(f):
            """reads a one column file in and stores
            the ids in a dictionary that is returned at the end"""
            ret = dict()
            with open(f, "r") as file:
                for line in file:
                    line = line.strip()
                    if(line not in ret):
                        ret[line] = 1
            return ret

        def filterFasta(inF, outF, ids2keep):
            oF = open(outF, "w")

            outputME = False
            with open(inF, "r") as file:
                for line in file:
                    line = line.strip()
                    if line.startswith(">"):
                        #this is the header
```

```
                    if ids2keep.get(line[1:],False):
                        oF.write(line +"\n")
                        outputME = True
                        print("Writing contig ", line[1:])
                    else:
                        outputME = False
                else:
                    if outputME:
                        oF.write(line +"\n")

        oF.close()


    parser = argparse.ArgumentParser(description="Filters a fasta file")
    parser.add_argument("inputFasta", type = str, help = "The input fasta file")
    parser.add_argument("inputIDS", type = str, help = "The IDS to keep")
    parser.add_argument("outputFasta", type = str,
                        help = "The output fasta file with filtered entries")
    args = parser.parse_args()
    idsFile = args.inputIDS
    inFasta = args.inputFasta
    outFasta = args.outputFasta

    ids = readIDS(idsFile)
    filterFasta(inFasta,outFasta, ids)
```

2. Cytoscape[40] is a well known tool to perform network analysis. It is well integrated with several online databases housing for example protein-protein interactions like EBI's IntAct[41]. It is also able to read and write a very simple text file called `.sif` to represent interactions between the nodes of a network. Sif formatted files are tab separated (`\t`) and each line represents a connection between the nodes of the network. For example:

```
node1 interaction1 node2
node1 interaction2 node3
node2 interaction1 node3
```

   represents two types of interactions between node1, node2 and node3. Normally nodes are represented as circles in a network (graph) and interactions as lines (that can be of different kinds) connecting nodes (edges). The following is an extract from the file pka.sif that has been downloaded by Cytoscape from the database IntAct and represents the interactions of the Protein Kinase A (PKA) of E.coli:

```
P75742  EBI-9168813 P76594
P21513  EBI-888473  P76594
P21513  EBI-15543881    P76594
```

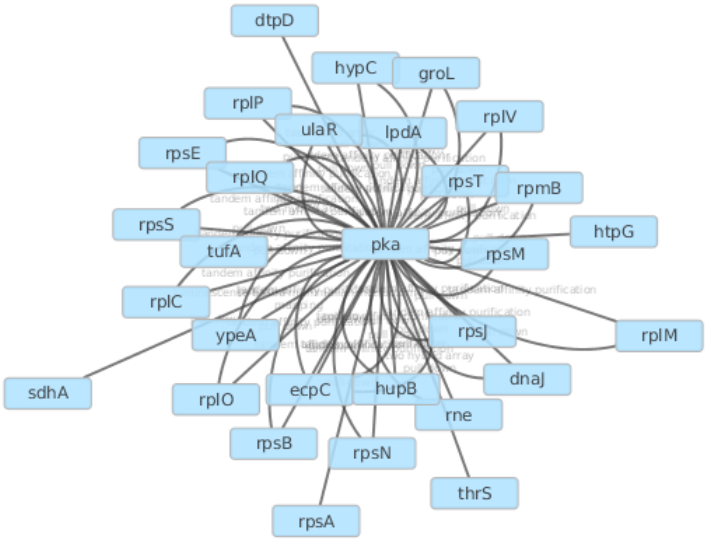
   the first and third columns represent proteins and the second is the interaction joining them. All the values are identifiers from the IntAct database. The cytoscape representation of the full set of interactions is:

Write a python script that reads in the .sif file (pka.sif is here but even better if any .sif file specified in input by the user) and stores the information in one (or more) suitable objects to be able to:

```
1. Print the interaction that is more present among the nodes;
2. Print the node that is connected to the highest number of
other nodes (no matter if on the left or right of the interaction);
```

Hint: you can store the information in a dictionary having the interaction as key and a list of tuples (node1,node2) as value. Although redundant, it is convenient to keep a list of unique nodes. **Note:** This will use more memory but it is

[40] https://www.cytoscape.org
[41] https://www.ebi.ac.uk/intact/



acceptable for small examples as it allows to quickly answer the questions.

Optional: check what these ids refer to on the IntAct database[42].

Show/Hide Solution

```
In [ ]: """exercises/parseSif.py"""

        import argparse

        def readSif(fn):
            """reads in the sif file fn and ouptputs a
            dictionary and a list:
            interDict : has "interaction" as key
            and [(node1,node2), (node3,node4)] as values
            nodes list : is the list of unique node names
            (both on the right or left hand of an interaction)
            """
            interDict = {}
            nodes = []
            with open(fn, "r") as myfile:
                for line in myfile:
                    line = line.strip()
                    [n1,inter,n2] = line.split("\t")
                    if inter not in interDict:
                        interDict[inter] = [(n1,n2)]
                    else:
                        interDict[inter].append((n1,n2))

                    if n1 not in nodes:
                        nodes.append(n1)
                    if n2 not in nodes:
                        nodes.append(n2)

            return interDict,nodes
```

[42] https://www.ebi.ac.uk/intact/

```
def getMostPresentInteraction(iDict):
    """gets the interaction dictionary as defined above and returns the
    most present interaction (with its count). If more than one, all are
    returned comma separated"""
    mpInter = ""
    mpInterCount = 0
    for inter in iDict:

        cnt = len(iDict[inter])

        if mpInterCount < cnt:
            mpInterCount = cnt
            mpInter = inter
        elif mpInterCount == cnt:
            #mpInterCount = cnt #not necessary
            mpInter = mpInter + "," + inter
    return (mpInter,mpInterCount)

def getMostPresentNode(iDict, nodeList):
    """gets the most highly connected node (or nodes, comma separated)
    and returns it with its count
    iDict : the interaction dictionary seen above
    nodeList : the node of unique node names seen above
    """
    mostPresentNode = ""
    mostPresentCount = 0
    #NOTE: iDict : {"inter1" : [(n1,n2), (n2,n3),(n3,n1), (n1,n3)],
    #               "inter2" : [(n5,n1), (n1,n2)] }
    #
    for n in nodeList:
        #the number of times a node is present is the sum of its count
        #in all interactions
        #both as first or second member of the couple
        curCnt = 0
        for i in iDict:
            pairsContainingN = [x for x in iDict[i] if n in x]
            curCnt = curCnt + len(pairsContainingN)
        if(curCnt > mostPresentCount):
            mostPresentNode = n
            mostPresentCount = curCnt
        elif curCnt == mostPresentCount:
            postPresentNode += "," + n
            mostPresentCount = curCnt

    return (mostPresentNode,mostPresentCount)

parser = argparse.ArgumentParser(description="""Reads and processes a .sif file""")
parser.add_argument("filename", type=str, help="The .sif file name")

args = parser.parse_args()
inputFile = args.filename
#inputFile = "file_samples/pka.sif"
interactions, nodeL = readSif(inputFile)
(mpI,mpICount) = getMostPresentInteraction(interactions)
print("The most present interaction(s): {}. Present {} times".format(mpI,mpICount))
(mpN, mpNCount) = getMostPresentNode(interactions,nodeL)
print("The most present node(s) {}. Present {} times".format(mpN,mpNCount))
```

3. Given a fasta[43] file like contigs82.fasta specified in input by a user, write a python script that counts, for each sequence, the number of times that a DNA or protein string specified in input appears.

If we run something like: python3 `find_stringInFasta.py contigs82.fasta TGCTCACAG`

the result should print lines like:

```
TGCTCACAG in MDC052568.000: 1 times
TGCTCACAG in MDC002479.192: 1 times
TGCTCACAG in MDC040033.7: 1 times
```

Modify the program so that it outputs also the list of all the indexes where the string appears in each sequence in the fasta file. Try to look for the following sequences:

```
TTTTCCTAGG
TGCTCCGAGCATGTGATAATCATTCCAAGCTCCAT
TAAACAT
GATTACA
```

Show/Hide Solution

```
In [ ]: """exercises/find_stringInFasta.py"""
```

```
        import argparse

        def getIndexes(string1,string2):
            """checks if string2 is present in string1 and returns
            all the positions at which string2 occurs in string1"""
            ret = []
            ind = string1.find(string2)

            while (ind > -1 and ind < len(string1)):
                ret.append(ind)
                ind = string1.find(string2,ind + 1)

            return ret

        def processFasta(file,testStr):
            """reads a fasta file entry by entry checks if the input string
               testStr is present in each sequence. Reporting how many times and where.
            """
            header = ""
            seq = ""
            with open(file, "r") as f:
                for line in f:
                    line = line.strip()
                    if(line.startswith(">")):
                        if(len(header) == 0 ):
                            #first entry:
                            header = line[1:]
                        else:
                            #this is a new entry
                            indexes = getIndexes(seq,testStr)
                            if len(indexes) > 0:
                                print("{} in {}: {} times ({})".format(testStr,
                                                                       header,
                                                                       len(indexes),
```

[43] https://en.wikipedia.org/wiki/FASTA_format

```
                                                                    indexes))
                        seq = ""
                        header = line[1:]
                else:
                    seq +=line
        #processing the final entry
        indexes = getIndexes(seq,testStr)
        if len(indexes) > 0:
            print("{} in {}: {} times ({})".format(testStr,
                                                   header,
                                                   len(indexes),
                                                   indexes))

parser = argparse.ArgumentParser(
    description="""Checks if a sequence is exactly contained in a fasta file"""
)
parser.add_argument("filename", type=str, help="The fasta file name")
parser.add_argument("query", type=str, help="The query string")

args = parser.parse_args()
#inFasta = args.filename
#testS = args.query
inFasta = "file_samples/contigs82.fasta"
testS = "TAAACAT"
processFasta(inFasta, testS)
```

4. The Fisher's dataset[44] regarding Petal and Sepal length and width in csv format can be found here. These are the measurements of the flowers of fifty plants each of the two species Iris setosa and Iris versicolor.

The header of the file is:

```
Species Number,Species Name,Petal width,Petal length,Sepal length,Sepal width
```

Write a python script that reads this file in input (feel free to hard-code the filename in the code) and computes the average petal length and width and sepal length and width for each of the three different Iris species. Print them to the screen alongside the number of elements.

Show/Hide Solution

```
In [5]: def readCSV(f):
            """reads the csv dataset and returns a dictionary with
            species name as key and, as value, a dictionary with
            four keys : petalLen, sepalLen, petalWidth, sepalWidth
            """
            ret = dict()
            with open(f, "r") as file:
                for line in file:
                    line = line.strip()
                    if not line.startswith("Species Number"):
                        data = line.split(",")
                        speciesName = data[1]
                        pWidth = int(data[2])
                        pLen = int(data[3])
                        sLen = int(data[4])
                        sWidth = int(data[5])
                        if(speciesName not in ret):
                            ret[speciesName] = {"petalLen" : [], "sepalLen" : [],
                                                "petalWidth" : [], "sepalWidth" : []
```

[44] http://onlinelibrary.wiley.com/doi/10.1111/j.1469-1809.1936.tb02137.x/abstract

```
                                        }
                    ret[speciesName]["petalLen"].append(pLen)
                    ret[speciesName]["sepalLen"].append(sLen)
                    ret[speciesName]["sepalWidth"].append(sWidth)
                    ret[speciesName]["petalWidth"].append(pWidth)
        return ret

    def printData(dataDict):
        for s in dataDict:
            avgPlen = sum(dataDict[s]["petalLen"])/len(dataDict[s]["petalLen"])
            avgPwid = sum(dataDict[s]["petalWidth"])/len(dataDict[s]["petalWidth"])
            avgSlen = sum(dataDict[s]["sepalLen"])/len(dataDict[s]["sepalLen"])
            avgSwid = sum(dataDict[s]["sepalWidth"])/len(dataDict[s]["sepalWidth"])
            print("Species {} has {} measurements:".format(s, len(dataDict[s]["petalLen"])))
            print("\t petal length {}".format(avgPlen))
            print("\t petal width {}".format(avgPwid))
            print("\t sepal length {}".format(avgSlen))
            print("\t sepal width {}".format(avgSwid))



    inFile = "file_samples/Fishers_Iris.csv"

    data = readCSV(inFile)
    printData(data)
```

```
Species I. Setosa has 50 measurements:
         petal length 14.62
         petal width 2.46
         sepal length 34.28
         sepal width 50.1
Species I. Verginica has 50 measurements:
         petal length 55.52
         petal width 20.06
         sepal length 29.74
         sepal width 65.88
Species I. Versicolor has 50 measurements:
         petal length 43.22
         petal width 13.26
         sepal length 27.64
         sepal width 59.36
```

CHAPTER 9

Practical 8

In this practical we will will practice Pandas.

9.1 Slides

The slides of the introduction can be found here: Intro

9.2 Libraries installation

First things first. Let's start off by installing the required libraries. In particular we will need three libraries. Try and see if they are already available by typing the following commands in the console or put them in a python script:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

if, upon execution, you do not get any error messages, you are sorted. Otherwise, you need to install them.

In **Linux** you can install the libraries by typing in a terminal `sudo pip3 install matplotlib`, `sudo pip3 install pandas` and `sudo pip3 install numpy` (or `sudo python3.X -m pip install matplotlib`, `sudo python3.X -m pip install pandas` and `sudo python3.X -m pip install numpy`), where X is your python version.

In **Windows** you can install the libraries by typing in the command prompt (to open it type `cmd` in the search) `pip3 install matplotlib`, `pip3 install pandas` and `pip3 install numpy`. If you are using anaconda you need to run these commands from the *anaconda prompt*.

Please install them in this order. You might not need to install numpy as matplotlib requires it. Once done that, try to perform the above imports again and they should work this time around.

9.3 Pandas

Pandas (the name comes from *panel data*) is a very efficient library to deal with numerical tables and time series. It is a quite complex library and here we will only scratch the surface of it. You can find a lot of information including the documentation on the Pandas website[45].

In particular the library pandas provides two data structures: **Series** and **DataFrames**.

9.4 Series

Series are 1-dimensional structures (like lists) containing data. Series are characterized by two types of information: the **values** and the **index** (a list of labels associated to the data), therefore they are a bit like a list and a bit like a dictionary. The index is optional and can be added by the library if not specified.

9.4.1 How to define and access a Series

There are several ways to define a Series. We can specify both the values and the index esplicitly, or through a dictionary, or let python add the default index for us. We can access the index with the **Series.index** method and the values with the **Series.values**.

```
In [1]: import pandas as pd
        import random

        print("Values and index explicitly defined")
        #values and index explicitely defined
        S = pd.Series([random.randint(0,20) for x in range(0,10)],
                      index = list("ABCDEFGHIL"))

        print(S)
        print("The index:", S.index)
        print("The values:", S.values)

        print("------------------------\n")
        print("From dictionary")
        #from a dictionary
        S1 = pd.Series({"one" : 1, "two" : 2, "ten": 10,
                        "three" : 3, "four": 4, "forty" : 40})
        print(S1)
        print(S1.index)
        print(S1.values)
        print("------------------------\n")
        print("Default index")
        #index added by default
        myData = [random.randint(0,10) for x in range(10)]
        S2 = pd.Series(myData)

        print(S2)
        print(S2.index)
        print(S2.values)

        print("------------------------\n")
        print("Same value repeated")
        S3 = pd.Series(1.27, range(10))
```

[45] http://pandas.pydata.org

```
        print(S3)
        print(S3.index)
        print(S3.values)
```

```
Values and index explicitly defined
A     9
B     6
C    20
D    10
E     0
F    13
G    10
H     4
I    14
L    15
dtype: int64
The index: Index(['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'L'], dtype='object')
The values: [ 9  6 20 10  0 13 10  4 14 15]
------------------------

From dictionary
one       1
two       2
ten      10
three     3
four      4
forty    40
dtype: int64
Index(['one', 'two', 'ten', 'three', 'four', 'forty'], dtype='object')
[ 1  2 10  3  4 40]
------------------------

Default index
0     8
1     2
2     8
3    10
4     1
5     5
6     3
7     8
8     9
9     5
dtype: int64
RangeIndex(start=0, stop=10, step=1)
[ 8  2  8 10  1  5  3  8  9  5]
------------------------

Same value repeated
0    1.27
1    1.27
2    1.27
3    1.27
4    1.27
5    1.27
6    1.27
7    1.27
8    1.27
9    1.27
```

```
dtype: float64
RangeIndex(start=0, stop=10, step=1)
[1.27 1.27 1.27 1.27 1.27 1.27 1.27 1.27 1.27 1.27]
```

Data in a series can be accessed by using the **label** (i.e. the index) as in a dictionary or through its **position** as in a list. Slicing is also allowed both by **position** and **index**. In the latter case, we can do `Series[S:E]` with **S and E indexes**, both **S and E are included**.

It is also possible to retrieve some elements by passing a **list** of positions or indexes. **Head** and **tail** methods can also be used to retrieve the top or bottom N elements with **Series.head(N)** or **Series.tail(N)**.

Note: When the method returns more than one element, the return type is a Series.

```
In [3]: import pandas as pd
        import random

        #values and index explicitely defined
        S = pd.Series([random.randint(0,20) for x in range(0,10)],
                      index = list("ABCDEFGHIL"))
        print(S)
        print("")

        print("Value at label \"A\":", S["A"])
        print("Value at index 1:", S[1])
        print("")

        print("Slicing from 1 to 3:") #note 3 excluded
        print(S[1:3])
        print("")
        print("Slicing from C to H:") #note H included!
        print(S["C":"H"])
        print("")

        print("Retrieving from list:")
        print(S[[1,3,5,7,9]])
        print(S[["A","C","E","G"]])
        print("")

        print("Top 3")
        print(S.head(3))
        print("")
        print("Bottom 3")
        print(S.tail(3))
```

```
A    15
B    11
C     4
D     7
E     1
F     4
G    15
H    14
I    14
L    17
dtype: int64

Value at label "A": 15
Value at index 1: 11
```

```
Slicing from 1 to 3:
B    11
C     4
dtype: int64

Slicing from C to H:
C     4
D     7
E     1
F     4
G    15
H    14
dtype: int64

Retrieving from list:
B    11
D     7
F     4
H    14
L    17
dtype: int64
A    15
C     4
E     1
G    15
dtype: int64

Top 3
A    15
B    11
C     4
dtype: int64

Bottom 3
H    14
I    14
L    17
dtype: int64
```

9.4.2 Operator broadcasting

Operations can automatically be broadcast to the entire Series. This is a quite cool feature and saves us from looping through the elements of the Series.

Example: Given a list of 10 integers and we want to divide them by 2. Without using pandas we would:

```
In [4]: import random

        listS = [random.randint(0,20) for x in range(0,10)]

        print(listS)

        for el in range(0,len(listS)):
            listS[el] /=2  #compact of X = X / 2

        print(listS)
```

```
[6, 4, 5, 19, 14, 16, 9, 3, 13, 11]
[3.0, 2.0, 2.5, 9.5, 7.0, 8.0, 4.5, 1.5, 6.5, 5.5]
```

With pandas instead:

```
In [5]: import pandas as pd
        import random

        S = pd.Series([random.randint(0,20) for x in range(0,10)],
                      index = list("ABCDEFGHIL"))

        print(S)
        print("")
        S1 = S / 2
        print(S1)
```

```
A     4
B    13
C    14
D     6
E    13
F     2
G    13
H    19
I    20
L     7
dtype: int64

A     2.0
B     6.5
C     7.0
D     3.0
E     6.5
F     1.0
G     6.5
H     9.5
I    10.0
L     3.5
dtype: float64
```

9.4.3 Filtering

We can also apply boolean operators to obtain only the **sub-Series** with all the values satisfying a specific condition. This allows us to **filter** the Series.

Calling the boolean operator on the series alone (e.g. S > 10) will return a Series with True at the indexes where the condition is met, False at the others. Passing such a Series to a Series of the same lenght will return only the elements where the condition is True. Check the code below to see this in action.

```
In [6]: import pandas as pd
        import random

        S = pd.Series([random.randint(0,20) for x in range(0,10)],
                      index = list("ABCDEFGHIL"))

        print(S)
        print("")
        S1 = S>10
        print(S1)
```

```
        print("")
        S2 = S[S > 10]
        print(S2)
```

```
A    14
B    18
C    10
D     6
E    17
F    14
G     2
H    17
I     4
L    15
dtype: int64

A     True
B     True
C    False
D    False
E     True
F     True
G    False
H     True
I    False
L     True
dtype: bool

A    14
B    18
E    17
F    14
H    17
L    15
dtype: int64
```

9.4.4 Missing data

Operations involving Series might have to deal with missing data or non-valid values (both cases are represented as **NaN**, that is **not a number**). Operations are carried out by aligning the Series based on their **indexes**. Indexes not in common end up in NaN values. Although most the operations that can be performed on series quite happily deal with NaNs, it is possible to **drop** NaN values or to **fill** them (i.e. replace their value with some other value). The sytax is:

```
Series.dropna()
```

or

```
Series.fillna(some_value)
```

Note that these operations do not modify the Series but rather return a new Series.

```
In [7]: import pandas as pd
        import random

        S = pd.Series([random.randint(0,10) for x in range(0,10)],
                      index = list("ABCDEFGHIL"))

        S1 = pd.Series([random.randint(0,10) for x in range(0,8)],
```

```
                        index = list("DEFGHAZY"))

        print("The dimensions of these Series:", S.shape)
        print("")
        print(S1)
        print("---- S + S1 ----")
        Ssum = S + S1
        print(Ssum)
        print("---- Dropping NaNs ----")
        print(Ssum.dropna())
        print("---- Filling NaNs ----")
        print(Ssum.fillna("my_value"))
```

```
The dimensions of these Series: (10,)

D     4
E     2
F     7
G     2
H     0
A     6
Z     1
Y    10
dtype: int64
---- S + S1 ----
A    10.0
B     NaN
C     NaN
D    14.0
E     4.0
F    14.0
G     8.0
H     1.0
I     NaN
L     NaN
Y     NaN
Z     NaN
dtype: float64
---- Dropping NaNs ----
A    10.0
D    14.0
E     4.0
F    14.0
G     8.0
H     1.0
dtype: float64
---- Filling NaNs ----
A          10
B    my_value
C    my_value
D          14
E           4
F          14
G           8
H           1
I    my_value
L    my_value
Y    my_value
Z    my_value
```

```
dtype: object
```

9.4.5 Computing stats

Pandas offer several operators to compute stats on the data stored in a Series. These include basic stats like **min**, **max** (and relative indexes with **argmin** and **argmax**) **mean**, **std**, **quantile** (to get the quantiles. A description of the data can be obtained by using the method **describe** and the counts for each value can be obtained by **value_counts**. Other methods available are **sum** and **cumsum** (for the sum and cumulative sum of the elements), **autocorr** and **corr** (for autocorrelation and correlation) and many others. For a complete list check the Pandas reference[46].

Note: as said before, when these methods do not return a single value, they return a Series.

Example: Fill the previous Series with the mean values of the series rather than NaNs.

```
In [8]: Ssum = S + S1
        print(Ssum)
        print("---- Filling with avg value ----")
        print(Ssum.fillna(Ssum.mean()))
```

```
A    10.0
B     NaN
C     NaN
D    14.0
E     4.0
F    14.0
G     8.0
H     1.0
I     NaN
L     NaN
Y     NaN
Z     NaN
dtype: float64
---- Filling with avg value ----
A    10.0
B     8.5
C     8.5
D    14.0
E     4.0
F    14.0
G     8.0
H     1.0
I     8.5
L     8.5
Y     8.5
Z     8.5
dtype: float64
```

Let's see some operators introduced above in action.

```
In [9]: import pandas as pd
        import random

        S = pd.Series([random.randint(0,10) for x in range(0,10)],
                      index = list("ABCDEFGHIL"))
        print("The data:")
        print(S)
        print("")
        print("Its description")
```

[46] http://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.html?highlight=series

```
        print(S.describe())
        print("")
        print("Specifying different quantiles:")
        print(S.quantile([0.1,0.2,0.8,0.9]))
        print("")
        print("Histogram:")
        print(S.value_counts())
        print("")
        print("The type is a Series:")
        print(type(S.value_counts()))
        print("Summing the values:")
        print(S.sum())
        print("")
        print("The cumulative sum:")
        print(S.cumsum())
```

```
The data:
A     3
B    10
C     3
D     4
E     9
F     9
G     5
H     0
I    10
L     5
dtype: int64

Its description
count    10.000000
mean      5.800000
std       3.489667
min       0.000000
25%       3.250000
50%       5.000000
75%       9.000000
max      10.000000
dtype: float64

Specifying different quantiles:
0.1     2.7
0.2     3.0
0.8     9.2
0.9    10.0
dtype: float64

Histogram:
10    2
9     2
5     2
3     2
4     1
0     1
dtype: int64

The type is a Series:
<class 'pandas.core.series.Series'>
Summing the values:
```

```
58

The cumulative sum:
A     3
B    13
C    16
D    20
E    29
F    38
G    43
H    43
I    53
L    58
dtype: int64
```

Example:

Create two Series from the lists [2, 4, 6, 8, 10, 12, 13, 14, 15, 16], [1, 3, 5, 7, 9, 11, 13, 14, 15, 16] using the same index for both: ['9B47', '468B', 'B228', '3C52', 'AE2E', 'DFF6', 'C38B', '2CE5', '0325', '398F'].

Let's compare the distribution stats of the two Series (mean value, max, min, quantiles) and get the index and value of the positions where the two series are the same. Finally, let's get the sub-series where the first has an higher value than the first quartile of the second and compute its stats.

```
In [10]: import pandas as pd

         L1 = [2, 4, 6, 8, 10, 12, 13, 14, 15, 16]
         L2 = [1, 3, 5, 7, 9, 11, 13, 14, 15, 16]
         I = ['9B47', '468B', 'B228', '3C52', 'AE2E', 'DFF6', 'C38B', '2CE5', '0325', '398F']
         L1Series = pd.Series(L1,index = I)
         L2Series = pd.Series(L2, index = I)
         #Let's describe the stats
         print("Stats of L1Series")
         print(L1Series.describe())
         print("")
         print("Stats of L2Series")
         print(L2Series.describe())
         print("")
         #This is a Series with boolean values (True means the two Series where the same)
         Leq = L1Series == L2Series
         print("Equality series")
         print(Leq)
         print("")
         #Get the subseries where both are the same
         Lsub = L1Series[Leq]
         print("Subseries of identicals")
         print(Lsub)
         print("")
         #Get the values that are the same
         print("Identical values:")
         print(Lsub.values)
         print("")
         #Get the indexes where the two series are the same
         print("Indexes of identical values:")
         print(Lsub.index)
         print("")
         firstQuartile = L2Series.quantile(0.25)
         print("The first quartile of L2Series:",firstQuartile)
         print("")
         #Get the subseries in which L1 is bigger than L2
```

```
        Lbig = L1Series[L1Series > firstQuartile]
        print("The subseries with L1 > L2")
        print(Lbig)
```

```
Stats of L1Series
count    10.000000
mean     10.000000
std       4.830459
min       2.000000
25%       6.500000
50%      11.000000
75%      13.750000
max      16.000000
dtype: float64

Stats of L2Series
count    10.00000
mean      9.40000
std       5.25357
min       1.00000
25%       5.50000
50%      10.00000
75%      13.75000
max      16.00000
dtype: float64

Equality series
9B47    False
468B    False
B228    False
3C52    False
AE2E    False
DFF6    False
C38B     True
2CE5     True
0325     True
398F     True
dtype: bool

Subseries of identicals
C38B    13
2CE5    14
0325    15
398F    16
dtype: int64

Identical values:
[13 14 15 16]

Indexes of identical values:
Index(['C38B', '2CE5', '0325', '398F'], dtype='object')

The first quartile of L2Series: 5.5

The subseries with L1 > L2
B228     6
3C52     8
AE2E    10
DFF6    12
```

```
C38B    13
2CE5    14
0325    15
398F    16
dtype: int64
```

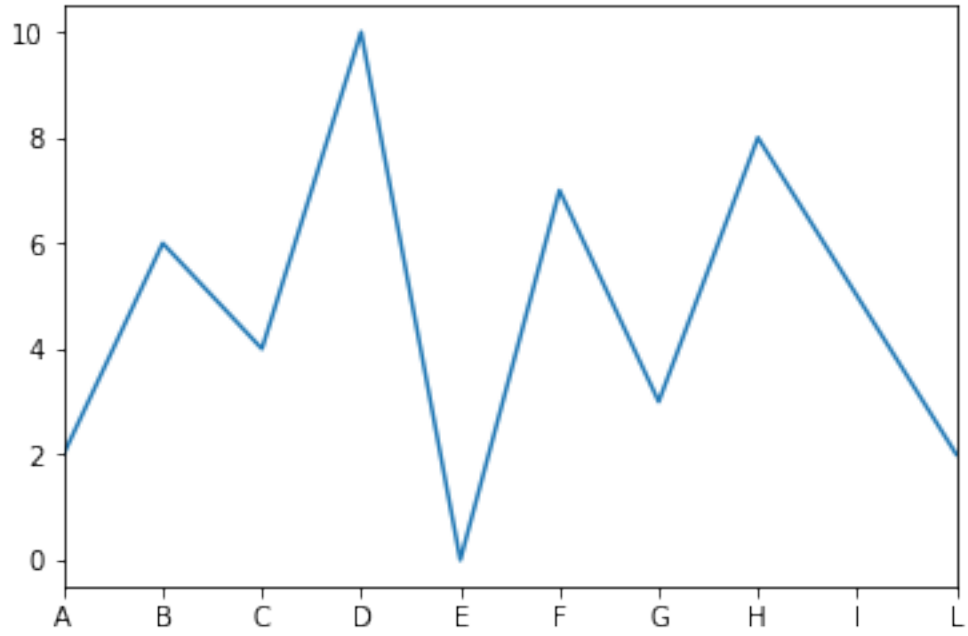
9.5 Plotting data

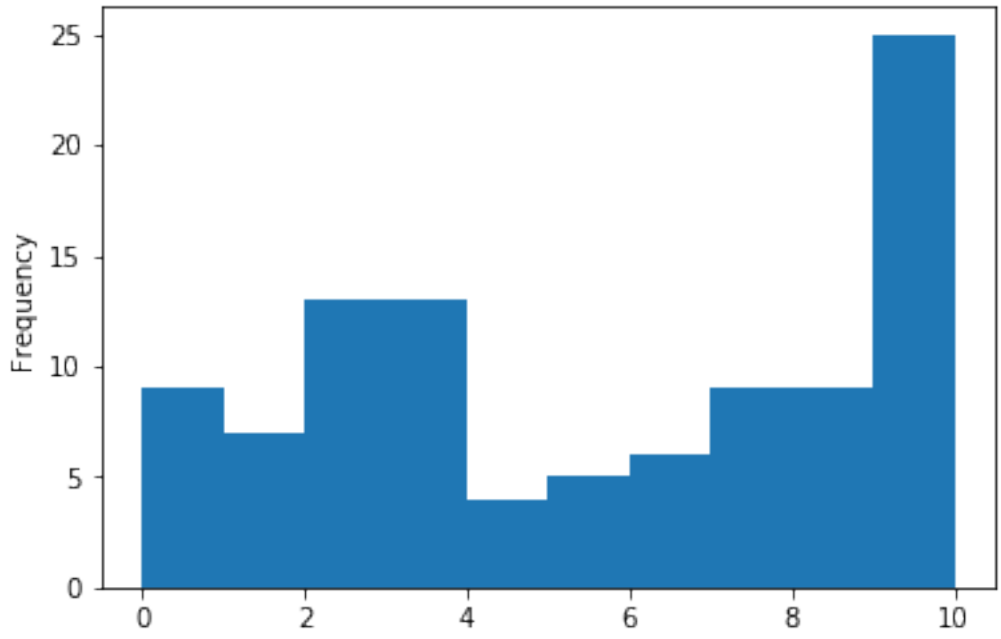
Using python's matplotlib it is possible to plot data. The basic syntax is `Series.plot(kind = "type")` the parameter `kind` can be used to produce several types of plots (examples include **line**, **hist**, **pie**, **bar**, see here[47] for all possible choices). Note that **matplotlib** needs to be imported and the pyplot needs to be shown with **pyplot.show()** to display the plot.

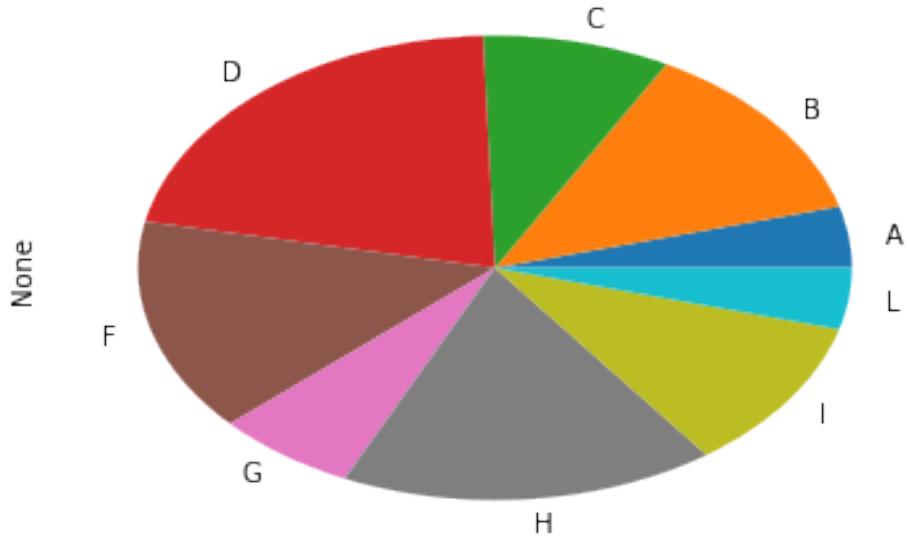
```
In [11]: import pandas as pd
         import matplotlib.pyplot as plt
         import random
         S = pd.Series([random.randint(0,10) for x in range(0,10)],
                       index = list("ABCDEFGHIL"))
         print("The data:")
         print(S)
         S1 = pd.Series([random.randint(0,10) for x in range(0,100)])

         S.plot()
         plt.show()
         plt.close()
         S1.plot(kind = "hist")
         plt.show()
         plt.close()
         S.plot(kind = "pie")
         plt.show()
         plt.close()
```

```
The data:
A     2
B     6
C     4
D    10
E     0
F     7
G     3
H     8
I     5
L     2
dtype: int64
```

[47] https://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.plot.html

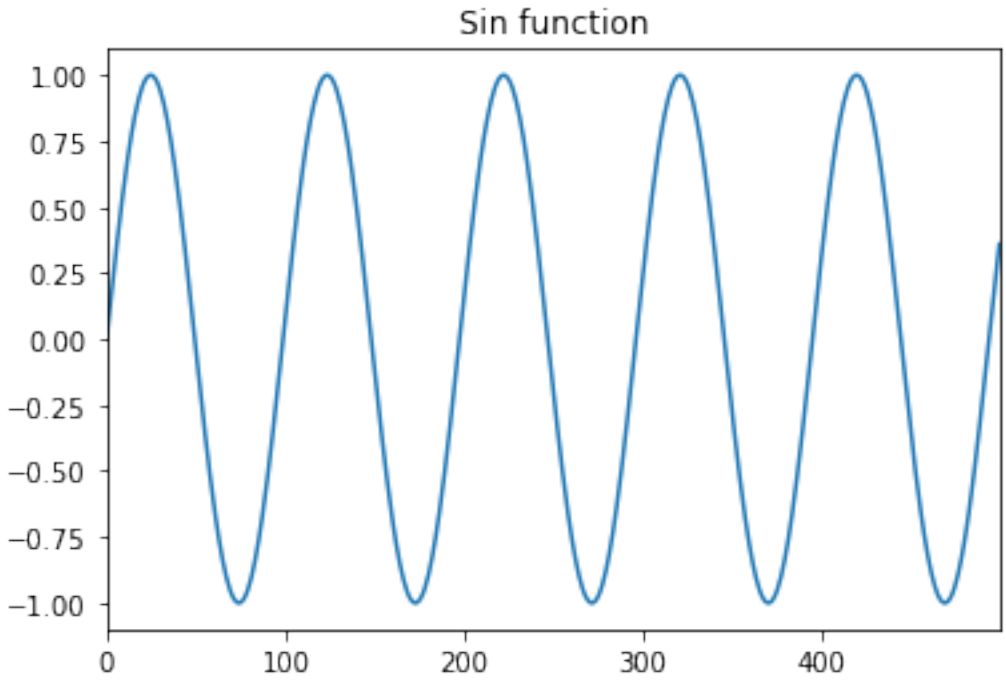

10
8
6
4
2
0
A
B
C
D
E
F
G
H
I
L

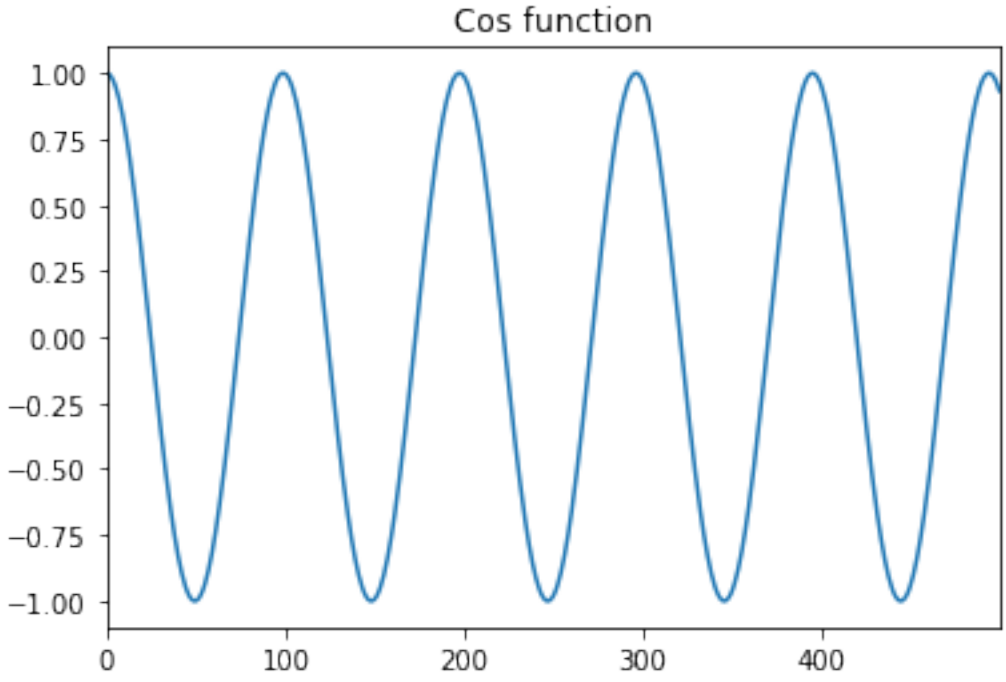

Frequency
25
20
15
10
5
0
0
2
4
6
8
10

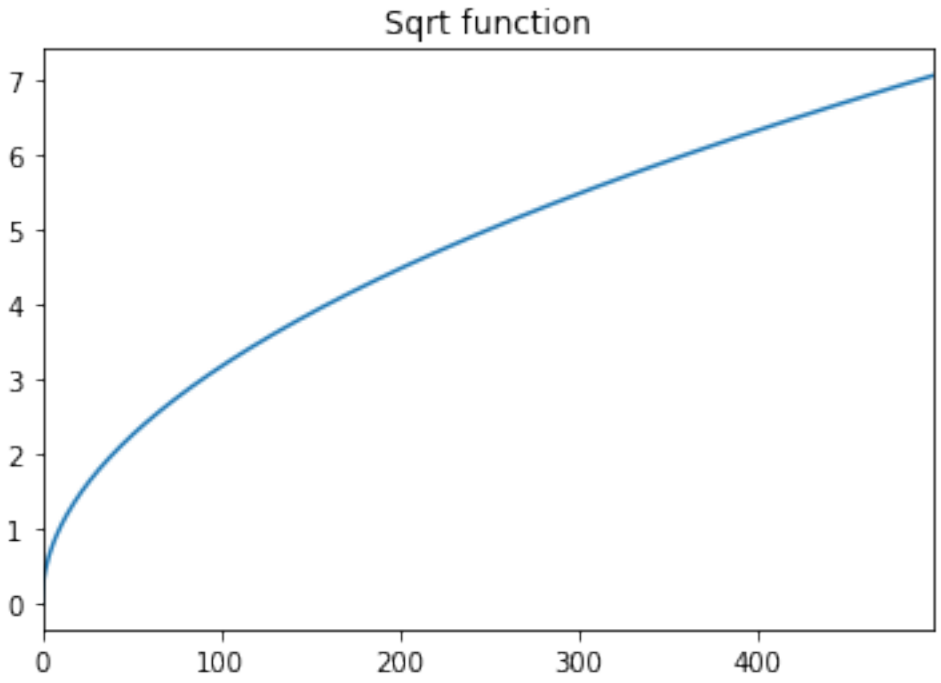


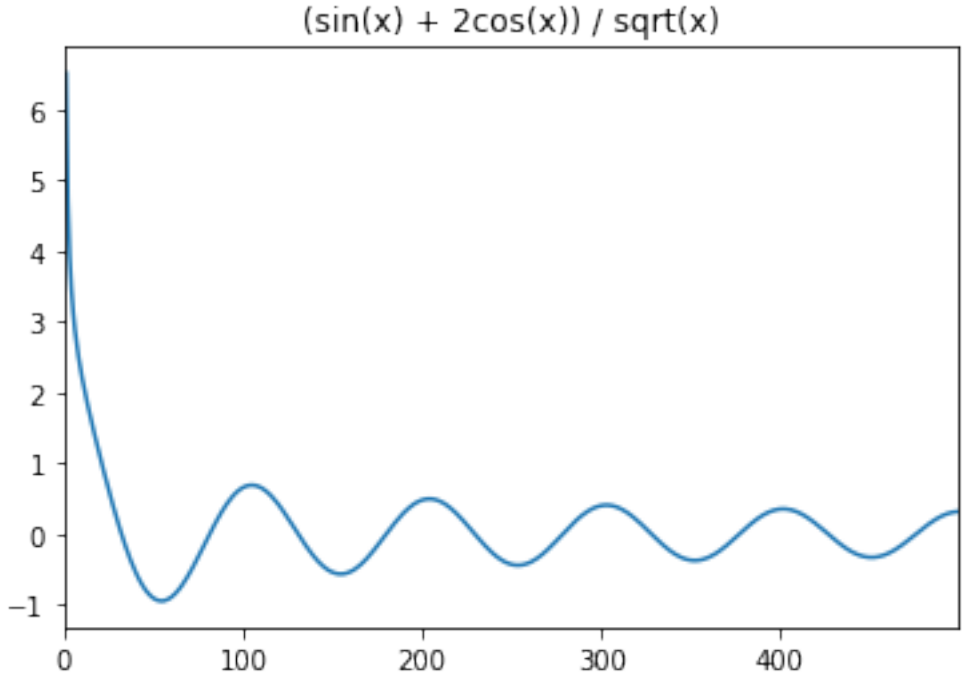
Example: Let's create a series representing the sin, cos and sqrt functions and plot them.

```
In [6]: import math
        import matplotlib.pyplot as plt
        import pandas as pd

        x = [i/10 for i in range(0,500)]


        y =  [math.sin(2*i/3.14 ) for i in x]
        y1 = [math.cos(2*i/3.14 ) for i in x]
        y2 = [math.sqrt(i) for i in x]
        #print(x)

        ySeries = pd.Series(y)
        ySeries1 = pd.Series(y1)
        ySeries2 = pd.Series(y2)

        ySeries.plot()
        plt.title("Sin function")
        plt.show()
        plt.close()
        ySeries1.plot()
        plt.title("Cos function")
        plt.show()
        plt.close()
        ySeries2.plot()
        plt.title("Sqrt function")
        plt.show()
        plt.close()
        ySeries2 = (ySeries + 2*ySeries1)/ySeries2
        ySeries2.plot()
        plt.title("(sin(x) + 2cos(x)) / sqrt(x)")
        plt.show()
```


Sin function
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
100
200
300
400


Cos function
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
100
200
300
400





9.6 Pandas DataFrames

DataFrames in pandas are the 2D analogous of Series. Dataframes are spreadsheet-like data structures with an ordered set of columns that can also be dishomogeneous. We can think about Dataframes as dictionaries of Series, each one representing a named column. Dataframes are described by an **index** that contains the labels of rows and a **columns** structure that holds the labels of the columns.

Note that the operation of extracting a column from a DataFrame returns a Series. Moreover, most (but not all!) of the operations that apply to Series also apply to DataFrames.

9.6.1 Define a DataFrame

There are several different ways to define a DataFrame. It is possible to create a DataFrame starting from a dictionary having Series as values. In this case, they **keys** of the dictionary are the **columns** of the DataFrame.

```
In [13]: import pandas as pd

         myData = {
             "temperature" : pd.Series([1, 3, 8, 13, 17, 20, 22, 22,18 ,13,6,2],
                                       index = ["Jan","Feb", "Mar","Apr","May","Jun",
                                                "Jul","Aug","Sep","Oct","Nov","Dec"]
                                        ),
             "dayLength" : pd.Series([9.7, 10.9, 12.5, 14.1, 15.6, 16.3, 15.9,
                                      14.6,13,11.4,10,9.3],
                                       index = ["Jan","Feb", "Mar","Apr","May","Jun",
                                                "Jul","Aug","Sep","Oct","Nov","Dec"]
                                        )

         }

         DF = pd.DataFrame(myData)
         print(DF)

         print(DF.columns)
         print(DF.index)
```

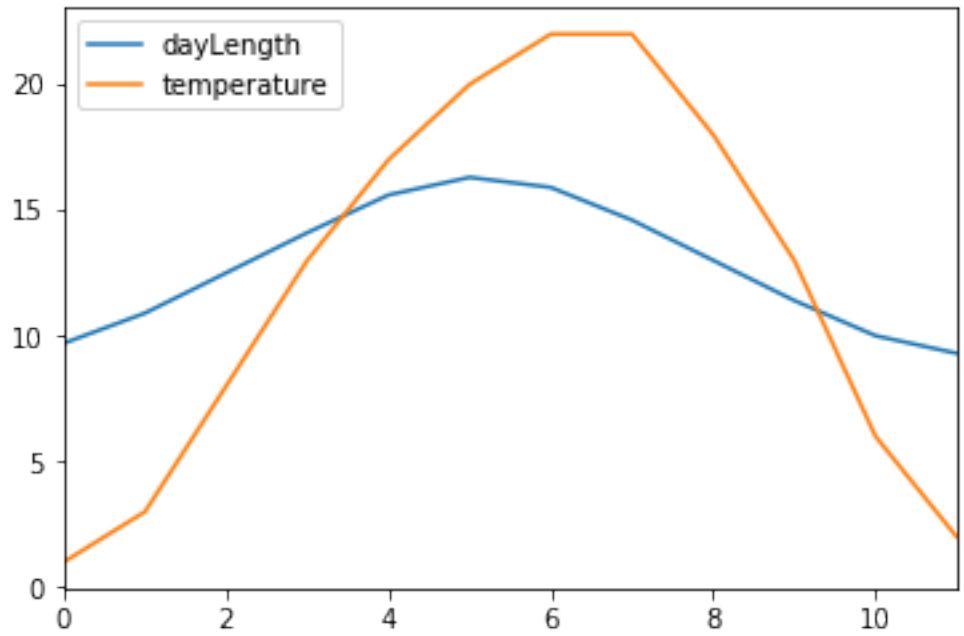
```
     dayLength  temperature
Jan        9.7            1
Feb       10.9            3
Mar       12.5            8
Apr       14.1           13
May       15.6           17
Jun       16.3           20
Jul       15.9           22
Aug       14.6           22
Sep       13.0           18
Oct       11.4           13
Nov       10.0            6
Dec        9.3            2
Index(['dayLength', 'temperature'], dtype='object')
Index(['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct',
       'Nov', 'Dec'],
      dtype='object')
```

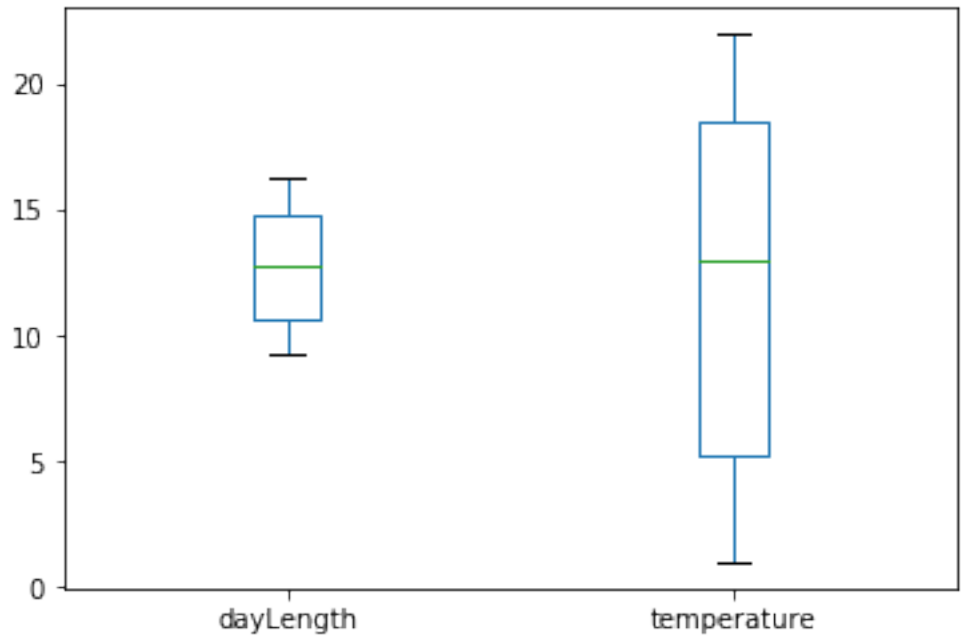
If the index is not specified, it is given by default:

```
In [14]: import pandas as pd
         import matplotlib.pyplot as plt

         myData = {
             "temperature" : pd.Series([1, 3, 8, 13, 17, 20,
                                        22, 22,18 ,13,6,2]),
             "dayLength" : pd.Series([9.7, 10.9, 12.5, 14.1, 15.6,
                                      16.3,15.9,14.6,13,11.4,10,9.3])

         }
```

```
        DF = pd.DataFrame(myData)
        print(DF)
        print(DF.index)

        DF.plot()
        plt.show()
        plt.close()
        DF.plot(kind ="box")
        plt.show()
```

```
    dayLength  temperature
0         9.7            1
1        10.9            3
2        12.5            8
3        14.1           13
4        15.6           17
5        16.3           20
6        15.9           22
7        14.6           22
8        13.0           18
9        11.4           13
10       10.0            6
11        9.3            2
RangeIndex(start=0, stop=12, step=1)
```





It is also possible to define a DataFrame from a list of dictionaries holding a set of values rather than Series. Note that when columns do not have the corresponding information a NaN is added. Indexes and columns can be changed after they have been defined.

```
In [15]: import pandas as pd

         myData = [{"A" : 1, "B" : 2, "C" : 3.2, "D" : 10},
                   {"A" : 1, "B" : 2, "F" : 3.2, "G" : 10, "H":1},
                   {"A" : 1, "B" : 2, "C" : 3.2, "D" : 1,
                    "E": 4.1, "F" : 3.2, "G" : 10, "H":1}

                  ]

         DF = pd.DataFrame(myData)
         print(DF)
         print("")
         #Let's change the columns and indexes
         columns = "val1,val2,val3,val4,val5,val6,val7,val8".split(',')
         inds = ["Day1", "Day2", "Day3"]

         DF.columns = columns
         DF.index = inds
         print(DF)
```

```
   A  B    C     D    E    F     G    H
0  1  2  3.2  10.0  NaN  NaN   NaN  NaN
1  1  2  NaN   NaN  NaN  3.2  10.0  1.0
2  1  2  3.2   1.0  4.1  3.2  10.0  1.0

      val1  val2  val3  val4  val5  val6  val7  val8
Day1     1     2   3.2  10.0   NaN   NaN   NaN   NaN
Day2     1     2   NaN   NaN   NaN   3.2  10.0   1.0
Day3     1     2   3.2   1.0   4.1   3.2  10.0   1.0
```

9.6.2 Loading data from external files

Pandas also provide methods to load data from external files. In particular, to load data from a .cvs file, we can use the method `pandas.read_csv(filename)`. This method has a lot of parameters, you can see all the details on its usage on the official documentation[48]. Some useful optional parameters are the **separator** (to specify the column separator like `sep="\t"` for tab separated files), the character to identify comments (`comment="#"`) or that to skip some lines (like the header for example or any initial comments to the file) `skiprows=N`. The rows to use as column header can be specified with the parameter `header` that accepts a number or a list of numbers. Similarly, the columns to use as index can be specified with the parameter `index_col`.

Other methods to read data in are `pandas.read_excel(filename)` that works in a similar way but can load excel files (see here[49]).

Example: Let's load the data stored in a csv datafile sampledata_orders.csv in a pandas DataFrame.

```
In [16]: import pandas as pd

         orders = pd.read_csv("file_samples/sampledata_orders.csv",
                              sep=",", index_col = 0, header = 0)
         print("First 5 entries:")
         print(orders.head())
         print("")
         print("The index:")
         print(orders.index)
         print("")
         print("The column names:")
         print(orders.columns)
         print("")
         print("A description of the numerical values:")
         print(orders.describe())
```

```
First 5 entries:
        Order ID  Order Date Order Priority  Order Quantity       Sales  \
Row ID
1              3  10/13/2010            Low               6    261.5400
49           293   10/1/2012           High              49  10123.0200
50           293   10/1/2012           High              27    244.5700
80           483   7/10/2011           High              30   4965.7595
85           515   8/28/2010  Not Specified              19    394.2700

        Discount       Ship Mode   Profit  Unit Price  Shipping Cost  \
Row ID
1           0.04     Regular Air  -213.25       38.94          35.00
49          0.07  Delivery Truck   457.81      208.16          68.02
50          0.01     Regular Air    46.71        8.69           2.99
80          0.08     Regular Air  1198.97      195.99           3.99
85          0.08     Regular Air    30.94       21.78           5.94

             Customer Name Province   Region Customer Segment  \
Row ID
1       Muhammed MacIntyre  Nunavut  Nunavut   Small Business
49            Barry French  Nunavut  Nunavut         Consumer
50            Barry French  Nunavut  Nunavut         Consumer
80           Clay Rozendal  Nunavut  Nunavut        Corporate
```

[48] http://pandas.pydata.org/pandas-docs/stable/generated/pandas.read_csv.html#pandas.read_csv
[49] http://pandas.pydata.org/pandas-docs/version/0.20/generated/pandas.read_excel.html

```
85          Carlos Soltero  Nunavut  Nunavut         Consumer

       Product Category            Product Sub-Category  \
Row ID
1       Office Supplies          Storage & Organization
49      Office Supplies                      Appliances
50      Office Supplies  Binders and Binder Accessories
80           Technology    Telephones and Communication
85      Office Supplies                      Appliances

                                           Product Name Product Container  \
Row ID
1        Eldon Base for stackable storage shelf, platinum         Large Box
49      1.7 Cubic Foot Compact "Cube" Office Refrigera...        Jumbo Drum
50       Cardinal Slant-D® Ring Binder, Heavy Gauge Vinyl         Small Box
80                                                  R380         Small Box
85                            Holmes HEPA Air Purifier        Medium Box

        Product Base Margin   Ship Date
Row ID
1                      0.80  10/20/2010
49                     0.58   10/2/2012
50                     0.39   10/3/2012
80                     0.58   7/12/2011
85                     0.50   8/30/2010

The index:
Int64Index([   1,   49,   50,   80,   85,   86,   97,   98,  103,  107,
            ...
            6492, 6526, 6657, 7396, 7586, 7765, 7766, 7906, 7907, 7914],
           dtype='int64', name='Row ID', length=8399)

The column names:
Index(['Order ID', 'Order Date', 'Order Priority', 'Order Quantity', 'Sales',
       'Discount', 'Ship Mode', 'Profit', 'Unit Price', 'Shipping Cost',
       'Customer Name', 'Province', 'Region', 'Customer Segment',
       'Product Category', 'Product Sub-Category', 'Product Name',
       'Product Container', 'Product Base Margin', 'Ship Date'],
      dtype='object')

A description of the numerical values:
           Order ID  Order Quantity         Sales     Discount        Profit  \
count   8399.000000     8399.000000   8399.000000  8399.000000   8399.000000
mean   29965.179783       25.571735   1775.878179     0.049671    181.184423
std    17260.883447       14.481071   3585.050525     0.031823   1196.653372
min        3.000000        1.000000      2.240000     0.000000 -14140.700000
25%    15011.500000       13.000000    143.195000     0.020000    -83.315000
50%    29857.000000       26.000000    449.420000     0.050000     -1.500000
75%    44596.000000       38.000000   1709.320000     0.080000    162.750000
max    59973.000000       50.000000  89061.050000     0.250000  27220.690000

        Unit Price  Shipping Cost  Product Base Margin
count  8399.000000    8399.000000          8336.000000
mean     89.346259      12.838557             0.512513
std     290.354383      17.264052             0.135589
min       0.990000       0.490000             0.350000
25%       6.480000       3.300000             0.380000
50%      20.990000       6.070000             0.520000
75%      85.990000      13.990000             0.590000
```

```
max     6783.020000     164.730000              0.850000
```

9.6.3 Extract values by row and column

Once a DataFrame is populated we can access its content. Several options are available:

1. Select by column `DataFrame[col]` returns a Series
2. Select by row label `DataFrame.loc[row_label]` returns a Series
3. Select row by integer location `DataFrame.iloc[row_position]` returns a Series
4. Slice rows `DataFrame[S:E]` (S and E are labels, both included) returns a DataFrame
5. Select rows by boolean vector `DataFrame[bool_vect]` returns a DataFrame

Note that if names are well formed (i.e. no spaces, no strange characters...) we can use DataFrame.col instead of DataFrame[col]. Here are some ways to extract data from the orders dataframe:

```
In [17]: import pandas as pd

         orders = pd.read_csv("file_samples/sampledata_orders.csv", sep=",", index_col =0, header=0)

         print("The Order Quantity column (top 5)")
         print(orders["Order Quantity"].head(5))
         print("")
         print("The Sales column (top 10)")
         print(orders.Sales.head(10))
         print("")
         print("The row with ID:50")
         r50 = orders.loc[50]
         print(r50)
         print("")
         print("The third row:")
         print(orders.iloc[3])

         print("The Order Quantity, Sales, Discount and Profit of the 2nd,4th, 6th and 8th row:")
         print(orders[1:8:2][["Order Quantity", "Sales","Discount", "Profit"]])
         print("The Order Quantity, Sales, Discount and  Profit of orders with discount > 10%:")
         print(orders[orders["Discount"] > 0.1][["Order Quantity", "Sales","Discount", "Profit"]])
```

```
The Order Quantity column (top 5)
Row ID
1      6
49    49
50    27
80    30
85    19
Name: Order Quantity, dtype: int64

The Sales column (top 10)
Row ID
1        261.5400
49     10123.0200
50       244.5700
80      4965.7595
85       394.2700
86       146.6900
97        93.5400
98       905.0800
```

```
103     2781.8200
107      228.4100
Name: Sales, dtype: float64

The row with ID:50
Order ID                                                          293
Order Date                                                  10/1/2012
Order Priority                                                   High
Order Quantity                                                     27
Sales                                                          244.57
Discount                                                         0.01
Ship Mode                                                 Regular Air
Profit                                                         46.71
Unit Price                                                      8.69
Shipping Cost                                                   2.99
Customer Name                                             Barry French
Province                                                       Nunavut
Region                                                         Nunavut
Customer Segment                                              Consumer
Product Category                                      Office Supplies
Product Sub-Category                    Binders and Binder Accessories
Product Name            Cardinal Slant-D® Ring Binder, Heavy Gauge Vinyl
Product Container                                          Small Box
Product Base Margin                                              0.39
Ship Date                                                  10/3/2012
Name: 50, dtype: object

The third row:
Order ID                                      483
Order Date                                7/10/2011
Order Priority                                High
Order Quantity                                  30
Sales                                      4965.76
Discount                                      0.08
Ship Mode                                Regular Air
Profit                                      1198.97
Unit Price                                   195.99
Shipping Cost                                  3.99
Customer Name                          Clay Rozendal
Province                                    Nunavut
Region                                      Nunavut
Customer Segment                          Corporate
Product Category                         Technology
Product Sub-Category    Telephones and Communication
Product Name                                   R380
Product Container                       Small Box
Product Base Margin                            0.58
Ship Date                                 7/12/2011
Name: 80, dtype: object
The Order Quantity, Sales, Discount and Profit of the 2nd,4th, 6th and 8th row:
        Order Quantity       Sales  Discount   Profit
Row ID
49                  49  10123.0200      0.07   457.81
80                  30   4965.7595      0.08  1198.97
86                  21    146.6900      0.05     4.43
98                  22    905.0800      0.09   127.70
The Order Quantity, Sales, Discount and  Profit of orders with discount > 10%:
        Order Quantity    Sales  Discount  Profit
Row ID
```

```
176                 11  663.784      0.25 -481.04
3721                22  338.520      0.21  -17.75
37                  43  586.110      0.11   98.44
2234                 1   27.960      0.17   -9.13
4900                49  651.900      0.16  -74.51
```

9.6.4 Broadcasting, filtering and computing stats

These work pretty much like on Series and pandas takes care of adding NaNs when it cannot perform some operations due to missing values and so on.

Obviously some operators, when applied to entire tables might not always make sense (like mean or sum of strings).

```
In [18]: import pandas as pd

         orders = pd.read_csv("file_samples/sampledata_orders.csv", sep=",", index_col =0, header=0)

         orders_10 = orders[["Sales","Profit", "Product Category"]].head(10)
         orders_5 = orders[["Sales","Profit", "Product Category"]].head()
         orders_20 = orders[["Sales","Profit", "Product Category"]].head(20)
         print(orders_20)
         print("")

         #Summing over the entries, does not make sense but...
         print("Top10 + Top5:")
         print(orders_10 + orders_5)
         print("")

         prod_cost = orders_20["Sales"] - orders_20["Profit"]
         print(prod_cost)

         print("")
         print("Technology orders")
         print(orders_20[orders_20["Product Category"] == "Technology"])
         print("")
         print("Office Supplies positive profit")
         print(orders_20[ (orders_20["Product Category"] == "Office Supplies") & (orders_20["Profit"]
```

```
             Sales   Profit Product Category
Row ID
1          261.5400  -213.25  Office Supplies
49       10123.0200   457.81  Office Supplies
50         244.5700    46.71  Office Supplies
80        4965.7595  1198.97       Technology
85         394.2700    30.94  Office Supplies
86         146.6900     4.43        Furniture
97          93.5400   -54.04  Office Supplies
98         905.0800   127.70  Office Supplies
103       2781.8200  -695.26  Office Supplies
107        228.4100  -226.36  Office Supplies
127        196.8500  -166.85  Office Supplies
128        124.5600   -14.33  Office Supplies
134        716.8400   134.72  Office Supplies
135       1474.3300   114.46       Technology
149         80.6100    -4.72  Office Supplies
160       1815.4900   782.91        Furniture
161        248.2600    93.80  Office Supplies
175       4462.2300   440.72        Furniture
176        663.7840  -481.04        Furniture
```

```
203       834.9040   -11.68       Technology

Top10 + Top5:
            Sales   Profit                Product Category
Row ID
1         523.080  -426.50  Office SuppliesOffice Supplies
49      20246.040   915.62  Office SuppliesOffice Supplies
50        489.140    93.42  Office SuppliesOffice Supplies
80       9931.519  2397.94            TechnologyTechnology
85        788.540    61.88  Office SuppliesOffice Supplies
86            NaN      NaN                             NaN
97            NaN      NaN                             NaN
98            NaN      NaN                             NaN
103           NaN      NaN                             NaN
107           NaN      NaN                             NaN

Row ID
1       474.7900
49     9665.2100
50      197.8600
80     3766.7895
85      363.3300
86      142.2600
97      147.5800
98      777.3800
103    3477.0800
107     454.7700
127     363.7000
128     138.8900
134     582.1200
135    1359.8700
149      85.3300
160    1032.5800
161     154.4600
175    4021.5100
176    1144.8240
203     846.5840
dtype: float64

Technology orders
            Sales   Profit Product Category
Row ID
80       4965.7595  1198.97       Technology
135      1474.3300   114.46       Technology
203       834.9040   -11.68       Technology

Office Supplies positive profit
           Sales  Profit Product Category
Row ID
49      10123.02  457.81  Office Supplies
50        244.57   46.71  Office Supplies
85        394.27   30.94  Office Supplies
98        905.08  127.70  Office Supplies
134       716.84  134.72  Office Supplies
161       248.26   93.80  Office Supplies
```

Statistics can be computed by **column** (normally the default) or by **row** (specifying `axis=1`) or on the **entire table**.

Here[50] you can find the complete list of methods that can be applied.

```
In [19]: import pandas as pd

         data = pd.read_csv("file_samples/random.csv", sep=",", header=0)

         print("Global description")
         print(data.describe())
         print("")

         print(data)
         print("")

         print("Let's reduce A in [0,1]")
         print(data["A"]/ data["A"].max())

         print("Let's reduce first row in [0,1]")
         print(data.iloc[0]/ data.loc[0].max())

         print("Mean and std values (by column)")
         print(data.mean())
         print(data.std())
         print("")
         print("Mean and std values (by column) - top 10 values")
         print(data.head(10).mean(axis=1))
         print(data.head(10).std(axis=1))

         print("Cumulative sum (by column)- top 10 rows")
         print(data.head(10).cumsum())

         print("Cumulative sum (by row) - top 10 rows")
         print(data.head(10).cumsum(axis = 1 ))
```

```
Global description
                A          B           C          D           E          F  \
count  50.000000  50.000000   50.000000  50.000000   50.000000  50.000000
mean   45.240000  49.760000   54.120000  50.500000   57.860000  49.180000
std    28.393417  32.193015   25.165282  28.321226   32.443364  28.488229
min     0.000000   0.000000    5.000000   1.000000    2.000000   1.000000
25%    21.750000  18.000000   35.750000  28.250000   29.250000  25.000000
50%    42.000000  49.500000   52.500000  49.500000   68.000000  52.000000
75%    63.250000  76.250000   73.750000  76.000000   86.750000  75.500000
max    99.000000  99.000000  100.000000  95.000000  100.000000  97.000000

                G         H          I          L
count   50.000000  50.00000  50.000000   50.000000
mean    49.780000  47.78000  46.920000   51.780000
std     29.673248  29.75978  31.056788   30.613649
min      1.000000   1.00000   0.000000    3.000000
25%     26.500000  24.50000  19.000000   26.250000
50%     51.000000  42.50000  44.000000   46.500000
75%     76.750000  72.00000  73.000000   82.000000
max    100.000000  99.00000  98.000000  100.000000

     A   B    C   D    E   F    G   H   I    L
0   28  69    5  41   12  62   90  32  33   85
1   91  59   64  94    4  51   45  52  96    5
2   54  74   53  92   20  84   85  21  98   73
```

[50] http://pandas.pydata.org/pandas-docs/version/0.20/api.html#api-dataframe-stats

```
3   99   0   39  12   90  15   62  38   4   67
4   98  10   35  77   46  97   88  72  15   37
5   44   9   91  57   98  63   52  77  62    7
6   76  17   12  41   69  34  100  29   0   91
7   52  25   43  59   87  91    1  90  23   90
8   56  33   38  91   13   1   34  17  22   82
9   29  18   42  48   34  16   64   7  46    9
10  81  77   38  93   63  90   57  58  52  100
11  57  97   23  59   38  93   46  49  88   86
12  57   0   79  78  100  76   16  31  79    8
13  18   1   67  44    2  17   53  51   6   23
14  77  95   94  88    9  25   54  23  36   35
15  87  42   63  33   95  53   11  58  53    3
16  34  24   24  28   74  79   86  98  42   90
17   1  31   20  63   70   3   29  39   5   87
18  35  17   77  15    9  63   86  95  95   21
19   0  85   13  13   60  74   44  78   7   56
20  61  20   47  21   85  78   53  14  37   70
21  81  15   89  54   38  59   72  35  59   12
22  20  81   76  15   64  18    9  47  11   81
23  57  94   73  19   21  79   81  45  71   40
24   2  58   83  67   99  46   76  70  50   45
25  64  13   52  60   32  70   32   4  73   20
26  27  49  100  48   70  30   58  48  42   36
27  82  89   93  73   30  64   18  54  93   82
28  19  85   98   7   84  71   77  20  18   63
29  34  95   68  23   68  38   73   1  28   10
30  33  60   66  80   94  41   53  99  29   57
31   0  99   18   2   29  28   11  30  36   56
32  20  61   33  89   52  59   50  55   4   33
33  82  72   41  33   92  16   18  99  73   91
34   8  28   22  29   83  53   47  10  42   34
35  64  50   24  59    6  83   92  91  87   98
36  42  73   50  17   91   9   39  37  97   50
37  27  49   67  61   21  25   94  96  16   98
38  44  44   48  70   86  85    9  31   4   46
39  36  93   64   1   96  89   14  75  10   84
40  20  13   34  87   11  10    2   5  91   27
41  54   2   61  80   98  53   96  40   2   26
42  42  88   79  31   41  23   50  95  77   26
43   7  14   56  95   70  28   78  31  34   95
44  40  65   34  44   97  36    3  90  55   13
45  97  61   45  32   91  26   84  33  57   80
46  60  49   47  92   81  51    7  18  66   47
47  39  98   90  33   68   5   26  72  76   37
48   6  18   74  51   77  19   28  21  81   46
49  20  69   54  26   25  80   36   8  65   31

Let's reduce A in [0,1]
0     0.282828
1     0.919192
2     0.545455
3     1.000000
4     0.989899
5     0.444444
6     0.767677
7     0.525253
8     0.565657
9     0.292929
```

```
10    0.818182
11    0.575758
12    0.575758
13    0.181818
14    0.777778
15    0.878788
16    0.343434
17    0.010101
18    0.353535
19    0.000000
20    0.616162
21    0.818182
22    0.202020
23    0.575758
24    0.020202
25    0.646465
26    0.272727
27    0.828283
28    0.191919
29    0.343434
30    0.333333
31    0.000000
32    0.202020
33    0.828283
34    0.080808
35    0.646465
36    0.424242
37    0.272727
38    0.444444
39    0.363636
40    0.202020
41    0.545455
42    0.424242
43    0.070707
44    0.404040
45    0.979798
46    0.606061
47    0.393939
48    0.060606
49    0.202020
Name: A, dtype: float64
Let's reduce first row in [0,1]
A    0.311111
B    0.766667
C    0.055556
D    0.455556
E    0.133333
F    0.688889
G    1.000000
H    0.355556
I    0.366667
L    0.944444
Name: 0, dtype: float64
Mean and std values (by column)
A    45.24
B    49.76
C    54.12
D    50.50
E    57.86
```

```
F    49.18
G    49.78
H    47.78
I    46.92
L    51.78
dtype: float64
A    28.393417
B    32.193015
C    25.165282
D    28.321226
E    32.443364
F    28.488229
G    29.673248
H    29.759780
I    31.056788
L    30.613649
dtype: float64

Mean and std values (by column) - top 10 values
0    45.7
1    56.1
2    65.4
3    42.6
4    57.5
5    56.0
6    46.9
7    56.1
8    38.7
9    31.3
dtype: float64
0    29.424291
1    33.013297
2    27.785688
3    35.647035
4    33.069792
5    30.342489
6    34.853503
7    32.942543
8    29.431842
9    18.826990
dtype: float64
Cumulative sum (by column)- top 10 rows
     A    B    C    D    E    F    G    H    I    L
0   28   69    5   41   12   62   90   32   33   85
1  119  128   69  135   16  113  135   84  129   90
2  173  202  122  227   36  197  220  105  227  163
3  272  202  161  239  126  212  282  143  231  230
4  370  212  196  316  172  309  370  215  246  267
5  414  221  287  373  270  372  422  292  308  274
6  490  238  299  414  339  406  522  321  308  365
7  542  263  342  473  426  497  523  411  331  455
8  598  296  380  564  439  498  557  428  353  537
9  627  314  422  612  473  514  621  435  399  546
Cumulative sum (by row) - top 10 rows
    A    B    C    D    E    F    G    H    I    L
0  28   97  102  143  155  217  307  339  372  457
1  91  150  214  308  312  363  408  460  556  561
2  54  128  181  273  293  377  462  483  581  654
3  99   99  138  150  240  255  317  355  359  426
```

```
4   98   108   143   220   266   363   451   523   538   575
5   44    53   144   201   299   362   414   491   553   560
6   76    93   105   146   215   249   349   378   378   469
7   52    77   120   179   266   357   358   448   471   561
8   56    89   127   218   231   232   266   283   305   387
9   29    47    89   137   171   187   251   258   304   313
```

9.6.5 Merging DataFrames

It is possible to merge together DataFrames having a common column name. The merge can be done with the pandas `merge` method. Upon merging, the two tables will be concatenated into a bigger one containing information from both DataFrames. The basic syntax is:

```
pandas.merge(DataFrame1, DataFrame2, on="col_name", how="inner/outer/left/right")
```

The column on which perform the merge is specified with the parameter `on` followed by the column name, while the behaviour of the merging depends on the parameter `how` that tells pandas how to behave when dealing with non-matching data. In particular:

```
1. how = inner : non-matching entries are discarded;
2. how = left : ids are taken from the first DataFrame;
3. how = right : ids are taken from the second DataFrame;
4. how = outer : ids from both are retained.
```

```
In [20]: import pandas as pd

         Snp1 = {"id": ["SNP_FB_0411211","SNP_FB_0412425","SNP_FB_0942385",
                        "CH01f09","Hi05f12x","SNP_FB_0942712" ],
                 "type" : ["SNP","SNP","SNP","SSR","SSR","SNP"]
                }
         Snp2 = {"id" : ["SNP_FB_0411211","SNP_FB_0412425",
                         "SNP_FB_0942385", "CH01f09","SNP_FB_0428218" ],
                 "chr" : ["1", "15","7","9","1"]
                }

         DFs1 = pd.DataFrame(Snp1)
         DFs2 = pd.DataFrame(Snp2)

         print(DFs1)
         print(DFs2)
         print("")
         print("Inner merge (only common in both)")
         inJ = pd.merge(DFs1,DFs2, on = "id", how = "inner")
         print(inJ)
         print("")
         print("Left merge (IDS from DFs1)")
         leftJ = pd.merge(DFs1,DFs2, on = "id", how = "left")
         print(leftJ)
         print("")
         print("Right merge (IDS from DFs2)")
         rightJ = pd.merge(DFs1,DFs2, on = "id", how = "right")
         print(rightJ)
         print("")
         print("Outer merge (IDS from both)")
         outJ = pd.merge(DFs1,DFs2, on = "id", how = "outer")
         print(outJ)
```

```
               id type
0  SNP_FB_0411211  SNP
1  SNP_FB_0412425  SNP
2  SNP_FB_0942385  SNP
3         CH01f09  SSR
4        Hi05f12x  SSR
5  SNP_FB_0942712  SNP
  chr              id
0   1  SNP_FB_0411211
1  15  SNP_FB_0412425
2   7  SNP_FB_0942385
3   9         CH01f09
4   1  SNP_FB_0428218

Inner merge (only common in both)
               id type chr
0  SNP_FB_0411211  SNP   1
1  SNP_FB_0412425  SNP  15
2  SNP_FB_0942385  SNP   7
3         CH01f09  SSR   9

Left merge (IDS from DFs1)
               id type  chr
0  SNP_FB_0411211  SNP    1
1  SNP_FB_0412425  SNP   15
2  SNP_FB_0942385  SNP    7
3         CH01f09  SSR    9
4        Hi05f12x  SSR  NaN
5  SNP_FB_0942712  SNP  NaN

Right merge (IDS from DFs2)
               id type chr
0  SNP_FB_0411211  SNP   1
1  SNP_FB_0412425  SNP  15
2  SNP_FB_0942385  SNP   7
3         CH01f09  SSR   9
4  SNP_FB_0428218  NaN   1

Outer merge (IDS from both)
               id type  chr
0  SNP_FB_0411211  SNP    1
1  SNP_FB_0412425  SNP   15
2  SNP_FB_0942385  SNP    7
3         CH01f09  SSR    9
4        Hi05f12x  SSR  NaN
5  SNP_FB_0942712  SNP  NaN
6  SNP_FB_0428218  NaN    1
```

Note that the columns we merge on do not necessarily need to be the same, we can specify a correspondence between the row of the first dataframe (the one on the left) and the second dataframe (the one on the right) specifying which columns must have the same values to perform the merge. This can be done by using the parameters `right_on = column_name` and `left_on = column_name`:

```
In [9]: %reset -s -f

        import pandas as pd

        d = dict({"A" : [1,2,3,4], "B" : [3,4,73,13]})
        d2 = dict({"E" : [1,4,3,13], "F" : [3,1,71,1]})
```

```
        DF = pd.DataFrame(d)
        DF2 = pd.DataFrame(d2)

        merged_onBE = DF.merge(DF2, left_on = 'B', right_on = 'E', how = "inner")
        merged_onAF = DF.merge(DF2, right_on = "F", left_on = "A", how = "outer")
        print("DF:")
        print(DF)
        print("DF2:")
        print(DF2)
        print("\ninner merge on BE")
        print(merged_onBE)
        print("\nouter merge on AF:")
        print(merged_onAF)
```

```
DF:
   A   B
0  1   3
1  2   4
2  3  73
3  4  13
DF2:
    E   F
0   1   3
1   4   1
2   3  71
3  13   1

inner merge on BE
   A   B   E   F
0  1   3   3  71
1  2   4   4   1
2  4  13  13   1

outer merge on AF:
     A     B     E     F
0  1.0   3.0   4.0   1.0
1  1.0   3.0  13.0   1.0
2  2.0   4.0   NaN   NaN
3  3.0  73.0   1.0   3.0
4  4.0  13.0   NaN   NaN
5  NaN   NaN   3.0  71.0
```

9.6.6 Grouping

Given a DataFrame, it is possible to apply the so called **split-apply-aggregate** processing method. Basically, this helps to group the data according to the value of some columns, apply a function on the different groups and aggregate the results in a new DataFrame.

Grouping of a DataFrame can be done by using the **DataFrame.groupby(columnName)** method which returns a `pandas.DataFrameGroupBy` object, which basically is composed of two information the **name** shared by each group and a **DataFrame** containing all the elements belonging to that group.

Given a grouped DataFrame, each group can be obtained with the **get_group(groupName)** it is also possible to loop over the groups:

```
for groupName, groupDataFrame in grouped:
    #do something
```

It is also possible to aggregate the groups by applying some functions to each of them (please refer to the documentation[51]).

Let's see these things in action.

Example Given a list of labels and integers group them by label and compute the mean of all values in each group.

```
In [18]: import pandas as pd

         test = {"x": ["a","a", "b", "b", "c", "c" ],
                 "y" : [2,4,0,5,5,10]
                }

         DF = pd.DataFrame(test)
         print(DF)
         print("")
         gDF = DF.groupby("x")
         for i,g in gDF:
             print("Group: ", i)
             print(g)
             print(type(g))

         aggDF = gDF.aggregate(pd.DataFrame.mean)
         print(aggDF)
```

```
   x   y
0  a   2
1  a   4
2  b   0
3  b   5
4  c   5
5  c  10

Group:  a
   x  y
0  a  2
1  a  4
<class 'pandas.core.frame.DataFrame'>
Group:  b
   x  y
2  b  0
3  b  5
<class 'pandas.core.frame.DataFrame'>
Group:  c
   x   y
4  c   5
5  c  10
<class 'pandas.core.frame.DataFrame'>
     y
x
a  3.0
b  2.5
c  7.5
```

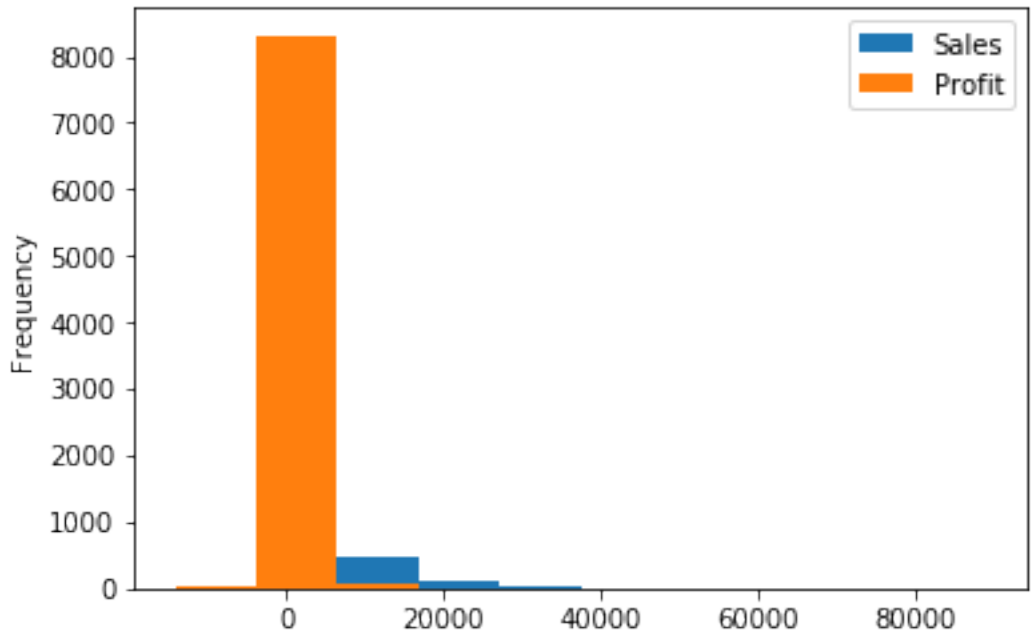
Example Given the orders data seen above, let's get the columns "Sales", "Profit" and "Product Category" and com-

[51] http://pandas.pydata.org/pandas-docs/version/0.20/api.html#api-dataframe-stats

pute the total sum and the mean of the sales and profits for each product category.

```
In [21]: import pandas as pd
         import matplotlib.pyplot as plt

         orders = pd.read_csv("file_samples/sampledata_orders.csv", sep=",",
                              index_col =0, header=0)

         SPC = orders[["Sales","Profit", "Product Category"]]
         print(SPC.head())

         SPC.plot(kind = "hist", bins = 10)
         plt.show()

         print("")
         grouped = SPC.groupby("Product Category")
         for i,g in grouped:
             print("Group: ", i)

         print("")
         print("Count elements per category:") #get the series corresponding to the column
                                               #and apply the value_counts() method
         print(orders["Product Category"].value_counts())
         print("")
         print("Total values:")
         print(grouped.aggregate(pd.DataFrame.sum)[["Sales"]])

         print("Mean values (sorted by profit):")
         mv_sorted = grouped.aggregate(pd.DataFrame.mean).sort_values(by="Profit")
         print(mv_sorted)
         print("")
         print("The most profitable is {}".format(mv_sorted.index[-1]))
```

```
             Sales   Profit Product Category
Row ID
1         261.5400  -213.25  Office Supplies
49      10123.0200   457.81  Office Supplies
50        244.5700    46.71  Office Supplies
80       4965.7595  1198.97       Technology
85        394.2700    30.94  Office Supplies
```



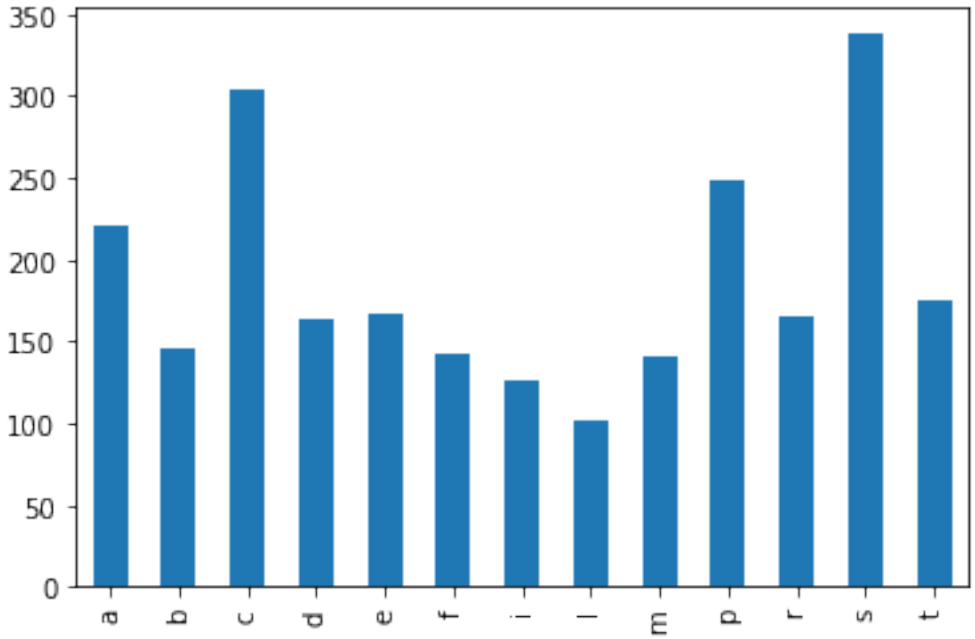

```
Group:  Furniture
Group:  Office Supplies
Group:  Technology

Count elements per category:
Office Supplies    4610
Technology         2065
Furniture          1724
Name: Product Category, dtype: int64

Total values:
                         Sales
Product Category
Furniture          5178590.542
Office Supplies    3752762.100
Technology         5984248.182
Mean values (sorted by profit):
                         Sales      Profit
Product Category
Furniture          3003.822820   68.116607
Office Supplies     814.048178  112.369072
Technology         2897.941008  429.207516

The most profitable is Technology
```

9.7 Exercises

1. The file top_3000_words.txt is a one-column file representing the top 3000 English words. Read the file and for each letter, count how many words start with that letter. Store this information in a dictionary. Create a pandas series from the dictionary and plot an histogram of all initials counting more than 100 words starting with them.

Show/Hide Solution

```
In [9]: import pandas as pd
        import matplotlib.pyplot as plt


        def readFile(file):
            fh = open(file, "r")
            initials = dict()
            for line in fh:
                line = line.strip().lower()
                init = line[0]
                if init in initials:
                    initials[init] += 1
                else:
                    initials[init] = 1
            fh.close()

            return initials

        def filter_and_plot(data, minCount = 10):
            series = pd.Series(myDict)

            filt_series = series[series > minCount]
            filt_series.plot(kind = 'bar')
            plt.show()

        inFile = "file_samples/top_3000_words.txt"

        myDict = readFile(inFile)
        filter_and_plot(myDict,100)
```



2. The file filt_aligns.tsv is a tab separated value file representing alignments of paired-end reads on some apple chromosomes. Paired end reads have the property of being X bases apart from each other as they have been sequenced from the two ends of some size-selected DNA molecules.

Each line of the file has the following information `readID\tChrPE1\tAlignmentPosition1\tChrPE2\tAlignmentPosit`

The two ends of the same pair have the same readID. Load the read pairs aligning on the same chromosome into two dictionaries. The first (`inserts`) having readID as keys and the insert size (i.e. the absolute value of AlignmentPosition1 - AlignmentPosition2) as value. The second dictionary (`chrs`)will have readID as key and chromosome ID as value. Example:

```
readID Chr11 31120 Chr11 31472
readID1 Chr7 12000 Chr11 11680
```

will result in:

```
inserts = {"readID" : 352, "readID1" : 320}
chrs = {"readID" : "Chr11", "readID1" : "Chr7"}
```

Once you have the two dictionaries:

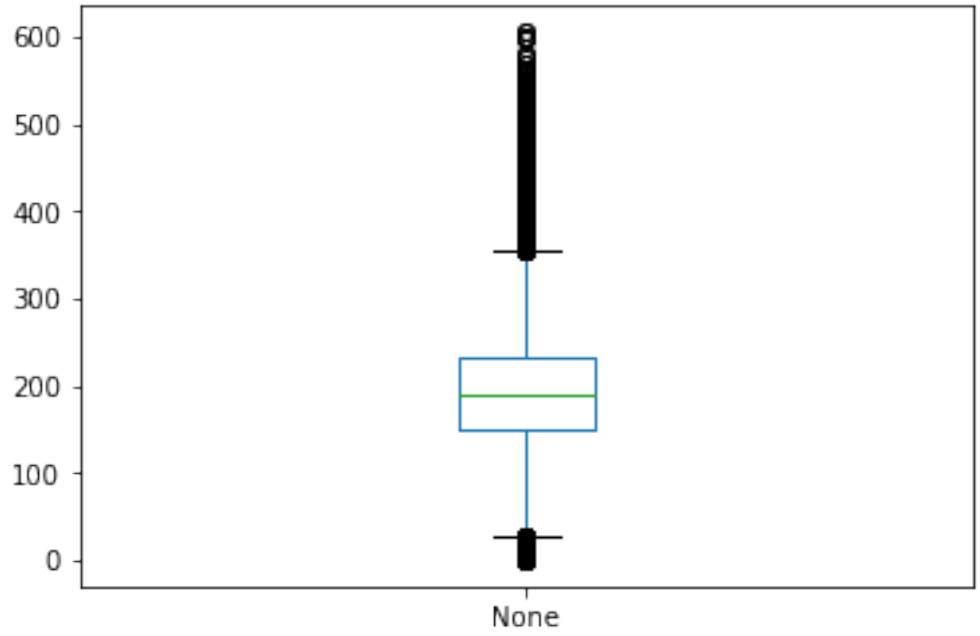
```
1. Create a Series with all the insert sizes and show some
of its stats with the method **describe**. What is the mean
insert size? How many paried end are we using to create
this distribution?
2. Display the first 5 values of the series
3. Make a box plot to assess the distribution of the values.
4. Make an histogram to see the values in a different way.
How does this distribution look like?
5. Create another series from the chromosome info and print
the first 10 elements
6. Create a DataFrame starting from the two Series and print
the first 10 elements
7. For each chromosome, get the average insert size of the
paired aligned to it (hint:use group by).
8. Make a box plot of the average insert size per chromosome.
```

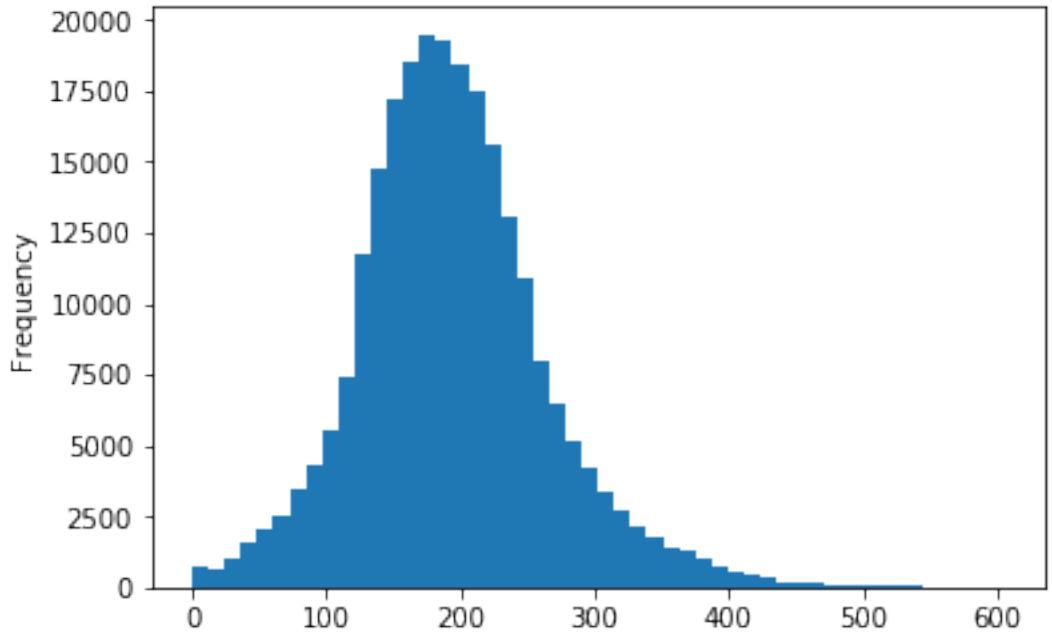
Show/Hide Solution

```
In [11]: import pandas as pd
         import matplotlib.pyplot as plt

         #Open the file and store the entries in the two dicts
         f = open("file_samples/filt_aligns.tsv")
         inserts = dict()
         chrs = dict()

         for line in f:
             line = line.strip()
             infos = line.split("\t")
             if(infos[1] == infos[3]):
                 inserts[infos[0]] = abs(int(infos[2]) - int(infos[4]))
```

```
            chrs[infos[0]] = infos[1] # or infos[3], they are the same!
        #Inserts series
        inSeries = pd.Series(inserts)
        descr = inSeries.describe()

        print(descr)
        print("")
        print("Mean insert size:", descr["mean"])
        #Question 2
        print("")
        print("First five elements:")
        print(inSeries.head(5))
        #Question 3
        inSeries.plot(kind = "box")
        plt.show()
        #Question 4
        inSeries.plot(kind = "hist", bins = 50)
        plt.show()
        #Question 5
        #Chromosomes series
        chrSeries = pd.Series(chrs)
        print(chrSeries.head(10))
        #Question 6
        DF = pd.DataFrame({"inserts": inSeries,"chrs" : chrSeries})
        print(DF.head(10))
        #Question 7
        grouped = DF.groupby("chrs")
        print("Average insert size by Chr:")
        ins_by_chr = grouped.aggregate(pd.DataFrame.mean).sort_index()
        print(grouped.aggregate(pd.DataFrame.mean).sort_index())
        #Question 8
        plt.close()
        ins_by_chr.plot(kind = "box")
        plt.show()
```

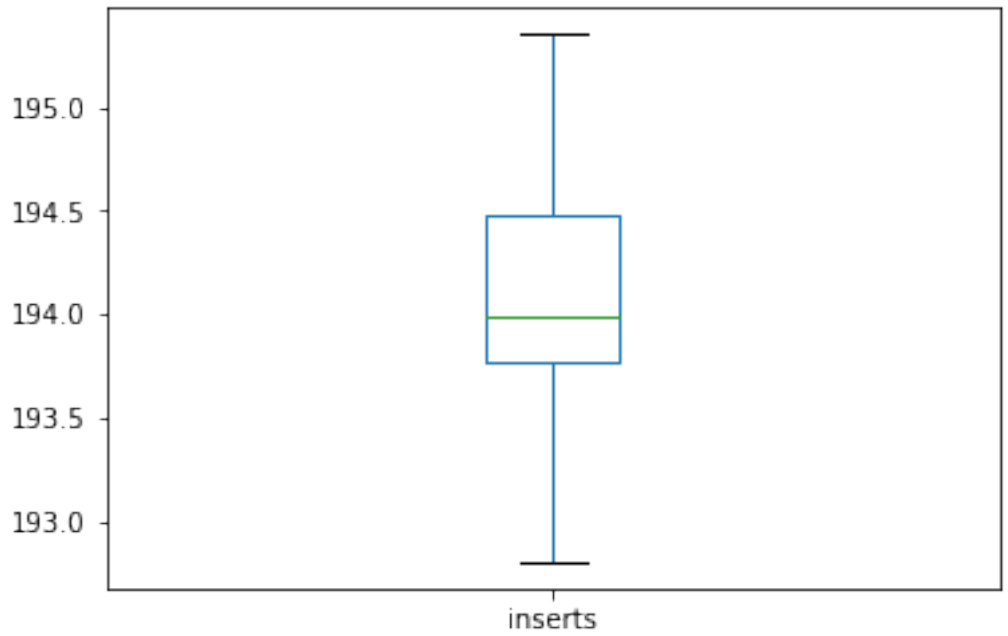
```
count    246499.000000
mean        194.126950
std          70.277551
min           0.000000
25%         150.000000
50%         189.000000
75%         232.000000
max         605.000000
dtype: float64

Mean insert size: 194.1269498050702

First five elements:
HWI-ST1296:74:C3FU9ACXX:7:2314:5661:3000    132
HWI-ST1296:74:C3FU9ACXX:6:2108:3151:3000    284
HWI-ST1296:75:C3F7CACXX:2:2302:4462:4000    294
HWI-ST1296:75:C3F7CACXX:1:1107:4328:4000      4
HWI-ST1296:75:C3F7CACXX:6:1205:3248:5000     69
dtype: int64
```






```
HWI-ST1296:74:C3FU9ACXX:7:2314:5661:3000     Chr5
HWI-ST1296:74:C3FU9ACXX:6:2108:3151:3000     Chr3
HWI-ST1296:75:C3F7CACXX:2:2302:4462:4000    Chr11
HWI-ST1296:75:C3F7CACXX:1:1107:4328:4000    Chr13
HWI-ST1296:75:C3F7CACXX:6:1205:3248:5000    Chr11
HWI-ST1296:67:C2F0FACXX:6:2305:7251:7000     Chr6
HWI-ST1296:75:C3F7CACXX:6:1203:5851:3100    Chr13
HWI-ST1296:75:C3F7CACXX:4:2309:3769:8000    Chr14
HWI-ST1296:75:C3F7CACXX:4:1212:5969:8000     Chr6
HWI-ST1296:74:C3FU9ACXX:2:2215:1855:8000     Chr4
dtype: object
```

```
                                          inserts   chrs
HWI-ST1296:74:C3FU9ACXX:7:2314:5661:3000      132   Chr5
HWI-ST1296:74:C3FU9ACXX:6:2108:3151:3000      284   Chr3
HWI-ST1296:75:C3F7CACXX:2:2302:4462:4000      294  Chr11
HWI-ST1296:75:C3F7CACXX:1:1107:4328:4000        4  Chr13
HWI-ST1296:75:C3F7CACXX:6:1205:3248:5000       69  Chr11
HWI-ST1296:67:C2F0FACXX:6:2305:7251:7000      142   Chr6
HWI-ST1296:75:C3F7CACXX:6:1203:5851:3100      197  Chr13
HWI-ST1296:75:C3F7CACXX:4:2309:3769:8000      298  Chr14
HWI-ST1296:75:C3F7CACXX:4:1212:5969:8000      369   Chr6
HWI-ST1296:74:C3FU9ACXX:2:2215:1855:8000      519   Chr4
Average insert size by Chr:
          inserts
chrs
Chr1   195.351513
Chr10  193.678891
Chr11  194.374923
Chr12  193.983594
Chr13  192.804249
Chr14  194.400139
Chr15  194.937928
Chr16  193.533066
Chr17  194.559222
Chr2   193.905932
Chr3   193.937721
Chr4   193.858397
Chr5   195.052776
Chr6   194.336027
Chr7   193.766347
Chr8   194.482513
Chr9   193.195012
```



3. Download the train.csv dataset. As the name says it is a .csv file. The file contains information regarding loans given or refused to applicants. Information on the gender, marital status, education, work and income of the applicant is reported alongside the amount and length of the loan and credit history (i.e. 0 if no previous

loan was given, 1 otherwise). Open it in a text editor or excel and inspect it first. Then, answer the following questions (if you have any doubts check here[52]):

() Load it into a pandas DataFrame (use column `Loan_ID` as index. Hint: use parm index_col).

(a) Get an idea of the data by visualizing its first 5 entries;

(b) How many total entries are present in the file? How many males and females?

(c) What is the average applicant income? Does the gender affect the income? Compute the average of the applicant income on the whole dataset and the average of the data grouped by Gender. How many Females have an income > than the average?

(d) How many loans have been given (i.e. Loan_Status equals Y)? What is the percentage of the loans given and that of the loans refused?

(e) What is the percentage of given/refused loans in the case of married people?

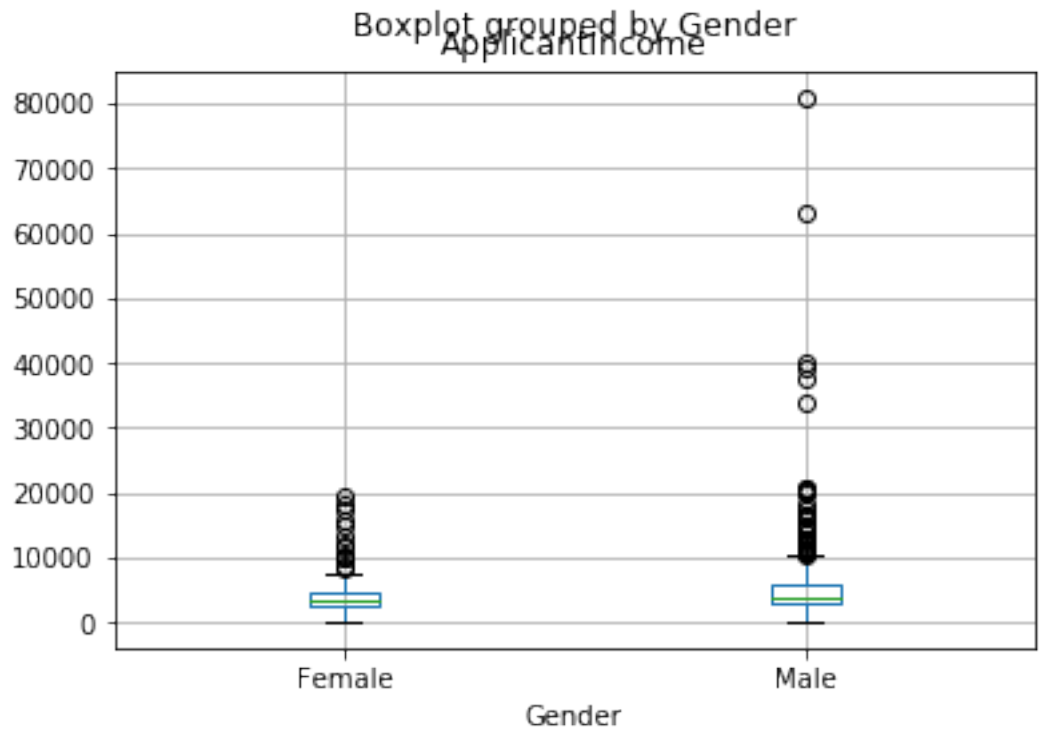
(f) What is the percentage of given/refused loans in the case of applicants with positive credit history (i.e. Credit_History equals 1)?

Show/Hide Solution

```
In [23]: import pandas as pd
         import matplotlib.pyplot as plt

         train = pd.read_csv("file_samples/train.csv", index_col = "Loan_ID")
         #Question 1:
         print(train.head())
         #Question 2:
         print("")
         entries = train.shape[0]
         print("The toal number of entries is {}".format(entries))
         print("")
         print("Males and Females")
         print(train["Gender"].value_counts())

         #Question 3:
         print("")
         avgIncome = train["ApplicantIncome"].mean()
         print("The average income is {}".format(avgIncome))
         train[["Gender", "ApplicantIncome"]].groupby("Gender").mean()
         train.boxplot(column = "ApplicantIncome", by = "Gender")
         plt.show()
         females = train[train["Gender"] == "Female"]
         fhi = females[females["ApplicantIncome"] > avgIncome][["Gender","ApplicantIncome"]]

         print("Female applicants with income higher than avg ({}): {}".format(avgIncome,len(fhi)))
         print(fhi)
         print("")
         #Question 4:
         loanSt = train["Loan_Status"].value_counts()
         print("The number of given loans is: {}".format(loanSt["Y"]))
         #loanSt is a Series, I can apply operator broadcasting on it
         #to get the percentage
         loanSt = loanSt / entries
         print("")
         print("The percentage of loans is the following")
         print(loanSt)
```

[52] http://pandas.pydata.org/pandas-docs/version/0.20/api.html

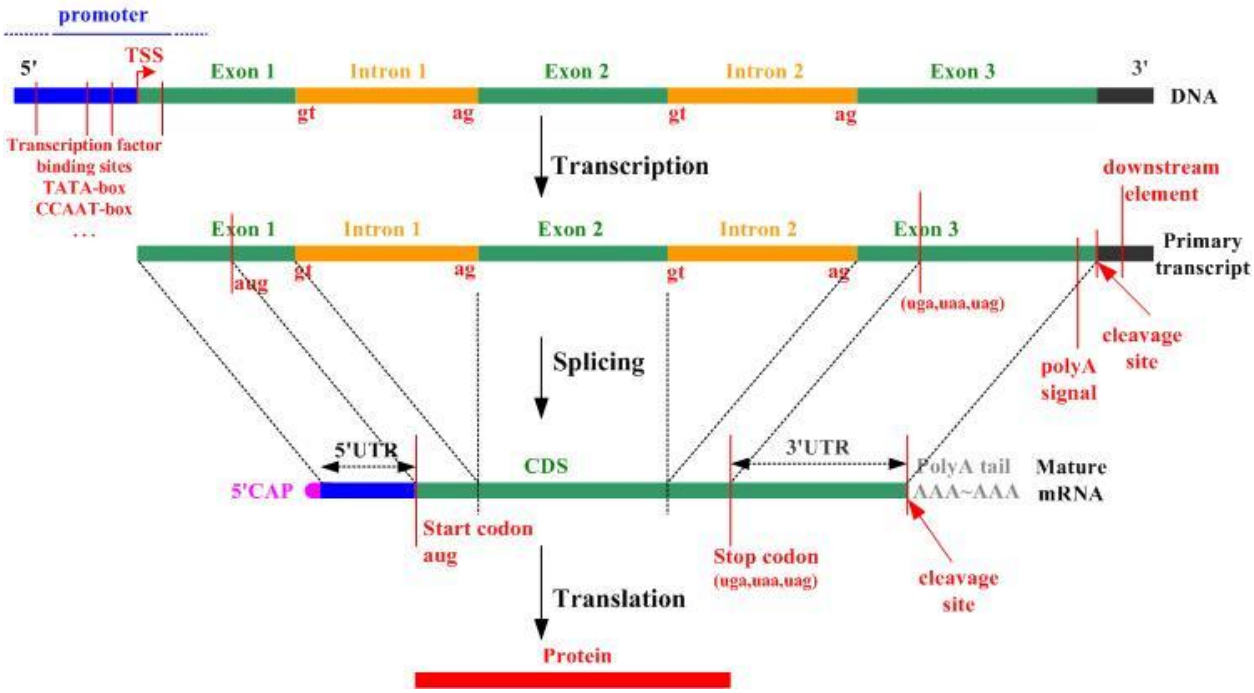
```
        #Question 5:
        loanSt =  train[train["Married"] == "Yes"]["Loan_Status"].value_counts()
        married = train["Married"][train["Married"] == "Yes"].count()
        print(married)
        print("The percentage of loans to married people ({} applicants)".format(married))
        print(loanSt/married)

        #Question 6:
        loanSt =  train[train["Credit_History"] == 1]["Loan_Status"].value_counts()
        chist = train["Credit_History"][train["Credit_History"] == 1].count()
        print(chist)
        print("The percentage of loans to applicants with credit history ({} applicants)".format(chi
        print(loanSt/chist)


        Gender Married Dependents     Education Self_Employed  \
Loan_ID
LP001002   Male      No          0      Graduate            No
LP001003   Male     Yes          1      Graduate            No
LP001005   Male     Yes          0      Graduate           Yes
LP001006   Male     Yes          0  Not Graduate            No
LP001008   Male      No          0      Graduate            No

          ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
Loan_ID
LP001002             5849                0.0         NaN             360.0
LP001003             4583             1508.0       128.0             360.0
LP001005             3000                0.0        66.0             360.0
LP001006             2583             2358.0       120.0             360.0
LP001008             6000                0.0       141.0             360.0

          Credit_History Property_Area Loan_Status
Loan_ID
LP001002             1.0         Urban           Y
LP001003             1.0         Rural           N
LP001005             1.0         Urban           Y
LP001006             1.0         Urban           Y
LP001008             1.0         Urban           Y

The toal number of entries is 614

Males and Females
Male      489
Female    112
Name: Gender, dtype: int64

The average income is 5403.459283387622

/usr/local/lib/python3.5/dist-packages/numpy/core/fromnumeric.py:57: FutureWarning: reshape is deprec
  return getattr(obj, method)(*args, **kwds)
```




```
Female applicants with income higher than avg (5403.459283387622): 21
          Gender  ApplicantIncome
Loan_ID
LP001186  Female            11500
LP001392  Female             7451
LP001422  Female            10408
LP001516  Female            14866
LP001519  Female            10000
LP001708  Female            10000
LP001776  Female             8333
LP001871  Female             7200
LP001945  Female             5417
LP002142  Female             5500
LP002194  Female            15759
LP002231  Female             6000
LP002301  Female             7441
LP002377  Female             8624
LP002393  Female            10047
LP002407  Female             7142
LP002582  Female            17263
LP002634  Female            13262
LP002731  Female            18165
LP002813  Female            19484
LP002959  Female            12000

The number of given loans is: 422

The percentage of loans is the following
Y    0.687296
N    0.312704
Name: Loan_Status, dtype: float64
398
```

```
The percentage of loans to married people (398 applicants)
Y    0.71608
N    0.28392
Name: Loan_Status, dtype: float64
475
The percentage of loans to applicants with credit history (475 applicants)
Y    0.795789
N    0.204211
Name: Loan_Status, dtype: float64
```

4. [Thanks to Stefano Teso] DNA transcription and translation into proteins follows this schema:



The file gene_table.csv is a comma separated value file representing a summary of the annotation of several human genes based on the Ensembl[53] annotation. For each gene it contains the following information:

```
gene_name,gene_biotype,chromosome,strand,transcript_count
```

where `gene_name` is based on the HGNC[54] nomenclature. `gene_biotype` represents the biotype (refer to VEGA[55] like protein_coding, pseudogene, lincRNA, miRNA etc. `chromosome` is where the feature is located, `strand` is a + or a - for the forward or reverse strand and `transcript_count` reports the number of isoforms of the gene.

A sample of the file follows:

```
TSPAN6,protein_coding,chrX,-,5
TNMD,protein_coding,chrX,+,2
DPM1,protein_coding,chr20,-,6
SCYL3,protein_coding,chr1,-,5
C1orf112,protein_coding,chr1,+,9
FGR,protein_coding,chr1,-,7
```

(continues on next page)

[53] http://www.ensembl.org/index.html
[54] http://www.genenames.org/
[55] http://vega.sanger.ac.uk/info/about/gene_and_transcript_types.html

(continued from previous page)

```
CFH,protein_coding,chr1,+,6
FUCA2,protein_coding,chr6,-,3
GCLC,protein_coding,chr6,-,13
```

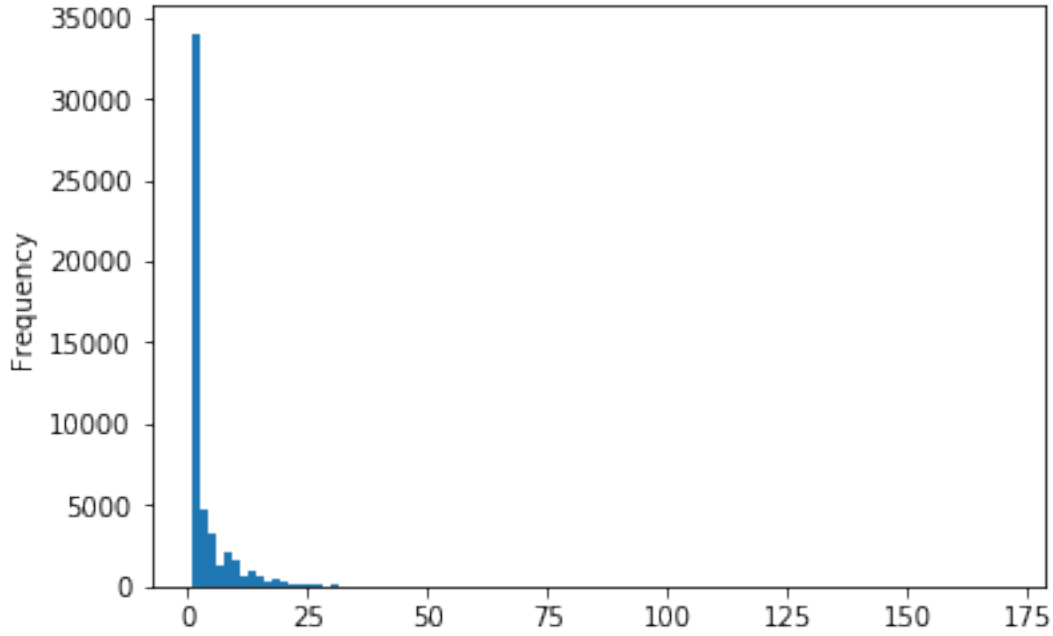
Write a python program that: 0. Loads the gene_table.csv in a DataFrame (inspect the first entries with head to check the content of the file); 1. Computes the number of genes annotated for the human genome; 2. Computes the minimum, maximum, average and median number of known isoforms per gene (consider the transcript_count column as a series). 3. Plots a histogram and a boxplot of the number of known isoforms per gene 4. Computes the number of different biotypes. How many genes do we have for each genotype? Plot the number of genes per biotype in a horizontal bar plot (hint: use also figsize = (10,10) to make it visible; 5. Computes the number of different chromosomes 6. Computes, for each chromosome, the number of genes it contains, and prints a horizontal barplot with the number of genes per chromosome. 7. Computes, for each chromosome, the percentage of genes located on the + strand 8. Computes, for each biotype, the average number of transcripts associated to genes belonging to the biotype. Finally, plots them in a vertical bar plot

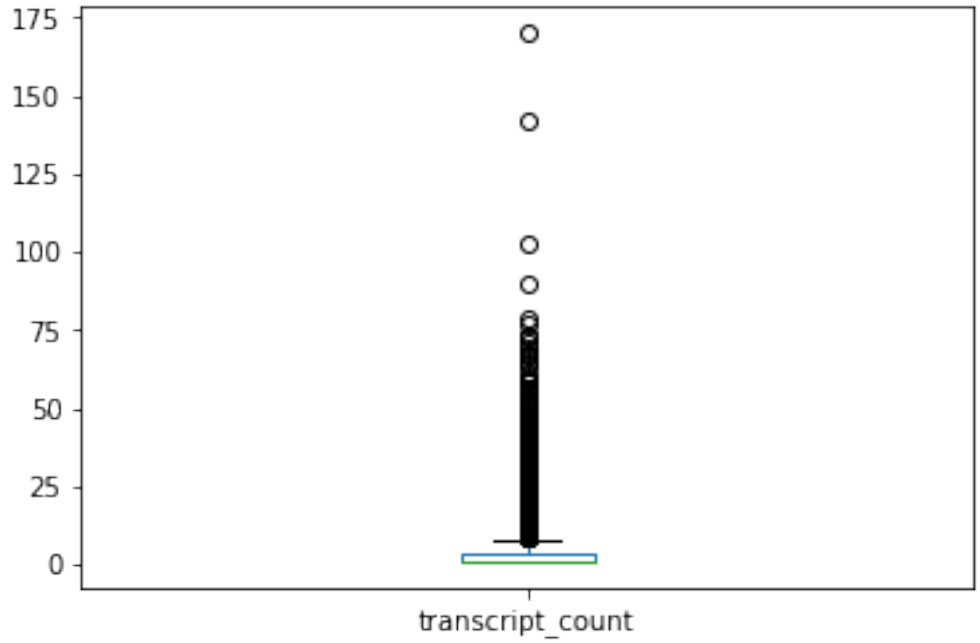
Show/Hide Solution

```
In [3]: import pandas as pd
        import matplotlib.pyplot as plt

        genes = pd.read_csv("file_samples/gene_table.csv", sep=",")
        print(genes.head())
        print("")
        #Question 1
        print("Total number of genes: {}".format(genes.shape[0]))
        #Question 2
        maxI = genes["transcript_count"].max()
        minI = genes["transcript_count"].min()
        avgI = genes["transcript_count"].mean()
        medI = genes["transcript_count"].median()
        print("")
        print("Isoform per gene stats:")
        print(" - max: {} min {} avg {} median {}".format(maxI, minI,avgI,medI))

        #Question 3
        genes["transcript_count"].plot(kind = "hist", bins = 100)
        plt.show()
        plt.close()
        genes["transcript_count"].plot(kind = "box")
        plt.show()
        print("")
        #Question 4
        biot = genes["gene_biotype"].value_counts()
        print("The number of different biotypes is", len(biot))
        print("")
        print(biot)
        plt.close()
        biot.plot(kind = "barh", figsize = (10,10))
        plt.show()
        #Question 5
        chrs = genes["chromosome"].unique()
        print("The total number of chromosomes is ", chrs.shape[0])

        #Question 6
        plt.close()
        chrs = genes["chromosome"].value_counts()
        chrs.plot(kind = "barh", figsize = (10,10))
```

```
        plt.show()

        #Question 7
        #Let's group the data by chromosome and strand
        #then loop through all the elements to count the
        #total elements per chromosome
        #to compute the percentage of the ones that are on + strand
        #The totals go on a dictionary with key chr

        grp_chrs = genes.groupby(["chromosome", "strand"])

        print(type(grp_chrs))
        for ind, grp in grp_chrs:
            print("{}({}) : {}".format(ind, type(ind), grp.shape[0]))

        tot_byChr = {}

        for ind, grp in grp_chrs:
            v = tot_byChr.get(ind[0],0)
            tot_byChr[ind[0]] = v + grp.shape[0]

        print(tot_byChr)

        for ind, grp in grp_chrs:
            if (ind[1] == "+"):

                print("% genes of {} on strand +: {:.2f}".format(ind[0], 100*grp.shape[0]/tot_byChr[



        #Question 8
        biot= grp_chrs = genes.groupby(["gene_biotype"])
        biot_transc = biot.aggregate(pd.DataFrame.mean)[["transcript_count"]]
        print(biot_transc)

        biot_transc.plot(kind = "bar", figsize = (10,10))
        plt.show()
```

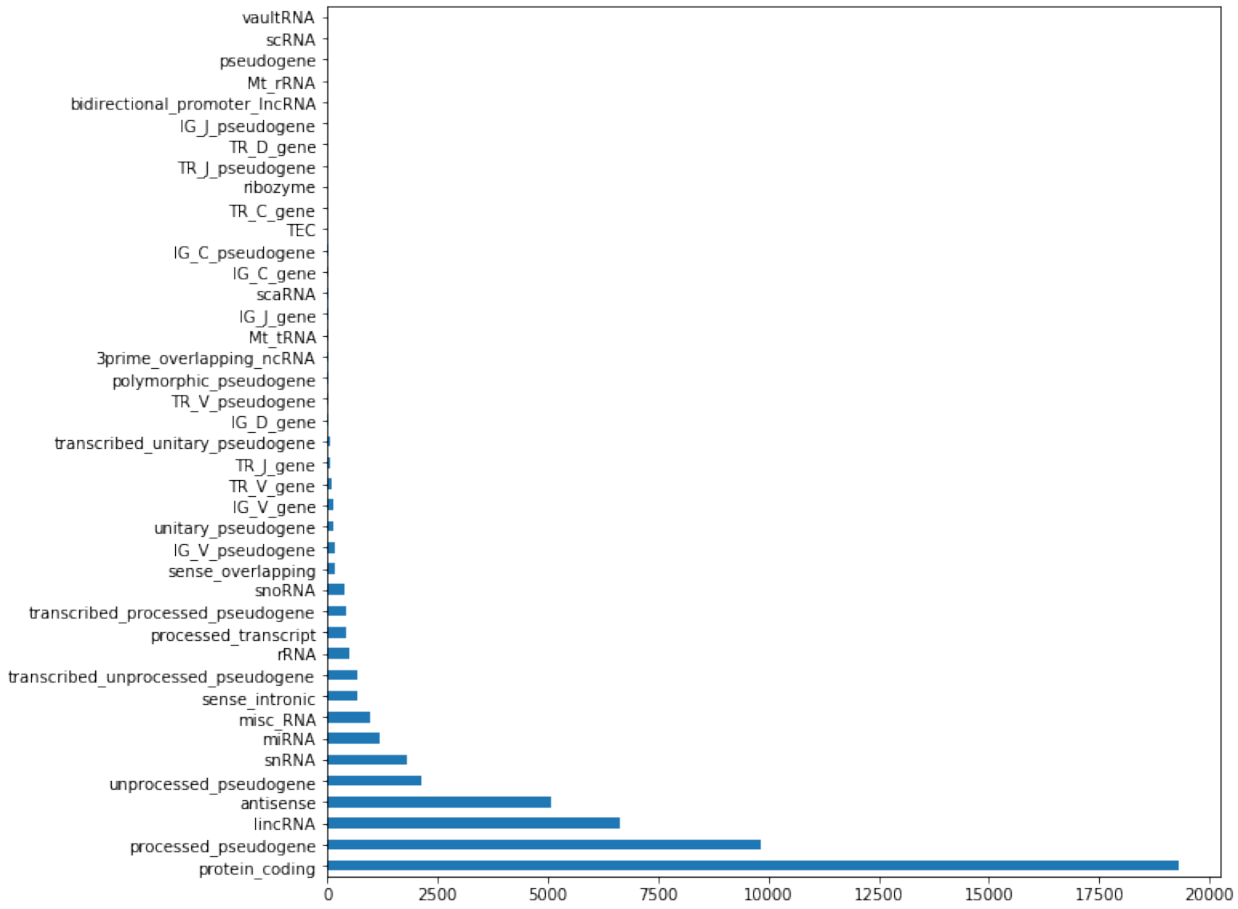
```
  gene_name    gene_biotype chromosome strand  transcript_count
0    TSPAN6  protein_coding       chrX      -                 5
1      TNMD  protein_coding       chrX      +                 2
2      DPM1  protein_coding      chr20      -                 6
3     SCYL3  protein_coding       chr1      -                 5
4  C1orf112  protein_coding       chr1      +                 9

Total number of genes: 51116

Isoform per gene stats:
 - max: 170 min 1 avg 3.7033218561702794 median 1.0
```



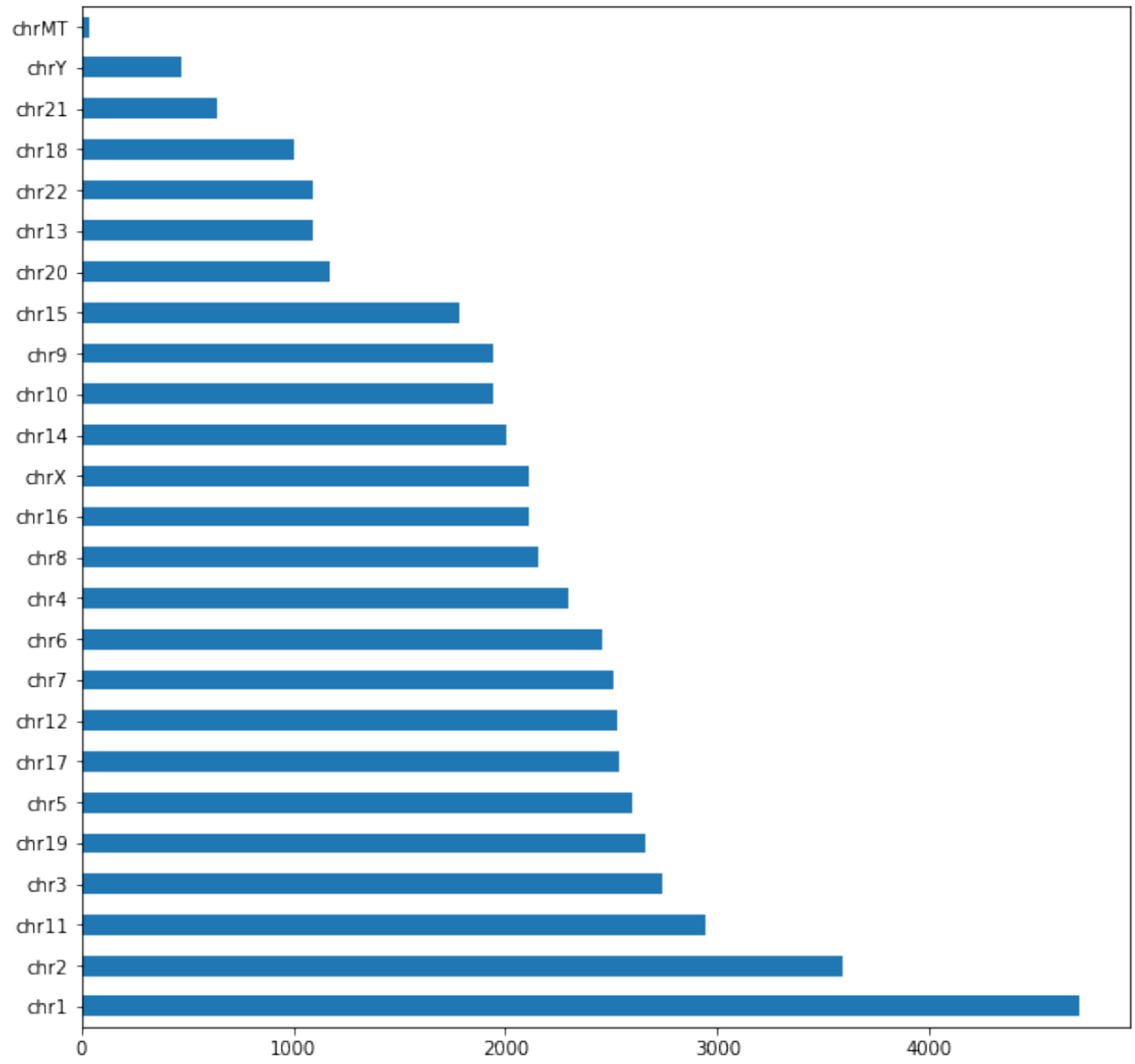



```
The number of different biotypes is 41

protein_coding                   19297
processed_pseudogene              9826
lincRNA                           6645
antisense                         5086
unprocessed_pseudogene            2143
snRNA                             1802
miRNA                             1173
misc_RNA                           955
```

```
sense_intronic                        686
transcribed_unprocessed_pseudogene    660
rRNA                                  512
processed_transcript                  438
transcribed_processed_pseudogene      425
snoRNA                                403
sense_overlapping                     174
IG_V_pseudogene                       173
unitary_pseudogene                    141
IG_V_gene                             130
TR_V_gene                              93
TR_J_gene                              71
transcribed_unitary_pseudogene         53
IG_D_gene                              33
TR_V_pseudogene                        30
polymorphic_pseudogene                 28
3prime_overlapping_ncRNA               26
Mt_tRNA                                22
IG_J_gene                              18
scaRNA                                 16
IG_C_gene                              14
IG_C_pseudogene                         9
TEC                                     6
TR_C_gene                               5
ribozyme                                5
TR_J_pseudogene                         4
TR_D_gene                               3
IG_J_pseudogene                         3
bidirectional_promoter_lncRNA           2
Mt_rRNA                                 2
pseudogene                              2
scRNA                                   1
vaultRNA                                1
Name: gene_biotype, dtype: int64
```




```
The total number of chromosomes is  25
```

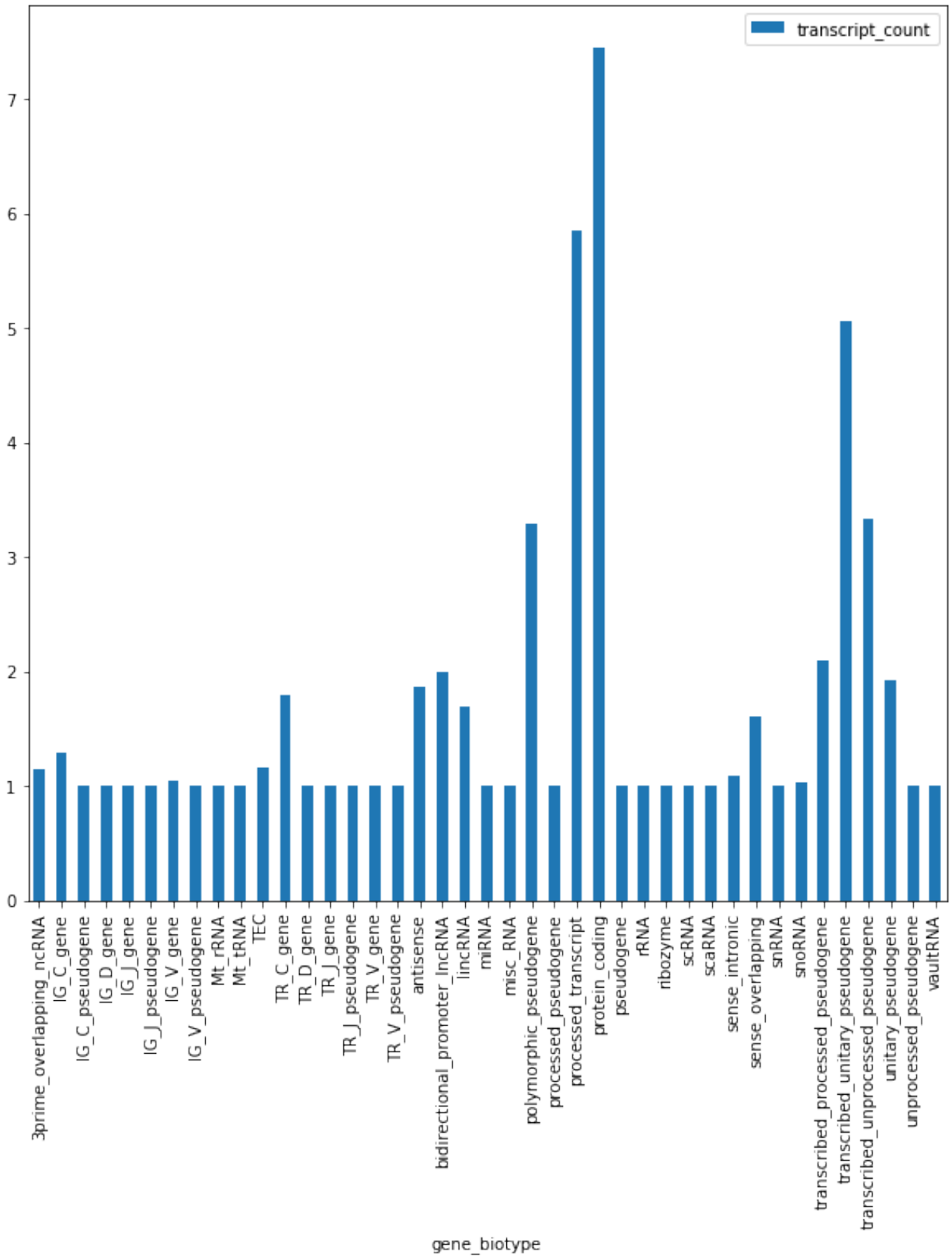



```
<class 'pandas.core.groupby.generic.DataFrameGroupBy'>
('chr1', '+')(<class 'tuple'>) : 2388
('chr1', '-')(<class 'tuple'>) : 2322
('chr10', '+')(<class 'tuple'>) : 999
('chr10', '-')(<class 'tuple'>) : 944
('chr11', '+')(<class 'tuple'>) : 1470
('chr11', '-')(<class 'tuple'>) : 1472
('chr12', '+')(<class 'tuple'>) : 1262
('chr12', '-')(<class 'tuple'>) : 1269
('chr13', '+')(<class 'tuple'>) : 537
('chr13', '-')(<class 'tuple'>) : 558
('chr14', '+')(<class 'tuple'>) : 1029
('chr14', '-')(<class 'tuple'>) : 976
('chr15', '+')(<class 'tuple'>) : 901
('chr15', '-')(<class 'tuple'>) : 883
('chr16', '+')(<class 'tuple'>) : 1077
('chr16', '-')(<class 'tuple'>) : 1036
('chr17', '+')(<class 'tuple'>) : 1227
```

```
('chr17', '-')(<class 'tuple'>) : 1306
('chr18', '+')(<class 'tuple'>) : 508
('chr18', '-')(<class 'tuple'>) : 491
('chr19', '+')(<class 'tuple'>) : 1361
('chr19', '-')(<class 'tuple'>) : 1303
('chr2', '+')(<class 'tuple'>) : 1803
('chr2', '-')(<class 'tuple'>) : 1786
('chr20', '+')(<class 'tuple'>) : 577
('chr20', '-')(<class 'tuple'>) : 592
('chr21', '+')(<class 'tuple'>) : 297
('chr21', '-')(<class 'tuple'>) : 342
('chr22', '+')(<class 'tuple'>) : 585
('chr22', '-')(<class 'tuple'>) : 505
('chr3', '+')(<class 'tuple'>) : 1345
('chr3', '-')(<class 'tuple'>) : 1395
('chr4', '+')(<class 'tuple'>) : 1152
('chr4', '-')(<class 'tuple'>) : 1145
('chr5', '+')(<class 'tuple'>) : 1303
('chr5', '-')(<class 'tuple'>) : 1295
('chr6', '+')(<class 'tuple'>) : 1222
('chr6', '-')(<class 'tuple'>) : 1237
('chr7', '+')(<class 'tuple'>) : 1320
('chr7', '-')(<class 'tuple'>) : 1187
('chr8', '+')(<class 'tuple'>) : 1039
('chr8', '-')(<class 'tuple'>) : 1113
('chr9', '+')(<class 'tuple'>) : 989
('chr9', '-')(<class 'tuple'>) : 952
('chrMT', '+')(<class 'tuple'>) : 28
('chrMT', '-')(<class 'tuple'>) : 9
('chrX', '+')(<class 'tuple'>) : 1078
('chrX', '-')(<class 'tuple'>) : 1031
('chrY', '+')(<class 'tuple'>) : 240
('chrY', '-')(<class 'tuple'>) : 230
{}
{'chr1': 2388}
{'chr1': 4710}
{'chr1': 4710, 'chr10': 999}
{'chr1': 4710, 'chr10': 1943}
{'chr1': 4710, 'chr10': 1943, 'chr11': 1470}
{'chr1': 4710, 'chr10': 1943, 'chr11': 2942}
{'chr1': 4710, 'chr10': 1943, 'chr11': 2942, 'chr12': 1262}
{'chr1': 4710, 'chr10': 1943, 'chr11': 2942, 'chr12': 2531}
{'chr1': 4710, 'chr10': 1943, 'chr11': 2942, 'chr12': 2531, 'chr13': 537}
{'chr1': 4710, 'chr10': 1943, 'chr11': 2942, 'chr12': 2531, 'chr13': 1095}
{'chr1': 4710, 'chr10': 1943, 'chr11': 2942, 'chr12': 2531, 'chr13': 1095, 'chr14': 1029}
{'chr1': 4710, 'chr10': 1943, 'chr11': 2942, 'chr12': 2531, 'chr13': 1095, 'chr14': 2005}
{'chr1': 4710, 'chr10': 1943, 'chr11': 2942, 'chr12': 2531, 'chr13': 1095, 'chr14': 2005, 'chr15': 90
{'chr1': 4710, 'chr10': 1943, 'chr11': 2942, 'chr12': 2531, 'chr13': 1095, 'chr14': 2005, 'chr15': 1
{'chr1': 4710, 'chr10': 1943, 'chr11': 2942, 'chr12': 2531, 'chr13': 1095, 'chr14': 2005, 'chr15': 1
{'chr1': 4710, 'chr10': 1943, 'chr11': 2942, 'chr12': 2531, 'chr13': 1095, 'chr14': 2005, 'chr15': 1
{'chr1': 4710, 'chr10': 1943, 'chr11': 2942, 'chr12': 2531, 'chr13': 1095, 'chr14': 2005, 'chr15': 1
{'chr1': 4710, 'chr10': 1943, 'chr11': 2942, 'chr12': 2531, 'chr13': 1095, 'chr14': 2005, 'chr15': 1
{'chr1': 4710, 'chr10': 1943, 'chr11': 2942, 'chr12': 2531, 'chr13': 1095, 'chr14': 2005, 'chr15': 1
{'chr1': 4710, 'chr10': 1943, 'chr11': 2942, 'chr12': 2531, 'chr13': 1095, 'chr14': 2005, 'chr15': 1
{'chr1': 4710, 'chr10': 1943, 'chr11': 2942, 'chr12': 2531, 'chr13': 1095, 'chr14': 2005, 'chr15': 1
{'chr1': 4710, 'chr10': 1943, 'chr11': 2942, 'chr12': 2531, 'chr13': 1095, 'chr14': 2005, 'chr15': 1
{'chr1': 4710, 'chr10': 1943, 'chr11': 2942, 'chr12': 2531, 'chr13': 1095, 'chr14': 2005, 'chr15': 1
{'chr1': 4710, 'chr10': 1943, 'chr11': 2942, 'chr12': 2531, 'chr13': 1095, 'chr14': 2005, 'chr15': 1
```

```
{'chr1': 4710, 'chr10': 1943, 'chr11': 2942, 'chr12': 2531, 'chr13': 1095, 'chr14': 2005, 'chr15': 1
{'chr1': 4710, 'chr10': 1943, 'chr11': 2942, 'chr12': 2531, 'chr13': 1095, 'chr14': 2005, 'chr15': 1
{'chr1': 4710, 'chr10': 1943, 'chr11': 2942, 'chr12': 2531, 'chr13': 1095, 'chr14': 2005, 'chr15': 1
{'chr1': 4710, 'chr10': 1943, 'chr11': 2942, 'chr12': 2531, 'chr13': 1095, 'chr14': 2005, 'chr15': 1
{'chr1': 4710, 'chr10': 1943, 'chr11': 2942, 'chr12': 2531, 'chr13': 1095, 'chr14': 2005, 'chr15': 1
{'chr1': 4710, 'chr10': 1943, 'chr11': 2942, 'chr12': 2531, 'chr13': 1095, 'chr14': 2005, 'chr15': 1
{'chr1': 4710, 'chr10': 1943, 'chr11': 2942, 'chr12': 2531, 'chr13': 1095, 'chr14': 2005, 'chr15': 1
{'chr1': 4710, 'chr10': 1943, 'chr11': 2942, 'chr12': 2531, 'chr13': 1095, 'chr14': 2005, 'chr15': 1
{'chr1': 4710, 'chr10': 1943, 'chr11': 2942, 'chr12': 2531, 'chr13': 1095, 'chr14': 2005, 'chr15': 1
{'chr1': 4710, 'chr10': 1943, 'chr11': 2942, 'chr12': 2531, 'chr13': 1095, 'chr14': 2005, 'chr15': 1
{'chr1': 4710, 'chr10': 1943, 'chr11': 2942, 'chr12': 2531, 'chr13': 1095, 'chr14': 2005, 'chr15': 1
{'chr1': 4710, 'chr10': 1943, 'chr11': 2942, 'chr12': 2531, 'chr13': 1095, 'chr14': 2005, 'chr15': 1
{'chr1': 4710, 'chr10': 1943, 'chr11': 2942, 'chr12': 2531, 'chr13': 1095, 'chr14': 2005, 'chr15': 1
{'chr1': 4710, 'chr10': 1943, 'chr11': 2942, 'chr12': 2531, 'chr13': 1095, 'chr14': 2005, 'chr15': 1
{'chr1': 4710, 'chr10': 1943, 'chr11': 2942, 'chr12': 2531, 'chr13': 1095, 'chr14': 2005, 'chr15': 1
{'chr1': 4710, 'chr10': 1943, 'chr11': 2942, 'chr12': 2531, 'chr13': 1095, 'chr14': 2005, 'chr15': 1
{'chr1': 4710, 'chr10': 1943, 'chr11': 2942, 'chr12': 2531, 'chr13': 1095, 'chr14': 2005, 'chr15': 1
{'chr1': 4710, 'chr10': 1943, 'chr11': 2942, 'chr12': 2531, 'chr13': 1095, 'chr14': 2005, 'chr15': 1
{'chr1': 4710, 'chr10': 1943, 'chr11': 2942, 'chr12': 2531, 'chr13': 1095, 'chr14': 2005, 'chr15': 1
{'chr1': 4710, 'chr10': 1943, 'chr11': 2942, 'chr12': 2531, 'chr13': 1095, 'chr14': 2005, 'chr15': 1
{'chr1': 4710, 'chr10': 1943, 'chr11': 2942, 'chr12': 2531, 'chr13': 1095, 'chr14': 2005, 'chr15': 1
{'chr1': 4710, 'chr10': 1943, 'chr11': 2942, 'chr12': 2531, 'chr13': 1095, 'chr14': 2005, 'chr15': 1
{'chr1': 4710, 'chr10': 1943, 'chr11': 2942, 'chr12': 2531, 'chr13': 1095, 'chr14': 2005, 'chr15': 1
% genes of chr1 on strand +: 50.70
% genes of chr10 on strand +: 51.42
% genes of chr11 on strand +: 49.97
% genes of chr12 on strand +: 49.86
% genes of chr13 on strand +: 49.04
% genes of chr14 on strand +: 51.32
% genes of chr15 on strand +: 50.50
% genes of chr16 on strand +: 50.97
% genes of chr17 on strand +: 48.44
% genes of chr18 on strand +: 50.85
% genes of chr19 on strand +: 51.09
% genes of chr2 on strand +: 50.24
% genes of chr20 on strand +: 49.36
% genes of chr21 on strand +: 46.48
% genes of chr22 on strand +: 53.67
% genes of chr3 on strand +: 49.09
% genes of chr4 on strand +: 50.15
% genes of chr5 on strand +: 50.15
% genes of chr6 on strand +: 49.69
% genes of chr7 on strand +: 52.65
% genes of chr8 on strand +: 48.28
% genes of chr9 on strand +: 50.95
% genes of chrMT on strand +: 75.68
% genes of chrX on strand +: 51.11
% genes of chrY on strand +: 51.06
                                  transcript_count
gene_biotype
3prime_overlapping_ncRNA                  1.153846
IG_C_gene                                 1.285714
IG_C_pseudogene                           1.000000
IG_D_gene                                 1.000000
IG_J_gene                                 1.000000
IG_J_pseudogene                           1.000000
IG_V_gene                                 1.053846
```

```
IG_V_pseudogene                    1.000000
Mt_rRNA                            1.000000
Mt_tRNA                            1.000000
TEC                                1.166667
TR_C_gene                          1.800000
TR_D_gene                          1.000000
TR_J_gene                          1.000000
TR_J_pseudogene                    1.000000
TR_V_gene                          1.010753
TR_V_pseudogene                    1.000000
antisense                          1.870232
bidirectional_promoter_lncRNA      2.000000
lincRNA                            1.690143
miRNA                              1.000000
misc_RNA                           1.000000
polymorphic_pseudogene             3.285714
processed_pseudogene               1.000204
processed_transcript               5.844749
protein_coding                     7.447893
pseudogene                         1.000000
rRNA                               1.000000
ribozyme                           1.000000
scRNA                              1.000000
scaRNA                             1.000000
sense_intronic                     1.087464
sense_overlapping                  1.609195
snRNA                              1.000000
snoRNA                             1.027295
transcribed_processed_pseudogene   2.101176
transcribed_unitary_pseudogene     5.056604
transcribed_unprocessed_pseudogene 3.339394
unitary_pseudogene                 1.921986
unprocessed_pseudogene             1.000467
vaultRNA                           1.000000
```



5. Write a function that creates and returns a data frame having columns with the labels specified through a list taken in input and ten rows of random data between 1 and 100.

(a) Create two random DataFrames one with labels l = [“A”, “B”, “C” ,”D”,”E”] and l1 = [“W”, “X”,”Y”,”Z”]

(b) Print the first 5 elements to inspect the two DataFrames and plot the valuse of the two DataFrames;

(c) Create a new Series S1 with values: [1, 1, 1, 0, 0, 1, 1, 0, 0, 1]. Get the Series corresponding to the column “C” and multiply it point-to-point by the Series S1 (hint use: Series.multiply(S1))

(d) Add this new series as a “C” column of the second DataFrame.

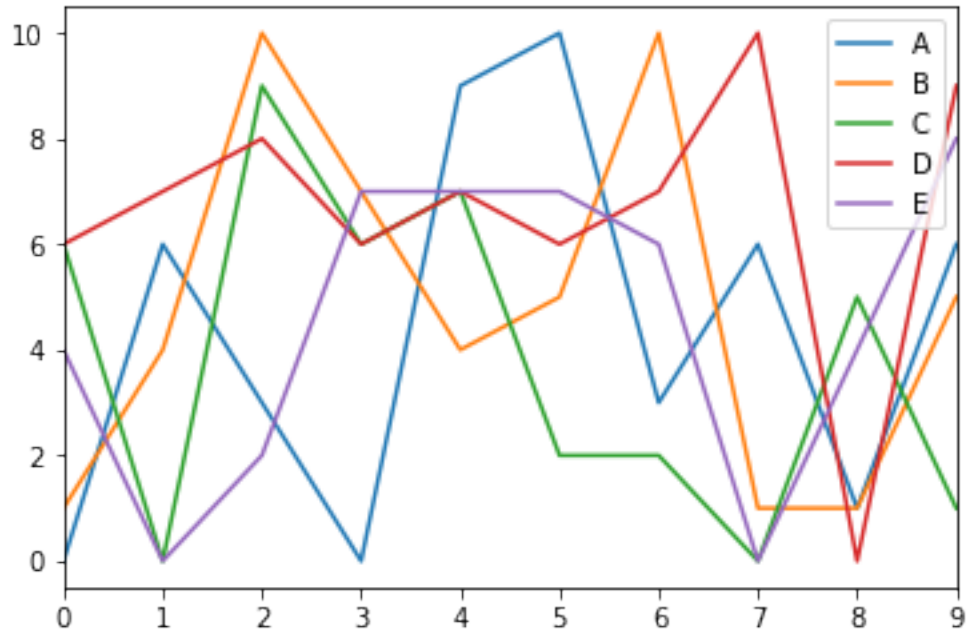
(e) Merge the two DataFrames based on the value of C. Perform an inner, outer, left and right merge and see the difference

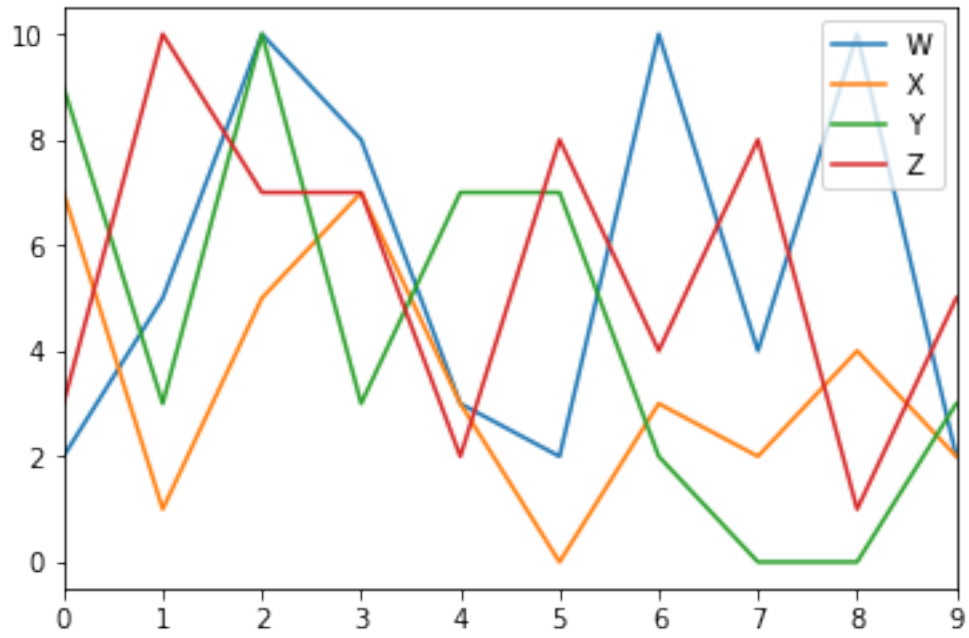
Show/Hide Solution

```
In [25]: import pandas as pd
         import matplotlib.pyplot as plt
         import random

         def createRandomDF(labels):
             """ returns a random dataframe with columns having names equal to the
             input labels"""
             tmp_dict = {}
             for l in labels:
                 tmp_dict[l] = []
                 for i in range(0,10):
                     tmp_dict[l].append(random.randint(0,10))

             DF = pd.DataFrame(tmp_dict)
             return DF

         #Question 1
         l = ["A", "B", "C" ,"D","E"]
         l1 = [ "W", "X","Y","Z"]
         df = createRandomDF(l)
         df1 = createRandomDF(l1)

         #Question 2
         print(df.head())
         print("")
         print(df1.head())
         plt.close()
         df.plot(kind = "line")
         plt.show()
         plt.close()
         df1.plot(kind = "line")
         plt.show()

         #Question 3
         S1 = pd.Series( [ 1, 1, 1, 0, 0, 1, 1, 0,0,1])
         Sc = df["C"]
         c1 = Sc.multiply(S1)
         print(c1)

         #Question 4
         df1["C"] = c1
         print(df1)
         print(df)
         #Question 5
         inJ = pd.merge(df,df1, how ="inner")
         print("")
         print("Inner Join DF - DF1")
```

```
print(inJ)
outJ = pd.merge(df,df1, how ="outer")
print("")
print("Outer Join DF - DF1")
print(outJ)
print("")
print("Left Join DF - DF1")
lJ = pd.merge(df,df1, on = "C", how ="left")
print(lJ)
print("")
print("Right Join DF - DF1")
rJ = pd.merge(df,df1, on = "C", how ="right")
print(rJ)
```

```
   A   B  C  D  E
0  0   1  6  6  4
1  6   4  0  7  0
2  3  10  9  8  2
3  0   7  6  6  7
4  9   4  7  7  7

    W  X   Y   Z
0   2  7   9   3
1   5  1   3  10
2  10  5  10   7
3   8  7   3   7
4   3  3   7   2
```






```
0    6
1    0
2    9
3    0
4    0
5    2
6    2
7    0
8    0
9    1
dtype: int64
    W  X   Y   Z  C
0   2  7   9   3  6
1   5  1   3  10  0
2  10  5  10   7  9
3   8  7   3   7  0
4   3  3   7   2  0
5   2  0   7   8  2
6  10  3   2   4  2
7   4  2   0   8  0
8  10  4   0   1  0
9   2  2   3   5  1
    A   B  C   D  E
0   0   1  6   6  4
1   6   4  0   7  0
2   3  10  9   8  2
3   0   7  6   6  7
4   9   4  7   7  7
5  10   5  2   6  7
6   3  10  2   7  6
7   6   1  0  10  0
8   1   1  5   0  4
9   6   5  1   9  8

Inner Join DF - DF1
```

```
     A   B  C   D  E   W  X   Y   Z
0    0   1  6   6  4   2  7   9   3
1    0   7  6   6  7   2  7   9   3
2    6   4  0   7  0   5  1   3  10
3    6   4  0   7  0   8  7   3   7
4    6   4  0   7  0   3  3   7   2
5    6   4  0   7  0   4  2   0   8
6    6   4  0   7  0  10  4   0   1
7    6   1  0  10  0   5  1   3  10
8    6   1  0  10  0   8  7   3   7
9    6   1  0  10  0   3  3   7   2
10   6   1  0  10  0   4  2   0   8
11   6   1  0  10  0  10  4   0   1
12   3  10  9   8  2  10  5  10   7
13  10   5  2   6  7   2  0   7   8
14  10   5  2   6  7  10  3   2   4
15   3  10  2   7  6   2  0   7   8
16   3  10  2   7  6  10  3   2   4
17   6   5  1   9  8   2  2   3   5

Outer Join DF - DF1
     A   B  C   D  E     W    X     Y     Z
0    0   1  6   6  4   2.0  7.0   9.0   3.0
1    0   7  6   6  7   2.0  7.0   9.0   3.0
2    6   4  0   7  0   5.0  1.0   3.0  10.0
3    6   4  0   7  0   8.0  7.0   3.0   7.0
4    6   4  0   7  0   3.0  3.0   7.0   2.0
5    6   4  0   7  0   4.0  2.0   0.0   8.0
6    6   4  0   7  0  10.0  4.0   0.0   1.0
7    6   1  0  10  0   5.0  1.0   3.0  10.0
8    6   1  0  10  0   8.0  7.0   3.0   7.0
9    6   1  0  10  0   3.0  3.0   7.0   2.0
10   6   1  0  10  0   4.0  2.0   0.0   8.0
11   6   1  0  10  0  10.0  4.0   0.0   1.0
12   3  10  9   8  2  10.0  5.0  10.0   7.0
13   9   4  7   7  7   NaN  NaN   NaN   NaN
14  10   5  2   6  7   2.0  0.0   7.0   8.0
15  10   5  2   6  7  10.0  3.0   2.0   4.0
16   3  10  2   7  6   2.0  0.0   7.0   8.0
17   3  10  2   7  6  10.0  3.0   2.0   4.0
18   1   1  5   0  4   NaN  NaN   NaN   NaN
19   6   5  1   9  8   2.0  2.0   3.0   5.0

Left Join DF - DF1
     A   B  C   D  E     W    X     Y     Z
0    0   1  6   6  4   2.0  7.0   9.0   3.0
1    6   4  0   7  0   5.0  1.0   3.0  10.0
2    6   4  0   7  0   8.0  7.0   3.0   7.0
3    6   4  0   7  0   3.0  3.0   7.0   2.0
4    6   4  0   7  0   4.0  2.0   0.0   8.0
5    6   4  0   7  0  10.0  4.0   0.0   1.0
6    3  10  9   8  2  10.0  5.0  10.0   7.0
7    0   7  6   6  7   2.0  7.0   9.0   3.0
8    9   4  7   7  7   NaN  NaN   NaN   NaN
9   10   5  2   6  7   2.0  0.0   7.0   8.0
10  10   5  2   6  7  10.0  3.0   2.0   4.0
11   3  10  2   7  6   2.0  0.0   7.0   8.0
12   3  10  2   7  6  10.0  3.0   2.0   4.0
13   6   1  0  10  0   5.0  1.0   3.0  10.0
```

```
14   6   1  0  10  0   8.0  7.0   3.0   7.0
15   6   1  0  10  0   3.0  3.0   7.0   2.0
16   6   1  0  10  0   4.0  2.0   0.0   8.0
17   6   1  0  10  0  10.0  4.0   0.0   1.0
18   1   1  5   0  4   NaN  NaN   NaN   NaN
19   6   5  1   9  8   2.0  2.0   3.0   5.0

Right Join DF - DF1
     A   B  C   D  E   W  X   Y   Z
0    0   1  6   6  4   2  7   9   3
1    0   7  6   6  7   2  7   9   3
2    6   4  0   7  0   5  1   3  10
3    6   1  0  10  0   5  1   3  10
4    6   4  0   7  0   8  7   3   7
5    6   1  0  10  0   8  7   3   7
6    6   4  0   7  0   3  3   7   2
7    6   1  0  10  0   3  3   7   2
8    6   4  0   7  0   4  2   0   8
9    6   1  0  10  0   4  2   0   8
10   6   4  0   7  0  10  4   0   1
11   6   1  0  10  0  10  4   0   1
12   3  10  9   8  2  10  5  10   7
13  10   5  2   6  7   2  0   7   8
14   3  10  2   7  6   2  0   7   8
15  10   5  2   6  7  10  3   2   4
16   3  10  2   7  6  10  3   2   4
17   6   5  1   9  8   2  2   3   5
```

CHAPTER 10

Practical 9

In this practical we will will practice Numpy and Matplotlib.

10.1 Slides

The slides of the introduction can be found here: Intro

10.2 Numpy

Numpy is a fundamental library for high performance scientific computations. It provides fast and memory efficient data structures like `ndarray` with broadcasting capabilities, standard mathematical functions that can be applied on the arrays avoiding loops, linear algebra functions, I/O methods and it is well integrated with programming languages like C.

To use the functionalities of the library we always need to import it with (standard name is np):

```
import numpy as np
```

The official documentation of the numpy library can be found at http://www.numpy.org/.

10.3 Numpy ndarray

Numpy ndarray is an N-dimensional array object designed to contain homogeneous data (i.e. all data must have the same type). Every array (also called *tensor*) comes with two information: the **shape** (a tuple describing the dimensions of the array) and a **dtype** (the data type of the array).

Given a list of homogeneous values the ndarray can be obtained by using the `np.array(list)` method. Given the numpy array its **shape**, **dimensionality** and **dtype** can be obtained with `array.shape`, `array.ndim` and `array.dtype`.

An example follows:

```
In [1]: import numpy as np

        Aint = np.array([[1,2,3], [4,5,6]])
        Afloat = np.array([[1.1,2,3], [4.2,5,6], [1,2,3]])

        print(Aint)
        print(type(Aint))
        print(Aint.shape)
        print(Aint.dtype)
        print("")
        print(Afloat)
        print(type(Afloat))
        print(Afloat.shape)
        print(Afloat.dtype)
```

```
[[1 2 3]
 [4 5 6]]
<class 'numpy.ndarray'>
(2, 3)
int64

[[1.1 2.  3. ]
 [4.2 5.  6. ]
 [1.  2.  3. ]]
<class 'numpy.ndarray'>
(3, 3)
float64
```

Example:

Let's create a ndarray holding the square root and one with the cubic root of the first five integers. Let's put them together in a 2-d array and check its dimensions.

```
In [2]: import numpy as np
        import math


        mysqrt = [math.sqrt(x) for x in range(0,5)]
        mycrt = [x**(1/3) for x in range(0,5)]
        myOtherRt = [x**(1/5) for x in range(0,5)]
        print("Original lists:")
        print(mysqrt)
        print(mycrt)
        print("")
        npData = np.array(mysqrt)
        print("Numpy ndarray:")
        print(npData)
        print("")
        print("The shape:", npData.shape)
        print("The dimensionality:", npData.ndim)
        print("The type:", npData.dtype)
        print("")
        twoDarray = np.array([mysqrt, mycrt, myOtherRt])
        print("The 2D array:")
        print(twoDarray)
        print("")
        print("The shape:", twoDarray.shape)
        print("The dimensionality:", twoDarray.ndim)
        print("The type:", twoDarray.dtype)
```

```
Original lists:
[0.0, 1.0, 1.4142135623730951, 1.7320508075688772, 2.0]
[0.0, 1.0, 1.2599210498948732, 1.4422495703074083, 1.5874010519681994]

Numpy ndarray:
[0.         1.         1.41421356 1.73205081 2.        ]

The shape: (5,)
The dimensionality: 1
The type: float64

The 2D array:
[[0.         1.         1.41421356 1.73205081 2.        ]
 [0.         1.         1.25992105 1.44224957 1.58740105]
 [0.         1.         1.14869835 1.24573094 1.31950791]]

The shape: (3, 5)
The dimensionality: 2
The type: float64
```

We can also create arrays (matrices) full of zeros or ones or even diagonal matrices with the following numpy methods:

1. Array: `np.zeros(N)` or matrix: `np.zeros((N,M))`
2. Array: `np.ones(N)` or matrix: `np.ones((N,M))`
3. Matrix: `np.eye(N)`

Numpy has its own **range** method that is called `np.arange(N)`. Evenly spaced values in a range can be obtained also with `np.linspace(S,E, num=N, endpoint=True/False)` to obtain N linearly spaced values from S to E (included, unless endpoint = False is specified).

```
In [3]: import numpy as np

        zeros = np.zeros(3)
        zMat = np.zeros((4,3))
        ones = np.ones(3)
        oMat = np.ones((3,2))
        diag = np.eye(4)
        rng = np.arange(5) #5 excluded!

        print("Zero array (1x3)")
        print(zeros)
        print("")
        D = zMat.shape
        print("Zero matrix ({}x{})".format(D[0],D[1]))
        print(zMat)
        print("")
        print("Ones array (1x3)")
        print(ones)
        print("")
        print("Ones matrix (3x2)")
        print(oMat)
        print("")
        print("Diagonal matrix")
        print(diag)
        print("")
        print("Range 0-4")
        print(rng)
        print("A diagonal matrix:")
        dm = np.diag(rng)
```

```
        print(dm)
        print("Its shape:")
        print(dm.shape)

        rng = np.arange(7)
        print("Range 0-6")
        print(rng)

        myRange = np.linspace(-5,2.5,num =6)
        print("6 linearly spaced elements in [-5 - 2.5]:")
        print(myRange)

        myRange = np.linspace(0,21,num =7, endpoint=False)
        print("7 linearly spaced elements in [0 - 21):")
        print(myRange)
```

```
Zero array (1x3)
[0. 0. 0.]

Zero matrix (4x3)
[[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]

Ones array (1x3)
[1. 1. 1.]

Ones matrix (3x2)
[[1. 1.]
 [1. 1.]
 [1. 1.]]

Diagonal matrix
[[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]

Range 0-4
[0 1 2 3 4]
A diagonal matrix:
[[0 0 0 0 0]
 [0 1 0 0 0]
 [0 0 2 0 0]
 [0 0 0 3 0]
 [0 0 0 0 4]]
Its shape:
(5, 5)
Range 0-6
[0 1 2 3 4 5 6]
6 linearly spaced elements in [-5 - 2.5]:
[-5.  -3.5 -2.  -0.5  1.   2.5]
7 linearly spaced elements in [0 - 21):
[ 0.  3.  6.  9. 12. 15. 18.]
```

10.4 Random arrays

Random arrays can be created from several different distributions (check here[56] for all options). Some examples include **uniform** distribution, **normal**, **binomial**,...

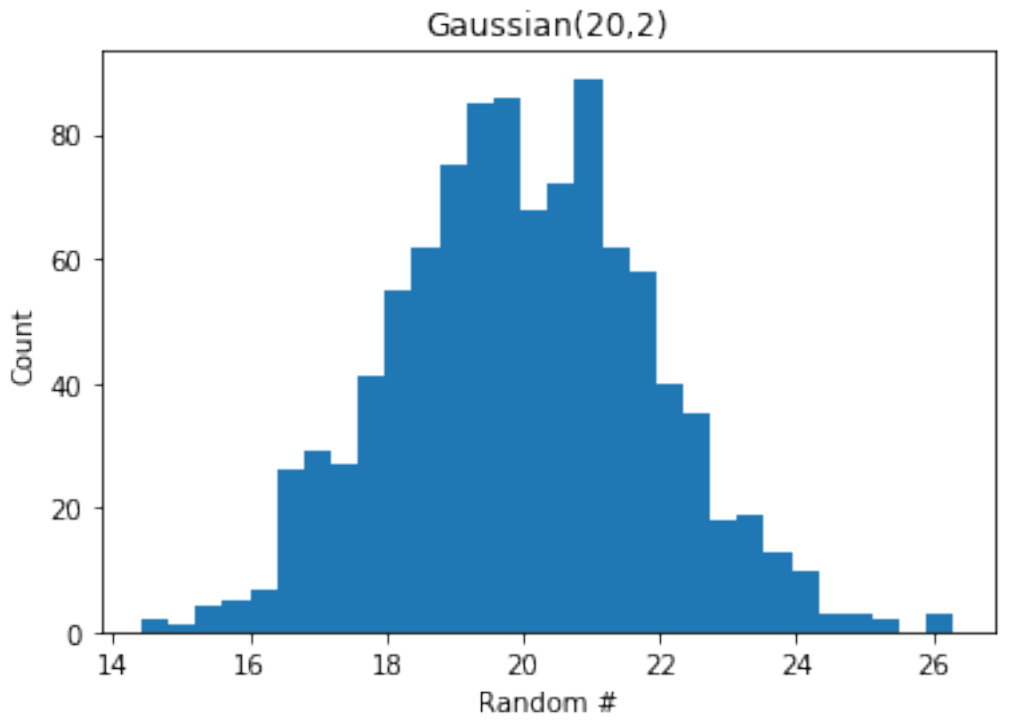
Example:

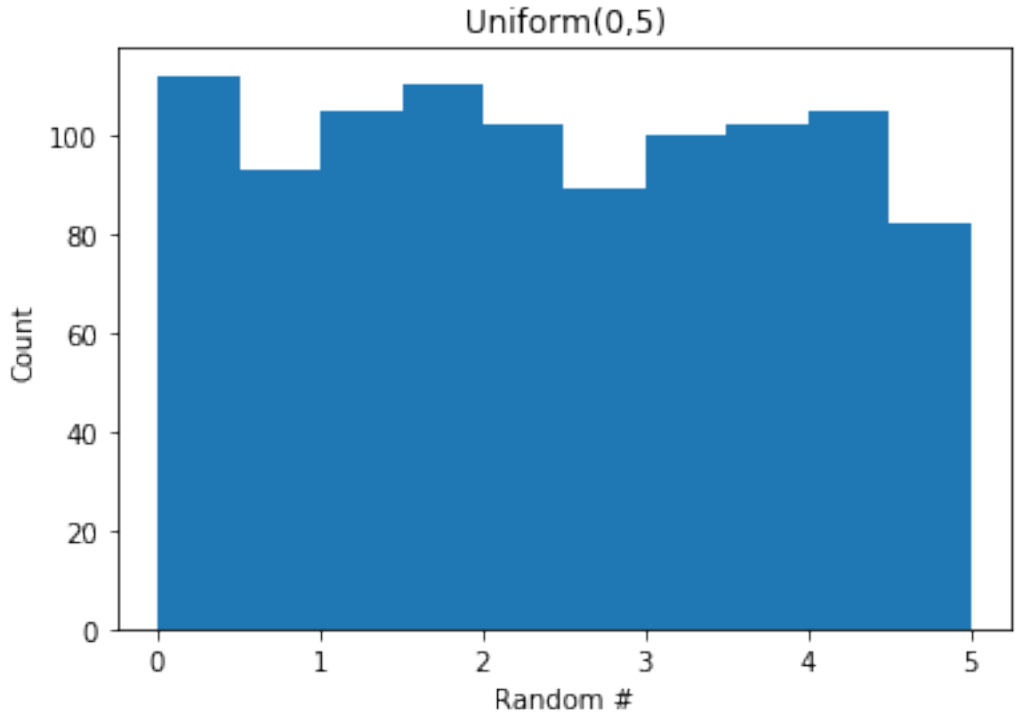
Create a random array of 1000 values drawn from: 1. a gaussian distribution with $\sigma = 20$ and $\mu = 2$ 2. a uniform distribution from 0 to 5 3. a binomial distribution with $p = 0.5$ and $n = 12$

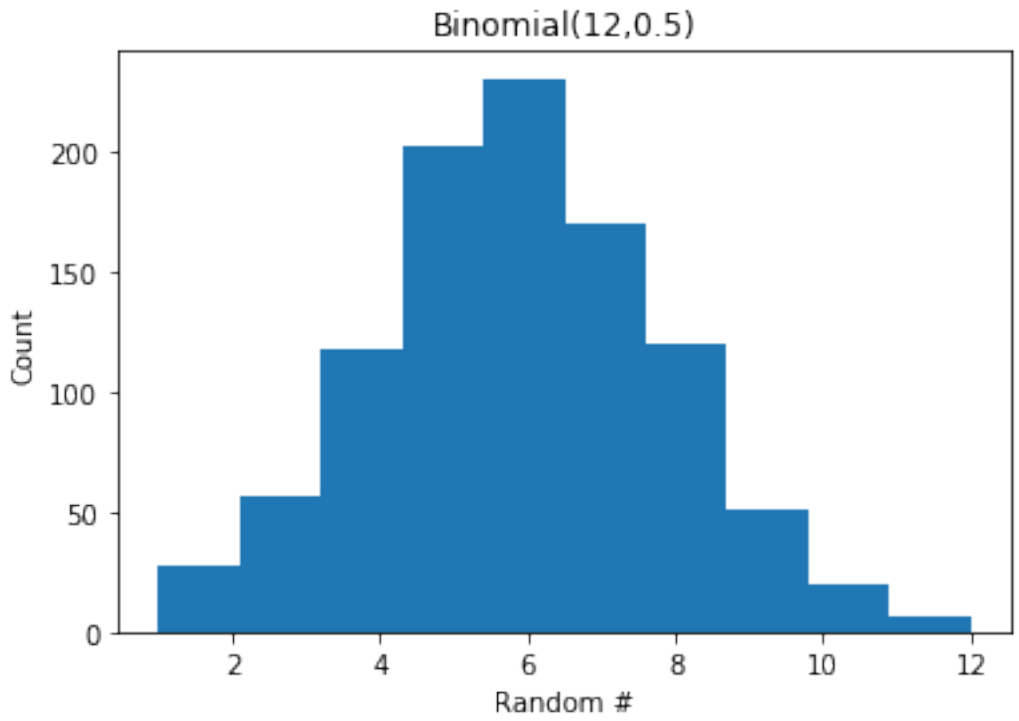
and plot their histogram.

```
In [4]: import numpy as np
        import matplotlib.pyplot as plt

        %matplotlib inline

        #get the gaussian random array
        g = np.random.normal(20,2, 1000)
        #a uniform random array with vals in [0,5]
        u = np.random.uniform(0,5, 1000)
        #get the binomial random array
        b = np.random.binomial(12,0.5, 1000)

        plt.hist(g, bins = 30)
        plt.title("Gaussian(20,2)")
        plt.xlabel("Random #")
        plt.ylabel("Count")
        plt.show()

        plt.hist(u, bins = 10)
        plt.title("Uniform(0,5)")
        plt.xlabel("Random #")
        plt.ylabel("Count")
        plt.show()
        plt.hist(b, bins = 10)
        plt.title("Binomial(12,0.5)")
        plt.xlabel("Random #")
        plt.ylabel("Count")
        plt.show()
```

[56] https://docs.scipy.org/doc/numpy-1.16.0/reference/routines.random.html#distributions


Gaussian(20,2)
Count
Random #
0
20
40
60
80
14
16
18
20
22
24
26


Uniform(0,5)
Count
Random #
0
20
40
60
80
100
0
1
2
3
4
5



It is also possible to specify the the size of the output (with the tuple `size`)to get a matrix instead.

```
In [5]: import numpy as np

        u = np.random.uniform(0,1, size = (4,3))

        print("Uniform distr. in (0,1). 4x3 matrix:")
        print("")
        print(u)
```

```
Uniform distr. in (0,1). 4x3 matrix:

[[0.01748314 0.68196938 0.7337711 ]
 [0.9061673  0.27011933 0.83324199]
 [0.13474732 0.21965773 0.24174557]
 [0.81930429 0.25657836 0.10198619]]
```

To always get the same values we need to initialize the random seed (a parameter of the pseudo-random sequence generator) always to the same value.

```
In [6]: import numpy as np

        u = np.random.uniform(0,1,size=(3,2))
        u1 = np.random.uniform(0,1,size=(3,2))
        u2 = np.random.uniform(0,1,size=(3,2))

        print(" u: {}\n\n u1:{}\n\n u2:{}".format(u,u1,u2))

        print("")
        print("With random seed reinit.")
        np.random.seed(0)
        u = np.random.uniform(0,1,3)
        np.random.seed(0)
        u1 = np.random.uniform(0,1,3)
```

```
        np.random.seed(0)
        u2 = np.random.uniform(0,1,3)

        print(" u: {}\n u1:{}\n u2:{}".format(u,u1,u2))
```

```
 u: [[0.1182575  0.88378038]
 [0.64969197 0.45585741]
 [0.0245672  0.28397606]]

 u1:[[0.43501974 0.6943427 ]
 [0.527252   0.96579963]
 [0.20484529 0.05242482]]

 u2:[[0.1545874  0.89794199]
 [0.75233458 0.56678759]
 [0.86651043 0.90583683]]

With random seed reinit.
 u: [0.5488135  0.71518937 0.60276338]
 u1:[0.5488135  0.71518937 0.60276338]
 u2:[0.5488135  0.71518937 0.60276338]
```

10.5 Numpy to and from pandas

Pandas Series and DataFrames are actually built on top of numpy ndarrays. Moreover, given a numpy matrix (or array) it is quite easy to convert it into a DataFrame (or Series).

```
In [7]: import pandas as pd
        import numpy as np

        myDict = {"one" : 1, "two" : 2, "three" : 3, "four" : 4}

        mySeries = pd.Series(myDict)

        print(myDict)
        print("")
        print(type(mySeries))
        print("")
        print(type(mySeries.values))
        print("")

        myMat = np.diag(np.arange(6))
        myDF = pd.DataFrame(myMat)
        print("Numpy matrix")
        print(myMat)
        print("")
        print("Pandas DataFrame")
        print(myDF)
        print("")
        print("Reindexed DataFrame")
        myDF = pd.DataFrame(myMat, index = list("ABCDEF"),
                            columns = ['a1','b2','c3','d4','e5','f6'])
        print(myDF)
```

```
{'one': 1, 'two': 2, 'three': 3, 'four': 4}

<class 'pandas.core.series.Series'>
```

```
<class 'numpy.ndarray'>

Numpy matrix
[[0 0 0 0 0 0]
 [0 1 0 0 0 0]
 [0 0 2 0 0 0]
 [0 0 0 3 0 0]
 [0 0 0 0 4 0]
 [0 0 0 0 0 5]]

Pandas DataFrame
   0  1  2  3  4  5
0  0  0  0  0  0  0
1  0  1  0  0  0  0
2  0  0  2  0  0  0
3  0  0  0  3  0  0
4  0  0  0  0  4  0
5  0  0  0  0  0  5

Reindexed DataFrame
   a1  b2  c3  d4  e5  f6
A   0   0   0   0   0   0
B   0   1   0   0   0   0
C   0   0   2   0   0   0
D   0   0   0   3   0   0
E   0   0   0   0   4   0
F   0   0   0   0   0   5
```

10.6 Reshaping

The shape of a numpy ndarray can be changed at any time with the methods `reshape((R,C))` where R and C are the rows and columns used to shape an array into a matrix, and `ravel` that is used to go the other way around from matrix to array.

```
In [8]: import numpy as np

        myA = np.arange(12)

        print("The array:")
        print(myA)
        print("")
        myB = myA.reshape((2,6))
        print("Reshaped (2x6):")
        print(myB)
        print("")
        myC = myA.reshape((3,4))
        print("Reshaped: (3x4)")
        print(myC)

        myD = myB.ravel()
        print("")
        print("Back to array:")
        print(myD)
```

```
The array:
[ 0  1  2  3  4  5  6  7  8  9 10 11]
```

```
Reshaped (2x6):
[[ 0  1  2  3  4  5]
 [ 6  7  8  9 10 11]]

Reshaped: (3x4)
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]

Back to array:
[ 0  1  2  3  4  5  6  7  8  9 10 11]
```

Be careful though, if shapes don't match an error **cannot reshape array** is returned:

```
import numpy as np

myA = np.arange(10)

myB = myA.reshape(2,7)
print(myB)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-9-2936ce504f51> in <module>()
      3 myA = np.arange(10)
      4
----> 5 myB = myA.reshape(2,7)
      6 print(myB)

ValueError: cannot reshape array of size 10 into shape (2,7)
```

10.7 Iterating over arrays and Indexing

It is possible to loop over the elements of an array by element or by row of the matrix. In the first case we use a the method `np.ndarray.flat` to make a matrix accessible as a one dimensional array. Check out the following examples:

```
In [9]: import numpy as np

        myA = np.arange(12)
        print(myA)
        print("")
        print("Matrix:")
        myA = myA.reshape((4,3))
        print(myA)

        print("Looping through elements:")# equivalent to:
        for el in myA.flat:               # for el in myA.ravel():
            print("Element:", el)         #     print("Element:",el)
```

```
        print("Looping row by row:")
        for el in myA:
            print("Row: ", el, "is a", type(el))
            for j in el:
                print("\tel:", j)
```

```
[ 0  1  2  3  4  5  6  7  8  9 10 11]

Matrix:
[[ 0  1  2]
 [ 3  4  5]
 [ 6  7  8]
 [ 9 10 11]]
Looping through elements:
Element: 0
Element: 1
Element: 2
Element: 3
Element: 4
Element: 5
Element: 6
Element: 7
Element: 8
Element: 9
Element: 10
Element: 11
Looping row by row:
Row:  [0 1 2] is a <class 'numpy.ndarray'>
        el: 0
        el: 1
        el: 2
Row:  [3 4 5] is a <class 'numpy.ndarray'>
        el: 3
        el: 4
        el: 5
Row:  [6 7 8] is a <class 'numpy.ndarray'>
        el: 6
        el: 7
        el: 8
Row:  [ 9 10 11] is a <class 'numpy.ndarray'>
        el: 9
        el: 10
        el: 11
```

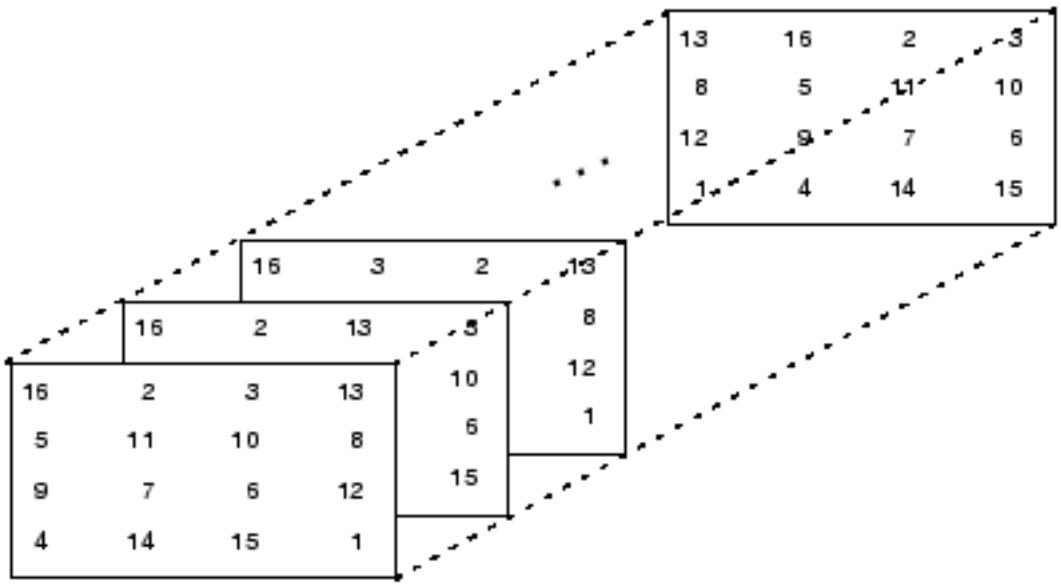
Indexing and slicing can be performed as usual with the **[]** operator. `np.ndarray[i,j]` returns the element at row `i` and column `j`.

A multi-dimensional matrix has more than 2 dimensions. They can be thought as several 2D matrices stacked one on top of the other:

Note that `np.ndarray[0,:,:]` is the whole first matrix. `np.ndarray[:,0,:]` is all the first rows, while `np.ndarray[:,:,0]` is all the first columns. Regarding slicing and indexing, the same reasoning applies to n-dimensional matrices. For example, `myB` below is a 3x3x3 matrix.

```
In [10]: import numpy as np
```

```
        myA = np.random.randint(0,10, size = (3,4))
        print(myA)
        print("")
```




```
print("myA[2,2] = ", myA[2,2])
print("myA[1,3] = ", myA[1,3])
print("myA[0,3] = ", myA[0,3])
print("second row:", myA[1,:])
print("")
print("3D matrix ",)
myB = np.random.randint(0,10, size = (3,3,3))
print(" - shape:", myB.shape)
print(myB)
print("")
print("myB[0,2,2] = ", myB[0,2,2])
print("Second matrix:")
print(myB[1,:,:])
print("Third row of second matrix:")
print(myB[1,2,:])
print("Second column of second matrix:")
print(myB[1,:,1])
```

```
[[3 7 9 3]
 [5 2 4 7]
 [6 8 8 1]]

myA[2,2] =  8
myA[1,3] =  7
myA[0,3] =  3
second row: [5 2 4 7]

3D matrix
 - shape: (3, 3, 3)
[[[6 7 7]
  [8 1 5]
  [9 8 9]]

 [[4 3 0]
  [3 5 0]
  [2 3 8]]
```

```
 [[1 3 3]
  [3 7 0]
  [1 9 9]]]

myB[0,2,2] =  9
Second matrix:
[[4 3 0]
 [3 5 0]
 [2 3 8]]
Third row of second matrix:
[2 3 8]
Second column of second matrix:
[3 5 3]
```

10.8 Broadcasting and arithmetic functions

Universal functions are very efficient element-wise functions that can be applied on ndarrays. As seen in Pandas, operations involving scalar values and arrays are broadcast to all the elements of the ndarray, while operations performed on arrays having the same size are done element by element.

Note also that it is possible to perform some operations only on a **sub-part** of an ndarray. Moreover, if operators are applied to **differently sized, but compatible, ndarrays** (es. matrix 3x3x3 and vector 3x1), numpy will perform broadcast of the operator. Let's see some examples:

```
In [11]: import numpy as np

         A = np.random.randint(0,10, size = (3,2))
         B = np.random.randint(0,10, size = (3,3,3))
         C = np.random.randint(0,10, size = (3,1))
         print("Matrix A 3x2")
         print(A)
         print("")
         print("Matrix B 3x3x3")
         print(B)
         print("")
         print("Matrix C 3x1")
         print(C)
         print("")
         print("A squared")
         print(A**2)
         print("")
         print("A square-rooted")
         print(np.sqrt(A))

         print("")
         print("B square-rooted")
         print(np.sqrt(B))

         print("A + C ")
         print(A + C)
         print("")

         print("B + C ")
         print(B  + C)

         print("")
```

```
        print("B")
        print(B)
        print("Sub array B - 20")
        B[:, 0:2 , 0:2 ] -= 20
        print(B)
```

```
Matrix A 3x2
[[0 4]
 [7 3]
 [2 7]]

Matrix B 3x3x3
[[[2 0 0]
  [4 5 5]
  [6 8 4]]

 [[1 4 9]
  [8 1 1]
  [7 9 9]]

 [[3 6 7]
  [2 0 3]
  [5 9 4]]]

Matrix C 3x1
[[4]
 [6]
 [4]]

A squared
[[ 0 16]
 [49  9]
 [ 4 49]]

A square-rooted
[[0.         2.        ]
 [2.64575131 1.73205081]
 [1.41421356 2.64575131]]

B square-rooted
[[[1.41421356 0.         0.        ]
  [2.         2.23606798 2.23606798]
  [2.44948974 2.82842712 2.        ]]

 [[1.         2.         3.        ]
  [2.82842712 1.         1.        ]
  [2.64575131 3.         3.        ]]

 [[1.73205081 2.44948974 2.64575131]
  [1.41421356 0.         1.73205081]
  [2.23606798 3.         2.        ]]]
A + C
[[ 4  8]
 [13  9]
 [ 6 11]]

B + C
[[[ 6  4  4]
  [10 11 11]
```

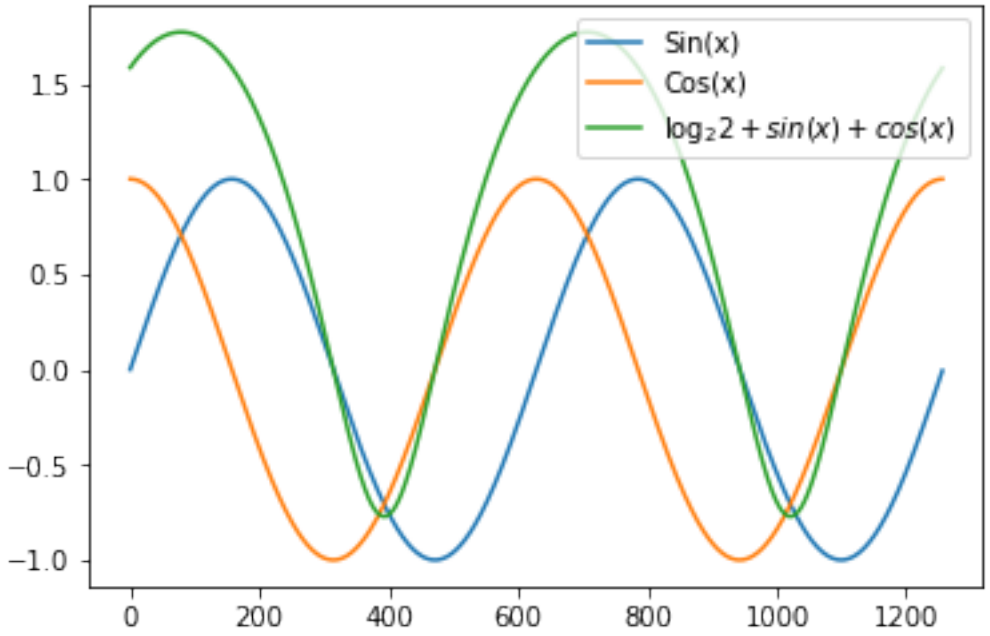
```
  [10 12  8]]

 [[ 5  8 13]
  [14  7  7]
  [11 13 13]]

 [[ 7 10 11]
  [ 8  6  9]
  [ 9 13  8]]]

B
[[[2 0 0]
  [4 5 5]
  [6 8 4]]

 [[1 4 9]
  [8 1 1]
  [7 9 9]]

 [[3 6 7]
  [2 0 3]
  [5 9 4]]]
Sub array B - 20
[[[-18 -20   0]
  [-16 -15   5]
  [  6   8   4]]

 [[-19 -16   9]
  [-12 -19   1]
  [  7   9   9]]

 [[-17 -14   7]
  [-18 -20   3]
  [  5   9   4]]]
```

A lot of mathematical and statistical functions can be applied. Examples include trigonometric functions like **sin, cos, tan,...**, statistical functions like **mean, std, max, min, cumsum,...** see the documentation[57] for a complete list.

Example: Let's create and plot a sin, cos and $\log_2(2 + sin + cos)$ from -2Pi to 2Pi

```
In [12]: import numpy as np
         import matplotlib.pyplot as plt

         A = np.arange(-2* np.pi, 2*np.pi, 0.01)

         sA = np.sin(A)
         cA = np.cos(A)


         plt.plot(sA)
         plt.plot(cA)
         plt.plot(np.log2(2 + sA + cA))

         plt.legend(["Sin(x)", "Cos(x)","$\log_2{2+sin(x)+cos(x)}$"])
         plt.show()
```

[57] https://docs.scipy.org/doc/numpy/reference/routines.math.html



10.9 Filtering

It is possible to filter `np.ndarrays` to retrieve the indexes (or the values) meeting specific conditions. The method `where` provides the index of those values. Note that if the `np.ndarray` is actually a matrix, `where` returns a tuple of indexes that are respectively the i and j coordinates of the elements fulfilling the condition.

Example: write a method that creates the np.ndarray with all the integers multiple of a certain number N that do not exceed M. The method should return also the indexes of those values in the original np.ndarray.

```
In [13]: import numpy as np

         def getMultiples(N,M):
             all = np.arange(1,M+1)
             ind = np.where(all % 27 == 0)
             filt = (all % 27 == 0)
             return ind, all[filt]


         res = getMultiples(11, 1000)
         print("Indexes: {}\nValues:{}".format(res[0],res[1]))
```

```
Indexes: (array([ 26,  53,  80, 107, 134, 161, 188, 215, 242, 269, 296, 323, 350,
       377, 404, 431, 458, 485, 512, 539, 566, 593, 620, 647, 674, 701,
       728, 755, 782, 809, 836, 863, 890, 917, 944, 971, 998]),)
Values:[ 27  54  81 108 135 162 189 216 243 270 297 324 351 378 405 432 459 486
 513 540 567 594 621 648 675 702 729 756 783 810 837 864 891 918 945 972
 999]
```

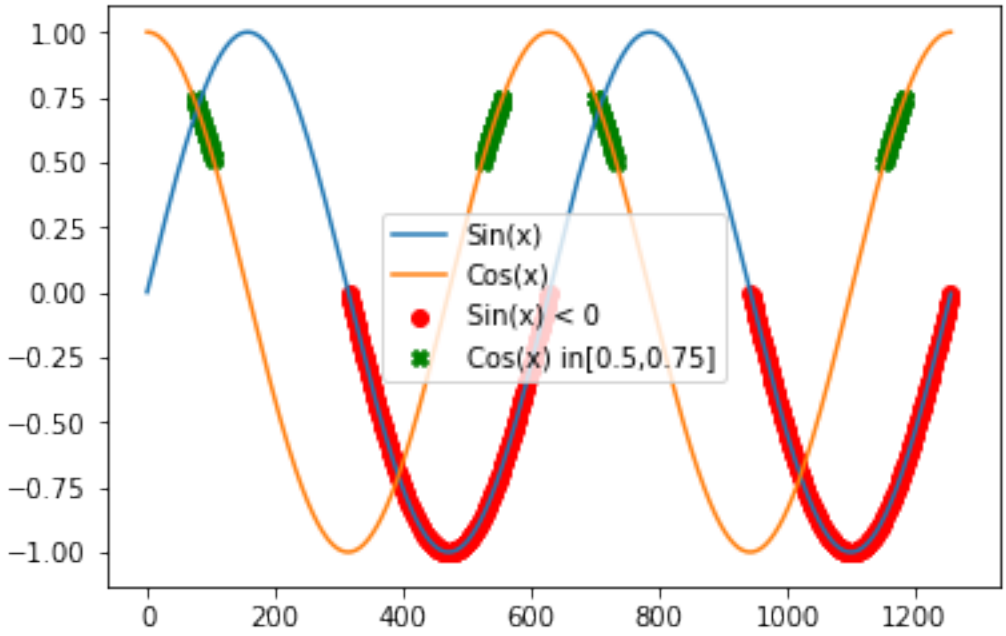
```
In [14]: import numpy as np

         myA = np.arange(12).reshape((3,4))
         print(myA)
         print("Elements divisible by 3")
         indexes = np.where(myA % 3 == 0)
```

```
        print(indexes)
        for ind in range(len(indexes[0])):
            i = indexes[0][ind]
            j = indexes[1][ind]
            print("({},{}) -> {}".format(i,j, myA[i][j]))
```

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]
Elements divisible by 3
(array([0, 0, 1, 2]), array([0, 3, 2, 1]))
(0,0) -> 0
(0,3) -> 3
(1,2) -> 6
(2,1) -> 9
```

Example: Let's do some filtering on the sine and cosine series plotted above.

```
In [15]: import numpy as np
         import matplotlib.pyplot as plt

         A = np.arange(-2* np.pi, 2*np.pi, 0.01)

         sA = np.sin(A)
         cA = np.cos(A)

         plt.plot(sA)
         plt.plot(cA)


         s0A_Y = sA[sA < 0]
         s0A_X = np.where(sA < 0)

         c0A_Y = cA[np.all( [cA > 0.5 , cA < 0.75], axis = 0)]
         c0A_X = np.where(np.all( [cA > 0.5 , cA < 0.75], axis = 0))

         plt.scatter(s0A_X,s0A_Y, marker='o', c = 'red')
         plt.scatter(c0A_X,c0A_Y, marker='X', c = 'green')

         plt.legend(["Sin(x)",  "Cos(x)","Sin(x) < 0", "Cos(x) in[0.5,0.75]"])
         plt.show()
```



Note that in the code above, `np.all` tests if the two conditions are True at the same time (i.e. AND). If we want to test if at least one is True we use `np.any`. See np.all[58] and np.any[59] for more info:

```
In [16]: import numpy as np

         v1 = [True, False,False, True]
         v2 = [False, False,True, True]
         v3 = [False, False, False, True]
         print("vals:")
         vals = np.array([v1,v2,v3])
         print(vals)
         print("\nANY(vals):")
         print(np.any(vals, axis=0))
         print("\nANY(vals) on rows:")
         print(np.any(vals, axis=1))
         print("\nALL(vals):")
         print(np.all(vals, axis=0))

         print("\nALL(vals) on rows:")
         print(np.all(vals, axis=1))


vals:
[[ True False False  True]
 [False False  True  True]
 [False False False  True]]

ANY(vals):
[ True False  True  True]

ANY(vals) on rows:
[ True  True  True]
```

[58] https://docs.scipy.org/doc/numpy-1.16.1/reference/generated/numpy.all.html#numpy.all
[59] https://docs.scipy.org/doc/numpy-1.16.1/reference/generated/numpy.all.html#numpy.any

```
ALL(vals):
[False False False  True]

ALL(vals) on rows:
[False False False]
```

10.10 Linear algebra

In numpy the product operator `*` is used to perform element-wise product, and therefore the arrays in input must have the same size. Matrix-matrix multiplication can be done through the `np.dot(x,y)` method, which is also a method of the object `np.ndarray` and therefore if `x` is a `np.ndarray` of compatible size with `y`, we can perform their product with `x.dot(y)`.

A ndarray can be transposed by using the method `T`.

```
In [17]: import numpy as np

         a = np.array([1, 2, 3, 1, 2, 3,1, 1 ,1])
         A = a.reshape((3,3))
         B = np.random.randint(0,10, size = (3,2))

         print("A (3x3)")
         print(A)
         print("")
         print("B (3x2)")
         print(B)
         print("")
         print("AxB (3x2)")
         print(A.dot(B))
         print("")
         print("A transposed:")
         print(A.T)
```

```
A (3x3)
[[1 2 3]
 [1 2 3]
 [1 1 1]]

B (3x2)
[[4 3]
 [4 4]
 [8 4]]

AxB (3x2)
[[36 23]
 [36 23]
 [16 11]]

A transposed:
[[1 1 1]
 [2 2 1]
 [3 3 1]]
```

Numpy has a module called `linalg` providing several functions that implement matrix decompositions, inversions, determinant calculation and so on. Some of these methods are `linalg.inv` for the matrix inversion, `linalg.qr` for the QR decomposition, `linalg.cholesky` for the Cholesky decomposition, `linalg.eig` to find the eigenvalues of a matrix, `linalg.solve` to solve a system of linear equations and so on.

The complete list of all available methods can be found in the documentation[60].

Example: Let's create a random 4x4 matrix and invert it. Let's finally check that the product of the matrix by its inversion yields the 4x4 identity matrix.

```
In [18]: import numpy as np
         from numpy import linalg

         A = np.random.randint(0,10, size = (4,4))
         print("Matrix A:")
         print(A)
         print("")
         print("inv(A)")
         A_1 = linalg.inv(A)
         print(A_1)
         print("")
         print(np.dot(A,A_1))
```

```
Matrix A:
[[3 7 5 5]
 [0 1 5 9]
 [3 0 5 0]
 [1 2 4 2]]

inv(A)
[[ 0.23121387  0.05780347  0.38150289 -0.83815029]
 [ 0.09537572 -0.10115607 -0.16763006  0.21676301]
 [-0.13872832 -0.03468208 -0.02890173  0.50289017]
 [ 0.06647399  0.1416185   0.03468208 -0.30346821]]

[[ 1.00000000e+00  2.77555756e-17  2.08166817e-17 -1.66533454e-16]
 [-4.16333634e-17  1.00000000e+00 -6.93889390e-18  5.55111512e-17]
 [ 2.77555756e-17  6.93889390e-18  1.00000000e+00  0.00000000e+00]
 [-2.77555756e-17  0.00000000e+00  6.93889390e-17  1.00000000e+00]]
```

Example: Let's solve the following system of linear equations: $\begin{cases} 2x + y - 2z = 3 \\ x - y - z = 0 \\ x + y + 3z = 12 \end{cases}$

```
In [19]: import numpy as np
         from numpy import linalg

         A = np.array([[2, 1, -2],[1,-1,-1], [1, 1 ,3]])
         b = np.array([3, 0, 12])

         print("A")
         print(A)
         print("")
         print("b")
         print(b)
         print("")
         #Let's find x such that: Ax = b
         x = linalg.solve(A,b)
         print("X")
         print(x)
         #Let's double check Ax = b:
         print("")
         print("Ax:")
         print(np.dot(A,x))
```

[60] https://docs.scipy.org/doc/numpy-1.13.0/reference/routines.linalg.html

```
A
[[ 2  1 -2]
 [ 1 -1 -1]
 [ 1  1  3]]

b
[ 3  0 12]

X
[3.5 1.  2.5]

Ax:
[ 3.  0. 12.]
```

10.11 Matplotlib

Matplotlib is a library created for producing publication quality plots. To use matplotlib one has to import it in the python program. Typically this is done with

```
import matplotlib.pyplot as plt
```

Documentation and several examples can be found at the matplotlib website[61].

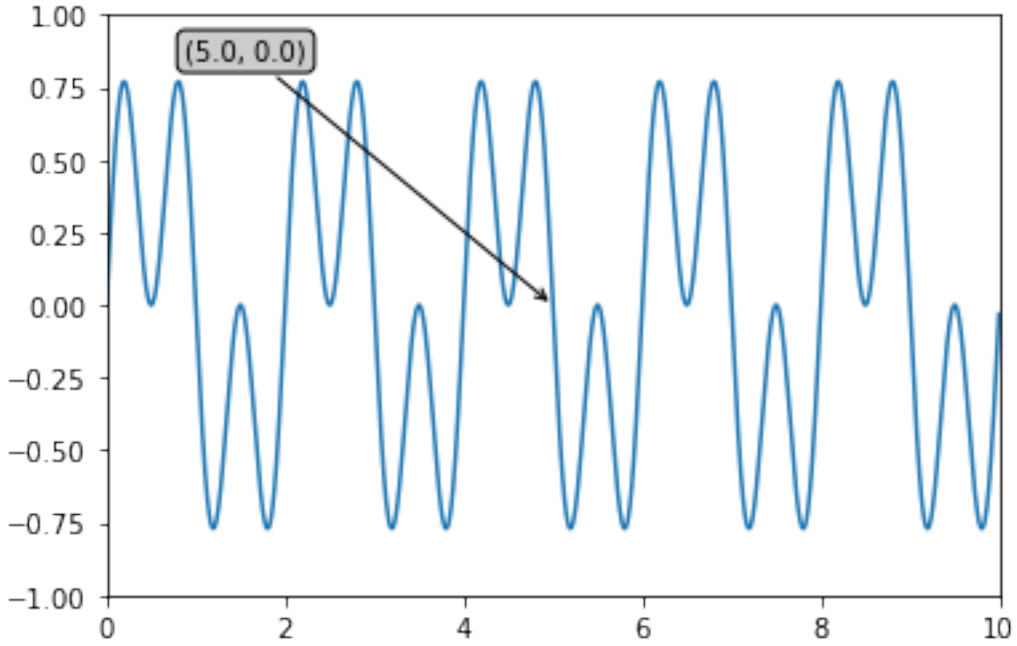
Figures can also be saved to files with `plt.savefig('filename')`.

Example: [adapted from documentation] Let's plot the function $cos(2 * \Pi * x) * sin(2 * \Pi * x)$ for x ranging from 0 to 10, annotating the point (5,0).

```
In [20]: import numpy as np
         import matplotlib.pyplot as plt

         x = np.arange(0, 10, 0.005)
         y = np.cos(np.pi*x) * np.sin(2*np.pi*x)

         fig = plt.figure()

         ax = fig.add_subplot(111)
         xdata = 5
         ydata = 0

         ax.plot(x, y)
         ax.set_xlim(0, 10)
         ax.set_ylim(-1, 1)

         xdata = 5
         ydata = 0

         bbox = dict(boxstyle="round", fc="0.8")
         arrowprops = dict(
             arrowstyle = "->")

         offset = 92
         ax.annotate('(%.1f, %.1f)'%(xdata, ydata),
                     (xdata, ydata), xytext=(-1.5*offset, offset), textcoords='offset points',
                     bbox=bbox, arrowprops=arrowprops)
```

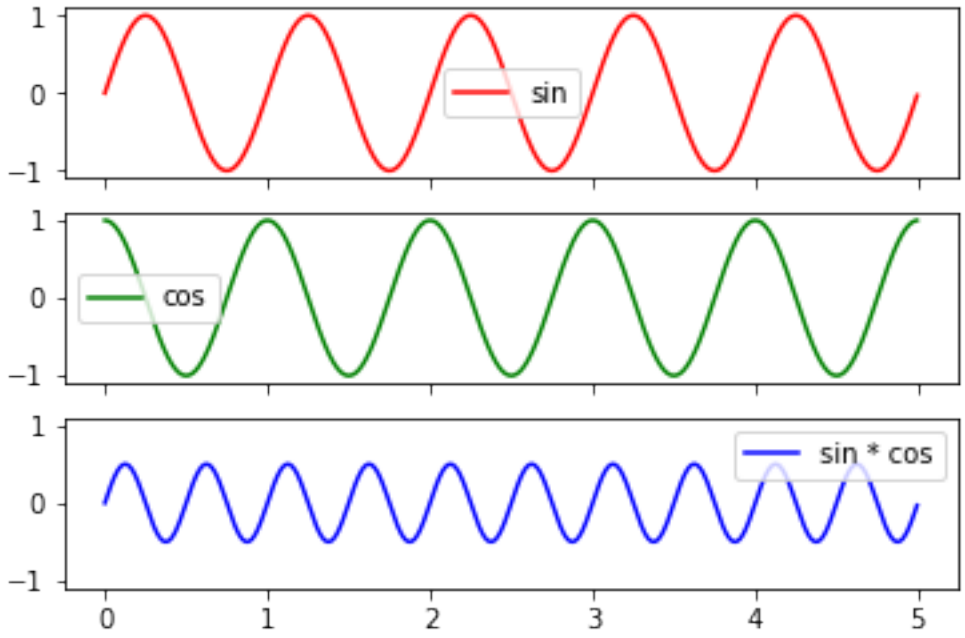
[61] https://matplotlib.org/index.html

```
plt.show()
```



Example: Let's plot simultaneusly $sin(2 * \Pi * x)$, $cos(2\Pi x)$ and $sin(2\Pi x) * cos(2\Pi x)$ for x ranging from 0 to 5.

```
In [21]: import numpy as np
         import matplotlib.pyplot as plt

         x = np.arange(0, 5, 0.005)
         y = np.sin(2*np.pi*x)
         y1 = np.cos(2*np.pi*x)
         y2 = np.sin(2*np.pi*x)*np.cos(2*np.pi*x)

         f, (ax1, ax2, ax3) = plt.subplots(3, sharex=True, sharey=True)
         ax1.plot(x, y,'r')
         ax2.plot(x, y1,'g')
         ax3.plot(x,y2,'b')
         ax1.legend(["sin"])
         ax2.legend(["cos"])
         ax3.legend(["sin * cos"])

         plt.show()
```



10.12 Exercises

1. Create the following functions:
   - (a) createRadomList : with parameters, N, min, max. Creates a list of N random integers ranging from min to max;
   - (b) **getIdentical** : with parameters two lists of integers L1 and L2 having the same size. It returns the list of indexes I where L1[I] == L2[I]
   - (c) **check** : gets lists L1, L2, identities (as computed by getIdentical) and a number N and prints if the first N and last N values in identities correspond to indexes of identical values in L1 and L2;
   - (d) implement **getIdentical** using numpy.ndarrays. Call it **getIdenticalNpy** (hint: subtract the two arrays and find zeros).

Test the software creating two lists of 100,000 random numbers from 0 to 10. You should get something similar to:

```
100000
100000
9032
Index: 4 L: 1 L1: 1
Index: 21 L: 10 L1: 10
Index: 32 L: 6 L1: 6
Index: 37 L: 5 L1: 5
Index: 47 L: 1 L1: 1
Index: 99974 L: 8 L1: 8
Index: 99982 L: 6 L1: 6
Index: 99983 L: 3 L1: 3
Index: 99984 L: 4 L1: 4
Index: 99987 L: 0 L1: 0

Identities are OK!
```

(continues on next page)

(continued from previous page)

```
9032
Index: 4 L: 1 L1: 1
Index: 21 L: 10 L1: 10
Index: 32 L: 6 L1: 6
Index: 37 L: 5 L1: 5
Index: 47 L: 1 L1: 1
Index: 99974 L: 8 L1: 8
Index: 99982 L: 6 L1: 6
Index: 99983 L: 3 L1: 3
Index: 99984 L: 4 L1: 4
Index: 99987 L: 0 L1: 0

Identities are OK!
```

Show/Hide Solution

```
In [22]: %reset -s -f

         import random
         import numpy as np
         import time


         def createRandomList(N=0, min=0, max=10):
             """creates a list of N random numbers in [min,max]"""
             L = []
             for i in range(0,N):
                 L.append(random.randint(min,max))
             #L = [random.randint(min,max) for i in range(0,N)]
             return L


         def getIdentical(list1,list2):
             """return indexes of identical values
             (lists assumed of same size)"""

             return [i for i in range(0,len(list1)) if list1[i] == list2[i]]

         def getIdenticalNpy(list1,list2):
             """return indexes of identical values
             (lists assumed of same size)"""
             nda1 = np.array(list1)
             nda2 = np.array(list2)
             diff = nda1 - nda2
             ret = np.where(diff == 0)
             return ret


         def check(list1, list2, identical,N=10):
             ret = True
             for index in identical[0:5]:
                 if(list1[index] !=list2[index]):
                     ret = False

                 print("Index: {} L: {} L1: {}".format(index, list1[index], list2[index]))
                 #assert(L[index] ==  L1[index])
             for index in identical[-N:]:
                 if(list1[index] !=list2[index]):
```

```
                ret = False
            print("Index: {} L: {} L1: {}".format(index, list1[index], list2[index]))
            #assert(L[index] ==  L1[index])

        return ret

    def checkRND(list1, list2, identical,N=10):
        ret = True
        for index in createRandomList(N,0,len(identical)):
            I = identical[index]
            if(list1[I] !=list2[I]):
                ret = False

            print("Index: {} L: {} L1: {}".format(index, list1[index], list2[index]))
            #assert(L[index] ==  L1[index])

        return ret



    L = createRandomList(100000)
    L1 = createRandomList(100000)
    print(len(L))
    print(len(L1))

    identical = getIdentical(L,L1)
    print(len(identical))

    if check(L,L1,identical,5):
        print("\nIdentities are OK!")
    else:
        print("\nSomething went wrong!")


    id1 = getIdenticalNpy(L,L1)
    print(len(id1[0]))
    if check(L,L1,identical,5):
        print("\nIdentities are OK!")
    else:
        print("\nSomething went wrong!")
```

```
100000
100000
9143
Index: 16 L: 2 L1: 2
Index: 23 L: 6 L1: 6
Index: 51 L: 2 L1: 2
Index: 55 L: 1 L1: 1
Index: 84 L: 8 L1: 8
Index: 99935 L: 10 L1: 10
Index: 99957 L: 2 L1: 2
Index: 99966 L: 3 L1: 3
Index: 99992 L: 4 L1: 4
Index: 99995 L: 5 L1: 5

Identities are OK!
9143
Index: 16 L: 2 L1: 2
Index: 23 L: 6 L1: 6
```

```
Index: 51 L: 2 L1: 2
Index: 55 L: 1 L1: 1
Index: 84 L: 8 L1: 8
Index: 99935 L: 10 L1: 10
Index: 99957 L: 2 L1: 2
Index: 99966 L: 3 L1: 3
Index: 99992 L: 4 L1: 4
Index: 99995 L: 5 L1: 5

Identities are OK!
```

2. Write a function that converts a numpy ndarray of temperatures expressed in Degrees Celsius into Degrees Farenheit. The formula to convert a temperature C in Celsius into F in Farenheit is the following:

$$F = C * 9/5 + 32$$

Write then a function that converts a numpy ndarray of temperatures in Farenheit into Celsius.

Finally:

(a) apply the Celsius to Farenheit conversion on an ndarray containing the following October's minimum and maximum temperatures in Trento: tmin = [12, 11, 11, 8, 9, 10, 3, 8, 4, 5, 10, 9, 8, 9, 8, 7, 6, 4, 5, 6, 9, 9, 3, 3, 5] and tmax =[15, 22, 18, 20, 22, 22, 20, 21, 21, 21, 21, 23, 24, 24, 24, 25, 22, 22, 20, 20, 19, 15, 20, 23, 19];

(b) check that both functions work correctly by converting the values from Celsius to Farenheit and back to Celsius;

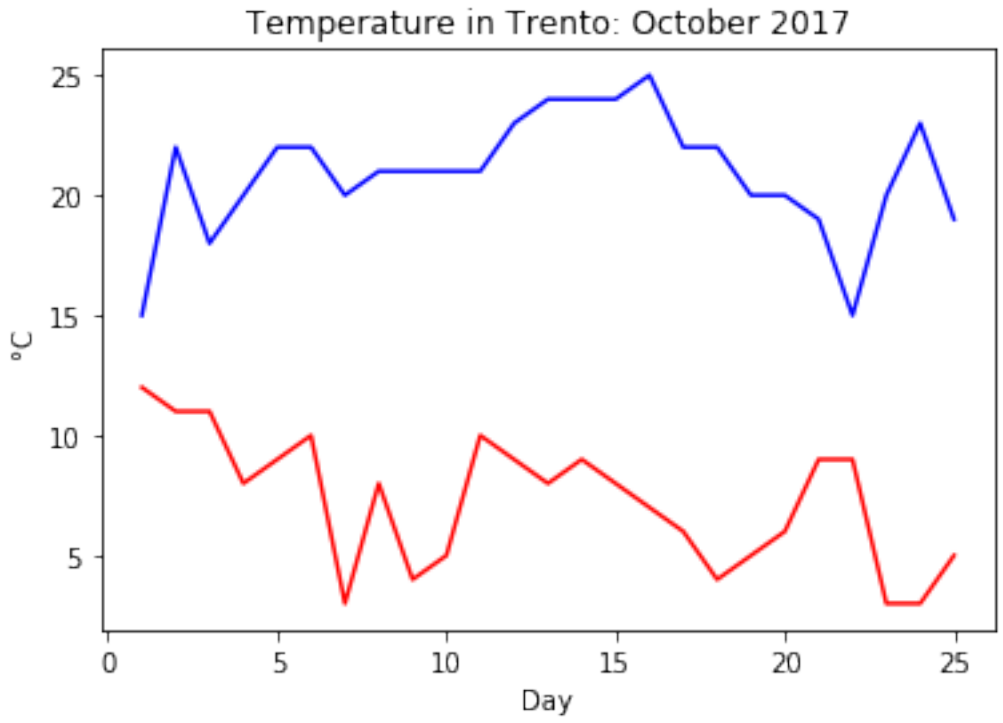
(c) plot the minimum and maximum temperatures in celsius on the same graph. Since the temperatures refer to the first 25 days of the month of October 2017, the x coordinate can be a range(1,26);

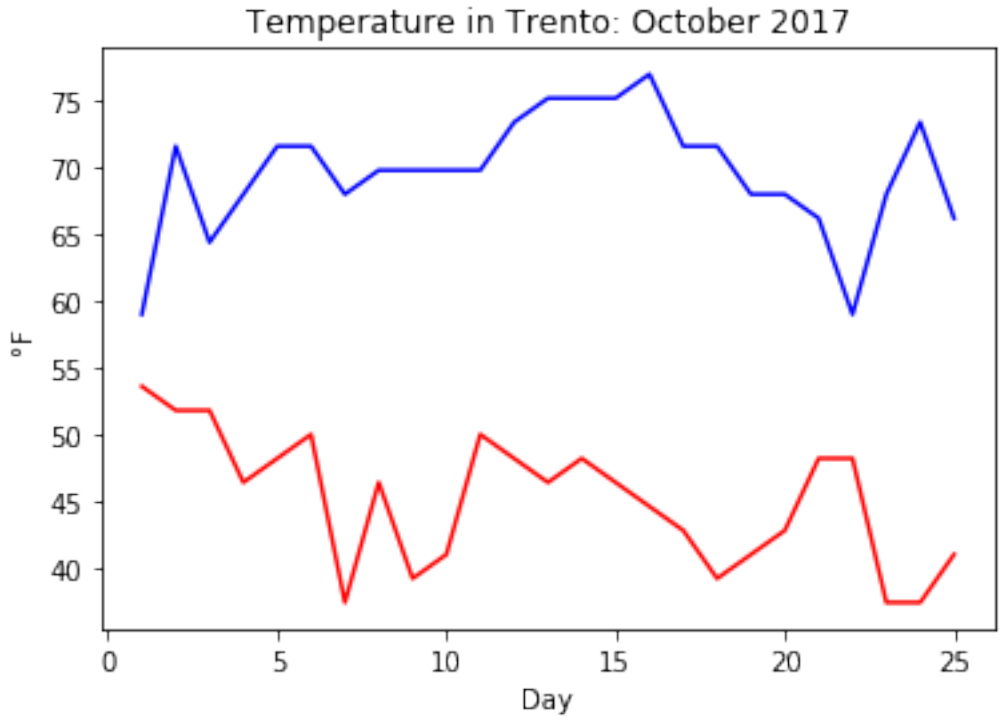
Show/Hide Solution

```
In [23]: %reset -s -f
         import numpy as np
         import matplotlib.pyplot as plt

         %matplotlib inline


         def Celsius2Farenheit(CA):
             FA = CA * 9/5 + 32
             return FA

         def Farenheit2Celsius(FA):
             CA = (FA - 32) * 5/9
             return CA

         tmin = [12, 11, 11, 8, 9, 10, 3, 8, 4, 5, 10, 9, 8, 9, 8, 7, 6, 4, 5, 6, 9, 9, 3, 3, 5]
         tmax =[15, 22, 18, 20, 22, 22, 20, 21, 21, 21, 21, 23, 24, 24, 24, 25, 22, 22, 20, 20, 19, 1

         minC = np.array(tmin)
         maxC = np.array(tmax)

         minF = Celsius2Farenheit(minC)
         maxF = Celsius2Farenheit(maxC)

         print("T min in Celsius:")
         print(minC)
         print("")
         print("T min in Farenheit")
```

```
        print(minF)
        print("T max in Celsius:")
        print(maxC)
        print("")
        print("T max in Farenheit")
        print(maxF)

        print(Farenheit2Celsius(minF))

        #plotting the data:
        x = np.arange(1,26,1)

        plt.plot(x, minC, 'r', x,maxC, 'b')
        plt.title("Temperature in Trento: October 2017")
        plt.ylabel('°C')
        plt.xlabel('Day')
        plt.show()
        #plt.close()
        plt.plot(x, minF, 'r', x,maxF, 'b')
        plt.title("Temperature in Trento: October 2017")
        plt.ylabel('°F')
        plt.xlabel('Day')
        plt.show()
```

```
T min in Celsius:
[12 11 11  8  9 10  3  8  4  5 10  9  8  9  8  7  6  4  5  6  9  9  3  3
  5]

T min in Farenheit
[53.6 51.8 51.8 46.4 48.2 50.  37.4 46.4 39.2 41.  50.  48.2 46.4 48.2
 46.4 44.6 42.8 39.2 41.  42.8 48.2 48.2 37.4 37.4 41. ]
T max in Celsius:
[15 22 18 20 22 22 20 21 21 21 21 23 24 24 24 25 22 22 20 20 19 15 20 23
 19]

T max in Farenheit
[59.  71.6 64.4 68.  71.6 71.6 68.  69.8 69.8 69.8 69.8 73.4 75.2 75.2
 75.2 77.  71.6 71.6 68.  68.  66.2 59.  68.  73.4 66.2]
[12. 11. 11.  8.  9. 10.  3.  8.  4.  5. 10.  9.  8.  9.  8.  7.  6.  4.
  5.  6.  9.  9.  3.  3.  5.]
```





3. Write some python code to solve the following system of linear equations:

$$\begin{cases} 3x - 4y + 2z - 5w = -9 \\ -4x + 4y + 10z + w = 32 \\ -x + 2y + -7z - w = -7 \\ x + y + z + w = 1 \end{cases}$$

Define then a function to compute the Euclidean norm of a ndarray in input. Hint: $||X||_2 = \sqrt{{x_1}^2 + {x_2}^2 + ... + {x_n}^2}$

Compute then the Euclidean norm of the solution.

Implement a function that takes two N-dimensional vectors and computes the euclidean distance between them. Compute the distance between the solution of the system of equations above and the vector V = [7,120,11,33]. Check also that the distance between a vector and itself is zero. The Euclidean distance between two vectors P and Q can be computed as: $\sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}$.

Show/Hide Solution

```
In [24]: import numpy as np
         from numpy import linalg

         def eucNorm(arr): ### equivalent to np.linalg.norm
             """computes the Euclidean norm of the array arr"""
             return np.sqrt(np.power(arr,2).sum())

         def eucDist(v1, v2):
             """computes the Euclidean distance of two N-dimensional vectors"""
             dV = np.power(v2-v1,2)
             return np.sqrt(dV.sum())
         #AX = b

         A = [[3, 4, 2, -5],[4, 4, 10, 1],[-1, 2, -7, -1],[1, 1, 1, 1]]
         Amat = np.array(A)

         b = np.array([-9, 32, -7, 1])
         print("A matrix")
         print(Amat)
         print("")
         print("b vector")
         print(b)
         print("")
         Xsol = linalg.solve(A,b)
         print("Solution X:")
         print(Xsol)
         print(Amat.dot(Xsol))
         print("")
         #Euclidean norm:
         print("Euclidean norm of {}:".format(Xsol))
         print(eucNorm(Xsol))

         #Euclidean distance
         V = np.array([7,120,11,33])
         print("\nThe Euclidean distance of")
         print(Xsol)
         print("\nand\n")
         print(V)

         print("\nis:")
         print(eucDist(Xsol,V))

         print("\neucDist(V,V) = {}".format(eucDist(V,V)))


A matrix
[[ 3  4  2 -5]
```

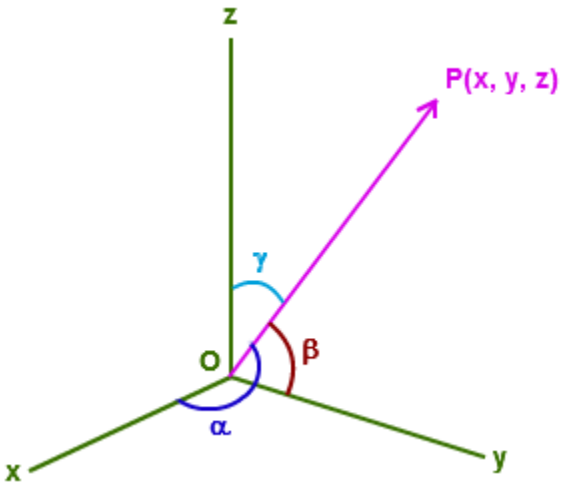
```
 [ 4  4 10  1]
 [-1  2 -7 -1]
 [ 1  1  1  1]]

b vector
[-9 32 -7  1]

Solution X:
[-15.88888889   9.28888889   5.64444444   1.95555556]
[-9. 32. -7.  1.]

Euclidean norm of [-15.88888889   9.28888889   5.64444444   1.95555556]:
19.35004386379691

The Euclidean distance of
[-15.88888889   9.28888889   5.64444444   1.95555556]

and

[  7 120  11  33]

is:
117.35966455387272

eucDist(V,V) = 0.0
```

4. Implement the following functions working on numpy ndarrays:

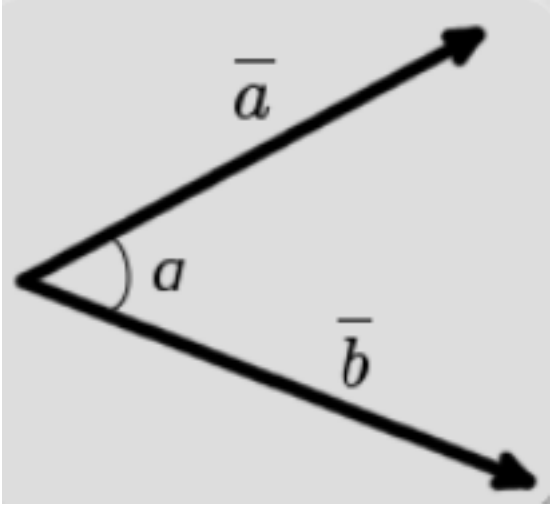
   (a) The component form of a vector `comp(A,B)` starting in point A and ending in point B is $\overline{AB}$. Hint: if $A = (a_1, ..., a_n)$ and $B = (b_1, ..., b_n)$, $\overline{AB} = (b_1 - a_1, ..., b_n - a_n)$.

   (b) The direction cosines of a 3D vector `dirCos(A)`. If $P = (x, y, z)$ the direction cosines are the cosines of the angles made by the vector with the three axes. They can be computed as:

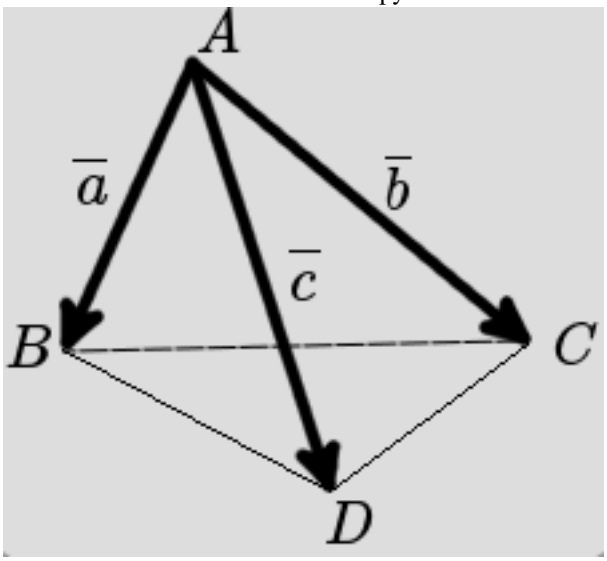
$$\frac{x}{\sqrt{(x^2 + y^2 + z^2}}, \frac{y}{\sqrt{(x^2 + y^2 + z^2}}, \frac{z}{\sqrt{(x^2 + y^2 + z^2}}$$



   (c) The angle between two vectors `angle(A,B)`. Given two vectors $A = (a_x, a_y, a_z)$ and $B = (b_x, b_y, b_z)$, $cos(\alpha) = \frac{A \cdot B}{||A||_2 ||B||_2}$ where the Euclidean norms are computed as in the previous example.



(d) The volume of the pyramid described by three 3D vectors $\overline{a}, \overline{b}, \overline{c}$:



the volume `pyrVol(a,b,c)` can be computed as: $\frac{1}{6}|det(M)|$, where $M = \begin{bmatrix} \overline{a} \\ \overline{b} \\ \overline{c} \end{bmatrix}$, det is the determinant of the matrix anx || represents the absolute value.

Define some numpy arrays starting from the following lists:

(a) A = [10, 3, 4, 7, 12], B = [6, 7, 8, 1, 1] and test `comp(A,B)`

(b) Va = [90,12,72] and test `dirCos(Va)`

(c) VA = [70,9,12] and VB = [1,-10,30] and test `angle(VA,VB)`

(d) a = [10, 20, 30], b = [4, 7, 21], c = [-5, 4, 12] and test `pyrVol(a,b,c)`

Check your solutions here[62].

Show/Hide Solution

[62] http://onlinemschool.com/math/assistance/

In [25]:
```
import numpy as np
from numpy import linalg

def comp(p1,p2):
    """component of vector starting in p1 and ending in p2"""
    return p2 - p1

def dirCos(v1):
    """direction cosines of v1"""
    norm = np.sqrt(np.power(v1,2).sum())
    return v1[0]/norm, v1[1]/norm, v1[2]/norm

def eucNorm(arr):
    """computes the Euclidean norm of the array arr"""
    return np.sqrt(np.power(arr,2).sum())

def angle(v1,v2):
    """computes the angle between two vectors v1 and v2"""
    dp = np.dot(v1,v2)
    cosVal = dp / (eucNorm(v1) * eucNorm(v2))
    return cosVal

def pyrVol(v1,v2,v3):
    """returns the volume of the pyramid described by v1,v2 and v3"""
    matM = np.concatenate((v1,v2,v3))
    matM = matM.reshape((v1.shape[0], 3))
    return abs(np.linalg.det(matM))/6


#Testing the functions
#1. comp(A,B)
A = np.array([10, 3, 4, 7, 12])
B = np.array([6, 7, 8, 1, 1])
print("A")
print(A)
print("")
print("B")
print(B)
print("")
print("comp(A,B)")
print(comp(A,B))
#2. dirCos(Va)
print("")
Va = np.array([90,12,72])
print("Va")
print(Va)
print("")
print("dirCos(Va)")
print(dirCos(Va))
#3. angle(VA,VB)
VA = np.array([70,9,12])
VB = np.array([1,-10,30])
print("")
print("VA")
print(VA)
print("")
print("VB")
print(VB)
print("")
```

```
        print("angle(VA,VB)")
        print(angle(VA,VB))
        print("")
        #4. pyrVol(a,b,c)
        a = np.array([10, 20, 30])
        b = np.array([ 4,  7, 21])
        c = np.array([-5,  4, 12])
        print("a")
        print(a)
        print("")
        print("b")
        print(b)
        print("")
        print("c")
        print(c)
        print("")
        print("pyrVol(a,b,c)")
        print(pyrVol(a,b,c))
```

```
A
[10  3  4  7 12]

B
[6 7 8 1 1]

comp(A,B)
[ -4   4   4  -6 -11]

Va
[90 12 72]

dirCos(Va)
(0.7766705596177466, 0.10355607461569954, 0.6213364476941973)

VA
[70  9 12]

VB
[  1 -10  30]

angle(VA,VB)
0.15011183829724634

a
[10 20 30]

b
[ 4  7 21]

c
[-5  4 12]

pyrVol(a,b,c)
254.99999999999977
```

5. Implement the power iteration algorithm which is a method to obtain an approximation of the biggest (in absolute value) eigenvalue of a matrix A.

The algorithm starts from a random vector b_0 and keeps computing $b_{k+1} = \frac{A \cdot b_k}{min(A \cdot b_k)}$. When the algorithm ends, the eigenvalue is actually the value: $min(A \cdot b_k)$

The powerIteration function will have the **maximum number of iterations** and **dThr** as parameters `powerIteration(A,maxIt, dThr)`. The computation will stop either when the maximum number of iterations `maxIt` is reached or when the Euclidean distance between `b_{k-1}` and `b_k` is lower than `dThr`.

Check the result against numpy eigenvalues obtained with `np.linalg.eig()` using various matrices.

Show/Hide Solution

In [26]:
```
import numpy as np
from numpy import linalg


def eucDist(v1, v2):
    """computes the Euclidean distance of two N-dimensional vectors"""
    dV = np.power(v2-v1,2)
    return np.sqrt(dV.sum())


def powerIteration(A,maxIt,dThr):
    b = np.random.randint(1,10,(A.shape[0]))
    it = 0
    dist = 1000000
    while( it < maxIt and dist > dThr ):
        tmp = np.dot(A,b)
        dist = eucDist(b,tmp)
        b = tmp/np.min(tmp)
        it += 1

    return np.min(tmp)



M = np.array([[1, 3, 7, 9], [ 4, 11, 7, 27], [22, 11, 57, 1], [4, 7, 27, 27]])

r = powerIteration(M,200, 0.0001)
print("Eigenvalue from power method: ", r)
print("FROM linalg:" , max(np.linalg.eig(M)[0]))

N = np.array([[0, 11, -5],[-2, 17, -7],[-4, 26, -10]])
print(N)
r = powerIteration(N,200, 0.0001)
print("Eigenvalue from power method: ", r)
print("FROM linalg:" , max(np.linalg.eig(N)[0]))
N1 = np.array([[2, -12],[1, -5]])
r = powerIteration(N1,200, 0.0001)
print("Eigenvalue from power method: ", r)
print("FROM linalg:" , max(np.linalg.eig(N1)[0]))
N2 = np.array([[1, 2, 0],[-2, 1, 2], [1,3,1]])

r = powerIteration(N2,200, 0.0001)
print("Eigenvalue from power method: ", r)
print("FROM linalg:" , max(np.linalg.eig(N2)[0]))

R = np.random.randint(0,20, size=(5,5))
r = powerIteration(R,200, 0.0001)
print("Eigenvalue from power method: ", r)
print("FROM linalg:" , max(np.linalg.eig(R)[0]))
```

```
Eigenvalue from power method:  68.24845123079758
```

```
FROM linalg: 68.24845123079754
[[  0  11  -5]
 [ -2  17  -7]
 [ -4  26 -10]]
Eigenvalue from power method:  3.9999999999999982
FROM linalg: 4.000000000000021
Eigenvalue from power method:  -2.0
FROM linalg: -0.9999999999999982
Eigenvalue from power method:  3.0
FROM linalg: (3.0000000000000004+0j)
Eigenvalue from power method:  50.69133949667758
FROM linalg: (50.69133949667753+0j)
```

6. Implement a function `movingAvg(A,n)` where A is a numpy one dimensional array and n is the window size on which computing the average, that outputs the moving average over a numpy one dimensional array. Es. A = [1,2,3,4,5] `movingAvg(A,2) = [1.5, 2.5, 3.5, 4.5]`, while `movingAvg(A,3) = [2,3,4]` without using for loops. Hint: use cumsum and clever slicing. Assess the smoothing effect of the moving average by creating a numpy array containing a sinusoidal wave with some additional noise (i.e. use np.sin and np.random.rand) and testing several values of the window size n.

Show/Hide Solution

In [27]:

```
import numpy as np
import matplotlib.pyplot as plt

def movingAvg(v, n):
    """computes the moving average of n values in the array v"""
    cs = np.cumsum(v)
    print(cs)
    print("\t",cs[n:])
    print("\t", cs[:-n])
    cs[n:] = cs[n:] - cs[:-n]
    print("CS:", cs)
    return cs[n - 1:] / n



A = np.arange(1,8,1)

print("A:")
print(A)
print(movingAvg(A,2))
print(movingAvg(A,3))

X = np.arange(0, 5, 0.005)
#B = np.random(100)
B = np.sin(2*np.pi*X)
B += np.random.rand(1000)
C = movingAvg(B,5)
D = movingAvg(B,10)
E = movingAvg(B,20)
#X = np.arange(0,B.shape[0])

plt.plot(X,B)
plt.title("No moving average")
plt.show()
plt.close()
plt.plot(np.arange(0,C.shape[0]),C)
```

```
        plt.title("Window size: 5")
        plt.show()
        plt.close()
        plt.title("Window size: 10")
        plt.plot(np.arange(0,D.shape[0]),D)
        plt.show()
        plt.close()
        plt.title("Window size: 20")
        plt.plot(np.arange(0,E.shape[0]),E)
        plt.show()
```

```
A:
[1 2 3 4 5 6 7]
[ 1  3  6 10 15 21 28]
         [ 6 10 15 21 28]
         [ 1  3  6 10 15]
CS: [ 1  3  5  7  9 11 13]
[1.5 2.5 3.5 4.5 5.5 6.5]
[ 1  3  6 10 15 21 28]
         [10 15 21 28]
         [ 1  3  6 10]
CS: [ 1  3  6  9 12 15 18]
[2. 3. 4. 5. 6.]
[2.48753144e-01 8.56321237e-01 1.51115369e+00 2.17751391e+00
 2.52592877e+00 3.63511225e+00 4.26961894e+00 5.33417086e+00
 6.28234002e+00 6.85876808e+00 7.98158289e+00 8.71682655e+00
 9.96605430e+00 1.09444751e+01 1.22519897e+01 1.33985118e+01
 1.46055198e+01 1.56158856e+01 1.71077960e+01 1.83138696e+01
 1.93255099e+01 2.05448101e+01 2.12014273e+01 2.21643140e+01
 2.35090346e+01 2.45062190e+01 2.58532031e+01 2.70320829e+01
 2.79380702e+01 2.90265075e+01 3.04054894e+01 3.18234427e+01
 3.32420959e+01 3.47560388e+01 3.62844487e+01 3.76068737e+01
 3.94082473e+01 4.06935638e+01 4.20592052e+01 4.38920094e+01
 4.56492599e+01 4.73134421e+01 4.83822522e+01 5.02776516e+01
 5.19741801e+01 5.39607155e+01 5.51022785e+01 5.69659665e+01
 5.81264861e+01 5.97415523e+01 6.08653723e+01 6.27128870e+01
 6.45182327e+01 6.60828954e+01 6.74821934e+01 6.85390488e+01
 7.02187648e+01 7.16482242e+01 7.33388630e+01 7.51655390e+01
 7.70921170e+01 7.88888011e+01 7.98302917e+01 8.11080244e+01
 8.27428420e+01 8.38054782e+01 8.52028215e+01 8.61179015e+01
 8.71622260e+01 8.80078283e+01 8.96105430e+01 9.06246227e+01
 9.17404877e+01 9.34186800e+01 9.48520631e+01 9.55910088e+01
 9.64402500e+01 9.77230403e+01 9.89376929e+01 9.97884927e+01
 1.01310492e+02 1.02486541e+02 1.03558001e+02 1.04656952e+02
 1.05868828e+02 1.06634764e+02 1.07458764e+02 1.08065756e+02
 1.08620073e+02 1.09903183e+02 1.10951751e+02 1.11721201e+02
 1.12197306e+02 1.12669805e+02 1.12915216e+02 1.13506067e+02
 1.13943196e+02 1.14733648e+02 1.15174190e+02 1.15385205e+02
 1.15409883e+02 1.15445722e+02 1.16062324e+02 1.16421913e+02
 1.16833159e+02 1.17573396e+02 1.18376353e+02 1.18375107e+02
 1.18789495e+02 1.18773827e+02 1.18485461e+02 1.18905101e+02
 1.18856994e+02 1.18843310e+02 1.19005848e+02 1.19382906e+02
 1.19530134e+02 1.19893743e+02 1.19631459e+02 1.19867422e+02
 1.19465273e+02 1.19805157e+02 1.19855222e+02 1.19409417e+02
 1.19672241e+02 1.19695990e+02 1.19220963e+02 1.18684164e+02
 1.18431851e+02 1.17667359e+02 1.17065812e+02 1.16663417e+02
 1.16193259e+02 1.15796092e+02 1.15197414e+02 1.14893192e+02
 1.14852221e+02 1.14051998e+02 1.13639601e+02 1.12830788e+02
 1.12596591e+02 1.12032357e+02 1.11629195e+02 1.10836558e+02
```

```
1.09999119e+02 1.09499487e+02 1.08862985e+02 1.08807855e+02
1.08575153e+02 1.08324310e+02 1.08228030e+02 1.07311946e+02
1.06866112e+02 1.06455026e+02 1.06424848e+02 1.05729307e+02
1.04987848e+02 1.04112225e+02 1.03160072e+02 1.03129308e+02
1.02848168e+02 1.02692440e+02 1.02044393e+02 1.01713049e+02
1.00872177e+02 1.00466798e+02 1.00567987e+02 1.00583750e+02
1.00077581e+02 1.00212070e+02 9.96347551e+01 9.97939189e+01
9.99647834e+01 1.00013875e+02 9.99153542e+01 1.00082535e+02
9.96910086e+01 9.98786403e+01 9.98590930e+01 9.92594228e+01
9.90188711e+01 9.86049285e+01 9.90509311e+01 9.90202600e+01
9.90358977e+01 9.92213797e+01 9.91641850e+01 9.89039374e+01
9.93579306e+01 9.92090406e+01 9.94113426e+01 9.93566685e+01
9.92058231e+01 9.98498714e+01 1.00635410e+02 1.01439810e+02
1.02221032e+02 1.02900971e+02 1.03171326e+02 1.03221016e+02
1.03628257e+02 1.03891902e+02 1.04087181e+02 1.04234716e+02
1.05085644e+02 1.05253506e+02 1.06211468e+02 1.06576557e+02
1.06904769e+02 1.07273364e+02 1.08254428e+02 1.08838534e+02
1.09627198e+02 1.10581714e+02 1.11868045e+02 1.13049079e+02
1.13801161e+02 1.14441685e+02 1.15032886e+02 1.15896568e+02
1.16746472e+02 1.17815519e+02 1.19136225e+02 1.20493162e+02
1.21461228e+02 1.22548262e+02 1.23458381e+02 1.24997038e+02
1.25824399e+02 1.27311551e+02 1.28899264e+02 1.30503752e+02
1.31607502e+02 1.32842058e+02 1.34305964e+02 1.35469792e+02
1.36745472e+02 1.37860281e+02 1.39249914e+02 1.40235407e+02
1.41986259e+02 1.43023509e+02 1.44510927e+02 1.45793654e+02
1.47353485e+02 1.49300606e+02 1.50938291e+02 1.51969216e+02
1.53397645e+02 1.54907168e+02 1.56443346e+02 1.58124245e+02
1.59399868e+02 1.60524290e+02 1.61909080e+02 1.63853174e+02
1.65022593e+02 1.66902493e+02 1.68414882e+02 1.69832088e+02
1.71665185e+02 1.73064670e+02 1.74718614e+02 1.76035394e+02
1.77844266e+02 1.79425297e+02 1.81001226e+02 1.82189688e+02
1.83790795e+02 1.85253937e+02 1.86302974e+02 1.87253668e+02
1.88820572e+02 1.90529850e+02 1.91716958e+02 1.93015049e+02
1.94557318e+02 1.95675854e+02 1.97265152e+02 1.98453810e+02
1.99862363e+02 2.01333290e+02 2.02684640e+02 2.03353096e+02
2.04463748e+02 2.05316173e+02 2.05804665e+02 2.06625845e+02
2.07252654e+02 2.08440430e+02 2.08782752e+02 2.10020726e+02
2.10624785e+02 2.11199635e+02 2.11403344e+02 2.11745011e+02
2.12271604e+02 2.13295004e+02 2.13457409e+02 2.14434121e+02
2.15303610e+02 2.15726362e+02 2.15990272e+02 2.16128908e+02
2.16618039e+02 2.16494679e+02 2.16322904e+02 2.16533557e+02
2.16352941e+02 2.16325891e+02 2.16238035e+02 2.16152488e+02
2.15915419e+02 2.15530307e+02 2.15220012e+02 2.15384502e+02
2.15877004e+02 2.16358308e+02 2.16231535e+02 2.15832406e+02
2.15883383e+02 2.15760781e+02 2.16112767e+02 2.15516759e+02
2.15615446e+02 2.15196738e+02 2.14709188e+02 2.14621582e+02
2.14097132e+02 2.13972836e+02 2.13681127e+02 2.13278136e+02
2.12988496e+02 2.12414805e+02 2.12245073e+02 2.11768923e+02
2.11224642e+02 2.11135544e+02 2.11130735e+02 2.10235861e+02
2.09517432e+02 2.08905657e+02 2.08752041e+02 2.08761615e+02
2.08748300e+02 2.08665560e+02 2.07970001e+02 2.07966451e+02
2.07217844e+02 2.06324243e+02 2.06275196e+02 2.05509110e+02
2.05200851e+02 2.04263646e+02 2.04002240e+02 2.03896272e+02
2.03186422e+02 2.02589562e+02 2.01995275e+02 2.01783769e+02
2.01070520e+02 2.00301492e+02 1.99821008e+02 1.99207721e+02
1.99142083e+02 1.98488819e+02 1.98114902e+02 1.98196743e+02
1.97986413e+02 1.98026622e+02 1.98157814e+02 1.98118424e+02
1.98047486e+02 1.98265341e+02 1.98530773e+02 1.98275488e+02
1.97661810e+02 1.97293292e+02 1.96808223e+02 1.96612802e+02
```

```
1.96156307e+02 1.96198341e+02 1.96045322e+02 1.96431667e+02
1.96917708e+02 1.97010602e+02 1.96859646e+02 1.97054729e+02
1.97583366e+02 1.97651361e+02 1.97894422e+02 1.97887084e+02
1.98093838e+02 1.98277408e+02 1.98338441e+02 1.98687873e+02
1.98872920e+02 1.99151847e+02 1.99614027e+02 2.00333211e+02
2.00666718e+02 2.01622288e+02 2.02547397e+02 2.02690196e+02
2.03069171e+02 2.03671741e+02 2.03963750e+02 2.04530370e+02
2.05519157e+02 2.06478663e+02 2.07410064e+02 2.08459330e+02
2.09032379e+02 2.09771225e+02 2.10873246e+02 2.12206472e+02
2.13231903e+02 2.14023645e+02 2.14589707e+02 2.15862127e+02
2.16457796e+02 2.17443382e+02 2.18611343e+02 2.20194767e+02
2.20968808e+02 2.22081858e+02 2.22835139e+02 2.23927861e+02
2.25320606e+02 2.26389829e+02 2.27408596e+02 2.28351379e+02
2.29772848e+02 2.31328860e+02 2.32877123e+02 2.34716991e+02
2.35624521e+02 2.37189473e+02 2.38719641e+02 2.40249262e+02
2.42163088e+02 2.43140254e+02 2.44805319e+02 2.46594915e+02
2.48087009e+02 2.49408663e+02 2.51191617e+02 2.52284422e+02
2.53724485e+02 2.55243944e+02 2.56937900e+02 2.58028292e+02
2.59254078e+02 2.60659942e+02 2.62275351e+02 2.64150001e+02
2.65751114e+02 2.66860492e+02 2.68809655e+02 2.70641735e+02
2.72095512e+02 2.73958741e+02 2.75429898e+02 2.77270959e+02
2.79005683e+02 2.80864976e+02 2.82661066e+02 2.83557842e+02
2.84576942e+02 2.85793157e+02 2.87554316e+02 2.88644500e+02
2.89575481e+02 2.91211897e+02 2.92387260e+02 2.94002242e+02
2.94847020e+02 2.96169449e+02 2.97247137e+02 2.97936531e+02
2.99220779e+02 3.00030261e+02 3.00605704e+02 3.01174690e+02
3.01717522e+02 3.03079245e+02 3.04244908e+02 3.05540119e+02
3.06580825e+02 3.07448503e+02 3.08061967e+02 3.09338920e+02
3.09949799e+02 3.10638591e+02 3.11204218e+02 3.12340179e+02
3.12640171e+02 3.13062267e+02 3.13805406e+02 3.13900025e+02
3.14507274e+02 3.14953510e+02 3.15174719e+02 3.15319024e+02
3.15708204e+02 3.15919697e+02 3.16188835e+02 3.16308169e+02
3.17029973e+02 3.16884422e+02 3.16672209e+02 3.16676862e+02
3.16899765e+02 3.17161793e+02 3.17133271e+02 3.17678558e+02
3.17548698e+02 3.17761063e+02 3.17862819e+02 3.18113789e+02
3.18502230e+02 3.18779116e+02 3.18906254e+02 3.18943191e+02
3.18594142e+02 3.18034721e+02 3.17368388e+02 3.16860179e+02
3.16521947e+02 3.16253788e+02 3.16217855e+02 3.16349515e+02
3.15622508e+02 3.14868770e+02 3.14582158e+02 3.14436549e+02
3.14379873e+02 3.14397950e+02 3.14451600e+02 3.13910521e+02
3.13339799e+02 3.12527314e+02 3.12243666e+02 3.11924511e+02
3.11804286e+02 3.10913856e+02 3.10419518e+02 3.10005038e+02
3.09248568e+02 3.08418087e+02 3.08277668e+02 3.07336696e+02
3.06809291e+02 3.05929563e+02 3.05394507e+02 3.05386781e+02
3.04828200e+02 3.04709408e+02 3.03858140e+02 3.03169099e+02
3.02621835e+02 3.02080766e+02 3.01822373e+02 3.01249337e+02
3.01058277e+02 3.00806457e+02 3.00329312e+02 2.99900330e+02
2.99670529e+02 2.98913491e+02 2.98926881e+02 2.98790147e+02
2.98745976e+02 2.98532788e+02 2.97914297e+02 2.97612225e+02
2.97333052e+02 2.96992783e+02 2.96385309e+02 2.96509657e+02
2.96031656e+02 2.96075880e+02 2.96243271e+02 2.96369016e+02
2.96846405e+02 2.96495712e+02 2.96937100e+02 2.96569143e+02
2.96735935e+02 2.96801441e+02 2.97016607e+02 2.97102716e+02
2.97044593e+02 2.96845573e+02 2.97176341e+02 2.97862684e+02
2.98110566e+02 2.98239322e+02 2.98257063e+02 2.98310964e+02
2.98532360e+02 2.98663785e+02 2.98991615e+02 2.99151873e+02
2.99342811e+02 3.00355522e+02 3.00705023e+02 3.01482849e+02
3.02504994e+02 3.03240395e+02 3.03702781e+02 3.04241115e+02
3.05042224e+02 3.05967606e+02 3.06742825e+02 3.07978295e+02
```

```
3.09211071e+02 3.10647324e+02 3.11212103e+02 3.12669878e+02
3.13650232e+02 3.15141511e+02 3.16469720e+02 3.18118381e+02
3.19562210e+02 3.20633862e+02 3.21863894e+02 3.22990394e+02
3.24125819e+02 3.25176878e+02 3.26481866e+02 3.27990686e+02
3.29112354e+02 3.30497476e+02 3.31491163e+02 3.32542015e+02
3.33493648e+02 3.35382135e+02 3.36315771e+02 3.37435232e+02
3.38999155e+02 3.40040819e+02 3.41891298e+02 3.43586835e+02
3.45535512e+02 3.47030836e+02 3.48323355e+02 3.49868417e+02
3.51797263e+02 3.53317531e+02 3.54584738e+02 3.56461643e+02
3.57831589e+02 3.58828534e+02 3.60068334e+02 3.61374255e+02
3.63215320e+02 3.64649740e+02 3.66062911e+02 3.67359307e+02
3.69191041e+02 3.71076949e+02 3.72998616e+02 3.74293111e+02
3.76164086e+02 3.77846972e+02 3.79398968e+02 3.80504599e+02
3.81565385e+02 3.82558513e+02 3.84290287e+02 3.85374518e+02
3.86598126e+02 3.87842195e+02 3.89349335e+02 3.90900677e+02
3.91724296e+02 3.92812513e+02 3.94292791e+02 3.95723732e+02
3.96413931e+02 3.97132398e+02 3.97972423e+02 3.98556824e+02
3.99463240e+02 4.00024848e+02 4.01018845e+02 4.01662550e+02
4.02627108e+02 4.03083371e+02 4.04368272e+02 4.05579824e+02
4.06220311e+02 4.06680633e+02 4.07118413e+02 4.07758241e+02
4.07923567e+02 4.08657380e+02 4.09128474e+02 4.09537291e+02
4.10346656e+02 4.11024281e+02 4.11915824e+02 4.12173652e+02
4.12945861e+02 4.13559394e+02 4.13729437e+02 4.14132960e+02
4.14172840e+02 4.14768248e+02 4.14571659e+02 4.14445355e+02
4.14260264e+02 4.14266142e+02 4.14585596e+02 4.14658513e+02
4.14664435e+02 4.14155940e+02 4.14045515e+02 4.13546985e+02
4.13167453e+02 4.13486940e+02 4.13064914e+02 4.13261940e+02
4.13380286e+02 4.12832326e+02 4.12709069e+02 4.12074620e+02
4.12031995e+02 4.11879302e+02 4.11882224e+02 4.11534528e+02
4.11605063e+02 4.10793670e+02 4.10210252e+02 4.10034298e+02
4.09547580e+02 4.08802777e+02 4.07980211e+02 4.07856669e+02
4.07378756e+02 4.07300746e+02 4.07065452e+02 4.06499261e+02
4.05890485e+02 4.05418435e+02 4.05315380e+02 4.05057097e+02
4.04064223e+02 4.03758874e+02 4.03678382e+02 4.03389331e+02
4.02568310e+02 4.02056266e+02 4.01204467e+02 4.00575774e+02
4.00530604e+02 4.00477993e+02 3.99792246e+02 3.99171584e+02
3.98820740e+02 3.98843057e+02 3.98061081e+02 3.97400243e+02
3.97368973e+02 3.96969859e+02 3.96992513e+02 3.96317289e+02
3.96005630e+02 3.95504819e+02 3.95012344e+02 3.94669066e+02
3.94331631e+02 3.93938866e+02 3.94124869e+02 3.94149506e+02
3.94192506e+02 3.93821107e+02 3.93761393e+02 3.93927665e+02
3.94135470e+02 3.93917917e+02 3.94152963e+02 3.94379816e+02
3.94039569e+02 3.94451524e+02 3.94467066e+02 3.94556328e+02
3.94636573e+02 3.94865681e+02 3.95177833e+02 3.95397022e+02
3.96015120e+02 3.96424712e+02 3.96638759e+02 3.96899016e+02
3.97584521e+02 3.97838605e+02 3.97987269e+02 3.98015242e+02
3.98891269e+02 3.99841226e+02 4.00024136e+02 4.00452718e+02
4.00753424e+02 4.01025757e+02 4.02113005e+02 4.02388025e+02
4.03617201e+02 4.03992643e+02 4.05165130e+02 4.06070374e+02
4.06806416e+02 4.07545907e+02 4.08729050e+02 4.09497614e+02
4.10636687e+02 4.11663054e+02 4.12683847e+02 4.14147092e+02
4.15289522e+02 4.16729291e+02 4.18092289e+02 4.18792158e+02
4.20249815e+02 4.21173792e+02 4.22805910e+02 4.23598945e+02
4.24702531e+02 4.25592419e+02 4.26877025e+02 4.28524128e+02
4.29666643e+02 4.30678320e+02 4.31884894e+02 4.33589780e+02
4.34634991e+02 4.35780108e+02 4.36778737e+02 4.38425328e+02
4.39771617e+02 4.41042751e+02 4.42729961e+02 4.44041855e+02
4.45751913e+02 4.47554801e+02 4.48764579e+02 4.50733959e+02
4.51894344e+02 4.53184692e+02 4.54364487e+02 4.55709499e+02
```

```
4.57187587e+02 4.58705325e+02 4.60551045e+02 4.62428182e+02
4.63630573e+02 4.65229383e+02 4.66309463e+02 4.67728726e+02
4.69002116e+02 4.70259498e+02 4.71671858e+02 4.73319441e+02
4.74293450e+02 4.76063630e+02 4.77674751e+02 4.78711992e+02
4.80495481e+02 4.81828874e+02 4.83637699e+02 4.84625114e+02
4.85930535e+02 4.86970894e+02 4.88004037e+02 4.89302209e+02
4.90908475e+02 4.92375051e+02 4.93736416e+02 4.94908497e+02
4.96418581e+02 4.97473025e+02 4.98882684e+02 5.00225707e+02
5.00921297e+02 5.02146512e+02 5.02584463e+02 5.03304440e+02
5.03902132e+02 5.04747733e+02 5.05793603e+02 5.06170271e+02
5.06933883e+02 5.08090438e+02 5.08506466e+02 5.09340042e+02
5.10058255e+02 5.10162427e+02 5.10701044e+02 5.11441225e+02
5.11485200e+02 5.12333311e+02 5.12790602e+02 5.12727155e+02
5.12826235e+02 5.13623476e+02 5.14018415e+02 5.13907744e+02
5.13946599e+02 5.14124311e+02 5.13836244e+02 5.13909122e+02
5.14030456e+02 5.13876986e+02 5.14039846e+02 5.14339095e+02
5.14093176e+02 5.14204634e+02 5.14308430e+02 5.14694887e+02
5.14885378e+02 5.15120816e+02 5.14973812e+02 5.14497849e+02
5.14809117e+02 5.14231366e+02 5.13973854e+02 5.13291836e+02
5.13465174e+02 5.13639944e+02 5.13550316e+02 5.13073228e+02
5.12483283e+02 5.11887844e+02 5.11138831e+02 5.10773634e+02
5.10010624e+02 5.09409600e+02 5.09106530e+02 5.08893193e+02
5.07966409e+02 5.07436231e+02 5.07119773e+02 5.06997102e+02
5.06490140e+02 5.06471657e+02 5.05745175e+02 5.04763122e+02
5.04248848e+02 5.03505455e+02 5.03329173e+02 5.02562439e+02
5.01875041e+02 5.01670707e+02 5.01393735e+02 5.00964098e+02
5.00686759e+02 5.00129479e+02 4.99166206e+02 4.98217268e+02
4.97777433e+02 4.96919843e+02 4.96041142e+02 4.96088904e+02
4.96043080e+02 4.95304100e+02 4.94428458e+02 4.94509384e+02
4.93943381e+02 4.93302198e+02 4.93184689e+02 4.92503438e+02
4.91997574e+02 4.92222558e+02 4.92133052e+02 4.91946623e+02
4.91659995e+02 4.91773184e+02 4.91276717e+02 4.91631148e+02
4.91904486e+02 4.91960059e+02 4.91467139e+02 4.91658953e+02
4.92090484e+02 4.92161070e+02 4.92089516e+02 4.91812645e+02
4.92199422e+02 4.92745706e+02 4.92536940e+02 4.93016934e+02
4.92785304e+02 4.93534216e+02 4.93961893e+02 4.94357897e+02
4.94528514e+02 4.95363697e+02 4.95566812e+02 4.96363548e+02]
        [  3.63511225   4.26961894   5.33417086   6.28234002   6.85876808
  7.98158289   8.71682655   9.9660543   10.94447506  12.25198972
 13.39851181  14.60551976  15.61588556  17.10779599  18.31386957
 19.32550987  20.54481013  21.20142732  22.164314    23.50903465
 24.50621904  25.85320309  27.03208286  27.93807017  29.02650751
 30.40548941  31.82344275  33.24209592  34.75603877  36.28444872
 37.60687368  39.40824733  40.69356382  42.05920524  43.89200936
 45.64925986  47.31344213  48.38225218  50.27765156  51.97418011
 53.96071546  55.10227846  56.96596648  58.12648615  59.74155227
 60.86537225  62.71288704  64.51823273  66.08289543  67.48219343
 68.53904877  70.21876479  71.64822424  73.338863    75.16553901
 77.09211703  78.88880114  79.83029171  81.1080244   82.74284202
 83.80547822  85.2028215   86.11790152  87.16222597  88.00782834
 89.61054304  90.62462274  91.74048766  93.41868002  94.85206305
 95.59100876  96.44025003  97.72304029  98.93769287  99.78849275
101.310492   102.48654133 103.55800093 104.65695232 105.86882802
106.63476352 107.45876387 108.06575551 108.62007307 109.90318338
110.95175117 111.72120109 112.1973056  112.66980532 112.9152158
113.50606689 113.94319601 114.73364781 115.17419017 115.3852046
115.40988333 115.4457222  116.06232446 116.42191299 116.83315897
117.57339579 118.37635343 118.37510717 118.78949549 118.77382676
118.48546076 118.9051015  118.85699409 118.8433101  119.00584792
```

```
119.38290587 119.53013404 119.89374328 119.63145852 119.86742198
119.46527267 119.80515728 119.85522156 119.40941737 119.67224086
119.69598988 119.2209629  118.68416381 118.43185128 117.66735898
117.06581206 116.66341696 116.19325901 115.79609241 115.19741444
114.89319226 114.85222081 114.05199804 113.63960066 112.830788
112.59659117 112.03235718 111.62919534 110.83655841 109.99911892
109.49948686 108.86298489 108.80785488 108.5751534  108.32431046
108.2280302  107.31194607 106.86611182 106.45502592 106.4248476
105.72930678 104.98784831 104.11222549 103.16007196 103.12930759
102.84816762 102.69243977 102.04439339 101.71304893 100.87217714
100.46679821 100.56798667 100.58374989 100.07758092 100.2120705
 99.63475513  99.79391894  99.9647834  100.01387492  99.91535423
100.08253541  99.69100859  99.87864028  99.85909298  99.25942279
 99.01887105  98.60492854  99.05093113  99.02026002  99.03589771
 99.22137973  99.16418505  98.90393743  99.35793061  99.2090406
 99.41134259  99.35666851  99.20582311  99.84987138 100.63540956
101.43980975 102.22103202 102.90097104 103.17132567 103.2210163
103.62825747 103.89190237 104.08718053 104.23471602 105.08564362
105.25350554 106.21146761 106.57655749 106.90476946 107.2733636
108.2544284  108.83853353 109.62719755 110.58171424 111.8680447
113.04907946 113.80116104 114.44168526 115.03288637 115.89656839
116.74647179 117.81551941 119.13622473 120.49316204 121.461228
122.54826173 123.45838132 124.99703791 125.82439922 127.31155148
128.89926387 130.503752   131.60750249 132.84205766 134.30596397
135.4697924  136.74547225 137.86028116 139.24991353 140.2354066
141.986259   143.02350913 144.51092744 145.7936543  147.3534845
149.30060618 150.93829113 151.96921553 153.3976447  154.90716811
156.44334561 158.12424468 159.3998675  160.52429003 161.90908041
163.85317447 165.02259262 166.90249333 168.41488245 169.83208755
171.66518548 173.06467021 174.71861433 176.03539428 177.84426573
179.42529727 181.001226   182.18968843 183.790795   185.25393663
186.3029739  187.25366773 188.82057245 190.52985012 191.71695758
193.01504852 194.55731827 195.67585359 197.26515206 198.45381028
199.86236265 201.33328974 202.68464036 203.35309624 204.46374835
205.31617311 205.80466535 206.62584549 207.25265411 208.44043034
208.78275196 210.02072579 210.62478453 211.19963466 211.40334448
211.74501127 212.271604   213.29500373 213.45740918 214.43412147
215.30361    215.72636164 215.990272   216.12890782 216.61803929
216.49467942 216.32290417 216.53355665 216.35294084 216.32589072
216.23803464 216.15248791 215.91541859 215.53030693 215.22001193
215.38450169 215.87700423 216.35830782 216.23153512 215.83240616
215.88338267 215.76078096 216.11276675 215.51675909 215.61544642
215.19673814 214.70918813 214.62158163 214.09713158 213.97283568
213.6811272  213.27813562 212.9884955  212.41480499 212.24507302
211.76892337 211.22464187 211.13554416 211.13073459 210.23586113
209.51743161 208.90565729 208.75204061 208.76161528 208.74829973
208.66555973 207.9700013  207.96645058 207.21784389 206.32424348
206.27519609 205.50910979 205.20085133 204.26364572 204.00224012
203.89627199 203.18642164 202.58956177 201.99527479 201.78376936
201.07052009 200.30149242 199.82100758 199.20772136 199.14208343
198.48881874 198.11490151 198.19674309 197.98641286 198.02662169
198.15781438 198.11842423 198.04748605 198.26534055 198.53077271
198.27548761 197.66181028 197.29329245 196.80822317 196.61280249
196.15630656 196.19834099 196.04532225 196.43166672 196.91770772
197.01060212 196.8596464  197.05472893 197.58336553 197.65136096
197.89442224 197.8870839  198.09383817 198.27740846 198.33844061
198.68787253 198.87292012 199.15184667 199.6140266  200.33321086
200.66671832 201.62228785 202.54739692 202.69019553 203.06917128
203.67174126 203.96375046 204.5303697  205.51915711 206.4786627
```

```
207.41006412 208.45933044 209.03237868 209.77122469 210.87324646
212.20647172 213.23190345 214.02364452 214.58970657 215.86212678
216.45779613 217.44338226 218.61134346 220.19476679 220.96880844
222.08185754 222.83513937 223.92786142 225.32060572 226.38982869
227.40859563 228.35137944 229.77284761 231.32885964 232.87712346
234.71699101 235.62452127 237.18947255 238.71964127 240.24926165
242.16308849 243.14025385 244.80531944 246.59491485 248.0870093
249.40866251 251.19161737 252.28442226 253.72448463 255.24394357
256.93789998 258.02829227 259.2540785  260.65994203 262.2753514
264.15000052 265.75111394 266.86049217 268.80965547 270.64173489
272.09551217 273.95874092 275.4298982  277.27095889 279.00568331
280.86497624 282.66106574 283.55784158 284.57694151 285.79315676
287.55431645 288.64450038 289.57548127 291.21189701 292.38726005
294.00224243 294.84702    296.16944937 297.24713712 297.93653094
299.22077934 300.03026147 300.60570379 301.1746895  301.71752171
303.07924517 304.24490838 305.54011863 306.58082549 307.44850334
308.0619667  309.33892006 309.949799   310.63859119 311.20421768
312.34017908 312.6401707  313.06226701 313.8054062  313.90002457
314.50727395 314.95350969 315.17471915 315.31902412 315.70820363
315.91969674 316.18883532 316.30816946 317.02997326 316.88442159
316.67220855 316.67686236 316.89976471 317.16179329 317.13327074
317.67855824 317.54869756 317.76106281 317.86281871 318.1137892
318.50222961 318.77911621 318.90625419 318.94319081 318.59414187
318.03472067 317.36838804 316.86017868 316.52194692 316.25378818
316.21785474 316.34951509 315.62250764 314.86876975 314.5821578
314.43654935 314.37987268 314.39795013 314.45159989 313.91052081
313.33979948 312.52731447 312.24366575 311.92451094 311.80428629
310.91385594 310.41951815 310.00503811 309.24856842 308.41808727
308.27766811 307.33669647 306.80929064 305.92956268 305.39450674
305.38678073 304.82819983 304.70940798 303.85814039 303.16909878
302.621835   302.08076637 301.82237337 301.24933687 301.05827668
300.80645706 300.32931152 299.90032963 299.6705294  298.91349102
298.92688076 298.79014691 298.74597613 298.53278806 297.91429654
297.61222538 297.33305185 296.99278298 296.38530931 296.5096565
296.03165571 296.07588046 296.24327117 296.36901607 296.84640465
296.49571231 296.93710017 296.56914252 296.73593482 296.80144052
297.01660738 297.10271615 297.04459318 296.84557284 297.17634134
297.86268374 298.11056646 298.23932196 298.25706345 298.31096361
298.53236006 298.66378488 298.99161509 299.15187287 299.34281097
300.35552162 300.70502319 301.48284884 302.50499427 303.24039494
303.70278081 304.24111488 305.04222363 305.96760561 306.7428252
307.9782953  309.21107062 310.64732384 311.21210319 312.66987786
313.6502319  315.14151145 316.46972021 318.11838083 319.56221039
320.6338618  321.8638936  322.99039382 324.1258189  325.17687842
326.4818657  327.99068622 329.11235442 330.49747626 331.49116323
332.54201504 333.49364845 335.38213452 336.31577136 337.43523209
338.99915536 340.04081865 341.89129831 343.58683523 345.53551245
347.03083634 348.32335473 349.86841726 351.79726271 353.31753071
354.58473774 356.46164309 357.83158857 358.82853388 360.0683336
361.37425545 363.21532017 364.6497401  366.06291055 367.3593065
369.19104114 371.07694869 372.9986155  374.2931114  376.16408589
377.84697199 379.39896782 380.50459932 381.56538451 382.55851291
384.29028651 385.37451819 386.59812567 387.84219458 389.3493348
390.90067654 391.72429635 392.81251258 394.29279145 395.72373181
396.41393082 397.13239755 397.97242304 398.55682352 399.4632402
400.0248484  401.01884529 401.66255012 402.62710774 403.0833713
404.36827216 405.57982447 406.2203113  406.68063314 407.11841266
407.75824066 407.9235667  408.65738012 409.12847355 409.53729088
410.34665585 411.02428055 411.91582385 412.17365177 412.9458613
```

```
413.55939403 413.72943736 414.13295956 414.17283963 414.76824844
414.57165876 414.4453552  414.26026394 414.26614205 414.58559572
414.65851267 414.66443532 414.15593987 414.0455148  413.5469852
413.1674532  413.48694008 413.0649143  413.26194007 413.38028634
412.83232579 412.70906912 412.07461992 412.03199484 411.8793021
411.88222367 411.53452765 411.60506281 410.79366973 410.21025161
410.03429769 409.54757985 408.80277658 407.98021084 407.85666918
407.37875564 407.30074563 407.0654516  406.49926104 405.89048481
405.41843481 405.31538006 405.05709668 404.06422292 403.75887421
403.67838161 403.38933081 402.56830987 402.05626603 401.20446734
400.57577428 400.53060407 400.47799262 399.79224631 399.17158367
398.82074002 398.84305655 398.0610814  397.40024341 397.36897319
396.9698589  396.99251331 396.31728918 396.00562984 395.5048189
395.01234446 394.66906642 394.33163062 393.93886643 394.12486857
394.14950598 394.19250586 393.82110745 393.76139288 393.92766526
394.13547038 393.91791746 394.15296342 394.3798159  394.03956872
394.45152369 394.46706586 394.55632842 394.63657305 394.86568113
395.17783338 395.39702184 396.0151205  396.42471201 396.63875865
396.89901594 397.58452132 397.83860495 397.98726923 398.01524166
398.8912685  399.84122571 400.02413642 400.45271847 400.75342377
401.02575671 402.11300477 402.38802527 403.61720082 403.99264278
405.16513043 406.07037445 406.80641649 407.54590676 408.7290502
409.49761399 410.63668658 411.66305408 412.68384652 414.14709207
415.28952238 416.72929104 418.09228856 418.79215767 420.24981483
421.17379186 422.80591014 423.5989454  424.70253067 425.59241863
426.87702475 428.52412776 429.66664304 430.67831996 431.88489368
433.58978035 434.63499136 435.78010843 436.77873688 438.42532769
439.77161745 441.04275112 442.72996067 444.04185497 445.7519135
447.55480123 448.76457878 450.73395944 451.89434412 453.18469158
454.36448687 455.70949909 457.18758671 458.70532454 460.55104528
462.42818153 463.63057264 465.22938344 466.30946266 467.7287262
469.00211626 470.25949777 471.6718585  473.31944076 474.29345047
476.06363033 477.67475079 478.7119922  480.49548104 481.82887384
483.63769941 484.6251139  485.93053534 486.97089445 488.00403663
489.3022088  490.90847497 492.37505069 493.73641608 494.90849691
496.41858067 497.47302545 498.88268443 500.22570749 500.92129651
502.14651247 502.58446292 503.30444035 503.90213235 504.74773322
505.79360338 506.17027085 506.93388294 508.09043821 508.50646607
509.34004168 510.05825519 510.16242719 510.70104391 511.44122506
511.48520049 512.33331122 512.79060211 512.72715485 512.82623523
513.62347646 514.01841488 513.9077442  513.94659882 514.12431134
513.83624441 513.90912201 514.03045609 513.87698607 514.03984578
514.3390954  514.09317595 514.20463444 514.30842988 514.69488681
514.88537772 515.12081594 514.97381186 514.49784858 514.80911677
514.23136575 513.97385444 513.29183647 513.46517408 513.63994401
513.55031608 513.07322835 512.48328282 511.88784412 511.13883147
510.77363389 510.01062412 509.40960016 509.10653015 508.89319299
507.96640918 507.43623148 507.11977291 506.99710212 506.49013966
506.47165719 505.74517503 504.76312178 504.24884791 503.50545515
503.32917282 502.56243893 501.87504142 501.67070689 501.39373544
500.96409833 500.68675914 500.12947925 499.16620613 498.21726757
497.77743285 496.91984306 496.04114205 496.08890406 496.04307965
495.30410035 494.42845789 494.50938366 493.94338103 493.30219806
493.18468918 492.5034379  491.99757426 492.22255787 492.13305202
491.94662303 491.65999454 491.77318362 491.27671711 491.63114786
491.90448562 491.96005922 491.46713862 491.65895285 492.09048352
492.16107008 492.08951561 491.81264507 492.19942162 492.74570555
492.5369403  493.01693375 492.78530435 493.53421603 493.96189273
494.35789733 494.52851393 495.36369728 495.56681239 496.36354825]
```

```
        [2.48753144e-01 8.56321237e-01 1.51115369e+00 2.17751391e+00
 2.52592877e+00 3.63511225e+00 4.26961894e+00 5.33417086e+00
 6.28234002e+00 6.85876808e+00 7.98158289e+00 8.71682655e+00
 9.96605430e+00 1.09444751e+01 1.22519897e+01 1.33985118e+01
 1.46055198e+01 1.56158856e+01 1.71077960e+01 1.83138696e+01
 1.93255099e+01 2.05448101e+01 2.12014273e+01 2.21643140e+01
 2.35090346e+01 2.45062190e+01 2.58532031e+01 2.70320829e+01
 2.79380702e+01 2.90265075e+01 3.04054894e+01 3.18234427e+01
 3.32420959e+01 3.47560388e+01 3.62844487e+01 3.76068737e+01
 3.94082473e+01 4.06935638e+01 4.20592052e+01 4.38920094e+01
 4.56492599e+01 4.73134421e+01 4.83822522e+01 5.02776516e+01
 5.19741801e+01 5.39607155e+01 5.51022785e+01 5.69659665e+01
 5.81264861e+01 5.97415523e+01 6.08653723e+01 6.27128870e+01
 6.45182327e+01 6.60828954e+01 6.74821934e+01 6.85390488e+01
 7.02187648e+01 7.16482242e+01 7.33388630e+01 7.51655390e+01
 7.70921170e+01 7.88888011e+01 7.98302917e+01 8.11080244e+01
 8.27428420e+01 8.38054782e+01 8.52028215e+01 8.61179015e+01
 8.71622260e+01 8.80078283e+01 8.96105430e+01 9.06246227e+01
 9.17404877e+01 9.34186800e+01 9.48520631e+01 9.55910088e+01
 9.64402500e+01 9.77230403e+01 9.89376929e+01 9.97884927e+01
 1.01310492e+02 1.02486541e+02 1.03558001e+02 1.04656952e+02
 1.05868828e+02 1.06634764e+02 1.07458764e+02 1.08065756e+02
 1.08620073e+02 1.09903183e+02 1.10951751e+02 1.11721201e+02
 1.12197306e+02 1.12669805e+02 1.12915216e+02 1.13506067e+02
 1.13943196e+02 1.14733648e+02 1.15174190e+02 1.15385205e+02
 1.15409883e+02 1.15445722e+02 1.16062324e+02 1.16421913e+02
 1.16833159e+02 1.17573396e+02 1.18376353e+02 1.18375107e+02
 1.18789495e+02 1.18773827e+02 1.18485461e+02 1.18905101e+02
 1.18856994e+02 1.18843310e+02 1.19005848e+02 1.19382906e+02
 1.19530134e+02 1.19893743e+02 1.19631459e+02 1.19867422e+02
 1.19465273e+02 1.19805157e+02 1.19855222e+02 1.19409417e+02
 1.19672241e+02 1.19695990e+02 1.19220963e+02 1.18684164e+02
 1.18431851e+02 1.17667359e+02 1.17065812e+02 1.16663417e+02
 1.16193259e+02 1.15796092e+02 1.15197414e+02 1.14893192e+02
 1.14852221e+02 1.14051998e+02 1.13639601e+02 1.12830788e+02
 1.12596591e+02 1.12032357e+02 1.11629195e+02 1.10836558e+02
 1.09999119e+02 1.09499487e+02 1.08862985e+02 1.08807855e+02
 1.08575153e+02 1.08324310e+02 1.08228030e+02 1.07311946e+02
 1.06866112e+02 1.06455026e+02 1.06424848e+02 1.05729307e+02
 1.04987848e+02 1.04112225e+02 1.03160072e+02 1.03129308e+02
 1.02848168e+02 1.02692440e+02 1.02044393e+02 1.01713049e+02
 1.00872177e+02 1.00466798e+02 1.00567987e+02 1.00583750e+02
 1.00077581e+02 1.00212070e+02 9.96347551e+01 9.97939189e+01
 9.99647834e+01 1.00013875e+02 9.99153542e+01 1.00082535e+02
 9.96910086e+01 9.98786403e+01 9.98590930e+01 9.92594228e+01
 9.90188711e+01 9.86049285e+01 9.90509311e+01 9.90202600e+01
 9.90358977e+01 9.92213797e+01 9.91641850e+01 9.89039374e+01
 9.93579306e+01 9.92090406e+01 9.94113426e+01 9.93566685e+01
 9.92058231e+01 9.98498714e+01 1.00635410e+02 1.01439810e+02
 1.02221032e+02 1.02900971e+02 1.03171326e+02 1.03221016e+02
 1.03628257e+02 1.03891902e+02 1.04087181e+02 1.04234716e+02
 1.05085644e+02 1.05253506e+02 1.06211468e+02 1.06576557e+02
 1.06904769e+02 1.07273364e+02 1.08254428e+02 1.08838534e+02
 1.09627198e+02 1.10581714e+02 1.11868045e+02 1.13049079e+02
 1.13801161e+02 1.14441685e+02 1.15032886e+02 1.15896568e+02
 1.16746472e+02 1.17815519e+02 1.19136225e+02 1.20493162e+02
 1.21461228e+02 1.22548262e+02 1.23458381e+02 1.24997038e+02
 1.25824399e+02 1.27311551e+02 1.28899264e+02 1.30503752e+02
 1.31607502e+02 1.32842058e+02 1.34305964e+02 1.35469792e+02
```

```
1.36745472e+02 1.37860281e+02 1.39249914e+02 1.40235407e+02
1.41986259e+02 1.43023509e+02 1.44510927e+02 1.45793654e+02
1.47353485e+02 1.49300606e+02 1.50938291e+02 1.51969216e+02
1.53397645e+02 1.54907168e+02 1.56443346e+02 1.58124245e+02
1.59399868e+02 1.60524290e+02 1.61909080e+02 1.63853174e+02
1.65022593e+02 1.66902493e+02 1.68414882e+02 1.69832088e+02
1.71665185e+02 1.73064670e+02 1.74718614e+02 1.76035394e+02
1.77844266e+02 1.79425297e+02 1.81001226e+02 1.82189688e+02
1.83790795e+02 1.85253937e+02 1.86302974e+02 1.87253668e+02
1.88820572e+02 1.90529850e+02 1.91716958e+02 1.93015049e+02
1.94557318e+02 1.95675854e+02 1.97265152e+02 1.98453810e+02
1.99862363e+02 2.01333290e+02 2.02684640e+02 2.03353096e+02
2.04463748e+02 2.05316173e+02 2.05804665e+02 2.06625845e+02
2.07252654e+02 2.08440430e+02 2.08782752e+02 2.10020726e+02
2.10624785e+02 2.11199635e+02 2.11403344e+02 2.11745011e+02
2.12271604e+02 2.13295004e+02 2.13457409e+02 2.14434121e+02
2.15303610e+02 2.15726362e+02 2.15990272e+02 2.16128908e+02
2.16618039e+02 2.16494679e+02 2.16322904e+02 2.16533557e+02
2.16352941e+02 2.16325891e+02 2.16238035e+02 2.16152488e+02
2.15915419e+02 2.15530307e+02 2.15220012e+02 2.15384502e+02
2.15877004e+02 2.16358308e+02 2.16231535e+02 2.15832406e+02
2.15883383e+02 2.15760781e+02 2.16112767e+02 2.15516759e+02
2.15615446e+02 2.15196738e+02 2.14709188e+02 2.14621582e+02
2.14097132e+02 2.13972836e+02 2.13681127e+02 2.13278136e+02
2.12988496e+02 2.12414805e+02 2.12245073e+02 2.11768923e+02
2.11224642e+02 2.11135544e+02 2.11130735e+02 2.10235861e+02
2.09517432e+02 2.08905657e+02 2.08752041e+02 2.08761615e+02
2.08748300e+02 2.08665560e+02 2.07970001e+02 2.07966451e+02
2.07217844e+02 2.06324243e+02 2.06275196e+02 2.05509110e+02
2.05200851e+02 2.04263646e+02 2.04002240e+02 2.03896272e+02
2.03186422e+02 2.02589562e+02 2.01995275e+02 2.01783769e+02
2.01070520e+02 2.00301492e+02 1.99821008e+02 1.99207721e+02
1.99142083e+02 1.98488819e+02 1.98114902e+02 1.98196743e+02
1.97986413e+02 1.98026622e+02 1.98157814e+02 1.98118424e+02
1.98047486e+02 1.98265341e+02 1.98530773e+02 1.98275488e+02
1.97661810e+02 1.97293292e+02 1.96808223e+02 1.96612802e+02
1.96156307e+02 1.96198341e+02 1.96045322e+02 1.96431667e+02
1.96917708e+02 1.97010602e+02 1.96859646e+02 1.97054729e+02
1.97583366e+02 1.97651361e+02 1.97894422e+02 1.97887084e+02
1.98093838e+02 1.98277408e+02 1.98338441e+02 1.98687873e+02
1.98872920e+02 1.99151847e+02 1.99614027e+02 2.00333211e+02
2.00666718e+02 2.01622288e+02 2.02547397e+02 2.02690196e+02
2.03069171e+02 2.03671741e+02 2.03963750e+02 2.04530370e+02
2.05519157e+02 2.06478663e+02 2.07410064e+02 2.08459330e+02
2.09032379e+02 2.09771225e+02 2.10873246e+02 2.12206472e+02
2.13231903e+02 2.14023645e+02 2.14589707e+02 2.15862127e+02
2.16457796e+02 2.17443382e+02 2.18611343e+02 2.20194767e+02
2.20968808e+02 2.22081858e+02 2.22835139e+02 2.23927861e+02
2.25320606e+02 2.26389829e+02 2.27408596e+02 2.28351379e+02
2.29772848e+02 2.31328860e+02 2.32877123e+02 2.34716991e+02
2.35624521e+02 2.37189473e+02 2.38719641e+02 2.40249262e+02
2.42163088e+02 2.43140254e+02 2.44805319e+02 2.46594915e+02
2.48087009e+02 2.49408663e+02 2.51191617e+02 2.52284422e+02
2.53724485e+02 2.55243944e+02 2.56937900e+02 2.58028292e+02
2.59254078e+02 2.60659942e+02 2.62275351e+02 2.64150001e+02
2.65751114e+02 2.66860492e+02 2.68809655e+02 2.70641735e+02
2.72095512e+02 2.73958741e+02 2.75429898e+02 2.77270959e+02
2.79005683e+02 2.80864976e+02 2.82661066e+02 2.83557842e+02
2.84576942e+02 2.85793157e+02 2.87554316e+02 2.88644500e+02
```

```
2.89575481e+02 2.91211897e+02 2.92387260e+02 2.94002242e+02
2.94847020e+02 2.96169449e+02 2.97247137e+02 2.97936531e+02
2.99220779e+02 3.00030261e+02 3.00605704e+02 3.01174690e+02
3.01717522e+02 3.03079245e+02 3.04244908e+02 3.05540119e+02
3.06580825e+02 3.07448503e+02 3.08061967e+02 3.09338920e+02
3.09949799e+02 3.10638591e+02 3.11204218e+02 3.12340179e+02
3.12640171e+02 3.13062267e+02 3.13805406e+02 3.13900025e+02
3.14507274e+02 3.14953510e+02 3.15174719e+02 3.15319024e+02
3.15708204e+02 3.15919697e+02 3.16188835e+02 3.16308169e+02
3.17029973e+02 3.16884422e+02 3.16672209e+02 3.16676862e+02
3.16899765e+02 3.17161793e+02 3.17133271e+02 3.17678558e+02
3.17548698e+02 3.17761063e+02 3.17862819e+02 3.18113789e+02
3.18502230e+02 3.18779116e+02 3.18906254e+02 3.18943191e+02
3.18594142e+02 3.18034721e+02 3.17368388e+02 3.16860179e+02
3.16521947e+02 3.16253788e+02 3.16217855e+02 3.16349515e+02
3.15622508e+02 3.14868770e+02 3.14582158e+02 3.14436549e+02
3.14379873e+02 3.14397950e+02 3.14451600e+02 3.13910521e+02
3.13339799e+02 3.12527314e+02 3.12243666e+02 3.11924511e+02
3.11804286e+02 3.10913856e+02 3.10419518e+02 3.10005038e+02
3.09248568e+02 3.08418087e+02 3.08277668e+02 3.07336696e+02
3.06809291e+02 3.05929563e+02 3.05394507e+02 3.05386781e+02
3.04828200e+02 3.04709408e+02 3.03858140e+02 3.03169099e+02
3.02621835e+02 3.02080766e+02 3.01822373e+02 3.01249337e+02
3.01058277e+02 3.00806457e+02 3.00329312e+02 2.99900330e+02
2.99670529e+02 2.98913491e+02 2.98926881e+02 2.98790147e+02
2.98745976e+02 2.98532788e+02 2.97914297e+02 2.97612225e+02
2.97333052e+02 2.96992783e+02 2.96385309e+02 2.96509657e+02
2.96031656e+02 2.96075880e+02 2.96243271e+02 2.96369016e+02
2.96846405e+02 2.96495712e+02 2.96937100e+02 2.96569143e+02
2.96735935e+02 2.96801441e+02 2.97016607e+02 2.97102716e+02
2.97044593e+02 2.96845573e+02 2.97176341e+02 2.97862684e+02
2.98110566e+02 2.98239322e+02 2.98257063e+02 2.98310964e+02
2.98532360e+02 2.98663785e+02 2.98991615e+02 2.99151873e+02
2.99342811e+02 3.00355522e+02 3.00705023e+02 3.01482849e+02
3.02504994e+02 3.03240395e+02 3.03702781e+02 3.04241115e+02
3.05042224e+02 3.05967606e+02 3.06742825e+02 3.07978295e+02
3.09211071e+02 3.10647324e+02 3.11212103e+02 3.12669878e+02
3.13650232e+02 3.15141511e+02 3.16469720e+02 3.18118381e+02
3.19562210e+02 3.20633862e+02 3.21863894e+02 3.22990394e+02
3.24125819e+02 3.25176878e+02 3.26481866e+02 3.27990686e+02
3.29112354e+02 3.30497476e+02 3.31491163e+02 3.32542015e+02
3.33493648e+02 3.35382135e+02 3.36315771e+02 3.37435232e+02
3.38999155e+02 3.40040819e+02 3.41891298e+02 3.43586835e+02
3.45535512e+02 3.47030836e+02 3.48323355e+02 3.49868417e+02
3.51797263e+02 3.53317531e+02 3.54584738e+02 3.56461643e+02
3.57831589e+02 3.58828534e+02 3.60068334e+02 3.61374255e+02
3.63215320e+02 3.64649740e+02 3.66062911e+02 3.67359307e+02
3.69191041e+02 3.71076949e+02 3.72998616e+02 3.74293111e+02
3.76164086e+02 3.77846972e+02 3.79398968e+02 3.80504599e+02
3.81565385e+02 3.82558513e+02 3.84290287e+02 3.85374518e+02
3.86598126e+02 3.87842195e+02 3.89349335e+02 3.90900677e+02
3.91724296e+02 3.92812513e+02 3.94292791e+02 3.95723732e+02
3.96413931e+02 3.97132398e+02 3.97972423e+02 3.98556824e+02
3.99463240e+02 4.00024848e+02 4.01018845e+02 4.01662550e+02
4.02627108e+02 4.03083371e+02 4.04368272e+02 4.05579824e+02
4.06220311e+02 4.06680633e+02 4.07118413e+02 4.07758241e+02
4.07923567e+02 4.08657380e+02 4.09128474e+02 4.09537291e+02
4.10346656e+02 4.11024281e+02 4.11915824e+02 4.12173652e+02
4.12945861e+02 4.13559394e+02 4.13729437e+02 4.14132960e+02
```

```
4.14172840e+02 4.14768248e+02 4.14571659e+02 4.14445355e+02
4.14260264e+02 4.14266142e+02 4.14585596e+02 4.14658513e+02
4.14664435e+02 4.14155940e+02 4.14045515e+02 4.13546985e+02
4.13167453e+02 4.13486940e+02 4.13064914e+02 4.13261940e+02
4.13380286e+02 4.12832326e+02 4.12709069e+02 4.12074620e+02
4.12031995e+02 4.11879302e+02 4.11882224e+02 4.11534528e+02
4.11605063e+02 4.10793670e+02 4.10210252e+02 4.10034298e+02
4.09547580e+02 4.08802777e+02 4.07980211e+02 4.07856669e+02
4.07378756e+02 4.07300746e+02 4.07065452e+02 4.06499261e+02
4.05890485e+02 4.05418435e+02 4.05315380e+02 4.05057097e+02
4.04064223e+02 4.03758874e+02 4.03678382e+02 4.03389331e+02
4.02568310e+02 4.02056266e+02 4.01204467e+02 4.00575774e+02
4.00530604e+02 4.00477993e+02 3.99792246e+02 3.99171584e+02
3.98820740e+02 3.98843057e+02 3.98061081e+02 3.97400243e+02
3.97368973e+02 3.96969859e+02 3.96992513e+02 3.96317289e+02
3.96005630e+02 3.95504819e+02 3.95012344e+02 3.94669066e+02
3.94331631e+02 3.93938866e+02 3.94124869e+02 3.94149506e+02
3.94192506e+02 3.93821107e+02 3.93761393e+02 3.93927665e+02
3.94135470e+02 3.93917917e+02 3.94152963e+02 3.94379816e+02
3.94039569e+02 3.94451524e+02 3.94467066e+02 3.94556328e+02
3.94636573e+02 3.94865681e+02 3.95177833e+02 3.95397022e+02
3.96015120e+02 3.96424712e+02 3.96638759e+02 3.96899016e+02
3.97584521e+02 3.97838605e+02 3.97987269e+02 3.98015242e+02
3.98891269e+02 3.99841226e+02 4.00024136e+02 4.00452718e+02
4.00753424e+02 4.01025757e+02 4.02113005e+02 4.02388025e+02
4.03617201e+02 4.03992643e+02 4.05165130e+02 4.06070374e+02
4.06806416e+02 4.07545907e+02 4.08729050e+02 4.09497614e+02
4.10636687e+02 4.11663054e+02 4.12683847e+02 4.14147092e+02
4.15289522e+02 4.16729291e+02 4.18092289e+02 4.18792158e+02
4.20249815e+02 4.21173792e+02 4.22805910e+02 4.23598945e+02
4.24702531e+02 4.25592419e+02 4.26877025e+02 4.28524128e+02
4.29666643e+02 4.30678320e+02 4.31884894e+02 4.33589780e+02
4.34634991e+02 4.35780108e+02 4.36778737e+02 4.38425328e+02
4.39771617e+02 4.41042751e+02 4.42729961e+02 4.44041855e+02
4.45751913e+02 4.47554801e+02 4.48764579e+02 4.50733959e+02
4.51894344e+02 4.53184692e+02 4.54364487e+02 4.55709499e+02
4.57187587e+02 4.58705325e+02 4.60551045e+02 4.62428182e+02
4.63630573e+02 4.65229383e+02 4.66309463e+02 4.67728726e+02
4.69002116e+02 4.70259498e+02 4.71671858e+02 4.73319441e+02
4.74293450e+02 4.76063630e+02 4.77674751e+02 4.78711992e+02
4.80495481e+02 4.81828874e+02 4.83637699e+02 4.84625114e+02
4.85930535e+02 4.86970894e+02 4.88004037e+02 4.89302209e+02
4.90908475e+02 4.92375051e+02 4.93736416e+02 4.94908497e+02
4.96418581e+02 4.97473025e+02 4.98882684e+02 5.00225707e+02
5.00921297e+02 5.02146512e+02 5.02584463e+02 5.03304440e+02
5.03902132e+02 5.04747733e+02 5.05793603e+02 5.06170271e+02
5.06933883e+02 5.08090438e+02 5.08506466e+02 5.09340042e+02
5.10058255e+02 5.10162427e+02 5.10701044e+02 5.11441225e+02
5.11485200e+02 5.12333311e+02 5.12790602e+02 5.12727155e+02
5.12826235e+02 5.13623476e+02 5.14018415e+02 5.13907744e+02
5.13946599e+02 5.14124311e+02 5.13836244e+02 5.13909122e+02
5.14030456e+02 5.13876986e+02 5.14039846e+02 5.14339095e+02
5.14093176e+02 5.14204634e+02 5.14308430e+02 5.14694887e+02
5.14885378e+02 5.15120816e+02 5.14973812e+02 5.14497849e+02
5.14809117e+02 5.14231366e+02 5.13973854e+02 5.13291836e+02
5.13465174e+02 5.13639944e+02 5.13550316e+02 5.13073228e+02
5.12483283e+02 5.11887844e+02 5.11138831e+02 5.10773634e+02
5.10010624e+02 5.09409600e+02 5.09106530e+02 5.08893193e+02
5.07966409e+02 5.07436231e+02 5.07119773e+02 5.06997102e+02
```

```
 5.06490140e+02 5.06471657e+02 5.05745175e+02 5.04763122e+02
 5.04248848e+02 5.03505455e+02 5.03329173e+02 5.02562439e+02
 5.01875041e+02 5.01670707e+02 5.01393735e+02 5.00964098e+02
 5.00686759e+02 5.00129479e+02 4.99166206e+02 4.98217268e+02
 4.97777433e+02 4.96919843e+02 4.96041142e+02 4.96088904e+02
 4.96043080e+02 4.95304100e+02 4.94428458e+02 4.94509384e+02
 4.93943381e+02 4.93302198e+02 4.93184689e+02 4.92503438e+02
 4.91997574e+02 4.92222558e+02 4.92133052e+02 4.91946623e+02
 4.91659995e+02 4.91773184e+02 4.91276717e+02 4.91631148e+02
 4.91904486e+02 4.91960059e+02 4.91467139e+02 4.91658953e+02
 4.92090484e+02 4.92161070e+02 4.92089516e+02 4.91812645e+02
 4.92199422e+02 4.92745706e+02 4.92536940e+02 4.93016934e+02
 4.92785304e+02 4.93534216e+02 4.93961893e+02]
CS: [ 2.48753144e-01  8.56321237e-01  1.51115369e+00  2.17751391e+00
  2.52592877e+00  3.38635911e+00  3.41329771e+00  3.82301717e+00
  4.10482611e+00  4.33283930e+00  4.34647064e+00  4.44720761e+00
  4.63188344e+00  4.66213505e+00  5.39322164e+00  5.41692892e+00
  5.88869321e+00  5.64983126e+00  6.16332092e+00  6.06187985e+00
  5.92699806e+00  5.93929037e+00  5.58554176e+00  5.05651802e+00
  5.19516508e+00  5.18070917e+00  5.30839296e+00  5.83065554e+00
  5.77375617e+00  5.51747286e+00  5.89927038e+00  5.97023966e+00
  6.21001306e+00  6.81796860e+00  7.25794121e+00  7.20138427e+00
  7.58480458e+00  7.45146790e+00  7.30316646e+00  7.60756064e+00
  8.04238618e+00  7.90519481e+00  7.68868836e+00  8.21844632e+00
  8.08217075e+00  8.31145559e+00  7.78883633e+00  8.58371430e+00
  7.84883459e+00  7.76737216e+00  6.90465680e+00  7.61060858e+00
  7.55226625e+00  7.95640929e+00  7.74064116e+00  7.67367651e+00
  7.50587775e+00  7.12999151e+00  7.25596756e+00  7.68334558e+00
  8.55306826e+00  8.67003635e+00  8.18206747e+00  7.76916140e+00
  7.57730301e+00  6.71336119e+00  6.31402036e+00  6.28760981e+00
  6.05420157e+00  5.26498632e+00  5.80506482e+00  5.42180123e+00
  5.62258614e+00  6.25645405e+00  6.84423471e+00  5.98046573e+00
  5.81562729e+00  5.98255263e+00  5.51901285e+00  4.93642969e+00
  5.71948323e+00  6.04629130e+00  5.83496064e+00  5.71925945e+00
  6.08033528e+00  5.32427152e+00  4.97222254e+00  4.50775458e+00
  3.96312075e+00  4.03435536e+00  4.31698765e+00  4.26243721e+00
  4.13155009e+00  4.04973225e+00  3.01203242e+00  2.55431572e+00
  2.22199492e+00  2.53634221e+00  2.50438484e+00  2.46998880e+00
  1.90381644e+00  1.50252620e+00  1.32867665e+00  1.24772282e+00
  1.44795436e+00  2.16351246e+00  2.93063122e+00  2.31278271e+00
  2.36758250e+00  1.94066779e+00  9.12064968e-01  5.28748068e-01
  4.81886923e-01  5.38146111e-02  2.32021162e-01  8.97445113e-01
  6.25032549e-01  1.03674919e+00  7.88148427e-01  8.61574061e-01
  8.23667971e-02  2.75023231e-01 -3.85217221e-02 -2.22041150e-01
 -1.95181122e-01  2.30717212e-01 -5.84194376e-01 -1.17105776e+00
 -9.77566096e-01 -2.00488187e+00 -2.63017782e+00 -2.55754594e+00
 -2.49090479e+00 -2.63575887e+00 -2.46994455e+00 -2.17261980e+00
 -1.81119615e+00 -2.14126097e+00 -2.15649175e+00 -2.36662644e+00
 -2.29660109e+00 -2.81986363e+00 -2.42280271e+00 -2.80304226e+00
 -2.83166908e+00 -3.09710431e+00 -3.16937229e+00 -2.82134046e+00
 -2.26140501e+00 -1.67480846e+00 -1.27145666e+00 -1.55103882e+00
 -1.94174306e+00 -2.12012748e+00 -1.89946286e+00 -2.49872342e+00
 -2.32409776e+00 -2.75388632e+00 -3.29495396e+00 -3.29554000e+00
 -2.88113916e+00 -2.29540855e+00 -2.06783211e+00 -1.44702304e+00
 -2.25713045e+00 -2.38136941e+00 -2.12445309e+00 -1.46064350e+00
 -1.63546801e+00 -6.60106644e-01 -8.32043083e-01 -7.74067733e-01
 -6.18966489e-01 -6.37059979e-02 -2.96716269e-01  4.47780285e-01
 -1.02910347e-01 -8.61431188e-02 -1.54781934e-01 -6.55931440e-01
 -1.06366436e+00 -1.08608006e+00 -8.27709150e-01 -8.38832961e-01
```

```
-2.23525074e-01  2.02508677e-01  5.59256509e-01 -1.46993705e-01
 3.37670584e-01  1.73142884e-01  1.89962856e-01  1.92483464e-01
 3.01885681e-01  4.91940776e-01  1.42636896e+00  2.02846716e+00
 2.86436351e+00  3.69514793e+00  3.32145428e+00  2.58560674e+00
 2.18844772e+00  1.67087036e+00  1.18620949e+00  1.06339035e+00
 1.86462732e+00  1.62524807e+00  2.31956523e+00  2.48937697e+00
 2.67005344e+00  2.18771998e+00  3.00092286e+00  2.62706593e+00
 3.05064006e+00  3.67694477e+00  4.59468110e+00  4.79465106e+00
 4.96262751e+00  4.81448770e+00  4.45117214e+00  4.02852368e+00
 3.69739232e+00  4.01435836e+00  4.69453948e+00  5.46027567e+00
 5.56465961e+00  5.80178995e+00  5.64286192e+00  5.86081317e+00
 5.33123718e+00  5.85032348e+00  6.35100213e+00  7.04537068e+00
 6.61046459e+00  7.01765844e+00  6.99441250e+00  6.57052853e+00
 6.24172025e+00  6.25277866e+00  6.40785587e+00  5.92944262e+00
 6.51646660e+00  6.27803688e+00  6.65064628e+00  6.54374078e+00
 7.11807790e+00  7.31434719e+00  7.91478200e+00  7.45828809e+00
 7.60399040e+00  7.55368361e+00  7.14273942e+00  7.18595355e+00
 7.43065197e+00  7.12664533e+00  7.00191230e+00  7.40982887e+00
 6.89834794e+00  7.50262583e+00  7.89059241e+00  7.92300714e+00
 7.81201100e+00  8.04207759e+00  7.81612100e+00  7.62051183e+00
 8.01217817e+00  7.76011179e+00  7.93655579e+00  7.47107410e+00
 7.75540072e+00  7.40967091e+00  6.87767663e+00  6.25244173e+00
 6.63088402e+00  6.73905512e+00  6.46302095e+00  6.71207463e+00
 7.30365054e+00  6.85528114e+00  6.73530194e+00  6.73685270e+00
 6.84731412e+00  6.77597147e+00  7.00878677e+00  6.08794418e+00
 6.00993808e+00  5.45381046e+00  4.47137561e+00  3.94120514e+00
 3.89955788e+00  3.97668199e+00  3.46657886e+00  4.21606044e+00
 3.99893903e+00  3.94698054e+00  2.96291413e+00  2.96225930e+00
 2.25087821e+00  2.67021920e+00  2.25777452e+00  3.03077700e+00
 3.55859874e+00  3.45475764e+00  2.69526827e+00  2.67149864e+00
 2.18391782e+00  1.19106942e+00  5.96542527e-01  5.43284646e-01
 2.24033017e-01 -2.92148574e-01 -2.56644779e-01 -1.70416255e-01
-6.18138058e-01 -8.22633913e-01 -1.10587879e+00 -8.53532950e-01
-2.75483685e-01  4.42889222e-01  7.01228190e-01  6.12394233e-01
 4.98880978e-01 -1.16223268e-01 -2.45541066e-01 -7.14776024e-01
-2.16959740e-01 -6.86644529e-01 -1.05159283e+00 -1.49118512e+00
-1.41962751e+00 -1.64261074e+00 -1.51561094e+00 -1.43105251e+00
-1.63308613e+00 -1.68232658e+00 -1.72776266e+00 -1.91220384e+00
-2.05349374e+00 -1.85295134e+00 -1.28407041e+00 -2.00921189e+00
-2.25149176e+00 -2.31898458e+00 -2.38350355e+00 -2.36911931e+00
-1.48756140e+00 -8.51871873e-01 -9.35655993e-01 -7.85590033e-01
-1.54377139e+00 -2.42405625e+00 -2.39036364e+00 -2.46089151e+00
-2.76559925e+00 -2.95419817e+00 -2.32200337e+00 -2.37892410e+00
-2.32268815e+00 -2.61128956e+00 -2.26837093e+00 -2.21847076e+00
-2.82575190e+00 -2.88492922e+00 -2.76855419e+00 -2.78755343e+00
-2.64168593e+00 -2.58170135e+00 -2.18659091e+00 -1.62426450e+00
-1.22130851e+00 -1.11546174e+00 -3.31004351e-01  3.52272061e-03
-1.49257035e-01  2.78927690e-01  5.04151021e-01  1.17673228e-01
-4.56613950e-01 -7.54193601e-01 -1.45711738e+00 -1.91797022e+00
-2.11918105e+00 -1.46346929e+00 -1.24797020e+00 -3.76556444e-01
 3.04905233e-01  8.54295559e-01  6.61305411e-01  1.00940667e+00
 1.15169881e+00  7.33653239e-01  8.83820120e-01  1.02743750e+00
 1.03910924e+00  6.94042926e-01  6.87079651e-01  7.93450287e-01
 9.85836218e-01  1.05800851e+00  1.33661813e+00  1.99477025e+00
 1.97884580e+00  2.74936773e+00  3.39555024e+00  3.07616893e+00
 2.73596042e+00  3.00502294e+00  2.34146261e+00  1.98297278e+00
 2.82896158e+00  3.40949141e+00  3.73832286e+00  4.49557998e+00
 4.50200899e+00  4.25206758e+00  4.39458376e+00  4.79640760e+00
 4.77257301e+00  4.99126584e+00  4.81848188e+00  4.98888032e+00
```

```
 4.25132441e+00  4.21147881e+00  4.58769894e+00  5.60506022e+00
 5.10668166e+00  5.62406141e+00  5.39175711e+00  5.31651796e+00
 5.12583894e+00  5.42102025e+00  5.32673809e+00  5.51624007e+00
 5.84498618e+00  6.00825391e+00  6.48729477e+00  7.30839538e+00
 7.27314183e+00  7.41662494e+00  7.39078163e+00  7.37213819e+00
 7.44609748e+00  7.51573258e+00  7.61584689e+00  7.87527358e+00
 7.83774765e+00  7.24557402e+00  8.05136353e+00  7.47910282e+00
 7.12956978e+00  7.15693427e+00  7.52923747e+00  6.83667489e+00
 6.96965623e+00  6.93545740e+00  7.03140783e+00  7.21210055e+00
 7.72282167e+00  7.60641368e+00  8.14971344e+00  8.36638349e+00
 7.94551164e+00  8.20762698e+00  8.56940602e+00  8.46130342e+00
 8.36394842e+00  8.76946408e+00  8.70232482e+00  8.12794338e+00
 7.30598262e+00  6.78747345e+00  6.68934021e+00  5.98343465e+00
 6.01763969e+00  6.63495550e+00  6.59410329e+00  6.44792597e+00
 6.20251961e+00  6.59396810e+00  6.03524011e+00  5.54927089e+00
 5.21853691e+00  5.18324147e+00  4.43625441e+00  3.92755238e+00
 3.78099077e+00  3.85846583e+00  4.21464691e+00  4.93441484e+00
 5.40613599e+00  5.73098163e+00  4.98272153e+00  5.09401168e+00
 4.40968038e+00  4.05776570e+00  3.75571434e+00  4.27821238e+00
 3.30125064e+00  3.11246801e+00  3.16681500e+00  2.69580689e+00
 2.16709487e+00  2.31333899e+00  2.11245214e+00  1.51361792e+00
 1.80817905e+00  1.41242279e+00  1.23532563e+00  1.13345031e+00
 1.71094915e+00  1.17621796e+00  7.52511808e-01  4.88027040e-01
 5.91595248e-01  1.31820022e-01  2.48849152e-01  1.00634969e+00
 8.71835201e-01  8.61298105e-01  7.01025424e-01  9.80518453e-01
 8.23671370e-01  1.23041865e+00  1.14519138e+00  1.08037210e+00
 4.80352675e-01 -4.67508939e-01 -1.41072817e+00 -2.04607552e+00
-2.42124389e+00 -2.34035369e+00 -1.81686593e+00 -1.01887296e+00
-1.23767104e+00 -1.65317716e+00 -1.67163038e+00 -1.78130539e+00
-1.96964241e+00 -1.22455751e+00 -4.17169868e-01 -6.71636988e-01
-1.09674987e+00 -1.85255821e+00 -2.15428438e+00 -2.52708894e+00
-2.10623452e+00 -2.42594354e+00 -2.10779632e+00 -2.23862763e+00
-2.67594252e+00 -3.38619902e+00 -2.63618783e+00 -3.08282168e+00
-3.19574747e+00 -3.31900574e+00 -3.02358053e+00 -2.89088738e+00
-2.50849664e+00 -2.09988266e+00 -2.07142229e+00 -2.22540796e+00
-2.76494572e+00 -2.74743345e+00 -2.88703462e+00 -2.60880352e+00
-2.11082209e+00 -1.81537794e+00 -1.75145485e+00 -1.92204374e+00
-1.57880747e+00 -2.14478567e+00 -1.87957630e+00 -1.53916462e+00
-1.15435350e+00 -1.13774134e+00 -9.99194471e-01 -1.31465538e+00
-1.45709505e+00 -1.75319315e+00 -2.14747875e+00 -1.40464004e+00
-1.58056967e+00 -1.25717139e+00 -7.49511813e-01 -1.62932414e-02
 3.36748148e-01  4.64056597e-01  8.61219709e-01  3.25871352e-01
 3.66918747e-01 -4.49641330e-02  5.20895078e-01  1.65615981e-01
 4.75450665e-01  1.09638019e-01  3.74900820e-01  8.46076352e-01
 1.00785030e+00  1.19472878e+00  1.41149061e+00  1.13462227e+00
 6.69676321e-01  5.53218419e-01  7.52293132e-01  8.94809424e-01
 1.03184736e+00  1.82316156e+00  2.04123831e+00  2.49123374e+00
 3.35312140e+00  3.89758397e+00  3.34725920e+00  3.53609168e+00
 3.55937480e+00  3.46261134e+00  3.50243026e+00  4.27551448e+00
 4.96995575e+00  5.60510021e+00  5.24449757e+00  5.92705266e+00
 5.67193660e+00  5.93044083e+00  5.82239637e+00  6.90627765e+00
 6.89233254e+00  6.98362990e+00  6.72238215e+00  6.52067361e+00
 6.00743807e+00  5.61466802e+00  5.84800391e+00  6.12679262e+00
 6.12196060e+00  6.37165736e+00  6.31428482e+00  6.06014934e+00
 5.50296223e+00  6.26978010e+00  5.81829510e+00  5.94406886e+00
 6.45714032e+00  6.54717020e+00  6.50916379e+00  7.27106387e+00
 8.10028036e+00  8.03168098e+00  8.28253608e+00  7.97711895e+00
 8.21042748e+00  7.78201825e+00  7.55390140e+00  8.13828836e+00
 7.96317130e+00  7.03127117e+00  6.75080290e+00  6.78951771e+00
```

```
 6.75367708e+00  6.81815154e+00  7.23437667e+00  7.29097290e+00
 7.81678569e+00  7.86162852e+00  8.34887540e+00  8.23020085e+00
 8.80477939e+00  8.65593085e+00  8.32201913e+00  7.50598382e+00
 7.27227311e+00  6.39442701e+00  6.44331452e+00  5.97555037e+00
 6.09352635e+00  6.27681007e+00  6.79082189e+00  6.61039003e+00
 6.34977816e+00  6.21438690e+00  6.45059688e+00  6.37439701e+00
 5.51325428e+00  5.40810120e+00  5.15991046e+00  4.26403207e+00
 3.73950839e+00  3.61091758e+00  3.88644774e+00  3.69012708e+00
 4.07028422e+00  3.62013110e+00  4.34342376e+00  4.56097918e+00
 4.55776118e+00  4.05352540e+00  4.03504136e+00  3.38996850e+00
 2.34374222e+00  2.43706882e+00  2.44784041e+00  2.41887821e+00
 2.58841519e+00  3.10071385e+00  3.25844373e+00  3.04517823e+00
 3.40857043e+00  3.21273818e+00  2.70515681e+00  2.21713571e+00
 1.99918785e+00  1.82238713e+00  1.01226474e+00  7.15917840e-01
 1.27304385e-01  9.33024261e-02 -1.82652716e-01  8.68539100e-02
 2.19080118e-01 -1.04324072e-01 -2.20627253e-01 -1.03861052e+00
-1.49105947e+00 -1.17749524e+00 -1.09102557e+00 -7.83574730e-01
-1.66698861e-01 -3.35127405e-01 -7.77870961e-01 -9.90294373e-01
-1.22994523e+00 -1.50098423e+00 -9.50102120e-01 -1.17454147e+00
-4.69557115e-01 -1.23832511e+00 -1.66905049e+00 -1.84792598e+00
-1.98694780e+00 -2.80228623e+00 -2.81345889e+00 -2.35358244e+00
-2.65554205e+00 -2.24683422e+00 -1.73732498e+00 -1.48094979e+00
-1.96618437e+00 -1.96032083e+00 -1.98536556e+00 -2.00835492e+00
-2.43503813e+00 -2.13161060e+00 -1.74005320e+00 -1.92604925e+00
-2.48878681e+00 -2.00795689e+00 -2.55440686e+00 -3.10260733e+00
-2.85872674e+00 -2.09031725e+00 -2.26401972e+00 -2.03288368e+00
-1.75503426e+00 -1.68754753e+00 -2.41691122e+00 -2.39200290e+00
-1.80261048e+00 -1.85088113e+00 -1.85054324e+00 -1.74379222e+00
-1.39461357e+00 -1.86415429e+00 -1.95751443e+00 -2.32344689e+00
-1.98565856e+00 -2.06676341e+00 -1.37995032e+00 -8.62838485e-01
-4.76560551e-01 -5.10523173e-01 -1.77473549e-01 -1.97203312e-01
-1.40355999e-02 -2.74588402e-01  3.31855975e-01  6.18423021e-01
 1.11903453e-01  3.16053306e-01  5.49148402e-01  4.03364998e-01
 2.56757143e-01  8.26112413e-01  7.26309697e-01  9.29955978e-01
 1.45879208e+00  1.78813896e+00  1.77307752e+00  1.72118256e+00
 2.18749948e+00  1.82348445e+00  1.56255721e+00  1.37648301e+00
 1.99225256e+00  2.25670439e+00  2.18553147e+00  2.46544924e+00
 2.73818212e+00  2.13448820e+00  2.27177905e+00  2.36388885e+00
 3.16448235e+00  3.23921901e+00  4.13937372e+00  3.95736969e+00
 4.41839123e+00  3.92870595e+00  4.73640741e+00  4.33248356e+00
 4.56631213e+00  4.85663759e+00  5.13793976e+00  5.41804187e+00
 5.79190839e+00  6.09260446e+00  6.42923448e+00  6.10831115e+00
 6.10272276e+00  5.88426948e+00  6.07661910e+00  5.50665684e+00
 5.91037300e+00  5.34260380e+00  5.70323288e+00  5.71821762e+00
 6.06769764e+00  5.97578929e+00  6.29247505e+00  6.71275560e+00
 6.11086360e+00  6.11346539e+00  6.10041692e+00  6.54043401e+00
 6.18183711e+00  6.40775976e+00  6.94985224e+00  7.26311809e+00
 7.32658580e+00  7.78318378e+00  7.72182766e+00  8.00399877e+00
 7.85248914e+00  7.43277809e+00  6.80968564e+00  6.94492031e+00
 6.45362727e+00  6.81098042e+00  7.36635370e+00  8.06369466e+00
 7.92107355e+00  8.04179673e+00  7.60413812e+00  7.17768092e+00
 6.57393472e+00  6.62892513e+00  6.44247506e+00  7.00997810e+00
 6.56472427e+00  7.06151408e+00  7.41525302e+00  7.04013370e+00
 7.17604028e+00  7.53542337e+00  7.57406908e+00  6.95036311e+00
 7.21854313e+00  6.47541341e+00  6.17516280e+00  5.66450939e+00
 6.28336107e+00  6.44451535e+00  6.76552163e+00  6.90446028e+00
 7.11637187e+00  6.56455049e+00  6.50763374e+00  6.48929141e+00
 6.01279959e+00  5.72793180e+00  5.11143747e+00  4.42175592e+00
 3.67642486e+00  3.82643672e+00  3.64709091e+00  3.58580793e+00
```

```
  3.62944259e+00  4.18830586e+00  3.75873285e+00  3.54643830e+00
  3.88798433e+00  3.22854425e+00  2.61060570e+00  2.93475899e+00
  2.14515881e+00  2.27505603e+00  2.62817492e+00  2.02611094e+00
  1.38501017e+00  2.13827597e+00  1.68510366e+00  1.11714209e+00
  1.21944397e+00  1.29807611e+00  2.12767955e-01 -1.09292870e-01
  1.22711886e-01 -6.96127445e-02 -8.44655556e-02  5.02850989e-01
  1.84053946e-01  1.74178348e-01  4.31443811e-01  6.55041026e-01
  5.46282321e-01  1.02763999e+00  7.69177421e-01  1.89418695e-01
  1.14229959e-01 -6.54011977e-01 -1.14696150e+00 -1.68197539e+00
 -1.03267450e+00 -1.16917276e+00 -6.81049668e-01 -9.00626090e-01
 -8.08553644e-01 -1.57732996e+00 -2.50111254e+00 -2.77668218e+00
 -3.06260423e+00 -3.07368266e+00 -2.78131397e+00 -2.24563848e+00
 -2.80722472e+00 -2.57439264e+00 -2.28982725e+00 -2.10942802e+00
 -2.40305333e+00 -1.49475199e+00 -1.69105644e+00 -2.35665113e+00
 -2.74825421e+00 -2.98468451e+00 -3.14248437e+00 -3.18273610e+00
 -2.88808036e+00 -2.57814103e+00 -2.11171971e+00 -2.36507449e+00
 -1.87567979e+00 -1.74556217e+00 -2.50450075e+00 -3.17646786e+00
 -3.18666549e+00 -3.76691609e+00 -4.08833719e+00 -3.07730207e+00
 -2.17418792e+00 -2.47333249e+00 -2.49138516e+00 -1.53175839e+00
 -2.14552303e+00 -2.74088159e+00 -2.11941118e+00 -1.92501999e+00
 -2.51180940e+00 -1.72082317e+00 -1.16914605e+00 -1.23806615e+00
 -8.43443368e-01 -2.24390634e-01 -9.45840757e-01 -5.01904156e-01
 -4.21374106e-02  3.00064685e-01 -3.06045009e-01  3.82235738e-01
  4.59335656e-01  2.56584468e-01  1.29456394e-01  3.45506453e-01
  5.40468771e-01  6.55222035e-01  3.75870217e-01  9.27418135e-01
  9.72659280e-01  1.33479441e+00  1.21618718e+00  1.82095702e+00
  1.51158018e+00  2.57839293e+00  2.03259637e+00  2.40165551e+00]
[2.48753144e-01 8.56321237e-01 1.51115369e+00 2.17751391e+00
 2.52592877e+00 3.63511225e+00 4.26961894e+00 5.33417086e+00
 6.28234002e+00 6.85876808e+00 7.98158289e+00 8.71682655e+00
 9.96605430e+00 1.09444751e+01 1.22519897e+01 1.33985118e+01
 1.46055198e+01 1.56158856e+01 1.71077960e+01 1.83138696e+01
 1.93255099e+01 2.05448101e+01 2.12014273e+01 2.21643140e+01
 2.35090346e+01 2.45062190e+01 2.58532031e+01 2.70320829e+01
 2.79380702e+01 2.90265075e+01 3.04054894e+01 3.18234427e+01
 3.32420959e+01 3.47560388e+01 3.62844487e+01 3.76068737e+01
 3.94082473e+01 4.06935638e+01 4.20592052e+01 4.38920094e+01
 4.56492599e+01 4.73134421e+01 4.83822522e+01 5.02776516e+01
 5.19741801e+01 5.39607155e+01 5.51022785e+01 5.69659665e+01
 5.81264861e+01 5.97415523e+01 6.08653723e+01 6.27128870e+01
 6.45182327e+01 6.60828954e+01 6.74821934e+01 6.85390488e+01
 7.02187648e+01 7.16482242e+01 7.33388630e+01 7.51655390e+01
 7.70921170e+01 7.88888011e+01 7.98302917e+01 8.11080244e+01
 8.27428420e+01 8.38054782e+01 8.52028215e+01 8.61179015e+01
 8.71622260e+01 8.80078283e+01 8.96105430e+01 9.06246227e+01
 9.17404877e+01 9.34186800e+01 9.48520631e+01 9.55910088e+01
 9.64402500e+01 9.77230403e+01 9.89376929e+01 9.97884927e+01
 1.01310492e+02 1.02486541e+02 1.03558001e+02 1.04656952e+02
 1.05868828e+02 1.06634764e+02 1.07458764e+02 1.08065756e+02
 1.08620073e+02 1.09903183e+02 1.10951751e+02 1.11721201e+02
 1.12197306e+02 1.12669805e+02 1.12915216e+02 1.13506067e+02
 1.13943196e+02 1.14733648e+02 1.15174190e+02 1.15385205e+02
 1.15409883e+02 1.15445722e+02 1.16062324e+02 1.16421913e+02
 1.16833159e+02 1.17573396e+02 1.18376353e+02 1.18375107e+02
 1.18789495e+02 1.18773827e+02 1.18485461e+02 1.18905101e+02
 1.18856994e+02 1.18843310e+02 1.19005848e+02 1.19382906e+02
 1.19530134e+02 1.19893743e+02 1.19631459e+02 1.19867422e+02
 1.19465273e+02 1.19805157e+02 1.19855222e+02 1.19409417e+02
 1.19672241e+02 1.19695990e+02 1.19220963e+02 1.18684164e+02
```

```
1.18431851e+02 1.17667359e+02 1.17065812e+02 1.16663417e+02
1.16193259e+02 1.15796092e+02 1.15197414e+02 1.14893192e+02
1.14852221e+02 1.14051998e+02 1.13639601e+02 1.12830788e+02
1.12596591e+02 1.12032357e+02 1.11629195e+02 1.10836558e+02
1.09999119e+02 1.09499487e+02 1.08862985e+02 1.08807855e+02
1.08575153e+02 1.08324310e+02 1.08228030e+02 1.07311946e+02
1.06866112e+02 1.06455026e+02 1.06424848e+02 1.05729307e+02
1.04987848e+02 1.04112225e+02 1.03160072e+02 1.03129308e+02
1.02848168e+02 1.02692440e+02 1.02044393e+02 1.01713049e+02
1.00872177e+02 1.00466798e+02 1.00567987e+02 1.00583750e+02
1.00077581e+02 1.00212070e+02 9.96347551e+01 9.97939189e+01
9.99647834e+01 1.00013875e+02 9.99153542e+01 1.00082535e+02
9.96910086e+01 9.98786403e+01 9.98590930e+01 9.92594228e+01
9.90188711e+01 9.86049285e+01 9.90509311e+01 9.90202600e+01
9.90358977e+01 9.92213797e+01 9.91641850e+01 9.89039374e+01
9.93579306e+01 9.92090406e+01 9.94113426e+01 9.93566685e+01
9.92058231e+01 9.98498714e+01 1.00635410e+02 1.01439810e+02
1.02221032e+02 1.02900971e+02 1.03171326e+02 1.03221016e+02
1.03628257e+02 1.03891902e+02 1.04087181e+02 1.04234716e+02
1.05085644e+02 1.05253506e+02 1.06211468e+02 1.06576557e+02
1.06904769e+02 1.07273364e+02 1.08254428e+02 1.08838534e+02
1.09627198e+02 1.10581714e+02 1.11868045e+02 1.13049079e+02
1.13801161e+02 1.14441685e+02 1.15032886e+02 1.15896568e+02
1.16746472e+02 1.17815519e+02 1.19136225e+02 1.20493162e+02
1.21461228e+02 1.22548262e+02 1.23458381e+02 1.24997038e+02
1.25824399e+02 1.27311551e+02 1.28899264e+02 1.30503752e+02
1.31607502e+02 1.32842058e+02 1.34305964e+02 1.35469792e+02
1.36745472e+02 1.37860281e+02 1.39249914e+02 1.40235407e+02
1.41986259e+02 1.43023509e+02 1.44510927e+02 1.45793654e+02
1.47353485e+02 1.49300606e+02 1.50938291e+02 1.51969216e+02
1.53397645e+02 1.54907168e+02 1.56443346e+02 1.58124245e+02
1.59399868e+02 1.60524290e+02 1.61909080e+02 1.63853174e+02
1.65022593e+02 1.66902493e+02 1.68414882e+02 1.69832088e+02
1.71665185e+02 1.73064670e+02 1.74718614e+02 1.76035394e+02
1.77844266e+02 1.79425297e+02 1.81001226e+02 1.82189688e+02
1.83790795e+02 1.85253937e+02 1.86302974e+02 1.87253668e+02
1.88820572e+02 1.90529850e+02 1.91716958e+02 1.93015049e+02
1.94557318e+02 1.95675854e+02 1.97265152e+02 1.98453810e+02
1.99862363e+02 2.01333290e+02 2.02684640e+02 2.03353096e+02
2.04463748e+02 2.05316173e+02 2.05804665e+02 2.06625845e+02
2.07252654e+02 2.08440430e+02 2.08782752e+02 2.10020726e+02
2.10624785e+02 2.11199635e+02 2.11403344e+02 2.11745011e+02
2.12271604e+02 2.13295004e+02 2.13457409e+02 2.14434121e+02
2.15303610e+02 2.15726362e+02 2.15990272e+02 2.16128908e+02
2.16618039e+02 2.16494679e+02 2.16322904e+02 2.16533557e+02
2.16352941e+02 2.16325891e+02 2.16238035e+02 2.16152488e+02
2.15915419e+02 2.15530307e+02 2.15220012e+02 2.15384502e+02
2.15877004e+02 2.16358308e+02 2.16231535e+02 2.15832406e+02
2.15883383e+02 2.15760781e+02 2.16112767e+02 2.15516759e+02
2.15615446e+02 2.15196738e+02 2.14709188e+02 2.14621582e+02
2.14097132e+02 2.13972836e+02 2.13681127e+02 2.13278136e+02
2.12988496e+02 2.12414805e+02 2.12245073e+02 2.11768923e+02
2.11224642e+02 2.11135544e+02 2.11130735e+02 2.10235861e+02
2.09517432e+02 2.08905657e+02 2.08752041e+02 2.08761615e+02
2.08748300e+02 2.08665560e+02 2.07970001e+02 2.07966451e+02
2.07217844e+02 2.06324243e+02 2.06275196e+02 2.05509110e+02
2.05200851e+02 2.04263646e+02 2.04002240e+02 2.03896272e+02
2.03186422e+02 2.02589562e+02 2.01995275e+02 2.01783769e+02
2.01070520e+02 2.00301492e+02 1.99821008e+02 1.99207721e+02
```

```
1.99142083e+02 1.98488819e+02 1.98114902e+02 1.98196743e+02
1.97986413e+02 1.98026622e+02 1.98157814e+02 1.98118424e+02
1.98047486e+02 1.98265341e+02 1.98530773e+02 1.98275488e+02
1.97661810e+02 1.97293292e+02 1.96808223e+02 1.96612802e+02
1.96156307e+02 1.96198341e+02 1.96045322e+02 1.96431667e+02
1.96917708e+02 1.97010602e+02 1.96859646e+02 1.97054729e+02
1.97583366e+02 1.97651361e+02 1.97894422e+02 1.97887084e+02
1.98093838e+02 1.98277408e+02 1.98338441e+02 1.98687873e+02
1.98872920e+02 1.99151847e+02 1.99614027e+02 2.00333211e+02
2.00666718e+02 2.01622288e+02 2.02547397e+02 2.02690196e+02
2.03069171e+02 2.03671741e+02 2.03963750e+02 2.04530370e+02
2.05519157e+02 2.06478663e+02 2.07410064e+02 2.08459330e+02
2.09032379e+02 2.09771225e+02 2.10873246e+02 2.12206472e+02
2.13231903e+02 2.14023645e+02 2.14589707e+02 2.15862127e+02
2.16457796e+02 2.17443382e+02 2.18611343e+02 2.20194767e+02
2.20968808e+02 2.22081858e+02 2.22835139e+02 2.23927861e+02
2.25320606e+02 2.26389829e+02 2.27408596e+02 2.28351379e+02
2.29772848e+02 2.31328860e+02 2.32877123e+02 2.34716991e+02
2.35624521e+02 2.37189473e+02 2.38719641e+02 2.40249262e+02
2.42163088e+02 2.43140254e+02 2.44805319e+02 2.46594915e+02
2.48087009e+02 2.49408663e+02 2.51191617e+02 2.52284422e+02
2.53724485e+02 2.55243944e+02 2.56937900e+02 2.58028292e+02
2.59254078e+02 2.60659942e+02 2.62275351e+02 2.64150001e+02
2.65751114e+02 2.66860492e+02 2.68809655e+02 2.70641735e+02
2.72095512e+02 2.73958741e+02 2.75429898e+02 2.77270959e+02
2.79005683e+02 2.80864976e+02 2.82661066e+02 2.83557842e+02
2.84576942e+02 2.85793157e+02 2.87554316e+02 2.88644500e+02
2.89575481e+02 2.91211897e+02 2.92387260e+02 2.94002242e+02
2.94847020e+02 2.96169449e+02 2.97247137e+02 2.97936531e+02
2.99220779e+02 3.00030261e+02 3.00605704e+02 3.01174690e+02
3.01717522e+02 3.03079245e+02 3.04244908e+02 3.05540119e+02
3.06580825e+02 3.07448503e+02 3.08061967e+02 3.09338920e+02
3.09949799e+02 3.10638591e+02 3.11204218e+02 3.12340179e+02
3.12640171e+02 3.13062267e+02 3.13805406e+02 3.13900025e+02
3.14507274e+02 3.14953510e+02 3.15174719e+02 3.15319024e+02
3.15708204e+02 3.15919697e+02 3.16188835e+02 3.16308169e+02
3.17029973e+02 3.16884422e+02 3.16672209e+02 3.16676862e+02
3.16899765e+02 3.17161793e+02 3.17133271e+02 3.17678558e+02
3.17548698e+02 3.17761063e+02 3.17862819e+02 3.18113789e+02
3.18502230e+02 3.18779116e+02 3.18906254e+02 3.18943191e+02
3.18594142e+02 3.18034721e+02 3.17368388e+02 3.16860179e+02
3.16521947e+02 3.16253788e+02 3.16217855e+02 3.16349515e+02
3.15622508e+02 3.14868770e+02 3.14582158e+02 3.14436549e+02
3.14379873e+02 3.14397950e+02 3.14451600e+02 3.13910521e+02
3.13339799e+02 3.12527314e+02 3.12243666e+02 3.11924511e+02
3.11804286e+02 3.10913856e+02 3.10419518e+02 3.10005038e+02
3.09248568e+02 3.08418087e+02 3.08277668e+02 3.07336696e+02
3.06809291e+02 3.05929563e+02 3.05394507e+02 3.05386781e+02
3.04828200e+02 3.04709408e+02 3.03858140e+02 3.03169099e+02
3.02621835e+02 3.02080766e+02 3.01822373e+02 3.01249337e+02
3.01058277e+02 3.00806457e+02 3.00329312e+02 2.99900330e+02
2.99670529e+02 2.98913491e+02 2.98926881e+02 2.98790147e+02
2.98745976e+02 2.98532788e+02 2.97914297e+02 2.97612225e+02
2.97333052e+02 2.96992783e+02 2.96385309e+02 2.96509657e+02
2.96031656e+02 2.96075880e+02 2.96243271e+02 2.96369016e+02
2.96846405e+02 2.96495712e+02 2.96937100e+02 2.96569143e+02
2.96735935e+02 2.96801441e+02 2.97016607e+02 2.97102716e+02
2.97044593e+02 2.96845573e+02 2.97176341e+02 2.97862684e+02
2.98110566e+02 2.98239322e+02 2.98257063e+02 2.98310964e+02
```

```
2.98532360e+02 2.98663785e+02 2.98991615e+02 2.99151873e+02
2.99342811e+02 3.00355522e+02 3.00705023e+02 3.01482849e+02
3.02504994e+02 3.03240395e+02 3.03702781e+02 3.04241115e+02
3.05042224e+02 3.05967606e+02 3.06742825e+02 3.07978295e+02
3.09211071e+02 3.10647324e+02 3.11212103e+02 3.12669878e+02
3.13650232e+02 3.15141511e+02 3.16469720e+02 3.18118381e+02
3.19562210e+02 3.20633862e+02 3.21863894e+02 3.22990394e+02
3.24125819e+02 3.25176878e+02 3.26481866e+02 3.27990686e+02
3.29112354e+02 3.30497476e+02 3.31491163e+02 3.32542015e+02
3.33493648e+02 3.35382135e+02 3.36315771e+02 3.37435232e+02
3.38999155e+02 3.40040819e+02 3.41891298e+02 3.43586835e+02
3.45535512e+02 3.47030836e+02 3.48323355e+02 3.49868417e+02
3.51797263e+02 3.53317531e+02 3.54584738e+02 3.56461643e+02
3.57831589e+02 3.58828534e+02 3.60068334e+02 3.61374255e+02
3.63215320e+02 3.64649740e+02 3.66062911e+02 3.67359307e+02
3.69191041e+02 3.71076949e+02 3.72998616e+02 3.74293111e+02
3.76164086e+02 3.77846972e+02 3.79398968e+02 3.80504599e+02
3.81565385e+02 3.82558513e+02 3.84290287e+02 3.85374518e+02
3.86598126e+02 3.87842195e+02 3.89349335e+02 3.90900677e+02
3.91724296e+02 3.92812513e+02 3.94292791e+02 3.95723732e+02
3.96413931e+02 3.97132398e+02 3.97972423e+02 3.98556824e+02
3.99463240e+02 4.00024848e+02 4.01018845e+02 4.01662550e+02
4.02627108e+02 4.03083371e+02 4.04368272e+02 4.05579824e+02
4.06220311e+02 4.06680633e+02 4.07118413e+02 4.07758241e+02
4.07923567e+02 4.08657380e+02 4.09128474e+02 4.09537291e+02
4.10346656e+02 4.11024281e+02 4.11915824e+02 4.12173652e+02
4.12945861e+02 4.13559394e+02 4.13729437e+02 4.14132960e+02
4.14172840e+02 4.14768248e+02 4.14571659e+02 4.14445355e+02
4.14260264e+02 4.14266142e+02 4.14585596e+02 4.14658513e+02
4.14664435e+02 4.14155940e+02 4.14045515e+02 4.13546985e+02
4.13167453e+02 4.13486940e+02 4.13064914e+02 4.13261940e+02
4.13380286e+02 4.12832326e+02 4.12709069e+02 4.12074620e+02
4.12031995e+02 4.11879302e+02 4.11882224e+02 4.11534528e+02
4.11605063e+02 4.10793670e+02 4.10210252e+02 4.10034298e+02
4.09547580e+02 4.08802777e+02 4.07980211e+02 4.07856669e+02
4.07378756e+02 4.07300746e+02 4.07065452e+02 4.06499261e+02
4.05890485e+02 4.05418435e+02 4.05315380e+02 4.05057097e+02
4.04064223e+02 4.03758874e+02 4.03678382e+02 4.03389331e+02
4.02568310e+02 4.02056266e+02 4.01204467e+02 4.00575774e+02
4.00530604e+02 4.00477993e+02 3.99792246e+02 3.99171584e+02
3.98820740e+02 3.98843057e+02 3.98061081e+02 3.97400243e+02
3.97368973e+02 3.96969859e+02 3.96992513e+02 3.96317289e+02
3.96005630e+02 3.95504819e+02 3.95012344e+02 3.94669066e+02
3.94331631e+02 3.93938866e+02 3.94124869e+02 3.94149506e+02
3.94192506e+02 3.93821107e+02 3.93761393e+02 3.93927665e+02
3.94135470e+02 3.93917917e+02 3.94152963e+02 3.94379816e+02
3.94039569e+02 3.94451524e+02 3.94467066e+02 3.94556328e+02
3.94636573e+02 3.94865681e+02 3.95177833e+02 3.95397022e+02
3.96015120e+02 3.96424712e+02 3.96638759e+02 3.96899016e+02
3.97584521e+02 3.97838605e+02 3.97987269e+02 3.98015242e+02
3.98891269e+02 3.99841226e+02 4.00024136e+02 4.00452718e+02
4.00753424e+02 4.01025757e+02 4.02113005e+02 4.02388025e+02
4.03617201e+02 4.03992643e+02 4.05165130e+02 4.06070374e+02
4.06806416e+02 4.07545907e+02 4.08729050e+02 4.09497614e+02
4.10636687e+02 4.11663054e+02 4.12683847e+02 4.14147092e+02
4.15289522e+02 4.16729291e+02 4.18092289e+02 4.18792158e+02
4.20249815e+02 4.21173792e+02 4.22805910e+02 4.23598945e+02
4.24702531e+02 4.25592419e+02 4.26877025e+02 4.28524128e+02
4.29666643e+02 4.30678320e+02 4.31884894e+02 4.33589780e+02
```

```
4.34634991e+02 4.35780108e+02 4.36778737e+02 4.38425328e+02
4.39771617e+02 4.41042751e+02 4.42729961e+02 4.44041855e+02
4.45751913e+02 4.47554801e+02 4.48764579e+02 4.50733959e+02
4.51894344e+02 4.53184692e+02 4.54364487e+02 4.55709499e+02
4.57187587e+02 4.58705325e+02 4.60551045e+02 4.62428182e+02
4.63630573e+02 4.65229383e+02 4.66309463e+02 4.67728726e+02
4.69002116e+02 4.70259498e+02 4.71671858e+02 4.73319441e+02
4.74293450e+02 4.76063630e+02 4.77674751e+02 4.78711992e+02
4.80495481e+02 4.81828874e+02 4.83637699e+02 4.84625114e+02
4.85930535e+02 4.86970894e+02 4.88004037e+02 4.89302209e+02
4.90908475e+02 4.92375051e+02 4.93736416e+02 4.94908497e+02
4.96418581e+02 4.97473025e+02 4.98882684e+02 5.00225707e+02
5.00921297e+02 5.02146512e+02 5.02584463e+02 5.03304440e+02
5.03902132e+02 5.04747733e+02 5.05793603e+02 5.06170271e+02
5.06933883e+02 5.08090438e+02 5.08506466e+02 5.09340042e+02
5.10058255e+02 5.10162427e+02 5.10701044e+02 5.11441225e+02
5.11485200e+02 5.12333311e+02 5.12790602e+02 5.12727155e+02
5.12826235e+02 5.13623476e+02 5.14018415e+02 5.13907744e+02
5.13946599e+02 5.14124311e+02 5.13836244e+02 5.13909122e+02
5.14030456e+02 5.13876986e+02 5.14039846e+02 5.14339095e+02
5.14093176e+02 5.14204634e+02 5.14308430e+02 5.14694887e+02
5.14885378e+02 5.15120816e+02 5.14973812e+02 5.14497849e+02
5.14809117e+02 5.14231366e+02 5.13973854e+02 5.13291836e+02
5.13465174e+02 5.13639944e+02 5.13550316e+02 5.13073228e+02
5.12483283e+02 5.11887844e+02 5.11138831e+02 5.10773634e+02
5.10010624e+02 5.09409600e+02 5.09106530e+02 5.08893193e+02
5.07966409e+02 5.07436231e+02 5.07119773e+02 5.06997102e+02
5.06490140e+02 5.06471657e+02 5.05745175e+02 5.04763122e+02
5.04248848e+02 5.03505455e+02 5.03329173e+02 5.02562439e+02
5.01875041e+02 5.01670707e+02 5.01393735e+02 5.00964098e+02
5.00686759e+02 5.00129479e+02 4.99166206e+02 4.98217268e+02
4.97777433e+02 4.96919843e+02 4.96041142e+02 4.96088904e+02
4.96043080e+02 4.95304100e+02 4.94428458e+02 4.94509384e+02
4.93943381e+02 4.93302198e+02 4.93184689e+02 4.92503438e+02
4.91997574e+02 4.92222558e+02 4.92133052e+02 4.91946623e+02
4.91659995e+02 4.91773184e+02 4.91276717e+02 4.91631148e+02
4.91904486e+02 4.91960059e+02 4.91467139e+02 4.91658953e+02
4.92090484e+02 4.92161070e+02 4.92089516e+02 4.91812645e+02
4.92199422e+02 4.92745706e+02 4.92536940e+02 4.93016934e+02
4.92785304e+02 4.93534216e+02 4.93961893e+02 4.94357897e+02
4.94528514e+02 4.95363697e+02 4.95566812e+02 4.96363548e+02]
        [  7.98158289   8.71682655   9.9660543   10.94447506  12.25198972
 13.39851181  14.60551976  15.61588556  17.10779599  18.31386957
 19.32550987  20.54481013  21.20142732  22.164314    23.50903465
 24.50621904  25.85320309  27.03208286  27.93807017  29.02650751
 30.40548941  31.82344275  33.24209592  34.75603877  36.28444872
 37.60687368  39.40824733  40.69356382  42.05920524  43.89200936
 45.64925986  47.31344213  48.38225218  50.27765156  51.97418011
 53.96071546  55.10227846  56.96596648  58.12648615  59.74155227
 60.86537225  62.71288704  64.51823273  66.08289543  67.48219343
 68.53904877  70.21876479  71.64822424  73.338863    75.16553901
 77.09211703  78.88880114  79.83029171  81.1080244   82.74284202
 83.80547822  85.2028215   86.11790152  87.16222597  88.00782834
 89.61054304  90.62462274  91.74048766  93.41868002  94.85206305
 95.59100876  96.44025003  97.72304029  98.93769287  99.78849275
101.310492   102.48654133 103.55800093 104.65695232 105.86882802
106.63476352 107.45876387 108.06575551 108.62007307 109.90318338
110.95175117 111.72120109 112.1973056  112.66980532 112.9152158
113.50606689 113.94319601 114.73364781 115.17419017 115.3852046
```

```
115.40988333 115.4457222  116.06232446 116.42191299 116.83315897
117.57339579 118.37635343 118.37510717 118.78949549 118.77382676
118.48546076 118.9051015  118.85699409 118.8433101  119.00584792
119.38290587 119.53013404 119.89374328 119.63145852 119.86742198
119.46527267 119.80515728 119.85522156 119.40941737 119.67224086
119.69598988 119.2209629  118.68416381 118.43185128 117.66735898
117.06581206 116.66341696 116.19325901 115.79609241 115.19741444
114.89319226 114.85222081 114.05199804 113.63960066 112.830788
112.59659117 112.03235718 111.62919534 110.83655841 109.99911892
109.49948686 108.86298489 108.80785488 108.5751534  108.32431046
108.2280302  107.31194607 106.86611182 106.45502592 106.4248476
105.72930678 104.98784831 104.11222549 103.16007196 103.12930759
102.84816762 102.69243977 102.04439339 101.71304893 100.87217714
100.46679821 100.56798667 100.58374989 100.07758092 100.2120705
 99.63475513  99.79391894  99.9647834  100.01387492  99.91535423
100.08253541  99.69100859  99.87864028  99.85909298  99.25942279
 99.01887105  98.60492854  99.05093113  99.02026002  99.03589771
 99.22137973  99.16418505  98.90393743  99.35793061  99.2090406
 99.41134259  99.35666851  99.20582311  99.84987138 100.63540956
101.43980975 102.22103202 102.90097104 103.17132567 103.2210163
103.62825747 103.89190237 104.08718053 104.23471602 105.08564362
105.25350554 106.21146761 106.57655749 106.90476946 107.2733636
108.2544284  108.83853353 109.62719755 110.58171424 111.8680447
113.04907946 113.80116104 114.44168526 115.03288637 115.89656839
116.74647179 117.81551941 119.13622473 120.49316204 121.461228
122.54826173 123.45838132 124.99703791 125.82439922 127.31155148
128.89926387 130.503752   131.60750249 132.84205766 134.30596397
135.4697924  136.74547225 137.86028116 139.24991353 140.2354066
141.986259   143.02350913 144.51092744 145.7936543  147.3534845
149.30060618 150.93829113 151.96921553 153.3976447  154.90716811
156.44334561 158.12424468 159.3998675  160.52429003 161.90908041
163.85317447 165.02259262 166.90249333 168.41488245 169.83208755
171.66518548 173.06467021 174.71861433 176.03539428 177.84426573
179.42529727 181.001226   182.18968843 183.790795   185.25393663
186.3029739  187.25366773 188.82057245 190.52985012 191.71695758
193.01504852 194.55731827 195.67585359 197.26515206 198.45381028
199.86236265 201.33328974 202.68464036 203.35309624 204.46374835
205.31617311 205.80466535 206.62584549 207.25265411 208.44043034
208.78275196 210.02072579 210.62478453 211.19963466 211.40334448
211.74501127 212.271604   213.29500373 213.45740918 214.43412147
215.30361    215.72636164 215.990272   216.12890782 216.61803929
216.49467942 216.32290417 216.53355665 216.35294084 216.32589072
216.23803464 216.15248791 215.91541859 215.53030693 215.22001193
215.38450169 215.87700423 216.35830782 216.23153512 215.83240616
215.88338267 215.76078096 216.11276675 215.51675909 215.61544642
215.19673814 214.70918813 214.62158163 214.09713158 213.97283568
213.6811272  213.27813562 212.9884955  212.41480499 212.24507302
211.76892337 211.22464187 211.13554416 211.13073459 210.23586113
209.51743161 208.90565729 208.75204061 208.76161528 208.74829973
208.66555973 207.9700013  207.96645058 207.21784389 206.32424348
206.27519609 205.50910979 205.20085133 204.26364572 204.00224012
203.89627199 203.18642164 202.58956177 201.99527479 201.78376936
201.07052009 200.30149242 199.82100758 199.20772136 199.14208343
198.48881874 198.11490151 198.19674309 197.98641286 198.02662169
198.15781438 198.11842423 198.04748605 198.26534055 198.53077271
198.27548761 197.66181028 197.29329245 196.80822317 196.61280249
196.15630656 196.19834099 196.04532225 196.43166672 196.91770772
197.01060212 196.8596464  197.05472893 197.58336553 197.65136096
197.89442224 197.8870839  198.09383817 198.27740846 198.33844061
```

```
198.68787253 198.87292012 199.15184667 199.6140266  200.33321086
200.66671832 201.62228785 202.54739692 202.69019553 203.06917128
203.67174126 203.96375046 204.5303697  205.51915711 206.4786627
207.41006412 208.45933044 209.03237868 209.77122469 210.87324646
212.20647172 213.23190345 214.02364452 214.58970657 215.86212678
216.45779613 217.44338226 218.61134346 220.19476679 220.96880844
222.08185754 222.83513937 223.92786142 225.32060572 226.38982869
227.40859563 228.35137944 229.77284761 231.32885964 232.87712346
234.71699101 235.62452127 237.18947255 238.71964127 240.24926165
242.16308849 243.14025385 244.80531944 246.59491485 248.0870093
249.40866251 251.19161737 252.28442226 253.72448463 255.24394357
256.93789998 258.02829227 259.2540785  260.65994203 262.2753514
264.15000052 265.75111394 266.86049217 268.80965547 270.64173489
272.09551217 273.95874092 275.4298982  277.27095889 279.00568331
280.86497624 282.66106574 283.55784158 284.57694151 285.79315676
287.55431645 288.64450038 289.57548127 291.21189701 292.38726005
294.00224243 294.84702    296.16944937 297.24713712 297.93653094
299.22077934 300.03026147 300.60570379 301.1746895  301.71752171
303.07924517 304.24490838 305.54011863 306.58082549 307.44850334
308.0619667  309.33892006 309.949799   310.63859119 311.20421768
312.34017908 312.6401707  313.06226701 313.8054062  313.90002457
314.50727395 314.95350969 315.17471915 315.31902412 315.70820363
315.91969674 316.18883532 316.30816946 317.02997326 316.88442159
316.67220855 316.67686236 316.89976471 317.16179329 317.13327074
317.67855824 317.54869756 317.76106281 317.86281871 318.1137892
318.50222961 318.77911621 318.90625419 318.94319081 318.59414187
318.03472067 317.36838804 316.86017868 316.52194692 316.25378818
316.21785474 316.34951509 315.62250764 314.86876975 314.5821578
314.43654935 314.37987268 314.39795013 314.45159989 313.91052081
313.33979948 312.52731447 312.24366575 311.92451094 311.80428629
310.91385594 310.41951815 310.00503811 309.24856842 308.41808727
308.27766811 307.33669647 306.80929064 305.92956268 305.39450674
305.38678073 304.82819983 304.70940798 303.85814039 303.16909878
302.621835   302.08076637 301.82237337 301.24933687 301.05827668
300.80645706 300.32931152 299.90032963 299.6705294  298.91349102
298.92688076 298.79014691 298.74597613 298.53278806 297.91429654
297.61222538 297.33305185 296.99278298 296.38530931 296.5096565
296.03165571 296.07588046 296.24327117 296.36901607 296.84640465
296.49571231 296.93710017 296.56914252 296.73593482 296.80144052
297.01660738 297.10271615 297.04459318 296.84557284 297.17634134
297.86268374 298.11056646 298.23932196 298.25706345 298.31096361
298.53236006 298.66378488 298.99161509 299.15187287 299.34281097
300.35552162 300.70502319 301.48284884 302.50499427 303.24039494
303.70278081 304.24111488 305.04222363 305.96760561 306.7428252
307.9782953  309.21107062 310.64732384 311.21210319 312.66987786
313.6502319  315.14151145 316.46972021 318.11838083 319.56221039
320.6338618  321.8638936  322.99039382 324.1258189  325.17687842
326.4818657  327.99068622 329.11235442 330.49747626 331.49116323
332.54201504 333.49364845 335.38213452 336.31577136 337.43523209
338.99915536 340.04081865 341.89129831 343.58683523 345.53551245
347.03083634 348.32335473 349.86841726 351.79726271 353.31753071
354.58473774 356.46164309 357.83158857 358.82853388 360.0683336
361.37425545 363.21532017 364.6497401  366.06291055 367.3593065
369.19104114 371.07694869 372.9986155  374.2931114  376.16408589
377.84697199 379.39896782 380.50459932 381.56538451 382.55851291
384.29028651 385.37451819 386.59812567 387.84219458 389.3493348
390.90067654 391.72429635 392.81251258 394.29279145 395.72373181
396.41393082 397.13239755 397.97242304 398.55682352 399.4632402
400.0248484  401.01884529 401.66255012 402.62710774 403.0833713
```

```
404.36827216 405.57982447 406.2203113  406.68063314 407.11841266
407.75824066 407.9235667  408.65738012 409.12847355 409.53729088
410.34665585 411.02428055 411.91582385 412.17365177 412.9458613
413.55939403 413.72943736 414.13295956 414.17283963 414.76824844
414.57165876 414.4453552  414.26026394 414.26614205 414.58559572
414.65851267 414.66443532 414.15593987 414.0455148  413.5469852
413.1674532  413.48694008 413.0649143  413.26194007 413.38028634
412.83232579 412.70906912 412.07461992 412.03199484 411.8793021
411.88222367 411.53452765 411.60506281 410.79366973 410.21025161
410.03429769 409.54757985 408.80277658 407.98021084 407.85666918
407.37875564 407.30074563 407.0654516  406.49926104 405.89048481
405.41843481 405.31538006 405.05709668 404.06422292 403.75887421
403.67838161 403.38933081 402.56830987 402.05626603 401.20446734
400.57577428 400.53060407 400.47799262 399.79224631 399.17158367
398.82074002 398.84305655 398.0610814  397.40024341 397.36897319
396.9698589  396.99251331 396.31728918 396.00562984 395.5048189
395.01234446 394.66906642 394.33163062 393.93886643 394.12486857
394.14950598 394.19250586 393.82110745 393.76139288 393.92766526
394.13547038 393.91791746 394.15296342 394.3798159  394.03956872
394.45152369 394.46706586 394.55632842 394.63657305 394.86568113
395.17783338 395.39702184 396.0151205  396.42471201 396.63875865
396.89901594 397.58452132 397.83860495 397.98726923 398.01524166
398.8912685  399.84122571 400.02413642 400.45271847 400.75342377
401.02575671 402.11300477 402.38802527 403.61720082 403.99264278
405.16513043 406.07037445 406.80641649 407.54590676 408.7290502
409.49761399 410.63668658 411.66305408 412.68384652 414.14709207
415.28952238 416.72929104 418.09228856 418.79215767 420.24981483
421.17379186 422.80591014 423.5989454  424.70253067 425.59241863
426.87702475 428.52412776 429.66664304 430.67831996 431.88489368
433.58978035 434.63499136 435.78010843 436.77873688 438.42532769
439.77161745 441.04275112 442.72996067 444.04185497 445.7519135
447.55480123 448.76457878 450.73395944 451.89434412 453.18469158
454.36448687 455.70949909 457.18758671 458.70532454 460.55104528
462.42818153 463.63057264 465.22938344 466.30946266 467.7287262
469.00211626 470.25949777 471.6718585  473.31944076 474.29345047
476.06363033 477.67475079 478.7119922  480.49548104 481.82887384
483.63769941 484.6251139  485.93053534 486.97089445 488.00403663
489.3022088  490.90847497 492.37505069 493.73641608 494.90849691
496.41858067 497.47302545 498.88268443 500.22570749 500.92129651
502.14651247 502.58446292 503.30444035 503.90213235 504.74773322
505.79360338 506.17027085 506.93388294 508.09043821 508.50646607
509.34004168 510.05825519 510.16242719 510.70104391 511.44122506
511.48520049 512.33331122 512.79060211 512.72715485 512.82623523
513.62347646 514.01841488 513.9077442  513.94659882 514.12431134
513.83624441 513.90912201 514.03045609 513.87698607 514.03984578
514.3390954  514.09317595 514.20463444 514.30842988 514.69488681
514.88537772 515.12081594 514.97381186 514.49784858 514.80911677
514.23136575 513.97385444 513.29183647 513.46517408 513.63994401
513.55031608 513.07322835 512.48328282 511.88784412 511.13883147
510.77363389 510.01062412 509.40960016 509.10653015 508.89319299
507.96640918 507.43623148 507.11977291 506.99710212 506.49013966
506.47165719 505.74517503 504.76312178 504.24884791 503.50545515
503.32917282 502.56243893 501.87504142 501.67070689 501.39373544
500.96409833 500.68675914 500.12947925 499.16620613 498.21726757
497.77743285 496.91984306 496.04114205 496.08890406 496.04307965
495.30410035 494.42845789 494.50938366 493.94338103 493.30219806
493.18468918 492.5034379  491.99757426 492.22255787 492.13305202
491.94662303 491.65999454 491.77318362 491.27671711 491.63114786
491.90448562 491.96005922 491.46713862 491.65895285 492.09048352
```

```
492.16107008 492.08951561 491.81264507 492.19942162 492.74570555
492.5369403  493.01693375 492.78530435 493.53421603 493.96189273
494.35789733 494.52851393 495.36369728 495.56681239 496.36354825]
        [2.48753144e-01 8.56321237e-01 1.51115369e+00 2.17751391e+00
2.52592877e+00 3.63511225e+00 4.26961894e+00 5.33417086e+00
6.28234002e+00 6.85876808e+00 7.98158289e+00 8.71682655e+00
9.96605430e+00 1.09444751e+01 1.22519897e+01 1.33985118e+01
1.46055198e+01 1.56158856e+01 1.71077960e+01 1.83138696e+01
1.93255099e+01 2.05448101e+01 2.12014273e+01 2.21643140e+01
2.35090346e+01 2.45062190e+01 2.58532031e+01 2.70320829e+01
2.79380702e+01 2.90265075e+01 3.04054894e+01 3.18234427e+01
3.32420959e+01 3.47560388e+01 3.62844487e+01 3.76068737e+01
3.94082473e+01 4.06935638e+01 4.20592052e+01 4.38920094e+01
4.56492599e+01 4.73134421e+01 4.83822522e+01 5.02776516e+01
5.19741801e+01 5.39607155e+01 5.51022785e+01 5.69659665e+01
5.81264861e+01 5.97415523e+01 6.08653723e+01 6.27128870e+01
6.45182327e+01 6.60828954e+01 6.74821934e+01 6.85390488e+01
7.02187648e+01 7.16482242e+01 7.33388630e+01 7.51655390e+01
7.70921170e+01 7.88888011e+01 7.98302917e+01 8.11080244e+01
8.27428420e+01 8.38054782e+01 8.52028215e+01 8.61179015e+01
8.71622260e+01 8.80078283e+01 8.96105430e+01 9.06246227e+01
9.17404877e+01 9.34186800e+01 9.48520631e+01 9.55910088e+01
9.64402500e+01 9.77230403e+01 9.89376929e+01 9.97884927e+01
1.01310492e+02 1.02486541e+02 1.03558001e+02 1.04656952e+02
1.05868828e+02 1.06634764e+02 1.07458764e+02 1.08065756e+02
1.08620073e+02 1.09903183e+02 1.10951751e+02 1.11721201e+02
1.12197306e+02 1.12669805e+02 1.12915216e+02 1.13506067e+02
1.13943196e+02 1.14733648e+02 1.15174190e+02 1.15385205e+02
1.15409883e+02 1.15445722e+02 1.16062324e+02 1.16421913e+02
1.16833159e+02 1.17573396e+02 1.18376353e+02 1.18375107e+02
1.18789495e+02 1.18773827e+02 1.18485461e+02 1.18905101e+02
1.18856994e+02 1.18843310e+02 1.19005848e+02 1.19382906e+02
1.19530134e+02 1.19893743e+02 1.19631459e+02 1.19867422e+02
1.19465273e+02 1.19805157e+02 1.19855222e+02 1.19409417e+02
1.19672241e+02 1.19695990e+02 1.19220963e+02 1.18684164e+02
1.18431851e+02 1.17667359e+02 1.17065812e+02 1.16663417e+02
1.16193259e+02 1.15796092e+02 1.15197414e+02 1.14893192e+02
1.14852221e+02 1.14051998e+02 1.13639601e+02 1.12830788e+02
1.12596591e+02 1.12032357e+02 1.11629195e+02 1.10836558e+02
1.09999119e+02 1.09499487e+02 1.08862985e+02 1.08807855e+02
1.08575153e+02 1.08324310e+02 1.08228030e+02 1.07311946e+02
1.06866112e+02 1.06455026e+02 1.06424848e+02 1.05729307e+02
1.04987848e+02 1.04112225e+02 1.03160072e+02 1.03129308e+02
1.02848168e+02 1.02692440e+02 1.02044393e+02 1.01713049e+02
1.00872177e+02 1.00466798e+02 1.00567987e+02 1.00583750e+02
1.00077581e+02 1.00212070e+02 9.96347551e+01 9.97939189e+01
9.99647834e+01 1.00013875e+02 9.99153542e+01 1.00082535e+02
9.96910086e+01 9.98786403e+01 9.98590930e+01 9.92594228e+01
9.90188711e+01 9.86049285e+01 9.90509311e+01 9.90202600e+01
9.90358977e+01 9.92213797e+01 9.91641850e+01 9.89039374e+01
9.93579306e+01 9.92090406e+01 9.94113426e+01 9.93566685e+01
9.92058231e+01 9.98498714e+01 1.00635410e+02 1.01439810e+02
1.02221032e+02 1.02900971e+02 1.03171326e+02 1.03221016e+02
1.03628257e+02 1.03891902e+02 1.04087181e+02 1.04234716e+02
1.05085644e+02 1.05253506e+02 1.06211468e+02 1.06576557e+02
1.06904769e+02 1.07273364e+02 1.08254428e+02 1.08838534e+02
1.09627198e+02 1.10581714e+02 1.11868045e+02 1.13049079e+02
1.13801161e+02 1.14441685e+02 1.15032886e+02 1.15896568e+02
1.16746472e+02 1.17815519e+02 1.19136225e+02 1.20493162e+02
```

```
1.21461228e+02 1.22548262e+02 1.23458381e+02 1.24997038e+02
1.25824399e+02 1.27311551e+02 1.28899264e+02 1.30503752e+02
1.31607502e+02 1.32842058e+02 1.34305964e+02 1.35469792e+02
1.36745472e+02 1.37860281e+02 1.39249914e+02 1.40235407e+02
1.41986259e+02 1.43023509e+02 1.44510927e+02 1.45793654e+02
1.47353485e+02 1.49300606e+02 1.50938291e+02 1.51969216e+02
1.53397645e+02 1.54907168e+02 1.56443346e+02 1.58124245e+02
1.59399868e+02 1.60524290e+02 1.61909080e+02 1.63853174e+02
1.65022593e+02 1.66902493e+02 1.68414882e+02 1.69832088e+02
1.71665185e+02 1.73064670e+02 1.74718614e+02 1.76035394e+02
1.77844266e+02 1.79425297e+02 1.81001226e+02 1.82189688e+02
1.83790795e+02 1.85253937e+02 1.86302974e+02 1.87253668e+02
1.88820572e+02 1.90529850e+02 1.91716958e+02 1.93015049e+02
1.94557318e+02 1.95675854e+02 1.97265152e+02 1.98453810e+02
1.99862363e+02 2.01333290e+02 2.02684640e+02 2.03353096e+02
2.04463748e+02 2.05316173e+02 2.05804665e+02 2.06625845e+02
2.07252654e+02 2.08440430e+02 2.08782752e+02 2.10020726e+02
2.10624785e+02 2.11199635e+02 2.11403344e+02 2.11745011e+02
2.12271604e+02 2.13295004e+02 2.13457409e+02 2.14434121e+02
2.15303610e+02 2.15726362e+02 2.15990272e+02 2.16128908e+02
2.16618039e+02 2.16494679e+02 2.16322904e+02 2.16533557e+02
2.16352941e+02 2.16325891e+02 2.16238035e+02 2.16152488e+02
2.15915419e+02 2.15530307e+02 2.15220012e+02 2.15384502e+02
2.15877004e+02 2.16358308e+02 2.16231535e+02 2.15832406e+02
2.15883383e+02 2.15760781e+02 2.16112767e+02 2.15516759e+02
2.15615446e+02 2.15196738e+02 2.14709188e+02 2.14621582e+02
2.14097132e+02 2.13972836e+02 2.13681127e+02 2.13278136e+02
2.12988496e+02 2.12414805e+02 2.12245073e+02 2.11768923e+02
2.11224642e+02 2.11135544e+02 2.11130735e+02 2.10235861e+02
2.09517432e+02 2.08905657e+02 2.08752041e+02 2.08761615e+02
2.08748300e+02 2.08665560e+02 2.07970001e+02 2.07966451e+02
2.07217844e+02 2.06324243e+02 2.06275196e+02 2.05509110e+02
2.05200851e+02 2.04263646e+02 2.04002240e+02 2.03896272e+02
2.03186422e+02 2.02589562e+02 2.01995275e+02 2.01783769e+02
2.01070520e+02 2.00301492e+02 1.99821008e+02 1.99207721e+02
1.99142083e+02 1.98488819e+02 1.98114902e+02 1.98196743e+02
1.97986413e+02 1.98026622e+02 1.98157814e+02 1.98118424e+02
1.98047486e+02 1.98265341e+02 1.98530773e+02 1.98275488e+02
1.97661810e+02 1.97293292e+02 1.96808223e+02 1.96612802e+02
1.96156307e+02 1.96198341e+02 1.96045322e+02 1.96431667e+02
1.96917708e+02 1.97010602e+02 1.96859646e+02 1.97054729e+02
1.97583366e+02 1.97651361e+02 1.97894422e+02 1.97887084e+02
1.98093838e+02 1.98277408e+02 1.98338441e+02 1.98687873e+02
1.98872920e+02 1.99151847e+02 1.99614027e+02 2.00333211e+02
2.00666718e+02 2.01622288e+02 2.02547397e+02 2.02690196e+02
2.03069171e+02 2.03671741e+02 2.03963750e+02 2.04530370e+02
2.05519157e+02 2.06478663e+02 2.07410064e+02 2.08459330e+02
2.09032379e+02 2.09771225e+02 2.10873246e+02 2.12206472e+02
2.13231903e+02 2.14023645e+02 2.14589707e+02 2.15862127e+02
2.16457796e+02 2.17443382e+02 2.18611343e+02 2.20194767e+02
2.20968808e+02 2.22081858e+02 2.22835139e+02 2.23927861e+02
2.25320606e+02 2.26389829e+02 2.27408596e+02 2.28351379e+02
2.29772848e+02 2.31328860e+02 2.32877123e+02 2.34716991e+02
2.35624521e+02 2.37189473e+02 2.38719641e+02 2.40249262e+02
2.42163088e+02 2.43140254e+02 2.44805319e+02 2.46594915e+02
2.48087009e+02 2.49408663e+02 2.51191617e+02 2.52284422e+02
2.53724485e+02 2.55243944e+02 2.56937900e+02 2.58028292e+02
2.59254078e+02 2.60659942e+02 2.62275351e+02 2.64150001e+02
2.65751114e+02 2.66860492e+02 2.68809655e+02 2.70641735e+02
```

```
2.72095512e+02 2.73958741e+02 2.75429898e+02 2.77270959e+02
2.79005683e+02 2.80864976e+02 2.82661066e+02 2.83557842e+02
2.84576942e+02 2.85793157e+02 2.87554316e+02 2.88644500e+02
2.89575481e+02 2.91211897e+02 2.92387260e+02 2.94002242e+02
2.94847020e+02 2.96169449e+02 2.97247137e+02 2.97936531e+02
2.99220779e+02 3.00030261e+02 3.00605704e+02 3.01174690e+02
3.01717522e+02 3.03079245e+02 3.04244908e+02 3.05540119e+02
3.06580825e+02 3.07448503e+02 3.08061967e+02 3.09338920e+02
3.09949799e+02 3.10638591e+02 3.11204218e+02 3.12340179e+02
3.12640171e+02 3.13062267e+02 3.13805406e+02 3.13900025e+02
3.14507274e+02 3.14953510e+02 3.15174719e+02 3.15319024e+02
3.15708204e+02 3.15919697e+02 3.16188835e+02 3.16308169e+02
3.17029973e+02 3.16884422e+02 3.16672209e+02 3.16676862e+02
3.16899765e+02 3.17161793e+02 3.17133271e+02 3.17678558e+02
3.17548698e+02 3.17761063e+02 3.17862819e+02 3.18113789e+02
3.18502230e+02 3.18779116e+02 3.18906254e+02 3.18943191e+02
3.18594142e+02 3.18034721e+02 3.17368388e+02 3.16860179e+02
3.16521947e+02 3.16253788e+02 3.16217855e+02 3.16349515e+02
3.15622508e+02 3.14868770e+02 3.14582158e+02 3.14436549e+02
3.14379873e+02 3.14397950e+02 3.14451600e+02 3.13910521e+02
3.13339799e+02 3.12527314e+02 3.12243666e+02 3.11924511e+02
3.11804286e+02 3.10913856e+02 3.10419518e+02 3.10005038e+02
3.09248568e+02 3.08418087e+02 3.08277668e+02 3.07336696e+02
3.06809291e+02 3.05929563e+02 3.05394507e+02 3.05386781e+02
3.04828200e+02 3.04709408e+02 3.03858140e+02 3.03169099e+02
3.02621835e+02 3.02080766e+02 3.01822373e+02 3.01249337e+02
3.01058277e+02 3.00806457e+02 3.00329312e+02 2.99900330e+02
2.99670529e+02 2.98913491e+02 2.98926881e+02 2.98790147e+02
2.98745976e+02 2.98532788e+02 2.97914297e+02 2.97612225e+02
2.97333052e+02 2.96992783e+02 2.96385309e+02 2.96509657e+02
2.96031656e+02 2.96075880e+02 2.96243271e+02 2.96369016e+02
2.96846405e+02 2.96495712e+02 2.96937100e+02 2.96569143e+02
2.96735935e+02 2.96801441e+02 2.97016607e+02 2.97102716e+02
2.97044593e+02 2.96845573e+02 2.97176341e+02 2.97862684e+02
2.98110566e+02 2.98239322e+02 2.98257063e+02 2.98310964e+02
2.98532360e+02 2.98663785e+02 2.98991615e+02 2.99151873e+02
2.99342811e+02 3.00355522e+02 3.00705023e+02 3.01482849e+02
3.02504994e+02 3.03240395e+02 3.03702781e+02 3.04241115e+02
3.05042224e+02 3.05967606e+02 3.06742825e+02 3.07978295e+02
3.09211071e+02 3.10647324e+02 3.11212103e+02 3.12669878e+02
3.13650232e+02 3.15141511e+02 3.16469720e+02 3.18118381e+02
3.19562210e+02 3.20633862e+02 3.21863894e+02 3.22990394e+02
3.24125819e+02 3.25176878e+02 3.26481866e+02 3.27990686e+02
3.29112354e+02 3.30497476e+02 3.31491163e+02 3.32542015e+02
3.33493648e+02 3.35382135e+02 3.36315771e+02 3.37435232e+02
3.38999155e+02 3.40040819e+02 3.41891298e+02 3.43586835e+02
3.45535512e+02 3.47030836e+02 3.48323355e+02 3.49868417e+02
3.51797263e+02 3.53317531e+02 3.54584738e+02 3.56461643e+02
3.57831589e+02 3.58828534e+02 3.60068334e+02 3.61374255e+02
3.63215320e+02 3.64649740e+02 3.66062911e+02 3.67359307e+02
3.69191041e+02 3.71076949e+02 3.72998616e+02 3.74293111e+02
3.76164086e+02 3.77846972e+02 3.79398968e+02 3.80504599e+02
3.81565385e+02 3.82558513e+02 3.84290287e+02 3.85374518e+02
3.86598126e+02 3.87842195e+02 3.89349335e+02 3.90900677e+02
3.91724296e+02 3.92812513e+02 3.94292791e+02 3.95723732e+02
3.96413931e+02 3.97132398e+02 3.97972423e+02 3.98556824e+02
3.99463240e+02 4.00024848e+02 4.01018845e+02 4.01662550e+02
4.02627108e+02 4.03083371e+02 4.04368272e+02 4.05579824e+02
4.06220311e+02 4.06680633e+02 4.07118413e+02 4.07758241e+02
```

```
4.07923567e+02 4.08657380e+02 4.09128474e+02 4.09537291e+02
4.10346656e+02 4.11024281e+02 4.11915824e+02 4.12173652e+02
4.12945861e+02 4.13559394e+02 4.13729437e+02 4.14132960e+02
4.14172840e+02 4.14768248e+02 4.14571659e+02 4.14445355e+02
4.14260264e+02 4.14266142e+02 4.14585596e+02 4.14658513e+02
4.14664435e+02 4.14155940e+02 4.14045515e+02 4.13546985e+02
4.13167453e+02 4.13486940e+02 4.13064914e+02 4.13261940e+02
4.13380286e+02 4.12832326e+02 4.12709069e+02 4.12074620e+02
4.12031995e+02 4.11879302e+02 4.11882224e+02 4.11534528e+02
4.11605063e+02 4.10793670e+02 4.10210252e+02 4.10034298e+02
4.09547580e+02 4.08802777e+02 4.07980211e+02 4.07856669e+02
4.07378756e+02 4.07300746e+02 4.07065452e+02 4.06499261e+02
4.05890485e+02 4.05418435e+02 4.05315380e+02 4.05057097e+02
4.04064223e+02 4.03758874e+02 4.03678382e+02 4.03389331e+02
4.02568310e+02 4.02056266e+02 4.01204467e+02 4.00575774e+02
4.00530604e+02 4.00477993e+02 3.99792246e+02 3.99171584e+02
3.98820740e+02 3.98843057e+02 3.98061081e+02 3.97400243e+02
3.97368973e+02 3.96969859e+02 3.96992513e+02 3.96317289e+02
3.96005630e+02 3.95504819e+02 3.95012344e+02 3.94669066e+02
3.94331631e+02 3.93938866e+02 3.94124869e+02 3.94149506e+02
3.94192506e+02 3.93821107e+02 3.93761393e+02 3.93927665e+02
3.94135470e+02 3.93917917e+02 3.94152963e+02 3.94379816e+02
3.94039569e+02 3.94451524e+02 3.94467066e+02 3.94556328e+02
3.94636573e+02 3.94865681e+02 3.95177833e+02 3.95397022e+02
3.96015120e+02 3.96424712e+02 3.96638759e+02 3.96899016e+02
3.97584521e+02 3.97838605e+02 3.97987269e+02 3.98015242e+02
3.98891269e+02 3.99841226e+02 4.00024136e+02 4.00452718e+02
4.00753424e+02 4.01025757e+02 4.02113005e+02 4.02388025e+02
4.03617201e+02 4.03992643e+02 4.05165130e+02 4.06070374e+02
4.06806416e+02 4.07545907e+02 4.08729050e+02 4.09497614e+02
4.10636687e+02 4.11663054e+02 4.12683847e+02 4.14147092e+02
4.15289522e+02 4.16729291e+02 4.18092289e+02 4.18792158e+02
4.20249815e+02 4.21173792e+02 4.22805910e+02 4.23598945e+02
4.24702531e+02 4.25592419e+02 4.26877025e+02 4.28524128e+02
4.29666643e+02 4.30678320e+02 4.31884894e+02 4.33589780e+02
4.34634991e+02 4.35780108e+02 4.36778737e+02 4.38425328e+02
4.39771617e+02 4.41042751e+02 4.42729961e+02 4.44041855e+02
4.45751913e+02 4.47554801e+02 4.48764579e+02 4.50733959e+02
4.51894344e+02 4.53184692e+02 4.54364487e+02 4.55709499e+02
4.57187587e+02 4.58705325e+02 4.60551045e+02 4.62428182e+02
4.63630573e+02 4.65229383e+02 4.66309463e+02 4.67728726e+02
4.69002116e+02 4.70259498e+02 4.71671858e+02 4.73319441e+02
4.74293450e+02 4.76063630e+02 4.77674751e+02 4.78711992e+02
4.80495481e+02 4.81828874e+02 4.83637699e+02 4.84625114e+02
4.85930535e+02 4.86970894e+02 4.88004037e+02 4.89302209e+02
4.90908475e+02 4.92375051e+02 4.93736416e+02 4.94908497e+02
4.96418581e+02 4.97473025e+02 4.98882684e+02 5.00225707e+02
5.00921297e+02 5.02146512e+02 5.02584463e+02 5.03304440e+02
5.03902132e+02 5.04747733e+02 5.05793603e+02 5.06170271e+02
5.06933883e+02 5.08090438e+02 5.08506466e+02 5.09340042e+02
5.10058255e+02 5.10162427e+02 5.10701044e+02 5.11441225e+02
5.11485200e+02 5.12333311e+02 5.12790602e+02 5.12727155e+02
5.12826235e+02 5.13623476e+02 5.14018415e+02 5.13907744e+02
5.13946599e+02 5.14124311e+02 5.13836244e+02 5.13909122e+02
5.14030456e+02 5.13876986e+02 5.14039846e+02 5.14339095e+02
5.14093176e+02 5.14204634e+02 5.14308430e+02 5.14694887e+02
5.14885378e+02 5.15120816e+02 5.14973812e+02 5.14497849e+02
5.14809117e+02 5.14231366e+02 5.13973854e+02 5.13291836e+02
5.13465174e+02 5.13639944e+02 5.13550316e+02 5.13073228e+02
```

```
 5.12483283e+02 5.11887844e+02 5.11138831e+02 5.10773634e+02
 5.10010624e+02 5.09409600e+02 5.09106530e+02 5.08893193e+02
 5.07966409e+02 5.07436231e+02 5.07119773e+02 5.06997102e+02
 5.06490140e+02 5.06471657e+02 5.05745175e+02 5.04763122e+02
 5.04248848e+02 5.03505455e+02 5.03329173e+02 5.02562439e+02
 5.01875041e+02 5.01670707e+02 5.01393735e+02 5.00964098e+02
 5.00686759e+02 5.00129479e+02 4.99166206e+02 4.98217268e+02
 4.97777433e+02 4.96919843e+02 4.96041142e+02 4.96088904e+02
 4.96043080e+02 4.95304100e+02 4.94428458e+02 4.94509384e+02
 4.93943381e+02 4.93302198e+02 4.93184689e+02 4.92503438e+02
 4.91997574e+02 4.92222558e+02 4.92133052e+02 4.91946623e+02
 4.91659995e+02 4.91773184e+02 4.91276717e+02 4.91631148e+02
 4.91904486e+02 4.91960059e+02 4.91467139e+02 4.91658953e+02
 4.92090484e+02 4.92161070e+02 4.92089516e+02 4.91812645e+02
 4.92199422e+02 4.92745706e+02]
CS: [ 2.48753144e-01  8.56321237e-01  1.51115369e+00  2.17751391e+00
  2.52592877e+00  3.63511225e+00  4.26961894e+00  5.33417086e+00
  6.28234002e+00  6.85876808e+00  7.73282975e+00  7.86050531e+00
  8.45490061e+00  8.76696116e+00  9.72606094e+00  9.76339956e+00
  1.03359008e+01  1.02817147e+01  1.08254560e+01  1.14551015e+01
  1.13439270e+01  1.18279836e+01  1.12353730e+01  1.12198389e+01
  1.12570449e+01  1.11077072e+01  1.12476833e+01  1.14161973e+01
  1.08302742e+01  1.07126379e+01  1.10799795e+01  1.12786326e+01
  1.20406686e+01  1.25917248e+01  1.27754141e+01  1.31006546e+01
  1.35550442e+01  1.36614810e+01  1.41211351e+01  1.48655019e+01
  1.52437705e+01  1.54899994e+01  1.51401563e+01  1.55216128e+01
  1.56897314e+01  1.63538418e+01  1.56940311e+01  1.62724027e+01
  1.60672809e+01  1.58495429e+01  1.52161124e+01  1.53994449e+01
  1.61359805e+01  1.58052439e+01  1.55080133e+01  1.45783333e+01
  1.51164863e+01  1.46822578e+01  1.52123769e+01  1.54239867e+01
  1.62267448e+01  1.61759141e+01  1.53120590e+01  1.50251290e+01
  1.52606486e+01  1.52664294e+01  1.49840567e+01  1.44696773e+01
  1.38233630e+01  1.28422893e+01  1.25184260e+01  1.17358216e+01
  1.19101959e+01  1.23106556e+01  1.21092210e+01  1.17855305e+01
  1.12374285e+01  1.16051388e+01  1.17754669e+01  1.17806644e+01
  1.16999490e+01  1.18619186e+01  1.18175133e+01  1.12382723e+01
  1.10167650e+01  1.10437548e+01  1.10185138e+01  1.03427152e+01
  9.68238020e+00  1.01146906e+01  9.64125917e+00  9.23465976e+00
  8.63930467e+00  8.01285300e+00  7.04638777e+00  6.87130337e+00
  6.48443213e+00  6.66789229e+00  6.55411709e+00  5.48202122e+00
  4.45813216e+00  3.72452112e+00  3.86501886e+00  3.75210766e+00
  3.91794317e+00  4.06732890e+00  4.43315742e+00  3.64145936e+00
  3.61530532e+00  3.38862215e+00  3.07557743e+00  3.45937929e+00
  2.79466964e+00  2.42139711e+00  2.17268895e+00  1.80951008e+00
  1.15378062e+00  1.51863611e+00  8.41963038e-01  1.09359522e+00
  9.79811910e-01  9.00055780e-01  9.98227468e-01  5.66107276e-01
  6.66392939e-01  3.13084009e-01 -3.09171145e-01 -1.20957948e+00
 -1.19960725e+00 -2.20006300e+00 -2.39946061e+00 -3.14174032e+00
 -3.66196255e+00 -3.61332496e+00 -4.47482642e+00 -4.80279762e+00
 -4.36874209e+00 -4.63216576e+00 -4.79225061e+00 -4.83657098e+00
 -4.46922090e+00 -4.63105977e+00 -4.56406368e+00 -4.95953400e+00
 -5.19829552e+00 -5.39370540e+00 -5.98923592e+00 -5.24414317e+00
 -5.06444726e+00 -4.50647754e+00 -4.36856097e+00 -4.72041111e+00
 -4.76308352e+00 -4.38153249e+00 -3.57427132e+00 -3.77018007e+00
 -3.87513658e+00 -4.69562938e+00 -5.41508144e+00 -5.19500287e+00
 -5.37986258e+00 -4.61950631e+00 -4.82171843e+00 -4.74197699e+00
 -5.55267046e+00 -5.26250857e+00 -4.41986164e+00 -3.52847560e+00
 -3.08249105e+00 -2.91723710e+00 -3.21341249e+00 -2.89852083e+00
 -2.07960999e+00 -1.69917401e+00 -9.56822912e-01 -3.84262799e-01
```

```
-8.76978080e-01 -7.05109608e-01 -2.18487932e-01 -9.52647708e-01
-6.15884075e-01 -1.18899040e+00 -9.13852269e-01 -9.93614895e-01
-8.79456513e-01 -8.61155682e-01 -5.26823546e-01 -9.74702855e-01
-5.01162377e-01 -5.03821893e-02  3.92471533e-01  7.51739973e-01
 1.54891976e-01  8.29611360e-01  1.59951184e+00  2.21843002e+00
 3.05684697e+00  3.99703361e+00  3.81339506e+00  4.01197570e+00
 4.21691488e+00  4.53523386e+00  4.88135742e+00  4.38484464e+00
 4.45023406e+00  3.81369579e+00  3.99043559e+00  3.67558646e+00
 3.73344379e+00  4.05234730e+00  4.62617093e+00  4.94663116e+00
 5.54001703e+00  6.34699821e+00  6.78240108e+00  7.79557392e+00
 7.58969344e+00  7.86512777e+00  8.12811691e+00  8.62320478e+00
 8.49204338e+00  8.97698587e+00  9.50902718e+00  9.91144781e+00
 9.59318330e+00  9.49918227e+00  9.65722028e+00  1.05553526e+01
 1.07915129e+01  1.14149831e+01  1.21527921e+01  1.26882326e+01
 1.24712778e+01  1.23488956e+01  1.28447360e+01  1.29215307e+01
 1.32870909e+01  1.28632433e+01  1.34255143e+01  1.29238551e+01
 1.30869951e+01  1.25197571e+01  1.29034249e+01  1.29515966e+01
 1.30475205e+01  1.38308138e+01  1.41928189e+01  1.41089344e+01
 1.41477312e+01  1.46717615e+01  1.44570866e+01  1.51007355e+01
 1.48889401e+01  1.47306357e+01  1.45555959e+01  1.45525683e+01
 1.40843015e+01  1.49332778e+01  1.50172377e+01  1.49249194e+01
 1.52218399e+01  1.49404255e+01  1.53187468e+01  1.55111042e+01
 1.59351853e+01  1.55721228e+01  1.59786334e+01  1.52871951e+01
 1.53759126e+01  1.54218491e+01  1.46377884e+01  1.41889975e+01
 1.41019581e+01  1.44944558e+01  1.38726919e+01  1.35897513e+01
 1.35560923e+01  1.34861652e+01  1.34743571e+01  1.31998736e+01
 1.35593888e+01  1.40796220e+01  1.38640679e+01  1.28232461e+01
 1.27467908e+01  1.23011246e+01  1.12473471e+01  1.09499919e+01
 9.98750206e+00  9.98662007e+00  8.92038932e+00  8.68743605e+00
 7.94014417e+00  7.84653842e+00  6.93959613e+00  6.42883816e+00
 6.46693865e+00  6.66915824e+00  6.20475507e+00  5.99369113e+00
 6.52085804e+00  5.70563585e+00  5.36548748e+00  4.92927316e+00
 5.21469482e+00  4.74966815e+00  4.05130017e+00  3.23855292e+00
 2.89553166e+00  1.89176925e+00  9.34424636e-01  4.26126272e-01
-7.48534117e-02 -5.98600895e-01 -1.39802736e+00 -1.11017773e+00
-4.45899941e-01 -1.75248836e-01 -1.21405722e-01 -4.93484556e-01
-3.54651972e-01 -3.91706953e-01  1.97348156e-01 -1.35478342e-02
 3.95434493e-01 -1.87763551e-01 -1.16781610e+00 -1.73672618e+00
-2.13440354e+00 -1.85957048e+00 -2.20225546e+00 -2.48264534e+00
-3.12427125e+00 -3.10195410e+00 -3.37037340e+00 -3.42781477e+00
-3.48454626e+00 -3.48603747e+00 -2.96639699e+00 -3.73697455e+00
-4.16369560e+00 -4.37247833e+00 -4.23645489e+00 -3.65318972e+00
-3.49677329e+00 -3.10336363e+00 -3.25464058e+00 -3.16909359e+00
-3.91289070e+00 -3.91161765e+00 -3.24223551e+00 -3.39654750e+00
-3.55118928e+00 -4.49796955e+00 -4.74605961e+00 -4.76928774e+00
-4.78357966e+00 -5.37688881e+00 -5.22256910e+00 -4.54047412e+00
-5.20467601e+00 -5.20761737e+00 -5.37984374e+00 -5.05592436e+00
-4.86015668e+00 -5.40745325e+00 -5.07152012e+00 -4.39281868e+00
-4.00886193e+00 -3.75714767e+00 -2.91270571e+00 -2.18306819e+00
-1.77352153e+00 -9.42380816e-01 -6.11310723e-01 -2.13331124e-01
-4.53091229e-01 -9.03450636e-01 -1.17818969e+00 -1.41381920e+00
-2.00150782e+00 -1.92008324e+00 -2.00216380e+00 -1.83367383e+00
-1.61306499e+00 -1.26488549e+00 -8.02163884e-01 -2.38563522e-01
 7.75142368e-01  1.03855847e+00  1.73811568e+00  1.68874291e+00
 2.04851591e+00  1.84574174e+00  1.42073289e+00  1.67727041e+00
 2.01327372e+00  2.09711774e+00  2.03066106e+00  2.68184990e+00
 2.77229608e+00  3.73520395e+00  4.45355875e+00  4.41278707e+00
 4.73073067e+00  4.98386873e+00  5.09083034e+00  5.37852302e+00
 5.90513051e+00  6.14545184e+00  6.74334579e+00  6.83704259e+00
```

```
 6.48498176e+00  7.08102916e+00  7.80407518e+00  8.53473046e+00
 9.26815299e+00  9.49327482e+00  9.07054946e+00  9.38346408e+00
 9.04773201e+00  8.98405182e+00  9.57896478e+00  1.04235421e+01
 1.00955620e+01  9.87538582e+00  9.60323592e+00  9.90421691e+00
 1.07308992e+01  1.05277019e+01  1.09507995e+01  1.09079972e+01
 1.11615041e+01  1.11340929e+01  1.19083150e+01  1.26351335e+01
 1.27893819e+01  1.32616111e+01  1.33990355e+01  1.38594330e+01
 1.47544929e+01  1.47888744e+01  1.50324718e+01  1.52660552e+01
 1.52098858e+01  1.46916715e+01  1.55670961e+01  1.50949497e+01
 1.50048434e+01  1.49946819e+01  1.47748115e+01  1.48880384e+01
 1.44487591e+01  1.40650272e+01  1.41883421e+01  1.47413380e+01
 1.45594966e+01  1.45760699e+01  1.50851708e+01  1.53977913e+01
 1.51576122e+01  1.59304486e+01  1.61758197e+01  1.66110169e+01
 1.67303319e+01  1.67149757e+01  1.69099518e+01  1.66973494e+01
 1.57672860e+01  1.51514219e+01  1.54588043e+01  1.46857595e+01
 1.41455831e+01  1.39409381e+01  1.33815767e+01  1.31372662e+01
 1.21859543e+01  1.26116078e+01  1.26701956e+01  1.21433742e+01
 1.16664629e+01  1.13857611e+01  1.10302225e+01  9.96279249e+00
 9.33026166e+00  9.07700274e+00  9.39788838e+00  9.37066925e+00
 9.33368837e+00  9.51197240e+00  8.84118736e+00  9.30865859e+00
 9.34409522e+00  9.46390169e+00  9.48669597e+00  9.26093391e+00
 8.39526232e+00  7.52214838e+00  7.22458070e+00  6.45152123e+00
 6.44530725e+00  5.61458963e+00  5.22492015e+00  4.68043292e+00
 4.50398594e+00  3.57951766e+00  3.54866462e+00  3.24590245e+00
 3.22456707e+00  2.98439702e+00  2.16493460e+00  1.72335267e+00
 1.72504556e+00  1.84276917e+00  1.42506712e+00  1.75886150e+00
 1.35986224e+00  1.45289335e+00  8.32845446e-01  1.22936761e+00
 1.83002106e+00  2.10225385e+00  2.00648949e+00  1.78139752e+00
 1.46087113e+00  3.56162431e-01 -1.80309515e-01 -9.00884133e-01
-1.34087179e+00 -1.86000102e+00 -2.28437487e+00 -2.42960112e+00
-3.28374655e+00 -4.07442105e+00 -4.01198407e+00 -3.59817132e+00
-2.98851537e+00 -2.46222855e+00 -2.07034703e+00 -2.34326737e+00
-2.87805526e+00 -3.82220062e+00 -3.37884190e+00 -2.94425881e+00
-2.77787151e+00 -3.52269341e+00 -3.96035453e+00 -4.39291202e+00
-5.20303146e+00 -5.49243354e+00 -5.06213137e+00 -5.19061800e+00
-5.43437510e+00 -5.99494826e+00 -6.40977955e+00 -5.52707521e+00
-5.59131832e+00 -5.29563013e+00 -5.39042804e+00 -5.24898849e+00
-5.65583310e+00 -5.25593009e+00 -4.98691728e+00 -4.68022581e+00
-4.33623006e+00 -4.58032367e+00 -4.49888830e+00 -4.80907836e+00
-4.18761099e+00 -4.25560776e+00 -3.69495424e+00 -3.29061947e+00
-3.07639724e+00 -2.71654881e+00 -3.14398014e+00 -3.19423168e+00
-2.99625967e+00 -2.90754665e+00 -3.28522009e+00 -2.40383451e+00
-2.89522505e+00 -2.71426644e+00 -2.50270496e+00 -2.16377199e+00
-1.06789189e+00 -1.11651307e+00 -3.95951681e-01 -4.23640461e-01
 3.50625506e-01  2.91784015e-01  9.84951675e-01  1.02683569e+00
 8.01322017e-01  4.76556766e-01  3.29936687e-01  1.36697143e+00
 1.17346628e+00  1.67017944e+00  1.52112863e+00  1.50952309e+00
 1.51575267e+00  1.56106872e+00  1.94702191e+00  2.30630003e+00
 2.16646963e+00  2.49283788e+00  2.59445673e+00  3.24352688e+00
 4.24793082e+00  4.92943133e+00  5.17042076e+00  5.57733000e+00
 6.05060854e+00  6.81573274e+00  7.40001423e+00  7.62277368e+00
 8.50604743e+00  9.16447501e+00  8.70710892e+00  9.42948292e+00
 9.94745108e+00  1.09003966e+01  1.14274966e+01  1.21507752e+01
 1.28193852e+01  1.26555665e+01  1.26528230e+01  1.23430700e+01
 1.29137157e+01  1.25070006e+01  1.28316338e+01  1.28491748e+01
 1.26426342e+01  1.23790954e+01  1.19289528e+01  1.19081532e+01
 1.16297549e+01  1.23917407e+01  1.21899525e+01  1.22583537e+01
 1.25172897e+01  1.20501324e+01  1.27789439e+01  1.30893590e+01
 1.40443492e+01  1.44888213e+01  1.48297063e+01  1.44862827e+01
```

```
 1.54814914e+01  1.58822986e+01  1.55855824e+01  1.64208244e+01
 1.59402903e+01  1.52416986e+01  1.45328211e+01  1.43434191e+01
 1.48919654e+01  1.47813228e+01  1.42656478e+01  1.40417758e+01
 1.46063034e+01  1.46153056e+01  1.51670269e+01  1.54645775e+01
 1.60957523e+01  1.64727165e+01  1.61836476e+01  1.58548592e+01
 1.55024740e+01  1.51992064e+01  1.50992454e+01  1.42975695e+01
 1.35995102e+01  1.35490832e+01  1.31852489e+01  1.30537046e+01
 1.23253285e+01  1.23079133e+01  1.27274069e+01  1.31652189e+01
 1.21236443e+01  1.17578794e+01  1.13742974e+01  1.07146289e+01
 1.01139054e+01  9.12417186e+00  9.29454894e+00  8.85003754e+00
 8.33431628e+00  7.35963949e+00  7.95434134e+00  8.44742692e+00
 8.24788826e+00  8.12380961e+00  7.65517246e+00  7.73339226e+00
 6.90472141e+00  6.99483000e+00  6.50136581e+00  6.45391958e+00
 5.97838368e+00  5.44445608e+00  5.69551255e+00  5.49301864e+00
 5.82744864e+00  5.80115336e+00  5.80587066e+00  5.47557944e+00
 5.04436608e+00  5.23095756e+00  4.22500291e+00  3.42107465e+00
 2.34444010e+00  2.09249028e+00  1.63973442e+00  1.09911865e+00
 9.34997959e-01  2.29803129e-02 -1.27324827e-01 -1.22126324e+00
-1.40420556e+00 -9.58415120e-01 -1.19534964e+00 -1.00420198e+00
-1.20530939e+00 -1.82618688e+00 -1.95536620e+00 -2.08131995e+00
-2.01351996e+00 -1.66768309e+00 -1.28522953e+00 -1.95241244e+00
-1.45985149e+00 -2.46827034e+00 -3.17003472e+00 -2.79802810e+00
-3.16148928e+00 -3.27184335e+00 -4.05178400e+00 -4.02263293e+00
-4.50346803e+00 -4.23378202e+00 -4.53961121e+00 -4.29440869e+00
-4.31976681e+00 -4.61586288e+00 -4.23219978e+00 -3.74567990e+00
-3.91598792e+00 -4.09779497e+00 -3.70037403e+00 -3.91141481e+00
-4.49714173e+00 -4.44299501e+00 -4.68601746e+00 -4.84266053e+00
-4.78477599e+00 -4.57910406e+00 -4.27197660e+00 -4.58729054e+00
-4.85764159e+00 -4.54627427e+00 -4.50722847e+00 -4.65602261e+00
-3.83549416e+00 -3.60591539e+00 -3.53809077e+00 -4.16070344e+00
-3.78661647e+00 -3.66676477e+00 -3.80839556e+00 -4.17399013e+00
-3.72945077e+00 -3.46137698e+00 -3.24410461e+00 -2.82035292e+00
-2.80000744e+00 -2.49618173e+00 -2.24423696e+00 -1.57715364e+00
-8.76874085e-01 -7.51148953e-01 -1.78667198e-01  4.40949473e-01
-8.52998585e-02  3.02017706e-01  2.74559999e-01  7.35220973e-01
 8.75180165e-01  9.38015866e-01  1.04236300e+00  1.47910438e+00
 1.86215707e+00  2.04489611e+00  2.59918993e+00  2.44749225e+00
 3.11745546e+00  3.28227653e+00  3.35069618e+00  3.14956053e+00
 3.71343512e+00  4.44420387e+00  4.00901592e+00  4.02800646e+00
 4.11466513e+00  4.12674077e+00  4.52848344e+00  4.54942031e+00
 5.62993159e+00  5.97740113e+00  6.27386192e+00  6.22914874e+00
 6.78228008e+00  7.09318829e+00  7.97562642e+00  8.47185728e+00
 8.52368182e+00  9.27502882e+00  9.06664570e+00  1.01544493e+01
 1.01243920e+01  1.06589166e+01  1.12858721e+01  1.12462509e+01
 1.15207646e+01  1.16761779e+01  1.21692236e+01  1.19358913e+01
 1.20186842e+01  1.14453266e+01  1.15875024e+01  1.17948367e+01
 1.15743545e+01  1.18861623e+01  1.16350788e+01  1.24159885e+01
 1.18290812e+01  1.21811630e+01  1.20762062e+01  1.28329091e+01
 1.28945927e+01  1.25186234e+01  1.30633176e+01  1.33635350e+01
 1.38670198e+01  1.39650209e+01  1.41295874e+01  1.49538510e+01
 1.51156072e+01  1.47593639e+01  1.45928694e+01  1.46667480e+01
 1.44576260e+01  1.46634696e+01  1.47991318e+01  1.48733803e+01
 1.48659939e+01  1.44954240e+01  1.44151185e+01  1.45440346e+01
 1.46376294e+01  1.45499987e+01  1.44842718e+01  1.46141162e+01
 1.37424052e+01  1.36354488e+01  1.40441781e+01  1.34826088e+01
 1.41860184e+01  1.41001476e+01  1.46355832e+01  1.43656161e+01
 1.42586768e+01  1.36514537e+01  1.37105862e+01  1.32385785e+01
 1.32337242e+01  1.36630585e+01  1.32409350e+01  1.30796231e+01
 1.27808813e+01  1.28479116e+01  1.29521491e+01  1.32548130e+01
```

```
  1.29172599e+01  1.28443037e+01  1.16759880e+01  1.09293897e+01
  1.01657163e+01  9.83923631e+00  9.37502271e+00  8.69724540e+00
  8.05119851e+00  7.86473072e+00  7.58516956e+00  7.19352921e+00
  7.47379227e+00  6.85798685e+00  6.79891156e+00  6.69349184e+00
  5.69159711e+00  6.16304036e+00  5.85671917e+00  4.63671664e+00
  4.31976915e+00  4.28343478e+00  3.96015969e+00  3.74531701e+00
  3.24555491e+00  2.68308628e+00  2.35104392e+00  1.57581079e+00
  1.23985398e+00  1.14983123e+00  1.21361056e+00  7.15618944e-01
  7.47610756e-02  2.96890234e-01  3.61831067e-01  5.70575470e-01
  1.04913331e+00  1.21169393e+00  9.43355769e-01  6.20862506e-01
  7.69270984e-01 -1.07729656e-01 -1.19321514e-01 -9.12797969e-01
 -8.43255802e-01 -1.05494280e+00 -1.33506164e+00 -2.04758759e+00
 -2.49052903e+00 -2.61000446e+00 -3.67028530e+00 -3.45773185e+00
 -3.96323032e+00 -3.88223631e+00 -4.35864393e+00 -4.74675102e+00
 -5.58390690e+00 -5.63699687e+00 -5.36350991e+00 -4.89074200e+00
 -4.64869181e+00 -4.30197671e+00 -4.26544908e+00 -4.64647838e+00
 -4.85768224e+00 -5.38773784e+00 -4.63723636e+00 -4.87379255e+00
 -5.24473149e+00 -5.32639524e+00 -5.09640422e+00 -5.50755885e+00
 -5.05841589e+00 -4.63364253e+00 -5.08264178e+00 -5.28818757e+00
 -5.55173997e+00 -5.64259587e+00 -5.83389937e+00 -5.58180282e+00
 -5.35065578e+00 -5.65999798e+00 -6.25830125e+00 -5.62009558e+00
 -5.22282510e+00 -4.91506951e+00 -4.59274367e+00 -4.41640515e+00
 -4.04356779e+00 -3.86634620e+00 -3.91002764e+00 -3.35747733e+00
 -2.76846336e+00 -2.73620004e+00 -2.66666392e+00 -1.67105020e+00
 -1.28020356e+00 -5.43378683e-01 -5.30435643e-01 -5.63605019e-01
 -4.25684997e-02  2.14447058e-01  4.29521079e-01  3.94614439e-02
  9.22704509e-01  1.11455769e+00  6.32454685e-01  1.05687453e+00
  1.31816573e+00  1.87526318e+00  1.87140922e+00  2.19682724e+00
  2.43899831e+00  3.55105221e+00  3.36739077e+00  3.61784269e+00]
[2.48753144e-01 8.56321237e-01 1.51115369e+00 2.17751391e+00
 2.52592877e+00 3.63511225e+00 4.26961894e+00 5.33417086e+00
 6.28234002e+00 6.85876808e+00 7.98158289e+00 8.71682655e+00
 9.96605430e+00 1.09444751e+01 1.22519897e+01 1.33985118e+01
 1.46055198e+01 1.56158856e+01 1.71077960e+01 1.83138696e+01
 1.93255099e+01 2.05448101e+01 2.12014273e+01 2.21643140e+01
 2.35090346e+01 2.45062190e+01 2.58532031e+01 2.70320829e+01
 2.79380702e+01 2.90265075e+01 3.04054894e+01 3.18234427e+01
 3.32420959e+01 3.47560388e+01 3.62844487e+01 3.76068737e+01
 3.94082473e+01 4.06935638e+01 4.20592052e+01 4.38920094e+01
 4.56492599e+01 4.73134421e+01 4.83822522e+01 5.02776516e+01
 5.19741801e+01 5.39607155e+01 5.51022785e+01 5.69659665e+01
 5.81264861e+01 5.97415523e+01 6.08653723e+01 6.27128870e+01
 6.45182327e+01 6.60828954e+01 6.74821934e+01 6.85390488e+01
 7.02187648e+01 7.16482242e+01 7.33388630e+01 7.51655390e+01
 7.70921170e+01 7.88888011e+01 7.98302917e+01 8.11080244e+01
 8.27428420e+01 8.38054782e+01 8.52028215e+01 8.61179015e+01
 8.71622260e+01 8.80078283e+01 8.96105430e+01 9.06246227e+01
 9.17404877e+01 9.34186800e+01 9.48520631e+01 9.55910088e+01
 9.64402500e+01 9.77230403e+01 9.89376929e+01 9.97884927e+01
 1.01310492e+02 1.02486541e+02 1.03558001e+02 1.04656952e+02
 1.05868828e+02 1.06634764e+02 1.07458764e+02 1.08065756e+02
 1.08620073e+02 1.09903183e+02 1.10951751e+02 1.11721201e+02
 1.12197306e+02 1.12669805e+02 1.12915216e+02 1.13506067e+02
 1.13943196e+02 1.14733648e+02 1.15174190e+02 1.15385205e+02
 1.15409883e+02 1.15445722e+02 1.16062324e+02 1.16421913e+02
 1.16833159e+02 1.17573396e+02 1.18376353e+02 1.18375107e+02
 1.18789495e+02 1.18773827e+02 1.18485461e+02 1.18905101e+02
 1.18856994e+02 1.18843310e+02 1.19005848e+02 1.19382906e+02
 1.19530134e+02 1.19893743e+02 1.19631459e+02 1.19867422e+02
```

```
1.19465273e+02 1.19805157e+02 1.19855222e+02 1.19409417e+02
1.19672241e+02 1.19695990e+02 1.19220963e+02 1.18684164e+02
1.18431851e+02 1.17667359e+02 1.17065812e+02 1.16663417e+02
1.16193259e+02 1.15796092e+02 1.15197414e+02 1.14893192e+02
1.14852221e+02 1.14051998e+02 1.13639601e+02 1.12830788e+02
1.12596591e+02 1.12032357e+02 1.11629195e+02 1.10836558e+02
1.09999119e+02 1.09499487e+02 1.08862985e+02 1.08807855e+02
1.08575153e+02 1.08324310e+02 1.08228030e+02 1.07311946e+02
1.06866112e+02 1.06455026e+02 1.06424848e+02 1.05729307e+02
1.04987848e+02 1.04112225e+02 1.03160072e+02 1.03129308e+02
1.02848168e+02 1.02692440e+02 1.02044393e+02 1.01713049e+02
1.00872177e+02 1.00466798e+02 1.00567987e+02 1.00583750e+02
1.00077581e+02 1.00212070e+02 9.96347551e+01 9.97939189e+01
9.99647834e+01 1.00013875e+02 9.99153542e+01 1.00082535e+02
9.96910086e+01 9.98786403e+01 9.98590930e+01 9.92594228e+01
9.90188711e+01 9.86049285e+01 9.90509311e+01 9.90202600e+01
9.90358977e+01 9.92213797e+01 9.91641850e+01 9.89039374e+01
9.93579306e+01 9.92090406e+01 9.94113426e+01 9.93566685e+01
9.92058231e+01 9.98498714e+01 1.00635410e+02 1.01439810e+02
1.02221032e+02 1.02900971e+02 1.03171326e+02 1.03221016e+02
1.03628257e+02 1.03891902e+02 1.04087181e+02 1.04234716e+02
1.05085644e+02 1.05253506e+02 1.06211468e+02 1.06576557e+02
1.06904769e+02 1.07273364e+02 1.08254428e+02 1.08838534e+02
1.09627198e+02 1.10581714e+02 1.11868045e+02 1.13049079e+02
1.13801161e+02 1.14441685e+02 1.15032886e+02 1.15896568e+02
1.16746472e+02 1.17815519e+02 1.19136225e+02 1.20493162e+02
1.21461228e+02 1.22548262e+02 1.23458381e+02 1.24997038e+02
1.25824399e+02 1.27311551e+02 1.28899264e+02 1.30503752e+02
1.31607502e+02 1.32842058e+02 1.34305964e+02 1.35469792e+02
1.36745472e+02 1.37860281e+02 1.39249914e+02 1.40235407e+02
1.41986259e+02 1.43023509e+02 1.44510927e+02 1.45793654e+02
1.47353485e+02 1.49300606e+02 1.50938291e+02 1.51969216e+02
1.53397645e+02 1.54907168e+02 1.56443346e+02 1.58124245e+02
1.59399868e+02 1.60524290e+02 1.61909080e+02 1.63853174e+02
1.65022593e+02 1.66902493e+02 1.68414882e+02 1.69832088e+02
1.71665185e+02 1.73064670e+02 1.74718614e+02 1.76035394e+02
1.77844266e+02 1.79425297e+02 1.81001226e+02 1.82189688e+02
1.83790795e+02 1.85253937e+02 1.86302974e+02 1.87253668e+02
1.88820572e+02 1.90529850e+02 1.91716958e+02 1.93015049e+02
1.94557318e+02 1.95675854e+02 1.97265152e+02 1.98453810e+02
1.99862363e+02 2.01333290e+02 2.02684640e+02 2.03353096e+02
2.04463748e+02 2.05316173e+02 2.05804665e+02 2.06625845e+02
2.07252654e+02 2.08440430e+02 2.08782752e+02 2.10020726e+02
2.10624785e+02 2.11199635e+02 2.11403344e+02 2.11745011e+02
2.12271604e+02 2.13295004e+02 2.13457409e+02 2.14434121e+02
2.15303610e+02 2.15726362e+02 2.15990272e+02 2.16128908e+02
2.16618039e+02 2.16494679e+02 2.16322904e+02 2.16533557e+02
2.16352941e+02 2.16325891e+02 2.16238035e+02 2.16152488e+02
2.15915419e+02 2.15530307e+02 2.15220012e+02 2.15384502e+02
2.15877004e+02 2.16358308e+02 2.16231535e+02 2.15832406e+02
2.15883383e+02 2.15760781e+02 2.16112767e+02 2.15516759e+02
2.15615446e+02 2.15196738e+02 2.14709188e+02 2.14621582e+02
2.14097132e+02 2.13972836e+02 2.13681127e+02 2.13278136e+02
2.12988496e+02 2.12414805e+02 2.12245073e+02 2.11768923e+02
2.11224642e+02 2.11135544e+02 2.11130735e+02 2.10235861e+02
2.09517432e+02 2.08905657e+02 2.08752041e+02 2.08761615e+02
2.08748300e+02 2.08665560e+02 2.07970001e+02 2.07966451e+02
2.07217844e+02 2.06324243e+02 2.06275196e+02 2.05509110e+02
2.05200851e+02 2.04263646e+02 2.04002240e+02 2.03896272e+02
```

```
2.03186422e+02 2.02589562e+02 2.01995275e+02 2.01783769e+02
2.01070520e+02 2.00301492e+02 1.99821008e+02 1.99207721e+02
1.99142083e+02 1.98488819e+02 1.98114902e+02 1.98196743e+02
1.97986413e+02 1.98026622e+02 1.98157814e+02 1.98118424e+02
1.98047486e+02 1.98265341e+02 1.98530773e+02 1.98275488e+02
1.97661810e+02 1.97293292e+02 1.96808223e+02 1.96612802e+02
1.96156307e+02 1.96198341e+02 1.96045322e+02 1.96431667e+02
1.96917708e+02 1.97010602e+02 1.96859646e+02 1.97054729e+02
1.97583366e+02 1.97651361e+02 1.97894422e+02 1.97887084e+02
1.98093838e+02 1.98277408e+02 1.98338441e+02 1.98687873e+02
1.98872920e+02 1.99151847e+02 1.99614027e+02 2.00333211e+02
2.00666718e+02 2.01622288e+02 2.02547397e+02 2.02690196e+02
2.03069171e+02 2.03671741e+02 2.03963750e+02 2.04530370e+02
2.05519157e+02 2.06478663e+02 2.07410064e+02 2.08459330e+02
2.09032379e+02 2.09771225e+02 2.10873246e+02 2.12206472e+02
2.13231903e+02 2.14023645e+02 2.14589707e+02 2.15862127e+02
2.16457796e+02 2.17443382e+02 2.18611343e+02 2.20194767e+02
2.20968808e+02 2.22081858e+02 2.22835139e+02 2.23927861e+02
2.25320606e+02 2.26389829e+02 2.27408596e+02 2.28351379e+02
2.29772848e+02 2.31328860e+02 2.32877123e+02 2.34716991e+02
2.35624521e+02 2.37189473e+02 2.38719641e+02 2.40249262e+02
2.42163088e+02 2.43140254e+02 2.44805319e+02 2.46594915e+02
2.48087009e+02 2.49408663e+02 2.51191617e+02 2.52284422e+02
2.53724485e+02 2.55243944e+02 2.56937900e+02 2.58028292e+02
2.59254078e+02 2.60659942e+02 2.62275351e+02 2.64150001e+02
2.65751114e+02 2.66860492e+02 2.68809655e+02 2.70641735e+02
2.72095512e+02 2.73958741e+02 2.75429898e+02 2.77270959e+02
2.79005683e+02 2.80864976e+02 2.82661066e+02 2.83557842e+02
2.84576942e+02 2.85793157e+02 2.87554316e+02 2.88644500e+02
2.89575481e+02 2.91211897e+02 2.92387260e+02 2.94002242e+02
2.94847020e+02 2.96169449e+02 2.97247137e+02 2.97936531e+02
2.99220779e+02 3.00030261e+02 3.00605704e+02 3.01174690e+02
3.01717522e+02 3.03079245e+02 3.04244908e+02 3.05540119e+02
3.06580825e+02 3.07448503e+02 3.08061967e+02 3.09338920e+02
3.09949799e+02 3.10638591e+02 3.11204218e+02 3.12340179e+02
3.12640171e+02 3.13062267e+02 3.13805406e+02 3.13900025e+02
3.14507274e+02 3.14953510e+02 3.15174719e+02 3.15319024e+02
3.15708204e+02 3.15919697e+02 3.16188835e+02 3.16308169e+02
3.17029973e+02 3.16884422e+02 3.16672209e+02 3.16676862e+02
3.16899765e+02 3.17161793e+02 3.17133271e+02 3.17678558e+02
3.17548698e+02 3.17761063e+02 3.17862819e+02 3.18113789e+02
3.18502230e+02 3.18779116e+02 3.18906254e+02 3.18943191e+02
3.18594142e+02 3.18034721e+02 3.17368388e+02 3.16860179e+02
3.16521947e+02 3.16253788e+02 3.16217855e+02 3.16349515e+02
3.15622508e+02 3.14868770e+02 3.14582158e+02 3.14436549e+02
3.14379873e+02 3.14397950e+02 3.14451600e+02 3.13910521e+02
3.13339799e+02 3.12527314e+02 3.12243666e+02 3.11924511e+02
3.11804286e+02 3.10913856e+02 3.10419518e+02 3.10005038e+02
3.09248568e+02 3.08418087e+02 3.08277668e+02 3.07336696e+02
3.06809291e+02 3.05929563e+02 3.05394507e+02 3.05386781e+02
3.04828200e+02 3.04709408e+02 3.03858140e+02 3.03169099e+02
3.02621835e+02 3.02080766e+02 3.01822373e+02 3.01249337e+02
3.01058277e+02 3.00806457e+02 3.00329312e+02 2.99900330e+02
2.99670529e+02 2.98913491e+02 2.98926881e+02 2.98790147e+02
2.98745976e+02 2.98532788e+02 2.97914297e+02 2.97612225e+02
2.97333052e+02 2.96992783e+02 2.96385309e+02 2.96509657e+02
2.96031656e+02 2.96075880e+02 2.96243271e+02 2.96369016e+02
2.96846405e+02 2.96495712e+02 2.96937100e+02 2.96569143e+02
2.96735935e+02 2.96801441e+02 2.97016607e+02 2.97102716e+02
```

```
2.97044593e+02 2.96845573e+02 2.97176341e+02 2.97862684e+02
2.98110566e+02 2.98239322e+02 2.98257063e+02 2.98310964e+02
2.98532360e+02 2.98663785e+02 2.98991615e+02 2.99151873e+02
2.99342811e+02 3.00355522e+02 3.00705023e+02 3.01482849e+02
3.02504994e+02 3.03240395e+02 3.03702781e+02 3.04241115e+02
3.05042224e+02 3.05967606e+02 3.06742825e+02 3.07978295e+02
3.09211071e+02 3.10647324e+02 3.11212103e+02 3.12669878e+02
3.13650232e+02 3.15141511e+02 3.16469720e+02 3.18118381e+02
3.19562210e+02 3.20633862e+02 3.21863894e+02 3.22990394e+02
3.24125819e+02 3.25176878e+02 3.26481866e+02 3.27990686e+02
3.29112354e+02 3.30497476e+02 3.31491163e+02 3.32542015e+02
3.33493648e+02 3.35382135e+02 3.36315771e+02 3.37435232e+02
3.38999155e+02 3.40040819e+02 3.41891298e+02 3.43586835e+02
3.45535512e+02 3.47030836e+02 3.48323355e+02 3.49868417e+02
3.51797263e+02 3.53317531e+02 3.54584738e+02 3.56461643e+02
3.57831589e+02 3.58828534e+02 3.60068334e+02 3.61374255e+02
3.63215320e+02 3.64649740e+02 3.66062911e+02 3.67359307e+02
3.69191041e+02 3.71076949e+02 3.72998616e+02 3.74293111e+02
3.76164086e+02 3.77846972e+02 3.79398968e+02 3.80504599e+02
3.81565385e+02 3.82558513e+02 3.84290287e+02 3.85374518e+02
3.86598126e+02 3.87842195e+02 3.89349335e+02 3.90900677e+02
3.91724296e+02 3.92812513e+02 3.94292791e+02 3.95723732e+02
3.96413931e+02 3.97132398e+02 3.97972423e+02 3.98556824e+02
3.99463240e+02 4.00024848e+02 4.01018845e+02 4.01662550e+02
4.02627108e+02 4.03083371e+02 4.04368272e+02 4.05579824e+02
4.06220311e+02 4.06680633e+02 4.07118413e+02 4.07758241e+02
4.07923567e+02 4.08657380e+02 4.09128474e+02 4.09537291e+02
4.10346656e+02 4.11024281e+02 4.11915824e+02 4.12173652e+02
4.12945861e+02 4.13559394e+02 4.13729437e+02 4.14132960e+02
4.14172840e+02 4.14768248e+02 4.14571659e+02 4.14445355e+02
4.14260264e+02 4.14266142e+02 4.14585596e+02 4.14658513e+02
4.14664435e+02 4.14155940e+02 4.14045515e+02 4.13546985e+02
4.13167453e+02 4.13486940e+02 4.13064914e+02 4.13261940e+02
4.13380286e+02 4.12832326e+02 4.12709069e+02 4.12074620e+02
4.12031995e+02 4.11879302e+02 4.11882224e+02 4.11534528e+02
4.11605063e+02 4.10793670e+02 4.10210252e+02 4.10034298e+02
4.09547580e+02 4.08802777e+02 4.07980211e+02 4.07856669e+02
4.07378756e+02 4.07300746e+02 4.07065452e+02 4.06499261e+02
4.05890485e+02 4.05418435e+02 4.05315380e+02 4.05057097e+02
4.04064223e+02 4.03758874e+02 4.03678382e+02 4.03389331e+02
4.02568310e+02 4.02056266e+02 4.01204467e+02 4.00575774e+02
4.00530604e+02 4.00477993e+02 3.99792246e+02 3.99171584e+02
3.98820740e+02 3.98843057e+02 3.98061081e+02 3.97400243e+02
3.97368973e+02 3.96969859e+02 3.96992513e+02 3.96317289e+02
3.96005630e+02 3.95504819e+02 3.95012344e+02 3.94669066e+02
3.94331631e+02 3.93938866e+02 3.94124869e+02 3.94149506e+02
3.94192506e+02 3.93821107e+02 3.93761393e+02 3.93927665e+02
3.94135470e+02 3.93917917e+02 3.94152963e+02 3.94379816e+02
3.94039569e+02 3.94451524e+02 3.94467066e+02 3.94556328e+02
3.94636573e+02 3.94865681e+02 3.95177833e+02 3.95397022e+02
3.96015120e+02 3.96424712e+02 3.96638759e+02 3.96899016e+02
3.97584521e+02 3.97838605e+02 3.97987269e+02 3.98015242e+02
3.98891269e+02 3.99841226e+02 4.00024136e+02 4.00452718e+02
4.00753424e+02 4.01025757e+02 4.02113005e+02 4.02388025e+02
4.03617201e+02 4.03992643e+02 4.05165130e+02 4.06070374e+02
4.06806416e+02 4.07545907e+02 4.08729050e+02 4.09497614e+02
4.10636687e+02 4.11663054e+02 4.12683847e+02 4.14147092e+02
4.15289522e+02 4.16729291e+02 4.18092289e+02 4.18792158e+02
4.20249815e+02 4.21173792e+02 4.22805910e+02 4.23598945e+02
```

```
4.24702531e+02 4.25592419e+02 4.26877025e+02 4.28524128e+02
4.29666643e+02 4.30678320e+02 4.31884894e+02 4.33589780e+02
4.34634991e+02 4.35780108e+02 4.36778737e+02 4.38425328e+02
4.39771617e+02 4.41042751e+02 4.42729961e+02 4.44041855e+02
4.45751913e+02 4.47554801e+02 4.48764579e+02 4.50733959e+02
4.51894344e+02 4.53184692e+02 4.54364487e+02 4.55709499e+02
4.57187587e+02 4.58705325e+02 4.60551045e+02 4.62428182e+02
4.63630573e+02 4.65229383e+02 4.66309463e+02 4.67728726e+02
4.69002116e+02 4.70259498e+02 4.71671858e+02 4.73319441e+02
4.74293450e+02 4.76063630e+02 4.77674751e+02 4.78711992e+02
4.80495481e+02 4.81828874e+02 4.83637699e+02 4.84625114e+02
4.85930535e+02 4.86970894e+02 4.88004037e+02 4.89302209e+02
4.90908475e+02 4.92375051e+02 4.93736416e+02 4.94908497e+02
4.96418581e+02 4.97473025e+02 4.98882684e+02 5.00225707e+02
5.00921297e+02 5.02146512e+02 5.02584463e+02 5.03304440e+02
5.03902132e+02 5.04747733e+02 5.05793603e+02 5.06170271e+02
5.06933883e+02 5.08090438e+02 5.08506466e+02 5.09340042e+02
5.10058255e+02 5.10162427e+02 5.10701044e+02 5.11441225e+02
5.11485200e+02 5.12333311e+02 5.12790602e+02 5.12727155e+02
5.12826235e+02 5.13623476e+02 5.14018415e+02 5.13907744e+02
5.13946599e+02 5.14124311e+02 5.13836244e+02 5.13909122e+02
5.14030456e+02 5.13876986e+02 5.14039846e+02 5.14339095e+02
5.14093176e+02 5.14204634e+02 5.14308430e+02 5.14694887e+02
5.14885378e+02 5.15120816e+02 5.14973812e+02 5.14497849e+02
5.14809117e+02 5.14231366e+02 5.13973854e+02 5.13291836e+02
5.13465174e+02 5.13639944e+02 5.13550316e+02 5.13073228e+02
5.12483283e+02 5.11887844e+02 5.11138831e+02 5.10773634e+02
5.10010624e+02 5.09409600e+02 5.09106530e+02 5.08893193e+02
5.07966409e+02 5.07436231e+02 5.07119773e+02 5.06997102e+02
5.06490140e+02 5.06471657e+02 5.05745175e+02 5.04763122e+02
5.04248848e+02 5.03505455e+02 5.03329173e+02 5.02562439e+02
5.01875041e+02 5.01670707e+02 5.01393735e+02 5.00964098e+02
5.00686759e+02 5.00129479e+02 4.99166206e+02 4.98217268e+02
4.97777433e+02 4.96919843e+02 4.96041142e+02 4.96088904e+02
4.96043080e+02 4.95304100e+02 4.94428458e+02 4.94509384e+02
4.93943381e+02 4.93302198e+02 4.93184689e+02 4.92503438e+02
4.91997574e+02 4.92222558e+02 4.92133052e+02 4.91946623e+02
4.91659995e+02 4.91773184e+02 4.91276717e+02 4.91631148e+02
4.91904486e+02 4.91960059e+02 4.91467139e+02 4.91658953e+02
4.92090484e+02 4.92161070e+02 4.92089516e+02 4.91812645e+02
4.92199422e+02 4.92745706e+02 4.92536940e+02 4.93016934e+02
4.92785304e+02 4.93534216e+02 4.93961893e+02 4.94357897e+02
4.94528514e+02 4.95363697e+02 4.95566812e+02 4.96363548e+02]
        [ 19.32550987  20.54481013  21.20142732  22.164314    23.50903465
 24.50621904  25.85320309  27.03208286  27.93807017  29.02650751
 30.40548941  31.82344275  33.24209592  34.75603877  36.28444872
 37.60687368  39.40824733  40.69356382  42.05920524  43.89200936
 45.64925986  47.31344213  48.38225218  50.27765156  51.97418011
 53.96071546  55.10227846  56.96596648  58.12648615  59.74155227
 60.86537225  62.71288704  64.51823273  66.08289543  67.48219343
 68.53904877  70.21876479  71.64822424  73.338863    75.16553901
 77.09211703  78.88880114  79.83029171  81.1080244   82.74284202
 83.80547822  85.2028215   86.11790152  87.16222597  88.00782834
 89.61054304  90.62462274  91.74048766  93.41868002  94.85206305
 95.59100876  96.44025003  97.72304029  98.93769287  99.78849275
101.310492   102.48654133 103.55800093 104.65695232 105.86882802
106.63476352 107.45876387 108.06575551 108.62007307 109.90318338
110.95175117 111.72120109 112.1973056  112.66980532 112.9152158
113.50606689 113.94319601 114.73364781 115.17419017 115.3852046
```

```
115.40988333 115.4457222  116.06232446 116.42191299 116.83315897
117.57339579 118.37635343 118.37510717 118.78949549 118.77382676
118.48546076 118.9051015  118.85699409 118.8433101  119.00584792
119.38290587 119.53013404 119.89374328 119.63145852 119.86742198
119.46527267 119.80515728 119.85522156 119.40941737 119.67224086
119.69598988 119.2209629  118.68416381 118.43185128 117.66735898
117.06581206 116.66341696 116.19325901 115.79609241 115.19741444
114.89319226 114.85222081 114.05199804 113.63960066 112.830788
112.59659117 112.03235718 111.62919534 110.83655841 109.99911892
109.49948686 108.86298489 108.80785488 108.5751534  108.32431046
108.2280302  107.31194607 106.86611182 106.45502592 106.4248476
105.72930678 104.98784831 104.11222549 103.16007196 103.12930759
102.84816762 102.69243977 102.04439339 101.71304893 100.87217714
100.46679821 100.56798667 100.58374989 100.07758092 100.2120705
 99.63475513  99.79391894  99.9647834  100.01387492  99.91535423
100.08253541  99.69100859  99.87864028  99.85909298  99.25942279
 99.01887105  98.60492854  99.05093113  99.02026002  99.03589771
 99.22137973  99.16418505  98.90393743  99.35793061  99.2090406
 99.41134259  99.35666851  99.20582311  99.84987138 100.63540956
101.43980975 102.22103202 102.90097104 103.17132567 103.2210163
103.62825747 103.89190237 104.08718053 104.23471602 105.08564362
105.25350554 106.21146761 106.57655749 106.90476946 107.2733636
108.2544284  108.83853353 109.62719755 110.58171424 111.8680447
113.04907946 113.80116104 114.44168526 115.03288637 115.89656839
116.74647179 117.81551941 119.13622473 120.49316204 121.461228
122.54826173 123.45838132 124.99703791 125.82439922 127.31155148
128.89926387 130.503752   131.60750249 132.84205766 134.30596397
135.4697924  136.74547225 137.86028116 139.24991353 140.2354066
141.986259   143.02350913 144.51092744 145.7936543  147.3534845
149.30060618 150.93829113 151.96921553 153.3976447  154.90716811
156.44334561 158.12424468 159.3998675  160.52429003 161.90908041
163.85317447 165.02259262 166.90249333 168.41488245 169.83208755
171.66518548 173.06467021 174.71861433 176.03539428 177.84426573
179.42529727 181.001226   182.18968843 183.790795   185.25393663
186.3029739  187.25366773 188.82057245 190.52985012 191.71695758
193.01504852 194.55731827 195.67585359 197.26515206 198.45381028
199.86236265 201.33328974 202.68464036 203.35309624 204.46374835
205.31617311 205.80466535 206.62584549 207.25265411 208.44043034
208.78275196 210.02072579 210.62478453 211.19963466 211.40334448
211.74501127 212.271604   213.29500373 213.45740918 214.43412147
215.30361    215.72636164 215.990272   216.12890782 216.61803929
216.49467942 216.32290417 216.53355665 216.35294084 216.32589072
216.23803464 216.15248791 215.91541859 215.53030693 215.22001193
215.38450169 215.87700423 216.35830782 216.23153512 215.83240616
215.88338267 215.76078096 216.11276675 215.51675909 215.61544642
215.19673814 214.70918813 214.62158163 214.09713158 213.97283568
213.6811272  213.27813562 212.9884955  212.41480499 212.24507302
211.76892337 211.22464187 211.13554416 211.13073459 210.23586113
209.51743161 208.90565729 208.75204061 208.76161528 208.74829973
208.66555973 207.9700013  207.96645058 207.21784389 206.32424348
206.27519609 205.50910979 205.20085133 204.26364572 204.00224012
203.89627199 203.18642164 202.58956177 201.99527479 201.78376936
201.07052009 200.30149242 199.82100758 199.20772136 199.14208343
198.48881874 198.11490151 198.19674309 197.98641286 198.02662169
198.15781438 198.11842423 198.04748605 198.26534055 198.53077271
198.27548761 197.66181028 197.29329245 196.80822317 196.61280249
196.15630656 196.19834099 196.04532225 196.43166672 196.91770772
197.01060212 196.8596464  197.05472893 197.58336553 197.65136096
197.89442224 197.8870839  198.09383817 198.27740846 198.33844061
```

```
198.68787253 198.87292012 199.15184667 199.6140266  200.33321086
200.66671832 201.62228785 202.54739692 202.69019553 203.06917128
203.67174126 203.96375046 204.5303697  205.51915711 206.4786627
207.41006412 208.45933044 209.03237868 209.77122469 210.87324646
212.20647172 213.23190345 214.02364452 214.58970657 215.86212678
216.45779613 217.44338226 218.61134346 220.19476679 220.96880844
222.08185754 222.83513937 223.92786142 225.32060572 226.38982869
227.40859563 228.35137944 229.77284761 231.32885964 232.87712346
234.71699101 235.62452127 237.18947255 238.71964127 240.24926165
242.16308849 243.14025385 244.80531944 246.59491485 248.0870093
249.40866251 251.19161737 252.28442226 253.72448463 255.24394357
256.93789998 258.02829227 259.2540785  260.65994203 262.2753514
264.15000052 265.75111394 266.86049217 268.80965547 270.64173489
272.09551217 273.95874092 275.4298982  277.27095889 279.00568331
280.86497624 282.66106574 283.55784158 284.57694151 285.79315676
287.55431645 288.64450038 289.57548127 291.21189701 292.38726005
294.00224243 294.84702    296.16944937 297.24713712 297.93653094
299.22077934 300.03026147 300.60570379 301.1746895  301.71752171
303.07924517 304.24490838 305.54011863 306.58082549 307.44850334
308.0619667  309.33892006 309.949799   310.63859119 311.20421768
312.34017908 312.6401707  313.06226701 313.8054062  313.90002457
314.50727395 314.95350969 315.17471915 315.31902412 315.70820363
315.91969674 316.18883532 316.30816946 317.02997326 316.88442159
316.67220855 316.67686236 316.89976471 317.16179329 317.13327074
317.67855824 317.54869756 317.76106281 317.86281871 318.1137892
318.50222961 318.77911621 318.90625419 318.94319081 318.59414187
318.03472067 317.36838804 316.86017868 316.52194692 316.25378818
316.21785474 316.34951509 315.62250764 314.86876975 314.5821578
314.43654935 314.37987268 314.39795013 314.45159989 313.91052081
313.33979948 312.52731447 312.24366575 311.92451094 311.80428629
310.91385594 310.41951815 310.00503811 309.24856842 308.41808727
308.27766811 307.33669647 306.80929064 305.92956268 305.39450674
305.38678073 304.82819983 304.70940798 303.85814039 303.16909878
302.621835   302.08076637 301.82237337 301.24933687 301.05827668
300.80645706 300.32931152 299.90032963 299.6705294  298.91349102
298.92688076 298.79014691 298.74597613 298.53278806 297.91429654
297.61222538 297.33305185 296.99278298 296.38530931 296.5096565
296.03165571 296.07588046 296.24327117 296.36901607 296.84640465
296.49571231 296.93710017 296.56914252 296.73593482 296.80144052
297.01660738 297.10271615 297.04459318 296.84557284 297.17634134
297.86268374 298.11056646 298.23932196 298.25706345 298.31096361
298.53236006 298.66378488 298.99161509 299.15187287 299.34281097
300.35552162 300.70502319 301.48284884 302.50499427 303.24039494
303.70278081 304.24111488 305.04222363 305.96760561 306.7428252
307.9782953  309.21107062 310.64732384 311.21210319 312.66987786
313.6502319  315.14151145 316.46972021 318.11838083 319.56221039
320.6338618  321.8638936  322.99039382 324.1258189  325.17687842
326.4818657  327.99068622 329.11235442 330.49747626 331.49116323
332.54201504 333.49364845 335.38213452 336.31577136 337.43523209
338.99915536 340.04081865 341.89129831 343.58683523 345.53551245
347.03083634 348.32335473 349.86841726 351.79726271 353.31753071
354.58473774 356.46164309 357.83158857 358.82853388 360.0683336
361.37425545 363.21532017 364.6497401  366.06291055 367.3593065
369.19104114 371.07694869 372.9986155  374.2931114  376.16408589
377.84697199 379.39896782 380.50459932 381.56538451 382.55851291
384.29028651 385.37451819 386.59812567 387.84219458 389.3493348
390.90067654 391.72429635 392.81251258 394.29279145 395.72373181
396.41393082 397.13239755 397.97242304 398.55682352 399.4632402
400.0248484  401.01884529 401.66255012 402.62710774 403.0833713
```

```
404.36827216 405.57982447 406.2203113  406.68063314 407.11841266
407.75824066 407.9235667  408.65738012 409.12847355 409.53729088
410.34665585 411.02428055 411.91582385 412.17365177 412.9458613
413.55939403 413.72943736 414.13295956 414.17283963 414.76824844
414.57165876 414.4453552  414.26026394 414.26614205 414.58559572
414.65851267 414.66443532 414.15593987 414.0455148  413.5469852
413.1674532  413.48694008 413.0649143  413.26194007 413.38028634
412.83232579 412.70906912 412.07461992 412.03199484 411.8793021
411.88222367 411.53452765 411.60506281 410.79366973 410.21025161
410.03429769 409.54757985 408.80277658 407.98021084 407.85666918
407.37875564 407.30074563 407.0654516  406.49926104 405.89048481
405.41843481 405.31538006 405.05709668 404.06422292 403.75887421
403.67838161 403.38933081 402.56830987 402.05626603 401.20446734
400.57577428 400.53060407 400.47799262 399.79224631 399.17158367
398.82074002 398.84305655 398.0610814  397.40024341 397.36897319
396.9698589  396.99251331 396.31728918 396.00562984 395.5048189
395.01234446 394.66906642 394.33163062 393.93886643 394.12486857
394.14950598 394.19250586 393.82110745 393.76139288 393.92766526
394.13547038 393.91791746 394.15296342 394.3798159  394.03956872
394.45152369 394.46706586 394.55632842 394.63657305 394.86568113
395.17783338 395.39702184 396.0151205  396.42471201 396.63875865
396.89901594 397.58452132 397.83860495 397.98726923 398.01524166
398.8912685  399.84122571 400.02413642 400.45271847 400.75342377
401.02575671 402.11300477 402.38802527 403.61720082 403.99264278
405.16513043 406.07037445 406.80641649 407.54590676 408.7290502
409.49761399 410.63668658 411.66305408 412.68384652 414.14709207
415.28952238 416.72929104 418.09228856 418.79215767 420.24981483
421.17379186 422.80591014 423.5989454  424.70253067 425.59241863
426.87702475 428.52412776 429.66664304 430.67831996 431.88489368
433.58978035 434.63499136 435.78010843 436.77873688 438.42532769
439.77161745 441.04275112 442.72996067 444.04185497 445.7519135
447.55480123 448.76457878 450.73395944 451.89434412 453.18469158
454.36448687 455.70949909 457.18758671 458.70532454 460.55104528
462.42818153 463.63057264 465.22938344 466.30946266 467.7287262
469.00211626 470.25949777 471.6718585  473.31944076 474.29345047
476.06363033 477.67475079 478.7119922  480.49548104 481.82887384
483.63769941 484.6251139  485.93053534 486.97089445 488.00403663
489.3022088  490.90847497 492.37505069 493.73641608 494.90849691
496.41858067 497.47302545 498.88268443 500.22570749 500.92129651
502.14651247 502.58446292 503.30444035 503.90213235 504.74773322
505.79360338 506.17027085 506.93388294 508.09043821 508.50646607
509.34004168 510.05825519 510.16242719 510.70104391 511.44122506
511.48520049 512.33331122 512.79060211 512.72715485 512.82623523
513.62347646 514.01841488 513.9077442  513.94659882 514.12431134
513.83624441 513.90912201 514.03045609 513.87698607 514.03984578
514.3390954  514.09317595 514.20463444 514.30842988 514.69488681
514.88537772 515.12081594 514.97381186 514.49784858 514.80911677
514.23136575 513.97385444 513.29183647 513.46517408 513.63994401
513.55031608 513.07322835 512.48328282 511.88784412 511.13883147
510.77363389 510.01062412 509.40960016 509.10653015 508.89319299
507.96640918 507.43623148 507.11977291 506.99710212 506.49013966
506.47165719 505.74517503 504.76312178 504.24884791 503.50545515
503.32917282 502.56243893 501.87504142 501.67070689 501.39373544
500.96409833 500.68675914 500.12947925 499.16620613 498.21726757
497.77743285 496.91984306 496.04114205 496.08890406 496.04307965
495.30410035 494.42845789 494.50938366 493.94338103 493.30219806
493.18468918 492.5034379  491.99757426 492.22255787 492.13305202
491.94662303 491.65999454 491.77318362 491.27671711 491.63114786
491.90448562 491.96005922 491.46713862 491.65895285 492.09048352
```

```
492.16107008 492.08951561 491.81264507 492.19942162 492.74570555
492.5369403  493.01693375 492.78530435 493.53421603 493.96189273
494.35789733 494.52851393 495.36369728 495.56681239 496.36354825]
        [2.48753144e-01 8.56321237e-01 1.51115369e+00 2.17751391e+00
2.52592877e+00 3.63511225e+00 4.26961894e+00 5.33417086e+00
6.28234002e+00 6.85876808e+00 7.98158289e+00 8.71682655e+00
9.96605430e+00 1.09444751e+01 1.22519897e+01 1.33985118e+01
1.46055198e+01 1.56158856e+01 1.71077960e+01 1.83138696e+01
1.93255099e+01 2.05448101e+01 2.12014273e+01 2.21643140e+01
2.35090346e+01 2.45062190e+01 2.58532031e+01 2.70320829e+01
2.79380702e+01 2.90265075e+01 3.04054894e+01 3.18234427e+01
3.32420959e+01 3.47560388e+01 3.62844487e+01 3.76068737e+01
3.94082473e+01 4.06935638e+01 4.20592052e+01 4.38920094e+01
4.56492599e+01 4.73134421e+01 4.83822522e+01 5.02776516e+01
5.19741801e+01 5.39607155e+01 5.51022785e+01 5.69659665e+01
5.81264861e+01 5.97415523e+01 6.08653723e+01 6.27128870e+01
6.45182327e+01 6.60828954e+01 6.74821934e+01 6.85390488e+01
7.02187648e+01 7.16482242e+01 7.33388630e+01 7.51655390e+01
7.70921170e+01 7.88888011e+01 7.98302917e+01 8.11080244e+01
8.27428420e+01 8.38054782e+01 8.52028215e+01 8.61179015e+01
8.71622260e+01 8.80078283e+01 8.96105430e+01 9.06246227e+01
9.17404877e+01 9.34186800e+01 9.48520631e+01 9.55910088e+01
9.64402500e+01 9.77230403e+01 9.89376929e+01 9.97884927e+01
1.01310492e+02 1.02486541e+02 1.03558001e+02 1.04656952e+02
1.05868828e+02 1.06634764e+02 1.07458764e+02 1.08065756e+02
1.08620073e+02 1.09903183e+02 1.10951751e+02 1.11721201e+02
1.12197306e+02 1.12669805e+02 1.12915216e+02 1.13506067e+02
1.13943196e+02 1.14733648e+02 1.15174190e+02 1.15385205e+02
1.15409883e+02 1.15445722e+02 1.16062324e+02 1.16421913e+02
1.16833159e+02 1.17573396e+02 1.18376353e+02 1.18375107e+02
1.18789495e+02 1.18773827e+02 1.18485461e+02 1.18905101e+02
1.18856994e+02 1.18843310e+02 1.19005848e+02 1.19382906e+02
1.19530134e+02 1.19893743e+02 1.19631459e+02 1.19867422e+02
1.19465273e+02 1.19805157e+02 1.19855222e+02 1.19409417e+02
1.19672241e+02 1.19695990e+02 1.19220963e+02 1.18684164e+02
1.18431851e+02 1.17667359e+02 1.17065812e+02 1.16663417e+02
1.16193259e+02 1.15796092e+02 1.15197414e+02 1.14893192e+02
1.14852221e+02 1.14051998e+02 1.13639601e+02 1.12830788e+02
1.12596591e+02 1.12032357e+02 1.11629195e+02 1.10836558e+02
1.09999119e+02 1.09499487e+02 1.08862985e+02 1.08807855e+02
1.08575153e+02 1.08324310e+02 1.08228030e+02 1.07311946e+02
1.06866112e+02 1.06455026e+02 1.06424848e+02 1.05729307e+02
1.04987848e+02 1.04112225e+02 1.03160072e+02 1.03129308e+02
1.02848168e+02 1.02692440e+02 1.02044393e+02 1.01713049e+02
1.00872177e+02 1.00466798e+02 1.00567987e+02 1.00583750e+02
1.00077581e+02 1.00212070e+02 9.96347551e+01 9.97939189e+01
9.99647834e+01 1.00013875e+02 9.99153542e+01 1.00082535e+02
9.96910086e+01 9.98786403e+01 9.98590930e+01 9.92594228e+01
9.90188711e+01 9.86049285e+01 9.90509311e+01 9.90202600e+01
9.90358977e+01 9.92213797e+01 9.91641850e+01 9.89039374e+01
9.93579306e+01 9.92090406e+01 9.94113426e+01 9.93566685e+01
9.92058231e+01 9.98498714e+01 1.00635410e+02 1.01439810e+02
1.02221032e+02 1.02900971e+02 1.03171326e+02 1.03221016e+02
1.03628257e+02 1.03891902e+02 1.04087181e+02 1.04234716e+02
1.05085644e+02 1.05253506e+02 1.06211468e+02 1.06576557e+02
1.06904769e+02 1.07273364e+02 1.08254428e+02 1.08838534e+02
1.09627198e+02 1.10581714e+02 1.11868045e+02 1.13049079e+02
1.13801161e+02 1.14441685e+02 1.15032886e+02 1.15896568e+02
1.16746472e+02 1.17815519e+02 1.19136225e+02 1.20493162e+02
```

```
1.21461228e+02 1.22548262e+02 1.23458381e+02 1.24997038e+02
1.25824399e+02 1.27311551e+02 1.28899264e+02 1.30503752e+02
1.31607502e+02 1.32842058e+02 1.34305964e+02 1.35469792e+02
1.36745472e+02 1.37860281e+02 1.39249914e+02 1.40235407e+02
1.41986259e+02 1.43023509e+02 1.44510927e+02 1.45793654e+02
1.47353485e+02 1.49300606e+02 1.50938291e+02 1.51969216e+02
1.53397645e+02 1.54907168e+02 1.56443346e+02 1.58124245e+02
1.59399868e+02 1.60524290e+02 1.61909080e+02 1.63853174e+02
1.65022593e+02 1.66902493e+02 1.68414882e+02 1.69832088e+02
1.71665185e+02 1.73064670e+02 1.74718614e+02 1.76035394e+02
1.77844266e+02 1.79425297e+02 1.81001226e+02 1.82189688e+02
1.83790795e+02 1.85253937e+02 1.86302974e+02 1.87253668e+02
1.88820572e+02 1.90529850e+02 1.91716958e+02 1.93015049e+02
1.94557318e+02 1.95675854e+02 1.97265152e+02 1.98453810e+02
1.99862363e+02 2.01333290e+02 2.02684640e+02 2.03353096e+02
2.04463748e+02 2.05316173e+02 2.05804665e+02 2.06625845e+02
2.07252654e+02 2.08440430e+02 2.08782752e+02 2.10020726e+02
2.10624785e+02 2.11199635e+02 2.11403344e+02 2.11745011e+02
2.12271604e+02 2.13295004e+02 2.13457409e+02 2.14434121e+02
2.15303610e+02 2.15726362e+02 2.15990272e+02 2.16128908e+02
2.16618039e+02 2.16494679e+02 2.16322904e+02 2.16533557e+02
2.16352941e+02 2.16325891e+02 2.16238035e+02 2.16152488e+02
2.15915419e+02 2.15530307e+02 2.15220012e+02 2.15384502e+02
2.15877004e+02 2.16358308e+02 2.16231535e+02 2.15832406e+02
2.15883383e+02 2.15760781e+02 2.16112767e+02 2.15516759e+02
2.15615446e+02 2.15196738e+02 2.14709188e+02 2.14621582e+02
2.14097132e+02 2.13972836e+02 2.13681127e+02 2.13278136e+02
2.12988496e+02 2.12414805e+02 2.12245073e+02 2.11768923e+02
2.11224642e+02 2.11135544e+02 2.11130735e+02 2.10235861e+02
2.09517432e+02 2.08905657e+02 2.08752041e+02 2.08761615e+02
2.08748300e+02 2.08665560e+02 2.07970001e+02 2.07966451e+02
2.07217844e+02 2.06324243e+02 2.06275196e+02 2.05509110e+02
2.05200851e+02 2.04263646e+02 2.04002240e+02 2.03896272e+02
2.03186422e+02 2.02589562e+02 2.01995275e+02 2.01783769e+02
2.01070520e+02 2.00301492e+02 1.99821008e+02 1.99207721e+02
1.99142083e+02 1.98488819e+02 1.98114902e+02 1.98196743e+02
1.97986413e+02 1.98026622e+02 1.98157814e+02 1.98118424e+02
1.98047486e+02 1.98265341e+02 1.98530773e+02 1.98275488e+02
1.97661810e+02 1.97293292e+02 1.96808223e+02 1.96612802e+02
1.96156307e+02 1.96198341e+02 1.96045322e+02 1.96431667e+02
1.96917708e+02 1.97010602e+02 1.96859646e+02 1.97054729e+02
1.97583366e+02 1.97651361e+02 1.97894422e+02 1.97887084e+02
1.98093838e+02 1.98277408e+02 1.98338441e+02 1.98687873e+02
1.98872920e+02 1.99151847e+02 1.99614027e+02 2.00333211e+02
2.00666718e+02 2.01622288e+02 2.02547397e+02 2.02690196e+02
2.03069171e+02 2.03671741e+02 2.03963750e+02 2.04530370e+02
2.05519157e+02 2.06478663e+02 2.07410064e+02 2.08459330e+02
2.09032379e+02 2.09771225e+02 2.10873246e+02 2.12206472e+02
2.13231903e+02 2.14023645e+02 2.14589707e+02 2.15862127e+02
2.16457796e+02 2.17443382e+02 2.18611343e+02 2.20194767e+02
2.20968808e+02 2.22081858e+02 2.22835139e+02 2.23927861e+02
2.25320606e+02 2.26389829e+02 2.27408596e+02 2.28351379e+02
2.29772848e+02 2.31328860e+02 2.32877123e+02 2.34716991e+02
2.35624521e+02 2.37189473e+02 2.38719641e+02 2.40249262e+02
2.42163088e+02 2.43140254e+02 2.44805319e+02 2.46594915e+02
2.48087009e+02 2.49408663e+02 2.51191617e+02 2.52284422e+02
2.53724485e+02 2.55243944e+02 2.56937900e+02 2.58028292e+02
2.59254078e+02 2.60659942e+02 2.62275351e+02 2.64150001e+02
2.65751114e+02 2.66860492e+02 2.68809655e+02 2.70641735e+02
```

```
2.72095512e+02 2.73958741e+02 2.75429898e+02 2.77270959e+02
2.79005683e+02 2.80864976e+02 2.82661066e+02 2.83557842e+02
2.84576942e+02 2.85793157e+02 2.87554316e+02 2.88644500e+02
2.89575481e+02 2.91211897e+02 2.92387260e+02 2.94002242e+02
2.94847020e+02 2.96169449e+02 2.97247137e+02 2.97936531e+02
2.99220779e+02 3.00030261e+02 3.00605704e+02 3.01174690e+02
3.01717522e+02 3.03079245e+02 3.04244908e+02 3.05540119e+02
3.06580825e+02 3.07448503e+02 3.08061967e+02 3.09338920e+02
3.09949799e+02 3.10638591e+02 3.11204218e+02 3.12340179e+02
3.12640171e+02 3.13062267e+02 3.13805406e+02 3.13900025e+02
3.14507274e+02 3.14953510e+02 3.15174719e+02 3.15319024e+02
3.15708204e+02 3.15919697e+02 3.16188835e+02 3.16308169e+02
3.17029973e+02 3.16884422e+02 3.16672209e+02 3.16676862e+02
3.16899765e+02 3.17161793e+02 3.17133271e+02 3.17678558e+02
3.17548698e+02 3.17761063e+02 3.17862819e+02 3.18113789e+02
3.18502230e+02 3.18779116e+02 3.18906254e+02 3.18943191e+02
3.18594142e+02 3.18034721e+02 3.17368388e+02 3.16860179e+02
3.16521947e+02 3.16253788e+02 3.16217855e+02 3.16349515e+02
3.15622508e+02 3.14868770e+02 3.14582158e+02 3.14436549e+02
3.14379873e+02 3.14397950e+02 3.14451600e+02 3.13910521e+02
3.13339799e+02 3.12527314e+02 3.12243666e+02 3.11924511e+02
3.11804286e+02 3.10913856e+02 3.10419518e+02 3.10005038e+02
3.09248568e+02 3.08418087e+02 3.08277668e+02 3.07336696e+02
3.06809291e+02 3.05929563e+02 3.05394507e+02 3.05386781e+02
3.04828200e+02 3.04709408e+02 3.03858140e+02 3.03169099e+02
3.02621835e+02 3.02080766e+02 3.01822373e+02 3.01249337e+02
3.01058277e+02 3.00806457e+02 3.00329312e+02 2.99900330e+02
2.99670529e+02 2.98913491e+02 2.98926881e+02 2.98790147e+02
2.98745976e+02 2.98532788e+02 2.97914297e+02 2.97612225e+02
2.97333052e+02 2.96992783e+02 2.96385309e+02 2.96509657e+02
2.96031656e+02 2.96075880e+02 2.96243271e+02 2.96369016e+02
2.96846405e+02 2.96495712e+02 2.96937100e+02 2.96569143e+02
2.96735935e+02 2.96801441e+02 2.97016607e+02 2.97102716e+02
2.97044593e+02 2.96845573e+02 2.97176341e+02 2.97862684e+02
2.98110566e+02 2.98239322e+02 2.98257063e+02 2.98310964e+02
2.98532360e+02 2.98663785e+02 2.98991615e+02 2.99151873e+02
2.99342811e+02 3.00355522e+02 3.00705023e+02 3.01482849e+02
3.02504994e+02 3.03240395e+02 3.03702781e+02 3.04241115e+02
3.05042224e+02 3.05967606e+02 3.06742825e+02 3.07978295e+02
3.09211071e+02 3.10647324e+02 3.11212103e+02 3.12669878e+02
3.13650232e+02 3.15141511e+02 3.16469720e+02 3.18118381e+02
3.19562210e+02 3.20633862e+02 3.21863894e+02 3.22990394e+02
3.24125819e+02 3.25176878e+02 3.26481866e+02 3.27990686e+02
3.29112354e+02 3.30497476e+02 3.31491163e+02 3.32542015e+02
3.33493648e+02 3.35382135e+02 3.36315771e+02 3.37435232e+02
3.38999155e+02 3.40040819e+02 3.41891298e+02 3.43586835e+02
3.45535512e+02 3.47030836e+02 3.48323355e+02 3.49868417e+02
3.51797263e+02 3.53317531e+02 3.54584738e+02 3.56461643e+02
3.57831589e+02 3.58828534e+02 3.60068334e+02 3.61374255e+02
3.63215320e+02 3.64649740e+02 3.66062911e+02 3.67359307e+02
3.69191041e+02 3.71076949e+02 3.72998616e+02 3.74293111e+02
3.76164086e+02 3.77846972e+02 3.79398968e+02 3.80504599e+02
3.81565385e+02 3.82558513e+02 3.84290287e+02 3.85374518e+02
3.86598126e+02 3.87842195e+02 3.89349335e+02 3.90900677e+02
3.91724296e+02 3.92812513e+02 3.94292791e+02 3.95723732e+02
3.96413931e+02 3.97132398e+02 3.97972423e+02 3.98556824e+02
3.99463240e+02 4.00024848e+02 4.01018845e+02 4.01662550e+02
4.02627108e+02 4.03083371e+02 4.04368272e+02 4.05579824e+02
4.06220311e+02 4.06680633e+02 4.07118413e+02 4.07758241e+02
```

```
4.07923567e+02 4.08657380e+02 4.09128474e+02 4.09537291e+02
4.10346656e+02 4.11024281e+02 4.11915824e+02 4.12173652e+02
4.12945861e+02 4.13559394e+02 4.13729437e+02 4.14132960e+02
4.14172840e+02 4.14768248e+02 4.14571659e+02 4.14445355e+02
4.14260264e+02 4.14266142e+02 4.14585596e+02 4.14658513e+02
4.14664435e+02 4.14155940e+02 4.14045515e+02 4.13546985e+02
4.13167453e+02 4.13486940e+02 4.13064914e+02 4.13261940e+02
4.13380286e+02 4.12832326e+02 4.12709069e+02 4.12074620e+02
4.12031995e+02 4.11879302e+02 4.11882224e+02 4.11534528e+02
4.11605063e+02 4.10793670e+02 4.10210252e+02 4.10034298e+02
4.09547580e+02 4.08802777e+02 4.07980211e+02 4.07856669e+02
4.07378756e+02 4.07300746e+02 4.07065452e+02 4.06499261e+02
4.05890485e+02 4.05418435e+02 4.05315380e+02 4.05057097e+02
4.04064223e+02 4.03758874e+02 4.03678382e+02 4.03389331e+02
4.02568310e+02 4.02056266e+02 4.01204467e+02 4.00575774e+02
4.00530604e+02 4.00477993e+02 3.99792246e+02 3.99171584e+02
3.98820740e+02 3.98843057e+02 3.98061081e+02 3.97400243e+02
3.97368973e+02 3.96969859e+02 3.96992513e+02 3.96317289e+02
3.96005630e+02 3.95504819e+02 3.95012344e+02 3.94669066e+02
3.94331631e+02 3.93938866e+02 3.94124869e+02 3.94149506e+02
3.94192506e+02 3.93821107e+02 3.93761393e+02 3.93927665e+02
3.94135470e+02 3.93917917e+02 3.94152963e+02 3.94379816e+02
3.94039569e+02 3.94451524e+02 3.94467066e+02 3.94556328e+02
3.94636573e+02 3.94865681e+02 3.95177833e+02 3.95397022e+02
3.96015120e+02 3.96424712e+02 3.96638759e+02 3.96899016e+02
3.97584521e+02 3.97838605e+02 3.97987269e+02 3.98015242e+02
3.98891269e+02 3.99841226e+02 4.00024136e+02 4.00452718e+02
4.00753424e+02 4.01025757e+02 4.02113005e+02 4.02388025e+02
4.03617201e+02 4.03992643e+02 4.05165130e+02 4.06070374e+02
4.06806416e+02 4.07545907e+02 4.08729050e+02 4.09497614e+02
4.10636687e+02 4.11663054e+02 4.12683847e+02 4.14147092e+02
4.15289522e+02 4.16729291e+02 4.18092289e+02 4.18792158e+02
4.20249815e+02 4.21173792e+02 4.22805910e+02 4.23598945e+02
4.24702531e+02 4.25592419e+02 4.26877025e+02 4.28524128e+02
4.29666643e+02 4.30678320e+02 4.31884894e+02 4.33589780e+02
4.34634991e+02 4.35780108e+02 4.36778737e+02 4.38425328e+02
4.39771617e+02 4.41042751e+02 4.42729961e+02 4.44041855e+02
4.45751913e+02 4.47554801e+02 4.48764579e+02 4.50733959e+02
4.51894344e+02 4.53184692e+02 4.54364487e+02 4.55709499e+02
4.57187587e+02 4.58705325e+02 4.60551045e+02 4.62428182e+02
4.63630573e+02 4.65229383e+02 4.66309463e+02 4.67728726e+02
4.69002116e+02 4.70259498e+02 4.71671858e+02 4.73319441e+02
4.74293450e+02 4.76063630e+02 4.77674751e+02 4.78711992e+02
4.80495481e+02 4.81828874e+02 4.83637699e+02 4.84625114e+02
4.85930535e+02 4.86970894e+02 4.88004037e+02 4.89302209e+02
4.90908475e+02 4.92375051e+02 4.93736416e+02 4.94908497e+02
4.96418581e+02 4.97473025e+02 4.98882684e+02 5.00225707e+02
5.00921297e+02 5.02146512e+02 5.02584463e+02 5.03304440e+02
5.03902132e+02 5.04747733e+02 5.05793603e+02 5.06170271e+02
5.06933883e+02 5.08090438e+02 5.08506466e+02 5.09340042e+02
5.10058255e+02 5.10162427e+02 5.10701044e+02 5.11441225e+02
5.11485200e+02 5.12333311e+02 5.12790602e+02 5.12727155e+02
5.12826235e+02 5.13623476e+02 5.14018415e+02 5.13907744e+02
5.13946599e+02 5.14124311e+02 5.13836244e+02 5.13909122e+02
5.14030456e+02 5.13876986e+02 5.14039846e+02 5.14339095e+02
5.14093176e+02 5.14204634e+02 5.14308430e+02 5.14694887e+02
5.14885378e+02 5.15120816e+02 5.14973812e+02 5.14497849e+02
5.14809117e+02 5.14231366e+02 5.13973854e+02 5.13291836e+02
5.13465174e+02 5.13639944e+02 5.13550316e+02 5.13073228e+02
```

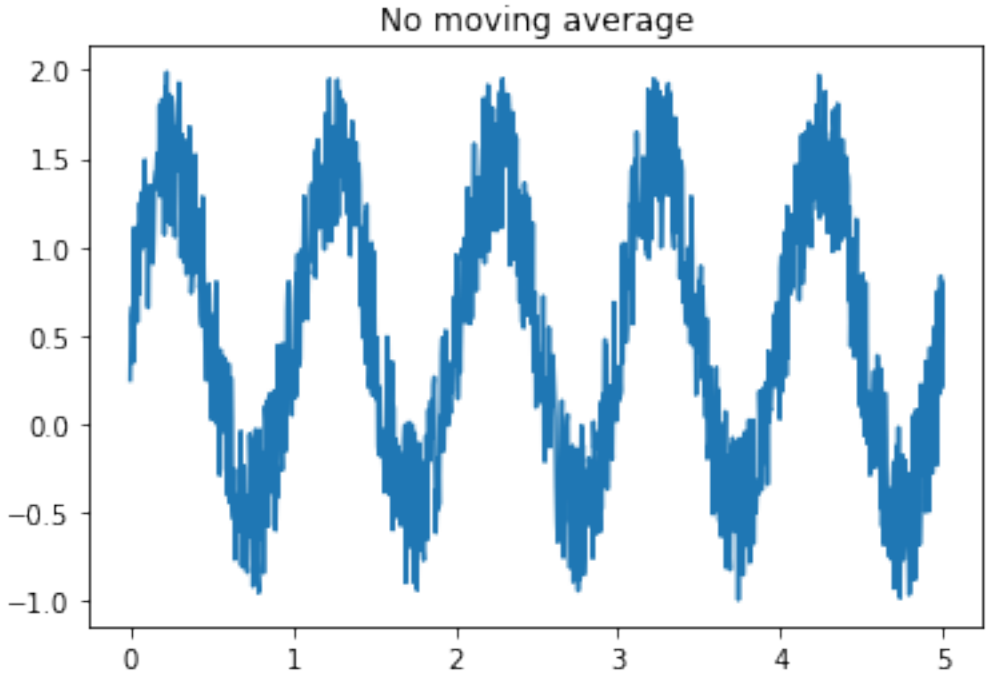
```
 5.12483283e+02 5.11887844e+02 5.11138831e+02 5.10773634e+02
 5.10010624e+02 5.09409600e+02 5.09106530e+02 5.08893193e+02
 5.07966409e+02 5.07436231e+02 5.07119773e+02 5.06997102e+02
 5.06490140e+02 5.06471657e+02 5.05745175e+02 5.04763122e+02
 5.04248848e+02 5.03505455e+02 5.03329173e+02 5.02562439e+02
 5.01875041e+02 5.01670707e+02 5.01393735e+02 5.00964098e+02
 5.00686759e+02 5.00129479e+02 4.99166206e+02 4.98217268e+02
 4.97777433e+02 4.96919843e+02 4.96041142e+02 4.96088904e+02
 4.96043080e+02 4.95304100e+02 4.94428458e+02 4.94509384e+02
 4.93943381e+02 4.93302198e+02 4.93184689e+02 4.92503438e+02
 4.91997574e+02 4.92222558e+02 4.92133052e+02 4.91946623e+02
 4.91659995e+02 4.91773184e+02 4.91276717e+02 4.91631148e+02]
CS: [ 2.48753144e-01  8.56321237e-01  1.51115369e+00  2.17751391e+00
  2.52592877e+00  3.63511225e+00  4.26961894e+00  5.33417086e+00
  6.28234002e+00  6.85876808e+00  7.98158289e+00  8.71682655e+00
  9.96605430e+00  1.09444751e+01  1.22519897e+01  1.33985118e+01
  1.46055198e+01  1.56158856e+01  1.71077960e+01  1.83138696e+01
  1.90767567e+01  1.96884889e+01  1.96902736e+01  1.99868001e+01
  2.09831059e+01  2.08711068e+01  2.15835842e+01  2.16979120e+01
  2.16557302e+01  2.21677394e+01  2.24239065e+01  2.31066162e+01
  2.32760416e+01  2.38115637e+01  2.40324590e+01  2.42083619e+01
  2.48027276e+01  2.50776783e+01  2.49514092e+01  2.55781398e+01
  2.63237500e+01  2.67686320e+01  2.71808249e+01  2.81133376e+01
  2.84651455e+01  2.94544964e+01  2.92490754e+01  2.99338836e+01
  3.01884160e+01  3.07150448e+01  3.04598828e+01  3.08894443e+01
  3.12761368e+01  3.13268567e+01  3.11977447e+01  3.09321751e+01
  3.08105175e+01  3.09546604e+01  3.12796578e+01  3.12735297e+01
  3.14428572e+01  3.15753590e+01  3.14480395e+01  3.08303728e+01
  3.07686619e+01  2.98447628e+01  3.01005430e+01  2.91519350e+01
  2.90357398e+01  2.82662761e+01  2.87451708e+01  2.79117357e+01
  2.72222549e+01  2.73357846e+01  2.73698696e+01  2.70519600e+01
  2.62214852e+01  2.60748161e+01  2.55988299e+01  2.46229537e+01
  2.42183750e+01  2.35977402e+01  2.37277092e+01  2.35489279e+01
  2.31259860e+01  2.28292853e+01  2.22559424e+01  2.19478540e+01
  2.14578471e+01  2.18953550e+01  2.13412081e+01  2.10965783e+01
  2.04568179e+01  1.92511253e+01  1.80631527e+01  1.79150581e+01
  1.75029460e+01  1.70106075e+01  1.62364973e+01  1.55967119e+01
  1.40993913e+01  1.29591809e+01  1.25043235e+01  1.17649607e+01
  1.09643309e+01  1.09386323e+01  1.09175896e+01  1.03093517e+01
  1.01694224e+01  8.87064338e+00  7.53370959e+00  7.18390041e+00
  6.65968849e+00  6.17350477e+00  6.09063212e+00  5.87683898e+00
  5.58693804e+00  5.16009548e+00  4.45726836e+00  4.48221738e+00
  4.05538934e+00  4.35943507e+00  3.79289710e+00  2.98750438e+00
  2.83908189e+00  2.12259409e+00  8.44609472e-01  3.09056636e-01
 -3.57644208e-01 -1.10646777e+00 -1.41964870e+00 -2.24168454e+00
 -2.66373508e+00 -3.04721769e+00 -3.80843348e+00 -4.48971362e+00
 -4.67791323e+00 -5.84174524e+00 -5.99185786e+00 -7.03663398e+00
 -6.86868150e+00 -7.77280009e+00 -8.22602623e+00 -8.57285896e+00
 -9.67312194e+00 -1.01965030e+01 -1.03579780e+01 -9.87630893e+00
 -9.85669788e+00 -9.34304852e+00 -8.83778186e+00 -9.35147088e+00
 -9.32714720e+00 -9.34106649e+00 -8.77256684e+00 -9.16388548e+00
 -9.86437250e+00 -9.93977255e+00 -1.04795287e+01 -9.70148041e+00
 -9.74842354e+00 -9.33991742e+00 -9.58480195e+00 -9.12350948e+00
 -9.12694178e+00 -9.03268865e+00 -8.29499822e+00 -8.22410498e+00
 -8.49757248e+00 -8.11223996e+00 -8.59327507e+00 -7.51802714e+00
 -6.90132842e+00 -6.44115100e+00 -6.50949337e+00 -5.64677137e+00
 -5.29683972e+00 -4.23358521e+00 -3.30097898e+00 -3.86988480e+00
 -3.82929657e+00 -4.08751123e+00 -2.99346225e+00 -2.69278890e+00
 -1.83627943e+00 -1.24541848e+00 -1.40380163e+00 -1.67981246e+00
```

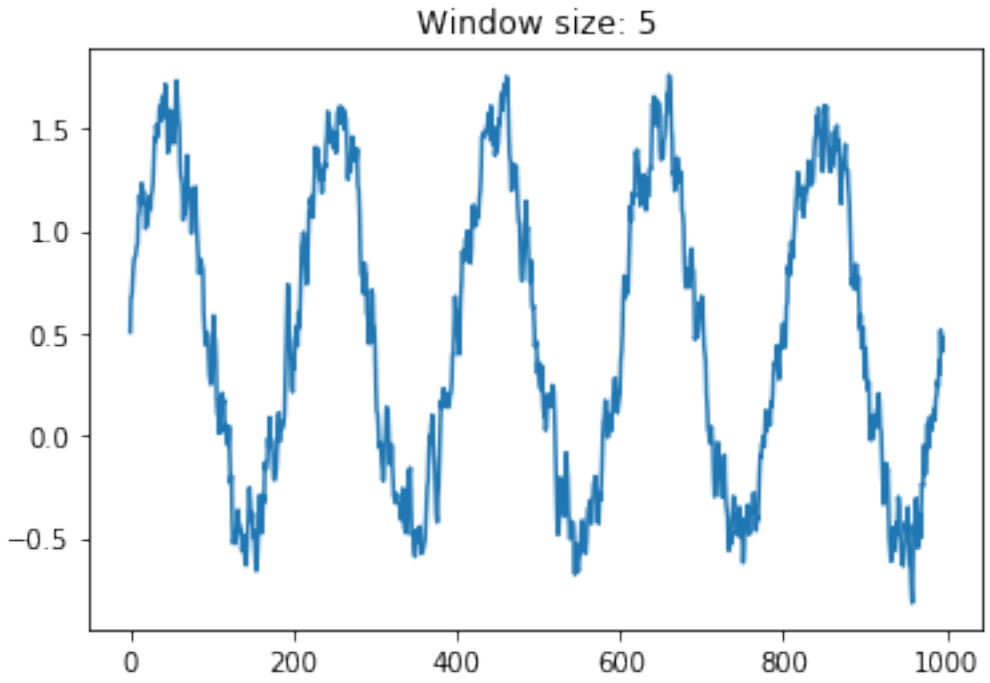
```
-7.19650309e-01 -1.00302990e+00 -2.23412542e-01 -4.37250429e-01
-7.58960293e-01 -1.64003535e-01  7.20055330e-01  1.35727434e+00
 2.53002342e+00  3.02233075e+00  3.31223268e+00  3.96159351e+00
 4.60938642e+00  5.28697383e+00  5.03624939e+00  5.21445600e+00
 6.04974590e+00  6.03212581e+00  7.04728256e+00  7.67262007e+00
 7.54683886e+00  8.06432300e+00  8.84308582e+00  9.48186502e+00
 1.04213744e+01  1.07318429e+01  1.12326351e+01  1.16092697e+01
 1.15801290e+01  1.15407142e+01  1.18615607e+01  1.26755521e+01
 1.31182143e+01  1.39236170e+01  1.50490442e+01  1.62584460e+01
 1.63755844e+01  1.72947562e+01  1.72469137e+01  1.84204804e+01
 1.89196298e+01  2.00381879e+01  2.06448355e+01  2.16652185e+01
 2.19803049e+01  2.22603434e+01  2.24379193e+01  2.24207129e+01
 2.29443112e+01  2.34185959e+01  2.42170272e+01  2.43388382e+01
 2.52397872e+01  2.52079897e+01  2.53747027e+01  2.53004923e+01
 2.58922565e+01  2.67523445e+01  2.74799098e+01  2.69721776e+01
 2.75732455e+01  2.75956166e+01  2.75440817e+01  2.76204927e+01
 2.77923650e+01  2.76822324e+01  2.76031164e+01  2.83833821e+01
 2.82771204e+01  2.90422122e+01  2.91649689e+01  2.95966810e+01
 2.96789265e+01  3.00411611e+01  3.02076869e+01  3.02417400e+01
 3.04907812e+01  3.01246911e+01  3.00629349e+01  3.02204729e+01
 3.03931503e+01  3.03467685e+01  2.98596283e+01  2.91294231e+01
 2.94207049e+01  3.00055601e+01  2.98078772e+01  2.91618740e+01
 2.95347257e+01  2.87733603e+01  2.88502696e+01  2.86217227e+01
 2.81971772e+01  2.82686195e+01  2.79660260e+01  2.73177020e+01
 2.66194826e+01  2.58908758e+01  2.48034393e+01  2.44361571e+01
 2.34618591e+01  2.31864937e+01  2.24797781e+01  2.27670581e+01
 2.18042121e+01  2.06697845e+01  1.96863869e+01  1.87299627e+01
 1.77142857e+01  1.76191501e+01  1.61922571e+01  1.59803112e+01
 1.54412474e+01  1.43930719e+01  1.33056316e+01  1.27758116e+01
 1.21542909e+01  1.11785063e+01  1.05182388e+01  9.90771116e+00
 9.10028672e+00  7.88546038e+00  7.45528268e+00  6.13176212e+00
 5.29063407e+00  4.33067227e+00  3.81666745e+00  3.63949042e+00
 3.60540023e+00  3.06330408e+00  2.77412593e+00  1.39828469e+00
 5.79772664e-01  3.44193192e-02  1.22494744e-01 -6.12148730e-01
-1.00259287e+00 -1.29794128e+00 -1.61371604e+00 -1.91197502e+00
-2.25580926e+00 -2.35305504e+00 -2.55690744e+00 -2.87435230e+00
-2.92692309e+00 -3.11550193e+00 -2.97493891e+00 -3.61557832e+00
-4.65236236e+00 -5.22276365e+00 -5.10080053e+00 -5.59654503e+00
-6.36595106e+00 -6.85512367e+00 -7.36072614e+00 -6.75514382e+00
-6.86714669e+00 -6.53117841e+00 -6.73918683e+00 -6.65513106e+00
-6.87928769e+00 -7.64859220e+00 -7.40593111e+00 -7.76902583e+00
-7.78764418e+00 -8.15115927e+00 -8.24283290e+00 -7.87265138e+00
-8.03822024e+00 -8.54598239e+00 -9.13545980e+00 -8.45209177e+00
-8.44691152e+00 -8.60416487e+00 -8.93103303e+00 -9.55389391e+00
-9.60621630e+00 -1.01767410e+01 -9.85509978e+00 -9.76970749e+00
-9.23143103e+00 -8.29762179e+00 -8.11738171e+00 -7.39068556e+00
-7.15336528e+00 -5.99830517e+00 -5.47146741e+00 -5.62078438e+00
-5.52461135e+00 -5.29626932e+00 -5.18705163e+00 -5.17096688e+00
-4.91421353e+00 -4.10315143e+00 -3.77568533e+00 -2.77605464e+00
-2.22437571e+00 -1.47821661e+00 -1.25525511e+00 -1.14201416e+00
-4.03047324e-01 -3.75260730e-01 -2.63392142e-01 -2.31340330e-01
 4.63521146e-02  1.20679116e-02 -1.92332100e-01  4.12384917e-01
 1.21110984e+00  1.85855422e+00  2.80580343e+00  3.72040837e+00
 4.51041176e+00  5.42394686e+00  6.50207466e+00  6.25852880e+00
 6.15146356e+00  6.66113914e+00  7.10410406e+00  7.47564077e+00
 7.93579157e+00  8.82730174e+00  9.51564188e+00  1.05722465e+01
 1.09385405e+01  1.14938162e+01  1.25348058e+01  1.35185992e+01
 1.43589833e+01  1.48717978e+01  1.49756800e+01  1.55289159e+01
 1.57910778e+01  1.58210944e+01  1.60639465e+01  1.75045713e+01
```

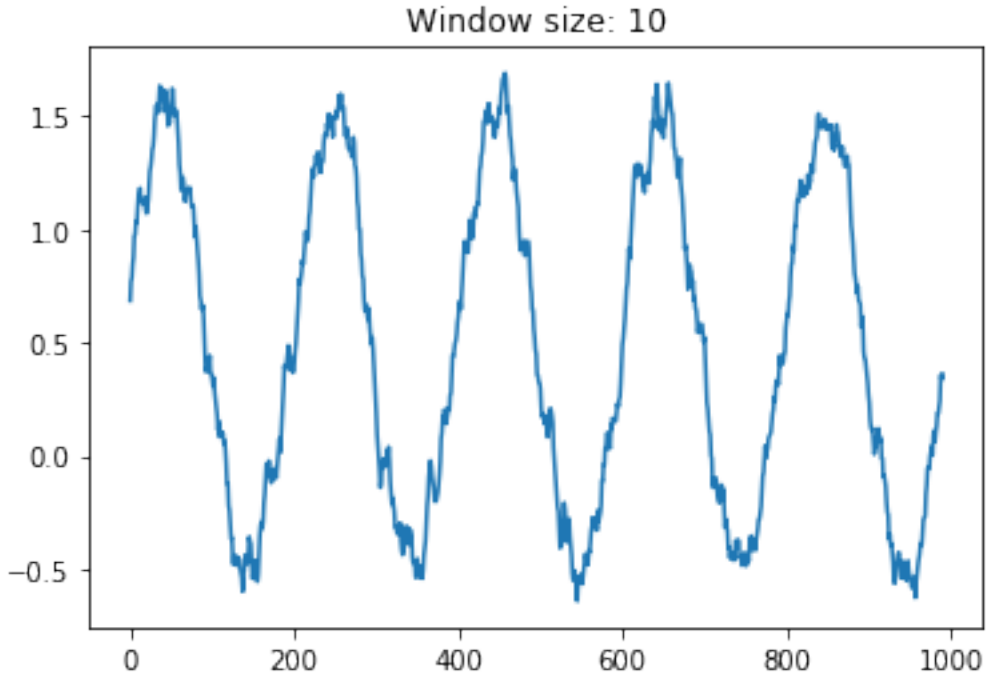
```
 1.78996372e+01  1.84101163e+01  1.88713889e+01  1.93974917e+01
 1.98014486e+01  1.99111660e+01  1.99985315e+01  1.98920490e+01
 2.07404689e+01  2.15576350e+01  2.20038770e+01  2.25105193e+01
 2.23926178e+01  2.31658280e+01  2.41299347e+01  2.43871349e+01
 2.57052924e+01  2.56968716e+01  2.61939760e+01  2.64001481e+01
 2.71182009e+01  2.73268050e+01  2.83564780e+01  2.83565608e+01
 2.84038789e+01  2.88541149e+01  2.95293043e+01  2.96769128e+01
 2.94812309e+01  2.93310824e+01  2.93982279e+01  2.94330095e+01
 3.01265927e+01  2.96710196e+01  3.00900142e+01  3.03924732e+01
 2.99324237e+01  3.08184871e+01  3.06245788e+01  3.06760440e+01
 3.09186740e+01  3.14563137e+01  3.14694484e+01  3.12734193e+01
 3.08524569e+01  3.05492132e+01  3.06164165e+01  3.06162081e+01
 3.03214028e+01  3.05519550e+01  3.01119087e+01  2.98522419e+01
 2.90959061e+01  2.93089572e+01  2.84374816e+01  2.72947961e+01
 2.71252672e+01  2.60715206e+01  2.51758056e+01  2.39037306e+01
 2.27118384e+01  2.22142689e+01  2.15838426e+01  2.19822770e+01
 2.20038840e+01  2.16553466e+01  2.05076502e+01  2.06944197e+01
 2.03743177e+01  1.94266942e+01  1.88169576e+01  1.83379367e+01
 1.77931507e+01  1.68928176e+01  1.65582691e+01  1.59634936e+01
 1.52864946e+01  1.49232482e+01  1.45690154e+01  1.41443346e+01
 1.39906819e+01  1.28404516e+01  1.19439269e+01  1.07680508e+01
 1.04491478e+01  9.43591825e+00  8.61024185e+00  7.33794230e+00
 6.94996570e+00  6.52320209e+00  5.92905306e+00  5.33837916e+00
 4.90852686e+00  4.69879580e+00  4.05741251e+00  4.21376462e+00
 3.99495566e+00  3.82560652e+00  3.73153505e+00  3.62416669e+00
 2.88593825e+00  2.11502393e+00  1.17955273e+00  5.52009220e-01
-5.08026348e-01 -6.30633413e-01 -4.54353812e-01 -3.27347271e-01
-1.27725706e+00 -2.29302353e+00 -2.55111295e+00 -3.24200889e+00
-3.16882488e+00 -3.36311268e+00 -3.41121882e+00 -4.20326839e+00
-5.16243013e+00 -6.25180174e+00 -6.66258845e+00 -7.01867987e+00
-6.78985558e+00 -7.12086473e+00 -6.94886990e+00 -6.85514057e+00
-7.27337849e+00 -7.83570091e+00 -7.94018663e+00 -9.01281862e+00
-8.81321700e+00 -8.93920707e+00 -9.18765106e+00 -9.04976862e+00
-9.55167285e+00 -9.68854215e+00 -1.05934595e+01 -1.07414220e+01
-1.07179645e+01 -1.04465481e+01 -1.04212924e+01 -1.06751741e+01
-1.07460096e+01 -1.01073989e+01 -1.00902066e+01 -1.01047085e+01
-9.57803902e+00 -9.50459625e+00 -9.35078735e+00 -8.54654956e+00
-8.06331451e+00 -7.39677462e+00 -7.48021020e+00 -7.77455535e+00
-7.49514797e+00 -7.71662501e+00 -7.47283107e+00 -6.65944228e+00
-6.59017929e+00 -6.00488591e+00 -5.57910220e+00 -4.88032080e+00
-4.21187203e+00 -4.31074475e+00 -3.39221135e+00 -3.33118711e+00
-2.93459458e+00 -2.11205050e+00 -1.91027338e+00 -1.68743075e+00
-1.70138295e+00 -1.68721522e+00 -7.37955208e-01  2.50458358e-01
 7.77514603e-01  1.24653898e+00  1.87175413e+00  1.80130711e+00
 2.50070435e+00  2.58790441e+00  2.74834393e+00  2.78285680e+00
 2.49640632e+00  3.85980931e+00  3.76792302e+00  4.91370632e+00
 5.76905945e+00  6.43895442e+00  6.68617343e+00  7.13839872e+00
 7.99763045e+00  9.12203278e+00  9.56648386e+00  1.01156116e+01
 1.11005042e+01  1.24080019e+01  1.29550397e+01  1.43589142e+01
 1.51178718e+01  1.64777266e+01  1.74781051e+01  1.89665080e+01
 2.02193994e+01  2.02783402e+01  2.11588704e+01  2.15075450e+01
 2.16208246e+01  2.19364835e+01  2.27790849e+01  2.37495713e+01
 2.40701308e+01  2.45298706e+01  2.47483380e+01  2.45637197e+01
 2.42825778e+01  2.47348107e+01  2.51036682e+01  2.47653542e+01
 2.53489235e+01  2.48993072e+01  2.54215781e+01  2.54684544e+01
 2.59733021e+01  2.63969745e+01  2.64594611e+01  2.68780234e+01
 2.76714438e+01  2.81406523e+01  2.81028720e+01  2.84709569e+01
 2.87192341e+01  2.83310576e+01  2.85771704e+01  2.88322404e+01
 2.97216717e+01  2.92676056e+01  2.97471392e+01  2.99240744e+01
```

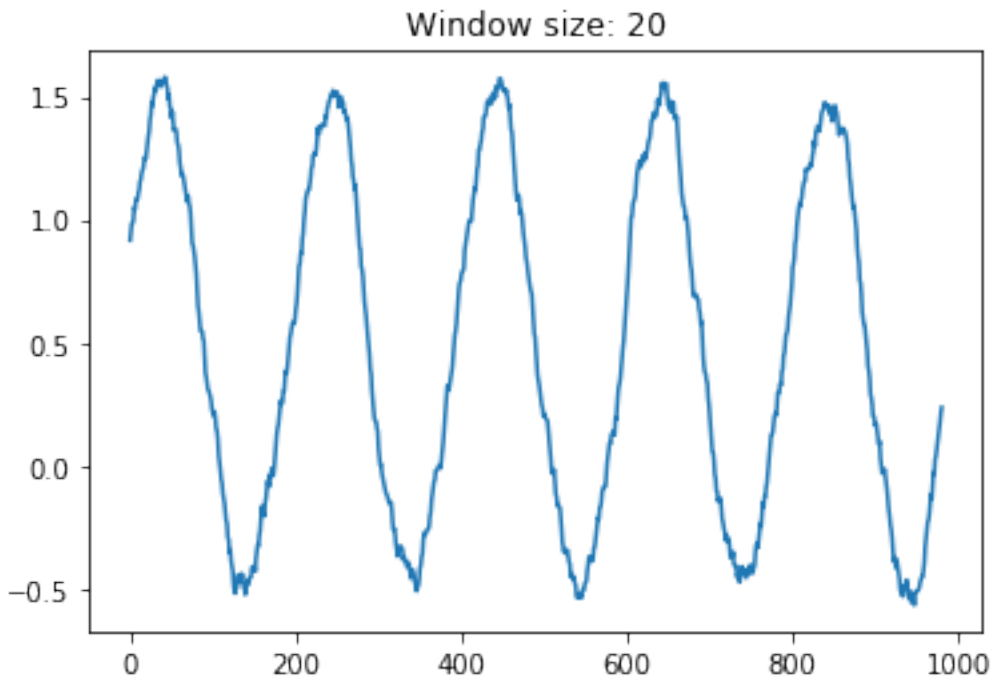
```
 3.01918858e+01  3.10361300e+01  3.11073172e+01  3.07062762e+01
 3.06285734e+01  3.08161356e+01  3.10756131e+01  3.06361821e+01
 2.97681218e+01  2.92409822e+01  2.97055488e+01  2.89128751e+01
 2.87665371e+01  2.90136607e+01  2.92810012e+01  2.95264211e+01
 2.85089762e+01  2.81627725e+01  2.82298809e+01  2.83644253e+01
 2.72228897e+01  2.60554489e+01  2.49738075e+01  2.42637121e+01
 2.32991543e+01  2.21778764e+01  2.16198775e+01  2.11579508e+01
 2.10617232e+01  2.05248584e+01  2.00779857e+01  2.02053063e+01
 1.96221856e+01  1.88384386e+01  1.77690779e+01  1.68575641e+01
 1.61992703e+01  1.58448675e+01  1.48356821e+01  1.38135591e+01
 1.39327250e+01  1.38918830e+01  1.39434008e+01  1.36168283e+01
 1.34826211e+01  1.35345456e+01  1.27105921e+01  1.24704094e+01
 1.15457319e+01  1.16848771e+01  1.02033866e+01  8.86553073e+00
 8.03995264e+00  7.58550892e+00  7.46718306e+00  6.90027201e+00
 6.74086862e+00  5.49855975e+00  4.91704125e+00  4.00969432e+00
 2.82079735e+00  2.46265953e+00  1.14909045e+00  1.08828830e+00
 4.34425032e-01 -7.27068233e-01 -1.02036824e+00 -2.05833963e+00
-2.14084479e+00 -2.88894633e+00 -2.68943509e+00 -2.91082756e+00
-2.65520113e+00 -3.47247232e+00 -4.37534411e+00 -4.62421498e+00
-5.11685547e+00 -5.35316329e+00 -6.06530397e+00 -5.69031602e+00
-5.78869756e+00 -6.18619446e+00 -5.99946270e+00 -6.76267903e+00
-7.48980153e+00 -7.41389098e+00 -7.39368906e+00 -7.01752324e+00
-7.96777192e+00 -8.12042790e+00 -8.20384206e+00 -8.14519684e+00
-9.03675294e+00 -8.73740370e+00 -9.00578427e+00 -9.45852341e+00
-9.01697577e+00 -8.32478396e+00 -8.18796452e+00 -8.68508551e+00
-8.55801562e+00 -8.45768908e+00 -9.00437020e+00 -9.09901763e+00
-8.52151162e+00 -8.44857592e+00 -8.32286676e+00 -8.73980750e+00
-8.05859307e+00 -8.25405531e+00 -8.66603715e+00 -8.72026440e+00
-8.23667924e+00 -8.11739960e+00 -7.07959877e+00 -6.42626830e+00
-6.33809821e+00 -6.65688517e+00 -6.03085343e+00 -5.24391841e+00
-4.68526964e+00 -4.92513908e+00 -3.90811797e+00 -3.02042751e+00
-3.32940447e+00 -2.51833521e+00 -2.52544744e+00 -1.76096076e+00
-1.36905679e+00 -6.39137769e-01  1.65488918e-01  7.27955427e-01
 1.68348988e+00  2.48584558e+00  2.51389007e+00  2.74950996e+00
 3.39201546e+00  4.01749750e+00  4.22587634e+00  4.08757639e+00
 4.75579812e+00  5.92330825e+00  5.87117299e+00  6.07290257e+00
 6.71385506e+00  6.57423302e+00  7.64593890e+00  7.83169684e+00
 8.98062777e+00  9.12696165e+00  9.98729704e+00  1.06733526e+01
 1.07912960e+01  1.11211947e+01  1.20902915e+01  1.25985981e+01
 1.30521653e+01  1.38244491e+01  1.46965773e+01  1.61318504e+01
 1.63982539e+01  1.68880653e+01  1.80681521e+01  1.83394392e+01
 1.94963911e+01  2.01480352e+01  2.06929054e+01  2.12109201e+01
 2.10853299e+01  2.15997759e+01  2.17118943e+01  2.24537533e+01
 2.28602265e+01  2.31324132e+01  2.31558435e+01  2.40921664e+01
 2.39983048e+01  2.41170544e+01  2.40948904e+01  2.42782356e+01
 2.44820951e+01  2.43134601e+01  2.46376721e+01  2.52496973e+01
 2.55020987e+01  2.63810094e+01  2.59586686e+01  2.71350140e+01
 2.71918134e+01  2.75922729e+01  2.74874621e+01  2.71853713e+01
 2.75209437e+01  2.80270046e+01  2.86661516e+01  2.88384012e+01
 2.89955813e+01  2.94492750e+01  2.95307258e+01  2.93033985e+01
 2.92304988e+01  2.92167467e+01  2.89418978e+01  2.92775858e+01
 2.85415370e+01  2.85088291e+01  2.89101720e+01  2.79780328e+01
 2.86011369e+01  2.86441823e+01  2.92732125e+01  2.89156148e+01
 2.87429486e+01  2.82655699e+01  2.74529914e+01  2.68740273e+01
 2.72779023e+01  2.71456673e+01  2.74269534e+01  2.71797707e+01
 2.74164644e+01  2.72135277e+01  2.72108259e+01  2.69062667e+01
 2.66278460e+01  2.60828821e+01  2.49097121e+01  2.45924481e+01
 2.34066513e+01  2.29188594e+01  2.21559040e+01  2.15451570e+01
 2.10033476e+01  2.11195438e+01  2.05024294e+01  2.00378329e+01
```

```
 1.91497802e+01  1.77873765e+01  1.69646278e+01  1.65327281e+01
 1.50666198e+01  1.48602858e+01  1.39079177e+01  1.25014474e+01
 1.19049387e+01  1.14769640e+01  1.14339520e+01  1.06033039e+01
 1.00444665e+01  9.37657811e+00  8.04264103e+00  7.73885115e+00
 7.09657315e+00  5.78654787e+00  5.53337971e+00  4.99905372e+00
 4.03492077e+00  4.04220724e+00  3.60738597e+00  3.25366175e+00
 3.40017723e+00  2.78750472e+00  2.18320974e+00  1.77069373e+00
 1.98288154e+00  6.07889288e-01 -4.45604384e-02 -6.15907735e-01
-4.81424736e-01 -4.84367327e-01 -2.85928334e-01 -8.35893658e-01
-1.54717327e+00 -1.98914195e+00 -2.90101432e+00 -3.56546151e+00
-4.08255183e+00 -4.79503428e+00 -5.20189974e+00 -5.80169382e+00
-6.91896854e+00 -7.68458446e+00 -7.85403895e+00 -7.50074645e+00
-8.31897711e+00 -7.75970856e+00 -8.22867940e+00 -8.52871469e+00
-9.21632617e+00 -1.01344889e+01 -1.02211433e+01 -1.05107894e+01
-1.06082414e+01 -1.02171372e+01 -9.74509603e+00 -9.80953556e+00
-9.32386497e+00 -9.28012091e+00 -9.94032402e+00 -1.06759254e+01
-1.01889763e+01 -1.05163884e+01 -1.10786309e+01 -1.09081981e+01
-1.04470600e+01 -1.11675568e+01 -1.13167171e+01 -1.02537381e+01
-1.03054669e+01 -1.02032571e+01 -1.01444836e+01 -1.00590010e+01
-9.87746716e+00 -9.44814902e+00 -9.26068342e+00 -9.01747531e+00
-9.02676461e+00 -8.35629562e+00 -7.88948902e+00 -6.58611971e+00
-5.87294723e+00 -4.95978384e+00 -4.57400344e+00 -4.42995122e+00
-3.95259614e+00 -3.14303027e+00 -2.33894228e+00 -2.69673859e+00
-1.74395941e+00 -5.56492512e-01 -6.47748876e-01  5.13495846e-01
 7.87730090e-01  1.31165816e+00  1.82884072e+00  2.41127430e+00
 2.86851939e+00  3.59051366e+00  4.29009528e+00  4.73240039e+00]
```




Window size: 5
1.5
1.0
0.5
0.0
−0.5
0
200
400
600
800
1000


Window size: 10
1.5
1.0
0.5
0.0
−0.5
0
200
400
600
800
1000


Window size: 20
1.5
1.0
0.5
0.0
−0.5
0
200
400
600
800
1000

CHAPTER 11

Practical 10

In this practical we will will get acquainted with Biopython.

11.1 Slides

The slides of the introduction can be found here: Intro

11.2 Biopython

From the Biopython tutorial: The Biopython Project is an international association of developers of freely available Python tools for computational molecular biology.

The goal of Biopython is **to make it as easy as possible to use Python for bioinformatics by creating high-quality, reusable modules and classes.** Biopython features include parsers for various Bioinformatics file formats (FASTA, FASTQ, BLAST, PDB, Clustalw, Genbank,...), access to online services (NCBI, Expasy,...), interfaces to common and not-so-common programs (Clustalw, DSSP, MSMS...), a standard sequence class, various clustering modules, a KD tree data structure etc. and even documentation.

In this practical we will see some features of Biopython but refer to biopython documentation[63] to discover all its features, recipes etc.

These notes are largely based on what available here[64].

11.3 Installation of Biopython

To test if Biopython is installed in your system you can import the library with:

[63] http://biopython.org/wiki/Documentation
[64] http://biopython.org/DIST/docs/tutorial/Tutorial.pdf

```
import Bio
```

if the execution gives:

```
import Bio
```

```
---------------------------------------------------------------------------
ImportError                               Traceback (most recent call last)
<ipython-input-1-f227b1b7f7f3> in <module>()
----> 1 import Bio

ImportError: No module named 'Bio'
```

Biopython needs to be installed.

In windows installing Biopython should be as easy as opening the command prompt as admininstrator (typing `cmd` and then right clicking on the link choosing run as admininstrator) and then `pip3 install biopython`. If that does not work, you should be able to install biopython with the following: `python3.xx -m pip install biopython` (where you should change python3.xx with the path to your python3.xx executable).

In linux `sudo pip3 install biopython` will install biopython for python3 up to python3.5. On python 3.6 and later the command is: `python3.6 -m pip install biopython` or, more in general, `pythonX.X -m pip install biopython`.

Detailed installation instructions can be found here[65].

11.4 General overview

Biopython provides the capability of parsing standard bioinformatics formats into python usable data structures. Some of the supported tools and formats are:

1. Blast output – both from standalone and WWW Blast
2. Clustalw
3. FASTA
4. GenBank
5. PubMed and Medline
6. ExPASy files, like Enzyme and Prosite
7. SCOP, including 'dom' and 'lin' files
8. UniGene
9. SwissProt

among the others. In particular it provides a sequence class to perform the most common operations on sequences, parsing of alignments, clustering etc. and it can quickly connect to the most popular databases to download data or parse information on the fly.

[65] http://biopython.org/DIST/docs/install/Installation.html

11.5 Sequence objects

Biopython has a specific object `Seq` to deal with biological sequences which is a lot more powerful than the `str` object of python when it comes to manage biological data. The object **Seq** has methods like **translate()** and **reverse_complement()** which are very handy.

As in the case of python's `str` objects, the object `Seq` is immutable. The mutable version is `MutableSeq`.

All the information on Seq objects can be found here[66].

11.5.1 The alphabet

`Seq` objects are composed by two information: the actual sequence and the alphabet the sequence refers to. We normally will use the IUPAC[67] alphabet as defined in the `Bio.Alphabet.IUPAC` which provides a suitable alphabet to deal with DNA, RNA and proteins. Several options are available, including `Bio.Alphabet.generic_dna` or `Bio.Alphabet.generic_protein`, `Bio.Alphabet.ThreeLetterProtein` and other.

Specifying an alphabet quickly tells you what information you are dealing with and moreover it provides a way to specify a constraint on the data.

If the definition of the alphabet is not specified, the default one is used (`Alphabet`).

```
In [1]: from Bio.Seq import Seq
        from Bio.Alphabet import IUPAC

        #No alphabet specified
        s = Seq("GATTACATAATA")
        dna_seq = Seq("GATTATACGTAC", IUPAC.unambiguous_dna)
        print("S:", s)
        print("S's alphabet:", s.alphabet)
        print("dna_seq:", dna_seq)
        print("dna_seq's alphabet:", dna_seq.alphabet)

        my_prot = Seq("MGNAAAAKKGSEQE", IUPAC.protein)
        print("my_prot:", my_prot)
        print("my_prot's alphabet:", my_prot.alphabet)
```

```
S: GATTACATAATA
S's alphabet: Alphabet()
dna_seq: GATTATACGTAC
dna_seq's alphabet: IUPACUnambiguousDNA()
my_prot: MGNAAAAKKGSEQE
my_prot's alphabet: IUPACProtein()
```

11.6 Looping, slicing and concatenation

Note also that **Seq objects behave like strings**, but the consistency of the alphabet is checked too. For example we cannot concatenate a **unambiguous_dna** with a **IUPAC.protein** sequence. This is the error message we would get:

To do it one should change the alphabet to `generic_alphabet`:

```
In [2]: from Bio.Seq import Seq
        from Bio.Alphabet import generic_alphabet
```

[66] http://biopython.org/DIST/docs/api/Bio.Seq.Seq-class.html
[67] http://www.chem.qmw.ac.uk/iupac/

```
my_mess = dna_seq + my_prot
```

```
S: GATTACATAATA
S's alphabet: Alphabet()
dna_seq: GATTATACGTAC
dna_seq's alphabet: IUPACUnambiguousDNA()
my_prot: MGNAAAAKKGSEQE
my_prot's alphabet: IUPACProtein()

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-20-4c1f6d65a691> in <module>()
     14 print("my_prot's alphabet:", my_prot.alphabet)
     15
---> 16 my_mess = dna_seq + my_prot

/usr/local/lib/python3.5/dist-packages/Bio/Seq.py in __add__(self, other)
    296                 raise TypeError(
    297                     "Incompatible alphabets {0!r} and {1!r}".format(
--> 298                         self.alphabet, other.alphabet))
    299             # They should be the same sequence type (or one of them is generic)
    300             a = Alphabet._consensus_alphabet([self.alphabet, other.alphabet])

TypeError: Incompatible alphabets IUPACUnambiguousDNA() and IUPACProtein()
```

```
        dna_seq = Seq("GATTATACGTAC", IUPAC.unambiguous_dna)
        my_prot = Seq("MGNAAAAKKGSEQE", IUPAC.protein)

        my_prot.alphabet = generic_alphabet

        #Does it really make sense though?!?
        print(dna_seq + my_prot)
```

```
GATTATACGTACMGNAAAAKKGSEQE
```

We can also loop through the elements and slice strings with the usual [S:E:step] operator (and as usual S is included, E is excluded). We can also convert a Seq object into a string.

```
In [3]: from Bio.Seq import Seq
        from Bio.Alphabet import IUPAC

        dna_seq = Seq("GATTATACGTACGGCTA", IUPAC.unambiguous_dna)

        for base in dna_seq:
            print(base, end = " ")

        print("")

        sub_seq = dna_seq[4:10]
        print(sub_seq)

        #Let's reverse the string:

        print("Reversed: ", dna_seq[::-1])
        #from Seq to string:
        dna_str = str(dna_seq)
        print("As string:", dna_str)
        print(type(dna_str))
```

```
G A T T A T A C G T A C G G C T A
```

```
ATACGT
Reversed:  ATCGGCATGCATATTAG
As string: GATTATACGTACGGCTA
<class 'str'>
```

The object Seq provides several methods similar to the corresponding methods of strings:

1. `Seq.count(s)` : counts the number of times s appears in the sequence;
2. `Seq.upper()` : makes the sequence of the object Seq upper case
3. `Seq.lower()` : makes the sequence of the object Seq lower case

Example: Let's compute the GC content of the sequence "GATTRWWACGTACGGCTASATTACSCCGGCTA".

```
In [4]: from Bio.Seq import Seq
        from Bio.Alphabet import IUPAC
        from Bio.SeqUtils import GC

        dna_seq = Seq("GATTRWWACGTACGGCTASATTACSCCGGCTA", IUPAC.ambiguous_dna)

        gc = (dna_seq.count("G") + dna_seq.count("C"))/len(dna_seq)
        print("GC % is {:.2f}".format(100*gc))

        correct_gc = GC(dna_seq)
        print("GC % is {:.2f}".format(correct_gc))

        print("original:", dna_seq)
        print("lower_case:",dna_seq.lower())
```

```
GC % is 40.62
GC % is 46.88
original: GATTRWWACGTACGGCTASATTACSCCGGCTA
lower_case: gattrwwacgtacggctasattacsccggcta
```

The difference in the results goes down to the fact that Biopython's GC method takes into account ambiguous nucleotides like (S which stands for C or G ref iupac[68]). This is a method provided in the **SeqUtils** module (more info on all the methods here[69]).

11.6.1 Complement and reverse complement

A usual operation on DNA strings is the reverse complement. Biopython provides two specific methods to do that:

1. `Seq.complement()` to complement the sequence
2. `Seq.reverse_complement()` to reverse complement the sequence.

Note that these methods will fail with a `Value Error` if we try to apply them to something that is not a nucleotide sequence (e.g. proteins).

```
In [5]: from Bio.Seq import Seq
        from Bio.Alphabet import IUPAC

        my_seq = Seq("GATCGATGGGCCTATATAGGATCGAAAATCGC", IUPAC.unambiguous_dna)

        print("Sequence (alphabet: {}):".format(my_seq.alphabet))
        print(my_seq)
        comp  = my_seq.complement()
        print("")
```

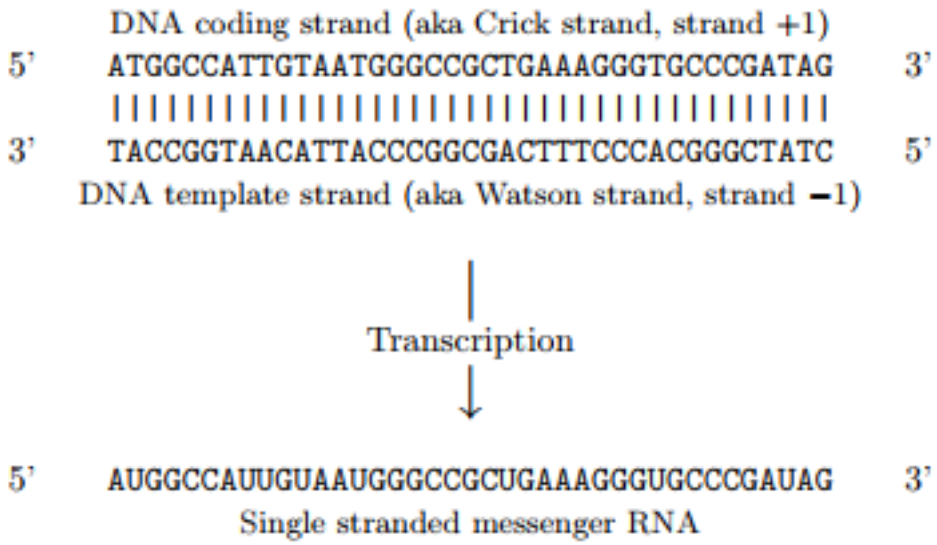
[68] https://www.bioinformatics.org/sms/iupac.html
[69] http://biopython.org/DIST/docs/api/Bio.SeqUtils-module.html

```
        print("Complement (alphabet: {}):".format(comp.alphabet))
        print(comp)
        print("")
        revcomp = my_seq.reverse_complement()
        print("Rev Complement (alphabet: {}):".format(revcomp.alphabet))
        print(revcomp)
```

```
Sequence (alphabet: IUPACUnambiguousDNA()):
GATCGATGGGCCTATATAGGATCGAAAATCGC

Complement (alphabet: IUPACUnambiguousDNA()):
CTAGCTACCCGGATATATCCTAGCTTTTAGCG

Rev Complement (alphabet: IUPACUnambiguousDNA()):
GCGATTTTCGATCCTATATAGGCCCATCGATC
```

11.6.2 Transcription and translation

Remember that:



[taken from the Biopython tutorial].

Biopython has a method to transcribe a DNA sequence into a mRNA sequence (and one to go back):

1. `Seq.transcribe()` that transcribes a DNA sequence into mRNA (converting Ts in Us).
2. `Seq.back_transcribe()` that back-transcribes a mRNA sequence into DNA (converting Us in Ts).

The alphabet of the transcribed sequence is converted into mRNA. Note that transcribing a protein will result in a **ValueError**.

Alike, a DNA or mRNA sequence can be translated into a protein with:

`Seq.translate(table=N, to_stop=True/False, stop_symbol =symbol)` that converts a DNA sequence or mRNA into the corresponding protein.

Optionally, the translate method accepts a parameter `table` to specify the NCBI translation table used (see below for codes). `to_stop` makes the translation stop at the first stop codon if True (**in this case the stop codon is not translated!**), `stop_symbol` to replace the asterisk with any other stop character (Es. "@").

1. The Standard Code
2. The Vertebrate Mitochondrial Code
3. The Yeast Mitochondrial Code
4. The Mold, Protozoan, and Coelenterate Mitochondrial Code and the Mycoplasma/Spiroplasma Code
5. The Invertebrate Mitochondrial Code
6. The Ciliate, Dasycladacean and Hexamita Nuclear Code
9. The Echinoderm and Flatworm Mitochondrial Code
10. The Euplotid Nuclear Code
11. The Bacterial, Archaeal and Plant Plastid Code
12. The Alternative Yeast Nuclear Code
13. The Ascidian Mitochondrial Code
14. The Alternative Flatworm Mitochondrial Code
16. Chlorophycean Mitochondrial Code
21. Trematode Mitochondrial Code
22. Scenedesmus obliquus Mitochondrial Code
23. Thraustochytrium Mitochondrial Code
24. Pterobranchia Mitochondrial Code
25. Candidate Division SR1 and Gracilibacteria Code
26. Pachysolen tannophilus Nuclear Code
27. Karyorelict Nuclear
28. Condylostoma Nuclear
29. Mesodinium Nuclear
30. Peritrich Nuclear
31. Blastocrithidia Nuclear

Stop codons will be represented by an asterisk.

Example: Let's translate into mRNA the coding sequence: ATGGCCATTGTAATGGGCCGCTGAAAGGGTGCCCGATAG.

```
In [6]: from Bio.Seq import Seq
        from Bio.Alphabet import IUPAC

        coding_dna = Seq("ATGGCCATTGTAATGGGCCGCTGAAAGGGTGCCCGATAG",
                         IUPAC.unambiguous_dna)
        print(coding_dna)

        mrna = coding_dna.transcribe()
        print(mrna)
        print("")
        print(mrna.alphabet)
        print("")
        print("... and back")
        print(mrna.back_transcribe())
        print("")
        print("Translation to protein:")
        prot = mrna.translate()
        print(prot)
        print("")
        print("Up to first stop:")
        print(mrna.translate(to_stop = True))
        print("")
        print("Mitocondrial translation: (TGA is W!)")
        mit_prot = mrna.translate(table=2)
        print(mit_prot)
```

```
ATGGCCATTGTAATGGGCCGCTGAAAGGGTGCCCGATAG
AUGGCCAUUGUAAUGGGCCGCUGAAAGGGUGCCCGAUAG

IUPACUnambiguousRNA()

... and back
ATGGCCATTGTAATGGGCCGCTGAAAGGGTGCCCGATAG

Translation to protein:
MAIVMGR*KGAR*

Up to first stop:
MAIVMGR

Mitocondrial translation: (TGA is W!)
MAIVMGRWKGAR*
```

Translation tables can be accessed at runtime through:

```
In [7]: from Bio.Data import CodonTable
        #equivalent to:
        #standard_table = CodonTable.unambiguous_dna_by_id[1]
        standard_table = CodonTable.unambiguous_dna_by_name["Standard"]
        #equivalent to:
        #standard_table = CodonTable.unambiguous_dna_by_id[2]
        mito_table = CodonTable.unambiguous_dna_by_name["Vertebrate Mitochondrial"]


        print(standard_table)
        print(standard_table)

        print(mito_table)
```

```
Table 1 Standard, SGC0

  |  T      |  C      |  A      |  G      |
--+---------+---------+---------+---------+--
T | TTT F   | TCT S   | TAT Y   | TGT C   | T
T | TTC F   | TCC S   | TAC Y   | TGC C   | C
T | TTA L   | TCA S   | TAA Stop| TGA Stop| A
T | TTG L(s)| TCG S   | TAG Stop| TGG W   | G
--+---------+---------+---------+---------+--
C | CTT L   | CCT P   | CAT H   | CGT R   | T
C | CTC L   | CCC P   | CAC H   | CGC R   | C
C | CTA L   | CCA P   | CAA Q   | CGA R   | A
C | CTG L(s)| CCG P   | CAG Q   | CGG R   | G
--+---------+---------+---------+---------+--
A | ATT I   | ACT T   | AAT N   | AGT S   | T
A | ATC I   | ACC T   | AAC N   | AGC S   | C
A | ATA I   | ACA T   | AAA K   | AGA R   | A
A | ATG M(s)| ACG T   | AAG K   | AGG R   | G
--+---------+---------+---------+---------+--
G | GTT V   | GCT A   | GAT D   | GGT G   | T
G | GTC V   | GCC A   | GAC D   | GGC G   | C
G | GTA V   | GCA A   | GAA E   | GGA G   | A
G | GTG V   | GCG A   | GAG E   | GGG G   | G
--+---------+---------+---------+---------+--
Table 1 Standard, SGC0

  |  T      |  C      |  A      |  G      |
```

```
--+---------+---------+---------+---------+--
T | TTT F   | TCT S   | TAT Y   | TGT C   | T
T | TTC F   | TCC S   | TAC Y   | TGC C   | C
T | TTA L   | TCA S   | TAA Stop| TGA Stop| A
T | TTG L(s)| TCG S   | TAG Stop| TGG W   | G
--+---------+---------+---------+---------+--
C | CTT L   | CCT P   | CAT H   | CGT R   | T
C | CTC L   | CCC P   | CAC H   | CGC R   | C
C | CTA L   | CCA P   | CAA Q   | CGA R   | A
C | CTG L(s)| CCG P   | CAG Q   | CGG R   | G
--+---------+---------+---------+---------+--
A | ATT I   | ACT T   | AAT N   | AGT S   | T
A | ATC I   | ACC T   | AAC N   | AGC S   | C
A | ATA I   | ACA T   | AAA K   | AGA R   | A
A | ATG M(s)| ACG T   | AAG K   | AGG R   | G
--+---------+---------+---------+---------+--
G | GTT V   | GCT A   | GAT D   | GGT G   | T
G | GTC V   | GCC A   | GAC D   | GGC G   | C
G | GTA V   | GCA A   | GAA E   | GGA G   | A
G | GTG V   | GCG A   | GAG E   | GGG G   | G
--+---------+---------+---------+---------+--
Table 2 Vertebrate Mitochondrial, SGC1

  |  T      |  C      |  A      |  G      |
--+---------+---------+---------+---------+--
T | TTT F   | TCT S   | TAT Y   | TGT C   | T
T | TTC F   | TCC S   | TAC Y   | TGC C   | C
T | TTA L   | TCA S   | TAA Stop| TGA W   | A
T | TTG L   | TCG S   | TAG Stop| TGG W   | G
--+---------+---------+---------+---------+--
C | CTT L   | CCT P   | CAT H   | CGT R   | T
C | CTC L   | CCC P   | CAC H   | CGC R   | C
C | CTA L   | CCA P   | CAA Q   | CGA R   | A
C | CTG L   | CCG P   | CAG Q   | CGG R   | G
--+---------+---------+---------+---------+--
A | ATT I(s)| ACT T   | AAT N   | AGT S   | T
A | ATC I(s)| ACC T   | AAC N   | AGC S   | C
A | ATA M(s)| ACA T   | AAA K   | AGA Stop| A
A | ATG M(s)| ACG T   | AAG K   | AGG Stop| G
--+---------+---------+---------+---------+--
G | GTT V   | GCT A   | GAT D   | GGT G   | T
G | GTC V   | GCC A   | GAC D   | GGC G   | C
G | GTA V   | GCA A   | GAA E   | GGA G   | A
G | GTG V(s)| GCG A   | GAG E   | GGG G   | G
--+---------+---------+---------+---------+--
```

11.6.3 MutableSeq

Seq objects are **immutable and therefore we cannot change their value**. If we need to do so, we need to convert them into `MutableSeq` objects to avoid a runtime **TypeError** that will occur when we try to change a Seq. Seq to MutableSeq conversion can be done by:

1. `Seq.tomutable()` to convert a Seq to a MutableSeq.
2. `MutableSeq.toseq()` to convert a MutableSeq to a Seq.

This will allow us to assign the value of characters within the MutableSeq with `MutableSeq[ind] = newValue`. Note also that methods like complement change the actual object if applied to a MutableSeq:

```
In [8]: from Bio.Seq import Seq
        from Bio.Alphabet import IUPAC

        coding_dna = Seq("ATGGCCATTGTAATGGGCCGCTGAAAGGGTGCCCGATAG", IUPAC.unambiguous_dna)
        print("Coding DNA:")
        print(coding_dna)
        mut_dna = coding_dna.tomutable()
        print("Mutable DNA:")
        print(mut_dna)
        mut_dna[10:20] = "TATATATATA"
        print("Mutable DNA now:")
        print(mut_dna)
        print("")
        print("Rev comp (Seq not changed!):")
        coding_dna.reverse_complement()
        print(coding_dna)
        print("Rev comp (MutableSeq changed!):")
        mut_dna.reverse_complement()
        print(mut_dna)
```

```
Coding DNA:
ATGGCCATTGTAATGGGCCGCTGAAAGGGTGCCCGATAG
Mutable DNA:
ATGGCCATTGTAATGGGCCGCTGAAAGGGTGCCCGATAG
Mutable DNA now:
ATGGCCATTGTATATATATACTGAAAGGGTGCCCGATAG

Rev comp (Seq not changed!):
ATGGCCATTGTAATGGGCCGCTGAAAGGGTGCCCGATAG
Rev comp (MutableSeq changed!):
CTATCGGGCACCCTTTCAGTATATATATACAATGGCCAT
```

11.7 Sequence annotation

Sequence records are annotations associated to sequences. In biopython there is a `SeqRecord` object to store this information. `SeqRecord` are part of the `Bio.SeqRecord` module which therefore needs to be imported prior to be used.

The **SeqRecord** class has the following attributes:

1. `SeqRecord.seq` : the sequence (the Seq object)
2. `SeqRecord.id` : the identifier of the sequence, typically an accession number
3. `SeqRecord.name` : a "common" name or identifier sometimes identical to the accession number
4. `SeqRecord.description` : a human readable description of the sequence
5. `SeqRecord.letter_annotations` : a per letter annotation using a restricted dictionary (e.g. quality)
6. `SeqRecord.annotations` : a dictionary of unstructured annotation (e.g. organism, publications,...)
7. `SeqRecord.features` : a list of SeqFeature objects with more structured information (e.g. genes pos).
8. `SeqRecord.dbxrefs` : a list of database cross references.

Information on SeqFeature objects can be found here[70].

[70] http://biopython.org/DIST/docs/tutorial/Tutorial.html#htoc38

Although you can manually create a SeqRecord, normally you will access SeqRecords populated by methods like SeqIO, here[71] you can find how to manually create a SeqRecord.

Example:

Read a fasta file NC005816.fna containing the whole sequence for Yersinia pestis biovar Microtus str. 91001 plasmid pPCP1 and retrieve some information about the sequence.

```
In [9]: from Bio import SeqIO

        record = SeqIO.read("file_samples/NC_005816.fna", "fasta")

        print(record)
        print("")
        print("Sequence [first 30 bases]:")
        print(record.seq[0:30])
        print("")
        print("The id:")
        print(record.id)
        print("")
        print("The description:")
        print(record.description)
        print("")
        print("The record is a: ", type(record))
```

```
ID: gi|45478711|ref|NC_005816.1|
Name: gi|45478711|ref|NC_005816.1|
Description: gi|45478711|ref|NC_005816.1| Yersinia pestis biovar Microtus str. 91001 plasmid pPCP1, c
Number of features: 0
Seq('TGTAACGAACGGTGCAATAGTGATCCACACCCAACGCCTGAAATCAGATCCAGG...CTG', SingleLetterAlphabet())

Sequence [first 30 bases]:
TGTAACGAACGGTGCAATAGTGATCCACAC

The id:
gi|45478711|ref|NC_005816.1|

The description:
gi|45478711|ref|NC_005816.1| Yersinia pestis biovar Microtus str. 91001 plasmid pPCP1, complete seque

The record is a:  <class 'Bio.SeqRecord.SeqRecord'>
```

11.8 Sequence I/O

The `Bio.SeqIO` module aims to provide a simple way to work with several different sequence file formats. Detailed information on the module can be found here[72].

WARNING: When dealing with very large FASTA or FASTQ files, the overhead of working with all these objects can make scripts too slow. In this case `SimpleFastaParser` and `FastqGeneralIterator` parsers might be better as they return just a tuple of strings for each record.

For example, to use `SimpleFastaParser`:

[71] http://biopython.org/DIST/docs/tutorial/Tutorial.html#htoc33
[72] http://biopython.org/wiki/SeqIO

```
In [8]: from Bio.SeqIO.FastaIO import SimpleFastaParser


        with open("file_samples/NC_005816.fna") as in_handle:
            for title, seq in SimpleFastaParser(in_handle):
                s = seq[0:50]+"\n"+seq[51:101]+"\n..."
                print("Header:\n{}\nSequence:\n{}".format(title,s))

        #Another example:
        print("\n")
        labels = ["1st","2nd","3rd"]

        with open("file_samples/contigs82.fasta") as cont_handle:
            for l in labels:
                ID, seq  = next(SimpleFastaParser(cont_handle))

                print(l, "entry:")
                print(ID, " has size ", len(seq))
                print(seq[:50]+"...")
                print("")
```

```
Header:
gi|45478711|ref|NC_005816.1| Yersinia pestis biovar Microtus str. 91001 plasmid pPCP1, complete seque
Sequence:
TGTAACGAACGGTGCAATAGTGATCCACACCCAACGCCTGAAATCAGATC
AGGGGGTAATCTGCTCTCCTGATTCAGGAGAGTTTATGGTCACTTTTGAG
...


1st entry:
MDC020656.85  has size  2802
GAGGGGTTTAGTTCCTCATACTCGCAAAGCAAAGATACATAAATTTAGAA...

2nd entry:
MDC013284.379  has size  5173
TATCGTTTCCTCTGAGTAGAATATCGTTATAACAAGATTTTTTTTTTCCT...

3rd entry:
MDC018185.241  has size  23761
AAAACGAGGAAAATCCATCTTGATGAACAGGAGATGCGGAGGAAAAAAAT...
```

11.8.1 BioSeqIO.parse()

The method `Bio.SeqIO.parse` is used to parse some sequence data into a **SeqRecord iterator**. In particular, the basic syntax is:

```
SeqRecordIterator = Bio.SeqIO.parse(filename, file_format)
```

where `filename` is typically an open handle to a file and `file_format` is a lower case string describing the file format. Possible options include **fasta, fastq-illumina, abi, ace, clustal...** all the options are available here[73].

Example: Let's read the first 10 entries in the .fasta file contigs82.fasta printing off the length of the sequence and the first 50 bases of each sequence followed by "...".

[73] http://biopython.org/wiki/SeqIO

```
In [11]: from Bio import SeqIO
         cnt = 0
         for seq_record in SeqIO.parse("file_samples/contigs82.fasta", "fasta"):
             if(cnt == 10):
                 break
             print("Seq {} has length {}".format(seq_record.id, len(seq_record)))
             print(seq_record.seq[:50]+"...")
             print("")
             cnt += 1
```

```
Seq MDC020656.85 has length 2802
GAGGGGTTTAGTTCCTCATACTCGCAAAGCAAAGATACATAAATTTAGAA...

Seq MDC001115.177 has length 3118
TGAATGGTGAAAATTAGCCAGAAGATCTTCTCCACACATGACATATGCAT...

Seq MDC013284.379 has length 5173
TATCGTTTCCTCTGAGTAGAATATCGTTATAACAAGATTTTTTTTTTCCT...

Seq MDC018185.243 has length 22724
CCATTAGTGACCCCCAATGCGGATTAACCAAGCACGGTCAAGATAACCAA...

Seq MDC018185.241 has length 23761
AAAACGAGGAAAATCCATCTTGATGAACAGGAGATGCGGAGGAAAAAAAT...

Seq MDC004527.213 has length 3551
CTTGTATGTTGAAGCTTTGTGAGTGGAGCATATAGGTTGAGGTAGTGTTC...

Seq MDC003661.174 has length 3334
AGTAAGGTTGTAGAATTCAATTTCCACGGAACTCCAGCACAGCTTAGGCA...

Seq MDC012176.157 has length 2236
ATAGTCACTGACAGCCGGTGACAGTTCGACAACAATAAGTTCAGGATGTT...

Seq MDC001204.812 has length 5804
TAATGTCCGAAAAAGATGAAAATGTAGTTTGCATGAAGAAGGAGCTGTCC...

Seq MDC001204.810 has length 9798
ATGCTGCTGACATATTCCAATCGACTACACGTCACATCCGTTGCTATTAT...
```

Note that `Bio.SeqIO.parse` returns an iterator, therefore it is possible to manually fetch one SeqRecord after the other with the `next(iterator)` method.

Example: Let's get the first 3 entries of the .fasta file contigs82.fasta printing off the length of the sequence and the first 50 bases of each sequence followed by "...".

```
In [12]: from Bio import SeqIO

         seqIterator = SeqIO.parse("file_samples/contigs82.fasta", "fasta")

         labels = ["1st","2nd","3rd"]
         for l in labels:
             seqRec = next(seqIterator)
             print(l, "entry:")
             print(seqRec.id, " has size ", len(seqRec.seq))
             print(seqRec.seq[:50]+"...")
             print("")
```

```
1st entry:
```

```
MDC020656.85  has size  2802
GAGGGGTTTAGTTCCTCATACTCGCAAAGCAAAGATACATAAATTTAGAA...

2nd entry:
MDC001115.177  has size  3118
TGAATGGTGAAAATTAGCCAGAAGATCTTCTCCACACATGACATATGCAT...

3rd entry:
MDC013284.379  has size  5173
TATCGTTTCCTCTGAGTAGAATATCGTTATAACAAGATTTTTTTTTTCCT...
```

11.8.2 Sequences as dictionaries

The module `Bio.SeqIO` also has three different ways to allow random access to elements:

1. `Bio.SeqIO.to_dict(file_handle/iterator)` : builds a dictionary of all the SeqRecords keeping them in memory and allowing modifications to the records. **This potentially uses a lot of memory but is very fast**;
2. `Bio.SeqIO.index(filename,file_type)` : builds a sort of read-only dictionary, parses the elements into SeqRecords on demand (i.e. it returns an iterator!). **This method is slower, but more memory efficient**;
3. `Bio.SeqIO.index_db(indexName.idx,filenames, file_format)` : builds a read-only dictionary, but stores ids and offsets on a SQLite3 database. **It is slower but uses less memory.**

The input of `Bio.SeqIO.to_dict(file_handle/iterator)` is a open file handler or the iterator to SeqRecords (like the output of Bio.SeqIO.parse()).

Example: Let's read all the .fasta file contigs82.fasta in memory printing off the entry corresponding to contig "MDC019140.399".

```
In [13]: from Bio import SeqIO
         seqDict = SeqIO.to_dict(SeqIO.parse("file_samples/contigs82.fasta", "fasta"))

         print(list(seqDict.keys())[0:5])
         print("")
         print("Number of sequences: ", len(seqDict))
         print("")
         mySeq = seqDict["MDC019140.399"]
         print("Description:", mySeq.description)
         print("")
         print(mySeq)
```

```
['MDC020656.85', 'MDC001115.177', 'MDC013284.379', 'MDC018185.243', 'MDC018185.241']

Number of sequences:  82

Description: MDC019140.399

ID: MDC019140.399
Name: MDC019140.399
Description: MDC019140.399
Number of features: 0
Seq('GACGGGCGGGGAGGGTTTGGTTTTTTTTGGTTTTTAAAAAATTCAGGTTTNNAA...TTT', SingleLetterAlphabet())
```

The input of `Bio.SeqIO.index(filename,file_type)` is the file name to read sequences from and the file type (like fasta, genebank,...).

Example: Let's read all the .fasta file contigs82.fasta in memory (as an index) printing off the entry corresponding to contig "MDC019140.399".

```
In [14]: from Bio import SeqIO
         seqDict =  SeqIO.index("file_samples/contigs82.fasta", "fasta")
         print("SeqDict is an iterator:")
         print(seqDict.keys())
         print("")
         print(list(seqDict.keys()))
         print("")
         print("Number of sequences: ", len(seqDict))
         print("")
         mySeq = seqDict["MDC019140.399"]
         print("Description:", mySeq.description)
         print("")
         print(mySeq)
```

```
SeqDict is an iterator:
<odict_iterator object at 0x7fc4c416ff68>

['MDC020656.85', 'MDC001115.177', 'MDC013284.379', 'MDC018185.243', 'MDC018185.241', 'MDC004527.213',

Number of sequences:  82

Description: MDC019140.399

ID: MDC019140.399
Name: MDC019140.399
Description: MDC019140.399
Number of features: 0
Seq('GACGGGCGGGGAGGGTTTGGTTTTTTTTGGTTTTTAAAAAATTCAGGTTTNNAA...TTT', SingleLetterAlphabet())
```

The input of `Bio.SeqIO.index_db(indexName.idx,filenames, file_format)` is the name of the (SQLite3) index to store (.idx), the file (or files) to index and the format of those files like fasta, genebank,...

Example: Let's read all the .fasta file contigs82.fasta in memory (as an index) printing off the entry corresponding to contig "MDC019140.399".

```
In [15]: from Bio import SeqIO
         seqDict =  SeqIO.index_db("file_samples/ctgs.idx","file_samples/contigs82.fasta", "fasta")
         print("SeqDict is an iterator:")
         print(seqDict.keys())
         print("")
         print(list(seqDict.keys()))
         print("")
         print("Number of sequences: ", len(seqDict))
         print("")
         mySeq = seqDict["MDC019140.399"]
         print("Description:", mySeq.description)
         print("")
         print(mySeq)
```

```
SeqDict is an iterator:
<generator object _SQLiteManySeqFilesDict.__iter__ at 0x7fc4c416fbf8>

['MDC020656.85', 'MDC001115.177', 'MDC013284.379', 'MDC018185.243', 'MDC018185.241', 'MDC004527.213',

Number of sequences:  82

Description: MDC019140.399
```

```
ID: MDC019140.399
Name: MDC019140.399
Description: MDC019140.399
Number of features: 0
Seq('GACGGGCGGGGAGGGTTTGGTTTTTTTTGGTTTTTAAAAAATTCAGGTTTNNAA...TTT', SingleLetterAlphabet())
```

Note that a file ctgs.idx has been created to store ids and offsets.

11.8.3 Writing sequence files

SeqRecords can be written out to files by using

```
N = Bio.SeqIO.write(records,out_filename, file_format)
```

where **records** is a list of the SeqRecords to write, **out_filename** is the string with the filename to write and **file_format** is the format of the file to write. **N** is the number of sequences written.

WARNING: If you write a file that is already present, `SeqIO.write` will just rewrite it without telling you.

Example: Write three protein records to file then read the file and print the three sequences.

```
In [16]: from Bio.Seq import Seq
         from Bio.SeqRecord import SeqRecord
         from Bio.Alphabet import generic_protein

         rec1 = SeqRecord(Seq("MMYQQGCFAGGTVLRLAKDLAENNRGARVLVVCSEITAVTFRGPSETHLDSMVGQALFGD" \
         +"GAGAVIVGSDPDLSVERPLYELVWTGATLLPDSEGAIDGHLREVGLTFHLLKDVPGLISK" \
         +"NIEKSLKEAFTPLGISDWNSTFWIAHPGGPAILDQVEAKLGLKEEKMRATREVLSEYGNM" \
         +"SSAC", generic_protein),
         id="gi|14150838|gb|AAK54648.1|AF376133_1",
         description="chalcone synthase [Cucumis sativus]")

         rec2 = SeqRecord(Seq("YPDYYFRITNREHKAELKEKFQRMCDKSMIKKRYMYLTEEILKENPSMCEYMAPSLDARQ" \
         +"DMVVVEIPKLGKEAAVKAIKEWGQ", generic_protein),
         id="gi|13919613|gb|AAK33142.1|",
         description="chalcone synthase [Fragaria vesca subsp. bracteata]")

         rec3 = SeqRecord(Seq("MVTVEEFRRAQCAEGPATVMAIGTATPSNCVDQSTYPDYYFRITNSEHKVELKEKFKRMC" \
         +"EKSMIKKRYMHLTEEILKENPNICAYMAPSLDARQDIVVVEVPKLGKEAAQKAIKEWGQP" \
         +"KSKITHLVFCTTSGVDMPGCDYQLTKLLGLRPSVKRFMMYQQGCFAGGTVLRMAKDLAEN" \
         +"NKGARVLVVCSEITAVTFRGPNDTHLDSLVGQALFGDGAAAVIIGSDPIPEVERPLFELV" \
         +"SAAQTLLPDSEGAIDGHLREVGLTFHLLKDVPGLISKNIEKSLVEAFQPLGISDWNSLFW" \
         +"IAHPGGPAILDQVELKLGLKQEKLKATRKVLSNYGNMSSACVLFILDEMRKASAKEGLGT" \
         +"TGEGLEWGVLFGFGPGLTVETVVLHSVAT", generic_protein),
         id="gi|13925890|gb|AAK49457.1|",
         description="chalcone synthase [Nicotiana tabacum]")

         my_records = [rec1, rec2, rec3]
         print(rec1)

         seqWritten = SeqIO.write(my_records, "file_samples/my_proteins.fa", "fasta")
         print(seqWritten, "sequences written!")
         print("")
         print("Reading file")
         seqIT = SeqIO.parse("file_samples/my_proteins.fa", "fasta")
```

```
        cnt = 1
        for s in seqIT:
            print("Sequence ", cnt, "(", s.id, ")")
            print(s.seq)
            print("")
            cnt += 1
```

```
ID: gi|14150838|gb|AAK54648.1|AF376133_1
Name: <unknown name>
Description: chalcone synthase [Cucumis sativus]
Number of features: 0
Seq('MMYQQGCFAGGTVLRLAKDLAENNRGARVLVVCSEITAVTFRGPSETHLDSMVG...SAC', ProteinAlphabet())
3 sequences written!

Reading file
Sequence  1 ( gi|14150838|gb|AAK54648.1|AF376133_1 )
MMYQQGCFAGGTVLRLAKDLAENNRGARVLVVCSEITAVTFRGPSETHLDSMVGQALFGDGAGAVIVGSDPDLSVERPLYELVWTGATLLPDSEGAIDGHL

Sequence  2 ( gi|13919613|gb|AAK33142.1| )
YPDYYFRITNREHKAELKEKFQRMCDKSMIKKRYMYLTEEILKENPSMCEYMAPSLDARQDMVVVEIPKLGKEAAVKAIKEWGQ

Sequence  3 ( gi|13925890|gb|AAK49457.1| )
MVTVEEFRRAQCAEGPATVMAIGTATPSNCVDQSTYPDYYFRITNSEHKVELKEKFKRMCEKSMIKKRYMHLTEEILKENPNICAYMAPSLDARQDIVVVE
```

11.9 Multiple sequence alignment

Multiple Sequence Alignments, as the name says, are a collection of multiple sequences which have been aligned together – usually with the insertion of gap characters, and addition of leading or trailing gaps – such that all the sequence strings are the same length.

Such an alignment can be regarded as a matrix of letters, in biopython each row is a `SeqRecord` object. These alignments are stored in an object `MultipleSeqAlignment`

11.9.1 Parsing multiple sequence alignments

The function `Bio.AlignIO.parse()` returns an iterator of `MultipleSeqAlignment` objects. In the frequent case that we have to deal with a single alignment we will have to use the `Bio.AlignIO.read()` function.

The basic syntax of the two functions:

```
Bio.AlignIO.parse(file_handle, alignment_format)
Bio.AlignIO.read(file_handle, alignment_format)
```

where `file_handle` is the handler to the opened file, while the `alignment_format` is a lower case string with the alignment format (e.g. fasta, clustal, stockholm, mauve, phylip,...). Check here[74] for all possible formats.

When more than one multiple alignment has to be read we will have to use the `parse` function, for single entries, we would use the function `read`.

[74] http://biopython.org/wiki/AlignIO

Example: Load and visualize the seed alignment of the Piwi (PF02171) family[75] stored in the pfam (stockholm) format. Information on the format can be found here[76]. The file is available here: PF02171_seed.sth.

```
In [17]: from Bio import AlignIO

         alignments = AlignIO.read("file_samples/PF02171_seed.sth", "stockholm")

         print(alignments)
```

```
SingleLetterAlphabet() alignment with 16 rows and 395 columns
YLFFILDK-NSPEP-YGSIKRVCNTMLGVPSQCAISKHILQS--...QDV AGO1_SCHPO/500-799
FILCILPERKTSDI-YGPWKKICLTEEGIHTQCICPIKI-----...FTK AGO6_ARATH/541-851
FILCVLPDKKNSDL-YGPWKKKNLTEFGIVTQCMAPTRQPND--...FMK AGO4_ARATH/577-885
CIIVVLQS-KNSDI-YMTVKEQSDIVHGIMSQCVLMKNVSRP--...HVK TAG76_CAEEL/660-966
LIVVVLPG--KTPI-YAEVKRVGDTVLGIATQCVQAKNAIRT--...HLV O16720_CAEEL/566-867
TFVFIITD-DSITT-LHQRYKMIEKDTKMIVQDMKLSKALSV--...LWN O62275_CAEEL/594-924
DILVGIAR-EKKPD-VHDILKYFEESIGLQTIQLCQQTVDKMMG...NYK YQ53_CAEEL/650-977
TIVFGIIA-EKRPD-MHDILKYFEEKLGQQTIQISSETADKFMR...NYK NRDE3_CAEEL/673-1001
MLVVMLAD-DNKTR-YDSLKKYLCVECPIPNQCVNLRTLAGKSK...SLH Q17567_CAEEL/397-708
IVMVVMRS-PNEEK-YSCIKKRTCVDRPVPSQVVTLKVIAPRQQ...SIN AUB_DROME/555-852
LILCLVPN-DNAER-YSSIKKRGYVDRAVPTQVVTLKTTKNRSL...NLH PIWI_DROME/538-829
IVVCLLSS-NRKDK-YDAIKKYLCTDCPTPSQCVVARTLGKQQT...SIH PIWL1_HUMAN/555-847
GIMLVLPE-YNTPL-YYKLKSYLINS--IPSQFMRYDILSNRNL...VNR PIWI_ARCFU/110-406
CFALIIGKEKYKDNDYYEILKKQLFDLKIISQNILWENWRKDDK...NWK Y1321_METJA/426-699
LVIVFLEEYPKVDP-YKSFLLYDFVKRELLKKMIPSQVILNRTL...GIE O67434_AQUAE/419-694
LLLAILPD-NNGSL-YGDLKRICETELGLISQCCLTKHVFKI--...YLE AGO10_ARATH/625-946
```

The code above prints a summary of the information, but each `Bio.Align.MultipleSeqAlignment` is composed of several alignment records (`SeqRecord`) that we can access with all their information looping through the object. Each `SeqRecord` contains several information like the **ID, Name, Description, Number of features, start, end and sequence**.

Example:

Load the seed alignment of the Piwi (PF02171) family[77] stored in the pfam (stockholm) format PF02171_seed.sth. For each record print the description, start and the end point, the sequence and the external dbxrefs (if any).

```
In [31]: from Bio import AlignIO

         alignments = AlignIO.read("file_samples/PF02171_seed.sth", "stockholm")


         for align in alignments:
             start = align.annotations["start"]
             end = align.annotations["end"]
             seq = align.seq
             desc = align.description
             dbref = ",".join([x for x in align.dbxrefs])
             print("{} S:{} E:{}".format(desc, start, end))
             if(len(dbref) > 0):
                 print(dbref)
             print("{}".format(seq))
             print("")
```

```
ID: AGO1_SCHPO/500-799
Name: AGO1_SCHPO
Description: AGO1_SCHPO/500-799
```

[75] http://pfam.xfam.org/family/Piwi
[76] https://en.wikipedia.org/wiki/Stockholm_format
[77] http://pfam.xfam.org/family/Piwi

```
Number of features: 0
/accession=O74957.1
/start=500
/end=799
Seq('YLFFILDK-NSPEP-YGSIKRVCNTMLGVPSQCAISKHILQS---------KPQ...QDV', SingleLetterAlphabet())
AGO1_SCHPO/500-799 S:500 E:799
YLFFILDK-NSPEP-YGSIKRVCNTMLGVPSQCAISKHILQS---------KPQYCANLGMKINVKVGGIN-CSLIPKSNP----LGNVPTL---------

ID: AGO6_ARATH/541-851
Name: AGO6_ARATH
Description: AGO6_ARATH/541-851
Number of features: 0
/accession=O48771.2
/start=541
/end=851
Seq('FILCILPERKTSDI-YGPWKKICLTEEGIHTQCICPIKI------------SDQ...FTK', SingleLetterAlphabet())
AGO6_ARATH/541-851 S:541 E:851
FILCILPERKTSDI-YGPWKKICLTEEGIHTQCICPIKI------------SDQYLTNVLLKINSKLGGIN-SLLGIEYSYNIPLINKIPTL---------

ID: AGO4_ARATH/577-885
Name: AGO4_ARATH
Description: AGO4_ARATH/577-885
Number of features: 0
/accession=Q9ZVD5.2
/start=577
/end=885
Seq('FILCVLPDKKNSDL-YGPWKKKNLTEFGIVTQCMAPTRQPND-----------Q...FMK', SingleLetterAlphabet())
AGO4_ARATH/577-885 S:577 E:885
FILCVLPDKKNSDL-YGPWKKKNLTEFGIVTQCMAPTRQPND-----------QYLTNLLLKINAKLGGLN-SMLSVERTPAFTVISKVPTI---------

ID: TAG76_CAEEL/660-966
Name: TAG76_CAEEL
Description: TAG76_CAEEL/660-966
Number of features: 0
/accession=P34681.2
/start=660
/end=966
Seq('CIIVVLQS-KNSDI-YMTVKEQSDIVHGIMSQCVLMKNVSRP---------TPA...HVK', SingleLetterAlphabet())
TAG76_CAEEL/660-966 S:660 E:966
CIIVVLQS-KNSDI-YMTVKEQSDIVHGIMSQCVLMKNVSRP---------TPATCANIVLKLNMKMGGIN--SRIVADKITNKYLVDQPTM---------

ID: O16720_CAEEL/566-867
Name: O16720_CAEEL
Description: O16720_CAEEL/566-867
Number of features: 0
/accession=O16720.2
/start=566
/end=867
Seq('LIVVVLPG--KTPI-YAEVKRVGDTVLGIATQCVQAKNAIRT---------TPQ...HLV', SingleLetterAlphabet())
O16720_CAEEL/566-867 S:566 E:867
LIVVVLPG--KTPI-YAEVKRVGDTVLGIATQCVQAKNAIRT---------TPQTLSNLCLKMNVKLGGVN-SILLPNVRPR---IFNEPVI---------

ID: O62275_CAEEL/594-924
Name: O62275_CAEEL
Description: O62275_CAEEL/594-924
Number of features: 0
/accession=O62275.1
/start=594
/end=924
```

```
Seq('TFVFIITD-DSITT-LHQRYKMIEKDTKMIVQDMKLSKALSV--IN---AGKRL...LWN', SingleLetterAlphabet())
O62275_CAEEL/594-924 S:594 E:924
TFVFIITD-DSITT-LHQRYKMIEKDTKMIVQDMKLSKALSV--IN---AGKRLTLENVINKTNVKLGGSN--YVFVDAKKQL-----DSHL---------

ID: YQ53_CAEEL/650-977
Name: YQ53_CAEEL
Description: YQ53_CAEEL/650-977
Number of features: 0
/accession=Q09249.1
/start=650
/end=977
Seq('DILVGIAR-EKKPD-VHDILKYFEESIGLQTIQLCQQTVDKMMGG----QGGRQ...NYK', SingleLetterAlphabet())
YQ53_CAEEL/650-977 S:650 E:977
DILVGIAR-EKKPD-VHDILKYFEESIGLQTIQLCQQTVDKMMGG----QGGRQTIDNVMRKFNLKCGGTNFFVEIPNAVRGKAVCSNNETLRKKLLEHVQ

ID: NRDE3_CAEEL/673-1001
Name: NRDE3_CAEEL
Description: NRDE3_CAEEL/673-1001
Number of features: 0
/accession=Q21691.1
/start=673
/end=1001
Seq('TIVFGIIA-EKRPD-MHDILKYFEEKLGQQTIQISSETADKFMRD----HGGKQ...NYK', SingleLetterAlphabet())
NRDE3_CAEEL/673-1001 S:673 E:1001
TIVFGIIA-EKRPD-MHDILKYFEEKLGQQTIQISSETADKFMRD----HGGKQTIDNVIRKLNPKCGGTNFLIDVPESVGHRVVCNNSAEMRAKLYAKTQ

ID: Q17567_CAEEL/397-708
Name: Q17567_CAEEL
Description: Q17567_CAEEL/397-708
Number of features: 0
/accession=Q17567.1
/start=397
/end=708
Seq('MLVVMLAD-DNKTR-YDSLKKYLCVECPIPNQCVNLRTLAGKSKDGGENKNLGS...SLH', SingleLetterAlphabet())
Q17567_CAEEL/397-708 S:397 E:708
MLVVMLAD-DNKTR-YDSLKKYLCVECPIPNQCVNLRTLAGKSKDGGENKNLGSIVLKIVLQMICKTGGA---LWKVNIPL-------KSTM---------

ID: AUB_DROME/555-852
Name: AUB_DROME
Description: AUB_DROME/555-852
Number of features: 0
/accession=O76922.1
/start=555
/end=852
Seq('IVMVVMRS-PNEEK-YSCIKKRTCVDRPVPSQVVTLKVIAPRQQKP---TGLMS...SIN', SingleLetterAlphabet())
AUB_DROME/555-852 S:555 E:852
IVMVVMRS-PNEEK-YSCIKKRTCVDRPVPSQVVTLKVIAPRQQKP---TGLMSIATKVVIQMNAKLMGA---PWQVVIPL-------HGLM---------

ID: PIWI_DROME/538-829
Name: PIWI_DROME
Description: PIWI_DROME/538-829
Number of features: 0
/accession=Q9VKM1.1
/start=538
/end=829
Seq('LILCLVPN-DNAER-YSSIKKRGYVDRAVPTQVVTLKTTKNRSL--------MS...NLH', SingleLetterAlphabet())
PIWI_DROME/538-829 S:538 E:829
LILCLVPN-DNAER-YSSIKKRGYVDRAVPTQVVTLKTTKNRSL--------MSIATKIAIQLNCKLGYT---PWMIELPL-------SGLM---------
```

```
ID: PIWL1_HUMAN/555-847
Name: PIWL1_HUMAN
Description: PIWL1_HUMAN/555-847
Number of features: 0
/accession=Q96J94.1
/start=555
/end=847
Seq('IVVCLLSS-NRKDK-YDAIKKYLCTDCPTPSQCVVARTLGKQQT-------VMA...SIH', SingleLetterAlphabet())
PIWL1_HUMAN/555-847 S:555 E:847
IVVCLLSS-NRKDK-YDAIKKYLCTDCPTPSQCVVARTLGKQQT-------VMAIATKIALQMNCKMGGE---LWRVDIPL-------KLVM---------

ID: PIWI_ARCFU/110-406
Name: PIWI_ARCFU
Description: PIWI_ARCFU/110-406
Database cross-references: PDB; 2W42 B; 110-406;, PDB; 1YTU B; 110-406;, PDB; 2BGG B; 110-406;, PDB;
Number of features: 0
/accession=O28951.1
/start=110
/end=406
Per letter annotation for: secondary_structure
Seq('GIMLVLPE-YNTPL-YYKLKSYLINS--IPSQFMRYDILSNRNL--------TF...VNR', SingleLetterAlphabet())
PIWI_ARCFU/110-406 S:110 E:406
PDB; 2W42 B; 110-406;,PDB; 1YTU B; 110-406;,PDB; 2BGG B; 110-406;,PDB; 1W9H A; 110-406;,PDB; 2BGG A;
GIMLVLPE-YNTPL-YYKLKSYLINS--IPSQFMRYDILSNRNL--------TFYVDNLLVQFVSKLGGK---PWILNVDPEK-----------------

ID: Y1321_METJA/426-699
Name: Y1321_METJA
Description: Y1321_METJA/426-699
Number of features: 0
/accession=Q58717.1
/start=426
/end=699
Seq('CFALIIGKEKYKDNDYYEILKKQLFDLKIISQNILWENWRKDDK--------GY...NWK', SingleLetterAlphabet())
Y1321_METJA/426-699 S:426 E:699
CFALIIGKEKYKDNDYYEILKKQLFDLKIISQNILWENWRKDDK--------GYMTNNLLIQIMGKLGIK---YFILDSKTPYDY---------------

ID: O67434_AQUAE/419-694
Name: O67434_AQUAE
Description: O67434_AQUAE/419-694
Database cross-references: PDB; 1YVU A; 419-694;, PDB; 2F8S A; 419-694;, PDB; 2F8T A; 419-694;, PDB;
Number of features: 0
/accession=O67434.1
/start=419
/end=694
Per letter annotation for: secondary_structure
Seq('LVIVFLEEYPKVDP-YKSFLLYDFVKRELLKKMIPSQVILNRTLKN---ENLKF...GIE', SingleLetterAlphabet())
O67434_AQUAE/419-694 S:419 E:694
PDB; 1YVU A; 419-694;,PDB; 2F8S A; 419-694;,PDB; 2F8T A; 419-694;,PDB; 2F8S B; 419-694;,PDB; 2NUB A;
LVIVFLEEYPKVDP-YKSFLLYDFVKRELLKKMIPSQVILNRTLKN---ENLKFVLLNVAEQVLAKTGNIP--YKLKEIEGKVDA---------------

ID: AGO10_ARATH/625-946
Name: AGO10_ARATH
Description: AGO10_ARATH/625-946
Number of features: 0
/accession=Q9XGW1.1
/start=625
/end=946
Seq('LLLAILPD-NNGSL-YGDLKRICETELGLISQCCLTKHVFKI---------SKQ...YLE', SingleLetterAlphabet())
AGO10_ARATH/625-946 S:625 E:946
```

```
LLLAILPD-NNGSL-YGDLKRICETELGLISQCCLTKHVFKI---------SKQYLANVSLKINVKMGGRN-TVLVDAISCRIPLVSDIPTI---------
```

11.9.2 Writing multiple alignments

Biopython provides a function `Bio.AlignIO.write()` to write alignments to file. The basic syntax is:

```
N = Bio.AlignIO.write(alignments,outfile,file_format)
```

where `alignments` are a MultipleSeqAlignment object with the alignments to write to the output file with name `outfile` that has format `file_format` (a low case string with the file format). N is the number of entries written to the file.

Example: Let's create some multiple sequence alignments and store them in phylip[78] format. The output file is present here my_malign.phy. Finally, read and print the content of the file.

In [19]:
```
from Bio.Alphabet import generic_dna
from Bio.Seq import Seq
from Bio.SeqRecord import SeqRecord
from Bio.Align import MultipleSeqAlignment
from Bio import AlignIO

align1 = MultipleSeqAlignment([
SeqRecord(Seq("ACTGCTAGCTAG", generic_dna), id="Alpha"),
SeqRecord(Seq("ACT-CTAGCTAG", generic_dna), id="Beta"),
SeqRecord(Seq("ACTGCTAGDTAG", generic_dna), id="Gamma"),
])
align2 = MultipleSeqAlignment([
SeqRecord(Seq("GTCAGC-AG", generic_dna), id="Delta"),
SeqRecord(Seq("GACAGCTAG", generic_dna), id="Epsilon"),
SeqRecord(Seq("GTCAGCTAG", generic_dna), id="Zeta"),
])
align3 = MultipleSeqAlignment([
SeqRecord(Seq("ACTAGTACAGCTG", generic_dna), id="Eta"),
SeqRecord(Seq("ACTAGTACAGCT-", generic_dna), id="Theta"),
SeqRecord(Seq("-CTACTACAGGTG", generic_dna), id="Iota"),
])

my_alignments = [align1, align2, align3]
N = AlignIO.write(my_alignments, "file_samples/my_malign.phy", "phylip")

print("{} entries written to the file".format(N))
print("")
alignments = AlignIO.parse("file_samples/my_malign.phy", "phylip")
for align in alignments:
    print(align)
```

```
3 entries written to the file

SingleLetterAlphabet() alignment with 3 rows and 12 columns
ACTGCTAGCTAG Alpha
ACT-CTAGCTAG Beta
ACTGCTAGDTAG Gamma
SingleLetterAlphabet() alignment with 3 rows and 9 columns
GTCAGC-AG Delta
```

[78] http://www.phylo.org/tools/obsolete/phylip.html

```
GACAGCTAG Epsilon
GTCAGCTAG Zeta
SingleLetterAlphabet() alignment with 3 rows and 13 columns
ACTAGTACAGCTG Eta
ACTAGTACAGCT- Theta
-CTACTACAGGTG Iota
```

Note that it is also possible to convert one format into the other (provided that all information needed for the second format is available) by using the function:

```
Bio.AlignIO.convert(input_file, input_file_format, output_file, output_file_format)
```

basically by passing the input file name and format and output file name and format.

Ex:

```
Bio.AlignIO.convert("PF05371_seed.sth", "stockholm", "PF05371_seed.aln", "clustal")
```

converts a stockholm file into a clustal.

11.9.3 Manipulating alignments

It is possible to slice alignments using the [] operator applied on a `SeqRecord`. In particular:

1. `SeqRecord[i,j]` returns the jth character of alignment i as a string;
2. `SeqRecord[:,j]` returns all the jth characters of the multiple alignment as a string;
3. `SeqRecord[:,i:j]` returns a MultipleSeqAlignment with the sub-alignments going for i to j (excluded)
4. `SeqRecord[a:b,i:j]` similar to 3. but for alignments going from a to b (excluded) only

One can also combine different portions of the alignment by slicing and adding the slices together with

```
SeqRecord[:,i:j] + SeqRecord[:,t:k]
```

Example: Load the seed alignment of the Piwi (PF02171) family[79] stored in the pfam (stockholm) format PF02171_seed.sth. For each record print the description, the first 20 bases and the final 20 bases of the sequence.

```
In [20]: from Bio import AlignIO

         alignments = AlignIO.read("file_samples/PF02171_seed.sth", "stockholm")

         edited_aligns = alignments[:,0:20] + alignments[:,-20:]


         for align in edited_aligns:
             print(align.description, "\t", align.seq)
```

```
AGO1_SCHPO/500-799      YLFFILDK-NSPEP-YGSIKLVPPVYYAHLVSNLARYQDV
AGO6_ARATH/541-851      FILCILPERKTSDI-YGPWKIVAPVRYAHLAAAQVAQFTK
AGO4_ARATH/577-885      FILCVLPDKKNSDL-YGPWKVVAPICYAHLAAAQLGTFMK
TAG76_CAEEL/660-966     CIIVVLQS-KNSDI-YMTVKIPTPVYYADLVATRARCHVK
O16720_CAEEL/566-867    LIVVVLPG--KTPI-YAEVKIPAPAYYAHLVAFRARYHLV
O62275_CAEEL/594-924    TFVFIITD-DSITT-LHQRYLPTPLYVANEYAKRGRNLWN
YQ53_CAEEL/650-977      DILVGIAR-EKKPD-VHDILVPDVLYAAENLAKRGRNNYK
NRDE3_CAEEL/673-1001    TIVFGIIA-EKRPD-MHDILIPNVSYAAQNLAKRGHNNYK
Q17567_CAEEL/397-708    MLVVMLAD-DNKTR-YDSLKVPAPCQYAHKLAFLTAQSLH
```

[79] http://pfam.xfam.org/family/Piwi

```
AUB_DROME/555-852       IVMVVMRS-PNEEK-YSCIKVPAVCHYAHKLAFLVAESIN
PIWI_DROME/538-829      LILCLVPN-DNAER-YSSIKVPAVCQYAKKLATLVGTNLH
PIWL1_HUMAN/555-847     IVVCLLSS-NRKDK-YDAIKVPAPCQYAHKLAFLVGQSIH
PIWI_ARCFU/110-406      GIMLVLPE-YNTPL-YYKLKLPVTVNYPKLVAGIIANVNR
Y1321_METJA/426-699     CFALIIGKEKYKDNDYYEILIPAPIHYADKFVKALGKNWK
O67434_AQUAE/419-694    LVIVFLEEYPKVDP-YKSFLLPATVHYSDKITKLMLRGIE
AGO10_ARATH/625-946     LLLAILPD-NNGSL-YGDLKIVPPAYYAHLAAFRARFYLE
```

Example: Convert the seed alignment of the Piwi (PF02171) family[80] stored in the pfam (stockholm) format PF02171_seed.sth into phylip format. Print some stats on the data.

```
In [21]: from Bio import AlignIO

         alignments = AlignIO.read("file_samples/PF02171_seed.sth", "stockholm")

         out = AlignIO.convert("file_samples/PF02171_seed.sth",
                               "stockholm",
                               "file_samples/PF02171_seed.aln",
                               "clustal")

         print("N. of seq: {}\nLen of seq: {}".format(
                                                        len(alignments),
                                                        len(alignments[0])))
         print("{} multiple alignments converted to phylip".format(out))
```

```
N. of seq: 16
Len of seq: 395
1 multiple alignments converted to phylip
```

11.9.4 Biopython's pairwise2 alignment

Biopython supports running some classic multiple alignment tools like clustal[81], MUSCLE[82] but these require to be installed in your system first.

Biopython has its own module to make pairwise alignment. It provides two algorithms: Smith-Waterman[83] for local alignment and Needleman-Wunsch[84] for global alignment. These methods are implemented in two Biopython functions of the `Bio.pairwise2` module:

`pairwise2.align.globalxx()` and `pairwise2.align.localxx()`.

The basic syntax of the two methods is the same:

```
aligns = pairwise2.align.globalxx(seq1,seq2)
aligns = pairwise2.align.localxx(seq1,seq2)
```

where seq1 and seq2 are two `str` objects. These methods return a list of alignments (at least one) that have the same **optimal score**. Each alignment is represented as tuples with the following 5 elements in order:

1. The alignment of the first sequence;
2. The alignment of the second sequence;
3. The alignment score;
4. The start of the alignment (for global alignments this is always 0);

[80] http://pfam.xfam.org/family/Piwi
[81] http://www.sciencedirect.com/science/article/pii/0378111988903307?via%3Dihub
[82] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC390337/
[83] https://en.wikipedia.org/wiki/Smith%E2%80%93Waterman_algorithm
[84] https://en.wikipedia.org/wiki/Needleman%E2%80%93Wunsch_algorithm

5. The end of the alignment (for global alignments this is always the length of the alignment).

The **xx** in `globalxx` or `localxx` encodes the penalties and scores for gaps and matches (and mismatches) of the alignment. The first letter is **the score for a match** the second letter is **the penalty for a gap**:

Match parameters can be:

- `x` : means that a match scores 1 a mismatch 0;
- `m` : the match and mismatch score are passed as additional params after the sequence (es. `aligns = pairwise2.align.globalmx(seq1,seq2, 1, -1)` to set 1 as match score and -1 as mismatch penalty.

Gap parameters can be:

- `x` : gap penalty is 0;
- `s` : same gap open and gap extend penalties for the 2 sequences (passed as additional params after seqs).
- `d` : different gap open and gap extend penalties for the 2 seqs (additional params after the seqs).

Example:

Let's perform the alignment of the two sequences "ACCGTTATATAGGCCA" and "ACGTACTAGTATAGGCCA"

```
In [34]: from Bio import pairwise2
         from Bio import SeqIO


         alignments = pairwise2.align.globalxx("ACCGTTATATAGGCCA",
                                               "ACGTACTAGTATAGGCCA")

         for i in range(len(alignments)):
             print(alignments[i])

         print("")
         print("Looping through aligns")
         for align in alignments:
                 print(align[0])
                 print(align[1])
                 print("Score: {}, Start: {}, End: {}".format(align[2],
                                                              align[3],
                                                              align[4]))
                 print("")

         alignments = pairwise2.align.globalms("ACCGTTATATAGGCCA",
                                               "ACGTACTAGTATAGGCCA",
                                               1,-1,-0.5,-0.2)
         print("")
         print("Match: 1, Mismatch: -1, Gap open: -0.5, Gap extend: -0.2")
         for align in alignments:
             print(align[0])
             print(align[1])
             print("Score: {}, Start: {}, End: {}".format(align[2],
                                                          align[3],
                                                          align[4]))
             print("")
```

```
('ACCGT--TA-TATAGGCCA', 'A-CGTACTAGTATAGGCCA', 15.0, 0, 19)
('ACCGT--TA-TATAGGCCA', 'AC-GTACTAGTATAGGCCA', 15.0, 0, 19)

Looping through aligns
```

```
ACCGT--TA-TATAGGCCA
A-CGTACTAGTATAGGCCA
Score: 15.0, Start: 0, End: 19

ACCGT--TA-TATAGGCCA
AC-GTACTAGTATAGGCCA
Score: 15.0, Start: 0, End: 19


Match: 1, Mismatch: -1, Gap open: -0.5, Gap extend: -0.2
ACCGT--TA-TATAGGCCA
A-CGTACTAGTATAGGCCA
Score: 13.5, Start: 0, End: 19

ACCGT--TA-TATAGGCCA
AC-GTACTAGTATAGGCCA
Score: 13.5, Start: 0, End: 19
```

It is also possible to specify a substitution matrix like the BLOSUM[85] to improve the alignment of protein sequences. Substitution matrices like the **blosum62** are present in Biopython and before using it we need to import it with `from Bio.SubsMat.MatrixInfo import blosum62`.

You can find all the substitution matrices available here[86].

Example: Align the alpha and beta subunits of the human hemoglobin protein stored in hem_alpha.fasta and hem_beta.fasta with globalxx and with a blosum62 substitution matrix and gap open -10 and gap extension -0.5.

```
In [23]: from Bio import pairwise2
         from Bio import SeqIO
         from Bio.pairwise2 import format_alignment
         from Bio.SubsMat.MatrixInfo import blosum62



         seq1 = SeqIO.read("file_samples/hem_alpha.fasta", "fasta")
         seq2 = SeqIO.read("file_samples/hem_beta.fasta", "fasta")


         alignments = pairwise2.align.globalxx(seq1.seq,seq2.seq)
         print(len(alignments), "optimal alignments")

         align = alignments[0]
         print(align[0])
         print(align[1])
         print("Score: {}, Start: {}, End: {}".format(align[2],align[3],align[4]))
         print("")

         alignments = pairwise2.align.globalds(seq1.seq, seq2.seq, blosum62, -10, -0.5)
         print(len(alignments), "optimal alignments")

         align = alignments[0]
         print(align[0])
         print(align[1])
         print("Score: {}, Start: {}, End: {}".format(align[2],align[3],align[4]))
```

```
80 optimal alignments
```

[85] https://en.wikipedia.org/wiki/BLOSUM
[86] http://biopython.org/DIST/docs/api/

```
MV-LSPADKTNV---K-A--A-WGKVGAHAG---EY-GA-EALE-RMFLSF----PTTK-TY--FPHFDL-SH-G--S---AQVK-G------HGKKV--A
MVHL-----T--PEEKSAVTALWGKV-----NVDE-VG-GEAL-GR--L--LVVYP---WT-QRF--F--ES-FGDLSTPDA-V-MGNPKVKAHGKKVLGA
Score: 72.0, Start: 0, End: 217

2 optimal alignments
MV-LSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTYFPHF-DLS-----HGSAQVKGHGKKVADALTNAVAHVDDMPNALSALSDLHAHKLRV
MVHLTPEEKSAVTALWGKV--NVDEVGGEALGRLLVVYPWTQRFFESFGDLSTPDAVMGNPKVKAHGKKVLGAFSDGLAHLDNLKGTFATLSELHCDKLHV
Score: 292.5, Start: 0, End: 149
```

11.10 Exercises

1. Write a python function that reads a genebank file given in input and prints off the following information:
   (a) Identifier, name and description;
   (b) The first 100 characters of the sequence;
   (c) Number of external references (dbxrefs) and ids of the external refs.
   (d) The name of the organism (hint: check the annotations dictionary at the key "organism")
   (e) Retrieve and print all (if any) associated publications (hint: annotation dictionary, key:"references")
   (f) Retrieve and print all the locations of "CDS" features of the sequence (hint: check the features)

Hint: go back and check the details of the `SeqRecord` object.

Test the program downloading some files from genebank like this[87] or this[88]

Show/Hide Solution

```
In [12]: %reset -s -f

         from Bio import SeqIO

         def getGenBankInfo(filename):
             record = SeqIO.read(filename, "genbank")
             #print(record.annotations.keys())
             #print(record.annotations)

             ident = record.id
             name = record.name
             desc = record.description
             numDbrefs = len(record.dbxrefs)
             dbrefs = ",".join(record.dbxrefs)
             organism = record.annotations["organism"]
             pubs = record.annotations["references"]

             print("ID:{}".format(ident) )
             print("NAME:{}".format(name))
             print("DESC:{}".format(desc) )
             print("ORGANISM:{}".format(organism))
             print("")
             print("CDS features:")
             for f in record.features:
                 if(f.type == "CDS"):
                     print(f.type, f.location)
             print("num DBrefs {}:".format(numDbrefs))
```

[87] https://www.ncbi.nlm.nih.gov/nuccore/NC_005816.1/
[88] https://www.ncbi.nlm.nih.gov/nuccore/AB026718.1

```
            if(numDbrefs > 0):
                print("Refs:{}".format(dbrefs))
            print("")
            if(len(pubs) > 0):
                print("PUBLICATIONS:")
                pcount = 1
                for p in pubs:
                    #print(p)
                    print("REFERENCE [{}]".format(pcount))
                    print("{}".format(p.title))
                    print("{}".format(p.authors))
                    print("{}".format(p.journal))
                    print("")
                    pcount += 1



        fn = "file_samples/NC_005816.gb"
        getGenBankInfo(fn)
```

```
ID:NC_005816.1
NAME:NC_005816
DESC:Yersinia pestis biovar Microtus str. 91001 plasmid pPCP1, complete sequence
ORGANISM:Yersinia pestis biovar Microtus str. 91001

CDS features:
CDS [86:1109](+)
CDS [1105:1888](+)
CDS [2924:3119](+)
CDS [<3358:3442](+)
CDS [4354:4780](+)
CDS [4814:5888](-)
CDS [6004:6421](+)
CDS [6663:7602](+)
CDS [7788:8088](-)
CDS [8087:8435](-)
CDS [8507:8639](-)
CDS [9264:9558](-)
num DBrefs 3:
Refs:BioProject:PRJNA224116,BioSample:SAMN02602970,Assembly:GCF_000007885.1

PUBLICATIONS:
REFERENCE [1]
Genetics of metabolic variations between Yersinia pestis biovars and the proposal of a new biovar, mi
Zhou,D., Tong,Z., Song,Y., Han,Y., Pei,D., Pang,X., Zhai,J., Li,M., Cui,B., Qi,Z., Jin,L., Dai,R., Du
J. Bacteriol. 186 (15), 5147-5152 (2004)

REFERENCE [2]
Complete genome sequence of Yersinia pestis strain 91001, an isolate avirulent to humans
Song,Y., Tong,Z., Wang,J., Wang,L., Guo,Z., Han,Y., Zhang,J., Pei,D., Zhou,D., Qin,H., Pang,X., Han,Y
DNA Res. 11 (3), 179-197 (2004)

REFERENCE [3]
Direct Submission
Song,Y., Tong,Z., Wang,L., Han,Y., Zhang,J., Pei,D., Wang,J., Zhou,D., Han,Y., Pang,X., Zhai,J., Chen
Submitted (24-APR-2003) The Institute of Microbiology and Epidemiology, Academy of Military Medical S
```

2. Write a python program that loads a pfam file (stockholm format .sth) and reports for each record of the align-

ment:

(a) the id of the entry

(b) the start and end points

(c) the number of gaps and the % of gaps on the total length of the alignment

(d) the number of external database references (dbxrefs), and the first 3 external references comma separated (hint: use join).

Print these information to the screen. Finally, write this information in a tab separated file (.tsv) having the following format: `#ID\tstart\tend\tnum_gaps\tpercentage_gaps\tdbxrefs`.

You can test the program using the TP53[89] family file PF00870_seed.sth.

Show/Hide Solution

```
In [25]: %reset  -s -f

         from Bio import AlignIO

         def getStockholmInfo(fn):

             alignments = AlignIO.read(fn, "stockholm")
             print(alignments)
             retVals = []
             for align in alignments:
                 ident = align.id
                 start = align.annotations["start"]
                 end = align.annotations["end"]
                 gaps = align.seq.count("-")
                 #percentage gaps
                 percGaps = 100*gaps/(end - start)
                 dbxrfsN = len(align.dbxrefs)
                 #get top five external refs
                 dbxrfs = ",".join(align.dbxrefs[0:3])
                 print("ID: {} start: {} end: {} gaps: {} % gaps: {:.2f} refs: {} {}".format(
                     ident, start, end, gaps, percGaps, dbxrfsN, dbxrfs))
                 retVals.append([ident, start, end, gaps, "{:.2f}".format(percGaps), dbxrfsN, dbxrfs])
             return retVals

         def writeStockholmInfo(outFile, data):
             fh = open(outFile, "w")
             fh.write("#ID\tstart\tend\tnum_gaps\tpercentage_gaps\tdbxrefs\n")
             for entry in data:
                 fh.write("\t".join([str(x) for x in entry]) + "\n")
             fh.close()
             print("Written {} entries.".format(len(data)))

         file = "file_samples/PF00870_seed.sth"
         outFile = "file_samples/PF00870_seed.tsv"

         infos = getStockholmInfo(file)
         print("")
         writeStockholmInfo(outFile, infos)


SingleLetterAlphabet() alignment with 5 rows and 197 columns
SSVPSQKTYQGSYGFRLGFLHSGTAKSVTCTYSPALNKMFCQLA...ENL P53_HUMAN/95-289
```

[89] https://en.wikipedia.org/wiki/TP53

```
STVPETSDYPGDHGFRLRFPQSGTAKSVTCTYSPDLNKLFCQLA...SNF P53_DANRE/63-257
-AVPSTEDYAGSYGLKLEFQQNGTAKSVTCTYSTDLNKLFCQLA...DN- F7A9U0_XENTR/70-263
PVIPSNTDYPGPHHFEVTFQQSSTAKSATWTYSPLLKKLYCQIA...DHY P73_HUMAN/113-309
TTVPVTTDYPGSYELELRFQKSGTAKSVTSTYSETLNKLYCQLA...ESR P53_ORYLA/80-270
ID: P53_HUMAN/95-289 start: 95 end: 289 gaps: 2 % gaps: 1.03 refs: 210 PDB; 3Q06 B; 96-289;,PDB; 4MZI
ID: P53_DANRE/63-257 start: 63 end: 257 gaps: 2 % gaps: 1.03 refs: 0
ID: F7A9U0_XENTR/70-263 start: 70 end: 263 gaps: 3 % gaps: 1.55 refs: 0
ID: P73_HUMAN/113-309 start: 113 end: 309 gaps: 0 % gaps: 0.00 refs: 45 PDB; 4A63 E; 114-309;,PDB; 4C
ID: P53_ORYLA/80-270 start: 80 end: 270 gaps: 6 % gaps: 3.16 refs: 0

Written 5 entries.
```

3. Given a multiple sequence alignment stored in "phylip" format, write three methods: *readAlignment* that reads the input file and prints the number of sequences present, *printAlignments(alignments)* that prints the alignments to the screen and *computeConsensus(alignments,minFrequency)* that creates the consensus of all the alignments. MinFrequency is the minimum frequency (that has to be > 0.5) to keep a base in the consensus. "?" is put put if frequency < minFrequency for all possible bases.

Ex. if alignments are:

```
ATC-G
AAC-G
AAG-G
ATCGT
```

computeConsensus(alignments,0.6) is:

```
A?C-G
```

Test the script with the file alpha-globin.phy.

Show/Hide Solution

```
In [26]: %reset -s -f
         """
         NOTE:
         Three different solutions are provided for computeConsensus.
         One with standard methods, one with set and one with collections.Counter
         """


         from Bio import AlignIO
         #needed for computeConsensusV3
         from collections import Counter

         def readAlignment(fn):

             alignments = AlignIO.read(fn, "phylip")
             print("{} alignments present in \"{}\"\n".format(len(alignments),fn))
             return alignments

         def printAlignments(aldata):
             for align in aldata:
                 print("{}\n{}".format(align.id, align.seq))

         def computeConsensus(aldata, minFreq = 0.51):
             #this solution uses sets.
             #set(list("ATTTAC")) returns {'A', 'T', 'C'}
             consensus = ""
             if(minFreq < 0.51):
```

```
            print("Warning: minFreq ({}) not valid!".format(minFreq))
            return None
        for j in range(len(aldata[0])):
            chrSeq = aldata[:,j]
            singleChars = [(x,chrSeq.count(x)) for x in chrSeq ]
            if(len(singleChars) == 1):
                consensus +=chrSeq[0][0]
            else:
                cons = [x[0] for x in singleChars if x[1] > minFreq*len(chrSeq)]
                if(len(cons) == 1):
                    consensus +=cons[0]
                else:
                    consensus += "?"
        return consensus

    def computeConsensusV2(aldata, minFreq = 0.51):
        #this solution uses sets.
        #set(list("ATTTAC")) returns {'A', 'T', 'C'}
        consensus = ""
        if(minFreq < 0.51):
            print("Warning: minFreq ({}) not valid!".format(minFreq))
            return None
        for j in range(len(aldata[0])):
            chrSeq = aldata[:,j]
            singleChars = set(list(chrSeq))
            if(len(singleChars) == 1):
                consensus +=chrSeq[0]
            else:
                cons = [x for x in singleChars if chrSeq.count(x) > minFreq*len(chrSeq)]
                if(len(cons) == 1):
                    consensus +=cons[0]
                else:
                    consensus += "?"
        return consensus

    def computeConsensusV3(aldata, minFreq = 0.51):
        #this solution uses collections.Counter
        #collections.Counter("ATTTAC") returns the dictionary: {'T': 3, 'A': 2, 'C': 1}
        #
        consensus = ""
        if(minFreq < 0.51):
            print("Warning: minFreq ({}) not valid!".format(minFreq))
            return None
        for j in range(len(aldata[0])):
            chrSeq = aldata[:,j]
            singleChars = Counter(chrSeq)
            if(len(singleChars) == 1):
                consensus +=chrSeq[0]
            else:
                cons = [x for x in singleChars if singleChars[x] > minFreq*len(chrSeq)]
                if(len(cons) == 1):
                    consensus +=cons[0]
                else:
                    consensus += "?"
        return consensus

    file = "file_samples/alpha-globin.phy"
```

```
        als = readAlignment(file)
        printAlignments(als)
        out = computeConsensusV2(als,minFreq = 0.7)
        print("Consensus:\n{}".format(out))
        #out1 = computeConsensusV2(als,minFreq = 0.7)
        #print("Consensus:\n{}".format(out1))
        #out2 = computeConsensusV3(als,minFreq = 0.7)
        #print("Consensus:\n{}".format(out2))
        #assert(out1 == out2)
        #assert(out == out1)
```

```
4 alignments present in "file_samples/alpha-globin.phy"

ENA|BAA205
ATGAGTCTCTCTGATAAGGACAAGGCTGCTGTGAAAGCCCTATGGGCTAAGATCAGCCCCAAAGCCGATGATATCGGCGCTGAAGCTCTCGGCAGAATGCT
ENA|CAA284
ATGTCTCTGACCAGGACTGAGAGGACCATCATCCTGTCCCTGTGGAGCAAGATCTCCACACAGGCAGACGTCATTGGCACCGAGACCCTGGAGAGGCTCTT
ENA|CAA237
ATGGTGCTGTCTCCTGCCGACAAGACCAACGTCAAGGCCGCCTGGGGTAAGGTCGGCGCGCACGCTGGCGAGTATGGTGCGGAGGCCCTGGAGAGGATGTT
ENA|CAA240
ATGGTGCTCTCTGGGGAAGACAAAAGCAACATCAAGGCTGCCTGGGGGAAGATTGGTGGCCATGGTGCTGAATATGGAGCTGAAGCCCTGGAAAGGATGTT
Consensus:
ATG???CT?TCT??????GACAAGACCA?C?TCAAGGCC???TGGGG?AAGATC?GC?C?CA?GC?G??GA???TGG?GC?GA?GCCCTGGA?AGGATGTT
```

4. Load the contigs present in the filtered_contigs.fasta file and translate each DNA sequence into the corresponding protein. Count the number of stop codons (i.e. *) for each sequence and print them to the user (e.g. MDC020656.85 51). Finally, write the translated proteins in another .fasta file (e.g. filtered_contigs_translated.fasta).

Show/Hide Solution

```
In [27]: %reset -s -f

        from Bio import SeqIO

        seqIterator = SeqIO.parse("file_samples/filtered_contigs.fasta", "fasta")

        sRcds = []
        for s in seqIterator:
            #to avoid problems if not divisible by 3!
            seq1 = s.seq.tomutable()
            if(len(seq1) % 3 != 0):
                while( len(seq1) % 3 != 0):
                    seq1 +="N"
            transl = seq1.toseq().translate()
            stop_cnt = transl.count("*")
            print(s.id,"\t", stop_cnt)
            s.seq = transl
            sRcds.append(s)

        N = SeqIO.write(sRcds,"file_samples/filtered_contigs_translated.fasta", "fasta")
        print("")
        print("{} sequences written to output file".format(N))
```

```
MDC001115.177    65
MDC013284.379    93
MDC018185.243    418
MDC018185.241    467
MDC004527.213    48
MDC012176.157    30
MDC001204.810    155
```

```
MDC004389.256    43
MDC018297.229    317
MDC001802.364    45
MDC014057.243    23
MDC021015.302    19
MDC017187.314    83
MDC012865.410    68
MDC000427.83     42
MDC017187.319    251
MDC004364.265    145
MDC002360.219    60
MDC015155.172    47
MDC019140.398    67
MDC019140.399    56
MDC011390.337    71
MDC007154.375    248
MDC010588.505    29
MDC002519.240    500
MDC006346.711    376
MDC011551.182    806
MDC002717.156    55
MDC006346.719    157
MDC007838.447    141
MDC007018.186    47
MDC017873.233    35
MDC016296.138    176
MDC019067.226    49
MDC014019.318    45
MDC007995.528    53
MDC026961.60     71
MDC020963.161    65
MDC005174.220    132
MDC040033.7      142
MDC019674.147    89
MDC010450.877    40
MDC007097.457    89
MDC016278.70     69

44 sequences written to output file
```

5. Write the following three functions:

   (a) `countMatches(s1,s2)` that takes two sequences aligned with pairwise2 local or global of the same length and returns how many elements match;

   (b) `countMismatches(s1,s2)` that takes two sequences aligned with pairwise2 local or global of the same length and returns how many elements are different (i.e. they are not gaps but the character is different);

   (c) `countGapOpens(s1,s2)` that takes two sequences aligned with pairwise2 local or global of the same length and returns the number of gap opened in the alignment;

   (d) `countGapExtensions(s1,s2)` that takes two sequences aligned with pairwise2 local or global of the same length and returns the number of gap extensions in the alignment

   (e) `getScore(s1,s2, matchScore, mismatchPenalty, gapOpenPenalty, gapExtensionPenalty)` and returns the score of the alignment of s1 and s2 (aligned with pairwise2 as before) given the scores and penalties in input.

Finally, align the sequences of the Interleukin-12[90] chain A (let's call it s1) IL12A.fasta and B IL12B.fasta (let's call it s2) with the following parameters:

```
pairwise2.align.globalxx(s1,s2)
pairwise2.align.globalms(s1,s2,,1,-0.1,-1,-0.1)
pairwise2.align.globalxs(s1, s2,1,-0.5,-5,-0.5)
pairwise2.align.globalxd(s1, s2,1,-0.5,-5,-0.5,-5,-5)
```

and check the score as computed from pairwise2 and from your functions (use the first of all the alignments to check). Note the effect of changing parameters.

Show/Hide Solution

```
In [2]: %reset -s -f

        from Bio import pairwise2
        from Bio import SeqIO


        def countMatches(a1,a2):
            return sum([1 for i in range(len(a1)) if a1[i] == a2[i]])

        def countMismatches(a1,a2):
            return sum([1 for i in range(len(a1)) if a1[i] != a2[i] and a1[i] != "-" and a2[i] != "-

        def countGapOpens(a1,a2):
            gaps = [1 for i in range(1,len(a1)-1)
                    #if ((a1[i] == "-" or a2[i] == "-") and a1[i-1] != "-" and a2[i-1] !="-")
                    if ((a1[i] == "-" and a1[i-1] != "-") or (a2[i] == "-" and a2[i-1] !="-"))
                   ]
            return sum(gaps)

        def countGapExtensions(a1,a2):
            gaps = [ 1 for i in range(1,len(a1)-1)
                    if ((a1[i] == "-" and a1[i-1] == "-") or (a2[i] == "-" and a2[i-1] =="-"))
                   ]
            return sum(gaps)

        def getScore(s1,s2, matchScore, mismatchPenalty, gapOpenPenalty, gapExtensionPenalty):
            mat = countMatches(s1,s2)
            mis = countMismatches(s1,s2)
            gapo = countGapOpens(s1,s2)
            gape = countGapExtensions(s1,s2)
            score = matchScore*mat + mis*mismatchPenalty + gapo*gapOpenPenalty + gape*gapExtensionPen
            return score

        def getSequenceData(filename):
            record = SeqIO.read(filename, "fasta")
            return record.seq

        def printInfo(a1,a2,computedScore,Mscore,mScore,goScore,geScore):
            print(a1)
            print(a2)
            print(computedScore)
            print("")
            M = countMatches(a1,a2)
            m = countMismatches(a1,a2)
            go = countGapOpens(a1,a2)
```

[90] https://en.wikipedia.org/wiki/Interleukin_12

```
            ge = countGapExtensions(a1,a2)
            print("Matches:{} Mismatches:{} GapOpens:{} GapExtensions:{}".format(M,m,go,ge))
            print("Recomputed Score: {}".format(getScore(a1,a2,Mscore,mScore,goScore,geScore)))

        s1 = getSequenceData("file_samples/IL12A.fasta")
        s2 = getSequenceData("file_samples/IL12B.fasta")

        aligns = pairwise2.align.globalxx(s1,s2)
        a1 = aligns[0][0]
        a2 = aligns[0][1]
        sc = aligns[0][2]
        printInfo(a1,a2,sc, 1,0,0,0)


        print("")
        aligns = pairwise2.align.globalms(s1,s2,1,-0.1,-1,-0.1)
        a1 = aligns[0][0]
        a2 = aligns[0][1]
        sc = aligns[0][2]
        printInfo(a1,a2,sc, 1,-0.1,-1,-0.1)


        print("")
        aligns = pairwise2.align.globalms(s1, s2,1,-0.5,-5,-0.5)

        a1 = aligns[0][0]
        a2 = aligns[0][1]
        sc = aligns[0][2]
        printInfo(a1,a2,sc, 1,-0.5,-5,-0.5)

        aligns = pairwise2.align.globalmd(s1, s2,1,-0.5,-5,-0.5,-5,-5)

        a1 = aligns[0][0]
        a2 = aligns[0][1]
        sc = aligns[0][2]
        printInfo(a1,a2,sc, 1,-0.5,-5,-0.5)
```

```
MCPAR------S---L--L---LVAT---L---VL----LDHLSL----ARNLP---VA--TP-D-P---GMFPC----LHHS-QNLLRAVSNM---LQ--
MC---HQQLVISWFSLVFLASPLVA-IWELKKDV-YVVELD----WYPDA---PGEMV-VLT-CDTPEEDG----ITWTL---DQ------S--SEVL-GS
103.0

Matches:103 Mismatches:0 GapOpens:117 GapExtensions:223
Recomputed Score: 103

MC------PARSL--L---LVA--TL------VLLDHLSLARNLPVA-TPD-P---GMFPCLHHSQNLLRAVSNM-LQKARQTLEF-----YPC------T
MCHQQLVISWFSLVFLASPLVAIWELKKDVYVVELDWYPDAPGEMVVLTCDTPEEDGITWTLDQSSEVL--GSGKTL--TIQVKEFGDAGQYTCHKGGEVI
28.19999999999986

Matches:84 Mismatches:127 GapOpens:34 GapExtensions:91
Recomputed Score: 28.199999999999996

MCPARSLLLVATLVLLDHLSLA----------RNLPVATPDPGMFPCLHHSQNLLRAVSNMLQKARQTL----EFYPCTSEEIDHEDITKDKTSTVEACLP
MCHQQLVISWFSLVFLASPLVAIWELKKDVYVVELDWYPDAPGEMVVLTCDTPEEDGITWTLDQSSEVLGSGKTLTIQVKEFGDAGQYTCHKGGEVLSHSL
-122.0

Matches:46 Mismatches:173 GapOpens:6 GapExtensions:103
Recomputed Score: -122.0
MCPARSLLLVATLVLLDHLSLA----------RNLPVATPDPGMFPCLHHSQNLLRAVSNMLQKARQTL----EFYPCTSEEIDHEDITKDKTSTVEACLP
MCHQQLVISWFSLVFLASPLVAIWELKKDVYVVELDWYPDAPGEMVVLTCDTPEEDGITWTLDQSSEVLGSGKTLTIQVKEFGDAGQYTCHKGGEVLSHSL
```

```
-122.0

Matches:46 Mismatches:173 GapOpens:6 GapExtensions:103
Recomputed Score: -122.0
```

CHAPTER 12

Practical 11

In this practical we will will see some other functionalities of Biopython.

12.1 Slides

The slides of the introduction can be found here: Intro

12.2 Biopython

From the Biopython tutorial: The Biopython Project is an international association of developers of freely available Python tools for computational molecular biology. The goal of Biopython is to make it as easy as possible to use Python for bioinformatics by creating high-quality, reusable modules and classes. Biopython features include parsers for various Bioinformatics file formats (BLAST, Clustalw, FASTA, Genbank,...), access to online services (NCBI, Expasy,...), interfaces to common and not-so-common programs (Clustalw, DSSP, MSMS...), a standard sequence class, various clustering modules, a KD tree data structure etc. and even documentation.

In this practical we will see some features of Biopython but refer to biopython documentation[91] to discover all its features, recipes etc.

These notes are largely based on what is available here[92].

12.3 BLAST

Blast (Basic logical alignment search tool)[93] is a well known tool to find similarities between biological sequences. It compares DNA or protein sequences and calculates the statistical significance of the matches found.

[91] http://biopython.org/wiki/Documentation
[92] http://biopython.org/DIST/docs/tutorial/Tutorial.pdf
[93] https://www.ncbi.nlm.nih.gov/pubmed/2231712

The typical interaction with BLAST sees the user submit some sequences to the tool to get an alignment and then the hits are parsed to obtain information on the matches. Both these steps can be performed from within Biopython. Although it is possible to interact directly with a local installation of BLAST, in this practical we will work with the tool made available by NCBI (available here[94]).

12.3.1 The function qblast

The online version of blast can be accessed through the `Bio.Blast.NCBIWWW.qblast()` function.

It's basic syntax is the following (first import `from Bio.Blast import NCBIWWW`):

```
result_handle = Bio.Blast.NCBIWWW.qblast(blast_program, database, query_str)
```

where `blast_program` is the program to perform the alignment. The options are **blastn, blastp, blastx, tblast or tblastx**. `database` is the database to search against and `query_str` is a string containing the query to search against the database. The query can be a sequence or a fasta file entry or an identifier like a GI number (NCIBI's sequence identification number). Among the others, some optional parameters are the output format (`format_type` that by default is "XML" which is the most stable output format but results can be stored also as text with "Text"). It is also possible to specify an expectation value cut-off to filter out alignments `expect` (the e-value threshold, default value is 10.0).

Some databases to search against are reported below:

Table 2. Contents of the common BLAST sequence databases

Database	Type	Content
nr (nt) default	Nucleotide	All GenBank + EMBL + DDBJ + PDB sequences, excluding sequences from PAT, EST, STS, GSS, WGS, TSA and phase 0, 1 or 2 HTGS sequences, mostly non-redundant.
refseq_rna	Nucleotide	Curated (NM_, NR_) plus predicted (XM_, XR_) sequences from NCBI Reference Sequence Project.
refseq_genomic	Nucleotide	Genomic sequences from NCBI Reference Sequence Project.
refseq_ representative_ genomes	Nucleotide	NCBI RefSeq Reference and Representative genomes across broad taxonomy groups including eukaryotes, bacteria, archaea, viruses and viroids. These genomes are among the best quality genomes available with minimum redundancy - one genome per species for eukaryotes and diverse isolates for the same species for others.
chromosome	Nucleotide	Complete genomes and complete chromosomes from the NCBI Reference Sequence project.
Human G+T	Nucleotide	The genomic sequences plus curated and predicted RNAs from the current build of the human genome.
Mouse G+T	Nucleotide	The genomic sequences plus curated and predicted RNAs from the current build of the mouse genome.
est	Nucleotide	Database of GenBank + EMBL + DDBJ sequences from EST division
HTGS	Nucleotide	Unfinished High Throughput Genomic Sequences; Sequences: phases 0, 1 and 2
wgs	Nucleotide	Assemblies of Whole Genome Shotgun sequences.
pat	Nucleotide	Nucleotides from the Patent division of GenBank.
pdb	Nucleotide	Nucleotide sequences from the 3-dimensional structure records from Protein Data Bank.
TSA	Nucleotide	Transcriptome Shotgun Assemblies, assembled from RNA-seq SRA data
16S microbial	Nucleotide	16S Microbial rRNA sequences from Targeted Loci Project
nr default	Protein	Non-redundant GenBank CDS translations + RefSeq + PDB + SwissProt + PIR + PRF, excluding those in PAT, TSA, and env_nr.
refseq_protein	Protein	Protein sequences from NCBI Reference Sequence project.
swissprot	Protein	Last major release of the UniProtKB/SWISS-PROT protein sequence database (no incremental updates).
Landmark	Protein	The landmark database includes proteomes from representative genomes spanning a wide taxonomic range
pat	Protein	Proteins from the Patent division of GenBank.
pdb	Protein	Protein sequences from the 3-dimensional structure records from the Protein Data Bank.
env_nr	Protein	Protein sequences translated from the CDS annotation of metagenomic nucleotide sequences.
tsa_nr	Protein	Protein sequences translated from CDSs annotated on transcriptome shotgun assemblies.

[94] https://blast.ncbi.nlm.nih.gov/Blast.cgi

The query string can be obtained in different ways, for example it is possible to load sequences from a fasta file with:

```
from Bio.Blast import NCBIWWW
fasta_string = open("myfile.fasta").read()
result_handle = NCBIWWW.qblast("blastn", "nt", fasta_string)
```

or we can give a SeqRecord:

```
from Bio.Blast import NCBIWWW
from Bio import SeqIO
record = SeqIO.read("myfile.fasta", format="fasta")
result_handle = NCBIWWW.qblast("blastn", "nt", record.seq)
```

It is also possible to specify some optional parameters in the `entrez_query` for example we can limit the search to specific organisms with: `entrez_query='"Malus Domestica" [Organism]'`.

Example: Let's align the first 100 bases of the first entry of the file contigs82.fasta to the Malus Domestica genome.

NOTE: this can take several minutes.

```
In [2]: from Bio.Blast import NCBIWWW
        from Bio import SeqIO



        records = SeqIO.parse("file_samples/contigs82.fasta", format="fasta")
        rec = next(records)
        seq = rec.seq[0:100]
        print("Aligning {} [{}] to Malus Dom.".format(rec.id,
                                                      seq[0:10]+"..."+seq[90:101]))
        result_handle = NCBIWWW.qblast("blastn", "nt",
                                       seq,
                                       entrez_query='"Malus Domestica" [Organism]'
                                      )
```

```
Aligning MDC020656.85 [GAGGGGTTTA...TTGGCAGCAA] to Malus Dom.
```

Note that the previous code does not output anything, it just returns a `result_handle`. We need to parse it to get some results.

12.3.2 Parsing qblast output

Once the qblast call returns, it gives the results in a handle object `result_handle` that we can parse or we can write to disk to avoid having to rerun the query other times. If we expect to get one alignment only, we can use the method **read** otherwise (if we have multiple query sequences) we should use the method **parse**:

```
blast_record = NCBIXML.read(result_handle)
```

or

```
blast_records = NCBIXML.parse(result_handle)
```

Note that to use these methods we first need to import the `NCBIXML` module with `from Bio.Blast import NCBIXML`.

These methods are analogous to what seen in the case of SeqIO and AlignIO. In the case of multiple entries we can loop through them with:

```
blast_records = NCBIXML.parse(result_handle)
for record in blast_records:
    #do something with it...
```

or we can retrieve one record at a time with `record = next(blast_records)`.

12.3.3 Saving results to file

To save the results present in the result_handle we can simply write them to file. In case we have only one entry we can read it and write it to file:

```
out_f = open("my_blast_result.xml", "w")
out_f.write(result_handle.read())
out_f.close
result_handle.close()
```

If we have more than one entry we need to loop through all the entries and save them in the file:

```
out_f = open("my_blast_result.xml", "w")
for entry in result_handle.parse():
    out_f.write(entry)
out_f.close
result_handle.close()
```

Example:

Let's BLAST the galactosidase alpha (gi number: 2717) against the human database on NCBI and save the results to file. (**Note that it can take several seconds/minutes to run!**).

```
In [38]: from Bio.Blast import NCBIWWW

         result_handle = NCBIWWW.qblast("blastn", "nt", "2717")


         with open("file_samples/blast_res.xml","w") as out_f:
             out_f.write(result_handle.read())

         result_handle.close()
```

12.3.4 Open a blast .XML file

A BLAST output file can be read by opening the file to get the handler and then parse it with the method **parse** seen above:

```
from Bio.Blast import NCBIXML
result_handle = open("my_blast.xml")
blast_records = NCBIXML.parse(result_handle)
```

This will end up in a handle to the blast results.

12.3.5 The BLAST record class

The `Bio.Blast.Record.Blast` class holds the results of the alignment. In particular it is composed of the following three information:

1. *query*: the identifier of the query (a string).
2. *Descriptions* : a list of Description objects. Each `Description` holds the following information:
   - `Description.title` : a string with the title of the hit;
   - `Description.score` : a float with the score of the alignment;
   - `Description.num_alignments` : an int with the number of alignments with the same subject;
   - `Description.e` : a float with the e-value of the alignment.
3. *Alignments* : a list of Alignment objects. Each `Alignment` holds the following information:
   - `Alignment.title` : a string with the title of the hit (identical to `Description.title`);
   - `Alignment.length` : an int with the length of the alignment;
   - `Alignment.hsps` : a list of HSP objects (High Scoring Pair). Each `HSP` has the following info:
     - `HSP.score` : the BLAST score of the hit
     - `HSP.bits` : the bits score of the hit (x: on average 2^x pairs to find such a good hit by chance)
     - `HSP.expect` : the evalue of the hit
     - `HSP.num_alignments` : the number of alignments for the same subject
     - `HSP.identities` : the number of identities between query and subject
     - `HSP.positives` : the number of identical bases/aminos or having similar chemical properties
     - `HSP.gaps` : the number of gaps between query and subject
     - `HSP.strand` : a **tuple** with (query,subject) strands
     - `HSP.frame` : a **tuple** with the frame shifts
     - `HSP.query/HSP.sbjct` : query/subject sequence
     - `HSP.query_start/HSP.sbjct_start` :query/subject start point
     - `HSP.match` : the match sequence (basically "l" for matches and spaces for mismatches)
     - `HSP.align_length` : the alignment length.

More information on the BLAST record can be found here[95].

Example:

Let's blast the serum albumin sequence (gi number 23307792[96]) on the human genome and report all the information reported by BLAST. (warning might take a while to run!)

```
In [39]: from Bio.Blast import NCBIWWW
         from Bio.Blast import NCBIXML

         result_handle = NCBIWWW.qblast("blastn", "nt", "23307792",
                                        entrez_query='"Homo Sapiens" [Organism]'
                                        )



         for res in NCBIXML.parse(result_handle):
             for d in res.descriptions:

                 print("TITLE:{}\nSCORE:{}\nN.ALIGN:{}\nE-VAL:{}".format(
```

[95] http://biopython.org/DIST/docs/api/Bio.Blast.Record-module.html
[96] https://www.ncbi.nlm.nih.gov/nuccore/AF542069.1

```
                d.title,d.score, d.num_alignments,d.e))

        for a in res.alignments:
            print("Align Title:{}\nAlign Len: {}".format(a.title, a.length))



            for h in a.hsps:
                s = h.score
                b = h.bits
                e = h.expect
                n = h.num_alignments
                i = h.identities
                p = h.positives
                g = h.gaps
                st = h.strand
                f = h.frame
                q = h.query
                sb = h.sbjct
                qs = h.query_start
                ss = h.sbjct_start
                qe = h.query_end
                se = h.sbjct_end
                m = h.match
                al = h.align_length

                print("Score: {} Bits: {} E-val: {}".format(s,b,e))
                print("N.aligns:{} Ident:{} Pos.:{} Gaps:{} Align len:{}".format(
                    n,i,p,g,al))
                print("Strand: {} Frame: {}".format(st,f))
                print("Query:", q, " start:", qs, " end:", qe)
                print("Match:",m)
                print("Subjc:",sb, " start:", ss, " end:", se)


    result_handle.close()
```

```
TITLE:gi|23307792|gb|AF542069.1| Homo sapiens serum albumin (HSA) mRNA, complete cds
SCORE:4352.0
N.ALIGN:1
E-VAL:0.0
TITLE:gi|1519245814|ref|NM_000477.7| Homo sapiens albumin (ALB), mRNA
SCORE:4305.0
N.ALIGN:1
E-VAL:0.0
TITLE:gi|28591|emb|V00495.1| H.sapiens mRNA for serum albumin
SCORE:4253.0
N.ALIGN:2
E-VAL:0.0
TITLE:gi|7770116|gb|AF119840.1|AF119840 Homo sapiens PRO0903 mRNA, complete cds
SCORE:4062.0
N.ALIGN:1
E-VAL:0.0
TITLE:gi|25058738|gb|BC039235.1| Homo sapiens albumin, mRNA (cDNA clone IMAGE:4768004), containing fr
SCORE:4056.0
N.ALIGN:1
E-VAL:0.0
TITLE:gi|23243417|gb|BC036003.1| Homo sapiens albumin, mRNA (cDNA clone MGC:32850 IMAGE:4724105), cor
SCORE:4052.0
```

```
N.ALIGN:1
E-VAL:0.0
TITLE:gi|158258946|dbj|AK292755.1| Homo sapiens cDNA FLJ78413 complete cds, highly similar to Homo sa
SCORE:4036.0
N.ALIGN:1
E-VAL:0.0
TITLE:gi|28589|emb|V00494.1| Human messenger RNA for serum albumin (HSA)
SCORE:4033.0
N.ALIGN:1
E-VAL:0.0
TITLE:gi|21706455|gb|BC034023.1| Homo sapiens albumin, mRNA (cDNA clone MGC:22784 IMAGE:4734617), cor
SCORE:4029.0
N.ALIGN:1
E-VAL:0.0
TITLE:gi|51476389|emb|CR749331.1| Homo sapiens mRNA; cDNA DKFZp779N1935 (from clone DKFZp779N1935)
SCORE:4027.0
N.ALIGN:1
E-VAL:0.0
TITLE:gi|21706470|gb|BC034026.1| Homo sapiens cDNA clone IMAGE:4734794, containing frame-shift errors
SCORE:4010.0
N.ALIGN:1
E-VAL:0.0
TITLE:gi|194391079|dbj|AK298437.1| Homo sapiens cDNA FLJ54371 complete cds, highly similar to Serum a
SCORE:3994.0
N.ALIGN:1
E-VAL:0.0
TITLE:gi|408407943|gb|AF130077.2| Homo sapiens clone FLB9714 PRO2619 mRNA, complete cds
SCORE:3937.0
N.ALIGN:1
E-VAL:0.0
TITLE:gi|62113340|gb|AY960291.1| Homo sapiens serum albumin mRNA, complete cds
SCORE:3907.0
N.ALIGN:1
E-VAL:0.0
TITLE:gi|49176516|gb|AY550967.1| Homo sapiens cell growth inhibiting protein 42 mRNA, complete cds
SCORE:3848.0
N.ALIGN:1
E-VAL:0.0
TITLE:gi|115607206|gb|DQ986150.1| Homo sapiens serum albumin mRNA, complete cds
SCORE:3733.0
N.ALIGN:1
E-VAL:0.0
TITLE:gi|164692402|dbj|AK314794.1| Homo sapiens cDNA, FLJ95666, highly similar to Homo sapiens albumi
SCORE:3676.0
N.ALIGN:1
E-VAL:0.0
TITLE:gi|52001696|gb|AY728024.1| Homo sapiens serum albumin precursor, mRNA, complete cds
SCORE:3630.0
N.ALIGN:1
E-VAL:0.0
TITLE:gi|123981021|gb|DQ891414.2| Synthetic construct clone IMAGE:100004044; FLH176691.01X; RZPDo8390
SCORE:3626.0
N.ALIGN:1
E-VAL:0.0
TITLE:gi|123995824|gb|DQ894588.2| Synthetic construct Homo sapiens clone IMAGE:100009048; FLH176687.0
SCORE:3624.0
N.ALIGN:1
E-VAL:0.0
TITLE:gi|6013426|gb|AF190168.1|AF190168 Homo sapiens serum albumin precursor, mRNA, complete cds
```

```
SCORE:3615.0
N.ALIGN:1
E-VAL:0.0
TITLE:gi|332356379|gb|HQ537426.1| Homo sapiens albumin mRNA, partial cds
SCORE:3471.0
N.ALIGN:1
E-VAL:0.0
TITLE:gi|194391111|dbj|AK298461.1| Homo sapiens cDNA FLJ50830 complete cds, highly similar to Serum a
SCORE:3084.0
N.ALIGN:2
E-VAL:0.0
TITLE:gi|763428|gb|U22961.1|HSU22961 Human mRNA clone with similarity to L-glycerol-3-phosphate:NAD c
SCORE:2748.0
N.ALIGN:1
E-VAL:0.0
TITLE:gi|27692692|gb|BC041789.1| Homo sapiens albumin, mRNA (cDNA clone MGC:32888 IMAGE:4766983), cor
SCORE:2620.0
N.ALIGN:2
E-VAL:0.0
TITLE:gi|164694201|dbj|AK308044.1| Homo sapiens cDNA, FLJ97992
SCORE:2536.0
N.ALIGN:2
E-VAL:0.0
TITLE:gi|7770216|gb|AF119890.1|AF119890 Homo sapiens PRO2675 mRNA, complete cds
SCORE:2488.0
N.ALIGN:1
E-VAL:0.0
TITLE:gi|47118014|gb|AY544124.1| Homo sapiens growth-inhibiting protein 20 mRNA, complete cds
SCORE:2453.0
N.ALIGN:2
E-VAL:0.0
TITLE:gi|7959790|gb|AF116645.1|AF116645 Homo sapiens PRO1708 mRNA, complete cds
SCORE:2003.0
N.ALIGN:1
E-VAL:0.0
TITLE:gi|6650825|gb|AF118090.1|AF118090 Homo sapiens PRO2044 mRNA, complete cds
SCORE:1983.0
N.ALIGN:1
E-VAL:0.0
TITLE:gi|168745965|emb|CU691683.1| Synthetic construct Homo sapiens gateway clone IMAGE:100022016 3'
SCORE:1951.0
N.ALIGN:1
E-VAL:0.0
TITLE:gi|168745964|emb|CU691682.1| Synthetic construct Homo sapiens gateway clone IMAGE:100022016 5'
SCORE:1896.0
N.ALIGN:1
E-VAL:0.0
TITLE:gi|23241674|gb|BC035969.1| Homo sapiens albumin, mRNA (cDNA clone MGC:32581 IMAGE:4714468), cor
SCORE:1767.0
N.ALIGN:2
E-VAL:0.0
TITLE:gi|15679995|gb|BC014308.1|BC014308 Homo sapiens, clone IMAGE:3934797, mRNA, partial cds
SCORE:1571.0
N.ALIGN:1
E-VAL:0.0
TITLE:gi|649152310|gb|KJ905683.1| Synthetic construct Homo sapiens clone ccsbBroadEn_15353 ALB gene,
SCORE:1387.0
N.ALIGN:2
E-VAL:0.0
```

```
TITLE:gi|168740012|emb|CU690239.1| Synthetic construct Homo sapiens gateway clone IMAGE:100022252 3'
SCORE:1182.0
N.ALIGN:1
E-VAL:0.0
TITLE:gi|37181745|gb|AY358313.1| Homo sapiens DNA66677 ALB (UNQ696) mRNA, partial cds
SCORE:1018.0
N.ALIGN:1
E-VAL:0.0
TITLE:gi|168740011|emb|CU690238.1| Synthetic construct Homo sapiens gateway clone IMAGE:100022252 5'
SCORE:982.0
N.ALIGN:2
E-VAL:0.0
TITLE:gi|6642751|gb|AF113014.1|AF113014 Homo sapiens clone FLB3512 mRNA sequence
SCORE:618.0
N.ALIGN:1
E-VAL:1.15187e-155
TITLE:gi|219842221|ref|NG_009291.1| Homo sapiens albumin (ALB), RefSeqGene on chromosome 4
SCORE:581.0
N.ALIGN:15
E-VAL:2.37628e-145
TITLE:gi|152112963|gb|EF649953.1| Homo sapiens albumin (ALB) gene, complete cds
SCORE:581.0
N.ALIGN:15
E-VAL:2.37628e-145
TITLE:gi|19387810|gb|AC108157.3| Homo sapiens BAC clone RP11-580P21 from 4, complete sequence
SCORE:581.0
N.ALIGN:16
E-VAL:2.37628e-145
TITLE:gi|178343|gb|M12523.1|HUMALBGC Human serum albumin (ALB) gene, complete cds
SCORE:581.0
N.ALIGN:15
E-VAL:2.37628e-145
TITLE:gi|1036032687|gb|AH007061.2| Homo sapiens chromosome 4 serum albumin (ALB) gene, partial cds; a
SCORE:579.0
N.ALIGN:1
E-VAL:8.29405e-145
TITLE:gi|408407822|gb|AF116616.2| Homo sapiens PRO0998 mRNA, complete cds
SCORE:577.0
N.ALIGN:1
E-VAL:2.89491e-144
TITLE:gi|338858118|gb|AH002596.2|SEG_HUMALB Homo sapiens albumin (ALB) gene, partial cds
SCORE:573.0
N.ALIGN:2
E-VAL:3.52672e-143
TITLE:gi|1122817323|gb|DQ145726.2| Homo sapiens ARNT-interacting protein 2 (AINP2) mRNA, complete cds
SCORE:404.0
N.ALIGN:1
E-VAL:1.40699e-97
TITLE:gi|32428|emb|X51365.1| Homo sapiens mRNA for albumin (clone pHA19)
SCORE:232.0
N.ALIGN:1
E-VAL:6.8383e-51
TITLE:gi|1338686512|ref|NG_056261.1| Homo sapiens albumin (ALB) 5' regulatory region (LOC111832671) c
SCORE:221.0
N.ALIGN:1
E-VAL:1.23639e-47
TITLE:gi|7959920|gb|AF116711.1|AF116711 Homo sapiens PRO2646 mRNA, complete cds
SCORE:216.0
N.ALIGN:1
```

```
E-VAL:1.50624e-46
Align Title:gi|23307792|gb|AF542069.1| Homo sapiens serum albumin (HSA) mRNA, complete cds
Align Len: 2176
Score: 4352.0 Bits: 3925.42 E-val: 0.0
N.aligns:None Ident:2176 Pos.:2176 Gaps:0 Align len:2176
Strand: (None, None) Frame: (1, 1)
Query: TCTCTTCTGTCAACCCCACGCGCCTTTGGCACAATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTC
Match: ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: TCTCTTCTGTCAACCCCACGCGCCTTTGGCACAATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTC
Align Title:gi|1519245814|ref|NM_000477.7| Homo sapiens albumin (ALB), mRNA
Align Len: 2285
Score: 4305.0 Bits: 3883.04 E-val: 0.0
N.aligns:None Ident:2168 Pos.:2168 Gaps:1 Align len:2177
Strand: (None, None) Frame: (1, 1)
Query: TCTCTTCTGTCAACCCCACGCGCCTTTGGCACAATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTC
Match: |||||||||||||||||| ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: TCTCTTCTGTCAACCCCACACGCCTTTGGCACAATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTC
Align Title:gi|28591|emb|V00495.1| H.sapiens mRNA for serum albumin
Align Len: 2251
Score: 4253.0 Bits: 3836.15 E-val: 0.0
N.aligns:None Ident:2159 Pos.:2159 Gaps:4 Align len:2177
Strand: (None, None) Frame: (1, 1)
Query: TCTCTTCTGTCAACCCCACGCGCCTTTGGCACAATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTC
Match: |||||||||||||||||||  ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: TCTCTTCTGTCAACCCCACGC--CTTTGGCACAATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTC
Score: 103.0 Bits: 94.1598 E-val: 1.32244e-15
N.aligns:None Ident:56 Pos.:56 Gaps:2 Align len:58
Strand: (None, None) Frame: (1, -1)
Query: TGACAATGAAGAGACATTTTTGAAAAAATACTTATATGAAATTGCCAGAAGACATCCT  start: 489  end: 546
Match: |||||  ||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: TGACA--GAAGAGACATTTTTGAAAAAATACTTATATGAAATTGCCAGAAGACATCCT  start: 56  end: 1
Align Title:gi|7770116|gb|AF119840.1|AF119840 Homo sapiens PRO0903 mRNA, complete cds
Align Len: 2720
Score: 4062.0 Bits: 3663.93 E-val: 0.0
N.aligns:None Ident:2043 Pos.:2043 Gaps:0 Align len:2051
Strand: (None, None) Frame: (1, 1)
Query: TCTCTTCTGTCAACCCCACGCGCCTTTGGCACAATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTC
Match: |||||||||||||||||| ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: TCTCTTCTGTCAACCCCACACGCCTTTGGCACAATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTC
Align Title:gi|25058738|gb|BC039235.1| Homo sapiens albumin, mRNA (cDNA clone IMAGE:4768004), contain
Align Len: 2085
Score: 4056.0 Bits: 3658.52 E-val: 0.0
N.aligns:None Ident:2042 Pos.:2042 Gaps:1 Align len:2050
Strand: (None, None) Frame: (1, 1)
Query: TCTCTTCTGTCAACCCCACGCGCCTTTGGCACAATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTC
Match: |||||||||||||||||| ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: TCTCTTCTGTCAACCCCACACGCCTTTGGCACAATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTC
Align Title:gi|23243417|gb|BC036003.1| Homo sapiens albumin, mRNA (cDNA clone MGC:32850 IMAGE:472410
Align Len: 2087
Score: 4052.0 Bits: 3654.91 E-val: 0.0
N.aligns:None Ident:2041 Pos.:2041 Gaps:0 Align len:2051
Strand: (None, None) Frame: (1, 1)
Query: TCTCTTCTGTCAACCCCACGCGCCTTTGGCACAATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTC
Match: |||||||||||||||||| ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: TCTCTTCTGTCAACCCCACACGCCTTTGGCACAATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTC
Align Title:gi|158258946|dbj|AK292755.1| Homo sapiens cDNA FLJ78413 complete cds, highly similar to
Align Len: 2054
Score: 4036.0 Bits: 3640.48 E-val: 0.0
N.aligns:None Ident:2037 Pos.:2037 Gaps:3 Align len:2049
```

```
Strand: (None, None) Frame: (1, 1)
Query: TCTCTTCTGTCAACCCCACGCGCCTTTGGCACAATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTC
Match: |||||||||||||||||| ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: TCTCTTCTGTCAACCCCACACGCCTTTGGCACAATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTC
Align Title:gi|28589|emb|V00494.1| Human messenger RNA for serum albumin (HSA)
Align Len: 2055
Score: 4033.0 Bits: 3637.78 E-val: 0.0
N.aligns:None Ident:2036 Pos.:2036 Gaps:0 Align len:2049
Strand: (None, None) Frame: (1, 1)
Query: TCTCTTCTGTCAACCCCACGCGCCTTTGGCACAATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTC
Match: |||||||||||||||||| ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: TCTCTTCTGTCAACCCCACACGCCTTTGGCACAATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTC
Align Title:gi|21706455|gb|BC034023.1| Homo sapiens albumin, mRNA (cDNA clone MGC:22784 IMAGE:473461
Align Len: 2068
Score: 4029.0 Bits: 3634.17 E-val: 0.0
N.aligns:None Ident:2028 Pos.:2028 Gaps:0 Align len:2037
Strand: (None, None) Frame: (1, 1)
Query: TCTCTTCTGTCAACCCCACGCGCCTTTGGCACAATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTC
Match: |||||||||||||||||| ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: TCTCTTCTGTCAACCCCACACGCCTTTGGCACAATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTC
Align Title:gi|51476389|emb|CR749331.1| Homo sapiens mRNA; cDNA DKFZp779N1935 (from clone DKFZp779N19
Align Len: 2075
Score: 4027.0 Bits: 3632.37 E-val: 0.0
N.aligns:None Ident:2036 Pos.:2036 Gaps:0 Align len:2051
Strand: (None, None) Frame: (1, 1)
Query: TCTCTTCTGTCAACCCCACGCGCCTTTGGCACAATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTC
Match: |||||||||||||||||| ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: TCTCTTCTGTCAACCCCACACGCCTTTGGCACAATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTC
Align Title:gi|21706470|gb|BC034026.1| Homo sapiens cDNA clone IMAGE:4734794, containing frame-shift
Align Len: 2059
Score: 4010.0 Bits: 3617.04 E-val: 0.0
N.aligns:None Ident:2025 Pos.:2025 Gaps:1 Align len:2037
Strand: (None, None) Frame: (1, 1)
Query: TCTCTTCTGTCAACCCCACGCGCCTTTGGCACAATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTC
Match: |||||||||||||||||| ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: TCTCTTCTGTCAACCCCACACGCCTTTGGCACAATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTC
Align Title:gi|194391079|dbj|AK298437.1| Homo sapiens cDNA FLJ54371 complete cds, highly similar to S
Align Len: 2082
Score: 3994.0 Bits: 3602.61 E-val: 0.0
N.aligns:None Ident:2040 Pos.:2040 Gaps:27 Align len:2076
Strand: (None, None) Frame: (1, 1)
Query: TCTCTTCTGTCAACCCCACGCGCCTTTGGCACAATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTC
Match: |||||||||||||||||| ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: TCTCTTCTGTCAACCCCACACGCCTTTGGCACAATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTC
Align Title:gi|408407943|gb|AF130077.2| Homo sapiens clone FLB9714 PRO2619 mRNA, complete cds
Align Len: 2560
Score: 3937.0 Bits: 3551.22 E-val: 0.0
N.aligns:None Ident:1982 Pos.:1982 Gaps:0 Align len:1991
Strand: (None, None) Frame: (1, 1)
Query: CTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTGTGTTTCGTCGAGATGCACACAAGAGTGAGGTTGCTCATCGGTTTAAAGATTTGGGAGAAG
Match: || || ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: CTGTTACTCTTTAGCTCGGCTTATTCCAGGGGTGTGTTTCGTCGAGATGCACACAAGAGTGAGGTTGCTCATCGGTTTAAAGATTTGGGAGAAG
Align Title:gi|62113340|gb|AY960291.1| Homo sapiens serum albumin mRNA, complete cds
Align Len: 1981
Score: 3907.0 Bits: 3524.17 E-val: 0.0
N.aligns:None Ident:1970 Pos.:1970 Gaps:0 Align len:1981
Strand: (None, None) Frame: (1, 1)
Query: CTCTTCTGTCAACCCCACGCGCCTTTGGCACAATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTGT
Match: ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
```

```
Subjc: CTCTTCTGTCAACCCCACGCGCCTTTGGCACAATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTGT
Align Title:gi|49176516|gb|AY550967.1| Homo sapiens cell growth inhibiting protein 42 mRNA, complete
Align Len: 1939
Score: 3848.0 Bits: 3470.97 E-val: 0.0
N.aligns:None Ident:1933 Pos.:1933 Gaps:0 Align len:1939
Strand: (None, None) Frame: (1, 1)
Query: TGGCACAATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTGTGTTTCGTCGAGATGCACACAAGAGT
Match: ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: TGGCACAATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTGTGTTTCGTCGAGATGCACACAAGAGT
Align Title:gi|115607206|gb|DQ986150.1| Homo sapiens serum albumin mRNA, complete cds
Align Len: 1884
Score: 3733.0 Bits: 3367.27 E-val: 0.0
N.aligns:None Ident:1877 Pos.:1877 Gaps:0 Align len:1884
Strand: (None, None) Frame: (1, 1)
Query: CCTTTGGCACAATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTGTGTTTCGTCGAGATGCACACAA
Match: ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: CCTTTGGCACAATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTGTGTTTCGTCGAGATGCACACAA
Align Title:gi|164692402|dbj|AK314794.1| Homo sapiens cDNA, FLJ95666, highly similar to Homo sapiens
Align Len: 1871
Score: 3676.0 Bits: 3315.88 E-val: 0.0
N.aligns:None Ident:1853 Pos.:1853 Gaps:0 Align len:1863
Strand: (None, None) Frame: (1, 1)
Query: TCTCTTCTGTCAACCCCACGCGCCTTTGGCACAATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTC
Match: |||||||||||||||||| ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: TCTCTTCTGTCAACCCCACACGCCTTTGGCACAATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTC
Align Title:gi|52001696|gb|AY728024.1| Homo sapiens serum albumin precursor, mRNA, complete cds
Align Len: 1842
Score: 3630.0 Bits: 3274.4 E-val: 0.0
N.aligns:None Ident:1824 Pos.:1824 Gaps:0 Align len:1830
Strand: (None, None) Frame: (1, 1)
Query: ATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTGTGTTTCGTCGAGATGCACACAAGAGTGAGGTTC
Match: ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: ATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTGTGTTTCGTCGAGATGCACACAAGAGTGAGGTTC
Align Title:gi|123981021|gb|DQ891414.2| Synthetic construct clone IMAGE:100004044; FLH176691.01X; RZP
Align Len: 1870
Score: 3626.0 Bits: 3270.79 E-val: 0.0
N.aligns:None Ident:1825 Pos.:1825 Gaps:0 Align len:1833
Strand: (None, None) Frame: (1, 1)
Query: CACAATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTGTGTTTCGTCGAGATGCACACAAGAGTGAG
Match: ||| ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: CACCATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTGTGTTTCGTCGAGATGCACACAAGAGTGAG
Align Title:gi|123995824|gb|DQ894588.2| Synthetic construct Homo sapiens clone IMAGE:100009048; FLH17
Align Len: 1870
Score: 3624.0 Bits: 3268.99 E-val: 0.0
N.aligns:None Ident:1824 Pos.:1824 Gaps:0 Align len:1832
Strand: (None, None) Frame: (1, 1)
Query: CACAATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTGTGTTTCGTCGAGATGCACACAAGAGTGAG
Match: ||| ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: CACCATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTGTGTTTCGTCGAGATGCACACAAGAGTGAG
Align Title:gi|6013426|gb|AF190168.1|AF190168 Homo sapiens serum albumin precursor, mRNA, complete co
Align Len: 1830
Score: 3615.0 Bits: 3260.88 E-val: 0.0
N.aligns:None Ident:1821 Pos.:1821 Gaps:0 Align len:1830
Strand: (None, None) Frame: (1, 1)
Query: ATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTGTGTTTCGTCGAGATGCACACAAGAGTGAGGTTC
Match: ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: ATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTGTGTTTCGTCGAGATGCACACAAGAGTGAGGTTC
Align Title:gi|332356379|gb|HQ537426.1| Homo sapiens albumin mRNA, partial cds
Align Len: 1758
```

```
Score: 3471.0 Bits: 3131.03 E-val: 0.0
N.aligns:None Ident:1749 Pos.:1749 Gaps:0 Align len:1758
Strand: (None, None) Frame: (1, 1)
Query: GATGCACACAAGAGTGAGGTTGCTCATCGGTTTAAAGATTTGGGAGAAGAAAATTTCAAAGCCTTAGTGTTGATTGCCTTTGCTCAGTATCTTC
Match: |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| |||||||||||||||||||||||||
Subjc: GATGCACACAAGAGTGAGGTTGCTCATCGGTTTAAAGATTTGGGAGAAGAAAATTTCAAAGCCTTGGTGTTGATTGCCTTTGCTCAGTATCTTC
Align Title:gi|194391111|dbj|AK298461.1| Homo sapiens cDNA FLJ50830 complete cds, highly similar to S
Align Len: 1799
Score: 3084.0 Bits: 2782.08 E-val: 0.0
N.aligns:None Ident:1551 Pos.:1551 Gaps:0 Align len:1557
Strand: (None, None) Frame: (1, 1)
Query: AATGAAGAGACATTTTTGAAAAAATACTTATATGAAATTGCCAGAAGACATCCTTACTTCTATGCCCCGGAACTCCTTTTCTTTGCTAAAAGGT
Match: |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| |||||||||||||||||||||||||
Subjc: AATGAAGAGACATTTTTGAAAAAATACTTATATGAAATTGCCAGAAGACATCCTTACTTTTATGCCCCGGAACTCCTTTTCTTTGCTAAAAGGT
Score: 472.0 Bits: 426.881 E-val: 4.90655e-116
N.aligns:None Ident:239 Pos.:239 Gaps:0 Align len:241
Strand: (None, None) Frame: (1, 1)
Query: TCTCTTCTGTCAACCCCACGCGCCTTTGGCACAATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTC
Match: |||||||||||||||||| |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: TCTCTTCTGTCAACCCCACACGCCTTTGGCACAATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTC
Align Title:gi|763428|gb|U22961.1|HSU22961 Human mRNA clone with similarity to L-glycerol-3-phosphate
Align Len: 3239
Score: 2748.0 Bits: 2479.12 E-val: 0.0
N.aligns:None Ident:1386 Pos.:1386 Gaps:0 Align len:1394
Strand: (None, None) Frame: (1, 1)
Query: CTGTCAACCCCACGCGCCTTTGGCACAATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTGTGTTTC
Match: |||||||||||||| |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: CTGTCAACCCCACACGCCTTTGGCACAATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTGTGTTTC
Align Title:gi|27692692|gb|BC041789.1| Homo sapiens albumin, mRNA (cDNA clone MGC:32888 IMAGE:476698
Align Len: 1510
Score: 2620.0 Bits: 2363.7 E-val: 0.0
N.aligns:None Ident:1327 Pos.:1327 Gaps:1 Align len:1337
Strand: (None, None) Frame: (1, 1)
Query: AAAAATTTGG-AGAAAGAGCTTTCAAAGCATGGGCAGTAGCTCGCCTGAGCCAGAGATTTCCCAAAGCTGAGTTTGCAGAAGTTTCCAAGTTAC
Match: ||| |||||| |||| | ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: AAAGATTTGGGAGAAGAAAATTTCAAAGCATGGGCAGTAGCTCGCCTGAGCCAGAGATTTCCCAAAGCTGAGTTTGCAGAAGTTTCCAAGTTAC
Score: 329.0 Bits: 297.94 E-val: 5.98796e-77
N.aligns:None Ident:166 Pos.:166 Gaps:0 Align len:167
Strand: (None, None) Frame: (1, 1)
Query: TCTCTTCTGTCAACCCCACGCGCCTTTGGCACAATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTC
Match: |||||||||||||||||| |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: TCTCTTCTGTCAACCCCACACGCCTTTGGCACAATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTC
Align Title:gi|164694201|dbj|AK308044.1| Homo sapiens cDNA, FLJ97992
Align Len: 1544
Score: 2536.0 Bits: 2287.96 E-val: 0.0
N.aligns:None Ident:1279 Pos.:1279 Gaps:1 Align len:1285
Strand: (None, None) Frame: (1, 1)
Query: TGCAAAACAAGAACCTGAGAGAAATGAATGCTTCTTGCAACACAAAGATGACAACCCAAACCTCCCCCGATTGGTGAGACCAGAGGTTGATGTC
Match: ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: TGCAAAACAAGAACCTGAGAGAAATGAATGCTTCTTGCAACACAAAGATGACAACCCAAACCTCCCCCGATTGGTGAGACCAGAGGTTGATGTC
Score: 506.0 Bits: 457.539 E-val: 2.89746e-125
N.aligns:None Ident:256 Pos.:256 Gaps:0 Align len:258
Strand: (None, None) Frame: (1, 1)
Query: TCTCTTCTGTCAACCCCACGCGCCTTTGGCACAATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTC
Match: |||||||||||||||||| |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: TCTCTTCTGTCAACCCCACACGCCTTTGGCACAATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTC
Align Title:gi|7770216|gb|AF119890.1|AF119890 Homo sapiens PRO2675 mRNA, complete cds
Align Len: 1618
Score: 2488.0 Bits: 2244.68 E-val: 0.0
```

```
N.aligns:None Ident:1247 Pos.:1247 Gaps:0 Align len:1249
Strand: (None, None) Frame: (1, 1)
Query: AGTTAGTGACAGATCTTACCAAAGTCCACACGGAATGCTGCCATGGAGATCTGCTTGAATGTGCTGATGACAGGGCGGACCTTGCCAAGTATAT
Match: ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: AGTTAGTGACAGATCTTACCAAAGTCCACACGGAATGCTGCCATGGAGATCTGCTTGAATGTGCTGATGACAGGGCGGACCTTGCCAAGTATAT
Align Title:gi|47118014|gb|AY544124.1| Homo sapiens growth-inhibiting protein 20 mRNA, complete cds
Align Len: 1363
Score: 2453.0 Bits: 2213.12 E-val: 0.0
N.aligns:None Ident:1242 Pos.:1242 Gaps:1 Align len:1251
Strand: (None, None) Frame: (1, 1)
Query: AAAAATTTGG-AGAAAGAGCTTTCAAAGCATGGGCAGTAGCTCGCCTGAGCCAGAGATTTCCCAAAGCTGAGTTTGCAGAAGTTTCCAAGTTAC
Match: ||| |||||| ||||  |  ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: AAAGATTTGGGAGAAGAAAATTTCAAAGCATGGGCAGTAGCTCGCCTGAGCCAGAGATTTCCCAAAGCTGAGTTTGCAGAAGTTTCCAAGTTAC
Score: 282.0 Bits: 255.561 E-val: 1.83335e-64
N.aligns:None Ident:141 Pos.:141 Gaps:0 Align len:141
Strand: (None, None) Frame: (1, 1)
Query: TGGCACAATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTGTGTTTCGTCGAGATGCACACAAGAGT
Match: ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: TGGCACAATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTGTGTTTCGTCGAGATGCACACAAGAGT
Align Title:gi|7959790|gb|AF116645.1|AF116645 Homo sapiens PRO1708 mRNA, complete cds
Align Len: 1251
Score: 2003.0 Bits: 1807.36 E-val: 0.0
N.aligns:None Ident:1006 Pos.:1006 Gaps:0 Align len:1009
Strand: (None, None) Frame: (1, 1)
Query: TTTTGTTGAAAGTAAGGATGTTTGCAAAAACTATGCTGAGGCAAAGGATGTCTTCCTGGGCATGTTTTTGTATGAATATGCAAGAAGGCATCCT
Match: ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: TTTTGTTGAAAGTAAGGATGTTTGCAAAAACTATGCTGAGGCAAAGGATGTCTTCCTGGGCATGTTTTTGTATGAATATGCAAGAAGGCATCCT
Align Title:gi|6650825|gb|AF118090.1|AF118090 Homo sapiens PRO2044 mRNA, complete cds
Align Len: 1759
Score: 1983.0 Bits: 1789.33 E-val: 0.0
N.aligns:None Ident:996 Pos.:996 Gaps:0 Align len:999
Strand: (None, None) Frame: (1, 1)
Query: CAAAAACTATGCTGAGGCAAAGGATGTCTTCCTGGGCATGTTTTTGTATGAATATGCAAGAAGGCATCCTGATTACTCTGTCGTGCTGCTGCTG
Match: ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: CAAAAACTATGCTGAGGCAAAGGATGTCTTCCTGGGCATGTTTTTGTATGAATATGCAAGAAGGCATCCTGATTACTCTGTCGTGCTGCTGCTG
Align Title:gi|168745965|emb|CU691683.1| Synthetic construct Homo sapiens gateway clone IMAGE:100022
Align Len: 1128
Score: 1951.0 Bits: 1760.47 E-val: 0.0
N.aligns:None Ident:1034 Pos.:1034 Gaps:9 Align len:1061
Strand: (None, None) Frame: (1, -1)
Query: TGTGCTGATGACAGGGCGGACCTTGCCAAGTATAT-CTGTGAAAATCAAGATT-CGATCTCCAGTAAACT-GAAGGAATGCTGTGAAAAACCTG
Match: || |||||  ||||||| |||||||||||||| | | |||  |||||||||||| |||| |||||||||| |||||||||| ||||||||||
Subjc: TGGGCTGAGAACAGGGCCGACCTTGCCAAGTTTTTTCTGGAAAAATCAAGATTTCGATTTCCAGTAAACTAGAAGGAATGC-GTGAAAAACCTT
Align Title:gi|168745964|emb|CU691682.1| Synthetic construct Homo sapiens gateway clone IMAGE:100022
Align Len: 1251
Score: 1896.0 Bits: 1710.88 E-val: 0.0
N.aligns:None Ident:1082 Pos.:1082 Gaps:15 Align len:1153
Strand: (None, None) Frame: (1, 1)
Query: ATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTGTGTTTCGTCGAGATGCACACAAGAGTGAGGTTC
Match: ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: ATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTGTGTTTCGTCGAGATGCACACAAGAGTGAGGTTC
Align Title:gi|23241674|gb|BC035969.1| Homo sapiens albumin, mRNA (cDNA clone MGC:32581 IMAGE:4714468
Align Len: 1449
Score: 1767.0 Bits: 1594.56 E-val: 0.0
N.aligns:None Ident:888 Pos.:888 Gaps:0 Align len:891
Strand: (None, None) Frame: (1, 1)
Query: ATATAAAACCACTCTAGAGAAGTGCTGTGCCGCTGCAGATCCTCATGAATGCTATGCCAAAGTGTTCGATGAATTTAAACCTCTTGTGGAAGAC
Match: |||| ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: ATATGAAACCACTCTAGAGAAGTGCTGTGCCGCTGCAGATCCTCATGAATGCTATGCCAAAGTGTTCGATGAATTTAAACCTCTTGTGGAAGAC
```

```
Score: 1048.0 Bits: 946.252 E-val: 0.0
N.aligns:None Ident:527 Pos.:527 Gaps:0 Align len:529
Strand: (None, None) Frame: (1, 1)
Query: TCTCTTCTGTCAACCCCACGCGCCTTTGGCACAATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTC
Match: |||||||||||||||||| ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: TCTCTTCTGTCAACCCCACACGCCTTTGGCACAATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTC
Align Title:gi|15679995|gb|BC014308.1|BC014308 Homo sapiens, clone IMAGE:3934797, mRNA, partial cds
Align Len: 823
Score: 1571.0 Bits: 1417.83 E-val: 0.0
N.aligns:None Ident:790 Pos.:790 Gaps:0 Align len:793
Strand: (None, None) Frame: (1, 1)
Query: AGAATTTAATCAAACAAAATTGTGAGCTTTTTGAGCAGCTTGGAGAGTACAAATTCCAGAATGCGCTATTAGTTCGTTACACCAAGAAAGTACC
Match: ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: AGAATTTAATCAAACAAAATTGTGAGCTTTTTGAGCAGCTTGGAGAGTACAAATTCCAGAATGCGCTATTAGTTCGTTACACCAAGAAAGTACC
Align Title:gi|649152310|gb|KJ905683.1| Synthetic construct Homo sapiens clone ccsbBroadEn_15353 ALB
Align Len: 1320
Score: 1387.0 Bits: 1251.92 E-val: 0.0
N.aligns:None Ident:698 Pos.:698 Gaps:0 Align len:701
Strand: (None, None) Frame: (1, 1)
Query: ATATAAAACCACTCTAGAGAAGTGCTGTGCCGCTGCAGATCCTCATGAATGCTATGCCAAAGTGTTCGATGAATTTAAACCTCTTGTGGAAGAC
Match: |||| |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: ATATGAAACCACTCTAGAGAAGTGCTGTGCCGCTGCAGATCCTCATGAATGCTATGCCAAAGTGTTCGATGAATTTAAACCTCTTGTGGAAGAC
Score: 987.0 Bits: 891.249 E-val: 0.0
N.aligns:None Ident:495 Pos.:495 Gaps:0 Align len:496
Strand: (None, None) Frame: (1, 1)
Query: ATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTGTGTTTCGTCGAGATGCACACAAGAGTGAGGTTC
Match: ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: ATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTGTGTTTCGTCGAGATGCACACAAGAGTGAGGTTC
Align Title:gi|168740012|emb|CU690239.1| Synthetic construct Homo sapiens gateway clone IMAGE:100022
Align Len: 1356
Score: 1182.0 Bits: 1067.08 E-val: 0.0
N.aligns:None Ident:647 Pos.:647 Gaps:12 Align len:670
Strand: (None, None) Frame: (1, -1)
Query: CATGAATG-CTAT--GCCAAA-GTGTT-CGATG-AATTTAAA-CCTCTTGTGG-AAGAGCCTCAG-AATTTAAT-CAAACAAAATT-GTGAGCT
Match: |||||||| |||   |||||| ||||| ||||| |||||||| ||| |||||| |||||||||| |||||||| |||||||||| ||||||
Subjc: CATGAATGACTAATGGCCAAAAGTGTTTCGATGGAATTTAAAACCTTTTGTGGTAAGAGCCTCAGGAATTTAATTCAAACAAAATTTGTGAGCT
Align Title:gi|37181745|gb|AY358313.1| Homo sapiens DNA66677 ALB (UNQ696) mRNA, partial cds
Align Len: 524
Score: 1018.0 Bits: 919.201 E-val: 0.0
N.aligns:None Ident:517 Pos.:517 Gaps:0 Align len:523
Strand: (None, None) Frame: (1, 1)
Query: AACCCCACGCGCCTTTGGCACAATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTGTGTTTCGTCGA
Match: |||||||   || || ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: AACCCCANANGCTTTGGGCACAATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTGTGTTTCGTCGA
Align Title:gi|168740011|emb|CU690238.1| Synthetic construct Homo sapiens gateway clone IMAGE:100022
Align Len: 1442
Score: 982.0 Bits: 886.74 E-val: 0.0
N.aligns:None Ident:494 Pos.:494 Gaps:0 Align len:496
Strand: (None, None) Frame: (1, 1)
Query: ATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTGTGTTTCGTCGAGATGCACACAAGAGTGAGGTTC
Match: ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: ATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTGTGTTTCGTCGAGATGCACACAAGAGTGAGGTTC
Score: 143.0 Bits: 130.227 E-val: 1.83659e-26
N.aligns:None Ident:122 Pos.:122 Gaps:13 Align len:145
Strand: (None, None) Frame: (1, 1)
Query: ATATAAAACCACTCTAGAGAA-GTGCTGTGCC-GCTGCAGATCCTCATGAATGCTA-TGCCAAAGTGTTCGATGAATTTAAACC--TCTTGTGC
Match: |||| |||||||||||||| |||||||||| ||||||||||||||||||||||| || || ||||| |||||||| ||||||   |||| |
Subjc: ATATGAAACCACTCTAGAGAAAGTGCTGTGCCCGCTGCAGATCCTCATGAATGCTATTGACATAGTGTACGATGAATCTAAACCCTCCTTGGGC
Align Title:gi|6642751|gb|AF113014.1|AF113014 Homo sapiens clone FLB3512 mRNA sequence
```

```
Align Len: 1476
Score: 618.0 Bits: 558.527 E-val: 1.15187e-155
N.aligns:None Ident:312 Pos.:312 Gaps:0 Align len:314
Strand: (None, None) Frame: (1, 1)
Query: AAAGCTGTTATGGATGATTTCGCAGCTTTTGTAGAGAAGTGCTGCAAGGCTGACGATAAGGAGACCTGCTTTGCCGAGGAGGGTAAAAAACTTC
Match: ||||||| ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: AAAGCTGCTATGGATGATTTCGCAGCTTTTGTAGAGAAGTGCTGCAAGGCTGACGATAAGGAGACCTGCTTTGCCGAGGAGGGTAAAAAACTTC
Align Title:gi|219842221|ref|NG_009291.1| Homo sapiens albumin (ALB), RefSeqGene on chromosome 4
Align Len: 24158
Score: 581.0 Bits: 525.165 E-val: 2.37628e-145
N.aligns:None Ident:294 Pos.:294 Gaps:1 Align len:295
Strand: (None, None) Frame: (1, 1)
Query: TCAGCCTACCATGAGAATAAGAGAAAGAAAATGAAGATCAAAAGCTTATTCATCTGTTTTTCTTTTTCGTTGGTGTAAAGCCAACACCCTGTCT
Match: ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: TCAGCCTACCATGAGAATAAGAGAAAGAAAATGAAGATCAAAAGCTTATTCATCTGTTTTTCTTTTTCGTTGGTGTAAAGCCAACACCCTGTCT
Score: 449.0 Bits: 406.143 E-val: 1.60396e-109
N.aligns:None Ident:226 Pos.:226 Gaps:0 Align len:227
Strand: (None, None) Frame: (1, 1)
Query: TATCTATCCGTGGTCCTGAACCAGTTATGTGTGTTGCATGAGAAAACGCCAGTAAGTGACAGAGTCACCAAATGCTGCACAGAATCCTTGGTGA
Match: || ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: TAGCTATCCGTGGTCCTGAACCAGTTATGTGTGTTGCATGAGAAAACGCCAGTAAGTGACAGAGTCACCAAATGCTGCACAGAATCCTTGGTGA
Score: 436.0 Bits: 394.421 E-val: 2.90002e-106
N.aligns:None Ident:218 Pos.:218 Gaps:0 Align len:218
Strand: (None, None) Frame: (1, 1)
Query: GGCGGACCTTGCCAAGTATATCTGTGAAAATCAAGATTCGATCTCCAGTAAACTGAAGGAATGCTGTGAAAAACCTCTGTTGGAAAAATCCCAC
Match: ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: GGCGGACCTTGCCAAGTATATCTGTGAAAATCAAGATTCGATCTCCAGTAAACTGAAGGAATGCTGTGAAAAACCTCTGTTGGAAAAATCCCAC
Score: 425.0 Bits: 384.502 E-val: 5.24337e-103
N.aligns:None Ident:214 Pos.:214 Gaps:0 Align len:215
Strand: (None, None) Frame: (1, 1)
Query: CATACCCTTTTTGGAGACAAATTATGCACAGTTGCAACTCTTCGTGAAACCTATGGTGAAATGGCTGACTGCTGTGCAAAACAAGAACCTGAGA
Match: ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: CATACCCTTTTTGGAGACAAATTATGCACAGTTGCAACTCTTCGTGAAACCTATGGTGAAATGGCTGACTGCTGTGCAAAACAAGAACCTGAGA
Score: 278.0 Bits: 251.955 E-val: 2.23348e-63
N.aligns:None Ident:139 Pos.:139 Gaps:0 Align len:139
Strand: (None, None) Frame: (1, 1)
Query: GCTATTAGTTCGTTACACCAAGAAAGTACCCCAAGTGTCAACTCCAACTCTTGTAGAGGTCTCAAGAAACCTAGGAAAAGTGGGCAGCAAATGT
Match: ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: GCTATTAGTTCGTTACACCAAGAAAGTACCCCAAGTGTCAACTCCAACTCTTGTAGAGGTCTCAAGAAACCTAGGAAAAGTGGGCAGCAAATGT
Score: 264.0 Bits: 239.331 E-val: 1.40948e-59
N.aligns:None Ident:132 Pos.:132 Gaps:0 Align len:132
Strand: (None, None) Frame: (1, 1)
Query: GTGTTGATTGCCTTTGCTCAGTATCTTCAGCAGTGTCCATTTGAAGATCATGTAAAATTAGTGAATGAAGTAACTGAATTTGCAAAAACATGTC
Match: ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: GTGTTGATTGCCTTTGCTCAGTATCTTCAGCAGTGTCCATTTGAAGATCATGTAAAATTAGTGAATGAAGTAACTGAATTTGCAAAAACATGTC
Score: 264.0 Bits: 239.331 E-val: 1.40948e-59
N.aligns:None Ident:132 Pos.:132 Gaps:0 Align len:132
Strand: (None, None) Frame: (1, 1)
Query: GGGCAGTAGCTCGCCTGAGCCAGAGATTTCCCAAAGCTGAGTTTGCAGAAGTTTCCAAGTTAGTGACAGATCTTACCAAAGTCCACACGGAATC
Match: ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: GGGCAGTAGCTCGCCTGAGCCAGAGATTTCCCAAAGCTGAGTTTGCAGAAGTTTCCAAGTTAGTGACAGATCTTACCAAAGTCCACACGGAATC
Score: 264.0 Bits: 239.331 E-val: 1.40948e-59
N.aligns:None Ident:138 Pos.:138 Gaps:0 Align len:142
Strand: (None, None) Frame: (1, 1)
Query: AAACAAACTGCACTTGTTGAGCTTGTGAAACACAAGCCCAAGGCAACAAAAGAGCAACTGAAAGCTGTTATGGATGATTTCGCAGCTTTTGTAC
Match: ||||  | ||||||||||||||| ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: AAACTCAGTGCACTTGTTGAGCTCGTGAAACACAAGCCCAAGGCAACAAAAGAGCAACTGAAAGCTGTTATGGATGATTTCGCAGCTTTTGTAC
Score: 261.0 Bits: 236.626 E-val: 1.7171e-58
N.aligns:None Ident:132 Pos.:132 Gaps:0 Align len:133
```

```
Strand: (None, None) Frame: (1, 1)
Query: GTTTTTGTATGAATATGCAAGAAGGCATCCTGATTACTCTGTCGTGCTGCTGCTGAGACTTGCCAAGACATATAAAACCACTCTAGAGAAGTGC
Match: |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: GTTTTTGTATGAATATGCAAGAAGGCATCCTGATTACTCTGTCGTGCTGCTGCTGAGACTTGCCAAGACATATGAAACCACTCTAGAGAAGTGC
Score: 256.0 Bits: 232.118 E-val: 2.09185e-57
N.aligns:None Ident:131 Pos.:131 Gaps:0 Align len:133
Strand: (None, None) Frame: (1, 1)
Query: ATACTTATATGAAATTGCCAGAAGACATCCTTACTTCTATGCCCCGGAACTCCTTTTCTTTGCTAAAAGGTATAAAGCTGCTTTTACAGAATGT
Match: |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: ATACTTATATGAAATTGCCAGAAGACATCCTTACTTTTATGCCCCGGAACTCCTTTTCTTTGCTAAAAGGTATAAAGCTGCTTTTACAGAATGT
Score: 221.0 Bits: 200.559 E-val: 1.23639e-47
N.aligns:None Ident:115 Pos.:115 Gaps:0 Align len:118
Strand: (None, None) Frame: (1, 1)
Query: TCTCTTCTGTCAACCCCACGCGCCTTTGGCACAATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTC
Match: |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: TCTCTTCTGTCAACCCCACACGCCTTTGGCACAATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTC
Score: 200.0 Bits: 181.623 E-val: 3.3177e-42
N.aligns:None Ident:100 Pos.:100 Gaps:0 Align len:100
Strand: (None, None) Frame: (1, 1)
Query: GTTCGATGAATTTAAACCTCTTGTGGAAGAGCCTCAGAATTTAATCAAACAAAATTGTGAGCTTTTTGAGCAGCTTGGAGAGTACAAATTCCAC
Match: |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: GTTCGATGAATTTAAACCTCTTGTGGAAGAGCCTCAGAATTTAATCAAACAAAATTGTGAGCTTTTTGAGCAGCTTGGAGAGTACAAATTCCAC
Score: 197.0 Bits: 178.918 E-val: 4.04179e-41
N.aligns:None Ident:100 Pos.:100 Gaps:0 Align len:101
Strand: (None, None) Frame: (1, 1)
Query: AGCTCGATGAACTTCGGGATGAAGGGAAGGCTTCGTCTGCCAAACAGAGGCTCAAGTGTGCCAGTCTCCAAAAATTTGGAGAAAGAGCTTTCAA
Match: |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: AGCTCGATGAACTTCGGGATGAAGGGAAGGCTTCGTCTGCCAAACAGAGACTCAAGTGTGCCAGTCTCCAAAAATTTGGAGAAAGAGCTTTCAA
Score: 135.0 Bits: 123.014 E-val: 2.72574e-24
N.aligns:None Ident:69 Pos.:69 Gaps:0 Align len:70
Strand: (None, None) Frame: (1, 1)
Query: AGGGTAAAAAACTTGTTGCTGCAAGTCGAGCTGCCTTAGGCTTATAACATCACATTTAAAAGCATCTCAG  start: 1817  end: 1886
Match: |||||||||||||||||||||||||||| ||||||||||||||||||||||||||||||||||||||||||||
Subjc: AGGGTAAAAAACTTGTTGCTGCAAGTCAAGCTGCCTTAGGCTTATAACATCACATTTAAAAGCATCTCAG  start: 20998  end: 210
Score: 117.0 Bits: 106.783 E-val: 2.09555e-19
N.aligns:None Ident:60 Pos.:60 Gaps:0 Align len:61
Strand: (None, None) Frame: (1, 1)
Query: CACACAAGAGTGAGGTTGCTCATCGGTTTAAAGATTTGGGAGAAGAAAATTTCAAAGCCTT  start: 110  end: 170
Match: || ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: CAGACAAGAGTGAGGTTGCTCATCGGTTTAAAGATTTGGGAGAAGAAAATTTCAAAGCCTT  start: 5859  end: 5919
Align Title:gi|152112963|gb|EF649953.1| Homo sapiens albumin (ALB) gene, complete cds
Align Len: 21070
Score: 581.0 Bits: 525.165 E-val: 2.37628e-145
N.aligns:None Ident:294 Pos.:294 Gaps:1 Align len:295
Strand: (None, None) Frame: (1, 1)
Query: TCAGCCTACCATGAGAATAAGAGAAAGAAAATGAAGATCAAAAGCTTATTCATCTGTTTTTCTTTTTCGTTGGTGTAAAGCCAACACCCTGTCT
Match: |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: TCAGCCTACCATGAGAATAAGAGAAAGAAAATGAAGATCAAAAGCTTATTCATCTGTTTTTCTTTTTCGTTGGTGTAAAGCCAACACCCTGTCT
Score: 449.0 Bits: 406.143 E-val: 1.60396e-109
N.aligns:None Ident:226 Pos.:226 Gaps:0 Align len:227
Strand: (None, None) Frame: (1, 1)
Query: TATCTATCCGTGGTCCTGAACCAGTTATGTGTGTTGCATGAGAAAACGCCAGTAAGTGACAGAGTCACCAAATGCTGCACAGAATCCTTGGTGA
Match: || ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: TAGCTATCCGTGGTCCTGAACCAGTTATGTGTGTTGCATGAGAAAACGCCAGTAAGTGACAGAGTCACCAAATGCTGCACAGAATCCTTGGTGA
Score: 436.0 Bits: 394.421 E-val: 2.90002e-106
N.aligns:None Ident:218 Pos.:218 Gaps:0 Align len:218
Strand: (None, None) Frame: (1, 1)
Query: GGCGGACCTTGCCAAGTATATCTGTGAAAATCAAGATTCGATCTCCAGTAAACTGAAGGAATGCTGTGAAAAACCTCTGTTGGAAAAATCCCAC
Match: ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
```

```
Subjc: GGCGGACCTTGCCAAGTATATCTGTGAAAATCAAGATTCGATCTCCAGTAAACTGAAGGAATGCTGTGAAAAACCTCTGTTGGAAAAATCCCAC
Score: 425.0 Bits: 384.502 E-val: 5.24337e-103
N.aligns:None Ident:214 Pos.:214 Gaps:0 Align len:215
Strand: (None, None) Frame: (1, 1)
Query: CATACCCTTTTTGGAGACAAATTATGCACAGTTGCAACTCTTCGTGAAACCTATGGTGAAATGGCTGACTGCTGTGCAAAACAAGAACCTGAGA
Match: ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: CATACCCTTTTTGGAGACAAATTATGCACAGTTGCAACTCTTCGTGAAACCTATGGTGAAATGGCTGACTGCTGTGCAAAACAAGAACCTGAGA
Score: 278.0 Bits: 251.955 E-val: 2.23348e-63
N.aligns:None Ident:139 Pos.:139 Gaps:0 Align len:139
Strand: (None, None) Frame: (1, 1)
Query: GCTATTAGTTCGTTACACCAAGAAAGTACCCCAAGTGTCAACTCCAACTCTTGTAGAGGTCTCAAGAAACCTAGGAAAAGTGGGCAGCAAATGT
Match: ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: GCTATTAGTTCGTTACACCAAGAAAGTACCCCAAGTGTCAACTCCAACTCTTGTAGAGGTCTCAAGAAACCTAGGAAAAGTGGGCAGCAAATGT
Score: 264.0 Bits: 239.331 E-val: 1.40948e-59
N.aligns:None Ident:132 Pos.:132 Gaps:0 Align len:132
Strand: (None, None) Frame: (1, 1)
Query: GTGTTGATTGCCTTTGCTCAGTATCTTCAGCAGTGTCCATTTGAAGATCATGTAAAATTAGTGAATGAAGTAACTGAATTTGCAAAAACATGTC
Match: ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: GTGTTGATTGCCTTTGCTCAGTATCTTCAGCAGTGTCCATTTGAAGATCATGTAAAATTAGTGAATGAAGTAACTGAATTTGCAAAAACATGTC
Score: 264.0 Bits: 239.331 E-val: 1.40948e-59
N.aligns:None Ident:132 Pos.:132 Gaps:0 Align len:132
Strand: (None, None) Frame: (1, 1)
Query: GGGCAGTAGCTCGCCTGAGCCAGAGATTTCCCAAAGCTGAGTTTGCAGAAGTTTCCAAGTTAGTGACAGATCTTACCAAAGTCCACACGGAATC
Match: ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: GGGCAGTAGCTCGCCTGAGCCAGAGATTTCCCAAAGCTGAGTTTGCAGAAGTTTCCAAGTTAGTGACAGATCTTACCAAAGTCCACACGGAATC
Score: 264.0 Bits: 239.331 E-val: 1.40948e-59
N.aligns:None Ident:138 Pos.:138 Gaps:0 Align len:142
Strand: (None, None) Frame: (1, 1)
Query: AAACAAACTGCACTTGTTGAGCTTGTGAAACACAAGCCCAAGGCAACAAAAGAGCAACTGAAAGCTGTTATGGATGATTTCGCAGCTTTTGTAC
Match: ||||  | |||||||||||||||| ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: AAACTCAGTGCACTTGTTGAGCTCGTGAAACACAAGCCCAAGGCAACAAAAGAGCAACTGAAAGCTGTTATGGATGATTTCGCAGCTTTTGTAC
Score: 261.0 Bits: 236.626 E-val: 1.7171e-58
N.aligns:None Ident:132 Pos.:132 Gaps:0 Align len:133
Strand: (None, None) Frame: (1, 1)
Query: GTTTTTGTATGAATATGCAAGAAGGCATCCTGATTACTCTGTCGTGCTGCTGCTGAGACTTGCCAAGACATATAAAACCACTCTAGAGAAGTGC
Match: |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| ||||||||||||||||||
Subjc: GTTTTTGTATGAATATGCAAGAAGGCATCCTGATTACTCTGTCGTGCTGCTGCTGAGACTTGCCAAGACATATGAAACCACTCTAGAGAAGTGC
Score: 256.0 Bits: 232.118 E-val: 2.09185e-57
N.aligns:None Ident:131 Pos.:131 Gaps:0 Align len:133
Strand: (None, None) Frame: (1, 1)
Query: ATACTTATATGAAATTGCCAGAAGACATCCTTACTTCTATGCCCCGGAACTCCTTTTCTTTGCTAAAAGGTATAAAGCTGCTTTTACAGAATGT
Match: |||||||||||||||||||||||||||||||||||||||| |||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: ATACTTATATGAAATTGCCAGAAGACATCCTTACTTTTATGCCCCGGAACTCCTTTTCTTTGCTAAAAGGTATAAAGCTGCTTTTACAGAATGT
Score: 221.0 Bits: 200.559 E-val: 1.23639e-47
N.aligns:None Ident:115 Pos.:115 Gaps:0 Align len:118
Strand: (None, None) Frame: (1, 1)
Query: TCTCTTCTGTCAACCCCACGCGCCTTTGGCACAATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTC
Match: |||||||||||||||||||| ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: TCTCTTCTGTCAACCCCACACGCCTTTGGCACAATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTC
Score: 200.0 Bits: 181.623 E-val: 3.3177e-42
N.aligns:None Ident:100 Pos.:100 Gaps:0 Align len:100
Strand: (None, None) Frame: (1, 1)
Query: GTTCGATGAATTTAAACCTCTTGTGGAAGAGCCTCAGAATTTAATCAAACAAAATTGTGAGCTTTTTGAGCAGCTTGGAGAGTACAAATTCCAC
Match: ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: GTTCGATGAATTTAAACCTCTTGTGGAAGAGCCTCAGAATTTAATCAAACAAAATTGTGAGCTTTTTGAGCAGCTTGGAGAGTACAAATTCCAC
Score: 197.0 Bits: 178.918 E-val: 4.04179e-41
N.aligns:None Ident:100 Pos.:100 Gaps:0 Align len:101
Strand: (None, None) Frame: (1, 1)
Query: AGCTCGATGAACTTCGGGATGAAGGGAAGGCTTCGTCTGCCAAACAGAGGCTCAAGTGTGCCAGTCTCCAAAAATTTGGAGAAAGAGCTTTCAA
```

```
Match: |||||||||||||||||||||||||||||||||||||||||||||||||| ||||||||||||||||||||||||||||||||||||||||||||||
Subjc: AGCTCGATGAACTTCGGGATGAAGGGAAGGCTTCGTCTGCCAAACAGAGACTCAAGTGTGCCAGTCTCCAAAAATTTGGAGAAAGAGCTTTCAA
Score: 135.0 Bits: 123.014 E-val: 2.72574e-24
N.aligns:None Ident:69 Pos.:69 Gaps:0 Align len:70
Strand: (None, None) Frame: (1, 1)
Query: AGGGTAAAAAACTTGTTGCTGCAAGTCGAGCTGCCTTAGGCTTATAACATCACATTTAAAAGCATCTCAG  start: 1817  end: 1886
Match: |||||||||||||||||||||||||| ||||||||||||||||||||||||||||||||||||||||||
Subjc: AGGGTAAAAAACTTGTTGCTGCAAGTCAAGCTGCCTTAGGCTTATAACATCACATTTAAAAGCATCTCAG  start: 17948  end: 180
Score: 117.0 Bits: 106.783 E-val: 2.09555e-19
N.aligns:None Ident:60 Pos.:60 Gaps:0 Align len:61
Strand: (None, None) Frame: (1, 1)
Query: CACACAAGAGTGAGGTTGCTCATCGGTTTAAAGATTTGGGAGAAGAAAATTTCAAAGCCTT  start: 110  end: 170
Match: || ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: CAGACAAGAGTGAGGTTGCTCATCGGTTTAAAGATTTGGGAGAAGAAAATTTCAAAGCCTT  start: 2809  end: 2869
Align Title:gi|19387810|gb|AC108157.3| Homo sapiens BAC clone RP11-580P21 from 4, complete sequence
Align Len: 167001
Score: 581.0 Bits: 525.165 E-val: 2.37628e-145
N.aligns:None Ident:294 Pos.:294 Gaps:1 Align len:295
Strand: (None, None) Frame: (1, 1)
Query: TCAGCCTACCATGAGAATAAGAGAAAGAAAATGAAGATCAAAAGCTTATTCATCTGTTTTTCTTTTTCGTTGGTGTAAAGCCAACACCCTGTCT
Match: ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: TCAGCCTACCATGAGAATAAGAGAAAGAAAATGAAGATCAAAAGCTTATTCATCTGTTTTTCTTTTTCGTTGGTGTAAAGCCAACACCCTGTCT
Score: 449.0 Bits: 406.143 E-val: 1.60396e-109
N.aligns:None Ident:226 Pos.:226 Gaps:0 Align len:227
Strand: (None, None) Frame: (1, 1)
Query: TATCTATCCGTGGTCCTGAACCAGTTATGTGTGTTGCATGAGAAAACGCCAGTAAGTGACAGAGTCACCAAATGCTGCACAGAATCCTTGGTGA
Match: || ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: TAGCTATCCGTGGTCCTGAACCAGTTATGTGTGTTGCATGAGAAAACGCCAGTAAGTGACAGAGTCACCAAATGCTGCACAGAATCCTTGGTGA
Score: 436.0 Bits: 394.421 E-val: 2.90002e-106
N.aligns:None Ident:218 Pos.:218 Gaps:0 Align len:218
Strand: (None, None) Frame: (1, 1)
Query: GGCGGACCTTGCCAAGTATATCTGTGAAAATCAAGATTCGATCTCCAGTAAACTGAAGGAATGCTGTGAAAAACCTCTGTTGGAAAAATCCCAC
Match: ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: GGCGGACCTTGCCAAGTATATCTGTGAAAATCAAGATTCGATCTCCAGTAAACTGAAGGAATGCTGTGAAAAACCTCTGTTGGAAAAATCCCAC
Score: 425.0 Bits: 384.502 E-val: 5.24337e-103
N.aligns:None Ident:214 Pos.:214 Gaps:0 Align len:215
Strand: (None, None) Frame: (1, 1)
Query: CATACCCTTTTTGGAGACAAATTATGCACAGTTGCAACTCTTCGTGAAACCTATGGTGAAATGGCTGACTGCTGTGCAAAACAAGAACCTGAGA
Match: ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: CATACCCTTTTTGGAGACAAATTATGCACAGTTGCAACTCTTCGTGAAACCTATGGTGAAATGGCTGACTGCTGTGCAAAACAAGAACCTGAGA
Score: 278.0 Bits: 251.955 E-val: 2.23348e-63
N.aligns:None Ident:139 Pos.:139 Gaps:0 Align len:139
Strand: (None, None) Frame: (1, 1)
Query: GCTATTAGTTCGTTACACCAAGAAAGTACCCCAAGTGTCAACTCCAACTCTTGTAGAGGTCTCAAGAAACCTAGGAAAAGTGGGCAGCAAATGT
Match: ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: GCTATTAGTTCGTTACACCAAGAAAGTACCCCAAGTGTCAACTCCAACTCTTGTAGAGGTCTCAAGAAACCTAGGAAAAGTGGGCAGCAAATGT
Score: 264.0 Bits: 239.331 E-val: 1.40948e-59
N.aligns:None Ident:132 Pos.:132 Gaps:0 Align len:132
Strand: (None, None) Frame: (1, 1)
Query: GTGTTGATTGCCTTTGCTCAGTATCTTCAGCAGTGTCCATTTGAAGATCATGTAAAATTAGTGAATGAAGTAACTGAATTTGCAAAAACATGTC
Match: ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: GTGTTGATTGCCTTTGCTCAGTATCTTCAGCAGTGTCCATTTGAAGATCATGTAAAATTAGTGAATGAAGTAACTGAATTTGCAAAAACATGTC
Score: 264.0 Bits: 239.331 E-val: 1.40948e-59
N.aligns:None Ident:132 Pos.:132 Gaps:0 Align len:132
Strand: (None, None) Frame: (1, 1)
Query: GGGCAGTAGCTCGCCTGAGCCAGAGATTTCCCAAAGCTGAGTTTGCAGAAGTTTCCAAGTTAGTGACAGATCTTACCAAAGTCCACACGGAATC
Match: ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: GGGCAGTAGCTCGCCTGAGCCAGAGATTTCCCAAAGCTGAGTTTGCAGAAGTTTCCAAGTTAGTGACAGATCTTACCAAAGTCCACACGGAATC
Score: 264.0 Bits: 239.331 E-val: 1.40948e-59
```

```
N.aligns:None Ident:138 Pos.:138 Gaps:0 Align len:142
Strand: (None, None) Frame: (1, 1)
Query: AAACAAACTGCACTTGTTGAGCTTGTGAAACACAAGCCCAAGGCAACAAAAGAGCAACTGAAAGCTGTTATGGATGATTTCGCAGCTTTTGTAC
Match: ||||  | ||||||||||||||| ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: AAACTCAGTGCACTTGTTGAGCTCGTGAAACACAAGCCCAAGGCAACAAAAGAGCAACTGAAAGCTGTTATGGATGATTTCGCAGCTTTTGTAC
Score: 261.0 Bits: 236.626 E-val: 1.7171e-58
N.aligns:None Ident:132 Pos.:132 Gaps:0 Align len:133
Strand: (None, None) Frame: (1, 1)
Query: GTTTTTGTATGAATATGCAAGAAGGCATCCTGATTACTCTGTCGTGCTGCTGCTGAGACTTGCCAAGACATATAAAACCACTCTAGAGAAGTGC
Match: |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| ||||||||||||||||||
Subjc: GTTTTTGTATGAATATGCAAGAAGGCATCCTGATTACTCTGTCGTGCTGCTGCTGAGACTTGCCAAGACATATGAAACCACTCTAGAGAAGTGC
Score: 256.0 Bits: 232.118 E-val: 2.09185e-57
N.aligns:None Ident:131 Pos.:131 Gaps:0 Align len:133
Strand: (None, None) Frame: (1, 1)
Query: ATACTTATATGAAATTGCCAGAAGACATCCTTACTTCTATGCCCCGGAACTCCTTTTCTTTGCTAAAAGGTATAAAGCTGCTTTTACAGAATGT
Match: |||||||||||||||||||||||||||||||||||| ||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: ATACTTATATGAAATTGCCAGAAGACATCCTTACTTTTATGCCCCGGAACTCCTTTTCTTTGCTAAAAGGTATAAAGCTGCTTTTACAGAATGT
Score: 221.0 Bits: 200.559 E-val: 1.23639e-47
N.aligns:None Ident:115 Pos.:115 Gaps:0 Align len:118
Strand: (None, None) Frame: (1, 1)
Query: TCTCTTCTGTCAACCCCACGCGCCTTTGGCACAATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTC
Match: |||||||||||||||||| ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: TCTCTTCTGTCAACCCCACACGCCTTTGGCACAATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTC
Score: 200.0 Bits: 181.623 E-val: 3.3177e-42
N.aligns:None Ident:100 Pos.:100 Gaps:0 Align len:100
Strand: (None, None) Frame: (1, 1)
Query: GTTCGATGAATTTAAACCTCTTGTGGAAGAGCCTCAGAATTTAATCAAACAAAATTGTGAGCTTTTTGAGCAGCTTGGAGAGTACAAATTCCAC
Match: ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: GTTCGATGAATTTAAACCTCTTGTGGAAGAGCCTCAGAATTTAATCAAACAAAATTGTGAGCTTTTTGAGCAGCTTGGAGAGTACAAATTCCAC
Score: 197.0 Bits: 178.918 E-val: 4.04179e-41
N.aligns:None Ident:100 Pos.:100 Gaps:0 Align len:101
Strand: (None, None) Frame: (1, 1)
Query: AGCTCGATGAACTTCGGGATGAAGGGAAGGCTTCGTCTGCCAAACAGAGGCTCAAGTGTGCCAGTCTCCAAAAATTTGGAGAAAGAGCTTTCAA
Match: |||||||||||||||||||||||||||||||||||||||||||||||||||| ||||||||||||||||||||||||||||||||||||||||||||
Subjc: AGCTCGATGAACTTCGGGATGAAGGGAAGGCTTCGTCTGCCAAACAGAGACTCAAGTGTGCCAGTCTCCAAAAATTTGGAGAAAGAGCTTTCAA
Score: 135.0 Bits: 123.014 E-val: 2.72574e-24
N.aligns:None Ident:69 Pos.:69 Gaps:0 Align len:70
Strand: (None, None) Frame: (1, 1)
Query: AGGGTAAAAAACTTGTTGCTGCAAGTCGAGCTGCCTTAGGCTTATAACATCACATTTAAAAGCATCTCAG  start: 1817  end: 1886
Match: |||||||||||||||||||||||||| ||||||||||||||||||||||||||||||||||||||||||||
Subjc: AGGGTAAAAAACTTGTTGCTGCAAGTCAAGCTGCCTTAGGCTTATAACATCACATTTAAAAGCATCTCAG  start: 86100  end: 861
Score: 117.0 Bits: 106.783 E-val: 2.09555e-19
N.aligns:None Ident:60 Pos.:60 Gaps:0 Align len:61
Strand: (None, None) Frame: (1, 1)
Query: CACACAAGAGTGAGGTTGCTCATCGGTTTAAAGATTTGGGAGAAGAAAATTTCAAAGCCTT  start: 110  end: 170
Match: || ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: CAGACAAGAGTGAGGTTGCTCATCGGTTTAAAGATTTGGGAGAAGAAAATTTCAAAGCCTT  start: 70961  end: 71021
Score: 60.0 Bits: 55.3874 E-val: 0.000332357
N.aligns:None Ident:45 Pos.:45 Gaps:0 Align len:55
Strand: (None, None) Frame: (1, 1)
Query: ATATCTGTGAAAATCAAGATTCGATCTCCAGTAAACTGAAGGAATGCTGTGAAAA  start: 893  end: 947
Match: |||| |||  ||| |||||||| |||||||| ||| | || || ||||||||||
Subjc: ATATTTGTTCAAAACAAGATTCTATCTCCAGCAAAATCAAAGAGTGCTGTGAAAA  start: 157736  end: 157790
Align Title:gi|178343|gb|M12523.1|HUMALBGC Human serum albumin (ALB) gene, complete cds
Align Len: 19002
Score: 581.0 Bits: 525.165 E-val: 2.37628e-145
N.aligns:None Ident:294 Pos.:294 Gaps:1 Align len:295
Strand: (None, None) Frame: (1, 1)
Query: TCAGCCTACCATGAGAATAAGAGAAAGAAAATGAAGATCAAAAGCTTATTCATCTGTTTTTCTTTTTCGTTGGTGTAAAGCCAACACCCTGTCT
```

```
Match: ||||||
```

```
Query: TCTCTTCTGTCAACCCCACGCGCCTTTGGCACAATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTC
Match: |||||||||||||||||| ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: TCTCTTCTGTCAACCCCACACGCCTTTGGCACAATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTC
Score: 200.0 Bits: 181.623 E-val: 3.3177e-42
N.aligns:None Ident:100 Pos.:100 Gaps:0 Align len:100
Strand: (None, None) Frame: (1, 1)
Query: GTTCGATGAATTTAAACCTCTTGTGGAAGAGCCTCAGAATTTAATCAAACAAAATTGTGAGCTTTTTGAGCAGCTTGGAGAGTACAAATTCCAC
Match: ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: GTTCGATGAATTTAAACCTCTTGTGGAAGAGCCTCAGAATTTAATCAAACAAAATTGTGAGCTTTTTGAGCAGCTTGGAGAGTACAAATTCCAC
Score: 197.0 Bits: 178.918 E-val: 4.04179e-41
N.aligns:None Ident:100 Pos.:100 Gaps:0 Align len:101
Strand: (None, None) Frame: (1, 1)
Query: AGCTCGATGAACTTCGGGATGAAGGGAAGGCTTCGTCTGCCAAACAGAGGCTCAAGTGTGCCAGTCTCCAAAAATTTGGAGAAAGAGCTTTCAA
Match: |||||||||||||||||||||||||||||||||||||||||||||||||| |||||||||||||||||||||||||||||||||||||||||||
Subjc: AGCTCGATGAACTTCGGGATGAAGGGAAGGCTTCGTCTGCCAAACAGAGACTCAAGTGTGCCAGTCTCCAAAAATTTGGAGAAAGAGCTTTCAA
Score: 135.0 Bits: 123.014 E-val: 2.72574e-24
N.aligns:None Ident:69 Pos.:69 Gaps:0 Align len:70
Strand: (None, None) Frame: (1, 1)
Query: AGGGTAAAAAACTTGTTGCTGCAAGTCGAGCTGCCTTAGGCTTATAACATCACATTTAAAAGCATCTCAG  start: 1817  end: 1886
Match: |||||||||||||||||||||||||| ||||||||||||||||||||||||||||||||||||||||||
Subjc: AGGGTAAAAAACTTGTTGCTGCAAGTCAAGCTGCCTTAGGCTTATAACATCACATTTAAAAGCATCTCAG  start: 17686  end: 177
Score: 117.0 Bits: 106.783 E-val: 2.09555e-19
N.aligns:None Ident:60 Pos.:60 Gaps:0 Align len:61
Strand: (None, None) Frame: (1, 1)
Query: CACACAAGAGTGAGGTTGCTCATCGGTTTAAAGATTTGGGAGAAGAAAATTTCAAAGCCTT  start: 110  end: 170
Match: || ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: CAGACAAGAGTGAGGTTGCTCATCGGTTTAAAGATTTGGGAGAAGAAAATTTCAAAGCCTT  start: 2561  end: 2621
Align Title:gi|1036032687|gb|AH007061.2| Homo sapiens chromosome 4 serum albumin (ALB) gene, partial
Align Len: 523
Score: 579.0 Bits: 523.362 E-val: 8.29405e-145
N.aligns:None Ident:293 Pos.:293 Gaps:1 Align len:294
Strand: (None, None) Frame: (1, 1)
Query: TCAGCCTACCATGAGAATAAGAGAAAGAAAATGAAGATCAAAAGCTTATTCATCTGTTTTTCTTTTTCGTTGGTGTAAAGCCAACACCCTGTCT
Match: |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| ||||||||||||||||||||||||||||||
Subjc: TCAGCCTACCATGAGAATAAGAGAAAGAAAATGAAGATCAAAAGCTTATTCATCTGTTTT-CTTTTTCGTTGGTGTAAAGCCAACACCCTGTCT
Align Title:gi|408407822|gb|AF116616.2| Homo sapiens PRO0998 mRNA, complete cds
Align Len: 1649
Score: 577.0 Bits: 521.558 E-val: 2.89491e-144
N.aligns:None Ident:290 Pos.:290 Gaps:0 Align len:291
Strand: (None, None) Frame: (1, -1)
Query: AGCTTTTGTAGAGAAGTGCTGCAAGGCTGACGATAAGGAGACCTGCTTTGCCGAGGAGGGTAAAAAACTTGTTGCTGCAAGTCGAGCTGCCTTA
Match: |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| |||||||||
Subjc: AGCTTTTGTAGAGAAGTGCTGCAAGGCTGACGATAAGGAGACCTGCTTTGCCGAGGAGGGTAAAAAACTTGTTGCTGCAAGTCAAGCTGCCTTA
Align Title:gi|338858118|gb|AH002596.2|SEG_HUMALB Homo sapiens albumin (ALB) gene, partial cds
Align Len: 1863
Score: 573.0 Bits: 517.951 E-val: 3.52672e-143
N.aligns:None Ident:290 Pos.:290 Gaps:1 Align len:291
Strand: (None, None) Frame: (1, 1)
Query: CCTACCATGAGAATAAGAGAAAGAAAATGAAGATCAAAAGCTTATTCATCTGTTTTTCTTTTTCGTTGGTGTAAAGCCAACACCCTGTCTAAAA
Match: ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: CCTACCATGAGAATAAGAGAAAGAAAATGAAGATCAAAAGCTTATTCATCTGTTTTTCTTTTTCGTTGGTGTAAAGCCAACACCCTGTCTAAAA
Score: 221.0 Bits: 200.559 E-val: 1.23639e-47
N.aligns:None Ident:115 Pos.:115 Gaps:0 Align len:118
Strand: (None, None) Frame: (1, 1)
Query: TCTCTTCTGTCAACCCCACGCGCCTTTGGCACAATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTC
Match: |||||||||||||||||| ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: TCTCTTCTGTCAACCCCACACGCCTTTGGCACAATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTC
Align Title:gi|1122817323|gb|DQ145726.2| Homo sapiens ARNT-interacting protein 2 (AINP2) mRNA, comple
Align Len: 1054
```

```
Score: 404.0 Bits: 365.567 E-val: 1.40699e-97
N.aligns:None Ident:205 Pos.:205 Gaps:0 Align len:207
Strand: (None, None) Frame: (1, 1)
Query: AGCTGCCTTAGGCTTATAACATCACATTTAAAAGCATCTCAGCCTACCATGAGAATAAGAGAAAGAAAATGAAGATCAAAAGCTTATTCATCTC
Match: |||||||| ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: AGCTGCCTTTGGCTTATAACATCACATTTAAAAGCATCTCAGCCTACCATGAGAATAAGAGAAAGAAAATGAAGATCAAAAGCTTATTCATCTC
Align Title:gi|32428|emb|X51365.1| Homo sapiens mRNA for albumin (clone pHA19)
Align Len: 117
Score: 232.0 Bits: 210.477 E-val: 6.8383e-51
N.aligns:None Ident:116 Pos.:116 Gaps:0 Align len:116
Strand: (None, None) Frame: (1, 1)
Query: AGCTTATTCATCTGTTTTTCTTTTTCGTTGGTGTAAAGCCAACACCCTGTCTAAAAAACATAAATTTCTTTAATCATTTTGCCTCTTTTCTCTC
Match: ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: AGCTTATTCATCTGTTTTTCTTTTTCGTTGGTGTAAAGCCAACACCCTGTCTAAAAAACATAAATTTCTTTAATCATTTTGCCTCTTTTCTCTC
Align Title:gi|1338686512|ref|NG_056261.1| Homo sapiens albumin (ALB) 5' regulatory region (LOC111832
Align Len: 6858
Score: 221.0 Bits: 200.559 E-val: 1.23639e-47
N.aligns:None Ident:115 Pos.:115 Gaps:0 Align len:118
Strand: (None, None) Frame: (1, 1)
Query: TCTCTTCTGTCAACCCCACGCGCCTTTGGCACAATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTC
Match: |||||||||||||||||| ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: TCTCTTCTGTCAACCCCACACGCCTTTGGCACAATGAAGTGGGTAACCTTTATTTCCCTTCTTTTTCTCTTTAGCTCGGCTTATTCCAGGGGTC
Align Title:gi|7959920|gb|AF116711.1|AF116711 Homo sapiens PRO2646 mRNA, complete cds
Align Len: 1160
Score: 216.0 Bits: 196.05 E-val: 1.50624e-46
N.aligns:None Ident:108 Pos.:108 Gaps:0 Align len:108
Strand: (None, None) Frame: (1, -1)
Query: GGAAGAGCCTCAGAATTTAATCAAACAAAATTGTGAGCTTTTTGAGCAGCTTGGAGAGTACAAATTCCAGAATGCGCTATTAGTTCGTTACACC
Match: ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subjc: GGAAGAGCCTCAGAATTTAATCAAACAAAATTGTGAGCTTTTTGAGCAGCTTGGAGAGTACAAATTCCAGAATGCGCTATTAGTTCGTTACACC
```

Example: Let's align the first 100 bases of the first 5 entries of the file contigs82.fasta to the Malus Domestica genome, writing the results to a apple_first5.xml file. Sample output is here: apple_first5.xml. **NOTE: this can take several minutes.**

```
In [40]: from Bio.Blast import NCBIWWW
         from Bio import SeqIO

         rc = SeqIO.parse("file_samples/contigs82.fasta", format="fasta")
         fasta_string = ""
         ids2align =[]
         for i in range(5):
             entry = next(rc)
             fasta_string +=">" + entry.id +"\n"+entry.seq +"\n"
             ids2align.append(entry.id)
         print("Aligning the following {} entries:\n\t{}".format(
                                                        len(ids2align),
                                                        "\n\t".join(ids2align)
                                                        ))
         result_handle = NCBIWWW.qblast("blastn", "nt",
                                        fasta_string,
                                        entrez_query='"Malus Domestica" [Organism]'
                                       )

         with open("file_samples/blast_res_apple.xml","w") as out_f:
             out_f.write(result_handle.read())
         print("Output written to \"blast_res_apple.xml\"")
```

```
Aligning the following 5 entries:
        MDC020656.85
```

```
        MDC001115.177
        MDC013284.379
        MDC018185.243
        MDC018185.241
Output written to "blast_res_apple.xml"
```

12.4 Getting data from NCBI

Biopython provides a module (`Bio.Entrez`) to pull data off resources like PubMed or GenBank, and other repositories programmatically through Entrez[97].

There are some limitations (mostly taken care directly by Biopython) that you should be aware of when you use NCBI's services (e.g. they recommend not to run more than 3 requests per second on weekdays, ...). Check here[98] if you want to learn what these limitations are.

First of all we need to import the Entrez module with (`from Bio import Entrez`) and then we can start interacting with Entrez, then we should specify (optional) an email setting `Entrez.email` (this is optional but you get a warning if you do not specify it – the email would be used to notify the user in case of excessive usage).

In particular the module (complete info on Entrez module are here[99]) provides, among the others, the following functions:

1. `res_handle = Entrez.einfo(db)` returns a summary of the Entez databases as a results handle. `db` is an optional parameter specifying the resource of interest;
2. `res_handle = Entrez.esearch(db, term,id)` returns all the entries in `db` having query matching the term `term`. It is also possible to specify an `id` to get the information relative to that resource id;
3. `res_handle = Entrez.efetch(db, id, rettype, retmode)` returns full record corresponding to the identifier `id` from the database `db` formatted in `rettype` (eg. gb, fasta,... complete list[100]) and return mode `retmode` (eg. text);
4. `res_handle = Entrez.esummary(db, id)` returns the summary of the entry `id` from the database `db` as a handle;
5. `result = Entrez.read(res_handle)` reads the information on the XML handle `res_handle` and stores them in a dictionary, list or string, depending on the case.

Example:

Let's get a list of all available databases in Entrez as a dictionary. Let's then get a summary of the entries in 'sra'.

```
In [41]: from Bio import Entrez
         import datetime


         Entrez.email = "my_email"
         handle = Entrez.einfo()
         res = Entrez.read(handle)
         #print(res)
         print("")
         print("As a list:")
         print(res['DbList'])

         res = Entrez.read(Entrez.einfo(db = "sra"))
```

[97] http://www.ncbi.nlm.nih.gov/Entrez/
[98] http://www.ncbi.nlm.nih.gov/books/NBK25497/#chapter2.Usage_Guidelines_and_Requiremen
[99] http://biopython.org/DIST/docs/api/Bio.Entrez-module.html
[100] https://www.ncbi.nlm.nih.gov/books/NBK25499/table/chapter4.T._valid_values_of__retmode_and/?report=objectonly

```
        #uncomment to see all the information captured
        #print(res)
        #for el in res["DbInfo"].keys():
        #    print(el)
        date = res["DbInfo"]["LastUpdate"]
        dt = datetime.datetime.strptime(date, "%Y/%m/%d %H:%M")
        print("")
        print("Entries count: {:,}".format(int(res["DbInfo"]["Count"])))
        print("LastUpdate: {}/{}/{} {}:{}".format(dt.day,
                                                  dt.month,
                                                  dt.year,
                                                  dt.hour,
                                                  dt.minute))
        print("Description:", res["DbInfo"]["Description"])
```

```
As a list:
['pubmed', 'protein', 'nuccore', 'ipg', 'nucleotide', 'structure', 'sparcle', 'genome', 'annotinfo',

Entries count: 9,274,756
LastUpdate: 28/10/2019 11:54
Description: SRA Database
```

Note that effectively **einfo** returned a handler to an object that can be read by the **read** function that produces a dictionary. This dictionary had one key only "DbList" that is the list of available databases in the first case, while the key when db was specified is "DbInfo".

Example: Fetch the first three entries (retmax = 3) in pubmed that are related to the species "Malus Domestica" and report the title of the publication.

```
In [42]: from Bio import Entrez

         Entrez.email = "my_email"
         handle = Entrez.esearch(db="pubmed", term="Malus Domestica", retmax = 3)
         res = Entrez.read(handle)
         for el in res.keys():
             print(el , " : ", res[el])

         print("")
         for ids in res["IdList"]:
             print("Results for id:", ids)
             handle = Entrez.esummary(db="pubmed",  id = ids)
             res = Entrez.read(handle)
             #uncomment to see all info
             #print(res)
             for r in res:
                 print(r["Title"])
                 print(r["Source"])
                 print("")
```

```
Count  :  5900
RetMax  :  3
RetStart  :  0
IdList  :  ['31655474', '31655083', '31649709']
TranslationSet  :  [DictElement({'From': 'Malus Domestica', 'To': '"malus"[MeSH Terms] OR "malus"[All
TranslationStack  :  [DictElement({'Term': '"malus"[MeSH Terms]', 'Field': 'MeSH Terms', 'Count': '5
QueryTranslation  :  "malus"[MeSH Terms] OR "malus"[All Fields] OR ("malus"[All Fields] AND "domestic
```

```
Results for id: 31655474
The role of glucose-6-phosphate dehydrogenase in reactive oxygen species metabolism in apple exocarp
Food Chem

Results for id: 31655083
Construction of full-length infectious cDNA clones of Apple stem grooving virus using Gibson Assembly
Virus Res

Results for id: 31649709
Red to Brown: An Elevated Anthocyanic Response in Apple Drives Ethylene to Advance Maturity and Fruit
Front Plant Sci
```

Example: Retrieve genbank formatted information of the Malus x domestica MYB domain class transcription factor (MYB1) mRNA complete cds (nucleotide database id:HM122614.1). Parse it as a SeqRecord, printing only the sequence (remember previous practical's SeqIO).

```
In [43]: from Bio import Entrez
         from Bio import SeqIO

         Entrez.email = "my_email"
         handle = Entrez.efetch(db="nucleotide",
                                id = "HM122614.1",
                                rettype = "gb",
                                retmode="text")
         my_seq = SeqIO.read(handle, format = "genbank")
         print(handle.read())
         print(my_seq)
         print("")
         print("SEQUENCE:")
         print(my_seq.seq)
```

```
ID: HM122614.1
Name: HM122614
Description: Malus x domestica MYB domain class transcription factor (MYB1) mRNA, complete cds
Number of features: 3
/molecule_type=mRNA
/topology=linear
/data_file_division=HTC
/date=15-AUG-2010
/accessions=['HM122614']
/sequence_version=1
/keywords=['HTC']
/source=Malus domestica (apple)
/organism=Malus domestica
/taxonomy=['Eukaryota', 'Viridiplantae', 'Streptophyta', 'Embryophyta', 'Tracheophyta', 'Spermatophyt
/references=[Reference(title='Transcription Factors in Apple', ...), Reference(title='The FruiTFul da
Seq('TTTGGTCTGCTGGGTAGGTACTCATAAAAACAAACCAACCGAAGCCTCCGAACC...AAA', IUPACAmbiguousDNA())

SEQUENCE:
TTTGGTCTGCTGGGTAGGTACTCATAAAAACAAACCAACCGAAGCCTCCGAACCGACCACCAATGACGGCCCCAAACGGCGCCGTCCCCAAACAAGCCGAC
```

12.5 Getting data from ExPASy

Similarly to what done with Entrez, it is possible to pull data out of ExPASy (https://www.expasy.org/) through the Bio.ExPASy module. We will not cover this in detail. All information can be found here: Bio.ExPASy module[101].

As an example, we will see how to download a couple of SwissProt entries (the human, rhesus macaque and mouse P53 protein) and perform pairwise sequence alignment of Human vs Macaque and Human vs Mouse.

```
In [44]: from Bio import ExPASy
         from Bio import SwissProt
         from Bio import pairwise2

         #the ids of the human, rhesus and mouse proteins
         accessions = ["P04637", "P56424", "P02340"]
         sequences = []
         for accession in accessions:
             handle = ExPASy.get_sprot_raw(accession)
             record = SwissProt.read(handle)
             print("Organism: {}".format(record.organism))
             print(record.entry_name)
             print(",".join(record.accessions))
             print(record.keywords)

             print(record.sequence[:30] + "...")
             sequences.append(record.sequence)

         print("\n\nPairwise sequence alignment of Human vs Rhesus P53:")

         aligns = pairwise2.align.globalxx(sequences[0],sequences[1])
         for al in aligns[0:3]:
             print("Score {}:".format(al[2]))
             print(al[0][0:50])
             print(al[1][0:50])

         print("\n\nPairwise sequence alignment of Human vs Mouse P53:")

         aligns = pairwise2.align.globalxx(sequences[0],sequences[2])
         for al in aligns[0:3]:
             print("Score {}:".format(al[2]))
             print(al[0][0:50])
             print(al[1][0:50])
```

```
Organism: Homo sapiens (Human).
P53_HUMAN
P04637,Q15086,Q15087,Q15088,Q16535,Q16807,Q16808,Q16809,Q16810,Q16811,Q16848,Q2XN98,Q3LRW1,Q3LRW2,Q3I
['3D-structure', 'Acetylation', 'Activator', 'Alternative promoter usage', 'Alternative splicing', 'A
MEEPQSDPSVEPPLSQETFSDLWKLLPENN...
Organism: Macaca mulatta (Rhesus macaque).
P53_MACMU
P56424
['Acetylation', 'Activator', 'Apoptosis', 'Biological rhythms', 'Cell cycle', 'Complete proteome', 'C
MEEPQSDPSIEPPLSQETFSDLWKLLPENN...
Organism: Mus musculus (Mouse).
P53_MOUSE
P02340,Q9QUP3
['3D-structure', 'Acetylation', 'Activator', 'Apoptosis', 'Biological rhythms', 'Cell cycle', 'Comple
MTAMEESQSDISLELPLSQETFSGLWKLLP...
```

[101] http://biopython.org/DIST/docs/api/Bio.ExPASy-module.html#__package__

```
Pairwise sequence alignment of Human vs Rhesus P53:
Score 376.0:
MEEPQSDPSV-EPPLSQETFSDLWKLLPENNVLSPLPSQAM-DDLMLSPD
MEEPQSDPS-IEPPLSQETFSDLWKLLPENNVLSPLPSQA-VDDLMLSPD
Score 376.0:
MEEPQSDPSVEPPLSQETFSDLWKLLPENNVLSPLPSQAM-DDLMLSPDD
MEEPQSDPSIEPPLSQETFSDLWKLLPENNVLSPLPSQA-VDDLMLSPDD
Score 376.0:
MEEPQSDPSV-EPPLSQETFSDLWKLLPENNVLSPLPSQAMDDLMLSPDD
MEEPQSDPS-IEPPLSQETFSDLWKLLPENNVLSPLPSQAVDDLMLSPDD


Pairwise sequence alignment of Human vs Mouse P53:
Score 315.0:
M---EEP-QSDP-SV-EP-PLSQETFSD-LWKLLPENNVLSP---LPSQA
MTAMEE-SQSD-IS-LE-LPLSQETFS-GLWKLLP------PEDILPS--
Score 315.0:
---MEEP-QSDP-SV-EP-PLSQETFSD-LWKLLPENNVLSP---LPSQA
MTAMEE-SQSD-IS-LE-LPLSQETFS-GLWKLLP------PEDILPS--
Score 315.0:
M---EEPQSDP-SV-EP-PLSQETFSD-LWKLLPENNVLSP---LPSQA-
MTAMEESQSD-IS-LE-LPLSQETFS-GLWKLLP------PEDILPS--P
```

You can find here[102] all the details on how to deal with the SwissProt records.

12.6 3D structure and PDB

Biopython can also deal with data coming from the Protein Data Bank database[103]. It is a database of structural information of 3D shapes of proteins, nucleic acids, and complex assemblies. The database currently contains more than 157,000 total structures.

To deal with this kind of data we first need to import Biopython's module `Bio.PDB` with `from Bio.PDB import *`. All the information on this module can be found here[104].

It is possible to download a structure directly from PDB by using a `PDBList` object that features a function called `download_pdb_files` having the basic syntax:

```
PDBList.download_pdb_files(pdb_codes, pdir, file_format)
```

that downloads the `file_format` formatted structures defined in the `pdb_codes` list of 4 symbols structure Ids from PDB, stores them in the directory `pdir`. The safer `file_format` to use is "mmCif". The function will not download the structures more than once. If a file is already present in the specified directory, a message **Structure exists** will be displayed.

Example:

Let's programmatically download two different structures of the DNA polymerase 3C2K[105] and 3C2L[106]

```
In [45]: from Bio.PDB import *

         pdbl = PDBList()
```

[102] http://biopython.org/DIST/docs/api/toc-Bio.SeqRecord-module.html
[103] https://www.rcsb.org/pdb/home/home.do
[104] http://biopython.org/DIST/docs/api/Bio.PDB-module.html#__package__
[105] https://www.rcsb.org/pdb/explore/explore.do?structureId=3C2K
[106] https://www.rcsb.org/pdb/explore/explore.do?structureId=3C2L

```
structures = ["3C2K", "3C2L"]
el = pdbl.download_pdb_files(structures,
                             file_format = "mmCif",
                             pdir = "file_samples/")
```

```
Structure exists: 'file_samples/3c2k.cif'
Structure exists: 'file_samples/3c2l.cif'
```

Macromolecular Crystallographic Information Files (mmCif files .cif) is a paired collection of names (starting with "_") and values.They also contain a description of the 3D placement of every crystalized atom of the structure. A detailed description of the format can be found here[107].

Once the structures are available locally, one can start parsing them to do something useful. Parsing can be done through the `MMCIFParser` object:

```
parser = MMCIFParser()
```

The `parser` object has several methods able to deal with structures. One of these is the `get_structure` that creates a `PDB.Structure.Structure` object with all the data present in the structure file.

The basic syntax is:

```
structure = parser.get_structure(pdb_code, filename)
```

where `pdb_code` is the PDB code of the structure contained in the file `filename`. The method returns a `PDB.Structure.Structure` that contains one or more models.

A `Structure` consists of a collection of one or more `Model` (different 3D conformations of the very same structure) that is a collection of `Chain` that is a collection of `Residues` that is a collection of `Atoms`. Look in the documentation to get the information on each of these classes. This is the diagram of a structure:

Given a `Structure` we can obtain iterators to **models**, **chains**, **residues** or **atoms** with:

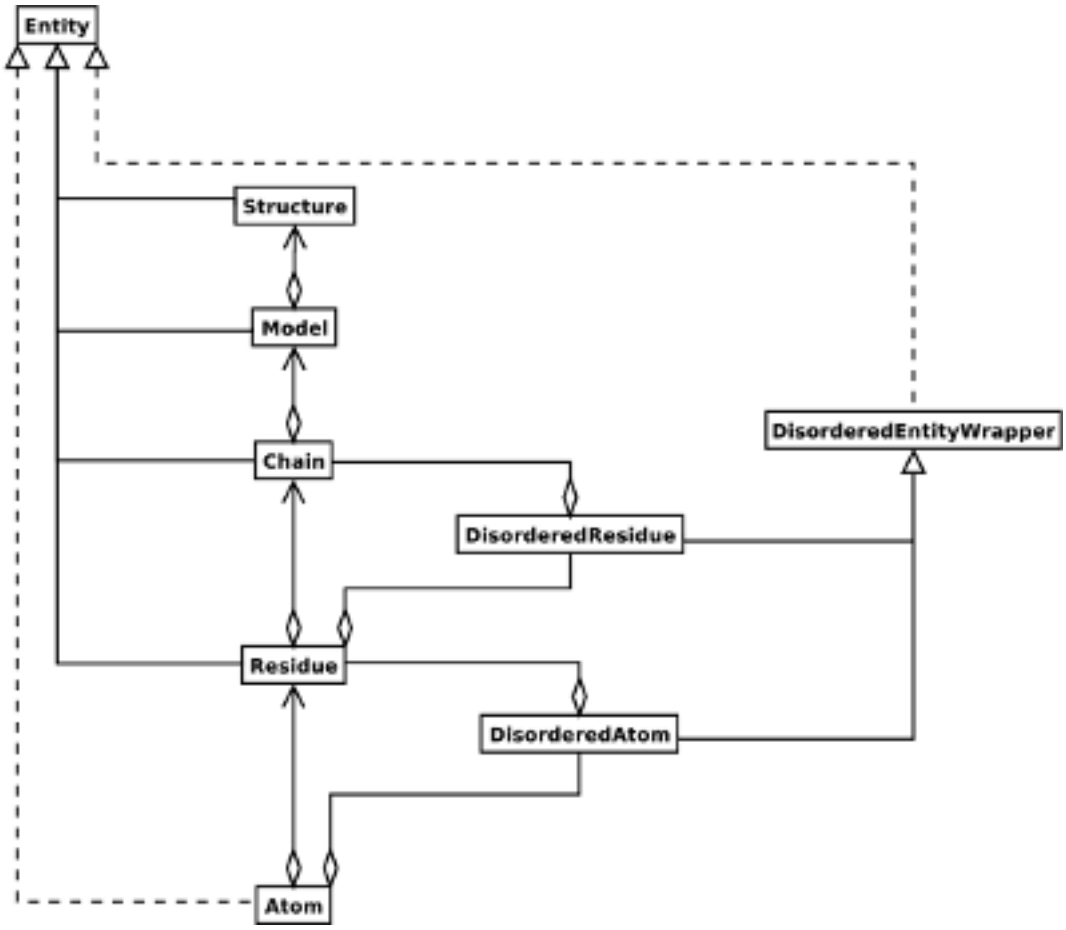
```
Structure.get_models()
Structure.get_chains()
Structure.get_residues()
Structure.get_atoms()
```

For each model obtained with `structure.get_models()` function we can loop through its chains, residues and atoms. For atoms we can get the 3D coordinates with `Atom.get_coord()`.

Example: Let's loop through all the models, chain, residues and atoms of the DNA polymerase structure 3C2K. Print the 3D coordinates of each atom.

```
In [46]: from Bio.PDB import *


         parser = MMCIFParser(QUIET=True) #To disable warnings


         filename = "file_samples/3c2l.cif"
         structure = parser.get_structure("3c2l", filename)

         for model in structure.get_models():
             print("model", model, "has {} chains".format(len(model)))

             for chain in model:
                 print(" - chain ", chain, "has {} residues".format(len(chain)))
```

[107] http://mmcif.wwpdb.org/docs/tutorials/mechanics/pdbx-mmcif-syntax.html


Entity
Structure
Model
Chain
DisorderedEntityWrapper
DisorderedResidue
Residue
DisorderedAtom
Atom

```
                for residue in chain:
                    print ("      - residue", residue.get_resname(), "has {} atoms".format(len(resi

                    for atom in residue:
                        x,y,z = atom.get_coord()
                        print("        - atom:", atom.get_name(), "x: {} y:{} z:{}".format(x,y,z))
```

```
model <Model id=0> has 4 chains
 - chain  <Chain id=T> has 41 residues
      - residue DC has 16 atoms
        - atom: O5' x: 30.740999221801758 y:-2.2209999561309814 z:16.618999481201172
        - atom: C5' x: 31.167999267578125 y:-0.9599999785423279 z:16.062999725341797
        - atom: C4' x: 29.996000289916992 y:-0.009999999776482582 z:15.932999610900879
        - atom: O4' x: 28.96299934387207 y:-0.6069999933242798 z:15.107000350952148
        - atom: C3' x: 29.320999145507812 y:0.38499999046325684 z:17.253000259399414
        - atom: O3' x: 29.166000366210938 y:1.8140000104904175 z:17.327999114990234
        - atom: C2' x: 27.96500015258789 y:-0.3019999861717224 z:17.187999725341797
        - atom: C1' x: 27.701000213623047 y:-0.3490000069141388 z:15.692000389099121
        - atom: N1 x: 26.76099967956543 y:-1.4010000228881836 z:15.24899959564209
        - atom: C2 x: 25.65999984741211 y:-1.034000039100647 z:14.456999778747559
        - atom: O2 x: 25.51300048828125 y:0.16200000047683716 z:14.140999794006348
        - atom: N3 x: 24.78499984741211 y:-1.9910000562667847 z:14.057000160217285
        - atom: C4 x: 24.97599983215332 y:-3.2660000324249268 z:14.414999961853027
        - atom: N4 x: 24.086000442504883 y:-4.172999858856201 z:14.0
        - atom: C5 x: 26.08799934387207 y:-3.6679999828338623 z:15.21399974822998
        - atom: C6 x: 26.94700050354004 y:-2.7139999866485596 z:15.604000091552734
      - residue DC has 19 atoms
        - atom: P x: 28.641000747680664 y:2.5 z:18.69099998474121
        - atom: OP1 x: 29.559999465942383 y:3.625 z:19.025999069213867
        - atom: OP2 x: 28.385000228881836 y:1.4290000200271606 z:19.698999404907227
        - atom: O5' x: 27.225000381469727 y:3.1240000724792248 z:18.30299949645996
        - atom: C5' x: 27.128999710083008 y:4.366000175476074 z:17.600000381469727
        - atom: C4' x: 25.67799949645996 y:4.741000175476074 z:17.399999618530273
        - atom: O4' x: 24.954999923706055 y:3.5789999961853027 z:16.92799949645996
        - atom: C3' x: 24.93600082397461 y:5.171000003814697 z:18.663000106811523
        - atom: O3' x: 23.81800079345703 y:5.980000019073486 z:18.29800033569336
        - atom: C2' x: 24.413000106811523 y:3.8489999771118164 z:19.187999725341797
        - atom: C1' x: 23.961999893188477 y:3.2190001010894775 z:17.882999420166016
        - atom: N1 x: 23.832000732421875 y:1.7519999742507935 z:17.87299919128418
        - atom: C2 x: 22.68899917602539 y:1.1799999475479126 z:17.298999786376953
        - atom: O2 x: 21.812999725341797 y:1.9190000295639038 z:16.829999923706055
        - atom: N3 x: 22.569000244140625 y:-0.16200000047683716 z:17.268999099731445
        - atom: C4 x: 23.52899932861328 y:-0.9279999732971191 z:17.78700065612793
        - atom: N4 x: 23.367000579833984 y:-2.252000093460083 z:17.738000869750977
        - atom: C5 x: 24.697999954223633 y:-0.37400001287460327 z:18.3799991607666
        - atom: C6 x: 24.808000564575195 y:0.9559999704360962 z:18.40399932861328
      - residue DG has 22 atoms
        - atom: P x: 23.493000030517578 y:7.309000015258789 z:19.131000518798828
        - atom: OP1 x: 24.016000747680664 y:8.472999572753906 z:18.36199951171875
        - atom: OP2 x: 23.934999465942383 y:7.0960001945495605 z:20.53100013732910
        - atom: O5' x: 21.90399932861328 y:7.382999897003174 z:19.136999130249023
        - atom: C5' x: 21.174999237060547 y:7.506999969482422 z:17.922000885009766
        - atom: C4' x: 19.93899917602539 y:6.644999980926514 z:17.98200035095215
        - atom: O4' x: 20.363000869750977 y:5.269000053405762 z:18.142000198364258
        - atom: C3' x: 19.03100013732910 y:6.951000213623047 z:19.179000854492188
        - atom: O3' x: 17.66200065612793 y:6.968999862670898 z:18.771999359130860
        - atom: C2' x: 19.263999938964844 y:5.7769999504089355 z:20.114000320434570
        - atom: C1' x: 19.552000045776367 y:4.665999889373779 z:19.127000808715820
```

```
    - atom: N9 x: 20.26799964904785 y:3.5169999599456787 z:19.67300033569336
    - atom: C8 x: 21.375 y:3.5269999504089355 z:20.490999221801758
    - atom: N7 x: 21.760000228881836 y:2.3299999237060547 z:20.850000381469727
    - atom: C5 x: 20.85700035095215 y:1.4789999723434448 z:20.225000381469727
    - atom: C6 x: 20.763999938964844 y:0.06800000369548798 z:20.246999740600586
    - atom: O6 x: 21.47800064086914 y:-0.7459999918937683 z:20.850000381469727
    - atom: N1 x: 19.707000732421875 y:-0.382999986410141 z:19.465999603271484
    - atom: C2 x: 18.85099983215332 y:0.4169999957084656 z:18.756000518798828
    - atom: N2 x: 17.89900016784668 y:-0.20800000429153442 z:18.055999755859375
    - atom: N3 x: 18.92099952697754 y:1.7330000400543213 z:18.733999252319336
    - atom: C4 x: 19.940000534057617 y:2.194999933242798 z:19.485000610351562
  - residue DA has 21 atoms
    - atom: P x: 16.55500030517578 y:7.6579999923706055 z:19.7189998626709
    - atom: OP1 x: 16.44099998474121 y:9.088000297546387 z:19.329999923706055
    - atom: OP2 x: 16.868000030517578 y:7.307000160217285 z:21.135000228881836
    - atom: O5' x: 15.189000129699707 y:6.940000057220459 z:19.316999435424805
    - atom: C5' x: 14.980999946594238 y:6.421999931335449 z:18.000999450683594
    - atom: C4' x: 13.972000122070312 y:5.296999931335449 z:18.041000366210938
    - atom: O4' x: 14.60099983215332 y:4.105000019073486 z:18.56999969482422
    - atom: C3' x: 12.777000427246094 y:5.566999912261963 z:18.947999954223633
    - atom: O3' x: 11.604000091552734 y:4.940000057220459 z:18.440000534057617
    - atom: C2' x: 13.196000099182129 y:4.954999923706055 z:20.273000717163086
    - atom: C1' x: 14.07699966430664 y:3.7829999923706055 z:19.85700035095215
    - atom: N9 x: 15.21399974822998 y:3.496999979019165 z:20.743000030517578
    - atom: C8 x: 15.923999786376953 y:4.367000102996826 z:21.542999267578125
    - atom: N7 x: 16.92300033569336 y:3.8059999942779954 z:22.18000030517578
    - atom: C5 x: 16.867000579833984 y:2.4760000705718994 z:21.781999588012695
    - atom: C6 x: 17.660999298095703 y:1.3539999723434448 z:22.099000930786133
    - atom: N6 x: 18.72599983215332 y:1.3949999809265137 z:22.900999069213867
    - atom: N1 x: 17.31800079345703 y:0.17200000858412552 z:21.548999786376953
    - atom: C2 x: 16.260000228881836 y:0.12600000202655792 z:20.73200035095215
    - atom: N3 x: 15.449000358581543 y:1.1030000448226929 z:20.347999572753906
    - atom: C4 x: 15.809000015258789 y:2.2679998874664307 z:20.913000106811523
  - residue DC has 19 atoms
    - atom: P x: 10.178999900817871 y:5.289999961853027 z:19.089000701904297
    - atom: OP1 x: 9.128999710083008 y:5.105999946594238 z:18.054000854492188
    - atom: OP2 x: 10.288999557495117 y:6.593999862670898 z:19.79800033569336
    - atom: O5' x: 9.972999572753906 y:4.1579999923706055 z:20.18600082397461
    - atom: C5' x: 9.642000198364258 y:2.8340001106262207 z:19.795000076293945
    - atom: C4' x: 9.98900032043457 y:1.8619999885559082 z:20.893999099731445
    - atom: O4' x: 11.369999885559082 y:1.9759999513626099 z:21.277000427246094
    - atom: C3' x: 9.21500015258789 y:2.0450000762939453 z:22.19499969482422
    - atom: O3' x: 8.050999641418457 y:1.2230000495910645 z:22.155000686645508
    - atom: C2' x: 10.194000244140625 y:1.5709999799728394 z:23.26799964904785
    - atom: C1' x: 11.472000122070312 y:1.246999979019165 z:22.483999252319336
    - atom: N1 x: 12.76099967956543 y:1.5789999961853027 z:23.125999450683594
    - atom: C2 x: 13.718000411987305 y:0.5619999766349792 z:23.277999877929688
    - atom: O2 x: 13.458999633789062 y:-0.578000009059906 z:22.86400032043457
    - atom: N3 x: 14.899999618530273 y:0.8500000238418579 z:23.868999481201172
    - atom: C4 x: 15.147000312805176 y:2.0880000591278076 z:24.297000885009766
    - atom: N4 x: 16.325000762939453 y:2.3239998817443848 z:24.8700008392334
    - atom: C5 x: 14.196999549865723 y:3.1410000324249268 z:24.152000427246094
    - atom: C6 x: 13.029999732971191 y:2.8459999561309814 z:23.56599998474121
  - residue DC has 19 atoms
    - atom: P x: 6.784999847412109 y:1.5880000591278076 z:23.06800079345703
    - atom: OP1 x: 7.255000114440918 y:1.5880000591278076 z:24.479000091552734
    - atom: OP2 x: 5.660999774932861 y:0.6949999928474426 z:22.66699981689453
    - atom: O5' x: 6.434000015258789 y:3.0880000591278076 z:22.659000396728516
    - atom: C5' x: 5.370999813079834 y:3.378000020980835 z:21.74799919128418
```

```
    - atom: C4' x: 4.375999927520752 y:4.328000068664551 z:22.378000259399414
    - atom: O4' x: 3.687000036239624 y:3.677000045776367 z:23.465999603271484
    - atom: C3' x: 4.9770002365112305 y:5.60699987411499 z:22.972999572753906
    - atom: O3' x: 4.706999778747559 y:6.71999979019165 z:22.107999801635742
    - atom: C2' x: 4.288000106811523 y:5.76200008392334 z:24.326000213623047
    - atom: C1' x: 3.2070000171661377 y:4.689000129699707 z:24.322999954223633
    - atom: N1 x: 2.882999897003174 y:4.071000099182129 z:25.628000259399414
    - atom: C2 x: 3.9179999828338623 y:3.555999994277954 z:26.45199966430664
    - atom: O2 x: 5.10099983215332 y:3.6510000228881836 z:26.08799934387207
    - atom: N3 x: 3.5899999141693115 y:2.9709999561309814 z:27.625
    - atom: C4 x: 2.311000108718872 y:2.88700008392334 z:27.996999740600586
    - atom: N4 x: 2.0390000343322754 y:2.2960000038146973 z:29.16200065612793
    - atom: C5 x: 1.2519999742507935 y:3.4059998989105225 z:27.194000244140625
    - atom: C6 x: 1.5789999961853027 y:3.9860000610351562 z:26.031999588012695
  - residue DG has 22 atoms
    - atom: P x: 5.339000225067139 y:8.164999961853027 z:22.430999755859375
    - atom: OP1 x: 5.185999870300293 y:9.013999938964844 z:21.215999603271484
    - atom: OP2 x: 6.695000171661377 y:7.960000038146973 z:23.01799964904785
    - atom: O5' x: 4.388999938964844 y:8.75 z:23.570999145507812
    - atom: C5' x: 2.9639999866485596 y:8.717000007629395 z:23.433000564575195
    - atom: C4' x: 2.2890000343322754 y:8.987000465393066 z:24.761999130249023
    - atom: O4' x: 2.5959999561309814 y:7.940000057220459 z:25.718000411987305
    - atom: C3' x: 2.671999931335449 y:10.305999755859375 z:25.44499969482422
    - atom: O3' x: 1.5190000534057617 y:10.883999824523926 z:26.075000762939453
    - atom: C2' x: 3.6530001163482666 y:9.859000205993652 z:26.516000747680664
    - atom: C1' x: 3.0190000534057617 y:8.543000221252441 z:26.93000030517578
    - atom: N9 x: 3.871000051498413 y:7.5920000076293945 z:27.639999389648438
    - atom: C8 x: 5.236999988555908 y:7.434000015258789 z:27.535999298095703
    - atom: N7 x: 5.703999996185303 y:6.49399995803833 z:28.315000534057617
    - atom: C5 x: 4.581999778747559 y:5.999000072479248 z:28.972000122070312
    - atom: C6 x: 4.456999778747559 y:4.964000225067139 z:29.95199966430664
    - atom: O6 x: 5.3429999351501465 y:4.252999782562256 z:30.444000244140625
    - atom: N1 x: 3.134000062942505 y:4.795000076293945 z:30.350000381469727
    - atom: C2 x: 2.063999891281128 y:5.514999866485596 z:29.8700008392334
    - atom: N2 x: 0.8629999756813049 y:5.197000026702881 z:30.371000289916992
    - atom: N3 x: 2.1649999618530273 y:6.474999904632568 z:28.964000701904297
    - atom: C4 x: 3.443000078201294 y:6.664000034332275 z:28.562999725341797
  - residue DC has 19 atoms
    - atom: P x: 0.33799999952316284 y:11.519000053405762 z:25.18000030517578
    - atom: OP1 x: 0.1550000011920929 y:10.725000381469727 z:23.934999465942383
    - atom: OP2 x: 0.6060000061988831 y:12.97599983215332 z:25.08799934387207
    - atom: O5' x: -0.9620000123977661 y:11.310999870300293 z:26.07699966430664
    - atom: C5' x: -1.5479999780654907 y:10.01099967956543 z:26.243999481201172
    - atom: C4' x: -2.0759999752044678 y:9.840999603271484 z:27.652999877929688
    - atom: O4' x: -1.0989999771118164 y:9.121000289916992 z:28.450000762939453
    - atom: C3' x: -2.3559999465942383 y:11.156999588012695 z:28.393999099731445
    - atom: O3' x: -3.6589999198913574 y:11.140999794006348 z:28.993000030517578
    - atom: C2' x: -1.2630000114440918 y:11.21500015258789 z:29.448999404907227
    - atom: C1' x: -1.0080000162124634 y:9.743000030517578 z:29.716999053955078
    - atom: N1 x: 0.3089999854564667 y:9.418000221252441 z:30.30299949645996
    - atom: C2 x: 0.3610000014305115 y:8.574000358581543 z:31.42300033569336
    - atom: O2 x: -0.6980000138282776 y:8.114999771118164 z:31.885000228881836
    - atom: N3 x: 1.5650000057220459 y:8.277999877929688 z:31.972000122070312
    - atom: C4 x: 2.681999921798706 y:8.782999992370605 z:31.444000244140625
    - atom: N4 x: 3.8420000076293945 y:8.456999778747559 z:32.00699996948242
    - atom: C5 x: 2.6589999198913574 y:9.640000343322754 z:30.309999465942383
    - atom: C6 x: 1.4630000591278076 y:9.928999900817871 z:29.774999618530273
  - residue DG has 22 atoms
    - atom: P x: -4.367000102996826 y:12.529000282287598 z:29.423999786376953
```

```
  - atom: OP1 x: -5.24399995803833 y:12.972000122070312 z:28.31399917602539
  - atom: OP2 x: -3.3519999980926514 y:13.472999572753906 z:29.975000381469727
  - atom: O5' x: -5.328999996185303 y:12.105999946594238 z:30.618999481201172
  - atom: C5' x: -5.7230000495910645 y:10.743000030517578 z:30.81599998474121
  - atom: C4' x: -5.9730000495910645 y:10.482000350952148 z:32.2859992980957
  - atom: O4' x: -4.754000186920166 y:10.02400016784668 z:32.917999267578125
  - atom: C3' x: -6.394000053405762 y:11.713000297546387 z:33.08300018310547
  - atom: O3' x: -7.164999961853027 y:11.305000305175781 z:34.207000732421875
  - atom: C2' x: -5.059999942779541 y:12.281000137329102 z:33.53200149536133
  - atom: C1' x: -4.284999847412109 y:11.005999565124512 z:33.8380012512207
  - atom: N9 x: -2.8359999656677246 y:11.102999687194824 z:33.691001892089844
  - atom: C8 x: -2.1500000953674316 y:11.842000007629395 z:32.762001037597656
  - atom: N7 x: -0.8539999723434448 y:11.725000381469727 z:32.86600112915039
  - atom: C5 x: -0.6690000295639038 y:10.854999542236328 z:33.930999755859375
  - atom: C6 x: 0.5249999761581421 y:10.35200023651123 z:34.5099983215332
  - atom: O6 x: 1.6929999589920044 y:10.571999549865723 z:34.17399978637695
  - atom: N1 x: 0.25699999928474426 y:9.508999824523926 z:35.584999084472656
  - atom: C2 x: -0.9919999837875366 y:9.182999610900879 z:36.04100036621094
  - atom: N2 x: -1.0390000343322754 y:8.371000289916992 z:37.106998443603516
  - atom: N3 x: -2.114000082015991 y:9.626999855041504 z:35.5
  - atom: C4 x: -1.8799999952316284 y:10.461000442504883 z:34.457000732421875
- residue DC has 19 atoms
  - atom: P x: -7.857999801635742 y:12.404000282287598 z:35.13100051879883
  - atom: OP1 x: -9.21500015258789 y:11.907999992370605 z:35.45399856567383
  - atom: OP2 x: -7.696000099182129 y:13.723999977111816 z:34.48099899291992
  - atom: O5' x: -6.958000183105469 y:12.425999641418457 z:36.45100021362305
  - atom: C5' x: -7.033999919891357 y:11.376999855041504 z:37.417999267578125
  - atom: C4' x: -5.870999813079834 y:11.456999778747559 z:38.382999420166016
  - atom: O4' x: -4.645999908447266 y:11.569000244140625 z:37.64799880981445
  - atom: C3' x: -5.867000102996826 y:12.637999534606934 z:39.358001708984375
  - atom: O3' x: -6.331999778747559 y:12.196000099182129 z:40.64400100708008
  - atom: C2' x: -4.410999774932861 y:13.088000297546387 z:39.395999908447266
  - atom: C1' x: -3.680999994277954 y:12.074999809265137 z:38.53200149536133
  - atom: N1 x: -2.5789999961853027 y:12.61400032043457 z:37.72200012207031
  - atom: C2 x: -1.2640000581741333 y:12.213000297546387 z:38.012001037597656
  - atom: O2 x: -1.0640000104904175 y:11.440999984741211 z:38.959999084472656
  - atom: N3 x: -0.2469999992032623 y:12.678999900817871 z:37.25299835205078
  - atom: C4 x: -0.4959999918937683 y:13.519000053405762 z:36.24800109863281
  - atom: N4 x: 0.5400000214576721 y:13.949999809265137 z:35.527000427246094
  - atom: C5 x: -1.8209999799728394 y:13.954999923706055 z:35.93899917602539
  - atom: C6 x: -2.822999954223633 y:13.482999801635742 z:36.69499969482422
- residue DA has 21 atoms
  - atom: P x: -6.264999866485596 y:13.177000045776367 z:41.92599868774414
  - atom: OP1 x: -7.065999984741211 y:12.534000396728516 z:43.00299835205078
  - atom: OP2 x: -6.591000080108643 y:14.555000305175781 z:41.50400161743164
  - atom: O5' x: -4.739999771118164 y:13.109999656677246 z:42.375999450683594
  - atom: C5' x: -4.144000053405762 y:11.85200023651123 z:42.70800018310547
  - atom: C4' x: -2.9830000400543213 y:12.027999877929688 z:43.66400146484375
  - atom: O4' x: -1.7519999742507935 y:12.300000190734863 z:42.95500183105469
  - atom: C3' x: -3.1029999256134033 y:13.125 z:44.70899963378906
  - atom: O3' x: -2.431999921798706 y:12.666999816894531 z:45.887001037597656
  - atom: C2' x: -2.390000104904175 y:14.291999816894531 z:44.04800033569336
  - atom: C1' x: -1.2949999570846558 y:13.611000061035156 z:43.23400115966797
  - atom: N9 x: -0.9739999771118164 y:14.23900032043457 z:41.952999114990234
  - atom: C8 x: -1.8300000429153442 y:14.854999542236328 z:41.07400131225586
  - atom: N7 x: -1.25 y:15.282999992370605 z:39.97800064086914
  - atom: C5 x: 0.07800000160932541 y:14.930999755859375 z:40.14799880981445
  - atom: C6 x: 1.2029999494552612 y:15.083999633789062 z:39.32500076293945
  - atom: N6 x: 1.1610000133514404 y:15.645000457763672 z:38.11600112915039
```

```
  - atom: N1 x: 2.38700008392334 y:14.62600040435791 z:39.78900146484375
  - atom: C2 x: 2.4230000972747803 y:14.048999786376953 z:40.99599838256836
  - atom: N3 x: 1.430999994277954 y:13.84000015258789 z:41.85900115966797
  - atom: C4 x: 0.26899999380111694 y:14.305000305175781 z:41.36800003051758
- residue DT has 20 atoms
  - atom: P x: -2.188999891281128 y:13.661999702453613 z:47.132999420166016
  - atom: OP1 x: -2.315999984741211 y:12.843000411987305 z:48.37300109863281
  - atom: OP2 x: -2.986999988555908 y:14.913000106811523 z:47.0
  - atom: O5' x: -0.6570000052452087 y:14.04800033569336 z:46.965999603271484
  - atom: C5' x: 0.3330000042915344 y:13.050999641418457 z:46.792999267578125
  - atom: C4' x: 1.6670000553131104 y:13.706999778747559 z:46.54499816894531
  - atom: O4' x: 1.6430000066757202 y:14.303999900817871 z:45.2239990234375
  - atom: C3' x: 1.9980000257492065 y:14.854999542236328 z:47.505001068115234
  - atom: O3' x: 3.4089999198913574 y:14.859000205993652 z:47.75600051879883
  - atom: C2' x: 1.628999948501587 y:16.091999053955078 z:46.70199966430664
  - atom: C1' x: 2.1050000190734863 y:15.644000053405762 z:45.33000183105469
  - atom: N1 x: 1.6460000276565552 y:16.400999069213867 z:44.13199996948242
  - atom: C2 x: 2.572999954223633 y:16.636999130249023 z:43.12900161743164
  - atom: O2 x: 3.7309999465942383 y:16.2810001373291 z:43.194000244140625
  - atom: N3 x: 2.0889999866485596 y:17.312999725341797 z:42.03900146484375
  - atom: C4 x: 0.8059999942779541 y:17.77199935913086 z:41.84700012207031
  - atom: O4 x: 0.5199999809265137 y:18.349000930786133 z:40.797000885009766
  - atom: C5 x: -0.11999999731779099 y:17.506999969482422 z:42.948001861572266
  - atom: C7 x: -1.534999966621399 y:17.979000091552734 z:42.840999603271484
  - atom: C6 x: 0.3440000116825104 y:16.844999313354492 z:44.018001556396484
- residue DC has 19 atoms
  - atom: P x: 3.9679999351501465 y:15.352999687194824 z:49.18199920654297
  - atom: OP1 x: 4.440999984741211 y:14.140000343322754 z:49.9010009765625
  - atom: OP2 x: 2.9700000286102295 y:16.26300048828125 z:49.823001861572266
  - atom: O5' x: 5.254000186920166 y:16.208999633789062 z:48.83000183105469
  - atom: C5' x: 6.129000186920166 y:15.824000358581543 z:47.78900146484375
  - atom: C4' x: 6.697999954223633 y:17.05900001525879 z:47.137001037597656
  - atom: O4' x: 5.788000106811523 y:17.520000457763672 z:46.10200119018555
  - atom: C3' x: 6.85699987411499 y:18.238000869750977 z:48.10900115966797
  - atom: O3' x: 8.050000190734863 y:18.95199966430664 z:47.797000885009766
  - atom: C2' x: 5.665999889373779 y:19.121000289916992 z:47.775001525878906
  - atom: C1' x: 5.61899995803833 y:18.917999267578125 z:46.27199935913086
  - atom: N1 x: 4.400000095367432 y:19.339000701904297 z:45.55400085449219
  - atom: C2 x: 4.5269999504089355 y:19.80699920654297 z:44.236000061035156
  - atom: O2 x: 5.656000137329102 y:19.850000381469727 z:43.71200180053711
  - atom: N3 x: 3.4189999103546143 y:20.195999145507812 z:43.56399917602539
  - atom: C4 x: 2.2239999771118164 y:20.1200008392334 z:44.14899826049805
  - atom: N4 x: 1.159999966621399 y:20.499000549316406 z:43.443000793457031
  - atom: C5 x: 2.065999984741211 y:19.648000717163086 z:45.48500061035156
  - atom: C6 x: 3.1689999103546143 y:19.274999618530273 z:46.145999908447266
- residue DA has 21 atoms
  - atom: P x: 9.348999977111816 y:18.784000396728516 z:48.71699905395508
  - atom: OP1 x: 9.670999526977539 y:17.33099937438965 z:48.79199981689453
  - atom: OP2 x: 9.105999946594238 y:19.55500030517578 z:49.965999603271484
  - atom: O5' x: 10.47599983215332 y:19.483999252319336 z:47.83599853515625
  - atom: C5' x: 10.871000289916992 y:18.892000198364258 z:46.59400177001953
  - atom: C4' x: 11.45199966430664 y:19.920999526977539 z:45.64799880981445
  - atom: O4' x: 10.427000045776367 y:20.615999221801758 z:44.90599822998047
  - atom: C3' x: 12.335000038146973 y:21.007999420166016 z:46.249000549316406
  - atom: O3' x: 13.369999885559082 y:21.326000213623047 z:45.31800079345703
  - atom: C2' x: 11.381999969482422 y:22.177000045776367 z:46.426998138427734
  - atom: C1' x: 10.312000274658203 y:21.95800018310547 z:45.35599899291992
  - atom: N9 x: 8.944000244140625 y:22.131000518798828 z:45.83399963378906
  - atom: C8 x: 8.444000244140625 y:21.815000534057617 z:47.073001861572266
```

```
  - atom: N7 x: 7.1620001792907715 y:22.04400062561035 z:47.196998596191406
  - atom: C5 x: 6.795000076293945 y:22.551000595092773 z:45.95899963378906
  - atom: C6 x: 5.565000057220459 y:22.979999542236328 z:45.452999114990234
  - atom: N6 x: 4.434999942779541 y:22.95800018310547 z:46.16299819946289
  - atom: N1 x: 5.5329999923706055 y:23.437000274658203 z:44.18199920654297
  - atom: C2 x: 6.673999786376953 y:23.458999633789062 z:43.481998443603516
  - atom: N3 x: 7.896999835968018 y:23.07699966430664 z:43.85300064086914
  - atom: C4 x: 7.885000228881836 y:22.624000549316406 z:45.11600112915039
- residue DG has 22 atoms
  - atom: P x: 14.439000129699707 y:22.465999603271484 z:45.67100143432617
  - atom: OP1 x: 15.732000350952148 y:22.033000946044922 z:45.07500076293945
  - atom: OP2 x: 14.36400032043457 y:22.801000595092773 z:47.12300109863281
  - atom: O5' x: 13.901000022888184 y:23.725000381469727 z:44.86199951171875
  - atom: C5' x: 13.479999542236328 y:23.606000900268555 z:43.500999450683594
  - atom: C4' x: 12.72700023651123 y:24.849000930786133 z:43.0880012512207
  - atom: O4' x: 11.375 y:24.797000885009766 z:43.606998443603516
  - atom: C3' x: 13.336999893188477 y:26.138999938964844 z:43.637001037597656
  - atom: O3' x: 13.461999893188477 y:27.09600067138672 z:42.59199905395508
  - atom: C2' x: 12.329000473022461 y:26.611000061035156 z:44.67499923706055
  - atom: C1' x: 11.03499984741211 y:26.066999435424805 z:44.11199951171875
  - atom: N9 x: 9.951000213623047 y:25.892000198364258 z:45.073001861572266
  - atom: C8 x: 10.029000282287598 y:25.426000595092773 z:46.36800003051758
  - atom: N7 x: 8.871000289916992 y:25.405000686645508 z:46.97800064086914
  - atom: C5 x: 7.974999904632568 y:25.875999450683594 z:46.02299880981445
  - atom: C6 x: 6.565999984741211 y:26.079999923706055 z:46.090999603271484
  - atom: O6 x: 5.789000034332275 y:25.85099983215332 z:47.0369987487793
  - atom: N1 x: 6.070000171661377 y:26.593000411987305 z:44.902000427246094
  - atom: C2 x: 6.811999797821045 y:26.860000610351562 z:43.779998779296875
  - atom: N2 x: 6.139999866485596 y:27.356000900268555 z:42.720001220703125
  - atom: N3 x: 8.112000465393066 y:26.660999298095703 z:43.69499969482422
  - atom: C4 x: 8.626999855041504 y:26.17799949645996 z:44.845001220703125
- residue DC has 19 atoms
  - atom: P x: 14.321999549865723 y:28.42799949645996 z:42.8380012512207
  - atom: OP1 x: 15.064000129699707 y:28.68000030517578 z:41.57600021362305
  - atom: OP2 x: 15.067000389099121 y:28.295000076293945 z:44.112998962402344
  - atom: O5' x: 13.229000091552734 y:29.56800079345703 z:43.03499984741211
  - atom: C5' x: 12.413000106811523 y:29.974000930786133 z:41.94300079345703
  - atom: C4' x: 11.097000122070312 y:30.51300048828125 z:42.446998596191406
  - atom: O4' x: 10.517000198364258 y:29.645000457763672 z:43.43600082397461
  - atom: C3' x: 11.199999809265137 y:31.851999282836914 z:43.159000396728516
  - atom: O3' x: 11.206000328063965 y:32.849998474121094 z:42.15399932861328
  - atom: C2' x: 9.986000061035156 y:31.849000930786133 z:44.07699966430664
  - atom: C1' x: 9.475000381469727 y:30.405000686645508 z:44.016998291015625
  - atom: N1 x: 9.112000465393066 y:29.808000564575195 z:45.314998626708984
  - atom: C2 x: 7.763000011444092 y:29.753999710083008 z:45.65299987792969
  - atom: O2 x: 6.926000118255615 y:30.17300033569336 z:44.83599853515625
  - atom: N3 x: 7.3979997634887695 y:29.2450008392334 z:46.8489990234375
  - atom: C4 x: 8.324999809265137 y:28.78700065612793 z:47.6870002746582
  - atom: N4 x: 7.909999847412109 y:28.29199981689453 z:48.85599899291992
  - atom: C5 x: 9.71399974822998 y:28.812999725341797 z:47.36399841308594
  - atom: C6 x: 10.059000015258789 y:29.32699966430664 z:46.17900085449219
- residue MN has 1 atoms
  - atom: MN x: 23.54800033569336 y:1.0 z:22.975000381469727
- residue HOH has 1 atoms
  - atom: O x: -0.03799999877810478 y:22.62299919128418 z:45.85499954223633
- residue HOH has 1 atoms
  - atom: O x: -0.2680000066757202 y:6.394999980926514 z:26.27400016784668
- residue HOH has 1 atoms
  - atom: O x: -2.617000102996826 y:17.14299964904785 z:38.38199996948242
```

```
    - residue HOH has 1 atoms
      - atom: O x: -2.8450000286102295 y:9.204000473022461 z:40.37200164794922
    - residue HOH has 1 atoms
      - atom: O x: -4.395999908447266 y:15.696999549865723 z:41.887001037597656
    - residue HOH has 1 atoms
      - atom: O x: 8.746000289916992 y:3.684000015258789 z:24.974000930786133
    - residue HOH has 1 atoms
      - atom: O x: 20.542999267578125 y:3.009999990463257 z:24.375
    - residue HOH has 1 atoms
      - atom: O x: -6.396999835968018 y:10.779000282287598 z:28.01799964904785
    - residue HOH has 1 atoms
      - atom: O x: -9.277000427246094 y:15.675000190734863 z:35.47200012207031
    - residue HOH has 1 atoms
      - atom: O x: 6.900000095367432 y:24.628000259399414 z:49.99599838256836
    - residue HOH has 1 atoms
      - atom: O x: 30.777999877929688 y:4.184000015258789 z:16.06800079345703
    - residue HOH has 1 atoms
      - atom: O x: 14.788999557495117 y:20.408000946044922 z:42.99700164794922
    - residue HOH has 1 atoms
      - atom: O x: -2.247999906539917 y:17.725000381469727 z:46.025001525878906
    - residue HOH has 1 atoms
      - atom: O x: 25.37700080871582 y:2.634999990463257 z:22.086999893188477
    - residue HOH has 1 atoms
      - atom: O x: 24.393999099731445 y:-1.1100000143051147 z:22.084999084472656
    - residue HOH has 1 atoms
      - atom: O x: 5.159999847412109 y:11.8100004196167 z:48.76900100708008
    - residue HOH has 1 atoms
      - atom: O x: -1.0690000057220046 y:4.329999923706055 z:24.06399917602539
    - residue HOH has 1 atoms
      - atom: O x: -0.26600000262260437 y:7.039999961853027 z:23.948999404907227
    - residue HOH has 1 atoms
      - atom: O x: 7.377999782562256 y:9.512999534606934 z:17.650999069213867
    - residue HOH has 1 atoms
      - atom: O x: 9.468000411987305 y:9.222999572753906 z:19.29199981689453
    - residue HOH has 1 atoms
      - atom: O x: 10.92199993133545 y:19.01799964904785 z:41.494998931884766
    - residue HOH has 1 atoms
      - atom: O x: -5.491000175476074 y:14.925999641418457 z:35.11899948120117
    - residue HOH has 1 atoms
      - atom: O x: -8.88599967956543 y:16.402999877929688 z:38.94599914550781
    - residue HOH has 1 atoms
      - atom: O x: -9.723999977111816 y:17.750999450683594 z:33.974998474121094
- chain  <Chain id=P> has 18 residues
    - residue DG has 19 atoms
      - atom: O5' x: -2.617000102996826 y:29.32900047302246 z:49.946998596191406
      - atom: C5' x: -3.3010001182556152 y:30.40399932861328 z:49.26599884033203
      - atom: C4' x: -2.434000015258789 y:30.941999435424805 z:48.154998779296875
      - atom: O4' x: -1.0850000381469727 y:30.972999572753906 z:48.6510009765625
      - atom: C3' x: -2.38700008392334 y:30.049999237060547 z:46.91699981689453
      - atom: O3' x: -3.3589999675750732 y:30.52400016784668 z:45.96099853515625
      - atom: C2' x: -0.9549999833106995 y:30.20599937438965 z:46.415000915527344
      - atom: C1' x: -0.17599999904632568 y:30.812999725341797 z:47.584999084472656
      - atom: N9 x: 0.9449999928474426 y:30.017000198364258 z:48.07699966430664
      - atom: C8 x: 0.9980000257492065 y:29.2549991607666 z:49.22100067138672
      - atom: N7 x: 2.1589999198913574 y:28.683000564575195 z:49.40299987792969
      - atom: C5 x: 2.9100000858306885 y:29.089000701904297 z:48.3129997253418
      - atom: C6 x: 4.244999885559082 y:28.812999725341797 z:47.972999572753906
      - atom: O6 x: 5.076000213623047 y:28.15399932861328 z:48.59700012207031
      - atom: N1 x: 4.5980000495910645 y:29.413999557495117 z:46.77299880981445
```

```
  - atom: C2 x: 3.7769999504089355 y:30.197999954223633 z:46.01499938964844
  - atom: N2 x: 4.291999816894531 y:30.69099998474121 z:44.88600158691406
  - atom: N3 x: 2.5409998893737793 y:30.482999801635742 z:46.33399963378906
  - atom: C4 x: 2.1710000038146973 y:29.89900016784668 z:47.481998443603516
- residue DC has 19 atoms
  - atom: P x: -3.7049999237060547 y:29.652999877929688 z:44.64699935913086
  - atom: OP1 x: -4.730999946594238 y:30.423999786376953 z:43.893001556396484
  - atom: OP2 x: -3.9800000190734863 y:28.243000030517578 z:45.04199981689453
  - atom: O5' x: -2.365999937057495 y:29.698999404907227 z:43.7859992980957
  - atom: C5' x: -1.8380000591278076 y:30.945999145507812 z:43.33399963378906
  - atom: C4' x: -0.75 y:30.716999053955078 z:42.310001373291016
  - atom: O4' x: 0.4480000138282776 y:30.246000289916992 z:42.97100067138672
  - atom: C3' x: -1.0950000286102295 y:29.672000885009766 z:41.250999450683594
  - atom: O3' x: -0.6209999918937683 y:30.075000762939453 z:39.96699905395508
  - atom: C2' x: -0.3880000114440918 y:28.417999267578125 z:41.731998443603516
  - atom: C1' x: 0.8130000233650208 y:28.95599937438965 z:42.499000549316406
  - atom: N1 x: 1.1870000362396240 y:28.141000747680664 z:43.66999816894531
  - atom: C2 x: 2.5039999485015870 y:27.666000366210938 z:43.784000396728516
  - atom: O2 x: 3.3299999237060547 y:27.927000045776367 z:42.89099884033203
  - atom: N3 x: 2.8429999351501465 y:26.937000274658203 z:44.86800003051758
  - atom: C4 x: 1.9320000410079956 y:26.677000045776367 z:45.81399917602539
  - atom: N4 x: 2.3199999332427980 y:25.972000122070312 z:46.88399887084961
  - atom: C5 x: 0.5879999995231628 y:27.132999420166016 z:45.71099853515625
  - atom: C6 x: 0.2630000114440918 y:27.854000091552734 z:44.63600158691406
- residue DT has 20 atoms
  - atom: P x: -0.8280000090599060 y:29.101999282836914 z:38.70800018310547
  - atom: OP1 x: -0.8669999837875366 y:29.930999755859375 z:37.46799850463867
  - atom: OP2 x: -1.9559999704360962 y:28.173999786376953 z:39.0099983215332
  - atom: O5' x: 0.5260000228881836 y:28.270999908447266 z:38.683998107910156
  - atom: C5' x: 1.7779999971389770 y:28.93600082397461 z:38.53799819946289
  - atom: C4' x: 2.8570001125335693 y:27.93600082397461 z:38.20399856567383
  - atom: O4' x: 3.2509999275207520 y:27.259000778198242 z:39.422000885009766
  - atom: C3' x: 2.3949999809265137 y:26.836000442504883 z:37.24700164794922
  - atom: O3' x: 3.4969999790191650 y:26.400999069213867 z:36.444000244140625
  - atom: C2' x: 1.9589999914169312 y:25.736000061035156 z:38.19599914550781
  - atom: C1' x: 3.0169999599456787 y:25.868000030517578 z:39.27899932861328
  - atom: N1 x: 2.6960000991821290 y:25.316999435424805 z:40.617000579833984
  - atom: C2 x: 3.7639999389648438 y:24.950000762939453 z:41.40399932861328
  - atom: O2 x: 4.9289999008178710 y:25.084999084472656 z:41.055999755859375
  - atom: N3 x: 3.4219999313354490 y:24.419000625610350 z:42.61800003051758
  - atom: C4 x: 2.1570000648498535 y:24.222000122070312 z:43.119998931884766
  - atom: O4 x: 2.0109999179840090 y:23.684000015258790 z:44.21900177001953
  - atom: C5 x: 1.0809999704360962 y:24.665000915527344 z:42.263999938964844
  - atom: C7 x: -0.3280000090599060 y:24.545999526977540 z:42.757999420166016
  - atom: C6 x: 1.3980000019073486 y:25.173999786376953 z:41.064998626708984
- residue DG has 22 atoms
  - atom: P x: 3.2839999198913574 y:26.125 z:34.874000549316406
  - atom: OP1 x: 2.9730000495910645 y:27.427000045776367 z:34.23400115966797
  - atom: OP2 x: 2.3570001125335693 y:24.972999572753906 z:34.70800018310547
  - atom: O5' x: 4.7270002365112305 y:25.694000244140625 z:34.374000549316406
  - atom: C5' x: 5.8720002174377440 y:26.429000854492188 z:34.78499984741211
  - atom: C4' x: 6.8429999351501465 y:25.516000747680664 z:35.494998931884766
  - atom: O4' x: 6.2280001640319820 y:24.899999618530273 z:36.65399932861328
  - atom: C3' x: 7.3550000190734860 y:24.358999252319336 z:34.64799880981445
  - atom: O3' x: 8.7139997482299800 y:24.148000717163086 z:35.01499938964844
  - atom: C2' x: 6.4650001525878910 y:23.198999404907227 z:35.07500076293945
  - atom: C1' x: 6.3249998092651370 y:23.479999542236328 z:36.55500030517578
  - atom: N9 x: 5.1560001373291020 y:22.92099952697754 z:37.22999954223633
  - atom: C8 x: 3.8529999256134033 y:22.968000411987305 z:36.803001403808594
```

```
  - atom: N7 x: 3.006999969482422 y:22.51300048828125 z:37.68600082397461
  - atom: C5 x: 3.7990000247955322 y:22.11199951171875 z:38.75199890136719
  - atom: C6 x: 3.437000036239624 y:21.558000564575195 z:40.00899887084961
  - atom: O6 x: 2.309999942779541 y:21.3439998626709 z:40.45800018310547
  - atom: N1 x: 4.552999973297119 y:21.260000228881836 z:40.77899932861328
  - atom: C2 x: 5.854000091552734 y:21.476999282836914 z:40.400001525878906
  - atom: N2 x: 6.788000106811523 y:21.107999801635742 z:41.284000396728516
  - atom: N3 x: 6.209000110626221 y:22.014999389648438 z:39.24100112915039
  - atom: C4 x: 5.132999897003174 y:22.309999465942383 z:38.47100067138672
- residue DA has 21 atoms
  - atom: P x: 9.545999526977539 y:22.961999893188477 z:34.34600067138672
  - atom: OP1 x: 10.937999725341797 y:23.468000411987305 z:34.21900177001953
  - atom: OP2 x: 8.817999839782715 y:22.490999221801758 z:33.14099884033203
  - atom: O5' x: 9.5 y:21.81399917602539 z:35.448001861572266
  - atom: C5' x: 10.081000328063965 y:22.02400016784668 z:36.736000061035156
  - atom: C4' x: 10.329000473022461 y:20.70199966430664 z:37.426998138427734
  - atom: O4' x: 9.138999938964844 y:20.261999130249023 z:38.12099838256836
  - atom: C3' x: 10.741000175476074 y:19.54400062561035 z:36.52299880981445
  - atom: O3' x: 11.741999626159668 y:18.784000396728516 z:37.20500183105469
  - atom: C2' x: 9.444000244140625 y:18.773000717163086 z:36.327999114990234
  - atom: C1' x: 8.732000350952148 y:18.981000900268555 z:37.65599822998047
  - atom: N9 x: 7.264999866485596 y:18.985000610351562 z:37.60900115966797
  - atom: C8 x: 6.453999996185303 y:19.448999404907227 z:36.606998443603516
  - atom: N7 x: 5.175000190734863 y:19.384000778198242 z:36.88800048828125
  - atom: C5 x: 5.138000011444092 y:18.83099937438965 z:38.15700149536133
  - atom: C6 x: 4.072999954223633 y:18.517000198364258 z:39.018001556396484
  - atom: N6 x: 2.7899999618530273 y:18.732999801635742 z:38.7239990234375
  - atom: N1 x: 4.375 y:17.97100067138672 z:40.21099853515625
  - atom: C2 x: 5.659999847412109 y:17.757999420166016 z:40.50899887084961
  - atom: N3 x: 6.749000072479248 y:18.011999130249023 z:39.78799819946289
  - atom: C4 x: 6.415999889373779 y:18.558000564575195 z:38.606998443603516
- residue DT has 20 atoms
  - atom: P x: 12.317000389099121 y:17.43600082397461 z:36.55500030517578
  - atom: OP1 x: 13.734999656677246 y:17.33300018310547 z:36.98099899291992
  - atom: OP2 x: 11.979999542236328 y:17.409000396728516 z:35.11199951171875
  - atom: O5' x: 11.479000091552734 y:16.302000045776367 z:37.290000915527344
  - atom: C5' x: 11.32800006866455 y:16.334999084472656 z:38.70600128173828
  - atom: C4' x: 10.307000160217285 y:15.317999839782715 z:39.159000396728516
  - atom: O4' x: 8.970999717712402 y:15.774999618530273 z:38.867000579833984
  - atom: C3' x: 10.42199993133545 y:13.913000106811523 z:38.55099868774414
  - atom: O3' x: 10.61299991607666 y:12.967000007629395 z:39.61199951171875
  - atom: C2' x: 9.071999549865723 y:13.696999549865723 z:37.87799835205078
  - atom: C1' x: 8.182999610900879 y:14.621000289916992 z:38.68199920654297
  - atom: N1 x: 6.90500020980835 y:15.03499984741211 z:38.07099914550781
  - atom: C2 x: 5.755000114440918 y:14.852999687194824 z:38.80699920654297
  - atom: O2 x: 5.739999771118164 y:14.33899974822998 z:39.91299819946289
  - atom: N3 x: 4.610000133514404 y:15.300000190734863 z:38.196998596191406
  - atom: C4 x: 4.497000217437744 y:15.899999618530273 z:36.95600128173828
  - atom: O4 x: 3.3940000534057617 y:16.273000717163086 z:36.55699920654297
  - atom: C5 x: 5.741000175476074 y:16.042999267578125 z:36.229000091552734
  - atom: C7 x: 5.723999977111816 y:16.673999786376953 z:34.87300109863281
  - atom: C6 x: 6.864999771118164 y:15.604000091552734 z:36.8129997253418
- residue DG has 22 atoms
  - atom: P x: 11.225000381469727 y:11.51099967956543 z:39.29899978637695
  - atom: OP1 x: 12.70300006866455 y:11.628000259399414 z:39.255001068115234
  - atom: OP2 x: 10.50100040435791 y:10.935999870300293 z:38.132999420166016
  - atom: O5' x: 10.819999694824219 y:10.654000282287598 z:40.57899856567383
  - atom: C5' x: 10.031000137329102 y:11.215999603271484 z:41.62900161743164
  - atom: C4' x: 8.626999855041504 y:10.654999732971191 z:41.597999572753906
```

```
  - atom: O4' x: 7.829999923706055 y:11.324000358581543 z:40.608001708984375
  - atom: C3' x: 8.498000144958496 y:9.168999671936035 z:41.27399826049805
  - atom: O3' x: 8.227999687194824 y:8.444999694824219 z:42.47800064086914
  - atom: C2' x: 7.328999996185303 y:9.090999603271484 z:40.29499816894531
  - atom: C1' x: 6.724999904632568 y:10.484000205993652 z:40.34600067138672
  - atom: N9 x: 6.067999839782715 y:10.968999862670898 z:39.12699890136719
  - atom: C8 x: 6.690000057220459 y:11.46500015258789 z:38.00699996948242
  - atom: N7 x: 5.853000164031982 y:11.904000282287598 z:37.10599899291992
  - atom: C5 x: 4.598999977111816 y:11.670999526977539 z:37.6510009765625
  - atom: C6 x: 3.306999921798706 y:11.958000183105469 z:37.13800048828125
  - atom: O6 x: 2.999000072479248 y:12.503999710083008 z:36.073001861572266
  - atom: N1 x: 2.312000036239624 y:11.548999786376953 z:38.00899887084961
  - atom: C2 x: 2.5190000534057617 y:10.954000473022461 z:39.215999603271484
  - atom: N2 x: 1.4140000343322754 y:10.656999588012695 z:39.900001525878906
  - atom: N3 x: 3.7149999141693115 y:10.678999900817871 z:39.7130012512207
  - atom: C4 x: 4.704999923706055 y:11.067999839782715 z:38.88600158691406
- residue DC has 19 atoms
  - atom: P x: 8.17199993133545 y:6.841000080108643 z:42.446998596191406
  - atom: OP1 x: 8.222999572753906 y:6.381999969482422 z:43.845001220703125
  - atom: OP2 x: 9.154999732971191 y:6.321000099182129 z:41.47800064086914
  - atom: O5' x: 6.704999923706055 y:6.544000148773193 z:41.90299987792969
  - atom: C5' x: 5.586999893188477 y:6.679999828338623 z:42.770999908447266
  - atom: C4' x: 4.313000202178955 y:6.329999923706055 z:42.04499816894531
  - atom: O4' x: 4.117000102996826 y:7.260000228881836 z:40.965999603271484
  - atom: C3' x: 4.315999984741211 y:4.946000099182129 z:41.40299987792969
  - atom: O3' x: 3.621000051498413 y:4.050000190734863 z:42.27799987792969
  - atom: C2' x: 3.617000102996826 y:5.1479997634887695 z:40.055999755859375
  - atom: C1' x: 3.2750000953674316 y:6.645999908447266 z:40.01100158691406
  - atom: N1 x: 3.4820001125335693 y:7.343999862670898 z:38.71900177001953
  - atom: C2 x: 2.378000020980835 y:7.947999954223633 z:38.071998596191406
  - atom: O2 x: 1.2549999952316284 y:7.870999813079834 z:38.58700180053711
  - atom: N3 x: 2.571000099182129 y:8.600000381469727 z:36.9109992980957
  - atom: C4 x: 3.7929999828338623 y:8.67300033569336 z:36.38600158691406
  - atom: N4 x: 3.941999912261963 y:9.338000297546387 z:35.24399948120117
  - atom: C5 x: 4.923999786376953 y:8.069000244140625 z:37.013999938964844
  - atom: C6 x: 4.724999904632568 y:7.420000076293945 z:38.16299819946289
- residue DG has 22 atoms
  - atom: P x: 3.5369999408721924 y:2.490000009536743 z:41.92900085449219
  - atom: OP1 x: 3.2130000591278076 y:1.7960000038146973 z:43.191001892089844
  - atom: OP2 x: 4.730999946594238 y:2.0999999046325684 z:41.15800094604492
  - atom: O5' x: 2.25 y:2.365000009536743 z:41.00199890136719
  - atom: C5' x: 0.9470000267028809 y:2.680999994277954 z:41.50899887084961
  - atom: C4' x: -0.08699999749660492 y:2.5480000972747803 z:40.415000915527344
  - atom: O4' x: 0.22200000286102295 y:3.51200008392334 z:39.380001068115234
  - atom: C3' x: -0.08799999952316284 y:1.1829999685287476 z:39.724998474121094
  - atom: O3' x: -1.4179999828338623 y:0.8140000104904175 z:39.356998443603516
  - atom: C2' x: 0.7760000228881836 y:1.4210000038146973 z:38.50400161743164
  - atom: C1' x: 0.375 y:2.8369998931884766 z:38.14500045776367
  - atom: N9 x: 1.3359999656677246 y:3.5820000171661377 z:37.33599853515625
  - atom: C8 x: 2.7079999446868896 y:3.490000009536743 z:37.36399841308594
  - atom: N7 x: 3.2909998893737793 y:4.26800012588501 z:36.49399948120117
  - atom: C5 x: 2.2430000305175781 y:4.920000076293945 z:35.858001708984375
  - atom: C6 x: 2.2590000629425049 y:5.873000144958496 z:34.814998626708984
  - atom: O6 x: 3.2360000610351562 y:6.339000225067139 z:34.21500015258789
  - atom: N1 x: 0.9739999771118164 y:6.2789998054504395 z:34.470001220703125
  - atom: C2 x: -0.17900000512599945 y:5.818999767303467 z:35.04999923706055
  - atom: N2 x: -1.3250000476837158 y:6.327000141143799 z:34.58100128173828
  - atom: N3 x: -0.20900000631809235 y:4.921999931335449 z:36.020999908447266
  - atom: C4 x: 1.0319999456405640 y:4.517000198364258 z:36.37200164794922
```

```
    - residue DC has 19 atoms
      - atom: P x: -1.694000005722046 y:-0.6140000224113464 z:38.65599822998047
      - atom: OP1 x: -2.994999885559082 y:-1.1480000019073486 z:39.14899826049805
      - atom: OP2 x: -0.46000000834465027 y:-1.430999994277954 z:38.821998596191406
      - atom: O5' x: -1.8730000257492065 y:-0.24400000274181366 z:37.11000061035156
      - atom: C5' x: -2.808000087738037 y:0.7670000195503235 z:36.698001861572266
      - atom: C4' x: -2.8580000400543213 y:0.8840000033378601 z:35.1879997253418
      - atom: O4' x: -1.7769999504089355 y:1.7239999771118164 z:34.69900131225586
      - atom: C3' x: -2.756999969482422 y:-0.4309999942779541 z:34.39899826049805
      - atom: O3' x: -3.569999933242798 y:-0.4269999861717224 z:33.20000076293945
      - atom: C2' x: -1.312000036239624 y:-0.4320000112056732 z:33.9370002746582
      - atom: C1' x: -1.125 y:1.0440000295639038 z:33.637001037597656
      - atom: N1 x: 0.2680000066757202 y:1.524999976158142 z:33.55699920654297
      - atom: C2 x: 0.5590000152587891 y:2.6029999256134033 z:32.71099853515625
      - atom: O2 x: -0.36800000071525574 y:3.1510000228881836 z:32.09000015258789
      - atom: N3 x: 1.8370000123977661 y:3.0220000743865967 z:32.59600067138672
      - atom: C4 x: 2.8010001182556152 y:2.4200000762939453 z:33.29399871826172
      - atom: N4 x: 4.046999931335449 y:2.8559999465942383 z:33.137001037597656
      - atom: C5 x: 2.5299999713897705 y:1.3420000076293945 z:34.178001403808594
      - atom: C6 x: 1.2660000324249268 y:0.9300000071525574 z:34.27899932861328
    - residue MN has 1 atoms
      - atom: MN x: -0.07199999690055847 y:21.77899932861328 z:37.319000244140625
    - residue HOH has 1 atoms
      - atom: O x: 6.046000003814697 y:3.1649999618530273 z:39.082000732421875
    - residue HOH has 1 atoms
      - atom: O x: 13.569999694824219 y:10.14799976348877 z:41.25299835205078
    - residue HOH has 1 atoms
      - atom: O x: -5.609000205993652 y:-0.41600000858306885 z:34.5
    - residue HOH has 1 atoms
      - atom: O x: 4.633999824523926 y:30.27400016784668 z:40.766998291015625
    - residue HOH has 1 atoms
      - atom: O x: -5.763999938964844 y:33.71799850463867 z:44.33599853515625
    - residue HOH has 1 atoms
      - atom: O x: 2.937000036239624 y:31.315000534057617 z:42.645999908447266
    - residue HOH has 1 atoms
      - atom: O x: 7.23199987411499 y:5.645999908447266 z:38.564998626708984
- chain  <Chain id=D> has 9 residues
    - residue DG has 23 atoms
      - atom: OP3 x: 21.908000946044922 y:-5.788000106811523 z:31.12700080871582
      - atom: P x: 22.32200050354004 y:-5.873000144958496 z:29.663000106811523
      - atom: OP1 x: 23.104999542236328 y:-4.658999919891357 z:29.17799949645996
      - atom: OP2 x: 22.988000869750977 y:-7.199999809265137 z:29.34600067138672
      - atom: O5' x: 20.930999755859375 y:-5.881999969482422 z:28.815000534057617
      - atom: C5' x: 19.81999969482422 y:-6.675000190734863 z:29.256000518798828
      - atom: C4' x: 19.066999435424805 y:-7.238999843597412 z:28.073999404907227
      - atom: O4' x: 18.231000900268555 y:-6.247000217437744 z:27.457000732421875
      - atom: C3' x: 19.908000946044922 y:-7.807000160217285 z:26.941999435424805
      - atom: O3' x: 19.149999618530273 y:-8.842000007629395 z:26.315000534057617
      - atom: C2' x: 20.09000015258789 y:-6.60699987411499 z:26.030000686645508
      - atom: C1' x: 18.798999786376953 y:-5.822999954223633 z:26.229999542236328
      - atom: N9 x: 18.93199920654297 y:-4.373000144958496 z:26.301000595092773
      - atom: C8 x: 19.9060001373291 y:-3.635999917984009 z:26.934999465942383
      - atom: N7 x: 19.700000762939453 y:-2.3499999046325684 z:26.86199951171875
      - atom: C5 x: 18.525999069213867 y:-2.2360000610351562 z:26.134000778198242
      - atom: C6 x: 17.79400062561035 y:-1.0950000286102295 z:25.75200080871582
      - atom: O6 x: 18.014999389648438 y:0.0860000029206276 z:26.02899932861328
      - atom: N1 x: 16.68000030517578 y:-1.4359999895095825 z:24.983999252319336
      - atom: C2 x: 16.304000854492188 y:-2.7179999351501465 z:24.660999298095703
      - atom: N2 x: 15.206000328063965 y:-2.86299991607666 z:23.91699981689453
```

```
  - atom: N3 x: 16.96299934387207 y:-3.7890000343322754 z:25.04400062561035
  - atom: C4 x: 18.05699920654297 y:-3.4769999980926514 z:25.764999389648438
- residue DT has 20 atoms
  - atom: P x: 19.722999572753906 y:-9.595000267028809 z:25.020999908447266
  - atom: OP1 x: 19.643999099731445 y:-11.03600025177002 z:25.367000579833984
  - atom: OP2 x: 21.0049991607666 y:-9.012999534606934 z:24.559999465942383
  - atom: O5' x: 18.625999450683594 y:-9.286999702453613 z:23.91699981689453
  - atom: C5' x: 17.94099998474121 y:-8.050000190734863 z:23.905000686645508
  - atom: C4' x: 17.011999130249023 y:-7.999000072479248 z:22.722000122070312
  - atom: O4' x: 16.64699935913086 y:-6.625 z:22.472999572753906
  - atom: C3' x: 17.648000717163086 y:-8.508000373840332 z:21.43199920654297
  - atom: O3' x: 16.665000915527344 y:-9.154999732971191 z:20.62299919128418
  - atom: C2' x: 18.21299934387207 y:-7.251999855041504 z:20.790000915527344
  - atom: C1' x: 17.34600067138672 y:-6.119999885559082 z:21.343000411987305
  - atom: N1 x: 18.05500030517578 y:-4.886000156402588 z:21.767000198364258
  - atom: C2 x: 17.41699981689453 y:-3.6809999942779541 z:21.559999465942383
  - atom: O2 x: 16.343000411987305 y:-3.5769999027252197 z:20.979000091552734
  - atom: N3 x: 18.089000701904297 y:-2.5910000801086426 z:22.051000595092773
  - atom: C4 x: 19.31100082397461 y:-2.578000068664551 z:22.687000274658203
  - atom: O4 x: 19.77400016784668 y:-1.5240000486373901 z:23.097999572753906
  - atom: C5 x: 19.947999954223633 y:-3.859999895095825 z:22.822999954223633
  - atom: C7 x: 21.301000595092773 y:-3.927000045776367 z:23.461000442504883
  - atom: C6 x: 19.29599952697754 y:-4.940000057220459 z:22.368000030517578
- residue DC has 19 atoms
  - atom: P x: 17.084999084472656 y:-9.819999694824219 z:19.219999313354492
  - atom: OP1 x: 16.177000045776367 y:-10.982000350952148 z:18.999000549316406
  - atom: OP2 x: 18.55699920654297 y:-10.012999534606934 z:19.167999267578125
  - atom: O5' x: 16.697999954223633 y:-8.699999809265137 z:18.16200065612793
  - atom: C5' x: 15.357000350952148 y:-8.246999740600586 z:18.04599952697754
  - atom: C4' x: 15.291000366210938 y:-7.078000068664551 z:17.093000411987305
  - atom: O4' x: 15.998000144958496 y:-5.955999851226807 z:17.679000854492188
  - atom: C3' x: 15.972000122070312 y:-7.322999954223633 z:15.744999885559082
  - atom: O3' x: 15.348999977111816 y:-6.510000228881836 z:14.746000289916992
  - atom: C2' x: 17.37299919128418 y:-6.801000118255615 z:15.98900032043457
  - atom: C1' x: 17.03700065612793 y:-5.560999870300293 z:16.798999786376953
  - atom: N1 x: 18.117000579833984 y:-4.954999923706055 z:17.600000381469727
  - atom: C2 x: 18.066999435424805 y:-3.5810000896453857 z:17.847000122070312
  - atom: O2 x: 17.121999740600586 y:-2.9189999103546143 z:17.381999969482422
  - atom: N3 x: 19.047000885009766 y:-3.006000004196167 z:18.57200050354004
  - atom: C4 x: 20.05699920654297 y:-3.744999885559082 z:19.03499984741211
  - atom: N4 x: 21.01300048828125 y:-3.13100004196167 z:19.738000869750977
  - atom: C5 x: 20.13599967956543 y:-5.1479997634887695 z:18.79599952697754
  - atom: C6 x: 19.152999877929688 y:-5.706999778747559 z:18.086000442504883
- residue DG has 22 atoms
  - atom: P x: 15.234000205993652 y:-7.048999786376953 z:13.239999771118164
  - atom: OP1 x: 14.206999778747559 y:-8.126999855041504 z:13.246000289916992
  - atom: OP2 x: 16.608999252319336 y:-7.3379998207092285 z:12.744999885559082
  - atom: O5' x: 14.673999786376953 y:-5.798999786376953 z:12.423999786376953
  - atom: C5' x: 13.835000038146973 y:-4.823999881744385 z:13.045000076293945
  - atom: C4' x: 14.350000381469727 y:-3.431999921798706 z:12.758999824523926
  - atom: O4' x: 15.515000343322754 y:-3.1540000438690186 z:13.57800006866455
  - atom: C3' x: 14.777999877929688 y:-3.1989998817443848 z:11.3100004196167
  - atom: O3' x: 14.291000366210938 y:-1.9329999685287476 z:10.85200023651123
  - atom: C2' x: 16.29599952697754 y:-3.2109999656677246 z:11.37399959564209
  - atom: C1' x: 16.589000701904297 y:-2.687999963760376 z:12.770999908447266
  - atom: N9 x: 17.829000047302246 y:-3.2009999752044678 z:13.352000236511230
  - atom: C8 x: 18.270000457763672 y:-4.506000041961670 z:13.321000099182129
  - atom: N7 x: 19.406000137329100 y:-4.679999828338623 z:13.937999725341797
  - atom: C5 x: 19.740999221801758 y:-3.417999982833862 z:14.406999588012695
```

```
      - atom: C6 x: 20.8700008392334 y:-2.99399995803833 z:15.142999649047852
      - atom: O6 x: 21.81599998474121 y:-3.6710000038146973 z:15.54800033569336
      - atom: N1 x: 20.833999633789062 y:-1.628999948501587 z:15.395999908447266
      - atom: C2 x: 19.836000442504883 y:-0.7789999842643738 z:14.994000434875488
      - atom: N2 x: 19.996999740600586 y:0.5090000033378601 z:15.331999778747559
      - atom: N3 x: 18.766000747680664 y:-1.1640000343322754 z:14.309000015258789
      - atom: C4 x: 18.785999298095703 y:-2.489000082015991 z:14.050000190734863
    - residue DG has 22 atoms
      - atom: P x: 14.321999549865723 y:-1.5820000171661377 z:9.281999588012695
      - atom: OP1 x: 13.01200008392334 y:-0.9649999737739563 z:8.95199966430664
      - atom: OP2 x: 14.77400016784668 y:-2.7820000648498535 z:8.529999732971191
      - atom: O5' x: 15.46500015258789 y:-0.47600001096725464 z:9.168999671936035
      - atom: C5' x: 15.302000045776367 y:0.796999990940094 z:9.78600025177002
      - atom: C4' x: 16.645999908447266 y:1.4179999828338623 z:10.090999603271484
      - atom: O4' x: 17.479999542236328 y:0.46399998664855957 z:10.789999961853027
      - atom: C3' x: 17.46299934387207 y:1.8680000305175781 z:8.876999855041504
      - atom: O3' x: 18.23699951171875 y:3.005000114440918 z:9.279000282287598
      - atom: C2' x: 18.448999404907227 y:0.7289999723434448 z:8.701000213623047
      - atom: C1' x: 18.743999481201172 y:0.4269999861717224 z:10.154000282287598
      - atom: N9 x: 19.35700035095215 y:-0.8669999837875366 z:10.437000274658203
      - atom: C8 x: 18.961999893188477 y:-2.0980000495910645 z:9.970000267028809
      - atom: N7 x: 19.707000732421875 y:-3.0769999027252197 z:10.41100025177002
      - atom: C5 x: 20.650999069213867 y:-2.4560000896453857 z:11.215999603271484
      - atom: C6 x: 21.711999893188477 y:-3.006999969482422 z:11.972000122070312
      - atom: O6 x: 22.041000366210938 y:-4.203999996185303 z:12.086999893188477
      - atom: N1 x: 22.422000885009766 y:-2.015000104904175 z:12.651000022888184
      - atom: C2 x: 22.142000198364258 y:-0.6679999828338623 z:12.607999801635742
      - atom: N2 x: 22.93899917602539 y:0.13600000739097595 z:13.335000038146973
      - atom: N3 x: 21.152999877929688 y:-0.1459999978542328 z:11.904999732971191
      - atom: C4 x: 20.45199966430664 y:-1.090000033378601 z:11.239999771118164
    - residue HOH has 1 atoms
      - atom: O x: 18.135000228881836 y:3.9839999675750732 z:12.008999824523926
    - residue HOH has 1 atoms
      - atom: O x: 13.862000465393066 y:-5.619999885559082 z:23.08300018310547
    - residue HOH has 1 atoms
      - atom: O x: 21.132999420166016 y:-7.02400016784668 z:13.020999908447266
    - residue HOH has 1 atoms
      - atom: O x: 24.173999786376953 y:-5.079999923706055 z:33.534000396728516
- chain  <Chain id=A> has 427 residues
    - residue THR has 7 atoms
      - atom: N x: 3.50600004196167 y:-24.027999877929688 z:29.959999084472656
      - atom: CA x: 4.690999984741211 y:-23.68400001525879 z:30.798999786376953
      - atom: C x: 5.771999835968018 y:-24.729000091552734 z:30.599000930786133
      - atom: O x: 6.264999866485596 y:-25.347000122070312 z:31.542999267578125
      - atom: CB x: 4.321000099182129 y:-23.61400032043457 z:32.292999267578125
      - atom: OG1 x: 4.0879998207092285 y:-24.940000534057617 z:32.81100082397461
      - atom: CG2 x: 3.069000005722046 y:-22.75 z:32.46699905395508
    - residue LEU has 8 atoms
      - atom: N x: 6.114999771118164 y:-24.933000564575195 z:29.336999893188477
      - atom: CA x: 7.14900016784668 y:-25.878000259399414 z:28.98900032043457
      - atom: C x: 8.472999572753906 y:-25.23200035095215 z:29.381999969482422
      - atom: O x: 9.387999534606934 y:-25.909000396728516 z:29.861000061035156
      - atom: CB x: 7.109000205993652 y:-26.163999557495117 z:27.488000869750977
      - atom: CG x: 5.886000156402588 y:-26.91900062561035 z:26.95199966430664
      - atom: CD1 x: 6.002999782562256 y:-27.079999923706055 z:25.44099998474121
      - atom: CD2 x: 5.789999961853027 y:-28.275999069213867 z:27.613000869750977
    - residue ASN has 8 atoms
      - atom: N x: 8.552000045776367 y:-23.913999557495117 z:29.201000213623047
      - atom: CA x: 9.74899959564209 y:-23.145999908447266 z:29.527999877929688
```

```
  - atom: C x: 9.545999526977539 y:-22.253000259399414 z:30.756999969482422
  - atom: O x: 10.102999687194824 y:-21.159000396728516 z:30.836000442504883
  - atom: CB x: 10.137999534606934 y:-22.27899932861328 z:28.33799934387207
  - atom: CG x: 10.458000183105469 y:-23.090999603271484 z:27.111000061035156
  - atom: OD1 x: 11.486000061035156 y:-23.759000778198242 z:27.051000595092773
  - atom: ND2 x: 9.572999954223633 y:-23.04400062561035 z:26.121000289916992
- residue GLY has 4 atoms
  - atom: N x: 8.763999938964844 y:-22.740999221801758 z:31.716999053955078
  - atom: CA x: 8.47599983215332 y:-21.97800064086914 z:32.922000885009766
  - atom: C x: 9.657999992370605 y:-21.47800064086914 z:33.73500061035156
  - atom: O x: 9.645999908447266 y:-20.341999053955078 z:34.20500183105469
- residue GLY has 4 atoms
  - atom: N x: 10.67199993133545 y:-22.316999435424805 z:33.91999816894531
  - atom: CA x: 11.829999923706055 y:-21.899999618530273 z:34.689998626708984
  - atom: C x: 12.550999641418457 y:-20.709999084472656 z:34.073001861572266
  - atom: O x: 13.067000389099121 y:-19.834999084472656 z:34.7869987487793
- residue ILE has 8 atoms
  - atom: N x: 12.583999633789062 y:-20.67799949645996 z:32.74399948120117
  - atom: CA x: 13.246000289916992 y:-19.611000061035156 z:32.01900100708008
  - atom: C x: 12.480999946594238 y:-18.308000564575195 z:32.0890007019043
  - atom: O x: 13.067999839782715 y:-17.257999420166016 z:32.32899856567383
  - atom: CB x: 13.434000015258789 y:-19.965999603271484 z:30.530000686645508
  - atom: CG1 x: 14.45300006866455 y:-21.104000091552734 z:30.38800048828125
  - atom: CG2 x: 13.859999656677246 y:-18.719999313354492 z:29.7450008392334
  - atom: CD1 x: 14.708000183105469 y:-21.54400062561035 z:28.957000732421875
- residue THR has 7 atoms
  - atom: N x: 11.17199993133545 y:-18.36400032043457 z:31.886999130249023
  - atom: CA x: 10.392000198364258 y:-17.139999389648438 z:31.91200065612793
  - atom: C x: 10.234000205993652 y:-16.545000076293945 z:33.3129997253418
  - atom: O x: 9.973999977111816 y:-15.357000350952148 z:33.443000793457O3
  - atom: CB x: 9.015999794006348 y:-17.327999114990234 z:31.190000534057617
  - atom: OG1 x: 8.112000465393066 y:-18.10099983215332 z:31.989999771118164
  - atom: CG2 x: 9.234999656677246 y:-18.034000396728516 z:29.868000030517578
- residue ASP has 8 atoms
  - atom: N x: 10.395000457763672 y:-17.35700035095215 z:34.356998443603516
  - atom: CA x: 10.33899974822998 y:-16.850000381469727 z:35.73699951171875
  - atom: C x: 11.638999938964844 y:-16.07200050354004 z:35.959999084472656
  - atom: O x: 11.659000396728516 y:-14.982999801635742 z:36.53799819946289
  - atom: CB x: 10.298999786376953 y:-17.986999511718750 z:36.75400161743164
  - atom: CG x: 8.946999549865723 y:-18.631000518798828 z:36.85300064086914
  - atom: OD1 x: 7.943999767303467 y:-17.89299964904785 z:36.93299865722656
  - atom: OD2 x: 8.88599967956543 y:-19.878000259399414 z:36.87099838256836
- residue MET has 8 atoms
  - atom: N x: 12.729999542236328 y:-16.667999267578125 z:35.49599838256836
  - atom: CA x: 14.029999732971191 y:-16.049999237060547 z:35.595001220703125
  - atom: C x: 13.96500015258789 y:-14.680999755859375 z:34.904998779296875
  - atom: O x: 14.3149995803833 y:-13.659000396728516 z:35.48699951171875
  - atom: CB x: 15.060999870300293 y:-16.926000595092773 z:34.89899826049805
  - atom: CG x: 16.468000411987305 y:-16.416000366210938 z:35.000999450683594
  - atom: SD x: 17.45599937438965 y:-17.030000686645508 z:33.643001556396484
  - atom: CE x: 18.0049991607666 y:-15.479999542236328 z:32.88600158691406
- residue LEU has 8 atoms
  - atom: N x: 13.51099967956543 y:-14.67199993133545 z:33.659000396728516
  - atom: CA x: 13.411999702453613 y:-13.447999954223633 z:32.88999938964844
  - atom: C x: 12.569999694824219 y:-12.394000053405762 z:33.59400177001953
  - atom: O x: 12.833000183105469 y:-11.189000129699707 z:33.472999572753906
  - atom: CB x: 12.795000076293945 y:-13.741000175476074 z:31.5310001373291
  - atom: CG x: 13.562000274658203 y:-14.668999671936035 z:30.604000091552734
  - atom: CD1 x: 12.758000373840332 y:-14.875 z:29.336999893188477
```

```
  - atom: CD2 x: 14.91100025177002 y:-14.074000358581543 z:30.27899932861328
- residue THR has 7 atoms
  - atom: N x: 11.553000450134277 y:-12.848999977111816 z:34.323001861572266
  - atom: CA x: 10.657999992370605 y:-11.9399995803833 z:35.020000045776367
  - atom: C x: 11.28600025177002 y:-11.27299976348877 z:36.21799850463867
  - atom: O x: 11.097000122070312 y:-10.074999809265137 z:36.4109992980957
  - atom: CB x: 9.369000434875488 y:-12.640999794006348 z:35.45600128173828
  - atom: OG1 x: 8.581999778747559 y:-12.916000366210938 z:34.29399871826172
  - atom: CG2 x: 8.564000129699707 y:-11.765999794006348 z:36.387001037597656
- residue GLU has 9 atoms
  - atom: N x: 12.020999908447266 y:-12.038000106811523 z:37.02199935913086
  - atom: CA x: 12.682999610900879 y:-11.47599983215332 z:38.189998626708984
  - atom: C x: 13.690999984741211 y:-10.45199966430664 z:37.67300033569336
  - atom: O x: 13.807000160217285 y:-9.345000267028809 z:38.196998596191406
  - atom: CB x: 13.37399959564209 y:-12.588000297546387 z:38.98899841308594
  - atom: CG x: 14.23799991607666 y:-12.11400032043457 z:40.15599822998047
  - atom: CD x: 14.40999984741211 y:-13.187000274658203 z:41.233001708984375
  - atom: OE1 x: 14.732999801635742 y:-14.33899974822998 z:40.88600158691406
  - atom: OE2 x: 14.22599983215332 y:-12.878000259399414 z:42.43600082397461
- residue LEU has 8 atoms
  - atom: N x: 14.407999992370605 y:-10.817999839782715 z:36.62200164794922
  - atom: CA x: 15.376999855041504 y:-9.906000137329102 z:36.029998779296875
  - atom: C x: 14.710000038146973 y:-8.588000297546387 z:35.63999938964844
  - atom: O x: 15.276000022888184 y:-7.519999980926514 z:35.840999603271484
  - atom: CB x: 16.006000518798828 y:-10.550999641418457 z:34.801998138427734
  - atom: CG x: 17.10099983215332 y:-11.543000221252441 z:35.15599822998047
  - atom: CD1 x: 17.541000366210938 y:-12.317999839782715 z:33.928001403808594
  - atom: CD2 x: 18.26300048828125 y:-10.758000373840332 z:35.744998931884766
- residue ALA has 5 atoms
  - atom: N x: 13.50100040435791 y:-8.6850004196167 z:35.09000015258789
  - atom: CA x: 12.727999687194824 y:-7.525000095367432 z:34.66899871826172
  - atom: C x: 12.454999923706055 y:-6.618000030517578 z:35.85900115966797
  - atom: O x: 12.86400032043457 y:-5.453999996185303 z:35.887001037597656
  - atom: CB x: 11.416000366210938 y:-7.9730000495910645 z:34.04499816894531
- residue ASN has 8 atoms
  - atom: N x: 11.75100040435791 y:-7.1519999504089355 z:36.84299850463867
  - atom: CA x: 11.447999954223633 y:-6.377999782562256 z:38.02799987792969
  - atom: C x: 12.694999694824219 y:-5.685999870300293 z:38.54800033569336
  - atom: O x: 12.654999732971191 y:-4.5079998970031774 z:38.917999267578125
  - atom: CB x: 10.857000350952148 y:-7.275000095367432 z:39.09700012207031
  - atom: CG x: 9.361000061035156 y:-7.355999946594238 z:38.999000549316406
  - atom: OD1 x: 8.668000221252441 y:-6.34499979019165 z:39.154998779296875
  - atom: ND2 x: 8.843999862670898 y:-8.548999786376953 z:38.73500061035156
- residue PHE has 11 atoms
  - atom: N x: 13.807000160217285 y:-6.4140000343322754 z:38.56399917602539
  - atom: CA x: 15.053000450134277 y:-5.831999778747559 z:39.0260009765625
  - atom: C x: 15.369000434875488 y:-4.565999984741211 z:38.233001708984375
  - atom: O x: 15.411999702453613 y:-3.4700000286102295 z:38.79100036621094
  - atom: CB x: 16.20800018310547 y:-6.817999839782715 z:38.86600112915039
  - atom: CG x: 17.55500030517578 y:-6.203999996185303 z:39.13100051879883
  - atom: CD1 x: 18.569000244140625 y:-6.275000095367432 z:38.180999755859375
  - atom: CD2 x: 17.791000366210938 y:-5.498000144958496 z:40.30799865722656
  - atom: CE1 x: 19.79800033569336 y:-5.644000053405762 z:38.39899826049805
  - atom: CE2 x: 19.00200080871582 y:-4.873000144958496 z:40.52799987792969
  - atom: CZ x: 20.007999420166016 y:-4.941999912261963 z:39.57600021362305
- residue GLU has 9 atoms
  - atom: N x: 15.581000328063965 y:-4.728000164031982 z:36.926998138427734
  - atom: CA x: 15.913999557495117 y:-3.611999988555908 z:36.04999923706055
  - atom: C x: 14.92199993133545 y:-2.4560000896453857 z:36.1609992980957
```

```
  - atom: O x: 15.28499984741211 y:-1.3009999990463257 z:35.96900177001953
  - atom: CB x: 16.007999420166016 y:-4.086999893188477 z:34.59299850463867
  - atom: CG x: 17.152000427246094 y:-5.060999870300293 z:34.27799987792969
  - atom: CD x: 18.538000106811523 y:-4.489999771118164 z:34.58700180053711
  - atom: OE1 x: 18.69700050354004 y:-3.2609999917984009 z:34.5260009765625
  - atom: OE2 x: 19.479000091552734 y:-5.265999794006348 z:34.875
- residue LYS has 9 atoms
  - atom: N x: 13.677000045776367 y:-2.7669999599456787 z:36.48400115966797
  - atom: CA x: 12.642999649047852 y:-1.753000020980835 z:36.61600112915039
  - atom: C x: 12.543999671936035 y:-1.121999979019165 z:37.99599838256836
  - atom: O x: 12.244000434875488 y:0.06400000303983688 z:38.12799835205078
  - atom: CB x: 11.277999877929688 y:-2.3440001010894775 z:36.2859992980957
  - atom: CG x: 10.192999839782715 y:-1.8370000123977661 z:37.207000732421875
  - atom: CD x: 8.888999938964844 y:-1.6490000486373901 z:36.49800109863281
  - atom: CE x: 8.100000381469727 y:-0.5469999990940094 z:37.20000076293945
  - atom: NZ x: 8.9350004196167 y:0.6930000185966492 z:37.34199905395508
- residue ASN has 8 atoms
  - atom: N x: 12.75100040435791 y:-1.9199999570846558 z:39.03200149536133
  - atom: CA x: 12.637999534606934 y:-1.3869999647140503 z:40.36899948120117
  - atom: C x: 13.968000411987305 y:-0.8429999947547913 z:40.900001525878906
  - atom: O x: 14.02299976348877 y:0.24799999594688416 z:41.45899963378906
  - atom: CB x: 12.067999839782715 y:-2.4579999446868896 z:41.319000244140625
  - atom: CG x: 10.592000007629395 y:-2.7790000438690186 z:41.04800033569336
  - atom: OD1 x: 9.795999526977539 y:-1.8949999809265137 z:40.73699951171875
  - atom: ND2 x: 10.22700023651123 y:-4.045000076293945 z:41.18899917602539
- residue VAL has 7 atoms
  - atom: N x: 15.041999816894531 y:-1.590999960899353 z:40.698001861572266
  - atom: CA x: 16.341999053955078 y:-1.1859999895095825 z:41.19900131225586
  - atom: C x: 17.093000411987305 y:-0.3190000057220459 z:40.21500015258789
  - atom: O x: 17.384000778198242 y:0.828000009059906 z:40.51300048828125
  - atom: CB x: 17.204999923706055 y:-2.4189999103546143 z:41.53900146484375
  - atom: CG1 x: 18.464000701904297 y:-1.99399995803833 z:42.32699966430664
  - atom: CG2 x: 16.385000228881836 y:-3.4179999828338623 z:42.33000183105469
- residue SER has 6 atoms
  - atom: N x: 17.406999588012695 y:-0.871999979019165 z:39.04600143432617
  - atom: CA x: 18.14699935913086 y:-0.14800000190734863 z:37.999000549316406
  - atom: C x: 17.43199920654297 y:1.0440000295639038 z:37.369998931884766
  - atom: O x: 18.065000534057617 y:1.8589999675750732 z:36.69300079345703
  - atom: CB x: 18.55500030517578 y:-1.0989999771118164 z:36.875
  - atom: OG x: 19.535999298095703 y:-2.0239999294281006 z:37.31100082397461
- residue GLN has 9 atoms
  - atom: N x: 16.125 y:1.1460000276565552 z:37.58000183105469
  - atom: CA x: 15.355999946594238 y:2.244999885559082 z:37.012001037597656
  - atom: C x: 15.548999786376953 y:2.385999917984009 z:35.499000549316406
  - atom: O x: 15.527999877929688 y:3.4839999675750732 z:34.96200180053711
  - atom: CB x: 15.715999603271484 y:3.553999900817871 z:37.7140007019043
  - atom: CG x: 15.02299976348877 y:3.7219998836517334 z:39.07899856567383
  - atom: CD x: 15.668000221252441 y:4.795000076293945 z:39.96500015258789
  - atom: OE1 x: 16.8799991607666 y:4.752999782562256 z:40.2400016784668
  - atom: NE2 x: 14.861000061035156 y:5.743000030517578 z:40.42499923706055
- residue ALA has 5 atoms
  - atom: N x: 15.746000289916992 y:1.2569999694824219 z:34.82899856567383
  - atom: CA x: 15.914999961853027 y:1.1920000314712524 z:33.37699890136719
  - atom: C x: 14.652999877929688 y:0.49300000071525574 z:32.85499954223633
  - atom: O x: 14.583000183105469 y:-0.7369999885559082 z:32.79999923706055
  - atom: CB x: 17.15399932861328 y:0.3790000081062317 z:33.03300094604492
- residue ILE has 8 atoms
  - atom: N x: 13.656999588012695 y:1.2949999570846558 z:32.49599838256836
  - atom: CA x: 12.378000259399414 y:0.7940000295639038 z:32.0260009765625
```

```
  - atom: C x: 12.460000038146973 y:0.032999999821186066 z:30.702999114990234
  - atom: O x: 11.57699966430664 y:-0.7519999742507935 z:30.37299919128418
  - atom: CB x: 11.407999992370605 y:1.9550000429153442 z:31.85700035095215
  - atom: CG1 x: 10.034000396728516 y:1.4700000286102295 z:31.36199951171875
  - atom: CG2 x: 12.008999824523926 y:2.937000036239624 z:30.867000579833984
  - atom: CD1 x: 9.13700008392334 y:0.8830000162124634 z:32.4379997253418
- residue HIS has 10 atoms
  - atom: N x: 13.517000198364258 y:0.2529999911785126 z:29.94099998474121
  - atom: CA x: 13.626999855041504 y:-0.42500001192092896 z:28.665000915527344
  - atom: C x: 14.244000434875488 y:-1.7960000038146973 z:28.760000228881836
  - atom: O x: 14.008999824523926 y:-2.6500000953674316 z:27.905000686645508
  - atom: CB x: 14.366999626159668 y:0.4620000123977661 z:27.674999237060547
  - atom: CG x: 13.645999908447266 y:1.7430000305175781 z:27.393999099731445
  - atom: ND1 x: 12.3100004196167 y:1.7760000228881836 z:27.051000595092773
  - atom: CD2 x: 14.04800033569336 y:3.0329999923706055 z:27.476999282836914
  - atom: CE1 x: 11.918999671936035 y:3.0320000648498535 z:26.941999435424805
  - atom: NE2 x: 12.954000473022461 y:3.813999891281128 z:27.19499969482422
- residue LYS has 9 atoms
  - atom: N x: 15.031999588012695 y:-2.0220000743865967 z:29.798999786376953
  - atom: CA x: 15.621000289916992 y:-3.3359999656677246 z:29.981000900268555
  - atom: C x: 14.463000297546387 y:-4.136000156402588 z:30.52199935913086
  - atom: O x: 14.262999534606934 y:-5.2829999923706055 z:30.141000747680664
  - atom: CB x: 16.773000717163086 y:-3.299999952316284 z:30.996000289916992
  - atom: CG x: 17.975000381469727 y:-2.4779999256134033 z:30.540000915527344
  - atom: CD x: 19.246000289916992 y:-2.9200000762939453 z:31.253000259399414
  - atom: CE x: 20.474000930786133 y:-2.2990000247955322 z:30.60099983215332
  - atom: NZ x: 21.74799919128418 y:-2.9100000858306885 z:31.034000396728516
- residue TYR has 12 atoms
  - atom: N x: 13.689000129699707 y:-3.5139999389648438 z:31.408000946044922
  - atom: CA x: 12.527999877929688 y:-4.175000190734863 z:31.989999771118164
  - atom: C x: 11.668999671936035 y:-4.672999858856201 z:30.836999893188477
  - atom: O x: 11.380999565124512 y:-5.86899995803833 z:30.731000900268555
  - atom: CB x: 11.722999572753906 y:-3.194999933242798 z:32.847999572753906
  - atom: CG x: 10.309000015258789 y:-3.6470000743865967 z:33.15999984741211
  - atom: CD1 x: 10.064000129699707 y:-4.732999801635742 z:33.992000579833984
  - atom: CD2 x: 9.215999603271484 y:-2.9639999866485596 z:32.64899826049805
  - atom: CE1 x: 8.758999824523926 y:-5.125 z:34.310001373291016
  - atom: CE2 x: 7.9070000648498535 y:-3.3469998836517334 z:32.96200180053711
  - atom: CZ x: 7.689000129699707 y:-4.423999786376953 z:33.79100036621094
  - atom: OH x: 6.39900016784668 y:-4.795000076293945 z:34.09600067138672
- residue ASN has 8 atoms
  - atom: N x: 11.267000198364258 y:-3.75 z:29.965999603271484
  - atom: CA x: 10.454999923706055 y:-4.125 z:28.819000244140625
  - atom: C x: 11.135000228881836 y:-5.242000102996826 z:28.03499984741211
  - atom: O x: 10.5 y:-6.247000217437744 z:27.711999893188477
  - atom: CB x: 10.203000006866455 y:-2.9200000762939453 z:27.91200065612793
  - atom: CG x: 9.149000016784668 y:-2.003999948501587 z:28.468000411987305
  - atom: OD1 x: 8.194000244140625 y:-2.4619998931884766 z:29.083999633789062
  - atom: ND2 x: 9.305000305175781 y:-0.7049999833106995 z:28.249000549316406
- residue ALA has 5 atoms
  - atom: N x: 12.42199993133545 y:-5.059999942779541 z:27.74799919128418
  - atom: CA x: 13.201000213623047 y:-6.046000003814697 z:27.020999908447266
  - atom: C x: 13.07800006866455 y:-7.461999893188477 z:27.60099983215332
  - atom: O x: 12.82699966430664 y:-8.409000396728516 z:26.867000579833984
  - atom: CB x: 14.645999908447266 y:-5.619999885559082 z:26.993999481201172
- residue TYR has 12 atoms
  - atom: N x: 13.26200008392334 y:-7.61299991607666 z:28.910999298095703
  - atom: CA x: 13.152000427246094 y:-8.927000045776367 z:29.538000106811523
  - atom: C x: 11.708000183105469 y:-9.369000434875488 z:29.48699951171875
```

```
  - atom: O x: 11.395999908447266 y:-10.520000457763672 z:29.2189998626709
  - atom: CB x: 13.609999656677246 y:-8.871000289916992 z:31.00200080871582
  - atom: CG x: 15.104999542236328 y:-8.961999893188477 z:31.177000045776367
  - atom: CD1 x: 15.8100004196167 y:-10.065999984741211 z:30.708999633789062
  - atom: CD2 x: 15.821000099182129 y:-7.946000099182129 z:31.804000854492188
  - atom: CE1 x: 17.191999435424805 y:-10.159000396728516 z:30.85700035095215
  - atom: CE2 x: 17.207000732421875 y:-8.029999732971191 z:31.95599937438965
  - atom: CZ x: 17.881999969482422 y:-9.13700008392334 z:31.481000900268555
  - atom: OH x: 19.249000549316406 y:-9.23799991607666 z:31.62700080871582
- residue ARG has 11 atoms
  - atom: N x: 10.82800006866455 y:-8.425999641418457 z:29.774999618530273
  - atom: CA x: 9.402000427246094 y:-8.663000106811523 z:29.78499984741211
  - atom: C x: 8.987000465393066 y:-9.234000205993652 z:28.422000885009766
  - atom: O x: 8.211000442504883 y:-10.1850004196167 z:28.336999893188477
  - atom: CB x: 8.715999603271484 y:-7.334000110626221 z:30.06999969482422
  - atom: CG x: 7.258999824523926 y:-7.432000160217285 z:30.378000259399414
  - atom: CD x: 7.006999969482422 y:-7.331999778747559 z:31.860000610351562
  - atom: NE x: 5.626999855041504 y:-6.922999858856201 z:32.09299850463867
  - atom: CZ x: 5.074999809265137 y:-5.839000225067139 z:31.549999237060547
  - atom: NH1 x: 5.791999816894531 y:-5.061999797821045 z:30.73699951171875
  - atom: NH2 x: 3.812000036239624 y:-5.519999980926514 z:31.82900047302246
- residue LYS has 9 atoms
  - atom: N x: 9.53499984741211 y:-8.657999992370605 z:27.354999542236328
  - atom: CA x: 9.25 y:-9.10099983215332 z:25.996000289916992
  - atom: C x: 9.82699966430664 y:-10.49899959564209 z:25.763999938964844
  - atom: O x: 9.098999977111816 y:-11.430000305175781 z:25.424999237060547
  - atom: CB x: 9.835000038146973 y:-8.100000381469727 z:24.999000549316406
  - atom: CG x: 9.510000228881836 y:-8.36299991607666 z:23.541000366210938
  - atom: CD x: 9.970000267028809 y:-7.190000057220459 z:22.68400001525879
  - atom: CE x: 10.038000106811523 y:-7.559000015258789 z:21.20800018310547
  - atom: NZ x: 10.956999778747559 y:-8.722000122070312 z:20.97100067138672
- residue ALA has 5 atoms
  - atom: N x: 11.128999710083008 y:-10.652999877929688 z:25.959999084472656
  - atom: CA x: 11.76200008392334 y:-11.949000358581543 z:25.777999877929688
  - atom: C x: 10.984999656677246 y:-13.067000389099121 z:26.46299934387207
  - atom: O x: 10.824000358581543 y:-14.14799976348877 z:25.902999877929688
  - atom: CB x: 13.173999786376953 y:-11.914999961853027 z:26.304000854492188
- residue ALA has 5 atoms
  - atom: N x: 10.5 y:-12.812000274658203 z:27.672000885009766
  - atom: CA x: 9.753999710083008 y:-13.829999923706055 z:28.402000427246094
  - atom: C x: 8.461000442504883 y:-14.230999946594238 z:27.68199920654297
  - atom: O x: 8.182000160217285 y:-15.414999961853027 z:27.506999969482422
  - atom: CB x: 9.442999839782715 y:-13.34000015258789 z:29.815000534057617
- residue SER has 6 atoms
  - atom: N x: 7.676000118255615 y:-13.246000289916992 z:27.266000747680664
  - atom: CA x: 6.428999900817871 y:-13.508999824523926 z:26.575000762939453
  - atom: C x: 6.638999938964844 y:-14.333000183105469 z:25.29599952697754
  - atom: O x: 5.808000087738037 y:-15.168999671936035 z:24.929000854492188
  - atom: CB x: 5.763000011444092 y:-12.182000160217285 z:26.229999542236328
  - atom: OG x: 4.60099983215332 y:-12.369999885559082 z:25.44099998474121
- residue VAL has 7 atoms
  - atom: N x: 7.751999855041504 y:-14.08899974822998 z:24.615999221801758
  - atom: CA x: 8.069000244140625 y:-14.795999526977539 z:23.388999938964844
  - atom: C x: 8.357000350952148 y:-16.267000198364258 z:23.677000045776367
  - atom: O x: 7.839000225067139 y:-17.152999877929688 z:22.985000610351562
  - atom: CB x: 9.284000396728516 y:-14.144000053405762 z:22.694000244140625
  - atom: CG1 x: 10.064000129699707 y:-15.180999755859375 z:21.92300033569336
  - atom: CG2 x: 8.812999725341797 y:-13.031000137329102 z:21.76099967956543
- residue ILE has 8 atoms
```

```
  - atom: N x: 9.189000129699707 y:-16.523000717163086 z:24.68899917602539
  - atom: CA x: 9.541000366210938 y:-17.88800048828125 z:25.08799934387207
  - atom: C x: 8.321999549865723 y:-18.58099937438965 z:25.70199966430664
  - atom: O x: 8.041000366210938 y:-19.75 z:25.413000106811523
  - atom: CB x: 10.6850004196167 y:-17.89900016784668 z:26.131999969482422
  - atom: CG1 x: 12.006999969482422 y:-17.5049991607666 z:25.482999801635742
  - atom: CG2 x: 10.8100004196167 y:-19.267000198364258 z:26.743000030517578
  - atom: CD1 x: 13.130999565124512 y:-17.316999435424805 z:26.476999282836914
- residue ALA has 5 atoms
  - atom: N x: 7.605999946594238 y:-17.854999542236328 z:26.55299949645996
  - atom: CA x: 6.414000034332275 y:-18.388999938964844 z:27.195999145507812
  - atom: C x: 5.311999797821045 y:-18.68000030517578 z:26.170000076293945
  - atom: O x: 4.185999870300293 y:-19.033000946044922 z:26.5310001373291
  - atom: CB x: 5.906000137329102 y:-17.415000915527344 z:28.23200035095215
- residue LYS has 9 atoms
  - atom: N x: 5.626999855041504 y:-18.524999618530273 z:24.888999938964844
  - atom: CA x: 4.650000095367432 y:-18.802000045776367 z:23.851999282836914
  - atom: C x: 5.250999927520752 y:-19.749000549316406 z:22.81100082397461
  - atom: O x: 4.5879998207092285 y:-20.131999969482422 z:21.847999572753906
  - atom: CB x: 4.186999797821045 y:-17.496999740600586 z:23.204999923706055
  - atom: CG x: 2.924999952316284 y:-17.617000579833984 z:22.378999710083008
  - atom: CD x: 2.450000047683716 y:-16.24799919128418 z:21.92099952697754
  - atom: CE x: 3.4739999771118164 y:-15.571999549865723 z:21.01799964904785
  - atom: NZ x: 3.049999952316284 y:-14.184000015258789 z:20.684999465942383
- residue TYR has 12 atoms
  - atom: N x: 6.513000011444092 y:-20.1200008392334 z:23.009000778198242
  - atom: CA x: 7.193999767303467 y:-21.051000595092773 z:22.108999252319336
  - atom: C x: 6.574999809265137 y:-22.388999938964844 z:22.45599937438965
  - atom: O x: 6.429999828338623 y:-22.719999313354492 z:23.631000518798828
  - atom: CB x: 8.675000190734863 y:-21.06599998474121 z:22.42099952697754
  - atom: CG x: 9.553000450134277 y:-21.915000915527344 z:21.53700065612793
  - atom: CD1 x: 9.6850004196167 y:-21.6560001373291 z:20.166000366210938
  - atom: CD2 x: 10.39799976348877 y:-22.868000030517578 z:22.111000061035156
  - atom: CE1 x: 10.666999816894531 y:-22.320999145507812 z:19.39699935913086
  - atom: CE2 x: 11.368000030517578 y:-23.527999877929688 z:21.361000061035156
  - atom: CZ x: 11.506999969482422 y:-23.253999710083008 z:20.016000747680664
  - atom: OH x: 12.527999877929688 y:-23.892000198364258 z:19.347999572753906
- residue PRO has 7 atoms
  - atom: N x: 6.186999797821045 y:-23.167999267578125 z:21.437999725341797
  - atom: CA x: 5.563000202178955 y:-24.483999252319336 z:21.61400032043457
  - atom: C x: 6.4720001220703125 y:-25.672000885009766 z:21.94700050354004
  - atom: O x: 5.98199987411499 y:-26.764999389648438 z:22.20800018310547
  - atom: CB x: 4.800000190734863 y:-24.670000076293945 z:20.30299949645996
  - atom: CG x: 5.709000110626221 y:-24.02400016784668 z:19.30900001525879
  - atom: CD x: 6.133999824523926 y:-22.7450008392334 z:20.025999069213867
- residue HIS has 10 atoms
  - atom: N x: 7.7820000648498535 y:-25.472000122070312 z:21.944000244140625
  - atom: CA x: 8.687000274658203 y:-26.566999435424805 z:22.27199935913086
  - atom: C x: 9.373000144958496 y:-26.291000366210938 z:23.597000122070312
  - atom: O x: 9.204000473022461 y:-25.219999313354492 z:24.173999786376953
  - atom: CB x: 9.75100040435791 y:-26.72800064086914 z:21.19099998474121
  - atom: CG x: 9.192999839782715 y:-26.843000411987305 z:19.81100082397461
  - atom: ND1 x: 8.234000205993652 y:-27.770999908447266 z:19.472000122070312
  - atom: CD2 x: 9.461999893188477 y:-26.148000717163086 z:18.680999755859375
  - atom: CE1 x: 7.934999942779541 y:-27.64299964904785 z:18.191999435424805
  - atom: NE2 x: 8.666000366210938 y:-26.665000915527344 z:17.687999725341797
- residue LYS has 9 atoms
  - atom: N x: 10.135000228881836 y:-27.26099967956543 z:24.09000015258789
  - atom: CA x: 10.866999626159668 y:-27.077999114990234 z:25.336000442504883
```

```
    - atom: C x: 12.291999816894531 y:-26.701000213623047 z:24.979000091552734
    - atom: O x: 13.027999877929688 y:-27.5 z:24.395999908447266
    - atom: CB x: 10.859999656677246 y:-28.347000122070312 z:26.18000030517578
    - atom: CG x: 12.022000312805176 y:-28.41900062561035 z:27.149999618530273
    - atom: CD x: 11.635000228881836 y:-29.1200008392334 z:28.43199920654297
    - atom: CE x: 10.968999862670898 y:-28.14699935913086 z:29.37700080871582
    - atom: NZ x: 11.902000427246094 y:-27.017000198364258 z:29.71500015258789
  - residue ILE has 8 atoms
    - atom: N x: 12.661999702453613 y:-25.47100067138672 z:25.326000213623047
    - atom: CA x: 13.979999542236328 y:-24.924999237060547 z:25.03700065612793
    - atom: C x: 15.112000465393066 y:-25.763999938964844 z:25.604999542236328
    - atom: O x: 15.272000312805176 y:-25.878999710083008 z:26.81800079345703
    - atom: CB x: 14.07800006866455 y:-23.479999542236328 z:25.554000854492188
    - atom: CG1 x: 13.07699966430664 y:-22.618000030517578 z:24.783000946044922
    - atom: CG2 x: 15.498000144958496 y:-22.948999404907227 z:25.399999618530273
    - atom: CD1 x: 13.050000190734863 y:-21.170000076293945 z:25.19499969482422
  - residue LYS has 9 atoms
    - atom: N x: 15.897000312805176 y:-26.341999053955078 z:24.70400047302246
    - atom: CA x: 17.009000778198242 y:-27.194000244140625 z:25.08300018310547
    - atom: C x: 18.363000869750977 y:-26.479000091552734 z:25.097000122070312
    - atom: O x: 19.3799991607666 y:-27.075000762939453 z:25.45400047302246
    - atom: CB x: 17.05900001525879 y:-28.413999557495117 z:24.1560001373291
    - atom: CG x: 15.869000434875488 y:-29.34600067138672 z:24.302000045776367
    - atom: CD x: 16.128000259399414 y:-30.667999267578125 z:23.60300064086914
    - atom: CE x: 15.015000343322754 y:-31.684999465942383 z:23.892000198364258
    - atom: NZ x: 15.324999809265137 y:-33.077999114990234 z:23.42300033569336
  - residue SER has 6 atoms
    - atom: N x: 18.378000259399414 y:-25.202999114990234 z:24.72100067138672
    - atom: CA x: 19.618999481201172 y:-24.429000854492188 z:24.722000122070312
    - atom: C x: 19.31399917602539 y:-22.95199966430664 z:24.527000427246094
    - atom: O x: 18.201000213623047 y:-22.58300018310547 z:24.15399932861328
    - atom: CB x: 20.55699920654297 y:-24.892000198364258 z:23.604000091552734
    - atom: OG x: 20.27899932861328 y:-24.198999404907227 z:22.398000717163086
  - residue GLY has 4 atoms
    - atom: N x: 20.309999465942383 y:-22.110000610351562 z:24.777999877929688
    - atom: CA x: 20.1200008392334 y:-20.68199920654297 z:24.617000579833984
    - atom: C x: 20.091999053955078 y:-20.316999435424805 z:23.14699935913086
    - atom: O x: 19.503999710083008 y:-19.30699920654297 z:22.756000518798828
  - residue ALA has 5 atoms
    - atom: N x: 20.73200035095215 y:-21.145999908447266 z:22.327999114990234
    - atom: CA x: 20.773000717163086 y:-20.899999618530273 z:20.902000427246094
    - atom: C x: 19.391000747680664 y:-21.14299964904785 z:20.302000045776367
    - atom: O x: 18.961999893188477 y:-20.430999755859375 z:19.392000198364258
    - atom: CB x: 21.799999237060547 y:-21.798999786376953 z:20.253000259399414
  - residue GLU has 9 atoms
    - atom: N x: 18.69300079345703 y:-22.150999069213867 z:20.809999465942383
    - atom: CA x: 17.35700035095215 y:-22.451000213623047 z:20.31999969482422
    - atom: C x: 16.514999389648438 y:-21.20599937438965 z:20.593000411987305
    - atom: O x: 15.798999786376953 y:-20.719999313354492 z:19.715999603271484
    - atom: CB x: 16.768999099731445 y:-23.655000686645508 z:21.06100082397461
    - atom: CG x: 15.477000023651123 y:-24.17300033569336 z:20.458999633789062
    - atom: CD x: 14.970999717712402 y:-25.433000564575195 z:21.138999938964844
    - atom: OE1 x: 15.795999526977539 y:-26.31999969482422 z:21.447000050354004
    - atom: OE2 x: 13.744999885559082 y:-25.54800033569336 z:21.347999572753906
  - residue ALA has 5 atoms
    - atom: N x: 16.632999420166016 y:-20.677000045776367 z:21.808000564575195
    - atom: CA x: 15.897000312805176 y:-19.48900032043457 z:22.211000442504883
    - atom: C x: 16.354000091552734 y:-18.23699951171875 z:21.452999114990234
    - atom: O x: 15.574999809265137 y:-17.315000534057617 z:21.225000381469727
```

```
  - atom: CB x: 16.05699920654297 y:-19.281999588012695 z:23.701000213623047
- residue LYS has 9 atoms
  - atom: N x: 17.624000549316406 y:-18.20800018310547 z:21.069000244140625
  - atom: CA x: 18.18400001525879 y:-17.07699966430664 z:20.341999053955078
  - atom: C x: 17.55699920654297 y:-16.8799991607666 z:18.937999725341797
  - atom: O x: 17.700000762939453 y:-15.824000358581543 z:18.320999145507812
  - atom: CB x: 19.695999145507812 y:-17.263999938964844 z:20.235000610351562
  - atom: CG x: 20.41699981689453 y:-16.229000091552734 z:19.392000198364258
  - atom: CD x: 20.399999618530273 y:-14.847999572753906 z:20.023000717163086
  - atom: CE x: 21.18199920654297 y:-13.862000465393066 z:19.165000915527344
  - atom: NZ x: 22.56599998474121 y:-14.347000122070312 z:18.958999633789062
- residue LYS has 9 atoms
  - atom: N x: 16.861000061035156 y:-17.892000198364258 z:18.434999465942383
  - atom: CA x: 16.226999282836914 y:-17.773000717163086 z:17.131000518798828
  - atom: C x: 15.065999984741211 y:-16.77199935913086 z:17.200000762939453
  - atom: O x: 14.831000328063965 y:-16.006999969482422 z:16.256999969482422
  - atom: CB x: 15.673999786376953 y:-19.125 z:16.680999755859375
  - atom: CG x: 16.683000564575195 y:-20.246999740600586 z:16.594999313354492
  - atom: CD x: 15.987000465393066 y:-21.54400062561035 z:16.172000885009766
  - atom: CE x: 16.908000946044922 y:-22.753999710083008 z:16.27899932861328
  - atom: NZ x: 16.2549991607666 y:-24.025999069213867 z:15.857000350952148
- residue LEU has 8 atoms
  - atom: N x: 14.343999862670898 y:-16.790000915527344 z:18.32200050354004
  - atom: CA x: 13.184000015258789 y:-15.925999641418457 z:18.53700065612793
  - atom: C x: 13.527999877929688 y:-14.432999610900879 z:18.64699935913086
  - atom: O x: 14.597000122070312 y:-14.064000129699707 z:19.136999130249023
  - atom: CB x: 12.437000274658203 y:-16.3799991607666 z:19.795000076293945
  - atom: CG x: 12.102999687194824 y:-17.868000030517578 z:19.90399932861328
  - atom: CD1 x: 11.484999656677246 y:-18.17799949645996 z:21.260000228881836
  - atom: CD2 x: 11.161999702453613 y:-18.250999450683594 z:18.78499984741211
- residue PRO has 7 atoms
  - atom: N x: 12.616000175476074 y:-13.555999755859375 z:18.187000274658203
  - atom: CA x: 12.782999992370605 y:-12.100000381469727 z:18.21500015258789
  - atom: C x: 12.5 y:-11.564000129699707 z:19.618000030517578
  - atom: O x: 11.369999885559082 y:-11.609999656677246 z:20.106000900268555
  - atom: CB x: 11.753000259399414 y:-11.631999969482422 z:17.20800018310547
  - atom: CG x: 10.605999946594238 y:-12.545000076293945 z:17.520000457763672
  - atom: CD x: 11.295999526977539 y:-13.909000396728516 z:17.631999969482422
- residue GLY has 4 atoms
  - atom: N x: 13.532999992370605 y:-11.04699993133545 z:20.26300048828125
  - atom: CA x: 13.359999656677246 y:-10.550000190734863 z:21.61199951171875
  - atom: C x: 14.352999687194824 y:-11.239999771118164 z:22.52199935913086
  - atom: O x: 14.593999862670898 y:-10.791000366210938 z:23.634000778198242
- residue VAL has 7 atoms
  - atom: N x: 14.920000076293945 y:-12.342000007629395 z:22.047000885009766
  - atom: CA x: 15.909000396728516 y:-13.072999954223633 z:22.801000595092773
  - atom: C x: 17.238000869750977 y:-12.873000144958496 z:22.11199951171875
  - atom: O x: 17.503000259399414 y:-13.482999801635742 z:21.072999954223633
  - atom: CB x: 15.623000144958496 y:-14.553999900817871 z:22.81399917602539
  - atom: CG1 x: 16.7549991607666 y:-15.277000427246094 z:23.52199935913086
  - atom: CG2 x: 14.295000076293945 y:-14.815999984741211 z:23.481000900268555
- residue GLY has 4 atoms
  - atom: N x: 18.07200050354004 y:-12.015999794006348 z:22.69700050354004
  - atom: CA x: 19.378999710083008 y:-11.729999542236328 z:22.134000778198242
  - atom: C x: 20.488000869750977 y:-12.513999938964844 z:22.802000045776367
  - atom: O x: 20.222000122070312 y:-13.487000465393066 z:23.506000518798828
- residue THR has 7 atoms
  - atom: N x: 21.72800064086914 y:-12.062999725341797 z:22.60300064086914
  - atom: CA x: 22.920000076293945 y:-12.718999862670898 z:23.145999908447266
```

```
  - atom: C x: 22.94099998474121 y:-12.92199993133545 z:24.6560001373291
  - atom: O x: 23.170000076293945 y:-14.029999732971191 z:25.128000259399414
  - atom: CB x: 24.21299934387207 y:-11.954999923706055 z:22.770000457763672
  - atom: OG1 x: 24.249000549316406 y:-11.710000038146973 z:21.356000900268555
  - atom: CG2 x: 25.434999465942383 y:-12.774999618530273 z:23.156999588012695
- residue LYS has 9 atoms
  - atom: N x: 22.709999084472656 y:-11.857999801635742 z:25.413000106811523
  - atom: CA x: 22.726999282836914 y:-11.961000442504883 z:26.865999221801758
  - atom: C x: 21.701000213623047 y:-12.954000473022461 z:27.388999938964844
  - atom: O x: 22.0 y:-13.739999771118164 z:28.275999069213867
  - atom: CB x: 22.475000381469727 y:-10.595999717712402 z:27.5
  - atom: CG x: 23.288000106811523 y:-9.480999946594238 z:26.8700008392334
  - atom: CD x: 24.777000427246094 y:-9.682999610900879 z:27.093000411987305
  - atom: CE x: 25.266000747680664 y:-8.854999542236328 z:28.26300048828125
  - atom: NZ x: 25.291000366210938 y:-7.4070000648498535 z:27.913999557495117
- residue ILE has 8 atoms
  - atom: N x: 20.489999771118164 y:-12.942000389099121 z:26.85099983215332
  - atom: CA x: 19.493000030517578 y:-13.871000289916992 z:27.351999282836914
  - atom: C x: 19.766000747680664 y:-15.331000328063965 z:26.969999313354492
  - atom: O x: 19.400999069213867 y:-16.253000259399414 z:27.704999923706055
  - atom: CB x: 18.073999404907227 y:-13.446999549865723 z:26.920000076293945
  - atom: CG1 x: 17.691999435424805 y:-12.152999877929688 z:27.6560001373291
  - atom: CG2 x: 17.062999725341797 y:-14.581000328063965 z:27.211999893188477
  - atom: CD1 x: 16.2549991607666 y:-11.734999656677246 z:27.492000579833984
- residue ALA has 5 atoms
  - atom: N x: 20.427000045776367 y:-15.54800033569336 z:25.836999893188477
  - atom: CA x: 20.742000579833984 y:-16.9060001373291 z:25.4060001373291
  - atom: C x: 21.82200050354004 y:-17.413000106811523 z:26.351999282836914
  - atom: O x: 21.839000701904297 y:-18.582000732421875 z:26.763999938964844
  - atom: CB x: 21.256999969482422 y:-16.88800048828125 z:23.981000900268555
- residue GLU has 9 atoms
  - atom: N x: 22.719999313354492 y:-16.493000030517578 z:26.68899917602539
  - atom: CA x: 23.83099937438965 y:-16.760000228881836 z:27.58300018310547
  - atom: C x: 23.28700065612793 y:-17.075000762939453 z:28.95800018310547
  - atom: O x: 23.738000869750977 y:-18.0049991607666 z:29.625999450683594
  - atom: CB x: 24.726999282836914 y:-15.53600025177002 z:27.64900016784668
  - atom: CG x: 25.850000381469727 y:-15.63599967956543 z:28.64299964904785
  - atom: CD x: 26.965999603271484 y:-14.670000076293945 z:28.30900001525879
  - atom: OE1 x: 27.51799964904785 y:-14.795999526977539 z:27.19700050354004
  - atom: OE2 x: 27.283000946044922 y:-13.788999557495117 z:29.141000747680664
- residue LYS has 9 atoms
  - atom: N x: 22.308000564575195 y:-16.295000076293945 z:29.381999969482422
  - atom: CA x: 21.725000381469727 y:-16.541000366210938 z:30.672000885009766
  - atom: C x: 20.95199966430664 y:-17.840999603271484 z:30.590999603271484
  - atom: O x: 20.99799919128418 y:-18.648000717163086 z:31.51300048828125
  - atom: CB x: 20.82699966430664 y:-15.378999710083008 z:31.089000701904297
  - atom: CG x: 21.610000610351562 y:-14.130999565124512 z:31.437000274658203
  - atom: CD x: 20.827999114990234 y:-13.210000038146973 z:32.361000061035156
  - atom: CE x: 21.672000885009766 y:-12.029999732971191 z:32.82699966430664
  - atom: NZ x: 22.187999725341797 y:-11.217000007629395 z:31.687999725341797
- residue ILE has 8 atoms
  - atom: N x: 20.256000518798828 y:-18.055999755859375 z:29.479000091552734
  - atom: CA x: 19.496000289916992 y:-19.285999298095703 z:29.30500030517578
  - atom: C x: 20.443000793457031 y:-20.496999740600586 z:29.367000579833984
  - atom: O x: 20.117000579833984 y:-21.52899932861328 z:29.961999893188477
  - atom: CB x: 18.714000701904297 y:-19.270000457763672 z:27.961999893188477
  - atom: CG1 x: 17.535999298095703 y:-18.297000885009766 z:28.06399917602539
  - atom: CG2 x: 18.194000244140625 y:-20.665000915527344 z:27.6299991607666
  - atom: CD1 x: 16.844999313354492 y:-18.011999130249023 z:26.757999420166016
```

```
- residue ASP has 8 atoms
  - atom: N x: 21.618000030517578 y:-20.37299919128418 z:28.761999130249023
  - atom: CA x: 22.56999969482422 y:-21.466999053955078 z:28.795000076293945
  - atom: C x: 22.993999481201172 y:-21.802000045776367 z:30.23200035095215
  - atom: O x: 23.159000396728516 y:-22.97800064086914 z:30.573999404907227
  - atom: CB x: 23.785999298095703 y:-21.145999908447266 z:27.93199920654297
  - atom: CG x: 23.558000564575195 y:-21.47599983215332 z:26.459999084472656
  - atom: OD1 x: 22.81999969482422 y:-22.45199966430664 z:26.187999725341797
  - atom: OD2 x: 24.12299919128418 y:-20.783000946044922 z:25.57900047302246
- residue GLU has 9 atoms
  - atom: N x: 23.165000915527344 y:-20.785999298095703 z:31.07900047302246
  - atom: CA x: 23.533000946044922 y:-21.05699920654297 z:32.4640007019043
  - atom: C x: 22.351999282836914 y:-21.722000122070312 z:33.15700149536133
  - atom: O x: 22.476999282836914 y:-22.834999084472656 z:33.65800094604492
  - atom: CB x: 23.89900016784668 y:-19.783000946044922 z:33.233001708984375
  - atom: CG x: 24.15999984741211 y:-20.05900001525879 z:34.72200012207031
  - atom: CD x: 24.485000610351562 y:-18.812999725341797 z:35.542999267578125
  - atom: OE1 x: 25.302000045776367 y:-17.989999771118164 z:35.09000015258789
  - atom: OE2 x: 23.93899917602539 y:-18.665000915527344 z:36.654998779296875
- residue PHE has 11 atoms
  - atom: N x: 21.201000213623047 y:-21.04800033569336 z:33.17599868774414
  - atom: CA x: 20.01799964904785 y:-21.60300064086914 z:33.83700180053711
  - atom: C x: 19.79599952697754 y:-23.05900001525879 z:33.45500183105469
  - atom: O x: 19.496999740600586 y:-23.89900016784668 z:34.303001403808594
  - atom: CB x: 18.76099967956543 y:-20.81599998474121 z:33.494998931884766
  - atom: CG x: 17.554000854492188 y:-21.259000778198242 z:34.27399826049805
  - atom: CD1 x: 17.402999877929688 y:-20.89900016784668 z:35.61000061035156
  - atom: CD2 x: 16.577999114990234 y:-22.05699920654297 z:33.68299865722656
  - atom: CE1 x: 16.29599952697754 y:-21.326000213623047 z:36.34700012207031
  - atom: CE2 x: 15.472000122070312 y:-22.486000061035156 z:34.4119987487793
  - atom: CZ x: 15.331999778747559 y:-22.1200008392334 z:35.74599838256836
- residue LEU has 8 atoms
  - atom: N x: 19.929000854492188 y:-23.354999542236328 z:32.16999816894531
  - atom: CA x: 19.76099967956543 y:-24.719999313354492 z:31.724000930786133
  - atom: C x: 20.87299919128418 y:-25.54400062561035 z:32.38100051879883
  - atom: O x: 20.597000122070312 y:-26.49799919128418 z:33.11199951171875
  - atom: CB x: 19.84000015258789 y:-24.78700065612793 z:30.19499969482422
  - atom: CG x: 18.53700065612793 y:-25.14900016784668 z:29.46299934387207
  - atom: CD1 x: 17.368999481201172 y:-24.40399932861328 z:30.06100082397461
  - atom: CD2 x: 18.67099952697754 y:-24.833999633789062 z:27.979999542236328
- residue ALA has 5 atoms
  - atom: N x: 22.125 y:-25.1560001373291 z:32.15299987792969
  - atom: CA x: 23.2549991607666 y:-25.881000518798828 z:32.73099899291992
  - atom: C x: 23.19700050354004 y:-25.913000106811523 z:34.263999938964844
  - atom: O x: 22.674999237060547 y:-26.856000900268555 z:34.8650016784668
  - atom: CB x: 24.572999954223633 y:-25.256999969482422 z:32.26499938964844
- residue THR has 7 atoms
  - atom: N x: 23.732999801635742 y:-24.874000549316406 z:34.891998291015625
  - atom: CA x: 23.746999740600586 y:-24.791000366210938 z:36.345001220703125
  - atom: C x: 22.393999099731445 y:-25.16200065612793 z:36.95800018310547
  - atom: O x: 22.26799964904785 y:-26.194000244140625 z:37.60900115966797
  - atom: CB x: 24.17799949645996 y:-23.371000289916992 z:36.810001373291016
  - atom: OG1 x: 23.086000442504883 y:-22.448999404907227 z:36.68899917602539
  - atom: CG2 x: 25.349000930786133 y:-22.87700080871582 z:35.946998596191406
- residue GLY has 4 atoms
  - atom: N x: 21.3799991607666 y:-24.339000701904297 z:36.72800064086914
  - atom: CA x: 20.069000244140625 y:-24.611000061035156 z:37.29199981689453
  - atom: C x: 19.61400032043457 y:-23.41699981689453 z:38.10900115966797
  - atom: O x: 18.61400032043457 y:-23.454999923706055 z:38.83599853515625
```

```
- residue LYS has 9 atoms
  - atom: N x: 20.375999450683594 y:-22.34000015258789 z:37.97700119018555
  - atom: CA x: 20.104999542236328 y:-21.104000091552734 z:38.680999755859375
  - atom: C x: 20.933000564575195 y:-20.047000885009766 z:37.987998962402344
  - atom: O x: 21.757999420166016 y:-20.365999221801758 z:37.137001037597656
  - atom: CB x: 20.530000686645508 y:-21.229999542236328 z:40.150001525878906
  - atom: CG x: 22.016000747680664 y:-21.55299949645996 z:40.37799835205078
  - atom: CD x: 22.30500030517578 y:-21.792999267578125 z:41.87099838256836
  - atom: CE x: 23.71500015258789 y:-22.350000381469727 z:42.15399932861328
  - atom: NZ x: 24.823999404907227 y:-21.365999221801758 z:41.97200012207031
- residue LEU has 8 atoms
  - atom: N x: 20.711000442504883 y:-18.78700065612793 z:38.33000183105469
  - atom: CA x: 21.5 y:-17.73200035095215 z:37.724998474121094
  - atom: C x: 22.31999969482422 y:-17.052000045776367 z:38.80699920654297
  - atom: O x: 21.7810001373291 y:-16.632999420166016 z:39.83000183105469
  - atom: CB x: 20.608999252319336 y:-16.709999084472656 z:37.027000427246094
  - atom: CG x: 21.32699966430664 y:-15.911999702453613 z:35.935001373291016
  - atom: CD1 x: 21.85099983215332 y:-16.854000091552734 z:34.85599899291992
  - atom: CD2 x: 20.36400032043457 y:-14.904999732971191 z:35.32699966430664
- residue ARG has 11 atoms
  - atom: N x: 23.6299991607666 y:-16.968000411987305 z:38.57899856567383
  - atom: CA x: 24.55500030517578 y:-16.33300018310547 z:39.516998291015625
  - atom: C x: 24.148000717163086 y:-14.880000114440918 z:39.749000549316406
  - atom: O x: 24.016000747680664 y:-14.442000389099121 z:40.891998291015625
  - atom: CB x: 25.979000091552734 y:-16.405000686645508 z:38.9640007019043
  - atom: CG x: 26.475000381469727 y:-17.833999633789062 z:38.73099899291992
  - atom: CD x: 27.805999755859375 y:-17.840999603271484 z:37.98899841308594
  - atom: NE x: 27.750999450683594 y:-16.979000091552734 z:36.808998107910156
  - atom: CZ x: 28.722999572753906 y:-16.86400032043457 z:35.904998779296875
  - atom: NH1 x: 29.840999603271484 y:-17.56399917602539 z:36.03200149536133
  - atom: NH2 x: 28.58300018310547 y:-16.033000946044922 z:34.875999450683594
- residue LYS has 9 atoms
  - atom: N x: 23.94099998474121 y:-14.14799976348877 z:38.65700149536133
  - atom: CA x: 23.523000717163086 y:-12.746999740600586 z:38.71799850463867
  - atom: C x: 22.263999938964844 y:-12.520000457763672 z:39.571998596191406
  - atom: O x: 22.094999313354492 y:-11.446999549865723 z:40.1609992980957
  - atom: CB x: 23.275999069213867 y:-12.211000442504883 z:37.30699920654297
  - atom: CG x: 22.777999877929688 y:-10.774999618530273 z:37.26499938964844
  - atom: CD x: 22.426000595092773 y:-10.364999771118164 z:35.8489990234375
  - atom: CE x: 21.812000274658203 y:-8.97700023651123 z:35.79999923706055
  - atom: NZ x: 22.783000946044922 y:-7.9079999923706055 z:36.16999816894531
- residue LEU has 8 atoms
  - atom: N x: 21.37700080871582 y:-13.51099967956543 z:39.630001068115234
  - atom: CA x: 20.15399932861328 y:-13.38599967956543 z:40.43600082397461
  - atom: C x: 20.409000396728516 y:-13.715999603271484 z:41.909000396728516
  - atom: O x: 19.76099967956543 y:-13.151000022888184 z:42.79199981689453
  - atom: CB x: 19.0310001373291 y:-14.29800033569336 z:39.91299819946289
  - atom: CG x: 18.204999923706055 y:-13.871000289916992 z:38.69599914550781
  - atom: CD1 x: 17.20599937438965 y:-14.956000328063965 z:38.374000549316406
  - atom: CD2 x: 17.489999771118164 y:-12.565999984741211 z:38.97100067138672
- residue GLU has 9 atoms
  - atom: N x: 21.33099937438965 y:-14.642000198364258 z:42.17399978637695
  - atom: CA x: 21.6560001373291 y:-14.987000465393066 z:43.553001403808594
  - atom: C x: 22.179000854492188 y:-13.70199966430664 z:44.17900085449219
  - atom: O x: 21.764999389648438 y:-13.3149995803833 z:45.27199935913086
  - atom: CB x: 22.73900032043457 y:-16.07699966430664 z:43.624000549316406
  - atom: CG x: 22.267000198364258 y:-17.479999542236328 z:43.24599838256836
  - atom: CD x: 21.235000610351562 y:-18.04599952697754 z:44.209999084472656
  - atom: OE1 x: 20.523000717163086 y:-18.98900032043457 z:43.81399917602539
```

```
  - atom: OE2 x: 21.13599967956543 y:-17.562999725341797 z:45.36000061035156
- residue LYS has 9 atoms
  - atom: N x: 23.073999404907227 y:-13.038000106811523 z:43.45399856567383
  - atom: CA x: 23.663999557495117 y:-11.78600025177002 z:43.90700149536133
  - atom: C x: 22.57200050354004 y:-10.781999588012695 z:44.24700164794922
  - atom: O x: 22.555999755859375 y:-10.211999893188477 z:45.334999084472656
  - atom: CB x: 24.594999313354492 y:-11.241999626159668 z:42.821998596191406
  - atom: CG x: 25.749000549316406 y:-12.201000213623047 z:42.54100036621094
  - atom: CD x: 26.69700050354004 y:-11.706000328063965 z:41.46500015258789
  - atom: CE x: 27.738000869750977 y:-12.774999618530273 z:41.16299819946289
  - atom: NZ x: 28.597999572753906 y:-12.456000328063965 z:39.9900016784668
- residue ILE has 8 atoms
  - atom: N x: 21.650999069213867 y:-10.579999923706055 z:43.3129997253418
  - atom: CA x: 20.538999557495117 y:-9.666999816894531 z:43.53300094604492
  - atom: C x: 19.79400062561035 y:-10.114999771118164 z:44.78499984741211
  - atom: O x: 19.698999404907227 y:-9.368000030517578 z:45.75299835205078
  - atom: CB x: 19.576000213623047 y:-9.663000106811523 z:42.31800079345703
  - atom: CG1 x: 20.308000564575195 y:-9.105999946594238 z:41.09299850463867
  - atom: CG2 x: 18.336999893188477 y:-8.845999717712402 z:42.625
  - atom: CD1 x: 19.481000900268555 y:-9.064000129699707 z:39.84400177001953
- residue ARG has 11 atoms
  - atom: N x: 19.283000946044922 y:-11.342000007629395 z:44.7599983215332
  - atom: CA x: 18.551000595092773 y:-11.920999526977539 z:45.88999938964844
  - atom: C x: 19.215999603271484 y:-11.687000274658203 z:47.249000549316406
  - atom: O x: 18.5310001373291 y:-11.527000427246094 z:48.262001037597656
  - atom: CB x: 18.378999710083008 y:-13.430000305175781 z:45.68199920654297
  - atom: CG x: 17.430999755859375 y:-13.803000450134277 z:44.56800079345703
  - atom: CD x: 17.413000106811523 y:-15.305000305175781 z:44.323001861572266
  - atom: NE x: 16.273000717163086 y:-15.689000129699707 z:43.492000579833984
  - atom: CZ x: 15.977999687194824 y:-16.937000274658203 z:43.132999420166016
  - atom: NH1 x: 16.740999221801758 y:-17.950000762939453 z:43.525001525878906
  - atom: NH2 x: 14.906000137329102 y:-17.176000595092773 z:42.38800048828125
- residue GLN has 9 atoms
  - atom: N x: 20.548999786376953 y:-11.678999900817871 z:47.263999938964844
  - atom: CA x: 21.320999145507812 y:-11.486000061035156 z:48.492000579833984
  - atom: C x: 21.540000915527344 y:-10.038999557495117 z:48.88999938964844
  - atom: O x: 21.20199966430664 y:-9.645999908447266 z:50.005001068115234
  - atom: CB x: 22.667999267578125 y:-12.187999725341797 z:48.374000549316406
  - atom: CG x: 22.65399932861328 y:-13.574000358581543 z:48.97700119018555
  - atom: CD x: 23.506999969482422 y:-14.543000221252441 z:48.20399856567383
  - atom: OE1 x: 24.591999053955078 y:-14.189000129699707 z:47.731998443603516
  - atom: NE2 x: 23.027000427246094 y:-15.782999992370605 z:48.07099914550781
- residue ASP has 8 atoms
  - atom: N x: 22.11400032043457 y:-9.253000259399414 z:47.983001708984375
  - atom: CA x: 22.3700008392334 y:-7.834000110626221 z:48.22999954223633
  - atom: C x: 21.216999053955078 y:-7.169000148773193 z:48.99300003051758
  - atom: O x: 20.115999221801758 y:-7.050000190734863 z:48.47700119018555
  - atom: CB x: 22.594999313354492 y:-7.125999927520752 z:46.89099884033203
  - atom: CG x: 22.950000762939453 y:-5.663000106811523 z:47.05400085449219
  - atom: OD1 x: 23.440000534057617 y:-5.063000202178955 z:46.07400131225586
  - atom: OD2 x: 22.735000610351562 y:-5.116000175476074 z:48.15800094604492
- residue ASP has 8 atoms
  - atom: N x: 21.47599983215332 y:-6.7230000495910645 z:50.21699905395508
  - atom: CA x: 20.43600082397461 y:-6.09499979019165 z:51.03499984741211
  - atom: C x: 19.680999755859375 y:-4.929999828338623 z:50.38100051879883
  - atom: O x: 18.44499969482422 y:-4.921999931335449 z:50.34199905395508
  - atom: CB x: 21.023000717163086 y:-5.610000133514404 z:52.36600112915039
  - atom: CG x: 20.0 y:-4.864999771118164 z:53.21699905395508
  - atom: OD1 x: 19.062999725341797 y:-5.508999824523926 z:53.75
```

```
  - atom: OD2 x: 20.125 y:-3.630000114440918 z:53.34400177001953
- residue THR has 7 atoms
  - atom: N x: 20.41900062561035 y:-3.941999912261963 z:49.88199996948242
  - atom: CA x: 19.788000106811523 y:-2.7799999713897705 z:49.266998291015625
  - atom: C x: 18.878999710083008 y:-3.131999969482422 z:48.10300064086914
  - atom: O x: 17.701000213623047 y:-2.7730000019073486 z:48.104000091552734
  - atom: CB x: 20.826000213623047 y:-1.7740000486373901 z:48.766998291015625
  - atom: OG1 x: 21.684999465942383 y:-1.3969999551773071 z:49.84700012207031
  - atom: CG2 x: 20.131000518798828 y:-0.531000018119812 z:48.229000091552734
- residue SER has 6 atoms
  - atom: N x: 19.43400001525879 y:-3.8239998817443848 z:47.11199951171875
  - atom: CA x: 18.676000595092773 y:-4.2210001945495605 z:45.930999755859375
  - atom: C x: 17.445999145507812 y:-4.991000175476074 z:46.375999450683594
  - atom: O x: 16.354000091552734 y:-4.8429999351501465 z:45.827999114990234
  - atom: CB x: 19.527999877929688 y:-5.125999927520752 z:45.04399871826172
  - atom: OG x: 20.857999801635742 y:-4.6479997634887695 z:44.95500183105469
- residue SER has 6 atoms
  - atom: N x: 17.650999069213867 y:-5.809000015258789 z:47.39699935913086
  - atom: CA x: 16.614999771118164 y:-6.6579999923706055 z:47.946998596191406
  - atom: C x: 15.543000221252441 y:-5.864999771118164 z:48.672000885009766
  - atom: O x: 14.366999626159668 y:-6.239999771118164 z:48.676998138427734
  - atom: CB x: 17.256999969482422 y:-7.659999847412109 z:48.895999908447266
  - atom: OG x: 16.29199981689453 y:-8.527000427246094 z:49.439998626708984
- residue SER has 6 atoms
  - atom: N x: 15.956999778747559 y:-4.765999794006348 z:49.28799819946289
  - atom: CA x: 15.032999992370605 y:-3.927000045776367 z:50.02299880981445
  - atom: C x: 14.21399974822998 y:-3.0350000858306885 z:49.106998443603516
  - atom: O x: 13.001999855041504 y:-2.947000026702881 z:49.257999420166016
  - atom: CB x: 15.791999816894531 y:-3.0929999351501465 z:51.06100082397461
  - atom: OG x: 16.2549991607666 y:-3.9210000038146973 z:52.12300109863281
- residue ILE has 8 atoms
  - atom: N x: 14.859999656677246 y:-2.371999979019165 z:48.159000396728516
  - atom: CA x: 14.121000289916992 y:-1.5190000534057617 z:47.24399948120117
  - atom: C x: 13.07699966430664 y:-2.390000104904175 z:46.57600021362305
  - atom: O x: 11.907999992370605 y:-2.01200008392334 z:46.41999816894531
  - atom: CB x: 15.031999588012695 y:-0.9359999895095825 z:46.16899871826172
  - atom: CG1 x: 16.11400032043457 y:-0.07800000160932541 z:46.83300018310547
  - atom: CG2 x: 14.199000358581543 y:-0.11400000005960464 z:45.18199920654297
  - atom: CD1 x: 17.23699951171875 y:0.3479999899864197 z:45.91999816894531
- residue ASN has 8 atoms
  - atom: N x: 13.520000457763672 y:-3.5810000896453857 z:46.20199966430664
  - atom: CA x: 12.65999984741211 y:-4.548999786376953 z:45.5620002746582
  - atom: C x: 11.432000160217285 y:-4.789000034332275 z:46.38999938964844
  - atom: O x: 10.333999633789062 y:-4.831999778747559 z:45.874000549316406
  - atom: CB x: 13.383999824523926 y:-5.864999771118164 z:45.374000549316406
  - atom: CG x: 13.378999710083008 y:-6.310999870300293 z:43.948001861572266
  - atom: OD1 x: 12.390000343322754 y:-6.114999771118164 z:43.23500061035156
  - atom: ND2 x: 14.47599983215332 y:-6.923999786376953 z:43.51100158691406
- residue PHE has 11 atoms
  - atom: N x: 11.60099983215332 y:-4.936999797821045 z:47.6879997253418
  - atom: CA x: 10.439000129699707 y:-5.195000171661377 z:48.50699996948242
  - atom: C x: 9.46399974822998 y:-4.026000022888184 z:48.55799865722656
  - atom: O x: 8.288000106811523 y:-4.184000015258789 z:48.26900100708008
  - atom: CB x: 10.843999862670898 y:-5.577000141143799 z:49.92599868774414
  - atom: CG x: 9.680000305175781 y:-5.688000202178955 z:50.8489990234375
  - atom: CD1 x: 8.789999961853027 y:-6.736999988555908 z:50.729000091552734
  - atom: CD2 x: 9.430999755859375 y:-4.703000068664551 z:51.792999267578125
  - atom: CE1 x: 7.666999816894531 y:-6.803999900817871 z:51.534000396728516
  - atom: CE2 x: 8.303000450134277 y:-4.763999938964844 z:52.60100173950195
```

```
  - atom: CZ x: 7.425000190734863 y:-5.815000057220459 z:52.47100067138672
- residue LEU has 8 atoms
  - atom: N x: 9.954999923706055 y:-2.8559999465942383 z:48.935001373291016
  - atom: CA x: 9.11299991607666 y:-1.6629999876022339 z:49.029998779296875
  - atom: C x: 8.187000274658203 y:-1.4440000057220046 z:47.8489990234375
  - atom: O x: 6.998000144958496 y:-1.2089999914169312 z:48.0359992980957
  - atom: CB x: 9.97599983215332 y:-0.42399999499320984 z:49.19599914550781
  - atom: CG x: 10.741999626159668 y:-0.367000013589859 z:50.50699996948242
  - atom: CD1 x: 11.949000358581543 y:0.5139999985694885 z:50.32600021362305
  - atom: CD2 x: 9.82699966430664 y:0.1289999932050705 z:51.612998962402344
- residue THR has 7 atoms
  - atom: N x: 8.727999687194824 y:-1.5180000066757202 z:46.63600158691406
  - atom: CA x: 7.913000106811523 y:-1.3020000457763672 z:45.45000076293945
  - atom: C x: 6.660999774932861 y:-2.1670000553131104 z:45.41999816894531
  - atom: O x: 5.74399995803833 y:-1.86899995803833 z:44.67499923706055
  - atom: CB x: 8.699999809265137 y:-1.5509999990463257 z:44.16999816894531
  - atom: OG1 x: 8.991000175476074 y:-2.943000078201294 z:44.060001373291016
  - atom: CG2 x: 9.998000144958496 y:-0.7699999809265137 z:44.17599868774414
- residue ARG has 11 atoms
  - atom: N x: 6.604000091552734 y:-3.2290000915527344 z:46.2239990234375
  - atom: CA x: 5.4070000648498535 y:-4.083000183105469 z:46.25
  - atom: C x: 4.270999908447266 y:-3.384000062942505 z:46.97600173950195
  - atom: O x: 3.1610000133514404 y:-3.9130001068115234 z:47.06999969482422
  - atom: CB x: 5.684999942779541 y:-5.426000118255615 z:46.926998138427734
  - atom: CG x: 6.485000133514404 y:-6.410999774932861 z:46.06700134277344
  - atom: CD x: 7.070000171661377 y:-7.552000045776367 z:46.89799880981445
  - atom: NE x: 6.085000038146973 y:-8.112000465393066 z:47.81800079345703
  - atom: CZ x: 6.336999893188477 y:-9.085000038146973 z:48.68600082397461
  - atom: NH1 x: 7.546000003814697 y:-9.621999740600586 z:48.762001037597656
  - atom: NH2 x: 5.382999897003174 y:-9.510000228881836 z:49.492000579833984
- residue VAL has 7 atoms
  - atom: N x: 4.554999828338623 y:-2.2019999027252197 z:47.51300048828125
  - atom: CA x: 3.5380001068115234 y:-1.4140000343322754 z:48.19200134277344
  - atom: C x: 3.002000093460083 y:-0.4729999899864197 z:47.124000549316406
  - atom: O x: 3.7679998874664307 y:0.1459999978542328 z:46.38100051879883
  - atom: CB x: 4.129000186920166 y:-0.6010000109672546 z:49.367000579833984
  - atom: CG1 x: 3.115000009536743 y:0.40299999713897705 z:49.87699890136719
  - atom: CG2 x: 4.514999866485596 y:-1.5449999570846558 z:50.49800109863281
- residue SER has 6 atoms
  - atom: N x: 1.6820000410079956 y:-0.3880000114440918 z:47.029998779296875
  - atom: CA x: 1.0529999732971191 y:0.453000009059906 z:46.0260009765625
  - atom: C x: 1.2309999465942383 y:1.902999997138977 z:46.388999938964844
  - atom: O x: 0.7540000081062317 y:2.3459999561309814 z:47.4370002746582
  - atom: CB x: -0.4350000023841858 y:0.12800000607967377 z:45.922000885009766
  - atom: OG x: -0.9639999866485596 y:0.6510000228881836 z:44.733001708984375
- residue GLY has 4 atoms
  - atom: N x: 1.9040000438690186 y:2.6449999809265137 z:45.51599884033203
  - atom: CA x: 2.1429998874664307 y:4.053999900817871 z:45.7760009765625
  - atom: C x: 3.634999990463257 y:4.316999912261963 z:45.90299987792969
  - atom: O x: 4.10099983215332 y:5.461999893188477 z:45.915000915527344
- residue ILE has 8 atoms
  - atom: N x: 4.392000198364258 y:3.2300000190734863 z:45.9900016784668
  - atom: CA x: 5.824999809265137 y:3.325000047683716 z:46.11600112915039
  - atom: C x: 6.491000175476074 y:2.7769999504089355 z:44.880001068115234
  - atom: O x: 6.464000225067139 y:1.5800000429153442 z:44.64099884033203
  - atom: CB x: 6.308000087738037 y:2.569000005722046 z:47.356998443603516
  - atom: CG1 x: 5.689000129699707 y:3.2200000286102295 z:48.59600067138672
  - atom: CG2 x: 7.823999881744385 y:2.5880000591278076 z:47.42300033569336
  - atom: CD1 x: 5.960000038146973 y:2.489000082015991 z:49.89400100708008
```

```
- residue GLY has 4 atoms
  - atom: N x: 7.071000099182129 y:3.6670000553131104 z:44.08700180053711
  - atom: CA x: 7.74399995803833 y:3.2360000610351562 z:42.8849983215332
  - atom: C x: 9.239999771118164 y:3.2160000801086426 z:43.108001708984375
  - atom: O x: 9.706999778747559 y:3.369999885559082 z:44.23500061035156
- residue PRO has 7 atoms
  - atom: N x: 10.027999877929688 y:3.0280001163482666 z:42.04600143432617
  - atom: CA x: 11.480999946594238 y:3.0039999948501587 z:42.20600128173828
  - atom: C x: 11.973999977111816 y:4.1529998779296875 z:43.07600021362305
  - atom: O x: 12.366000175476074 y:3.944999933242798 z:44.2140007019043
  - atom: CB x: 11.97599983215332 y:3.0999999046325684 z:40.766998291015625
  - atom: CG x: 10.942999839782715 y:2.306999921798706 z:40.03300094604492
  - atom: CD x: 9.645000457763672 y:2.819000005722046 z:40.638999938964844
- residue SER has 6 atoms
  - atom: N x: 11.925999641418457 y:5.367000102996826 z:42.54199981689453
  - atom: CA x: 12.390000343322754 y:6.557000160217285 z:43.24700164794922
  - atom: C x: 12.029000282287598 y:6.611000061035156 z:44.720001220703125
  - atom: O x: 12.913999557495117 y:6.75 z:45.55099868774414
  - atom: CB x: 11.88700008392334 y:7.824999809265137 z:42.54899978637695
  - atom: OG x: 12.267000198364258 y:9.003000259399414 z:43.255001068115234
- residue ALA has 5 atoms
  - atom: N x: 10.75 y:6.5229997634887695 z:45.060001373291016
  - atom: CA x: 10.368000030517578 y:6.559999942779541 z:46.47700119018555
  - atom: C x: 11.04699993133545 y:5.427999973297119 z:47.2599983215332
  - atom: O x: 11.673999786376953 y:5.670000076293945 z:48.2869987487793
  - atom: CB x: 8.85200023651123 y:6.466000080108643 z:46.63600158691406
- residue ALA has 5 atoms
  - atom: N x: 10.935999870300293 y:4.192999839782715 z:46.78499984741211
  - atom: CA x: 11.574999809265137 y:3.0989999771118164 z:47.499000549316406
  - atom: C x: 13.069999694824219 y:3.3540000915527344 z:47.66600036621094
  - atom: O x: 13.619000434875488 y:3.138000011444092 z:48.733001708984375
  - atom: CB x: 11.357000350952148 y:1.7940000295639038 z:46.77399826049805
- residue ARG has 11 atoms
  - atom: N x: 13.737000465393066 y:3.808000087738037 z:46.617000579833984
  - atom: CA x: 15.163999557495117 y:4.068999767303467 z:46.73899841308594
  - atom: C x: 15.439000129699707 y:5.125999927520752 z:47.801998138427734
  - atom: O x: 16.441999435424805 y:5.046999931335449 z:48.50899887084961
  - atom: CB x: 15.765000343322754 y:4.5269999504089355 z:45.414001464843750
  - atom: CG x: 17.187000274658203 y:5.013999938964844 z:45.55500030517578
  - atom: CD x: 17.774999618530273 y:5.38100004196167 z:44.2130012512207
  - atom: NE x: 17.91900062561035 y:4.206999778747559 z:43.36199951171875
  - atom: CZ x: 18.624000549316406 y:3.128000020980835 z:43.6879997253418
  - atom: NH1 x: 19.253999710083008 y:3.075000047683716 z:44.85200119018555
  - atom: NH2 x: 18.694000244140625 y:2.1010000705718994 z:42.85100173950195
- residue LYS has 9 atoms
  - atom: N x: 14.557000160217285 y:6.117000102996826 z:47.91299819946289
  - atom: CA x: 14.75 y:7.156000137329102 z:48.909000396728516
  - atom: C x: 14.637999534606934 y:6.564000129699707 z:50.31399917602539
  - atom: O x: 15.583000183105469 y:6.656000137329102 z:51.0880012512207
  - atom: CB x: 13.739999771118164 y:8.288999557495117 z:48.73099899291992
  - atom: CG x: 13.96399974822998 y:9.401000022888184 z:49.72600173950195
  - atom: CD x: 13.178999900817871 y:10.670000076293945 z:49.441001892089844
  - atom: CE x: 13.61400032043457 y:11.75100040435791 z:50.43299865722656
  - atom: NZ x: 12.967000007629395 y:13.07800006866455 z:50.22100067138672
- residue PHE has 11 atoms
  - atom: N x: 13.501999855041504 y:5.952000141143799 z:50.64099884033203
  - atom: CA x: 13.32699966430664 y:5.343999862670898 z:51.95600128173828
  - atom: C x: 14.543999671936035 y:4.506999969482422 z:52.38100051879883
  - atom: O x: 15.128999710083008 y:4.742000102996826 z:53.430999755859375
```

```
  - atom: CB x: 12.090999603271484 y:4.448999881744385 z:51.974998474121094
  - atom: CG x: 10.807000160217285 y:5.189000129699707 z:51.819000244140625
  - atom: CD1 x: 10.470999717712402 y:6.210000038146973 z:52.6879997253418
  - atom: CD2 x: 9.913000106811523 y:4.8480000495910645 z:50.8129997253418
  - atom: CE1 x: 9.260000228881836 y:6.885000228881836 z:52.5629997253418
  - atom: CE2 x: 8.70199966430664 y:5.513000011444092 z:50.67900085449219
  - atom: CZ x: 8.378000259399414 y:6.534999847412109 z:51.560001373291016
- residue VAL has 7 atoms
  - atom: N x: 14.911999702453613 y:3.5230000019073486 z:51.564998626708984
  - atom: CA x: 16.048999786376953 y:2.6570000648498535 z:51.86399841308594
  - atom: C x: 17.27199935913086 y:3.493000030517578 z:52.22100067138672
  - atom: O x: 17.93000030517578 y:3.2239999771118164 z:53.21799850463867
  - atom: CB x: 16.365999221801758 y:1.7050000429153442 z:50.66899871826172
  - atom: CG1 x: 17.589000701904297 y:0.8560000061988831 z:50.96099853515625
  - atom: CG2 x: 15.1850004196167 y:0.796999990940094 z:50.41600036621094
- residue ASP has 8 atoms
  - atom: N x: 17.565000534057617 y:4.51800012588501 z:51.428001403808594
  - atom: CA x: 18.70800018310547 y:5.386000156402588 z:51.70800018310547
  - atom: C x: 18.582000732421875 y:6.093999862670898 z:53.05699920654297
  - atom: O x: 19.562000274658203 y:6.625999927520752 z:53.564998626708984
  - atom: CB x: 18.885000228881836 y:6.446000099182129 z:50.612998962402344
  - atom: CG x: 19.240999221801758 y:5.841000080108643 z:49.27000045776367
  - atom: OD1 x: 19.964000701904297 y:4.815999984741211 z:49.257999420166016
  - atom: OD2 x: 18.80900001525879 y:6.396999835968018 z:48.229000091552734
- residue GLU has 9 atoms
  - atom: N x: 17.381000518798828 y:6.117000102996826 z:53.62699890136719
  - atom: CA x: 17.163999557495117 y:6.752999782562256 z:54.92300033569336
  - atom: C x: 16.847999572753906 y:5.6570000648498535 z:55.96799850463867
  - atom: O x: 16.23900032043457 y:5.9070000648498535 z:57.007999420166016
  - atom: CB x: 16.024999618530273 y:7.790999889373779 z:54.821998596191406
  - atom: CG x: 16.170000076293945 y:8.746000289916992 z:53.61800003051758
  - atom: CD x: 15.095000267028809 y:9.843000411987305 z:53.54600143432617
  - atom: OE1 x: 13.904000282287598 y:9.550999641418457 z:53.82699966430664
  - atom: OE2 x: 15.446000099182129 y:10.996000289916992 z:53.183998107910156
- residue GLY has 4 atoms
  - atom: N x: 17.257999420166016 y:4.429999828338623 z:55.65800094604492
  - atom: CA x: 17.05900001525879 y:3.325000047683716 z:56.571998596191406
  - atom: C x: 15.710000038146973 y:2.6449999809265137 z:56.645999908447266
  - atom: O x: 15.593000411987305 y:1.6239999532699585 z:57.33100128173828
- residue ILE has 8 atoms
  - atom: N x: 14.696999549865723 y:3.197999954223633 z:55.98099899291992
  - atom: CA x: 13.359000205993652 y:2.5959999561309814 z:55.972999572753906
  - atom: C x: 13.357000350952148 y:1.5210000276565552 z:54.89799880981445
  - atom: O x: 13.423999786376953 y:1.8480000495910645 z:53.7130012512207
  - atom: CB x: 12.27299976348877 y:3.6140000820159912 z:55.59700012207031
  - atom: CG1 x: 12.300999641418457 y:4.796999931335449 z:56.5620002746582
  - atom: CG2 x: 10.91100025177002 y:2.947999954223633 z:55.63399887084961
  - atom: CD1 x: 11.234000205993652 y:5.835999965667725 z:56.28099822998047
- residue LYS has 9 atoms
  - atom: N x: 13.265000343322754 y:0.25099998712539673 z:55.29199981689453
  - atom: CA x: 13.29800033569336 y:-0.828000009059906 z:54.305999755859375
  - atom: C x: 12.36299991607666 y:-2.012000083923340 z:54.54800033569336
  - atom: O x: 12.54699993133545 y:-3.068000078201294 z:53.96500015258789
  - atom: CB x: 14.72599983215332 y:-1.3580000400543213 z:54.194000244140625
  - atom: CG x: 15.210000038146973 y:-2.0199999809265137 z:55.479000091552734
  - atom: CD x: 16.617000579833984 y:-2.6059999465942383 z:55.35300064086914
  - atom: CE x: 17.683000564575195 y:-1.5269999504089355 z:55.19200134277344
  - atom: NZ x: 19.045000076293945 y:-2.1089999675750732 z:55.125999450683594
- residue THR has 7 atoms
```

```
  - atom: N x: 11.362000465393066 y:-1.8580000400543213 z:55.39500045776367
  - atom: CA x: 10.472999572753906 y:-2.9719998836517334 z:55.65399932861328
  - atom: C x: 9.102000236511230 y:-2.507999897003174 z:56.108001708984375
  - atom: O x: 8.946999549865723 y:-1.378999948501587 z:56.555999755859375
  - atom: CB x: 11.092000007629395 y:-3.8940000534057617 z:56.70600128173828
  - atom: OG1 x: 11.718000411987305 y:-3.0959999561309814 z:57.71200180053711
  - atom: CG2 x: 12.14799976348877 y:-4.789000034332275 z:56.07899856567383
- residue LEU has 8 atoms
  - atom: N x: 8.11299991607666 y:-3.38700008392334 z:55.98899841308594
  - atom: CA x: 6.741000175476074 y:-3.065000057220459 z:56.361000061035156
  - atom: C x: 6.663000106811523 y:-2.3540000915527344 z:57.698001861572266
  - atom: O x: 5.9079999923706055 y:-1.3930000066757202 z:57.867000579833984
  - atom: CB x: 5.900000095367432 y:-4.335999965667725 z:56.41899871826172
  - atom: CG x: 4.75600004196167 y:-4.4770002365112305 z:55.40999984741211
  - atom: CD1 x: 5.265999794006348 y:-4.355999946594238 z:53.98500061035156
  - atom: CD2 x: 4.0980000495910645 y:-5.828000068664551 z:55.604000091552734
- residue GLU has 9 atoms
  - atom: N x: 7.443999767303467 y:-2.8399999141693115 z:58.652000427246094
  - atom: CA x: 7.4720001220703125 y:-2.259000062942505 z:59.97999954223633
  - atom: C x: 7.995999813079834 y:-0.8249999880790710 z:59.93000030517578
  - atom: O x: 7.494999885559082 y:0.04899999871850014 z:60.625
  - atom: CB x: 8.338000297546387 y:-3.1240000724792480 z:60.90299987792969
  - atom: CG x: 7.719999790191650 y:-4.488999843597412 z:61.23899841308594
  - atom: CD x: 7.673999786376953 y:-5.454999923706055 z:60.05400085449219
  - atom: OE1 x: 8.753000259399414 y:-5.8520002365112305 z:59.5629997253418
  - atom: OE2 x: 6.560999870300293 y:-5.826000213623047 z:59.61600112915039
- residue ASP has 8 atoms
  - atom: N x: 9.005000114440918 y:-0.5849999785423279 z:59.104000091552734
  - atom: CA x: 9.571999549865723 y:0.7480000257492065 z:58.9739990234375
  - atom: C x: 8.531999588012695 y:1.7180000543594360 z:58.41600036621094
  - atom: O x: 8.362000465393066 y:2.8380000591278076 z:58.92399978637695
  - atom: CB x: 10.795999526977539 y:0.6819999814033508 z:58.07600021362305
  - atom: CG x: 11.923000335693360 y:-0.08399999886751175 z:58.7140007019043
  - atom: OD1 x: 11.635000228881836 y:-1.0809999704360962 z:59.4010009765625
  - atom: OD2 x: 13.090999603271484 y:0.3059999942779541 z:58.529998779296875
- residue LEU has 8 atoms
  - atom: N x: 7.834000110626221 y:1.2890000343322754 z:57.37099838256836
  - atom: CA x: 6.793000221252441 y:2.1170001029968260 z:56.79399871826172
  - atom: C x: 5.763000011444092 y:2.3210000991821290 z:57.895999908447266
  - atom: O x: 5.111000061035156 y:3.3589999675750732 z:57.95600128173828
  - atom: CB x: 6.164000034332275 y:1.4190000295639038 z:55.58599853515625
  - atom: CG x: 7.113999843597412 y:1.1069999933242798 z:54.428001403808594
  - atom: CD1 x: 6.370999813079834 y:0.3479999899864197 z:53.361000061035156
  - atom: CD2 x: 7.696000099182129 y:2.390000104904175 z:53.86399841308594
- residue ARG has 11 atoms
  - atom: N x: 5.632999897003174 y:1.3179999589920044 z:58.76499938964844
  - atom: CA x: 4.715000152587891 y:1.3680000305175781 z:59.90700149536133
  - atom: C x: 5.168000221252441 y:2.438999891281128 z:60.90299987792969
  - atom: O x: 4.3460001945495605 y:3.1530001163482666 z:61.48099899291992
  - atom: CB x: 4.678999900817871 y:0.017999999225139618 z:60.619998931884766
  - atom: CG x: 3.812999963760376 y:-1.0190000534057617 z:59.95800018310547
  - atom: CD x: 2.361999988555908 y:-0.7979999780654907 z:60.33000183105469
  - atom: NE x: 1.4429999589920044 y:-1.5540000200271606 z:59.483001708984375
  - atom: CZ x: 1.4910000562667847 y:-2.869999885559082 z:59.29999923706055
  - atom: NH1 x: 2.4179999828338623 y:-3.5999999046325684 z:59.909000396728516
  - atom: NH2 x: 0.6159999966621399 y:-3.4560000896453857 z:58.4900016784668
- residue LYS has 9 atoms
  - atom: N x: 6.4770002365112305 y:2.5450000762939453 z:61.104000091552734
  - atom: CA x: 7.011000156402588 y:3.5309998989105225 z:62.02199935913086
```

```
  - atom: C x: 6.9720001220703125 y:4.920000076293945 z:61.40399932861328
  - atom: O x: 6.544000148773193 y:5.894000053405762 z:62.04800033569336
  - atom: CB x: 8.460000038146973 y:3.2070000171661377 z:62.402000427246094
  - atom: CG x: 9.081999778747559 y:4.255000114440918 z:63.327999114990234
  - atom: CD x: 10.49899959564209 y:3.9030001163482666 z:63.790000915527344
  - atom: CE x: 11.552000045776367 y:4.183000087738037 z:62.72600173950195
  - atom: NZ x: 12.932000160217285 y:4.047999858856201 z:63.28200149536133
- residue ASN has 8 atoms
  - atom: N x: 7.419000148773193 y:5.011000156402588 z:60.154998779296875
  - atom: CA x: 7.459000110626221 y:6.291999816894531 z:59.483001708984375
  - atom: C x: 6.232999801635742 y:6.5929999351501465 z:58.65599822998047
  - atom: O x: 6.293000221252441 y:7.327000141143799 z:57.68299865722656
  - atom: CB x: 8.72700023651123 y:6.385000228881836 z:58.63999938964844
  - atom: CG x: 9.965999603271484 y:6.585999965667725 z:59.49399948120117
  - atom: OD1 x: 11.059000015258789 y:6.823999881744385 z:58.986000061035156
  - atom: ND2 x: 9.795000076293945 y:6.491000175476074 z:60.81100082397461
- residue GLU has 9 atoms
  - atom: N x: 5.109000205993652 y:6.044000148773193 z:59.08000183105469
  - atom: CA x: 3.8510000705718994 y:6.230999946594238 z:58.387001037597656
  - atom: C x: 3.5390000343322754 y:7.684000015258789 z:58.02299880981445
  - atom: O x: 3.0429999828338623 y:7.9670000076293945 z:56.933998107910156
  - atom: CB x: 2.7279999256134033 y:5.677000045776367 z:59.244998931884766
  - atom: CG x: 1.3849999904632568 y:5.681000232696533 z:58.58300018310547
  - atom: CD x: 0.2879999876022339 y:5.394999980926514 z:59.57400131225586
  - atom: OE1 x: 0.3709999918937683 y:4.359000205993652 z:60.27399826049805
  - atom: OE2 x: -0.6549999713897705 y:6.210999965667725 z:59.65399932861328
- residue ASP has 8 atoms
  - atom: N x: 3.8320000171661377 y:8.607999801635742 z:58.926998138427734
  - atom: CA x: 3.5360000133514404 y:10.008000373840332 z:58.665000915527344
  - atom: C x: 4.449999809265137 y:10.621000289916992 z:57.61399841308594
  - atom: O x: 4.502999782562256 y:11.836999893188477 z:57.472999572753906
  - atom: CB x: 3.615000009536743 y:10.812000274658203 z:59.96900177001953
  - atom: CG x: 5.039000034332275 y:11.223999977111816 z:60.327999114990234
  - atom: OD1 x: 5.9720001220703125 y:10.38599967956543 z:60.19599914550781
  - atom: OD2 x: 5.216000080108643 y:12.392000198364258 z:60.75600051879883
- residue LYS has 9 atoms
  - atom: N x: 5.159999847412109 y:9.774999618530273 z:56.874000549316406
  - atom: CA x: 6.084000110626221 y:10.223999977111816 z:55.83000183105469
  - atom: C x: 5.663000106811523 y:9.597999572753906 z:54.479000091552734
  - atom: O x: 6.453000068664551 y:9.491999626159668 z:53.529998779296875
  - atom: CB x: 7.515999794006348 y:9.8149995803833 z:56.21200180053711
  - atom: CG x: 8.607000350952148 y:10.362000465393066 z:55.303001403808594
  - atom: CD x: 9.99899959564209 y:9.871000289916992 z:55.709999084472656
  - atom: CE x: 10.60200023651123 y:10.678999900817871 z:56.86199951171875
  - atom: NZ x: 9.82699966430664 y:10.614000320434570 z:58.130001068115234
- residue LEU has 8 atoms
  - atom: N x: 4.395999908447266 y:9.199000358581543 z:54.41299819946289
  - atom: CA x: 3.8239998817443848 y:8.581000328063965 z:53.22800064086914
  - atom: C x: 2.6070001125335693 y:9.369999885559082 z:52.75899887084961
  - atom: O x: 1.850000023841858 y:9.902000427246094 z:53.573001861572266
  - atom: CB x: 3.36299991607666 y:7.158999919891357 z:53.55099868774414
  - atom: CG x: 4.328999996185303 y:6.171999931335449 z:54.19900131225586
  - atom: CD1 x: 3.621999979019165 y:4.828999996185303 z:54.38100051879883
  - atom: CD2 x: 5.563000202178955 y:6.019000053405762 z:53.32899856567383
- residue ASN has 8 atoms
  - atom: N x: 2.4019999504089355 y:9.430999755859375 z:51.45000076293945
  - atom: CA x: 1.2419999837875366 y:10.133999824523926 z:50.926998138427734
  - atom: C x: 0.014000000432133675 y:9.26099967956543 z:51.11600112915039
  - atom: O x: 0.12300000339746475 y:8.086999893188477 z:51.41600036621094
```

```
  - atom: CB x: 1.4190000295639038 y:10.456000328063965 z:49.446998596191406
  - atom: CG x: 1.8220000267028809 y:9.25100040435791 z:48.641998291015625
  - atom: OD1 x: 1.2480000257492065 y:8.178000450134277 z:48.77899932861328
  - atom: ND2 x: 2.812999963760376 y:9.425999641418457 z:47.790000915527344
- residue HIS has 10 atoms
  - atom: N x: -1.1540000438690186 y:9.850000381469727 z:50.94499969482422
  - atom: CA x: -2.4070000648498535 y:9.142999649047852 z:51.106998443603516
  - atom: C x: -2.4030001163482666 y:7.800000190734863 z:50.391998291015625
  - atom: O x: -2.8350000381469727 y:6.7829999923706055 z:50.939998626708984
  - atom: CB x: -3.5369999408721924 y:10.003999710083008 z:50.566001892089844
  - atom: CG x: -4.88100004196167 y:9.364999771118164 z:50.6619987487793
  - atom: ND1 x: -5.572999954223633 y:9.265000343322754 z:51.8489990234375
  - atom: CD2 x: -5.666999816894531 y:8.79800033569336 z:49.720001220703125
  - atom: CE1 x: -6.730000019073486 y:8.668000221252441 z:51.6349983215332
  - atom: NE2 x: -6.811999797821045 y:8.37399959564209 z:50.35100173950195
- residue HIS has 10 atoms
  - atom: N x: -1.8949999809265137 y:7.797999858856201 z:49.17100143432617
  - atom: CA x: -1.8609999418258667 y:6.583000183105469 z:48.375
  - atom: C x: -0.9599999785423279 y:5.51200008392334 z:48.986000061035156
  - atom: O x: -1.3209999799728394 y:4.3420000076293945 z:49.013999938964844
  - atom: CB x: -1.4359999895095825 y:6.927999973297119 z:46.933998107910156
  - atom: CG x: -1.406000018119812 y:5.757999897003174 z:45.99700164794922
  - atom: ND1 x: -1.434999942779541 y:5.908999919891357 z:44.62900161743164
  - atom: CD2 x: -1.2580000162124634 y:4.429999828338623 z:46.2239990234375
  - atom: CE1 x: -1.3009999990463257 y:4.728000164031982 z:44.05400085449219
  - atom: NE2 x: -1.190999984741211 y:3.815000057220459 z:45.000999450683594
- residue GLN has 9 atoms
  - atom: N x: 0.20800000429153442 y:5.910999774932861 z:49.47100067138672
  - atom: CA x: 1.1390000581741333 y:4.968999862670898 z:50.07500076293945
  - atom: C x: 0.5680000185966492 y:4.394000053405762 z:51.38800048828125
  - atom: O x: 0.800000011920929 y:3.2219998836517334 z:51.69599914550781
  - atom: CB x: 2.499000072479248 y:5.65500020980835 z:50.30099868774414
  - atom: CG x: 3.200000047683716 y:6.076000213623047 z:49.00299835205078
  - atom: CD x: 4.383999824523926 y:7.015999794006348 z:49.22100067138672
  - atom: OE1 x: 4.260000228881836 y:8.043000221252441 z:49.88100051879883
  - atom: NE2 x: 5.531000137329102 y:6.669000148773193 z:48.65399932861328
- residue ARG has 11 atoms
  - atom: N x: -0.18000000715255737 y:5.201000213623047 z:52.150001525878906
  - atom: CA x: -0.777999997138977 y:4.710000038146973 z:53.39099884033203
  - atom: C x: -1.7100000381469727 y:3.572999954223633 z:53.02399826049805
  - atom: O x: -1.562999963760376 y:2.4649999141693115 z:53.52199935913086
  - atom: CB x: -1.6130000352859497 y:5.7779998779296875 z:54.10200119018555
  - atom: CG x: -0.9010000228881836 y:6.5980000495910645 z:55.172000885009766
  - atom: CD x: -0.2150000035762787 y:7.790999889373779 z:54.56999969482422
  - atom: NE x: 0.1120000034570694 y:8.810999870300293 z:55.56100082397461
  - atom: CZ x: -0.7559999823570251 y:9.329000473022461 z:56.428001403808594
  - atom: NH1 x: -2.0209999084472656 y:8.916000366210938 z:56.444000244140625
  - atom: NH2 x: -0.3630000054836273 y:10.286999702453613 z:57.26100158691406
- residue ILE has 8 atoms
  - atom: N x: -2.6760001182556152 y:3.8610000610351562 z:52.15399932861328
  - atom: CA x: -3.6410000324249268 y:2.8570001125335693 z:51.694000244140625
  - atom: C x: -2.88700008392334 y:1.61899995803833 z:51.20600128173828
  - atom: O x: -3.2730000019073486 y:0.48100000619888306 z:51.500999450683594
  - atom: CB x: -4.520999908447266 y:3.4000000953674316 z:50.507999420166016
  - atom: CG1 x: -5.324999809265137 y:4.625999927520752 z:50.95399856567383
  - atom: CG2 x: -5.440999984741211 y:2.302999973297119 z:49.98400115966797
  - atom: CD1 x: -6.333000183105469 y:4.361000061035156 z:52.060001373291016
- residue GLY has 4 atoms
  - atom: N x: -1.8070000410079956 y:1.8609999418258667 z:50.465999603271484
```

```
  - atom: CA x: -0.996999979019165 y:0.7850000262260437 z:49.92599868774414
  - atom: C x: -0.5400000214576721 y:-0.16500000655651093 z:51.00400161743164
  - atom: O x: -0.7350000143051147 y:-1.371000051498413 z:50.88600158691406
- residue LEU has 8 atoms
  - atom: N x: 0.05900000035762787 y:0.4000000059604645 z:52.053001403808594
  - atom: CA x: 0.5699999928474426 y:-0.34200000762939453 z:53.20199966430664
  - atom: C x: -0.5519999861717224 y:-1.0260000228881836 z:53.93899917602539
  - atom: O x: -0.4169999957084656 y:-2.1730000972747803 z:54.332000732421875
  - atom: CB x: 1.2879999876022339 y:0.6060000061988831 z:54.16699981689453
  - atom: CG x: 1.7599999904632568 y:0.08100000023841858 z:55.53300094604492
  - atom: CD1 x: 2.7009999752044678 y:-1.0989999771118164 z:55.36000061035156
  - atom: CD2 x: 2.4820001125335693 y:1.2120000123977661 z:56.27399826049805
- residue LYS has 9 atoms
  - atom: N x: -1.6549999713897705 y:-0.3050000071525574 z:54.11899948120117
  - atom: CA x: -2.8510000705718994 y:-0.7960000038146973 z:54.805999755859375
  - atom: C x: -3.437999963760376 y:-2.068000078201294 z:54.21099853515625
  - atom: O x: -3.9430000078201294 y:-2.9210000038146973 z:54.93199920654297
  - atom: CB x: -3.940000057220459 y:0.2750000059604645 z:54.79199981689453
  - atom: CG x: -5.296000003814697 y:-0.2150000035762787 z:55.270999908447266
  - atom: CD x: -6.254000186920166 y:0.9430000185966492 z:55.45000076293945
  - atom: CE x: -7.507999897003174 y:0.5070000290870667 z:56.176998138427734
  - atom: NZ x: -8.45300006866455 y:1.6299999952316284 z:56.4119987487793
- residue TYR has 12 atoms
  - atom: N x: -3.3940000534057617 y:-2.177999973297119 z:52.891998291015625
  - atom: CA x: -3.928999900817871 y:-3.3459999561309814 z:52.2140007019043
  - atom: C x: -2.806999921798706 y:-4.0879998207092285 z:51.51499938964844
  - atom: O x: -3.015000104904175 y:-4.678999900817871 z:50.43899917602539
  - atom: CB x: -4.980999946594238 y:-2.921999931335449 z:51.1870002746582
  - atom: CG x: -6.118000030517578 y:-2.1530001163482666 z:51.79399871826172
  - atom: CD1 x: -6.923999786376953 y:-2.7249999046325684 z:52.77000045776367
  - atom: CD2 x: -6.373000144958496 y:-0.8399999737739563 z:51.41600036621094
  - atom: CE1 x: -7.953999996185303 y:-2.003000020980835 z:53.3650016784668
  - atom: CE2 x: -7.40500020980835 y:-0.10899999737739563 z:51.999000549316406
  - atom: CZ x: -8.187999725341797 y:-0.6949999928474426 z:52.97100067138672
  - atom: OH x: -9.218999862670898 y:0.01899999938905239 z:53.53499984741211
- residue PHE has 11 atoms
  - atom: N x: -1.6200000047683716 y:-4.072000026702881 z:52.117000579833984
  - atom: CA x: -0.4959999918937683 y:-4.738999843597412 z:51.47700119018555
  - atom: C x: -0.7870000004768372 y:-6.201000213623047 z:51.15299987792969
  - atom: O x: -0.6729999780654907 y:-6.626999855041504 z:50.005001068115234
  - atom: CB x: 0.7450000047683716 y:-4.664000034332275 z:52.333000183105547
  - atom: CG x: 1.934000015258789 y:-5.28000020980835 z:51.691001892089844
  - atom: CD1 x: 2.6610000133514404 y:-4.579999923706055 z:50.74100112915039
  - atom: CD2 x: 2.321000099182129 y:-6.566999912261963 z:52.01900100708008
  - atom: CE1 x: 3.7679998874664307 y:-5.1479997634887695 z:50.12099838256836
  - atom: CE2 x: 3.427000045776367 y:-7.1579999923706055 z:51.409000396728516
  - atom: CZ x: 4.156000137329102 y:-6.446000099182129 z:50.45500183105469
- residue GLY has 4 atoms
  - atom: N x: -1.159999966621399 y:-6.968999862670898 z:52.16699981689453
  - atom: CA x: -1.4630000591278076 y:-8.36299991607666 z:51.941001892089844
  - atom: C x: -2.6440000534057617 y:-8.527000427246094 z:51.00699996948242
  - atom: O x: -2.572000026702881 y:-9.300000190734863 z:50.05500030517578
- residue ASP has 8 atoms
  - atom: N x: -3.7300000190734863 y:-7.806000232696533 z:51.2760009765625
  - atom: CA x: -4.914000034332275 y:-7.888999938964844 z:50.43600082397461
  - atom: C x: -4.5960001945495605 y:-7.710999965667725 z:48.95600128173828
  - atom: O x: -4.974999904632568 y:-8.550000190734863 z:48.137001037597656
  - atom: CB x: -5.949999809265137 y:-6.8480000495910645 z:50.856998443603516
  - atom: CG x: -6.870999813079834 y:-7.361999988555908 z:51.93000030517578
```

```
  - atom: OD1 x: -6.843999862670898 y:-8.590999603271484 z:52.15800094604492
  - atom: OD2 x: -7.624000072479248 y:-6.559999942779541 z:52.5369987487793
- residue PHE has 11 atoms
  - atom: N x: -3.8889999389648438 y:-6.633999824523926 z:48.61899948120117
  - atom: CA x: -3.5399999618530273 y:-6.348999977111816 z:47.233001708984375
  - atom: C x: -2.683000087738037 y:-7.379000186920166 z:46.516998291015625
  - atom: O x: -2.6549999713897705 y:-7.396999835968018 z:45.290000915527344
  - atom: CB x: -2.8589999675750732 y:-4.974999904632568 z:47.108001708984375
  - atom: CG x: -3.7939999103546143 y:-3.812000036239624 z:47.305999755859375
  - atom: CD1 x: -5.133999824523926 y:-4.020999908447266 z:47.62699890136719
  - atom: CD2 x: -3.3239998817443848 y:-2.5139999389648438 z:47.233001708984375
  - atom: CE1 x: -5.98199987411499 y:-2.9639999866485596 z:47.88100051879883
  - atom: CE2 x: -4.168000221252441 y:-1.437999963760376 z:47.48699951171875
  - atom: CZ x: -5.502999782562256 y:-1.6649999618530273 z:47.814998626708984
- residue GLU has 9 atoms
  - atom: N x: -1.9630000591278076 y:-8.22700023651123 z:47.244998931884766
  - atom: CA x: -1.1390000581741333 y:-9.21399974822998 z:46.55099868774414
  - atom: C x: -1.9110000133514404 y:-10.48900032043457 z:46.32899856567383
  - atom: O x: -1.409000039100647 y:-11.414999961853027 z:45.70899963378906
  - atom: CB x: 0.16899999976158142 y:-9.505000114440918 z:47.30400085449219
  - atom: CG x: 1.1510000228881836 y:-8.329999923706055 z:47.29899978637695
  - atom: CD x: 2.6040000915527344 y:-8.77299976348877 z:47.356998443603516
  - atom: OE1 x: 2.9189999103546143 y:-9.692000389099121 z:48.141998291015625
  - atom: OE2 x: 3.434000015258789 y:-8.196000099182129 z:46.624000549316406
- residue LYS has 9 atoms
  - atom: N x: -3.138000011444092 y:-10.534000396728516 z:46.82699966430664
  - atom: CA x: -3.9719998836517334 y:-11.711000442504883 z:46.63999938964844
  - atom: C x: -4.692999839782715 y:-11.647000312805176 z:45.2859992980957
  - atom: O x: -5.2129998207092285 y:-10.595000267028809 z:44.89500045776367
  - atom: CB x: -5.033999919891357 y:-11.817000389099121 z:47.742000579833984
  - atom: CG x: -4.519999980926514 y:-12.019000053405762 z:49.15399932861328
  - atom: CD x: -5.681000232696533 y:-12.40999984741211 z:50.0629997253418
  - atom: CE x: -5.315000057220459 y:-12.395000457763672 z:51.53200149536133
  - atom: NZ x: -4.199999809265137 y:-13.324000358581543 z:51.86800003051758
- residue ARG has 11 atoms
  - atom: N x: -4.708000183105469 y:-12.767999649047852 z:44.57099914550781
  - atom: CA x: -5.4120001792907715 y:-12.85200023651123 z:43.29800033569336
  - atom: C x: -6.906000137329102 y:-12.857999801635742 z:43.64899826049805
  - atom: O x: -7.281000137329102 y:-13.098999977111816 z:44.79600143432617
  - atom: CB x: -5.052000045776367 y:-14.151000022888184 z:42.57400131225586
  - atom: CG x: -3.5899999141693115 y:-14.277000427246094 z:42.189998626708984
  - atom: CD x: -3.253000020980835 y:-13.343000411987305 z:41.04800033569336
  - atom: NE x: -4.007999897003174 y:-13.678000450134277 z:39.84000015258789
  - atom: CZ x: -4.105000019073486 y:-12.894000053405762 z:38.768001556396484
  - atom: NH1 x: -3.496999979019165 y:-11.711999893188477 z:38.73899841308594
  - atom: NH2 x: -4.809000015258789 y:-13.298999786376953 z:37.71799850463867
- residue ILE has 8 atoms
  - atom: N x: -7.75600004196167 y:-12.595000267028809 z:42.66600036621094
  - atom: CA x: -9.199000358581543 y:-12.581000328063965 z:42.89099884033203
  - atom: C x: -9.857999801635742 y:-13.762999534606934 z:42.17300033569336
  - atom: O x: -9.720000267028809 y:-13.904000282287598 z:40.957000732421875
  - atom: CB x: -9.817000389099121 y:-11.274000167846680 z:42.358001708984375
  - atom: CG1 x: -9.052000045776367 y:-10.076000213623047 z:42.92499923706055
  - atom: CG2 x: -11.300999641418457 y:-11.206000328063965 z:42.729000091552734
  - atom: CD1 x: -9.505000114440918 y:-8.73900032043457 z:42.39799880981445
- residue PRO has 7 atoms
  - atom: N x: -10.585000038146973 y:-14.626999855041504 z:42.9119987487793
  - atom: CA x: -11.241999626159668 y:-15.781999588012695 z:42.277000427246094
  - atom: C x: -12.177000045776367 y:-15.265999794006348 z:41.198001861572266
```

```
  - atom: O x: -12.9399995803833 y:-14.333000183105469 z:41.449001312255860
  - atom: CB x: -12.015000343322754 y:-16.426000595092773 z:43.42599868774414
  - atom: CG x: -11.246999740600586 y:-15.991999626159668 z:44.65399932861328
  - atom: CD x: -10.90999984741211 y:-14.5600004196167 z:44.3489990234375
- residue ARG has 11 atoms
  - atom: N x: -12.13700008392334 y:-15.86400032043457 z:40.007999420166016
  - atom: CA x: -12.998000144958496 y:-15.402000427246094 z:38.92300033569336
  - atom: C x: -14.454999923706055 y:-15.329000473022461 z:39.33000183105469
  - atom: O x: -15.230999946594238 y:-14.598999977111816 z:38.73400115966797
  - atom: CB x: -12.876999855041504 y:-16.285999298095703 z:37.683998107910156
  - atom: CG x: -13.494000434875488 y:-15.621000289916992 z:36.470001220703125
  - atom: CD x: -13.788999557495117 y:-16.57900047302246 z:35.345001220703125
  - atom: NE x: -14.845000267028809 y:-16.03700065612793 z:34.49399948120117
  - atom: CZ x: -14.718000411987305 y:-14.97599983215332 z:33.696998596191406
  - atom: NH1 x: -13.567000389099121 y:-14.322999954223633 z:33.61399841308594
  - atom: NH2 x: -15.767000198364258 y:-14.541999816894531 z:33.007999420166016
- residue GLU has 9 atoms
  - atom: N x: -14.829000473022461 y:-16.084999084472656 z:40.34700012207031
  - atom: CA x: -16.200000762939453 y:-16.082000732421875 z:40.83100128173828
  - atom: C x: -16.530000686645508 y:-14.770999908447266 z:41.529998779296875
  - atom: O x: -17.562000274658203 y:-14.151000022888184 z:41.26499938964844
  - atom: CB x: -16.413999557495117 y:-17.25200080871582 z:41.79399871826172
  - atom: CG x: -17.667999267578125 y:-17.152000427246094 z:42.63999938964844
  - atom: CD x: -18.077999114990234 y:-18.496999740600586 z:43.21699905395508
  - atom: OE1 x: -18.999000549316406 y:-18.52400016784668 z:44.064998626708984
  - atom: OE2 x: -17.48699951171875 y:-19.530000686645508 z:42.816001892089844
- residue GLU has 9 atoms
  - atom: N x: -15.652999877929688 y:-14.359000205993652 z:42.4370002746582
  - atom: CA x: -15.861000061035156 y:-13.123000144958496 z:43.167999267578125
  - atom: C x: -15.913000106811523 y:-11.949999809265137 z:42.1879997253418
  - atom: O x: -16.642000198364258 y:-10.979999542236328 z:42.409000396728516
  - atom: CB x: -14.75100040435791 y:-12.937999725341797 z:44.20800018310547
  - atom: CG x: -14.975000381469727 y:-13.76099967956543 z:45.46699905395508
  - atom: CD x: -13.857000350952148 y:-13.616000175476074 z:46.49100112915039
  - atom: OE1 x: -13.395000457763672 y:-12.477999687194824 z:46.71699905395508
  - atom: OE2 x: -13.449999809265137 y:-14.640999794006348 z:47.08100128173828
- residue MET has 8 atoms
  - atom: N x: -15.138999938964844 y:-12.048999786376953 z:41.106998443603516
  - atom: CA x: -15.130000114440918 y:-11.010000228881836 z:40.0890007019043
  - atom: C x: -16.57699966430664 y:-10.881999969482422 z:39.64799880981445
  - atom: O x: -17.172000885009766 y:-9.8100004196167 z:39.71099853515625
  - atom: CB x: -14.267999649047852 y:-11.425999641418457 z:38.893001556396484
  - atom: CG x: -12.784000396728516 y:-11.329000473022461 z:39.141998291015625
  - atom: SD x: -12.277000427246094 y:-9.61400032043457 z:39.38199996948242
  - atom: CE x: -11.970999717712402 y:-9.13599967956543 z:37.689998626708984
- residue LEU has 8 atoms
  - atom: N x: -17.142000198364258 y:-12.005999565124512 z:39.222999572753906
  - atom: CA x: -18.52199935913086 y:-12.043999671936035 z:38.777000427246094
  - atom: C x: -19.43400001525879 y:-11.295000076293945 z:39.74399948120117
  - atom: O x: -20.256000518798828 y:-10.489999771118164 z:39.3129997253418
  - atom: CB x: -18.979000091552734 y:-13.49899959564209 z:38.625999450683594
  - atom: CG x: -18.22599983215332 y:-14.336000442504883 z:37.57699966430664
  - atom: CD1 x: -18.56999969482422 y:-15.805999755859375 z:37.749000549316406
  - atom: CD2 x: -18.573999404907227 y:-13.864999771118164 z:36.16899871826172
- residue GLN has 9 atoms
  - atom: N x: -19.2810001373291 y:-11.53499984741211 z:41.04499816894531
  - atom: CA x: -20.1299991607666 y:-10.86299991607666 z:42.040000915527344
  - atom: C x: -19.868999481201172 y:-9.359000205993652 z:42.04899978637695
  - atom: O x: -20.73900032043457 y:-8.557000160217285 z:42.40599822998047
```

```
  - atom: CB x: -19.88800048828125 y:-11.430999755859375 z:43.44499969482422
  - atom: CG x: -19.590999603271484 y:-12.925000190734863 z:43.465999603271484
  - atom: CD x: -19.812999725341797 y:-13.557999610900879 z:44.827999114990234
  - atom: OE1 x: -19.44499969482422 y:-14.71399974822998 z:45.05099868774414
  - atom: NE2 x: -20.426000595092773 y:-12.809000015258789 z:45.742000579833984
- residue MET has 8 atoms
  - atom: N x: -18.6560001373291 y:-8.982999801635742 z:41.66299819946289
  - atom: CA x: -18.284000396728516 y:-7.578999996185303 z:41.612998962402344
  - atom: C x: -18.82900047302246 y:-7.03000020980835 z:40.29999923706055
  - atom: O x: -19.385000228881836 y:-5.929999828338623 z:40.25899887084961
  - atom: CB x: -16.759000778198242 y:-7.435999870300293 z:41.68299865722656
  - atom: CG x: -16.158000946044922 y:-7.941999912261963 z:42.99399948120117
  - atom: SD x: -14.380999565124512 y:-8.246999740600586 z:42.926998138427734
  - atom: CE x: -13.717000007629395 y:-6.666999816894531 z:43.327999114990234
- residue GLN has 9 atoms
  - atom: N x: -18.690000534057617 y:-7.821000099182129 z:39.236000061035156
  - atom: CA x: -19.16900062561035 y:-7.436999797821045 z:37.9119987487793
  - atom: C x: -20.66699981689453 y:-7.296000003814697 z:37.957000732421875
  - atom: O x: -21.280000686645508 y:-6.793000221252441 z:37.0260009765625
  - atom: CB x: -18.819000244140625 y:-8.5 z:36.875999450683594
  - atom: CG x: -19.424999237060547 y:-8.256999969482422 z:35.5
  - atom: CD x: -19.215999603271484 y:-9.430999755859375 z:34.56100082397461
  - atom: OE1 x: -19.83799934387207 y:-10.493000030517578 z:34.7140007019043
  - atom: NE2 x: -18.32699966430664 y:-9.251999855041504 z:33.58399963378906
- residue ASP has 8 atoms
  - atom: N x: -21.257999420166016 y:-7.749000072479248 z:39.04999923706055
  - atom: CA x: -22.69499969482422 y:-7.681000232696533 z:39.194000244140625
  - atom: C x: -23.049999237060547 y:-6.394999980926514 z:39.89699935913086
  - atom: O x: -23.94499969482422 y:-5.671999931335449 z:39.46900177001953
  - atom: CB x: -23.18600082397461 y:-8.876999855041504 z:40.00299835205078
  - atom: CG x: -24.58799934387207 y:-9.300999641418457 z:39.630001068115234
  - atom: OD1 x: -25.54599952697754 y:-8.546999931335449 z:39.91600036621094
  - atom: OD2 x: -24.722000122070312 y:-10.397000312805176 z:39.04600143432617
- residue ILE has 8 atoms
  - atom: N x: -22.33099937438965 y:-6.107999801635742 z:40.97600173950195
  - atom: CA x: -22.56999969482422 y:-4.901000022888184 z:41.7599983215332
  - atom: C x: -22.198999404907227 y:-3.635999917984009 z:40.992000579833984
  - atom: O x: -22.82900047302246 y:-2.5940001010894775 z:41.16600036621094
  - atom: CB x: -21.783000946044922 y:-4.942999839782715 z:43.0989990234375
  - atom: CG1 x: -22.229999542236328 y:-6.1579999923706055 z:43.92100143432617
  - atom: CG2 x: -22.006999969482422 y:-3.6570000648498535 z:43.88999938964844
  - atom: CD1 x: -21.645999908447266 y:-6.208000183105469 z:45.327999114990234
- residue VAL has 7 atoms
  - atom: N x: -21.18000030517578 y:-3.7290000915527344 z:40.14099884033203
  - atom: CA x: -20.750999450683594 y:-2.5759999752044678 z:39.361000061035156
  - atom: C x: -21.81100082397461 y:-2.3359999656677246 z:38.30500030517578
  - atom: O x: -22.53499984741211 y:-1.3489999771118164 z:38.35900115966797
  - atom: CB x: -19.393999099731445 y:-2.825000047683716 z:38.67300033569336
  - atom: CG1 x: -18.95199966430664 y:-1.5759999752044678 z:37.946998596191406
  - atom: CG2 x: -18.347000122070312 y:-3.2209999561309814 z:39.69900131225586
- residue LEU has 8 atoms
  - atom: N x: -21.909000396728516 y:-3.263000011444092 z:37.358001708984375
  - atom: CA x: -22.89699935913086 y:-3.1779999732971119 z:36.290000915527344
  - atom: C x: -24.26300048828125 y:-2.7330000400543213 z:36.8120002746582
  - atom: O x: -24.941999435424805 y:-1.906000018119812 z:36.19200134277344
  - atom: CB x: -23.040000915527344 y:-4.533999919891357 z:35.59600067138672
  - atom: CG x: -22.295000076293945 y:-4.75 z:34.27399826049805
  - atom: CD1 x: -20.80900001525879 y:-4.521999835968018 z:34.46500015258789
  - atom: CD2 x: -22.566999435424805 y:-6.160999774932861 z:33.75899887084961
```

```
- residue ASN has 8 atoms
  - atom: N x: -24.6560001373291 y:-3.2720000743865967 z:37.96200180053711
  - atom: CA x: -25.947999954223633 y:-2.943000078201294 z:38.551998138427734
  - atom: C x: -26.011999130249023 y:-1.4980000257492065 z:39.018001556396484
  - atom: O x: -26.774999618530273 y:-0.6959999799728394 z:38.49100112915039
  - atom: CB x: -26.240999221801758 y:-3.865999937057495 z:39.736000061035156
  - atom: CG x: -27.707000732421875 y:-3.8570001125335693 z:40.130001068115234
  - atom: OD1 x: -28.28499984741211 y:-2.805999994277954 z:40.42100143432617
  - atom: ND2 x: -28.31800079345703 y:-5.035999774932861 z:40.14099884033203
- residue GLU has 9 atoms
  - atom: N x: -25.208999633789062 y:-1.1749999523162842 z:40.02000045776367
  - atom: CA x: -25.179000854492188 y:0.16699999570846558 z:40.577999114990234
  - atom: C x: -25.06399917602539 y:1.25 z:39.516998291015625
  - atom: O x: -25.56100082397461 y:2.3570001125335693 z:39.70199966430664
  - atom: CB x: -24.020999908447266 y:0.2759999930858612 z:41.569000244140625
  - atom: CG x: -24.288000106811523 y:-0.4399999976158142 z:42.861000061035156
  - atom: CD x: -25.4689998626709 y:0.15700000524520874 z:43.57600021362305
  - atom: OE1 x: -25.375999450683594 y:1.3259999752044678 z:43.999000549316406
  - atom: OE2 x: -26.4950008392334 y:-0.5339999794960022 z:43.70399856567383
- residue VAL has 7 atoms
  - atom: N x: -24.398000717163086 y:0.9309999942779541 z:38.41400146484375
  - atom: CA x: -24.229999542236328 y:1.8899999856948853 z:37.33399963378906
  - atom: C x: -25.618999481201172 y:2.247999906539917 z:36.79499816894531
  - atom: O x: -25.95599937438965 y:3.4240000247955322 z:36.630001068115234
  - atom: CB x: -23.381000518798828 y:1.2999999523162842 z:36.17900085449219
  - atom: CG1 x: -23.075000762939453 y:2.36899995803833 z:35.16899871826172
  - atom: CG2 x: -22.10099983215332 y:0.7269999980926514 z:36.702999114990234
- residue LYS has 9 atoms
  - atom: N x: -26.413999557495117 y:1.215000033378601 z:36.52299880981445
  - atom: CA x: -27.770000457763672 y:1.378000020980835 z:36.016998291015625
  - atom: C x: -28.56100082397461 y:2.3529999256134033 z:36.87699890136719
  - atom: O x: -29.222999572753906 y:3.2490000724792248 z:36.358001708984375
  - atom: CB x: -28.500999450683594 y:0.03500000014901161 z:36.00199890136719
  - atom: CG x: -29.937999725341797 y:0.13699999451637268 z:35.51100158691406
  - atom: CD x: -30.6560001373291 y:-1.2070000171661377 z:35.52299880981445
  - atom: CE x: -31.038000106811523 y:-1.6349999904632568 z:36.933998107910156
  - atom: NZ x: -31.767000198364258 y:-2.940999984741211 z:36.91899871826172
- residue LYS has 9 atoms
  - atom: N x: -28.483999252319336 y:2.1700000762939453 z:38.189998626708984
  - atom: CA x: -29.195999145507812 y:3.01200008392334 z:39.145999908447266
  - atom: C x: -28.899999618530273 y:4.515999794006348 z:39.060001373291016
  - atom: O x: -29.618000030517578 y:5.326000213623047 z:39.63399887084961
  - atom: CB x: -28.916000366210938 y:2.5190000534057617 z:40.564998626708984
  - atom: CG x: -29.30299949645996 y:1.0679999589920044 z:40.792999267578125
  - atom: CD x: -29.030000686645508 y:0.6269999742507935 z:42.22800064086914
  - atom: CE x: -29.3700008392334 y:-0.847000002861023 z:42.40999984741211
  - atom: NZ x: -29.13800048828125 y:-1.3320000171661377 z:43.79899978637695
- residue VAL has 7 atoms
  - atom: N x: -27.844999313354492 y:4.8979997634887695 z:38.35499954223633
  - atom: CA x: -27.518999099731445 y:6.308000087738037 z:38.22700119018555
  - atom: C x: -28.152000427246094 y:6.7729997634887695 z:36.92399978637695
  - atom: O x: -28.701000213623047 y:7.872000217437744 z:36.827999114990234
  - atom: CB x: -25.989999771118164 y:6.511000156402588 z:38.16600036621094
  - atom: CG1 x: -25.635000228881836 y:7.994999885559082 z:38.32899856567383
  - atom: CG2 x: -25.32699966430664 y:5.671000003814697 z:39.244998931884766
- residue ASP has 8 atoms
  - atom: N x: -28.059999465942383 y:5.916999816894531 z:35.91699981689453
  - atom: CA x: -28.628000259399414 y:6.202000141143799 z:34.62099838256836
  - atom: C x: -28.538000106811523 y:4.939000129699707 z:33.79800033569336
```

```
  - atom: O x: -27.46500015258789 y:4.364999771118164 z:33.65299987792969
  - atom: CB x: -27.874000549316406 y:7.335000038146973 z:33.917999267578125
  - atom: CG x: -28.533000946044922 y:7.744999885559082 z:32.60300064086914
  - atom: OD1 x: -29.73699951171875 y:8.086999893188477 z:32.624000549316406
  - atom: OD2 x: -27.854000091552734 y:7.7210001945495605 z:31.55299949645996
- residue SER has 6 atoms
  - atom: N x: -29.67799949645996 y:4.511000156402588 z:33.266998291015625
  - atom: CA x: -29.750999450683594 y:3.305000066757202 z:32.45500183105469
  - atom: C x: -29.03700065612793 y:3.424999952316284 z:31.115999221801758
  - atom: O x: -29.08099937438965 y:2.493000030517578 z:30.319000244140625
  - atom: CB x: -31.204999923706055 y:2.9260001182556152 z:32.183998107910156
  - atom: OG x: -31.709999084472656 y:3.6610000133514404 z:31.082000732421875
- residue GLU has 9 atoms
  - atom: N x: -28.38800048828125 y:4.552000045776367 z:30.847999572753906
  - atom: CA x: -27.687999725341797 y:4.685999870300293 z:29.575000762939453
  - atom: C x: -26.202999114990234 y:4.35099983215332 z:29.645999908447266
  - atom: O x: -25.54199981689453 y:4.173999786376953 z:28.621999740600586
  - atom: CB x: -27.898000717163086 y:6.085999965667725 z:29.003999710083008
  - atom: CG x: -29.305999755859375 y:6.269999980926514 z:28.450000762939453
  - atom: CD x: -29.743999481201172 y:5.085999965667725 z:27.591999053955078
  - atom: OE1 x: -29.18400001525879 y:4.901000022888184 z:26.48900032043457
  - atom: OE2 x: -30.64299964904785 y:4.333000183105469 z:28.02899932861328
- residue TYR has 12 atoms
  - atom: N x: -25.69300079345703 y:4.261000156402588 z:30.868000030517578
  - atom: CA x: -24.302000045776367 y:3.9110000133514404 z:31.118000030517578
  - atom: C x: -24.009000778198242 y:2.4860000610351562 z:30.6299991607666
  - atom: O x: -24.80500030517578 y:1.5779999494552612 z:30.854000091552734
  - atom: CB x: -24.033000946044922 y:3.9600000381469727 z:32.612998962402344
  - atom: CG x: -23.69300079345703 y:5.311999797821045 z:33.18600082397461
  - atom: CD1 x: -22.573999404907227 y:6.00600004196167 z:32.75400161743164
  - atom: CD2 x: -24.45400047302246 y:5.866000175476074 z:34.209999084472656
  - atom: CE1 x: -22.222999572753906 y:7.206999778747559 z:33.33100128173828
  - atom: CE2 x: -24.111000061035156 y:7.065999984741211 z:34.78499984741211
  - atom: CZ x: -22.993999481201172 y:7.724999904632568 z:34.340999603271484
  - atom: OH x: -22.635000228881836 y:8.914999961853027 z:34.90700149536133
- residue ILE has 8 atoms
  - atom: N x: -22.875 y:2.2880001068115234 z:29.96299934387207
  - atom: CA x: -22.486000061035156 y:0.9539999961853027 z:29.506000518798828
  - atom: C x: -21.17099952697754 y:0.5669999718666077 z:30.18600082397461
  - atom: O x: -20.097000122070312 y:0.6959999799728394 z:29.594999313354492
  - atom: CB x: -22.257999420166016 y:0.8889999985694885 z:27.974000930786133
  - atom: CG1 x: -23.577999114990234 y:1.0160000324249268 z:27.23200035095215
  - atom: CG2 x: -21.613000869750977 y:-0.45100000500679016 z:27.584999084472656
  - atom: CD1 x: -23.41699981689453 y:0.8740000128746033 z:25.736000061035156
- residue ALA has 5 atoms
  - atom: N x: -21.249000549316406 y:0.10000000149011612 z:31.42799949645996
  - atom: CA x: -20.040000915527344 y:-0.296999990940094 z:32.15399932861328
  - atom: C x: -19.613000869750977 y:-1.6979999542236328 z:31.74799919128418
  - atom: O x: -20.31399917602539 y:-2.6679999828338623 z:32.02799987792969
  - atom: CB x: -20.28700065612793 y:-0.24199999868869781 z:33.65700149536133
- residue THR has 7 atoms
  - atom: N x: -18.457000732421875 y:-1.7899999618530273 z:31.097999572753906
  - atom: CA x: -17.916000366210938 y:-3.066999912261963 z:30.64299964904785
  - atom: C x: -16.632999420166016 y:-3.4679999351501465 z:31.371000289916992
  - atom: O x: -15.590999603271484 y:-2.819999933242798 z:31.200000762939453
  - atom: CB x: -17.61199951171875 y:-3.0190000534057617 z:29.134000778198242
  - atom: OG1 x: -18.81399917602539 y:-2.7090001106262207 z:28.420000076293945
  - atom: CG2 x: -17.062000274658203 y:-4.354000091552734 z:28.64299964904785
- residue VAL has 7 atoms
```

```
  - atom: N x: -16.701000213623047 y:-4.531000137329102 z:32.178001403808594
  - atom: CA x: -15.51200008392334 y:-5.008999824523926 z:32.893001556396484
  - atom: C x: -14.501999855041504 y:-5.539999961853027 z:31.885000228881836
  - atom: O x: -14.866999626159668 y:-6.291999816894531 z:31.0
  - atom: CB x: -15.82800006866455 y:-6.168000221252441 z:33.861000061035156
  - atom: CG1 x: -14.53499984741211 y:-6.660999774932861 z:34.49399948120117
  - atom: CG2 x: -16.80299949645996 y:-5.7170000076293945 z:34.933998107910156
- residue CYS has 6 atoms
  - atom: N x: -13.236000061035156 y:-5.158999919891357 z:32.01599884033203
  - atom: CA x: -12.208000183105469 y:-5.639999866485596 z:31.089000701904297
  - atom: C x: -10.972000122070312 y:-6.2220001220703125 z:31.773000717163086
  - atom: O x: -11.039999961853027 y:-6.798999786376953 z:32.85300064086914
  - atom: CB x: -11.751999855041504 y:-4.51800012588501 z:30.163000106811523
  - atom: SG x: -13.08899974822998 y:-3.6470000743865967 z:29.341999053955078
- residue GLY has 4 atoms
  - atom: N x: -9.829999923706055 y:-6.051000118255615 z:31.124000549316406
  - atom: CA x: -8.604000091552734 y:-6.572999954223633 z:31.680999755859375
  - atom: C x: -8.550999641418457 y:-8.071999549865723 z:31.476999282836914
  - atom: O x: -9.340999603271484 y:-8.630000114440918 z:30.722000122070312
- residue SER has 6 atoms
  - atom: N x: -7.61899995803833 y:-8.729999542236328 z:32.15399932861328
  - atom: CA x: -7.484000205993652 y:-10.166000366210938 z:32.025001525878906
  - atom: C x: -8.803000450134277 y:-10.85200023651123 z:32.362998962402344
  - atom: O x: -8.97700023651123 y:-12.031000137329102 z:32.077999114990234
  - atom: CB x: -6.353000164031982 y:-10.680999755859375 z:32.926998138427734
  - atom: OG x: -6.578000068664551 y:-10.366000175476074 z:34.28799819946289
- residue PHE has 11 atoms
  - atom: N x: -9.734999656677246 y:-10.123000144958496 z:32.96799850463867
  - atom: CA x: -11.02400016784668 y:-10.720999717712402 z:33.28300094604492
  - atom: C x: -11.821999549865723 y:-10.930999755859375 z:32.000999450683594
  - atom: O x: -12.414999961853027 y:-11.984999656677246 z:31.797000885009766
  - atom: CB x: -11.848999977111816 y:-9.84000015258789 z:34.20500183105469
  - atom: CG x: -13.270000457763672 y:-10.277000427246094 z:34.305999755859375
  - atom: CD1 x: -13.663000106811523 y:-11.180000305175781 z:35.28200149536133
  - atom: CD2 x: -14.206999778747559 y:-9.845999717712402 z:33.369998931884766
  - atom: CE1 x: -14.968000411987305 y:-11.656000137329102 z:35.32699966430664
  - atom: CE2 x: -15.517999649047852 y:-10.312000274658203 z:33.39799880981445
  - atom: CZ x: -15.902999877929688 y:-11.220999717712402 z:34.37900161743164
- residue ARG has 11 atoms
  - atom: N x: -11.862000465393066 y:-9.913000106811523 z:31.148000717163086
  - atom: CA x: -12.595000267028809 y:-10.017999649047852 z:29.895000457763672
  - atom: C x: -11.9350004196167 y:-11.093999862670898 z:29.05299949645996
  - atom: O x: -12.581999778747559 y:-11.767000198364258 z:28.249000549316406
  - atom: CB x: -12.574000358581543 y:-8.678999900817871 z:29.16200065612793
  - atom: CG x: -13.28499984741211 y:-8.682000160217285 z:27.829999923706055
  - atom: CD x: -14.185999870300293 y:-7.460000038146973 z:27.68899917602539
  - atom: NE x: -14.734000205993652 y:-7.388000011444092 z:26.347000122070312
  - atom: CZ x: -13.996999740600586 y:-7.210000038146973 z:25.257999420166016
  - atom: NH1 x: -12.6850004196167 y:-7.070000171661377 z:25.35300064086914
  - atom: NH2 x: -14.567999839782715 y:-7.215000152587891 z:24.062999725341797
- residue ARG has 11 atoms
  - atom: N x: -10.635000228881836 y:-11.258999824523926 z:29.259000778198242
  - atom: CA x: -9.885000228881836 y:-12.258999824523926 z:28.534000396728516
  - atom: C x: -10.24899959564209 y:-13.630999565124512 z:29.086000442504883
  - atom: O x: -9.66100025177002 y:-14.633999824523926 z:28.714000701904297
  - atom: CB x: -8.387999534606934 y:-12.006999969482422 z:28.687999725341797
  - atom: CG x: -7.964000225067139 y:-10.605999946594238 z:28.301000595092773
  - atom: CD x: -6.538000106811523 y:-10.583000183105469 z:27.792999267578125
  - atom: NE x: -5.581999778747559 y:-11.0600004196167 z:28.784000396728516
```

```
    - atom: CZ x: -5.169000148773193 y:-10.347000122070312 z:29.82200050354004
    - atom: NH1 x: -5.623000144958496 y:-9.119000434875488 z:30.011999130249023
    - atom: NH2 x: -4.296000003814697 y:-10.855999946594238 z:30.66900062561035
  - residue GLY has 4 atoms
    - atom: N x: -11.241999626159668 y:-13.666000366210938 z:29.965999603271484
    - atom: CA x: -11.654000282287598 y:-14.925000190734863 z:30.55900001525879
    - atom: C x: -10.503999710083008 y:-15.541999816894531 z:31.332000732421875
    - atom: O x: -9.788000106811523 y:-16.39900016784668 z:30.81100082397461
  - residue ALA has 5 atoms
    - atom: N x: -10.319000244140625 y:-15.102999687194824 z:32.573001861572266
    - atom: CA x: -9.23900032043457 y:-15.609000205993652 z:33.4109992980957
    - atom: C x: -9.798999786376953 y:-16.110000610351562 z:34.7239990234375
    - atom: O x: -10.76099967956543 y:-15.543999671936035 z:35.250999450683594
    - atom: CB x: -8.211999893188477 y:-14.51099967956543 z:33.6619987487793
  - residue GLU has 9 atoms
    - atom: N x: -9.192999839782715 y:-17.165000915527344 z:35.25699996948242
    - atom: CA x: -9.666000366210938 y:-17.747999191284180 z:36.512001037597656
    - atom: C x: -9.555000305175781 y:-16.809000015258790 z:37.69599914550781
    - atom: O x: -10.270000457763672 y:-16.947000503540040 z:38.68600082397461
    - atom: CB x: -8.932000160217285 y:-19.059000015258790 z:36.80099868774414
    - atom: CG x: -9.420000076293945 y:-20.181999206542970 z:35.91999816894531
    - atom: CD x: -10.944999694824219 y:-20.224000930786133 z:35.84400177001953
    - atom: OE1 x: -11.602000236511230 y:-20.437999725341797 z:36.89699935913086
    - atom: OE2 x: -11.484999656677246 y:-20.034000396728516 z:34.72700119018555
  - residue SER has 6 atoms
    - atom: N x: -8.654999732971191 y:-15.845999717712402 z:37.58000183105469
    - atom: CA x: -8.458000183105469 y:-14.861000061035156 z:38.625999450683594
    - atom: C x: -7.815999984741211 y:-13.618000030517578 z:38.00999983215332
    - atom: O x: -7.223999977111816 y:-13.684000015258789 z:36.92599868774414
    - atom: CB x: -7.578000068664551 y:-15.440999984741211 z:39.744998931884766
    - atom: OG x: -6.326000213623047 y:-15.878999710083008 z:39.25
  - residue SER has 6 atoms
    - atom: N x: -7.935999870300293 y:-12.489999771118164 z:38.70100021362305
    - atom: CA x: -7.374000072479248 y:-11.253000259399414 z:38.196998596191406
    - atom: C x: -6.690999984741211 y:-10.423000335693360 z:39.26900100708008
    - atom: O x: -7.107999801635742 y:-10.406000137329102 z:40.42399978637695
    - atom: CB x: -8.47599983215332 y:-10.421999931335450 z:37.54199981689453
    - atom: OG x: -9.085000038146973 y:-11.130999565124512 z:36.481998443603516
  - residue GLY has 4 atoms
    - atom: N x: -5.633999824523926 y:-9.727000236511230 z:38.882999420166016
    - atom: CA x: -4.954999923706055 y:-8.890999794006348 z:39.842998504638670
    - atom: C x: -5.926000118255615 y:-7.854000091552734 z:40.379001617431640
    - atom: O x: -5.889999866485596 y:-7.506000041961670 z:41.551998138427734
  - residue ASP has 8 atoms
    - atom: N x: -6.810999870300293 y:-7.369999885559082 z:39.519001007080080
    - atom: CA x: -7.783999919891357 y:-6.349999904632568 z:39.911998748779300
    - atom: C x: -9.005000114440918 y:-6.341000080108643 z:38.992000579833984
    - atom: O x: -9.175000190734863 y:-7.230999946594238 z:38.166999816894530
    - atom: CB x: -7.130000114440918 y:-4.967000007629395 z:39.865001678466800
    - atom: CG x: -6.768000125885010 y:-4.546000003814697 z:38.452999114990234
    - atom: OD1 x: -6.972000122070312 y:-5.339000225067139 z:37.508998870849610
    - atom: OD2 x: -6.277999877929688 y:-3.417000055313110 z:38.278999328613280
  - residue MET has 8 atoms
    - atom: N x: -9.845999717712402 y:-5.321000099182129 z:39.151000976562500
    - atom: CA x: -11.045000076293945 y:-5.147999763488770 z:38.334999084472656
    - atom: C x: -10.956000328063965 y:-3.848999977111816 z:37.555000305175780
    - atom: O x: -10.967000007629395 y:-2.779000043869019 z:38.138000488281250
    - atom: CB x: -12.298999786376953 y:-5.109000205993652 z:39.194000244140625
    - atom: CG x: -13.512999534606934 y:-4.709000110626221 z:38.383998870849610
```

```
  - atom: SD x: -15.081000328063965 y:-5.085999965667725 z:39.16600036621094
  - atom: CE x: -15.109999656677246 y:-6.809000015258789 z:38.875
- residue ASP has 8 atoms
  - atom: N x: -10.873000144958496 y:-3.950000047683716 z:36.23500061035156
  - atom: CA x: -10.781000137329102 y:-2.7780001163482666 z:35.367000579833984
  - atom: C x: -12.133000373840332 y:-2.5320000648498535 z:34.70500183105469
  - atom: O x: -12.66100025177002 y:-3.4049999713897705 z:34.04100036621094
  - atom: CB x: -9.710000038146973 y:-3.0380001068115234 z:34.321998596191406
  - atom: CG x: -8.418999671936035 y:-3.5199999809265137 z:34.94300079345703
  - atom: OD1 x: -7.6579999923706055 y:-2.6649999618530273 z:35.45100021362305
  - atom: OD2 x: -8.175999641418457 y:-4.752999782562256 z:34.95899963378906
- residue VAL has 7 atoms
  - atom: N x: -12.694000244140625 y:-1.3459999561309814 z:34.887001037597656
  - atom: CA x: -13.998000144958496 y:-1.0369999408721924 z:34.314998626708984
  - atom: C x: -13.968999862670898 y:0.0860000029206276 z:33.28300094604492
  - atom: O x: -13.496000289916992 y:1.1820000410079956 z:33.5629997253418
  - atom: CB x: -14.99899959564209 y:-0.6549999713897705 z:35.433998107910156
  - atom: CG1 x: -16.375 y:-0.4050000011920929 z:34.85599899291992
  - atom: CG2 x: -15.0649995803833 y:-1.7660000324249268 z:36.45800018310547
- residue LEU has 8 atoms
  - atom: N x: -14.458000183105469 y:-0.19699999690055847 z:32.082000732421875
  - atom: CA x: -14.527999877929688 y:0.8050000071525574 z:31.0310001373291
  - atom: C x: -15.949999809265137 y:1.3270000219345093 z:31.106000900268555
  - atom: O x: -16.90399932861328 y:0.5569999814033508 z:31.158000946044922
  - atom: CB x: -14.281000137329102 y:0.19300000369548798 z:29.663999557495117
  - atom: CG x: -14.069000244140625 y:1.2020000219345093 z:28.530000686645508
  - atom: CD1 x: -12.979000091552734 y:2.184999942779541 z:28.922000885009766
  - atom: CD2 x: -13.649999618530273 y:0.48399999737739563 z:27.270000457763672
- residue LEU has 8 atoms
  - atom: N x: -16.086999893188477 y:2.640000104904175 z:31.107999801635742
  - atom: CA x: -17.38800048828125 y:3.260999917984009 z:31.239999771118164
  - atom: C x: -17.714000701904297 y:4.26200008392334 z:30.121999740600586
  - atom: O x: -16.847000122070312 y:5.017000198364258 z:29.67799949645996
  - atom: CB x: -17.423999786376953 y:3.9570000171661377 z:32.59700012207031
  - atom: CG x: -18.70199966430664 y:4.685999870300293 z:32.948001861572266
  - atom: CD1 x: -19.889999389648438 y:3.7049999237060547 z:32.92599868774414
  - atom: CD2 x: -18.520000457763672 y:5.311999797821045 z:34.292999267578125
- residue THR has 7 atoms
  - atom: N x: -18.966999053955078 y:4.270999908447266 z:29.67099952697754
  - atom: CA x: -19.382999420166016 y:5.206999778747559 z:28.621999740600586
  - atom: C x: -20.781999588012695 y:5.7170000076293945 z:28.87299919128418
  - atom: O x: -21.530000686645508 y:5.166999816894531 z:29.683000564575195
  - atom: CB x: -19.409000396728516 y:4.563000202178955 z:27.23900032043457
  - atom: OG1 x: -20.46500015258789 y:3.6029999256134033 z:27.19499969482422
  - atom: CG2 x: -18.097000122070312 y:3.871000051498413 z:26.937999725341797
- residue HIS has 10 atoms
  - atom: N x: -21.13599967956543 y:6.7769999504089355 z:28.163999557495117
  - atom: CA x: -22.46500015258789 y:7.359000205993652 z:28.28700065612793
  - atom: C x: -22.768999099731445 y:8.1899995803833 z:27.04800033569336
  - atom: O x: -21.88599967956543 y:8.836999893188477 z:26.488000869750977
  - atom: CB x: -22.566999435424805 y:8.227999687194824 z:29.542999267578125
  - atom: CG x: -23.91200065612793 y:8.859999656677246 z:29.735000610351562
  - atom: ND1 x: -24.489999771118164 y:9.684000015258789 z:28.79199981689453
  - atom: CD2 x: -24.777999877929688 y:8.817000389099121 z:30.777000427246094
  - atom: CE1 x: -25.650999069213867 y:10.123000144958496 z:29.243999481201172
  - atom: NE2 x: -25.850000381469727 y:9.612000465393066 z:30.44700050354004
- residue PRO has 7 atoms
  - atom: N x: -24.027999877929688 y:8.15999984741211 z:26.590999603271484
  - atom: CA x: -24.441999435424805 y:8.918999671936035 z:25.409000396728516
```

```
  - atom: C x: -24.042999267578125 y:10.402999877929688 z:25.429000854492188
  - atom: O x: -23.601999282836914 y:10.946000099182129 z:24.416000366210938
  - atom: CB x: -25.95199966430664 y:8.711999893188477 z:25.395999908447266
  - atom: CG x: -26.070999145507812 y:7.288000106811523 z:25.88599967956543
  - atom: CD x: -25.117000579833984 y:7.2789998054504395 z:27.059999465942383
- residue SER has 6 atoms
  - atom: N x: -24.176000595092773 y:11.045000076293945 z:26.583999633789062
  - atom: CA x: -23.854000091552734 y:12.461999893188477 z:26.711000442504883
  - atom: C x: -22.368000030517578 y:12.763999938964844 z:26.673999786376953
  - atom: O x: -21.957000732421875 y:13.888999938964844 z:26.972999572753906
  - atom: CB x: -24.410999298095703 y:13.015000343322754 z:28.018999099731445
  - atom: OG x: -23.520000457763672 y:12.762999534606934 z:29.091999053955078
- residue PHE has 11 atoms
  - atom: N x: -21.558000564575195 y:11.779999732971191 z:26.30699920654297
  - atom: CA x: -20.12299919128418 y:11.99899959564209 z:26.285999298095703
  - atom: C x: -19.398000717163086 y:11.35200023651123 z:25.132999420166016
  - atom: O x: -19.242000579833984 y:10.137999534606934 z:25.084999084472656
  - atom: CB x: -19.506000518798828 y:11.520999908447266 z:27.60099983215332
  - atom: CG x: -18.007999420166016 y:11.626999855041504 z:27.636999130249023
  - atom: CD1 x: -17.367000579833984 y:12.77299976348877 z:27.167999267578125
  - atom: CD2 x: -17.236000061035156 y:10.593999862670898 z:28.156999588012695
  - atom: CE1 x: -15.989999771118164 y:12.890000343322754 z:27.218000411987305
  - atom: CE2 x: -15.854999542236328 y:10.704000473022461 z:28.211000442504883
  - atom: CZ x: -15.229999542236328 y:11.857000350952148 z:27.739999771118164
- residue THR has 7 atoms
  - atom: N x: -18.934999465942383 y:12.182999610900879 z:24.211000442504883
  - atom: CA x: -18.201000213623047 y:11.710000038146973 z:23.042999267578125
  - atom: C x: -17.07200050354004 y:12.701000213623047 z:22.77400016784668
  - atom: O x: -16.788000106811523 y:13.565999984741211 z:23.59600067138672
  - atom: CB x: -19.110000610351562 y:11.645999908447266 z:21.799999237060547
  - atom: OG1 x: -19.204999923706055 y:12.947999954223633 z:21.209999084472656
  - atom: CG2 x: -20.5049991607666 y:11.180000305175781 z:22.18400001525879
- residue SER has 6 atoms
  - atom: N x: -16.42799949645996 y:12.579000473022461 z:21.625999450683594
  - atom: CA x: -15.345999717712402 y:13.48900032043457 z:21.288000106811523
  - atom: C x: -15.72700023651123 y:14.932999610900879 z:21.63599967956543
  - atom: O x: -15.057999610900879 y:15.576000213623047 z:22.45199966430664
  - atom: CB x: -15.020000457763672 y:13.368000030517578 z:19.798999786376953
  - atom: OG x: -14.71500015258789 y:12.024999618530273 z:19.464000701904297
- residue GLU has 9 atoms
  - atom: N x: -16.80699920654297 y:15.42199993133545 z:21.023000717163086
  - atom: CA x: -17.32900047302246 y:16.783000946044922 z:21.238000869750977
  - atom: C x: -17.645000457763672 y:17.051000595092773 z:22.7189998626709
  - atom: O x: -16.93400001525879 y:17.798999786376953 z:23.395999908447266
  - atom: CB x: -18.590999603271484 y:16.97100067138672 z:20.389999389648438
  - atom: CG x: -19.5049991607666 y:15.744000434875488 z:20.41900062561035
  - atom: CD x: -20.75200080871582 y:15.880999565124512 z:19.555999755859375
  - atom: OE1 x: -20.6200008392334 y:16.08300018310547 z:18.32900047302246
  - atom: OE2 x: -21.8700008392334 y:15.776000022888184 z:20.10700035095215
- residue SER has 6 atoms
  - atom: N x: -18.738000869750977 y:16.454999923706055 z:23.19099998474121
  - atom: CA x: -19.179000854492188 y:16.530000686645508 z:24.583999633789062
  - atom: C x: -19.48200035095215 y:17.86400032043457 z:25.290000915527344
  - atom: O x: -20.475000381469727 y:17.961999893188477 z:26.020999908447266
  - atom: CB x: -18.18199920654297 y:15.746000289916992 z:25.451000213623047
  - atom: OG x: -16.867000579833984 y:16.288999557495117 z:25.367000579833984
- residue THR has 7 atoms
  - atom: N x: -18.636999130249023 y:18.87299919128418 z:25.0939998626709
  - atom: CA x: -18.801000595092773 y:20.159000396728516 z:25.781999588012695
```

```
  - atom: C x: -18.413999557495117 y:19.841999053955078 z:27.225000381469727
  - atom: O x: -18.558000564575195 y:20.67099952697754 z:28.12700080871582
  - atom: CB x: -20.27400016784668 y:20.67799949645996 z:25.78700065612793
  - atom: OG1 x: -20.80299949645996 y:20.674999237060547 z:24.45400047302246
  - atom: CG2 x: -20.33799934387207 y:22.104999542236328 z:26.34600067138672
- residue LYS has 9 atoms
  - atom: N x: -17.930999755859375 y:18.615999221801758 z:27.42099952697754
  - atom: CA x: -17.509000778198242 y:18.101999282836914 z:28.725000381469727
  - atom: C x: -18.625 y:18.062000274658203 z:29.767000198364258
  - atom: O x: -19.027000427246094 y:19.090999603271484 z:30.31800079345703
  - atom: CB x: -16.312000274658203 y:18.898000717163086 z:29.25200080871582
  - atom: CG x: -14.979999542236328 y:18.506999969482422 z:28.608999252319336
  - atom: CD x: -14.986000061035156 y:18.663999557495117 z:27.079999923706055
  - atom: CE x: -13.618000030517578 y:18.32900047302246 z:26.48699951171875
  - atom: NZ x: -13.541999816894531 y:18.54800033569336 z:25.016000747680664
- residue GLN has 9 atoms
  - atom: N x: -19.118999481201172 y:16.850000381469727 z:30.014999389648438
  - atom: CA x: -20.18400001525879 y:16.606000900268555 z:30.979000091552734
  - atom: C x: -19.53700065612793 y:16.427000045776367 z:32.35100173950195
  - atom: O x: -18.87299919128418 y:15.42199993133545 z:32.61600112915039
  - atom: CB x: -20.966999053955078 y:15.35099983215332 z:30.586000442504883
  - atom: CG x: -22.29400062561035 y:15.206999778747559 z:31.299999237060547
  - atom: CD x: -23.20400047302246 y:16.392000198364258 z:31.062000274658203
  - atom: OE1 x: -23.5310001373291 y:16.709999084472656 z:29.92300033569336
  - atom: NE2 x: -23.618000030517578 y:17.05500030517578 z:32.137001037597656
- residue PRO has 7 atoms
  - atom: N x: -19.719999313354492 y:17.417999267578125 z:33.23699951171875
  - atom: CA x: -19.187000274658203 y:17.464000701904297 z:34.60200119018555
  - atom: C x: -19.802000045776367 y:16.465999603271484 z:35.58100128173828
  - atom: O x: -20.979000091552734 y:16.569000244140625 z:35.93000030517578
  - atom: CB x: -19.45400047302246 y:18.90399932861328 z:35.012001037597656
  - atom: CG x: -20.766000747680664 y:19.173999786376953 z:34.35300064086914
  - atom: CD x: -20.55299949645996 y:18.604999542236328 z:32.970001220703125
- residue LYS has 9 atoms
  - atom: N x: -18.98699951171875 y:15.51099967956543 z:36.02000045776367
  - atom: CA x: -19.39900016784668 y:14.482000350952148 z:36.97200012207031
  - atom: C x: -20.075000762939453 y:13.255000114440918 z:36.362998962402344
  - atom: O x: -20.964000701904297 y:12.654999732971191 z:36.9739990234375
  - atom: CB x: -20.297000885009766 y:15.085000038146973 z:38.06100082397461
  - atom: CG x: -19.63599967956543 y:16.225000381469727 z:38.82099914550781
  - atom: CD x: -20.55299949645996 y:16.827999114990234 z:39.87300109863281
  - atom: CE x: -19.891000747680664 y:18.023000717163086 z:40.55099868774414
  - atom: NZ x: -20.739999771118164 y:18.586000442504883 z:41.63999938964844
- residue LEU has 8 atoms
  - atom: N x: -19.663999557495117 y:12.876999855041504 z:35.15700149536133
  - atom: CA x: -20.231000900268555 y:11.685999870300293 z:34.54800033569336
  - atom: C x: -19.660999298095703 y:10.541000366210938 z:35.36899948120117
  - atom: O x: -20.357999801635742 y:9.57800006866455 z:35.69499969482422
  - atom: CB x: -19.77899932861328 y:11.529000282287598 z:33.09600067138672
  - atom: CG x: -20.839000701904297 y:11.605999946594238 z:31.996999740600586
  - atom: CD1 x: -20.340999603271484 y:10.803999900817871 z:30.812000274658203
  - atom: CD2 x: -22.187000274658203 y:11.057000160217285 z:32.479000091552734
- residue LEU has 8 atoms
  - atom: N x: -18.3799991607666 y:10.666000366210938 z:35.709999084472656
  - atom: CA x: -17.700000762939453 y:9.656000137329102 z:36.494998931884766
  - atom: C x: -17.955999374389648 y:9.869999885559082 z:37.96799850463867
  - atom: O x: -18.058000564575195 y:8.918999671936035 z:38.729000091552734
  - atom: CB x: -16.20400047302246 y:9.694000244140625 z:36.222999572753906
  - atom: CG x: -15.364999771118164 y:8.619000434875488 z:36.91600036621094
```

```
  - atom: CD1 x: -15.930000305175781 y:7.230999946594238 z:36.617000579833984
  - atom: CD2 x: -13.913000106811523 y:8.743000030517578 z:36.44200134277344
- residue HIS has 10 atoms
  - atom: N x: -18.06999969482422 y:11.121999740600586 z:38.37699890136719
  - atom: CA x: -18.312000274658203 y:11.401000022888184 z:39.779998779296875
  - atom: C x: -19.5310001373291 y:10.654999732971191 z:40.31700134277344
  - atom: O x: -19.44099998474121 y:9.95300006866455 z:41.316001892089844
  - atom: CB x: -18.496999740600586 y:12.901000022888184 z:40.000999450683594
  - atom: CG x: -18.812000274658203 y:13.258999824523926 z:41.41999816894531
  - atom: ND1 x: -17.89900016784668 y:13.11299991607666 z:42.44200134277344
  - atom: CD2 x: -19.959999084472656 y:13.682999610900879 z:41.99700164794922
  - atom: CE1 x: -18.474000930786133 y:13.425000190734863 z:43.59000015258789
  - atom: NE2 x: -19.725000381469727 y:13.77400016784668 z:43.347999572753906
- residue GLN has 9 atoms
  - atom: N x: -20.66900062561035 y:10.812999725341797 z:39.64699935913086
  - atom: CA x: -21.92300033569336 y:10.175999641418457 z:40.055999755859375
  - atom: C x: -21.83099937438965 y:8.678000450134277 z:40.29800033569336
  - atom: O x: -22.37700080871582 y:8.17300033569336 z:41.268001556396484
  - atom: CB x: -23.016000747680664 y:10.432000160217285 z:39.018001556396484
  - atom: CG x: -23.368000030517578 y:11.890999794006348 z:38.82099914550781
  - atom: CD x: -24.46299934387207 y:12.07699966430664 z:37.78099822998047
  - atom: OE1 x: -25.597999572753906 y:11.62600040435791 z:37.9640007019043
  - atom: NE2 x: -24.124000549316406 y:12.73799991607666 z:36.67900085449219
- residue VAL has 7 atoms
  - atom: N x: -21.155000686645508 y:7.968999862670898 z:39.409000396728516
  - atom: CA x: -21.006000518798828 y:6.53000020980835 z:39.54600143432617
  - atom: C x: -20.003000259399414 y:6.197000026702881 z:40.6510009765625
  - atom: O x: -20.218000411987305 y:5.28000020980835 z:41.4379997253418
  - atom: CB x: -20.562999725341797 y:5.888000011444092 z:38.20899963378906
  - atom: CG1 x: -20.05699920654297 y:4.473999977111816 z:38.441001892089844
  - atom: CG2 x: -21.73699951171875 y:5.868000030517578 z:37.24599838256836
- residue VAL has 7 atoms
  - atom: N x: -18.913999557495117 y:6.947000026702881 z:40.720001220703125
  - atom: CA x: -17.933000564575195 y:6.690000057220459 z:41.75199890136719
  - atom: C x: -18.690000534057617 y:6.857999801635742 z:43.042999267578125
  - atom: O x: -18.660999298095703 y:6.008999824523926 z:43.92399978637695
  - atom: CB x: -16.781999588012695 y:7.709000110626221 z:41.72200012207031
  - atom: CG1 x: -15.850000381469727 y:7.456999778747559 z:42.880001068115234
  - atom: CG2 x: -16.020000457763672 y:7.6020002365112305 z:40.41999816894531
- residue GLU has 9 atoms
  - atom: N x: -19.398000717163086 y:7.968999862670898 z:43.132999420166016
  - atom: CA x: -20.167999267578125 y:8.276000022888184 z:44.319000244140625
  - atom: C x: -21.114999771118164 y:7.140999794006348 z:44.65999984741211
  - atom: O x: -21.065000534057617 y:6.604000091552734 z:45.75699996948242
  - atom: CB x: -20.961000442504883 y:9.5600004196167 z:44.09700012207031
  - atom: CG x: -21.607999801635742 y:10.109999656677246 z:45.33399963378906
  - atom: CD x: -22.40399932861328 y:11.36299991607666 z:45.042999267578125
  - atom: OE1 x: -23.4689998626709 y:11.255999565124512 z:44.395999908447266
  - atom: OE2 x: -21.961999893188477 y:12.458000183105469 z:45.452999114990234
- residue GLN has 9 atoms
  - atom: N x: -21.9689998626709 y:6.763000011444092 z:43.715999603271484
  - atom: CA x: -22.940000534057617 y:5.702000141143799 z:43.96500015258789
  - atom: C x: -22.292999267578125 y:4.389999866485596 z:44.44900131225586
  - atom: O x: -22.858999252319336 y:3.684000015258789 z:45.28799819946289
  - atom: CB x: -23.791000366210938 y:5.4720001220703125 z:42.70100021362305
  - atom: CG x: -24.867000579833984 y:4.372000217437744 z:42.79600143432617
  - atom: CD x: -26.01300048828125 y:4.7129998207092285 z:43.7400016784668
  - atom: OE1 x: -25.9689998626709 y:5.706999778747559 z:44.46099853515625
  - atom: NE2 x: -27.04400062561035 y:3.874000072479248 z:43.742000579833984
```

```
- residue LEU has 8 atoms
  - atom: N x: -21.11199951171875 y:4.065000057220459 z:43.93299865722656
  - atom: CA x: -20.437000274658203 y:2.8420000076293945 z:44.3489990234375
  - atom: C x: -19.951000213623047 y:2.9660000801086426 z:45.7859992980957
  - atom: O x: -19.659000396728516 y:1.9639999866485596 z:46.42499923706055
  - atom: CB x: -19.267000198364258 y:2.5199999809265137 z:43.4109992980957
  - atom: CG x: -19.673999786376953 y:1.9570000171661377 z:42.04199981689453
  - atom: CD1 x: -18.472999572753906 y:1.777999997138977 z:41.165000915527344
  - atom: CD2 x: -20.384000778198242 y:0.628000020980835 z:42.23400115966797
- residue GLN has 9 atoms
  - atom: N x: -19.868000030517578 y:4.198999881744385 z:46.28799819946289
  - atom: CA x: -19.44499969482422 y:4.461999893188477 z:47.66899871826172
  - atom: C x: -20.687999725341797 y:4.511000156402588 z:48.558998107910156
  - atom: O x: -20.60099983215332 y:4.388999938964844 z:49.7869987487793
  - atom: CB x: -18.71500015258789 y:5.809000015258789 z:47.7859992980957
  - atom: CG x: -17.40399932861328 y:5.919000148773193 z:47.03499984741211
  - atom: CD x: -16.65999984741211 y:7.2210001945495605 z:47.33300018310547
  - atom: OE1 x: -17.259000778198242 y:8.29800033569336 z:47.375999450683594
  - atom: NE2 x: -15.347000122070312 y:7.123000144958496 z:47.527000427246094
- residue LYS has 9 atoms
  - atom: N x: -21.84000015258789 y:4.710000038146973 z:47.92300033569336
  - atom: CA x: -23.118000030517578 y:4.790999889373779 z:48.62099838256836
  - atom: C x: -23.485000610351562 y:3.427999973297119 z:49.20899963378906
  - atom: O x: -23.861000061035156 y:3.321000099182129 z:50.37699890136719
  - atom: CB x: -24.209999084472656 y:5.244999885559082 z:47.6510009765625
  - atom: CG x: -25.493000030517578 y:5.758999824523926 z:48.308998107910156
  - atom: CD x: -25.284000396728516 y:7.151000022888184 z:48.91600036621094
  - atom: CE x: -26.61400032043457 y:7.869999885559082 z:49.20199966430664
  - atom: NZ x: -27.492000579833984 y:7.140999794006348 z:50.17300033569336
- residue VAL has 7 atoms
  - atom: N x: -23.381000518798828 y:2.38700008392334 z:48.391998291015625
  - atom: CA x: -23.690000534057617 y:1.0379999876022339 z:48.84199905395508
  - atom: C x: -22.56100082397461 y:0.5149999856948853 z:49.72800064086914
  - atom: O x: -22.729000091552734 y:-0.4690000116825104 z:50.45100021362305
  - atom: CB x: -23.85300064086914 y:0.07800000160932541 z:47.645999908447266
  - atom: CG1 x: -25.08300018310547 y:0.44999998807907104 z:46.845001220703125
  - atom: CG2 x: -22.618999481201172 y:0.13099999725818634 z:46.76300048828125
- residue HIS has 10 atoms
  - atom: N x: -21.413999557495117 y:1.187999963760376 z:49.66400146484375
  - atom: CA x: -20.229000091552734 y:0.8109999895095825 z:50.428001403808594
  - atom: C x: -19.409000396728516 y:-0.27000001072883606 z:49.709999084472656
  - atom: O x: -18.84000015258789 y:-1.152999997138977 z:50.35200119018555
  - atom: CB x: -20.610000610351562 y:0.33399999141693115 z:51.8390007019043
  - atom: CG x: -21.040000915527344 y:1.437000036239624 z:52.7599983215332
  - atom: ND1 x: -22.143999099731445 y:1.3350000381469727 z:53.582000732421875
  - atom: CD2 x: -20.5 y:2.6549999713897705 z:53.0099983215332
  - atom: CE1 x: -22.266000747680664 y:2.440000057220459 z:54.297000885009766
  - atom: NE2 x: -21.281999588012695 y:3.257999897003174 z:53.96900177001953
- residue PHE has 11 atoms
  - atom: N x: -19.360000610351562 y:-0.20000000298023224 z:48.37799835205078
  - atom: CA x: -18.570999145507812 y:-1.1480000019073486 z:47.59000015258789
  - atom: C x: -17.174999237060547 y:-0.5789999961853027 z:47.39500045776367
  - atom: O x: -16.20800018310547 y:-1.3270000219345093 z:47.3650016784668
  - atom: CB x: -19.187999725341797 y:-1.3940000534057617 z:46.220001220703125
  - atom: CG x: -18.344999313354492 y:-2.2720000743865967 z:45.32500076293945
  - atom: CD1 x: -18.054000854492188 y:-3.5820000171661377 z:45.685001373291016
  - atom: CD2 x: -17.84600067138672 y:-1.784999966621399 z:44.11899948120117
  - atom: CE1 x: -17.288999557495117 y:-4.38700008392334 z:44.86800003051758
  - atom: CE2 x: -17.076000213623047 y:-2.5869998931884766 z:43.29399871826172
```

```
  - atom: CZ x: -16.79800033569336 y:-3.8910000324249268 z:43.67100143432617
- residue ILE has 8 atoms
  - atom: N x: -17.072999954223633 y:0.7419999837875366 z:47.24300003051758
  - atom: CA x: -15.77400016784668 y:1.3860000371932983 z:47.08300018310547
  - atom: C x: -15.430999755859375 y:1.9910000562667847 z:48.42300033569336
  - atom: O x: -16.05900001525879 y:2.953000068664551 z:48.85200119018555
  - atom: CB x: -15.774999618530273 y:2.503000020980835 z:45.97999954223633
  - atom: CG1 x: -15.883000373840332 y:1.871000051498413 z:44.58399963378906
  - atom: CG2 x: -14.479000091552734 y:3.2899999618530273 z:46.02199935913086
  - atom: CD1 x: -15.685999870300293 y:2.8510000705718994 z:43.44300079345703
- residue THR has 7 atoms
  - atom: N x: -14.427000045776367 y:1.4140000343322754 z:49.073001861572266
  - atom: CA x: -13.984999656677246 y:1.8559999465942383 z:50.388999938964844
  - atom: C x: -12.88599967956543 y:2.9140000343322754 z:50.402000427246094
  - atom: O x: -12.743000030517578 y:3.63100004196167 z:51.39099884033203
  - atom: CB x: -13.479000091552734 y:0.6610000133514404 z:51.25299835205078
  - atom: OG1 x: -12.404000282287598 y:-0.0020000000949949026 z:50.58300018310547
  - atom: CG2 x: -14.581000328063965 y:-0.32600000500679016 z:51.51499938964844
- residue ASP has 8 atoms
  - atom: N x: -12.095000267028809 y:3.01200008392334 z:49.3390007019043
  - atom: CA x: -11.015000343322754 y:4.000999927520752 z:49.31800079345703
  - atom: C x: -10.654000282287598 y:4.551000118255615 z:47.93299865722656
  - atom: O x: -10.78600025177002 y:3.868000030517578 z:46.915000915527344
  - atom: CB x: -9.744000434875488 y:3.4130001068115234 z:49.9630012512207
  - atom: CG x: -10.015000343322754 y:2.7699999809265137 z:51.31800079345703
  - atom: OD1 x: -10.675000190734863 y:1.7079999446868896 z:51.349998474121094
  - atom: OD2 x: -9.574000358581543 y:3.3239998817443848 z:52.35100173950195
- residue THR has 7 atoms
  - atom: N x: -10.157999992370605 y:5.7870001792907715 z:47.93199920654297
  - atom: CA x: -9.746999740600586 y:6.508999824523926 z:46.73899841308594
  - atom: C x: -8.234999656677246 y:6.723999977111816 z:46.68000030517578
  - atom: O x: -7.664999961853027 y:7.307000160217285 z:47.59000015258789
  - atom: CB x: -10.407999992370605 y:7.89300012588501 z:46.715999603271484
  - atom: OG1 x: -11.79800033569336 y:7.748000144958496 z:46.40999984741211
  - atom: CG2 x: -9.73799991607666 y:8.803000450134277 z:45.6879997253418
- residue LEU has 8 atoms
  - atom: N x: -7.583000183105469 y:6.265999794006348 z:45.612998962402344
  - atom: CA x: -6.146999835968018 y:6.489999771118164 z:45.47800064086914
  - atom: C x: -5.991000175476074 y:7.702000141143799 z:44.57099914550781
  - atom: O x: -5.010000228881836 y:8.442000389099121 z:44.667999267578125
  - atom: CB x: -5.427999973297119 y:5.298999786376953 z:44.83100128173828
  - atom: CG x: -5.5980000495910645 y:3.871999979019165 z:45.35599899291992
  - atom: CD1 x: -4.579999923706055 y:2.9730000495910645 z:44.696998596191406
  - atom: CD2 x: -5.414999961853027 y:3.8259999752044678 z:46.847999572753906
- residue SER has 6 atoms
  - atom: N x: -6.9710001945495605 y:7.901000022888184 z:43.694000244140625
  - atom: CA x: -6.947999954223633 y:9.012999534606934 z:42.74399948120117
  - atom: C x: -8.258999824523926 y:9.112000465393066 z:41.98500061035156
  - atom: O x: -8.848999977111816 y:8.095000267028809 z:41.61399841308594
  - atom: CB x: -5.817999839782715 y:8.831999778747559 z:41.74100112915039
  - atom: OG x: -6.002999782562256 y:7.638999938964844 z:41.01300048828125
- residue LYS has 9 atoms
  - atom: N x: -8.70300006866455 y:10.340999603271484 z:41.74800109863281
  - atom: CA x: -9.954000473022461 y:10.569999694824219 z:41.04100036621094
  - atom: C x: -9.918000221252441 y:11.857999801635742 z:40.21900177001953
  - atom: O x: -9.762999534606934 y:12.961999893188477 z:40.74599838256836
  - atom: CB x: -11.107999801635742 y:10.605999946594238 z:42.04499816894531
  - atom: CG x: -12.496000289916992 y:10.840999603271484 z:41.47100067138672
  - atom: CD x: -13.52299976348877 y:10.961000442504883 z:42.61000061035156
```

```
  - atom: CE x: -14.862000465393066 y:11.560999870300293 z:42.15599822998047
  - atom: NZ x: -14.739999771118164 y:12.946999549865723 z:41.57699966430664
- residue GLY has 4 atoms
  - atom: N x: -10.03600025177002 y:11.694000244140625 z:38.90800094604492
  - atom: CA x: -10.041999816894531 y:12.824000358581543 z:38.00600051879883
  - atom: C x: -11.36400032043457 y:12.800000190734863 z:37.26900100708008
  - atom: O x: -12.270999908447266 y:12.064000129699707 z:37.66899871826172
- residue GLU has 9 atoms
  - atom: N x: -11.475000381469727 y:13.581999778747559 z:36.19599914550781
  - atom: CA x: -12.708999633789062 y:13.637999534606934 z:35.41299819946289
  - atom: C x: -12.991000175476074 y:12.404999732971191 z:34.569000244140625
  - atom: O x: -14.145000457763672 y:12.163000106811523 z:34.23899841308594
  - atom: CB x: -12.729000091552734 y:14.871000289916992 z:34.49399948120117
  - atom: CG x: -12.994000434875488 y:16.202999114990234 z:35.20000076293945
  - atom: CD x: -14.388999938964844 y:16.284000396728516 z:35.803001403808594
  - atom: OE1 x: -14.821999549865723 y:15.300000190734863 z:36.452999114990234
  - atom: OE2 x: -15.04800033569336 y:17.33300018310547 z:35.64099884033203
- residue THR has 7 atoms
  - atom: N x: -11.963000297546387 y:11.63700008392334 z:34.209999084472656
  - atom: CA x: -12.175000190734863 y:10.432999610900879 z:33.39699935913086
  - atom: C x: -11.51200008392334 y:9.157999992370605 z:33.95399856567383
  - atom: O x: -11.673999786376953 y:8.072999954223633 z:33.38600158691406
  - atom: CB x: -11.6850004196167 y:10.630000114440918 z:31.908000946044922
  - atom: OG1 x: -10.255999565124512 y:10.644000053405762 z:31.856000900268555
  - atom: CG2 x: -12.185999870300293 y:11.944999694824219 z:31.336999893188477
- residue LYS has 9 atoms
  - atom: N x: -10.781000137329102 y:9.277000427246094 z:35.06399917602539
  - atom: CA x: -10.095999717712402 y:8.119999885559082 z:35.63800048828125
  - atom: C x: -10.126999855041504 y:8.034000396728516 z:37.154998779296875
  - atom: O x: -9.835000038146973 y:9.008000373840332 z:37.849998474121094
  - atom: CB x: -8.642999649047852 y:8.10099983215332 z:35.18199920654297
  - atom: CG x: -7.811999797821045 y:7.0980000495910645 z:35.92300033569336
  - atom: CD x: -6.353000164031982 y:7.2220001220703125 z:35.58100128173828
  - atom: CE x: -5.540999889373779 y:6.206999778747559 z:36.380001068115234
  - atom: NZ x: -4.105000019073486 y:6.232999801635742 z:36.0
- residue PHE has 11 atoms
  - atom: N x: -10.454000473022461 y:6.8460001945495605 z:37.65800094604492
  - atom: CA x: -10.522000312805176 y:6.5879998207092285 z:39.090999603271484
  - atom: C x: -9.812000274658203 y:5.298999786376953 z:39.49700164794922
  - atom: O x: -10.163999557495117 y:4.2129998207092285 z:39.051998138427734
  - atom: CB x: -11.984000205993652 y:6.501999855041504 z:39.547000885009766
  - atom: CG x: -12.156000137329102 y:6.059999942779541 z:40.986000061035156
  - atom: CD1 x: -11.930999755859375 y:6.936999797821045 z:42.0359992980957
  - atom: CD2 x: -12.531000137329102 y:4.756999969482422 z:41.2859992980957
  - atom: CE1 x: -12.079000473022461 y:6.519000053405762 z:43.36600112915039
  - atom: CE2 x: -12.678999900817871 y:4.329999923706055 z:42.61800003051758
  - atom: CZ x: -12.45300006866455 y:5.210999965667725 z:43.652000427246094
- residue MET has 8 atoms
  - atom: N x: -8.805999755859375 y:5.428999900817871 z:40.35100173950195
  - atom: CA x: -8.081999778747559 y:4.275000095367432 z:40.875
  - atom: C x: -8.491999626159668 y:4.190000057220459 z:42.34199905395508
  - atom: O x: -8.345999717712402 y:5.159999847412109 z:43.09000015258789
  - atom: CB x: -6.578999996185303 y:4.486000061035156 z:40.770000457763672
  - atom: CG x: -6.034999847412109 y:4.218999862670898 z:39.39500045776367
  - atom: SD x: -4.295000076293945 y:4.629000186920166 z:39.29399871826172
  - atom: CE x: -3.693000078201294 y:3.8350000381469727 z:40.743000030517578
- residue GLY has 4 atoms
  - atom: N x: -9.012000083923340 y:3.0390000343322754 z:42.7599983215332
  - atom: CA x: -9.442999839782715 y:2.933000087738037 z:44.13600158691406
```

```
  - atom: C x: -9.482999801635742 y:1.5640000104904175 z:44.742000579833984
  - atom: O x: -8.850000381469727 y:0.6309999823570251 z:44.27000045776367
- residue VAL has 7 atoms
  - atom: N x: -10.26200008392334 y:1.4559999704360962 z:45.80400085449219
  - atom: CA x: -10.385000228881836 y:0.21899999678134918 z:46.54199981689453
  - atom: C x: -11.847000122070312 y:-0.14499999582767487 z:46.7239990234375
  - atom: O x: -12.666000366210938 y:0.7099999785423279 z:47.07899856567383
  - atom: CB x: -9.720000267028809 y:0.36800000071525574 z:47.94599914550781
  - atom: CG1 x: -9.960000038146973 y:-0.8809999823570251 z:48.790000915527344
  - atom: CG2 x: -8.218000411987305 y:0.6430000066757202 z:47.7869987487793
- residue CYS has 6 atoms
  - atom: N x: -12.168999671936035 y:-1.4140000343322754 z:46.46799850463867
  - atom: CA x: -13.529000282287598 y:-1.909000039100647 z:46.63399887084961
  - atom: C x: -13.513999938964844 y:-3.187000036239624 z:47.46699905395508
  - atom: O x: -12.456000328063965 y:-3.759000062942505 z:47.72200012207031
  - atom: CB x: -14.203000068664455 y:-2.1549999713897705 z:45.277000427246094
  - atom: SG x: -13.541000366210938 y:-3.48799991607666 z:44.275001525878906
- residue GLN has 9 atoms
  - atom: N x: -14.690999984741211 y:-3.625999927520752 z:47.900001525878906
  - atom: CA x: -14.809000015258789 y:-4.818999767303467 z:48.72600173950195
  - atom: C x: -16.209999084472656 y:-5.434999942779541 z:48.64799880981445
  - atom: O x: -17.215999603271484 y:-4.756999969482422 z:48.869998931884766
  - atom: CB x: -14.467000007629395 y:-4.459000110626221 z:50.178001403808594
  - atom: CG x: -14.763999938964844 y:-5.531000137329102 z:51.202999114990234
  - atom: CD x: -14.333999633789062 y:-5.130000114440918 z:52.604000091552734
  - atom: OE1 x: -14.741000175476074 y:-4.0879998207092285 z:53.11800003051758
  - atom: NE2 x: -13.508000373840332 y:-5.961999893188477 z:53.231998443603516
- residue LEU has 8 atoms
  - atom: N x: -16.267000198364258 y:-6.71999979019165 z:48.31399917602539
  - atom: CA x: -17.533000946044922 y:-7.429999828338623 z:48.22800064086914
  - atom: C x: -18.14900016784668 y:-7.479000091552734 z:49.61899948120117
  - atom: O x: -17.454999923706055 y:-7.729000091552734 z:50.595001220703125
  - atom: CB x: -17.297000885009766 y:-8.852999687194824 z:47.722999572753906
  - atom: CG x: -16.979000091552734 y:-9.00100040435791 z:46.236000061035156
  - atom: CD1 x: -16.43600082397461 y:-10.39799976348877 z:45.946998596191406
  - atom: CD2 x: -18.249000549316406 y:-8.720000267028809 z:45.43000030517578
- residue PRO has 7 atoms
  - atom: N x: -19.459999084472656 y:-7.241000175476074 z:49.733001708984375
  - atom: CA x: -20.0939998626709 y:-7.28000020980835 z:51.053001403808594
  - atom: C x: -20.091999053955078 y:-8.692999839782715 z:51.62099838256836
  - atom: O x: -20.04400062561035 y:-9.668000221252441 z:50.87300109863281
  - atom: CB x: -21.49799919128418 y:-6.7829999923706055 z:50.763999938964844
  - atom: CG x: -21.75200080871582 y:-7.359000205993652 z:49.42300033569336
  - atom: CD x: -20.466999053955078 y:-7.035999774932861 z:48.683998107910156
- residue SER has 6 atoms
  - atom: N x: -20.14299964904785 y:-8.800000190734863 z:52.944000244140625
  - atom: CA x: -20.150999069213867 y:-10.104999542236328 z:53.59000015258789
  - atom: C x: -21.516000747680664 y:-10.470999717712402 z:54.14400100708008
  - atom: O x: -22.22800064086914 y:-9.626999855041504 z:54.69300079345703
  - atom: CB x: -19.11400032043457 y:-10.157999992370605 z:54.71099853515625
  - atom: OG x: -17.80299949645996 y:-10.243000030517578 z:54.176998138427734
- residue LYS has 9 atoms
  - atom: N x: -21.854999542236328 y:-11.748000144958496 z:53.99100112915039
  - atom: CA x: -23.124000549316406 y:-12.317000389099121 z:54.44300079345703
  - atom: C x: -23.631999969482422 y:-11.708000183105469 z:55.74700164794922
  - atom: O x: -24.525999069213867 y:-10.859999656677246 z:55.74100112915039
  - atom: CB x: -22.985000610351562 y:-13.836000442504883 z:54.61800003051758
  - atom: CG x: -22.371000289916992 y:-14.569000244140625 z:53.42399978637695
  - atom: CD x: -20.875999450683594 y:-14.298999786376953 z:53.28200149536133
```

```
  - atom: CE x: -20.301000595092773 y:-14.970999717712402 z:52.04600143432617
  - atom: NZ x: -18.827999114990234 y:-14.781999588012695 z:51.97600173950195
- residue ASN has 8 atoms
  - atom: N x: -23.066999435424805 y:-12.161999702453613 z:56.861000061035156
  - atom: CA x: -23.440000534057617 y:-11.675999641418457 z:58.18600082397461
  - atom: C x: -22.19700050354004 y:-11.550000190734863 z:59.04600143432617
  - atom: O x: -21.503999710083008 y:-10.529999732971191 z:59.018001556396484
  - atom: CB x: -24.42300033569336 y:-12.637999534606934 z:58.86199951171875
  - atom: CG x: -25.77899932861328 y:-12.663999557495117 z:58.180999755859375
  - atom: OD1 x: -25.885000228881836 y:-12.98799991607666 z:56.99300003051758
  - atom: ND2 x: -26.826000213623047 y:-12.321999549865723 z:58.93000030517578
- residue ASP has 8 atoms
  - atom: N x: -21.91200065612793 y:-12.593999862670898 z:59.8120002746582
  - atom: CA x: -20.743000030517578 y:-12.5649995803833 z:60.667999267578125
  - atom: C x: -19.663999557495117 y:-13.559000015258789 z:60.22200012207031
  - atom: O x: -19.101999282836914 y:-14.295999526977539 z:61.034000396728516
  - atom: CB x: -21.1560001373291 y:-12.817000389099121 z:62.12099838256836
  - atom: CG x: -20.41699981689453 y:-11.913000106811523 z:63.0989990234375
  - atom: OD1 x: -20.413000106811523 y:-10.680999755859375 z:62.880001068115234
  - atom: OD2 x: -19.844999313354492 y:-12.430999755859375 z:64.08399963378906
- residue GLU has 9 atoms
  - atom: N x: -19.38800048828125 y:-13.57699966430664 z:58.917999267578125
  - atom: CA x: -18.354000091552734 y:-14.447999954223633 z:58.375
  - atom: C x: -17.06800079345703 y:-13.638999938964844 z:58.441001892089844
  - atom: O x: -16.729000091552734 y:-13.08899974822998 z:59.4900016784668
  - atom: CB x: -18.63800048828125 y:-14.824999809265137 z:56.91699981689453
  - atom: CG x: -19.93899917602539 y:-15.576000213623047 z:56.67300033569336
  - atom: CD x: -21.152000427246094 y:-14.701000213623047 z:56.88600158691406
  - atom: OE1 x: -21.05900001525879 y:-13.493000030517578 z:56.584999084472656
  - atom: OE2 x: -22.195999145507812 y:-15.218000411987305 z:57.3380012512207
- residue LYS has 9 atoms
  - atom: N x: -16.361000061035156 y:-13.54699993133545 z:57.319000244140625
  - atom: CA x: -15.116000175476074 y:-12.793999671936035 z:57.284000396728516
  - atom: C x: -15.039999961853027 y:-11.869000434875488 z:56.07699966430664
  - atom: O x: -15.192000389099121 y:-12.310999870300293 z:54.941001892089844
  - atom: CB x: -13.92300033569336 y:-13.751999855041504 z:57.27399826049805
  - atom: CG x: -12.5600004196167 y:-13.060999870300293 z:57.27399826049805
  - atom: CD x: -11.418999671936035 y:-14.069999694824219 z:57.36899948120117
  - atom: CE x: -11.454000473022461 y:-15.064000129699707 z:56.21500015258789
  - atom: NZ x: -10.368000030517578 y:-16.07900047302246 z:56.319000244140625
- residue GLU has 9 atoms
  - atom: N x: -14.805000305175781 y:-10.585000038146973 z:56.340999603271484
  - atom: CA x: -14.687000274658203 y:-9.57800006866455 z:55.292999267578125
  - atom: C x: -13.902999877929688 y:-10.180000305175781 z:54.125999450683594
  - atom: O x: -12.942000389099121 y:-10.918999671936035 z:54.34199905395508
  - atom: CB x: -13.907999992370605 y:-8.350000381469727 z:55.79999923706055
  - atom: CG x: -14.197999954223633 y:-7.88700008392334 z:57.22200012207031
  - atom: CD x: -13.293000221252441 y:-6.72599983215332 z:57.652000427246094
  - atom: OE1 x: -13.357000350952148 y:-5.650000095367432 z:57.01900100708008
  - atom: OE2 x: -12.512999534606934 y:-6.883999824523926 z:58.62099838256836
- residue TYR has 12 atoms
  - atom: N x: -14.315999984741211 y:-9.883000373840332 z:52.895999908447266
  - atom: CA x: -13.588000297546387 y:-10.373000144958496 z:51.72700119018555
  - atom: C x: -12.331000328063965 y:-9.508000373840332 z:51.678001403808594
  - atom: O x: -12.291000366210938 y:-8.444000244140625 z:52.292999267578125
  - atom: CB x: -14.402000427246094 y:-10.159000396728516 z:50.44300079345703
  - atom: CG x: -15.472000122070312 y:-11.201000213623047 z:50.17599868774414
  - atom: CD1 x: -15.234999656677246 y:-12.276000022888184 z:49.31800079345703
  - atom: CD2 x: -16.729000091552734 y:-11.105999946594238 z:50.7760009765625
```

```
  - atom: CE1 x: -16.225000381469727 y:-13.22700023651123 z:49.064998626708984
  - atom: CE2 x: -17.722999572753906 y:-12.048999786376953 z:50.52899932861328
  - atom: CZ x: -17.465999603271484 y:-13.100000381469727 z:49.67599868774414
  - atom: OH x: -18.458999633789062 y:-14.013999938964844 z:49.433998107910156
- residue PRO has 7 atoms
  - atom: N x: -11.282999992370605 y:-9.958999633789062 z:50.972999572753906
  - atom: CA x: -10.079999923706055 y:-9.125 z:50.917999267578125
  - atom: C x: -10.451000213623047 y:-7.831999778747559 z:50.20899963378906
  - atom: O x: -11.345000267028809 y:-7.835000038146973 z:49.35900115966797
  - atom: CB x: -9.114999771118164 y:-9.960000038146973 z:50.07699966430664
  - atom: CG x: -9.54699993133545 y:-11.354000091552734 z:50.3380012512207
  - atom: CD x: -11.050999641418457 y:-11.258999824523926 z:50.32500076293945
- residue HIS has 10 atoms
  - atom: N x: -9.805000305175781 y:-6.724999904632568 z:50.560001373291016
  - atom: CA x: -10.095000267028809 y:-5.4710001945495605 z:49.867000579833984
  - atom: C x: -9.428000450134277 y:-5.605000019073486 z:48.49399948120117
  - atom: O x: -8.277999877929688 y:-6.047999858856201 z:48.395999908447266
  - atom: CB x: -9.512999534606934 y:-4.267000198364258 z:50.619998931884766
  - atom: CG x: -10.348999977111816 y:-3.796999931335449 z:51.77000045776367
  - atom: ND1 x: -10.534000396728516 y:-4.544000148773193 z:52.91400146484375
  - atom: CD2 x: -11.008000373840332 y:-2.631999969482422 z:51.97100067138672
  - atom: CE1 x: -11.263999938964844 y:-3.8559999465942383 z:53.77299880981445
  - atom: NE2 x: -11.564000129699707 y:-2.691999912261963 z:53.224998474121094
- residue ARG has 11 atoms
  - atom: N x: -10.140999794006348 y:-5.234000205993652 z:47.4370002746582
  - atom: CA x: -9.592000007629395 y:-5.364999771118164 z:46.08599853515625
  - atom: C x: -9.380000114440918 y:-4.051000118255615 z:45.33399963378906
  - atom: O x: -10.031000137329102 y:-3.0390000343322754 z:45.611000061035156
  - atom: CB x: -10.51200008392334 y:-6.256999969482422 z:45.25899887084961
  - atom: CG x: -10.968999862670898 y:-7.511000156402588 z:45.96200180053711
  - atom: CD x: -9.836999893188477 y:-8.48799991607666 z:46.11199951171875
  - atom: NE x: -10.319000244140625 y:-9.800999641418457 z:46.51900100708008
  - atom: CZ x: -9.5600004196167 y:-10.888999938964844 z:46.54499816894531
  - atom: NH1 x: -8.28499984741211 y:-10.8100004196167 z:46.19200134277344
  - atom: NH2 x: -10.07699966430664 y:-12.057000160217285 z:46.902000427246094
- residue ARG has 11 atoms
  - atom: N x: -8.461999893188477 y:-4.076000213623047 z:44.375
  - atom: CA x: -8.197999954223633 y:-2.8940000534057617 z:43.558998107910156
  - atom: C x: -9.196999549865723 y:-2.7960000038146973 z:42.402000427246094
  - atom: O x: -9.545999526977539 y:-3.7890000343322754 z:41.75600051879883
  - atom: CB x: -6.775000095367432 y:-2.9260001182556152 z:42.994998931884766
  - atom: CG x: -5.702000141143799 y:-2.631999969482422 z:44.012001037597656
  - atom: CD x: -5.521999835968018 y:-1.1480000019073486 z:44.24100112915039
  - atom: NE x: -4.9710001945495605 y:-0.4950000047683716 z:43.0629997253418
  - atom: CZ x: -5.703999996185303 y:0.11400000005960464 z:42.13800048828125
  - atom: NH1 x: -7.0279998779296875 y:0.16500000655651093 z:42.25899887084961
  - atom: NH2 x: -5.113999843597412 y:0.652999997138977 z:41.07899856567383
- residue ILE has 8 atoms
  - atom: N x: -9.649999618530273 y:-1.5729999542236328 z:42.15399932861328
  - atom: CA x: -10.597000122070312 y:-1.2949999570846558 z:41.0880012512207
  - atom: C x: -10.175000190734863 y:-0.020999999716877937 z:40.34199905395508
  - atom: O x: -9.767999649047852 y:0.9660000205039978 z:40.95800018310547
  - atom: CB x: -12.017999649047852 y:-1.1380000114440918 z:41.68000030517578
  - atom: CG1 x: -13.008999824523926 y:-0.7609999775886536 z:40.59199905395508
  - atom: CG2 x: -12.031000137329102 y:-0.08799999952316284 z:42.75
  - atom: CD1 x: -14.376999855041504 y:-0.5450000166893005 z:41.137001037597656
- residue ASP has 8 atoms
  - atom: N x: -10.21500015258789 y:-0.07400000095367432 z:39.01499938964844
  - atom: CA x: -9.883000373840332 y:1.069000005722046 z:38.1619987487793
```

```
  - atom: C x: -11.109000205993652 y:1.2790000438690186 z:37.2760009765625
  - atom: O x: -11.706999778747559 y:0.3100000023841858 z:36.80699920654297
  - atom: CB x: -8.668000221252441 y:0.7850000262260437 z:37.270999908447266
  - atom: CG x: -7.4070000648498535 y:0.503000020980835 z:38.064998626708984
  - atom: OD1 x: -7.235000133514404 y:1.093000054359436 z:39.1510009765625
  - atom: OD2 x: -6.578999996185303 y:-0.30000001192092896 z:37.5890007019043
- residue ILE has 8 atoms
  - atom: N x: -11.487000465393066 y:2.5350000858306885 z:37.05500030517578
  - atom: CA x: -12.64799976348877 y:2.8480000495910645 z:36.23400115966797
  - atom: C x: -12.291999816894531 y:4.013000011444092 z:35.319000244140625
  - atom: O x: -11.878999710083008 y:5.072000026702881 z:35.79499816894531
  - atom: CB x: -13.845999717712402 y:3.256999969482422 z:37.12300109863281
  - atom: CG1 x: -14.107999801635742 y:2.1679999828338623 z:38.16699981689453
  - atom: CG2 x: -15.086999893188477 y:3.505000114440918 z:36.266998291015625
  - atom: CD1 x: -15.119000434875488 y:2.562000036239624 z:39.233001708984375
- residue ARG has 11 atoms
  - atom: N x: -12.4399995803833 y:3.8239998817443848 z:34.01100158691406
  - atom: CA x: -12.133000373840332 y:4.89300012588501 z:33.064998626708984
  - atom: C x: -13.319999694824219 y:5.24399995803833 z:32.17399978637695
  - atom: O x: -13.930999755859375 y:4.36899995803833 z:31.548999786376953
  - atom: CB x: -10.928000450134277 y:4.513000011444092 z:32.202999114990234
  - atom: CG x: -9.746000289916992 y:4.109000205993652 z:33.055999755859375
  - atom: CD x: -8.42300033569336 y:4.166999816894531 z:32.321998596191406
  - atom: NE x: -7.381999969482422 y:3.446000099182129 z:33.055999755859375
  - atom: CZ x: -6.118000030517578 y:3.3529999256134033 z:32.65599822998047
  - atom: NH1 x: -5.738999843597412 y:3.947000026702881 z:31.527999877929688
  - atom: NH2 x: -5.239999771118164 y:2.6530001163482666 z:33.36800003051758
- residue LEU has 8 atoms
  - atom: N x: -13.649999618530273 y:6.5329999923706055 z:32.150001525878906
  - atom: CA x: -14.743000030517578 y:7.039999961853027 z:31.347000122070312
  - atom: C x: -14.180999755859375 y:7.448999881744385 z:29.990999221801758
  - atom: O x: -13.451000213623047 y:8.435999870300293 z:29.905000686645508
  - atom: CB x: -15.355999946594238 y:8.258999824523926 z:32.018001556396484
  - atom: CG x: -16.48200035095215 y:8.942999839782715 z:31.2450008392334
  - atom: CD1 x: -17.711000442504883 y:8.062999725341797 z:31.25
  - atom: CD2 x: -16.80299949645996 y:10.270000457763672 z:31.878000259399414
- residue ILE has 8 atoms
  - atom: N x: -14.506999969482422 y:6.693999767303467 z:28.940000534057617
  - atom: CA x: -14.029000282287598 y:7.006000004196167 z:27.59600067138672
  - atom: C x: -15.149999618530273 y:7.5229997634887695 z:26.711000442504883
  - atom: O x: -16.299999237060547 y:7.142000198364258 z:26.871999740600586
  - atom: CB x: -13.437999725341797 y:5.7779998779296875 z:26.885000228881836
  - atom: CG1 x: -14.513999938964844 y:4.703999996185303 z:26.7549991607666
  - atom: CG2 x: -12.199999809265137 y:5.271999835968018 z:27.624000549316406
  - atom: CD1 x: -14.071999549865723 y:3.513000011444092 z:25.937999725341797
- residue PRO has 7 atoms
  - atom: N x: -14.821000099182129 y:8.404000282287598 z:25.766000747680664
  - atom: CA x: -15.810999870300293 y:8.979000091552734 z:24.83799934387207
  - atom: C x: -16.490999221801758 y:7.840000152587891 z:24.062999725341797
  - atom: O x: -15.810999870300293 y:6.901000022888184 z:23.624000549316406
  - atom: CB x: -14.961999893188477 y:9.873000144958496 z:23.937999725341797
  - atom: CG x: -13.824999809265137 y:10.289999961853027 z:24.843000411987305
  - atom: CD x: -13.48900032043457 y:8.99899959564209 z:25.56800079345703
- residue LYS has 9 atoms
  - atom: N x: -17.80900001525879 y:7.921999931335449 z:23.8799991607666
  - atom: CA x: -18.554000854492188 y:6.8480000495910645 z:23.209999084472656
  - atom: C x: -18.02199935913086 y:6.297999858856201 z:21.886999130249023
  - atom: O x: -17.986000061035156 y:5.081999778747559 z:21.684999465942383
  - atom: CB x: -20.007999420166016 y:7.251999855041504 z:22.993000030517578
```

```
  - atom: CG x: -20.899999618530273 y:6.093999862670898 z:22.548999786376953
  - atom: CD x: -21.552000045776367 y:5.38100004196167 z:23.746000289916992
  - atom: CE x: -22.51300048828125 y:4.2829999923706055 z:23.280000686645508
  - atom: NZ x: -23.47800064086914 y:3.8610000610351562 z:24.336000442504883
- residue ASP has 8 atoms
  - atom: N x: -17.625999450683594 y:7.181000232696533 z:20.983999252319336
  - atom: CA x: -17.1200008392334 y:6.754000186920166 z:19.68600082397461
  - atom: C x: -15.680000305175781 y:6.251999855041504 z:19.711999893188477
  - atom: O x: -15.003000259399414 y:6.2179999351501465 z:18.670000076293945
  - atom: CB x: -17.22599983215332 y:7.90500020980835 z:18.687000274658203
  - atom: CG x: -16.69099998474121 y:9.199000358581543 z:19.2450008392334
  - atom: OD1 x: -15.991000175476074 y:9.913999557495117 z:18.503999710083008
  - atom: OD2 x: -16.976999282836914 y:9.498000144958496 z:20.422000885009766
- residue GLN has 9 atoms
  - atom: N x: -15.208000183105469 y:5.869999885559082 z:20.895000457763672
  - atom: CA x: -13.840999603271484 y:5.36899995803833 z:21.02899932861328
  - atom: C x: -13.864999771118164 y:4.067999839782715 z:21.81100082397461
  - atom: O x: -12.859999656677246 y:3.375 z:21.93899917602539
  - atom: CB x: -12.97700023651123 y:6.406000137329102 z:21.72800064086914
  - atom: CG x: -13.196000099182129 y:7.809000015258789 z:21.20599937438965
  - atom: CD x: -11.937000274658203 y:8.630999565124512 z:21.261999130249023
  - atom: OE1 x: -11.22599983215332 y:8.621999740600586 z:22.266000747680664
  - atom: NE2 x: -11.645999908447266 y:9.350000381469727 z:20.18199920654297
- residue TYR has 12 atoms
  - atom: N x: -15.048999786376953 y:3.755000114440918 z:22.316999435424805
  - atom: CA x: -15.3100004196167 y:2.546999931335449 z:23.077999114990234
  - atom: C x: -14.543000221252441 y:1.3359999656677246 z:22.562999725341797
  - atom: O x: -13.607999801635742 y:0.8619999885559082 z:23.190000534057617
  - atom: CB x: -16.804000854492188 y:2.242000102996826 z:23.031999588012695
  - atom: CG x: -17.19499969482422 y:1.00600004196167 z:23.79199981689453
  - atom: CD1 x: -17.104999542236328 y:0.9660000205039978 z:25.180999755859375
  - atom: CD2 x: -17.6299991607666 y:-0.13699999451637268 z:23.121000289916992
  - atom: CE1 x: -17.437999725341797 y:-0.1850000023841858 z:25.889999389648438
  - atom: CE2 x: -17.964000701904297 y:-1.2970000505447388 z:23.815000534057617
  - atom: CZ x: -17.865999221801758 y:-1.319000005722046 z:25.20199966430664
  - atom: OH x: -18.17799949645996 y:-2.4670000076293945 z:25.902000427246094
- residue TYR has 12 atoms
  - atom: N x: -14.942999839782715 y:0.8270000219345093 z:21.410999298095703
  - atom: CA x: -14.288000106811523 y:-0.3409999907016754 z:20.8700008392334
  - atom: C x: -12.777999877929688 y:-0.21400000154972076 z:20.922000885009766
  - atom: O x: -12.097000122070312 y:-1.1469999551773071 z:21.32200050354004
  - atom: CB x: -14.779999732971191 y:-0.5960000157356262 z:19.445999145507812
  - atom: CG x: -16.284000396728516 y:-0.7670000195503235 z:19.385000228881836
  - atom: CD1 x: -16.902000427246094 y:-1.9190000295639038 z:19.878999710083008
  - atom: CD2 x: -17.097999572753906 y:0.25699999928474426 z:18.899999618530273
  - atom: CE1 x: -18.301000595092773 y:-2.0450000762939453 z:19.89900016784668
  - atom: CE2 x: -18.493999481201172 y:0.1469999998807907 z:18.916000366210938
  - atom: CZ x: -19.089000701904297 y:-1.0019999742507935 z:19.420000076293945
  - atom: OH x: -20.4689998626709 y:-1.069000005722046 z:19.485000610351562
- residue CYS has 6 atoms
  - atom: N x: -12.243000030517578 y:0.9359999895095825 z:20.545000076293945
  - atom: CA x: -10.789999961853027 y:1.1050000190734863 z:20.573999404907227
  - atom: C x: -10.241999626159668 y:1.0920000076293945 z:22.003999710083008
  - atom: O x: -9.145000457763672 y:0.6069999933242798 z:22.257999420166016
  - atom: CB x: -10.390999794006348 y:2.4019999504089355 z:19.875999450683594
  - atom: SG x: -9.300999641418457 y:2.1089999675750732 z:18.48900032043457
- residue GLY has 4 atoms
  - atom: N x: -11.019000053405762 y:1.6349999904632568 z:22.93199920654297
  - atom: CA x: -10.604000091552734 y:1.6460000276565552 z:24.31800079345703
```

```
  - atom: C x: -10.835000038146973 y:0.289000004529953 z:24.96299934387207
  - atom: O x: -10.050999641418457 y:-0.16200000047683716 z:25.792999267578125
- residue VAL has 7 atoms
  - atom: N x: -11.920000076293945 y:-0.36800000071525574 z:24.582000732421875
  - atom: CA x: -12.220999717712402 y:-1.6660000085830688 z:25.134000778198242
  - atom: C x: -11.131999969482422 y:-2.628000020980835 z:24.66900062561035
  - atom: O x: -10.866999626159668 y:-3.63700008392334 z:25.305999755859375
  - atom: CB x: -13.607000350952148 y:-2.1589999198913574 z:24.66699981689453
  - atom: CG1 x: -13.576000213623047 y:-2.502000093460083 z:23.184999465942383
  - atom: CG2 x: -14.036999702453613 y:-3.3519999980926514 z:25.479999542236328
- residue LEU has 8 atoms
  - atom: N x: -10.487000465393066 y:-2.309999942779541 z:23.559999465942383
  - atom: CA x: -9.4399995803833 y:-3.1710000038146973 z:23.065000534057617
  - atom: C x: -8.232999801635742 y:-2.9670000076293945 z:23.94700050354004
  - atom: O x: -7.6620001792907715 y:-3.9210000038146973 z:24.485000610351562
  - atom: CB x: -9.074000358581543 y:-2.805999994277954 z:21.634000778198242
  - atom: CG x: -7.961999893188477 y:-3.6480000019073486 z:21.007999420166016
  - atom: CD1 x: -8.39900016784668 y:-5.107999801635742 z:20.95599937438965
  - atom: CD2 x: -7.644999980926514 y:-3.119999885559082 z:19.613000869750977
- residue TYR has 12 atoms
  - atom: N x: -7.855000019073486 y:-1.7059999704360962 z:24.0939998626709
  - atom: CA x: -6.703000068664551 y:-1.3450000286102295 z:24.893999099731445
  - atom: C x: -6.7870001792907715 y:-1.8880000114440918 z:26.312999725341797
  - atom: O x: -5.877999782562256 y:-2.578000068664551 z:26.77199935913086
  - atom: CB x: -6.550000190734863 y:0.17399999499320984 z:24.950000762939453
  - atom: CG x: -5.408999919891357 y:0.6000000238418579 z:25.836000442504883
  - atom: CD1 x: -4.098999977111816 y:0.5410000085830688 z:25.382999420166016
  - atom: CD2 x: -5.636000156402588 y:0.9879999756813049 z:27.158000946044922
  - atom: CE1 x: -3.0409998893737793 y:0.8510000109672546 z:26.216999053955078
  - atom: CE2 x: -4.584000110626221 y:1.2960000038146973 z:28.003999710083008
  - atom: CZ x: -3.2880001068115234 y:1.222000002861023 z:27.525999069213867
  - atom: OH x: -2.2260000705718994 y:1.49399995803833 z:28.360000610351562
- residue PHE has 11 atoms
  - atom: N x: -7.882999897003174 y:-1.565000057220459 z:26.993999481201172
  - atom: CA x: -8.121000289916992 y:-1.968999981880188 z:28.378000259399414
  - atom: C x: -8.307999610900879 y:-3.4790000915527344 z:28.600000381469727
  - atom: O x: -8.409000396728516 y:-3.937999963760376 z:29.736000061035156
  - atom: CB x: -9.335000038146973 y:-1.2009999752044678 z:28.913999557495117
  - atom: CG x: -9.093999862670898 y:-0.5210000276565552 z:30.23200035095215
  - atom: CD1 x: -7.830999851226807 y:-0.009999999776482582 z:30.555999755859375
  - atom: CD2 x: -10.133000373840332 y:-0.37299999594688416 z:31.152999877929688
  - atom: CE1 x: -7.607999801635742 y:0.6340000033378601 z:31.783000946044922
  - atom: CE2 x: -9.920999526977539 y:0.27399998903274536 z:32.38800048828125
  - atom: CZ x: -8.659000396728516 y:0.7749999761581421 z:32.70100021362305
- residue THR has 7 atoms
  - atom: N x: -8.368000030517578 y:-4.252999782562256 z:27.52199935913086
  - atom: CA x: -8.527000427246094 y:-5.695000171661377 z:27.659000396728516
  - atom: C x: -7.127999782562256 y:-6.291999816894531 z:27.558000564575195
  - atom: O x: -6.834000110626221 y:-7.348999977111816 z:28.125
  - atom: CB x: -9.447999954223633 y:-6.2870001792907715 z:26.538000106811523
  - atom: OG1 x: -10.755999565124512 y:-5.709000110626221 z:26.632999420166016
  - atom: CG2 x: -9.590999603271484 y:-7.790999889373779 z:26.69499969482422
- residue GLY has 4 atoms
  - atom: N x: -6.26200008392334 y:-5.583000183105469 z:26.841999053955078
  - atom: CA x: -4.89900016784668 y:-6.0329999923706055 z:26.658000946044922
  - atom: C x: -4.8379998207092285 y:-7.366000175476074 z:25.944000244140625
  - atom: O x: -5.651000022888184 y:-7.636000156402588 z:25.055999755859375
- residue SER has 6 atoms
  - atom: N x: -3.8940000534057617 y:-8.215999603271484 z:26.344999313354492
```

```
  - atom: CA x: -2.950000047683716 y:-7.925000190734863 z:27.43899917602539
  - atom: C x: -1.9830000400543213 y:-6.807000160217285 z:27.10099983215332
  - atom: O x: -1.75600004196167 y:-6.502999782562256 z:25.93000030517578
  - atom: CB x: -2.1080000400543213 y:-9.154000282287598 z:27.775999069213867
  - atom: OG x: -1.0529999732971191 y:-9.291999816894531 z:26.840999603271484
- residue ASP has 8 atoms
  - atom: N x: -1.3969999551773071 y:-6.228000164031982 z:28.145000457763672
  - atom: CA x: -0.42899999022483826 y:-5.145999908447266 z:28.013999938964844
  - atom: C x: 0.6370000243186951 y:-5.468999862670898 z:26.96299934387207
  - atom: O x: 0.9330000281333923 y:-4.64900016784668 z:26.097000122070312
  - atom: CB x: 0.22100000083446503 y:-4.900000095367432 z:29.3700008392334
  - atom: CG x: 0.7870000004768372 y:-6.170000076293945 z:29.974000930786133
  - atom: OD1 x: 1.9320000410079956 y:-6.541999816894531 z:29.61400032043457
  - atom: OD2 x: 0.0740000095367432 y:-6.804999828338623 z:30.790000915527344
- residue ILE has 8 atoms
  - atom: N x: 1.215999960899353 y:-6.6620001792907715 z:27.04199981689453
  - atom: CA x: 2.2239999771118164 y:-7.057000160217285 z:26.06999969482422
  - atom: C x: 1.5240000486373901 y:-7.189000129699707 z:24.72800064086914
  - atom: O x: 2.115000009536743 y:-6.8979997634887695 z:23.695999145507812
  - atom: CB x: 2.88700008392334 y:-8.416000366210938 z:26.416000366210938
  - atom: CG1 x: 3.5199999809265137 y:-8.373000144958496 z:27.809999465942383
  - atom: CG2 x: 3.9570000171661377 y:-8.734999656677246 z:25.385000228881836
  - atom: CD1 x: 4.730000019073486 y:-7.460999965667725 z:27.913000106811523
- residue PHE has 11 atoms
  - atom: N x: 0.27000001072883606 y:-7.638999938964844 z:24.740999221801758
  - atom: CA x: -0.4909999966621399 y:-7.76800012588501 z:23.500999450683594
  - atom: C x: -0.550000011920929 y:-6.389999866485596 z:22.841999053955078
  - atom: O x: -0.3019999861717224 y:-6.252999782562256 z:21.641000747680664
  - atom: CB x: -1.9140000343322754 y:-8.262000083923340 z:23.777000427246094
  - atom: CG x: -2.802999973297119 y:-8.269000053405762 z:22.55900001525879
  - atom: CD1 x: -4.051000118255615 y:-7.645999908447266 z:22.586000442504883
  - atom: CD2 x: -2.3959999084472656 y:-8.892999649047852 z:21.381999969482422
  - atom: CE1 x: -4.880000114440918 y:-7.64300012588501 z:21.461000442504883
  - atom: CE2 x: -3.2209999561309814 y:-8.895000457763672 z:20.25200080871582
  - atom: CZ x: -4.4629998207092285 y:-8.267000198364258 z:20.29599952697754
- residue ASN has 8 atoms
  - atom: N x: -0.8730000257492065 y:-5.379000186920166 z:23.64900016784668
  - atom: CA x: -0.9649999737739563 y:-3.994999885559082 z:23.198999404907227
  - atom: C x: 0.36899998784065247 y:-3.4709999561309814 z:22.68199920654297
  - atom: O x: 0.4090000092983246 y:-2.625 z:21.791000366210938
  - atom: CB x: -1.4759999513626099 y:-3.121000051498413 z:24.334999084472656
  - atom: CG x: -2.9230000972747803 y:-3.3919999599456787 z:24.648000717163086
  - atom: OD1 x: -3.4649999141693115 y:-4.428999900817871 z:24.25200080871582
  - atom: ND2 x: -3.565000057220459 y:-2.4709999561309814 z:25.361000061035156
- residue LYS has 9 atoms
  - atom: N x: 1.4589999914169312 y:-3.9660000801086426 z:23.25200080871582
  - atom: CA x: 2.7829999923706055 y:-3.575000047683716 z:22.798999786376953
  - atom: C x: 2.8970000743865967 y:-4.184000015258789 z:21.39900016784668
  - atom: O x: 3.390000104904175 y:-3.553999900817871 z:20.46500015258789
  - atom: CB x: 3.865000009536743 y:-4.192999839782715 z:23.691999435424805
  - atom: CG x: 4.803999900817871 y:-3.2190001010894775 z:24.374000549316406
  - atom: CD x: 4.2220001220703125 y:-2.7219998836517334 z:25.68199920654297
  - atom: CE x: 5.249000072479248 y:-1.906999945640564 z:26.447999954223633
  - atom: NZ x: 4.71999979019165 y:-1.440000057220459 z:27.759000778198242
- residue ASN has 8 atoms
  - atom: N x: 2.4240000247955322 y:-5.421000003814697 z:21.270999908447266
  - atom: CA x: 2.4730000495910645 y:-6.144999980926514 z:20.006000518798828
  - atom: C x: 1.680999994277954 y:-5.458000183105469 z:18.91900062561035
  - atom: O x: 2.183000087738037 y:-5.236999988555908 z:17.82200050354004
```

```
  - atom: CB x: 1.9279999732971191 y:-7.566999912261963 z:20.170000076293945
  - atom: CG x: 2.819000005722046 y:-8.437000274658203 z:21.020999908447266
  - atom: OD1 x: 4.0329999923706055 y:-8.473999977111816 z:20.822999954223633
  - atom: ND2 x: 2.2230000495910645 y:-9.152000427246094 z:21.969999313354492
- residue MET has 8 atoms
  - atom: N x: 0.43299999833106995 y:-5.135000228881836 z:19.23699951171875
  - atom: CA x: -0.46299999952316284 y:-4.486000061035156 z:18.29400062561035
  - atom: C x: -0.013000000268220901 y:-3.0759999752044678 z:17.910999298095703
  - atom: O x: 0.13500000536441803 y:-2.7690000534057617 z:16.731000900268555
  - atom: CB x: -1.8730000257492065 y:-4.451000213623047 z:18.875
  - atom: CG x: -2.9140000343322754 y:-3.936000108718872 z:17.91699981689453
  - atom: SD x: -4.511000156402588 y:-3.989000082015991 z:18.68199920654297
  - atom: CE x: -5.236000061035156 y:-5.382999897003174 z:17.812000274658203
- residue ARG has 11 atoms
  - atom: N x: 0.20499999821186066 y:-2.2179999351501465 z:18.899999618530273
  - atom: CA x: 0.6489999890327454 y:-0.8560000061988831 z:18.618000030517578
  - atom: C x: 2.0329999923706055 y:-0.8849999904632568 z:17.986000061035156
  - atom: O x: 2.75600004196167 y:0.10899999737739563 z:18.027000427246094
  - atom: CB x: 0.7039999961853027 y:-0.02199999988079071 z:19.905000686645508
  - atom: CG x: -0.6389999985694885 y:0.16200000047683716 z:20.607999801635742
  - atom: CD x: -0.546999990940094 y:1.11899995803833 z:21.801000595092773
  - atom: NE x: 0.36800000071525574 y:0.6430000066757202 z:22.839000701904297
  - atom: CZ x: 1.6900000057220459 y:0.8040000200271606 z:22.823999404907227
  - atom: NH1 x: 2.2869999408721924 y:1.440999984741211 z:21.822999954223633
  - atom: NH2 x: 2.4210000038146973 y:0.3109999895095825 z:23.812000274658203
- residue ALA has 5 atoms
  - atom: N x: 2.3959999084472656 y:-2.0299999713897705 z:17.41200065612793
  - atom: CA x: 3.694999933242798 y:-2.2079999446868896 z:16.777000427246094
  - atom: C x: 3.558000087738037 y:-2.7639999389648438 z:15.357999801635742
  - atom: O x: 4.267000198364258 y:-2.3389999866485596 z:14.446999549865723
  - atom: CB x: 4.560999870300293 y:-3.131999969482422 z:17.625
- residue HIS has 10 atoms
  - atom: N x: 2.6530001163482666 y:-3.7170000076293945 z:15.168000221252441
  - atom: CA x: 2.447000026702881 y:-4.2870001792907715 z:13.842000007629395
  - atom: C x: 1.50600004196167 y:-3.38100004196167 z:13.050000190734863
  - atom: O x: 1.3990000486373901 y:-3.496000051498413 z:11.829999923706055
  - atom: CB x: 1.8489999771118164 y:-5.692999839782715 z:13.930999755859375
  - atom: CG x: 2.2960000038146973 y:-6.5980000495910645 z:12.826000213623047
  - atom: ND1 x: 3.5880000591278076 y:-7.076000213623047 z:12.737000465393066
  - atom: CD2 x: 1.6419999599456787 y:-7.076000213623047 z:11.741000175476074
  - atom: CE1 x: 3.7090001106262207 y:-7.809999942779541 z:11.645000457763672
  - atom: NE2 x: 2.5429999828338623 y:-7.827000141143799 z:11.02299976348877
- residue ALA has 5 atoms
  - atom: N x: 0.828000009059906 y:-2.4809999465942383 z:13.76200008392334
  - atom: CA x: -0.10000000149011612 y:-1.5369999408721924 z:13.149999618530273
  - atom: C x: 0.6909999847412109 y:-0.40799999237060547 z:12.517999649047852
  - atom: O x: 0.12399999797344208 y:0.5239999890327454 z:11.958000183105469
  - atom: CB x: -1.0499999523162842 y:-0.9750000238418579 z:14.199999809265137
- residue LEU has 8 atoms
  - atom: N x: 2.009999990463257 y:-0.4959999918937683 z:12.625
  - atom: CA x: 2.8910000324249268 y:0.5099999904632568 z:12.059000015258789
  - atom: C x: 3.50600004196167 y:-0.035000000149011161 z:10.78499984741211
  - atom: O x: 3.7899999618530273 y:0.7179999947547913 z:9.859999656677246
  - atom: CB x: 3.9809999465942383 y:0.875 z:13.062000274658203
  - atom: CG x: 3.4670000076293945 y:1.5980000495910645 z:14.305000305175781
  - atom: CD1 x: 4.455999851226807 y:1.4429999589920044 z:15.440999984741211
  - atom: CD2 x: 3.243000030517578 y:3.062999963760376 z:13.977999687194824
- residue GLU has 9 atoms
  - atom: N x: 3.7119998931884766 y:-1.3480000495910645 z:10.744999885559082
```

```
  - atom: CA x: 4.270999908447266 y:-2.0 z:9.562999725341797
  - atom: C x: 3.1630001068115234 y:-2.0859999656677246 z:8.519000053405762
  - atom: O x: 3.4089999198913574 y:-2.381999969482422 z:7.3460001945495605
  - atom: CB x: 4.775000095367432 y:-3.4110000133514404 z:9.897000312805176
  - atom: CG x: 6.122000217437744 y:-3.4679999351501465 z:10.611000061035156
  - atom: CD x: 6.538000106811523 y:-4.894000053405762 z:10.965999603271484
  - atom: OE1 x: 5.827000141143799 y:-5.545000076293945 z:11.767999649047852
  - atom: OE2 x: 7.573999881744385 y:-5.364999771118164 z:10.442000389099121
- residue LYS has 9 atoms
  - atom: N x: 1.937000036239624 y:-1.8250000476837158 z:8.96399974822998
  - atom: CA x: 0.7760000228881836 y:-1.8489999771118164 z:8.09000015258789
  - atom: C x: 0.4650000035762787 y:-0.4359999895095825 z:7.625
  - atom: O x: -0.1469999998807907 y:-0.23800000548362732 z:6.577000141143799
  - atom: CB x: -0.42500001192092896 y:-2.430999994277954 z:8.829000473022461
  - atom: CG x: -0.34700000286102295 y:-3.931999921798706 z:9.03600025177002
  - atom: CD x: -0.3109999895095825 y:-4.669000148773193 z:7.701000213623047
  - atom: CE x: -0.36899998784065247 y:-6.182000160217285 z:7.88100004196167
  - atom: NZ x: -0.4009999930858612 y:-6.886000156402588 z:6.570000171661377
- residue GLY has 4 atoms
  - atom: N x: 0.8970000147819519 y:0.5450000166893005 z:8.40999984741211
  - atom: CA x: 0.6600000262260437 y:1.930999994277954 z:8.053999900817871
  - atom: C x: -0.44699999690055847 y:2.5869998931884766 z:8.862000465393066
  - atom: O x: -1.2719999551773071 y:3.318000078201294 z:8.3100004196167
- residue PHE has 11 atoms
  - atom: N x: -0.4729999899864197 y:2.3239998817443848 z:10.166999816894531
  - atom: CA x: -1.4789999723434448 y:2.8940000534057617 z:11.060999870300293
  - atom: C x: -0.8289999961853027 y:3.25 z:12.390000343322754
  - atom: O x: 0.3659999966621399 y:3.0260000228881836 z:12.576000213623047
  - atom: CB x: -2.614000082015991 y:1.8949999809265137 z:11.293999671936035
  - atom: CG x: -3.365000009536743 y:1.534999966621399 z:10.04699993133545
  - atom: CD1 x: -2.7880001068115234 y:0.7170000076293945 z:9.079999923706055
  - atom: CD2 x: -4.638999938964844 y:2.0510001182556152 z:9.812999725341797
  - atom: CE1 x: -3.4639999866485596 y:0.42100000381469727 z:7.896999835968018
  - atom: CE2 x: -5.323999881744385 y:1.7610000371932983 z:8.633999824523926
  - atom: CZ x: -4.734000205993652 y:0.9449999928474426 z:7.675000190734863
- residue THR has 7 atoms
  - atom: N x: -1.6059999465942383 y:3.812999963760376 z:13.309000015258789
  - atom: CA x: -1.0820000171661377 y:4.171000003814697 z:14.621000289916992
  - atom: C x: -2.1449999809265137 y:4.130000114440918 z:15.713000297546387
  - atom: O x: -2.8350000381469727 y:5.118000030517578 z:15.982999801635742
  - atom: CB x: -0.4440000057220459 y:5.560999870300293 z:14.616999626159668
  - atom: OG1 x: -0.1979999989271164 y:5.96999979019165 z:15.970999717712402
  - atom: CG2 x: -1.3569999933242798 y:6.568999767303467 z:13.937999725341797
- residue ILE has 8 atoms
  - atom: N x: -2.260999917984009 y:2.9660000801086426 z:16.340000015258789
  - atom: CA x: -3.2179999351501465 y:2.749000072479248 z:17.410999298095703
  - atom: C x: -2.7730000019073486 y:3.4049999713897705 z:18.715999603271484
  - atom: O x: -1.5839999914169312 y:3.5250000953674316 z:18.99799919128418
  - atom: CB x: -3.384000062942505 y:1.25 z:17.70800018310547
  - atom: CG1 x: -3.8320000171661377 y:0.5090000033378601 z:16.45400047302246
  - atom: CG2 x: -4.36899995803833 y:1.0549999475479126 z:18.841999053955078
  - atom: CD1 x: -3.9230000972747803 y:-0.9929999709129333 z:16.635000228881836
- residue ASN has 8 atoms
  - atom: N x: -3.750999927520752 y:3.8329999446868896 z:19.49799919128418
  - atom: CA x: -3.5160000324249268 y:4.416999816894531 z:20.804000854492188
  - atom: C x: -4.771999835968018 y:3.9769999980926514 z:21.53499984741211
  - atom: O x: -5.671000003814697 y:3.4179999828338623 z:20.91699981689453
  - atom: CB x: -3.364000082015991 y:5.951000213623047 z:20.72800064086914
  - atom: CG x: -4.677000045776367 y:6.679999828338623 z:20.570999145507812
```

```
  - atom: OD1 x: -5.383999824523926 y:6.507999897003174 z:19.586000442504883
  - atom: ND2 x: -5.002999782562256 y:7.520999908447266 z:21.547000885009766
- residue GLU has 9 atoms
  - atom: N x: -4.8480000495910645 y:4.176000118255615 z:22.83799934387207
  - atom: CA x: -6.034999847412109 y:3.7279999256134033 z:23.54800033569336
  - atom: C x: -7.315999984741211 y:4.426000118255615 z:23.10700035095215
  - atom: O x: -8.413999557495117 y:4.002999782562256 z:23.4689998626709
  - atom: CB x: -5.839000225067139 y:3.9059998989105225 z:25.049999237060547
  - atom: CG x: -5.285999774932861 y:5.250999927520752 z:25.452999114990234
  - atom: CD x: -5.427000045776367 y:5.492000102996826 z:26.93899917602539
  - atom: OE1 x: -6.584000110626221 y:5.568999767303467 z:27.41699981689453
  - atom: OE2 x: -4.386000156402588 y:5.5960001945495605 z:27.628999710083008
- residue TYR has 12 atoms
  - atom: N x: -7.186999797821045 y:5.4730000495910645 z:22.301000595092773
  - atom: CA x: -8.359000205993652 y:6.21999979019165 z:21.871999740600586
  - atom: C x: -8.829000473022461 y:6.129000186920166 z:20.430999755859375
  - atom: O x: -10.012999534606934 y:6.314000129699707 z:20.163000106811523
  - atom: CB x: -8.189000129699707 y:7.686999797821045 z:22.2549991607666
  - atom: CG x: -8.407999992370605 y:7.910999774932861 z:23.726999282836914
  - atom: CD1 x: -9.579999923706055 y:7.461999893188477 z:24.343000411987305
  - atom: CD2 x: -7.442999839782715 y:8.538999557495117 z:24.511999130249023
  - atom: CE1 x: -9.788999557495117 y:7.626999855041504 z:25.70400047302246
  - atom: CE2 x: -7.639999866485596 y:8.711000442504883 z:25.878999710083008
  - atom: CZ x: -8.817999839782715 y:8.24899959564209 z:26.469999313354492
  - atom: OH x: -9.015999794006348 y:8.385000228881836 z:27.829000047302246
- residue THR has 7 atoms
  - atom: N x: -7.926000118255615 y:5.84499979019165 z:19.503999710083008
  - atom: CA x: -8.315999984741211 y:5.76200008392334 z:18.10300064086914
  - atom: C x: -7.355000019073486 y:4.928999900817871 z:17.274999618530273
  - atom: O x: -6.382999897003174 y:4.380000114440918 z:17.781999588012695
  - atom: CB x: -8.359000205993652 y:7.159999847412109 z:17.450000762939453
  - atom: OG1 x: -7.072000026702881 y:7.771999835968018 z:17.58099937438965
  - atom: CG2 x: -9.40999984741211 y:8.045999526977539 z:18.10300064086914
- residue ILE has 8 atoms
  - atom: N x: -7.651000022888184 y:4.840000152587891 z:15.984999656677246
  - atom: CA x: -6.809999942779541 y:4.129000186920166 z:15.03600025177002
  - atom: C x: -6.605000019073486 y:5.14300012588501 z:13.92300033569336
  - atom: O x: -7.561999797821045 y:5.546000003814697 z:13.265999794006348
  - atom: CB x: -7.505000114440918 y:2.8489999771118164 z:14.496000289916992
  - atom: CG1 x: -6.677000045776367 y:2.243000030517578 z:13.366000175476074
  - atom: CG2 x: -8.907999992370605 y:3.1640000343322754 z:14.024999618530273
  - atom: CD1 x: -5.2870001792907715 y:1.8519999980926514 z:13.770000457763672
- residue ARG has 11 atoms
  - atom: N x: -5.361999988555908 y:5.576000213623047 z:13.73900032043457
  - atom: CA x: -5.0320000648498535 y:6.578000068664551 z:12.72599983215332
  - atom: C x: -4.03000020980835 y:6.0970001220703125 z:11.682000160217285
  - atom: O x: -3.0840001106262207 y:5.367000102996826 z:11.996000289916992
  - atom: CB x: -4.446000099182129 y:7.829999923706055 z:13.387999534606934
  - atom: CG x: -5.349999904632568 y:8.541000366210938 z:14.371999740600586
  - atom: CD x: -4.564000129699707 y:9.571000099182129 z:15.16100025177002
  - atom: NE x: -5.436999797821045 y:10.416000366210938 z:15.961999893188477
  - atom: CZ x: -6.255000114440918 y:11.333999633789062 z:15.456000328063965
  - atom: NH1 x: -6.309000015258789 y:11.53499984741211 z:14.142999649047852
  - atom: NH2 x: -7.034999847412109 y:12.043000221252441 z:16.260000228881836
- residue PRO has 7 atoms
  - atom: N x: -4.21999979019165 y:6.513000011444092 z:10.418999671936035
  - atom: CA x: -3.3289999961853027 y:6.136000156402588 z:9.314000129699707
  - atom: C x: -2.0810000896453857 y:7.013000011444092 z:9.279999732971191
  - atom: O x: -2.1059999465942383 y:8.13700008392334 z:9.77299976348877
```

```
  - atom: CB x: -4.203000068664551 y:6.354000091552734 z:8.08899974822998
  - atom: CG x: -5.043000221252441 y:7.53000020980835 z:8.503999710083008
  - atom: CD x: -5.443999767303467 y:7.158999919891357 z:9.90999984741211
- residue LEU has 8 atoms
  - atom: N x: -0.9909999966621399 y:6.5 z:8.71500015258789
  - atom: CA x: 0.23899999260902405 y:7.2820000648498535 z:8.612000465393066
  - atom: C x: 0.4749999940395355 y:7.710999965667725 z:7.160999774932861
  - atom: O x: 0.6140000224113464 y:6.875999927520752 z:6.260000228881836
  - atom: CB x: 1.4550000429153442 y:6.493000030517578 z:9.128000259399414
  - atom: CG x: 1.6330000162124634 y:6.239999771118164 z:10.633000373840332
  - atom: CD1 x: 3.072000026702881 y:5.822000026702881 z:10.888999938964844
  - atom: CD2 x: 1.3270000219345093 y:7.493000030517578 z:11.430999755859375
- residue GLY has 4 atoms
  - atom: N x: 0.5199999809265137 y:9.02400016784668 z:6.947000026702881
  - atom: CA x: 0.7110000252723694 y:9.560999870300293 z:5.611999988555908
  - atom: C x: 2.055000066757202 y:9.293999671936035 z:4.956999778747559
  - atom: O x: 2.5769999027252197 y:8.173999786376953 z:5.01200008392334
- residue VAL has 5 atoms
  - atom: N x: 2.6010000705718994 y:10.335000038146973 z:4.326000213623047
  - atom: CA x: 3.885999917984009 y:10.26200008392334 z:3.632999897003174
  - atom: C x: 5.041999816894531 y:10.295999526977539 z:4.625999927520752
  - atom: O x: 5.809000015258789 y:9.336999893188477 z:4.73799991607666
  - atom: CB x: 4.046000003814697 y:11.435999870300293 z:2.6449999809265137
- residue THR has 7 atoms
  - atom: N x: 5.1620001792907715 y:11.411999702453613 z:5.3379998207092285
  - atom: CA x: 6.205999851226807 y:11.58899974822998 z:6.3429999351501465
  - atom: C x: 5.745999813079834 y:10.847000122070312 z:7.5879998207092285
  - atom: O x: 6.538000106811523 y:10.524999618530273 z:8.479999542236328
  - atom: CB x: 6.401000022888184 y:13.09000015258789 z:6.693999767303467
  - atom: OG1 x: 7.418000221252441 y:13.229000091552734 z:7.696000099182129
  - atom: CG2 x: 5.10099983215332 y:13.690999984741211 z:7.216000080108643
- residue GLY has 4 atoms
  - atom: N x: 4.451000213623047 y:10.559000015258789 z:7.620999813079834
  - atom: CA x: 3.871999979019165 y:9.878999710083008 z:8.756999969482422
  - atom: C x: 3.1549999713897705 y:10.939000129699707 z:9.560999870300293
  - atom: O x: 3.7009999752044678 y:11.47599983215332 z:10.526000022888184
- residue VAL has 7 atoms
  - atom: N x: 1.9320000410079956 y:11.253999710083008 z:9.142000198364258
  - atom: CA x: 1.121000051498413 y:12.267000198364258 z:9.810999870300293
  - atom: C x: -0.13600000739097595 y:11.644000053405762 z:10.413000106811523
  - atom: O x: -0.7139999866485596 y:10.715999603271484 z:9.848999977111816
  - atom: CB x: 0.6930000185966492 y:13.37600040435791 z:8.821000099182129
  - atom: CG1 x: 0.11800000071525574 y:14.564000129699707 z:9.583999633789062
  - atom: CG2 x: 1.8819999694824219 y:13.795000076293945 z:7.960999965667725
- residue ALA has 5 atoms
  - atom: N x: -0.5550000071525574 y:12.157999992370605 z:11.562000274658203
  - atom: CA x: -1.746000051498413 y:11.645999908447266 z:12.223999977111816
  - atom: C x: -2.989000082015991 y:11.880000114440918 z:11.366999626159668
  - atom: O x: -3.7219998836517334 y:12.854999542236328 z:11.567999839782715
  - atom: CB x: -1.9160000085830688 y:12.317000389099121 z:13.588000297546387
- residue GLY has 4 atoms
  - atom: N x: -3.2219998836517334 y:10.994000434875488 z:10.402999877929688
  - atom: CA x: -4.394999980926514 y:11.140999794006348 z:9.5649995803833
  - atom: C x: -5.603000164031982 y:11.170000076293945 z:10.477999687194824
  - atom: O x: -5.493000030517578 y:10.807000160217285 z:11.645999908447266
- residue GLU has 9 atoms
  - atom: N x: -6.754000186920166 y:11.597999572753906 z:9.97599983215332
  - atom: CA x: -7.928999900817871 y:11.63700008392334 z:10.82699966430664
  - atom: C x: -8.236000061035156 y:10.243000030517578 z:11.357999801635742
```

```
  - atom: O x: -7.811999797821045 y:9.243000030517578 z:10.781999588012695
  - atom: CB x: -9.128000259399414 y:12.204000473022461 z:10.067999839782715
  - atom: CG x: -8.973999977111816 y:13.675999641418457 z:9.729000091552734
  - atom: CD x: -8.668999671936035 y:14.520999908447266 z:10.954000473022461
  - atom: OE1 x: -9.545999526977539 y:14.63599967956543 z:11.836999893188477
  - atom: OE2 x: -7.546000003814697 y:15.064000129699707 z:11.038999557495117
- residue PRO has 7 atoms
  - atom: N x: -8.956999778747559 y:10.168000221252441 z:12.486000061035156
  - atom: CA x: -9.35099983215332 y:8.928999900817871 z:13.156000137329102
  - atom: C x: -10.333999633789062 y:8.079000473022461 z:12.36299991607666
  - atom: O x: -11.458999633789062 y:8.48799991607666 z:12.119999885559082
  - atom: CB x: -9.963000297546387 y:9.427000045776367 z:14.458000183105469
  - atom: CG x: -9.27299976348877 y:10.722000122070312 z:14.6899995803833
  - atom: CD x: -9.281999588012695 y:11.329000473022461 z:13.32800006866455
- residue LEU has 8 atoms
  - atom: N x: -9.914999961853027 y:6.882999897003174 z:11.980999946594238
  - atom: CA x: -10.781000137329102 y:5.994999885559082 z:11.220000267028809
  - atom: C x: -12.0 y:5.538000106811523 z:12.052000045776367
  - atom: O x: -11.925000190734863 y:5.4029998779296875 z:13.284000396728516
  - atom: CB x: -9.970000267028809 y:4.785999774932861 z:10.727999687194824
  - atom: CG x: -8.637999534606934 y:5.125 z:10.034000396728516
  - atom: CD1 x: -7.88100004196167 y:3.8529999256134033 z:9.711999893188477
  - atom: CD2 x: -8.88599967956543 y:5.926000118255615 z:8.770999908447266
- residue PRO has 7 atoms
  - atom: N x: -13.142999649047852 y:5.301000118255615 z:11.381999969482422
  - atom: CA x: -14.369999885559082 y:4.864999771118164 z:12.055999755859375
  - atom: C x: -14.175000190734863 y:3.4600000381469727 z:12.61299991607666
  - atom: O x: -13.482000350952148 y:2.6470000743865967 z:12.015000343322754
  - atom: CB x: -15.402999877929688 y:4.925000190734863 z:10.942999839782715
  - atom: CG x: -14.595000267028809 y:4.534999847412109 z:9.737000465393066
  - atom: CD x: -13.314000129699707 y:5.303999900817871 z:9.918000221252441
- residue VAL has 7 atoms
  - atom: N x: -14.791000366210938 y:3.1710000038146973 z:13.751999855041504
  - atom: CA x: -14.619999885559082 y:1.871999979019165 z:14.383999824523926
  - atom: C x: -15.878000259399414 y:1.4279999732971191 z:15.11400032043457
  - atom: O x: -16.320999145507812 y:2.0989999771118164 z:16.045000076293945
  - atom: CB x: -13.45199966430664 y:1.9279999732971191 z:15.414999961853027
  - atom: CG1 x: -13.395000457763672 y:0.6629999876022339 z:16.229000091552734
  - atom: CG2 x: -12.142000198364258 y:2.13700008392334 z:14.706000328063965
- residue ASP has 8 atoms
  - atom: N x: -16.44499969482422 y:0.29499998688697815 z:14.708000183105469
  - atom: CA x: -17.641000747680664 y:-0.2070000022649765 z:15.37600040435791
  - atom: C x: -17.44300079345703 y:-1.562000036239624 z:16.04199981689453
  - atom: O x: -18.38599967956543 y:-2.1410000324249268 z:16.584999084472656
  - atom: CB x: -18.812000274658203 y:-0.27399998903274536 z:14.397000312805176
  - atom: CG x: -19.378000259399414 y:1.090999960899353 z:14.090999603271484
  - atom: OD1 x: -19.974000930786133 y:1.7089999914169312 z:15.01099967956543
  - atom: OD2 x: -19.208999633789062 y:1.5449999570846558 z:12.937000274658203
- residue SER has 6 atoms
  - atom: N x: -16.21299934387207 y:-2.056999921798706 z:16.011999130249023
  - atom: CA x: -15.904000282287598 y:-3.3350000381469727 z:16.624000549316406
  - atom: C x: -14.416999816894531 y:-3.440999984741211 z:16.860000610351562
  - atom: O x: -13.630000114440918 y:-2.6700000762939453 z:16.297000885009766
  - atom: CB x: -16.336000442504883 y:-4.482999801635742 z:15.71399974822998
  - atom: OG x: -15.477999687194824 y:-4.5929999351501465 z:14.583999633789062
- residue GLU has 9 atoms
  - atom: N x: -14.02299976348877 y:-4.389999866485596 z:17.70199966430664
  - atom: CA x: -12.605999946594238 y:-4.583000183105469 z:17.94700050354004
  - atom: C x: -12.114999771118164 y:-5.175000190734863 z:16.631000518798828
```

```
  - atom: O x: -11.086999893188477 y:-4.754000186920166 z:16.094999313354492
  - atom: CB x: -12.359999656677246 y:-5.572000026702881 z:19.093000411987305
  - atom: CG x: -12.958000183105469 y:-5.177999973297119 z:20.433000564575195
  - atom: CD x: -12.744000434875488 y:-6.241000175476074 z:21.503000259399414
  - atom: OE1 x: -11.678000450134277 y:-6.239999771118164 z:22.149999618530273
  - atom: OE2 x: -13.63700008392334 y:-7.091000080108643 z:21.690000534057617
- residue LYS has 9 atoms
  - atom: N x: -12.881999969482422 y:-6.133999824523926 z:16.10300064086914
  - atom: CA x: -12.550999641418457 y:-6.808000087738037 z:14.845000267028809
  - atom: C x: -12.190999984741211 y:-5.770999908447266 z:13.795000076293945
  - atom: O x: -11.251999855041504 y:-5.955999851226807 z:13.017000198364258
  - atom: CB x: -13.72700023651123 y:-7.6579999923706055 z:14.350000381469727
  - atom: CG x: -13.319999694824219 y:-8.717000007629395 z:13.312999725341797
  - atom: CD x: -14.470999717712402 y:-9.666000366210938 z:12.951000213623047
  - atom: CE x: -13.95300006866455 y:-10.96500015258789 z:12.319999694824219
  - atom: NZ x: -13.071999549865723 y:-10.746999740600586 z:11.128999710083008
- residue ASP has 8 atoms
  - atom: N x: -12.949999809265137 y:-4.681000232696533 z:13.784000396728516
  - atom: CA x: -12.708999633789062 y:-3.5869998931884766 z:12.864999771118164
  - atom: C x: -11.27299976348877 y:-3.0889999866485596 z:12.994000434875488
  - atom: O x: -10.62399959564209 y:-2.819999933242798 z:11.98799991607666
  - atom: CB x: -13.6850004196167 y:-2.4489998817443848 z:13.14900016784668
  - atom: CG x: -15.019000053405762 y:-2.638000011444092 z:12.45199966430664
  - atom: OD1 x: -15.416999816894531 y:-3.799999952316284 z:12.22700023651123
  - atom: OD2 x: -15.673999786376953 y:-1.6230000257492065 z:12.137999534606934
- residue ILE has 8 atoms
  - atom: N x: -10.767000198364258 y:-2.9700000286102295 z:14.220000267028809
  - atom: CA x: -9.394000053405762 y:-2.506999969482422 z:14.414999961853027
  - atom: C x: -8.388999938964844 y:-3.5079998970031774 z:13.859999656677246
  - atom: O x: -7.361000061035156 y:-3.1089999675750732 z:13.307999610900879
  - atom: CB x: -9.074000358581543 y:-2.252000093460083 z:15.904000282287598
  - atom: CG1 x: -9.920999526977539 y:-1.0920000076293945 z:16.42300033569336
  - atom: CG2 x: -7.604000091552734 y:-1.9160000085830688 z:16.07200050354004
  - atom: CD1 x: -9.934000015258789 y:-0.9750000238418579 z:17.92300033569336
- residue PHE has 11 atoms
  - atom: N x: -8.67199993133545 y:-4.802999973297119 z:13.996999740600586
  - atom: CA x: -7.755000114440918 y:-5.816999912261963 z:13.468000411987305
  - atom: C x: -7.697000026702881 y:-5.741000175476074 z:11.954000473022461
  - atom: O x: -6.613999843597412 y:-5.801000118255615 z:11.366000175476074
  - atom: CB x: -8.1850004196167 y:-7.229000091552734 z:13.838000297546387
  - atom: CG x: -8.031000137329102 y:-7.553999900817871 z:15.281000137329102
  - atom: CD1 x: -8.97599983215332 y:-7.142000198364258 z:16.200000762939453
  - atom: CD2 x: -6.9629998207092285 y:-8.329000473022461 z:15.71399974822998
  - atom: CE1 x: -8.86400032043457 y:-7.50600004196167 z:17.531999588012695
  - atom: CE2 x: -6.8420000076293945 y:-8.699000358581543 z:17.04400062561035
  - atom: CZ x: -7.793000221252441 y:-8.289999961853027 z:17.95400047302246
- residue ASP has 8 atoms
  - atom: N x: -8.868000030517578 y:-5.627999782562256 z:11.32800006866455
  - atom: CA x: -8.956000328063965 y:-5.539999961853027 z:9.875
  - atom: C x: -7.9770002365112305 y:-4.494999885559082 z:9.343000411987305
  - atom: O x: -7.117000102996826 y:-4.798999786376953 z:8.513999938964844
  - atom: CB x: -10.378999710083008 y:-5.177999973297119 z:9.451000213623047
  - atom: CG x: -11.38599967956543 y:-6.243000030517578 z:9.824000358581543
  - atom: OD1 x: -11.04800033569336 y:-7.438000202178955 z:9.717000007629395
  - atom: OD2 x: -12.520000457763672 y:-5.888000011444092 z:10.206999778747559
- residue TYR has 12 atoms
  - atom: N x: -8.109999656677246 y:-3.2669999599456787 z:9.835000038146973
  - atom: CA x: -7.236000061035156 y:-2.171999931335449 z:9.430000305175781
  - atom: C x: -5.750999927520752 y:-2.553999900817871 z:9.388999938964844
```

```
    - atom: O x: -5.015999794006348 y:-2.0889999866485596 z:8.51200008392334
    - atom: CB x: -7.4120001792907715 y:-0.9829999804496765 z:10.37600040435791
    - atom: CG x: -8.711999893188477 y:-0.2290000021457672 z:10.222999572753906
    - atom: CD1 x: -8.961999893188477 y:0.5600000023841858 z:9.097999572753906
    - atom: CD2 x: -9.673999786376953 y:-0.26100000739097595 z:11.227999687194824
    - atom: CE1 x: -10.13700008392334 y:1.3040000200271606 z:8.98799991607666
    - atom: CE2 x: -10.85200023651123 y:0.4740000069141388 z:11.125
    - atom: CZ x: -11.074000358581543 y:1.253999948501587 z:10.008000373840332
    - atom: OH x: -12.22599983215332 y:1.99399995803833 z:9.934000015258789
  - residue ILE has 8 atoms
    - atom: N x: -5.306000232696533 y:-3.385999917984009 z:10.331999778747559
    - atom: CA x: -3.9000000953674316 y:-3.7909998893737793 z:10.378999710083008
    - atom: C x: -3.6470000743865967 y:-5.202000141143799 z:9.848999977111816
    - atom: O x: -2.5460000038146973 y:-5.744999885559082 z:9.998000144958496
    - atom: CB x: -3.3369998931884766 y:-3.7060000896453857 z:11.810999870300293
    - atom: CG1 x: -4.114999771118164 y:-4.64900016784668 z:12.734999656677246
    - atom: CG2 x: -3.4070000648498535 y:-2.2730000019073486 z:12.29800033569336
    - atom: CD1 x: -3.5190000534057617 y:-4.806000232696533 z:14.109000205993652
  - residue GLN has 9 atoms
    - atom: N x: -4.672999858856201 y:-5.791999816894531 z:9.23799991607666
    - atom: CA x: -4.566999912261963 y:-7.129000186920166 z:8.661999702453613
    - atom: C x: -3.9079999923706055 y:-8.130999565124512 z:9.602999687194824
    - atom: O x: -2.7799999713897705 y:-8.581999778747559 z:9.387999534606934
    - atom: CB x: -3.802999973297119 y:-7.051000118255615 z:7.3420000076293945
    - atom: CG x: -4.520999908447266 y:-6.198999881744385 z:6.308000087738037
    - atom: CD x: -3.6029999256134033 y:-5.708000183105469 z:5.218999862670898
    - atom: OE1 x: -2.9730000495910645 y:-6.502999782562256 z:4.520999908447266
    - atom: NE2 x: -3.515000104904175 y:-4.388000011444092 z:5.065999984741211
  - residue TRP has 14 atoms
    - atom: N x: -4.6479997634887695 y:-8.468000411987305 z:10.651000022888184
    - atom: CA x: -4.2129998207092285 y:-9.406999588012695 z:11.666000366210938
    - atom: C x: -5.4029998779296875 y:-10.319000244140625 z:11.925000190734863
    - atom: O x: -6.546999931335449 y:-9.857000350952148 z:11.963000297546387
    - atom: CB x: -3.867000102996826 y:-8.654000282287598 z:12.949999809265137
    - atom: CG x: -2.4119999408721924 y:-8.647000312805176 z:13.345000267028809
    - atom: CD1 x: -1.3270000219345093 y:-8.75100040435791 z:12.519000053405762
    - atom: CD2 x: -1.8910000324249268 y:-8.4350004196167 z:14.666000366210938
    - atom: NE1 x: -0.164000004529953 y:-8.61299991607666 z:13.244999885559082
    - atom: CE2 x: -0.4819999933242798 y:-8.416000366210938 z:14.562999725341797
    - atom: CE3 x: -2.4809999465942383 y:-8.253999710083008 z:15.925999641418457
    - atom: CZ2 x: 0.34700000286102295 y:-8.222999572753906 z:15.670999526977539
    - atom: CZ3 x: -1.6579999923706055 y:-8.062000274658203 z:17.027000427246094
    - atom: CH2 x: -0.257999986410141 y:-8.04800033569336 z:16.891000747680664
  - residue LYS has 9 atoms
    - atom: N x: -5.131999969482422 y:-11.611000061035156 z:12.079999923706055
    - atom: CA x: -6.172999858856201 y:-12.593999862670898 z:12.36400032043457
    - atom: C x: -6.895999908447266 y:-12.119000434875488 z:13.630999565124512
    - atom: O x: -6.25 y:-11.840999603271484 z:14.640000343322754
    - atom: CB x: -5.515999794006348 y:-13.961000442504883 z:12.612000465393066
    - atom: CG x: -6.453999996185303 y:-15.081999778747559 z:13.057999610900879
    - atom: CD x: -5.710000038146973 y:-16.18600082397461 z:13.833000183105469
    - atom: CE x: -4.585000038146973 y:-16.8439998626709 z:13.029000282287598
    - atom: NZ x: -3.36299991607666 y:-15.994999885559082 z:12.892999649047852
  - residue TYR has 12 atoms
    - atom: N x: -8.222000122070312 y:-12.010000228881836 z:13.583000183105469
    - atom: CA x: -8.960000038146973 y:-11.565999984741211 z:14.760000228881836
    - atom: C x: -8.760000228881836 y:-12.576000213623047 z:15.869999885559082
    - atom: O x: -8.963000297546387 y:-13.774999618530273 z:15.677000045776367
    - atom: CB x: -10.458000183105469 y:-11.418999671936035 z:14.470000267028809
```

```
  - atom: CG x: -11.286999702453613 y:-11.11299991607666 z:15.708000183105469
  - atom: CD1 x: -10.930999755859375 y:-10.07800006866455 z:16.576000213623047
  - atom: CD2 x: -12.423999786376953 y:-11.857999801635742 z:16.016000747680664
  - atom: CE1 x: -11.687000274658203 y:-9.791999816894531 z:17.722999572753906
  - atom: CE2 x: -13.187000274658203 y:-11.57800006866455 z:17.160999298095703
  - atom: CZ x: -12.8100004196167 y:-10.543999671936035 z:18.007999420166016
  - atom: OH x: -13.557000160217285 y:-10.265999794006348 z:19.131999969482422
- residue ARG has 11 atoms
  - atom: N x: -8.35099983215332 y:-12.07800006866455 z:17.030000686645508
  - atom: CA x: -8.109999656677246 y:-12.918000221252441 z:18.19099998474121
  - atom: C x: -9.045999526977539 y:-12.545000076293945 z:19.327999114990234
  - atom: O x: -9.180000305175781 y:-11.371999740600586 z:19.68600082397461
  - atom: CB x: -6.64300012588501 y:-12.793999671936035 z:18.628999710083008
  - atom: CG x: -5.697000026702881 y:-13.623000144958496 z:17.770999908447266
  - atom: CD x: -4.306000232696533 y:-13.03600025177002 z:17.694000244140625
  - atom: NE x: -3.683000087738037 y:-12.902999877929688 z:19.003999710083008
  - atom: CZ x: -2.4519999027252197 y:-12.432999610900879 z:19.19300079345703
  - atom: NH1 x: -1.718999981880188 y:-12.057999610900879 z:18.14900016784668
  - atom: NH2 x: -1.9579999446868896 y:-12.322999954223633 z:20.422000885009766
- residue GLU has 9 atoms
  - atom: N x: -9.708999633789062 y:-13.553999900817871 z:19.881999969482422
  - atom: CA x: -10.630000114440918 y:-13.32800006866455 z:20.979999542236328
  - atom: C x: -9.880999565124512 y:-12.907999992370605 z:22.240999221801758
  - atom: O x: -8.704000473022461 y:-13.222000122070312 z:22.416000366210938
  - atom: CB x: -11.458000183105469 y:-14.58899974822998 z:21.243000030517578
  - atom: CG x: -12.715999603271484 y:-14.678000450134277 z:20.395999908447266
  - atom: CD x: -13.70300006866455 y:-13.559000015258789 z:20.702999114990234
  - atom: OE1 x: -14.116000175476074 y:-13.442999839782715 z:21.875999450683594
  - atom: OE2 x: -14.069000244140625 y:-12.795999526977539 z:19.7810001373291
- residue PRO has 7 atoms
  - atom: N x: -10.557000160217285 y:-12.173999786376953 z:23.131999969482422
  - atom: CA x: -9.939000129699707 y:-11.720000267028809 z:24.371000289916992
  - atom: C x: -9.152999877929688 y:-12.814000129699707 z:25.082000732421875
  - atom: O x: -8.093999862670898 y:-12.553000450134277 z:25.649999618530273
  - atom: CB x: -11.133000373840332 y:-11.241999626159668 z:25.180000305175781
  - atom: CG x: -11.991999626159668 y:-10.654999732971191 z:24.128000259399414
  - atom: CD x: -11.949000358581543 y:-11.706999778747559 z:23.049999237060547
- residue LYS has 9 atoms
  - atom: N x: -9.663999557495117 y:-14.041000366210938 z:25.042999267578125
  - atom: CA x: -8.987000465393066 y:-15.14900016784668 z:25.709999084472656
  - atom: C x: -7.636000156402588 y:-15.522000312805176 z:25.11199951171875
  - atom: O x: -6.769999980926514 y:-16.047000885009766 z:25.81800079345703
  - atom: CB x: -9.890999794006348 y:-16.393999099731445 z:25.76099967956543
  - atom: CG x: -10.576000213623047 y:-16.767000198364258 z:24.45599937438965
  - atom: CD x: -11.342000007629395 y:-18.08099937438965 z:24.591999053955078
  - atom: CE x: -12.46399974822998 y:-18.19700050354004 z:23.55500030517578
  - atom: NZ x: -11.987000465393066 y:-18.033000946044922 z:22.152999877929688
- residue ASP has 8 atoms
  - atom: N x: -7.442999839782715 y:-15.236000061035156 z:23.82699966430664
  - atom: CA x: -6.190999984741211 y:-15.576000213623047 z:23.158000946044922
  - atom: C x: -5.249000072479248 y:-14.38599967956543 z:23.02199935913086
  - atom: O x: -4.289000034332275 y:-14.437000274658203 z:22.250999450683594
  - atom: CB x: -6.478000164031982 y:-16.13599967956543 z:21.766000747680664
  - atom: CG x: -7.806000232696533 y:-16.85700035095215 z:21.69099998474121
  - atom: OD1 x: -7.918000221252441 y:-17.975000381469727 z:22.240999221801758
  - atom: OD2 x: -8.744000434875488 y:-16.29400062561035 z:21.083000183105469
- residue ARG has 11 atoms
  - atom: N x: -5.5269999504089355 y:-13.3100004196167 z:23.753000259399414
  - atom: CA x: -4.684999942779541 y:-12.119999885559082 z:23.701000213623047
```

```
  - atom: C x: -3.753000020980835 y:-12.059000015258789 z:24.905000686645508
  - atom: O x: -2.9630000591278076 y:-11.131999969482422 z:25.038000106811523
  - atom: CB x: -5.535999774932861 y:-10.854000091552734 z:23.65399932861328
  - atom: CG x: -6.466000080108643 y:-10.753000259399414 z:22.45599937438965
  - atom: CD x: -7.349999904632568 y:-9.517000198364258 z:22.576000213623047
  - atom: NE x: -8.53499984741211 y:-9.595999717712402 z:21.73699951171875
  - atom: CZ x: -9.552000045776367 y:-8.748000144958496 z:21.80900001525879
  - atom: NH1 x: -9.529999732971191 y:-7.751999855041504 z:22.68600082397461
  - atom: NH2 x: -10.593000411987305 y:-8.897000312805176 z:21.003000259399414
- residue SER has 6 atoms
  - atom: N x: -3.8580000400543213 y:-13.041000366210938 z:25.791000366210938
  - atom: CA x: -2.989000082015991 y:-13.07699966430664 z:26.95599937438965
  - atom: C x: -1.5839999914169312 y:-13.416000366210938 z:26.474000930786133
  - atom: O x: -1.284999966621399 y:-14.574000358581543 z:26.16699981689453
  - atom: CB x: -3.4690001010894775 y:-14.126999855041504 z:27.961000442504883
  - atom: OG x: -4.433000087738037 y:-13.57800006866455 z:28.847000122070312
- residue GLU has 10 atoms
  - atom: N x: -0.7360000014305115 y:-12.394000053405762 z:26.395999908447266
  - atom: CA x: 0.6309999823570251 y:-12.565999984741211 z:25.93899917602539
  - atom: C x: 1.6269999742507935 y:-11.968999862670898 z:26.92799949645996
  - atom: O x: 2.7279999256134033 y:-12.541000366210938 z:27.077999114990234
  - atom: CB x: 0.7960000038146973 y:-11.9399995803833 z:24.552000045776367
  - atom: CG x: -0.11400000005960464 y:-12.574999809265137 z:23.51099967956543
  - atom: CD x: 0.17499999701976776 y:-12.116999626159668 z:22.09000015258789
  - atom: OE1 x: 0.09000000357627869 y:-10.902999877929688 z:21.815000534057617
  - atom: OE2 x: 0.4819999933242798 y:-12.97700023651123 z:21.23900032043457
  - atom: OXT x: 1.3029999732971191 y:-10.935999870300293 z:27.54400062561035
- residue MN has 1 atoms
  - atom: MN x: -6.214000225067139 y:-5.52400016784668 z:35.61399841308594
- residue MN has 1 atoms
  - atom: MN x: -6.142000198364258 y:-2.124000072479248 z:36.37300109863281
- residue MN has 1 atoms
  - atom: MN x: 13.84000015258789 y:-1.5140000581741333 z:61.231998443603516
- residue MN has 1 atoms
  - atom: MN x: 8.105999946594238 y:-25.408000946044922 z:15.770000457763672
- residue NA has 1 atoms
  - atom: NA x: 4.251999855041504 y:-0.11299999803304672 z:43.87799835205078
- residue NA has 1 atoms
  - atom: NA x: 16.698999404907227 y:-13.654000282287598 z:18.052000045776367
- residue F2A has 30 atoms
  - atom: PA x: -3.8420000076293945 y:-3.880000114440918 z:34.38800048828125
  - atom: O1A x: -5.063000202178955 y:-3.622999906539917 z:35.20000076293945
  - atom: O2A x: -2.532999992370605 y:-3.359999895095825 z:34.85300064086914
  - atom: C3A x: -3.6730000972747803 y:-5.749000072479248 z:34.435001373291016
  - atom: O5' x: -4.066999912261963 y:-3.2890000343322754 z:32.92499923706055
  - atom: PB x: -4.76800012588501 y:-7.072999954223633 z:33.6879997253418
  - atom: O1B x: -4.3150000572204590 y:-7.23199987411499 z:32.28099822998047
  - atom: O2B x: -6.175000190734863 y:-6.676000118255615 z:33.970001220703125
  - atom: O3B x: -4.465000152587891 y:-8.342000007629395 z:34.42300033569336
  - atom: PG x: -4.432000160217285 y:-8.357000350952148 z:35.915000915527344
  - atom: O1G x: -2.990999937057495 y:-8.444000244140625 z:36.2599983215332
  - atom: O2G x: -5.375 y:-9.385000228881836 z:36.42900085449219
  - atom: O3G x: -4.946000099182129 y:-7.052999973297119 z:36.409000396728516
  - atom: C5' x: -5.205999851226807 y:-3.6419999599456787 z:32.145999908447266
  - atom: C4' x: -4.960000038146973 y:-3.3380000591278076 z:30.687000274658203
  - atom: O4' x: -4.2230000495910645 y:-2.0969998836517334 z:30.54800033569336
  - atom: C1' x: -3.4019999504089355 y:-2.1610000133514404 z:29.38800048828125
  - atom: N9 x: -2.00600004196167 y:-1.9320000410079956 z:29.775999069213867
  - atom: C4 x: -1.0019999742507935 y:-1.4759999513626099 z:28.95199966430664
```

```
  - atom: N3 x: -1.1009999513626099 y:-1.1349999904632568 z:27.658000946044922
  - atom: C2 x: 0.08100000023841858 y:-0.7540000081062317 z:27.180999755859375
  - atom: N1 x: 1.2619999647140503 y:-0.6800000071525574 z:27.80299949645996
  - atom: C6 x: 1.3289999961853027 y:-1.0219999551773071 z:29.106000900268555
  - atom: N6 x: 2.5069999694824222 y:-0.9359999895095825 z:29.729999542236328
  - atom: C5 x: 0.14499999582767487 y:-1.4479999542236328 z:29.731000900268555
  - atom: N7 x: -0.12700000405311584 y:-1.8680000305175781 z:31.024999618530273
  - atom: C8 x: -1.4129999876022339 y:-2.138000011444092 z:31.003999710083008
  - atom: C2' x: -3.621999979019165 y:-3.5390000343322754 z:28.760000228881836
  - atom: C3' x: -4.086999893188477 y:-4.3429999351501465 z:29.964000701904297
  - atom: O3' x: -4.868000030517578 y:-5.460000038146973 z:29.570999145507812
- residue HOH has 1 atoms
  - atom: O x: -2.4089999198913574 y:8.484999656677246 z:43.20899963378906
- residue HOH has 1 atoms
  - atom: O x: -0.14300000667572021 y:-1.7710000276565552 z:48.332000732421875
- residue HOH has 1 atoms
  - atom: O x: -16.437999725341797 y:-4.840000152587891 z:18.80699920654297
- residue HOH has 1 atoms
  - atom: O x: 16.079999923706055 y:1.4079999923706055 z:30.270999908447266
- residue HOH has 1 atoms
  - atom: O x: 21.016000747680664 y:-1.5190000534057617 z:33.81999969482422
- residue HOH has 1 atoms
  - atom: O x: -6.730999946594238 y:0.6259999871253967 z:20.711999893188477
- residue HOH has 1 atoms
  - atom: O x: 20.124000549316406 y:-7.186999797821045 z:32.66400146484375
- residue HOH has 1 atoms
  - atom: O x: -12.125 y:5.327000141143799 z:17.954999923706055
- residue HOH has 1 atoms
  - atom: O x: -13.345999717712402 y:10.083999633789062 z:45.89799880981445
- residue HOH has 1 atoms
  - atom: O x: -8.428000450134277 y:6.59499979019165 z:30.35300064086914
- residue HOH has 1 atoms
  - atom: O x: -18.88599967956543 y:7.940000057220459 z:27.29599952697754
- residue HOH has 1 atoms
  - atom: O x: -12.711000442504883 y:-14.880000114440918 z:25.305999755859375
- residue HOH has 1 atoms
  - atom: O x: 13.637999534606934 y:-8.12399959564209 z:24.416000366210938
- residue HOH has 1 atoms
  - atom: O x: -4.113999843597412 y:-2.135999917984009 z:36.98699951171875
- residue HOH has 1 atoms
  - atom: O x: 12.25 y:-8.463000297546387 z:47.82099914550781
- residue HOH has 1 atoms
  - atom: O x: 13.711000442504883 y:2.4779999256134033 z:60.03499984741211
- residue HOH has 1 atoms
  - atom: O x: 13.440999984741211 y:-24.489999771118164 z:30.077999114990234
- residue HOH has 1 atoms
  - atom: O x: -1.4060000181198120 y:-12.001999855041504 z:50.17599868774414
- residue HOH has 1 atoms
  - atom: O x: -18.986000061035156 y:-14.868000030517578 z:32.553001403808594
- residue HOH has 1 atoms
  - atom: O x: -10.734000205993652 y:-14.918000221252441 z:48.27399826049805
- residue HOH has 1 atoms
  - atom: O x: 11.635000228881836 y:-24.492000579833984 z:31.9689998626709
- residue HOH has 1 atoms
  - atom: O x: 3.058000087738037 y:-2.069999933242798 z:43.28900146484375
- residue HOH has 1 atoms
  - atom: O x: 1.7419999837875366 y:-11.512999534606934 z:49.70600128173828
- residue HOH has 1 atoms
  - atom: O x: -25.611000061035156 y:13.20199966430664 z:33.939998626708984
```

```
- residue HOH has 1 atoms
  - atom: O x: -29.035999298095703 y:5.309000015258789 z:24.209999084472656
- residue HOH has 1 atoms
  - atom: O x: 0.31000000023841858 y:-5.184999942779541 z:48.196998596191406
- residue HOH has 1 atoms
  - atom: O x: -13.279000282287598 y:3.0230000019073486 z:18.819000244140625
- residue HOH has 1 atoms
  - atom: O x: -23.634000778198242 y:-16.91900062561035 z:58.840999603271484
- residue HOH has 1 atoms
  - atom: O x: -2.9119999408721924 y:-6.1479997634887695 z:54.277000427246094
- residue HOH has 1 atoms
  - atom: O x: -1.1610000133514404 y:10.043000221252441 z:47.38800048828125
- residue HOH has 1 atoms
  - atom: O x: 29.593000411987305 y:-14.854999542236328 z:40.95399856567383
- residue HOH has 1 atoms
  - atom: O x: -12.48900032043457 y:5.578999996185303 z:53.430999755859375
- residue HOH has 1 atoms
  - atom: O x: 12.057000160217285 y:2.822999954223633 z:36.19300079345703
- residue HOH has 1 atoms
  - atom: O x: 27.393999099731445 y:-13.095000267028809 z:37.10499954223633
- residue HOH has 1 atoms
  - atom: O x: -28.520000457763672 y:9.92199993133545 z:38.66999816894531
- residue HOH has 1 atoms
  - atom: O x: -5.872000217437744 y:10.038999557495117 z:19.493999481201172
- residue HOH has 1 atoms
  - atom: O x: 25.288000106811523 y:-19.02199935913086 z:24.20199966430664
- residue HOH has 1 atoms
  - atom: O x: -3.128999948501587 y:-4.6519999504089355 z:39.78499984741211
- residue HOH has 1 atoms
  - atom: O x: -15.873000144958496 y:-9.145000457763672 z:20.726999282836914
- residue HOH has 1 atoms
  - atom: O x: -5.986999988555908 y:-5.311999797821045 z:55.07600021362305
- residue HOH has 1 atoms
  - atom: O x: -22.631999969482422 y:-11.35200023651123 z:32.91999816894531
- residue HOH has 1 atoms
  - atom: O x: 15.930000305175781 y:-11.468000411987305 z:48.625
- residue HOH has 1 atoms
  - atom: O x: 10.845999717712402 y:10.20199966430664 z:45.08700180053711
- residue HOH has 1 atoms
  - atom: O x: -1.7580000162124634 y:-6.790999889373779 z:39.04999923706055
- residue HOH has 1 atoms
  - atom: O x: 0.8429999947547913 y:1.375 z:59.32500076293945
- residue HOH has 1 atoms
  - atom: O x: -23.67099952697754 y:9.42199993133545 z:51.630001068115234
- residue HOH has 1 atoms
  - atom: O x: -7.863999843597412 y:-7.434000015258789 z:35.81700134277344
- residue HOH has 1 atoms
  - atom: O x: -14.73900032043457 y:-2.5480000972747803 z:57.404998779296875
- residue HOH has 1 atoms
  - atom: O x: -12.61299991607666 y:-3.4010000228881836 z:56.3650016784668
- residue HOH has 1 atoms
  - atom: O x: -12.157999992370605 y:14.347999572753906 z:9.017999649047852
- residue HOH has 1 atoms
  - atom: O x: -10.178999900817871 y:16.46299934387207 z:7.783999919891357
- residue HOH has 1 atoms
  - atom: O x: -26.187999725341797 y:1.3630000352859497 z:32.53200149536133
- residue HOH has 1 atoms
  - atom: O x: -19.97100067138672 y:13.097999572753906 z:46.61899948120117
- residue HOH has 1 atoms
```

```
  - atom: O x: 24.52400016784668 y:-18.415000915527344 z:41.80699920654297
- residue HOH has 1 atoms
  - atom: O x: -18.277999877929688 y:-7.002999782562256 z:31.499000549316406
- residue HOH has 1 atoms
  - atom: O x: -17.966999053955078 y:-11.64799976348877 z:31.121000289916992
- residue HOH has 1 atoms
  - atom: O x: 19.69499969482422 y:-9.303000450134277 z:57.77000045776367
- residue HOH has 1 atoms
  - atom: O x: 20.47800064086914 y:-7.195000171661377 z:55.983001708984375
- residue HOH has 1 atoms
  - atom: O x: -20.400999069213867 y:-4.622000217437744 z:26.72800064086914
- residue HOH has 1 atoms
  - atom: O x: -23.23699951171875 y:-3.7679998874664307 z:26.531999588012695
- residue HOH has 1 atoms
  - atom: O x: -21.20199966430664 y:0.3619999885559082 z:21.589000701904297
- residue HOH has 1 atoms
  - atom: O x: -1.5870000123977661 y:-0.9639999866485596 z:42.207000732421875
- residue HOH has 1 atoms
  - atom: O x: -24.679000854492188 y:-1.2269999980926514 z:32.99100112915039
- residue HOH has 1 atoms
  - atom: O x: -27.02199935913086 y:10.708000183105469 z:33.94599914550781
- residue HOH has 1 atoms
  - atom: O x: 11.821000099182129 y:-2.0350000858306885 z:62.28200149536133
- residue HOH has 1 atoms
  - atom: O x: 13.04800033569336 y:-3.318000078201294 z:60.446998596191406
- residue HOH has 1 atoms
  - atom: O x: -16.23699951171875 y:-0.24300000071525574 z:10.319000244140625
- residue HOH has 1 atoms
  - atom: O x: -18.378000259399414 y:4.208000183105469 z:55.77299880981445
- residue HOH has 1 atoms
  - atom: O x: -17.55299949645996 y:3.2249999046325684 z:51.25199890136719
- residue HOH has 1 atoms
  - atom: O x: 9.491000175476074 y:-6.734000205993652 z:43.38999938964844
- residue HOH has 1 atoms
  - atom: O x: 23.27199935913086 y:-28.881000518798828 z:39.137001037597656
- residue HOH has 1 atoms
  - atom: O x: -13.192999839782715 y:17.964000701904297 z:19.21299934387207
- residue HOH has 1 atoms
  - atom: O x: 21.905000686645508 y:-4.89900016784668 z:35.422000885009766
- residue HOH has 1 atoms
  - atom: O x: 2.9800000190734863 y:-5.75 z:44.49599838256836
- residue HOH has 1 atoms
  - atom: O x: 4.548999786376953 y:-9.392000198364258 z:42.6619987487793
- residue HOH has 1 atoms
  - atom: O x: 4.017000198364258 y:-12.059000015258789 z:46.9379997253418
- residue HOH has 1 atoms
  - atom: O x: 6.761000156402588 y:-12.234000205993652 z:47.72800064086914
- residue HOH has 1 atoms
  - atom: O x: 5.676000118255615 y:-16.034000396728516 z:45.32099914550781
- residue HOH has 1 atoms
  - atom: O x: -7.611999988555908 y:-15.508000373840332 z:46.83700180053711
- residue HOH has 1 atoms
  - atom: O x: 10.524999618530273 y:-10.456999778747559 z:40.52899932861328
- residue HOH has 1 atoms
  - atom: O x: 6.034999847412109 y:-0.2840000092983246 z:30.01300048828125
- residue HOH has 1 atoms
  - atom: O x: 7.501999855041504 y:-17.215999603271484 z:20.152000427246094
- residue HOH has 1 atoms
  - atom: O x: 7.015999794006348 y:-19.85099983215332 z:17.801000595092773
```

```
- residue HOH has 1 atoms
  - atom: O x: 10.140999794006348 y:-20.489999771118164 z:16.503999710083008
- residue HOH has 1 atoms
  - atom: O x: -28.062999725341797 y:-9.74899959564209 z:39.11600112915039
- residue HOH has 1 atoms
  - atom: O x: -13.135000228881836 y:6.835000038146973 z:48.64500045776367
- residue HOH has 1 atoms
  - atom: O x: -12.461000442504883 y:13.706000328063965 z:21.788999557495117
- residue HOH has 1 atoms
  - atom: O x: -21.52199935913086 y:-11.213000297546387 z:47.9640007019043
- residue HOH has 1 atoms
  - atom: O x: -22.802000045776367 y:-13.795999526977539 z:49.27899932861328
- residue HOH has 1 atoms
  - atom: O x: 17.39699935913086 y:-18.308000564575195 z:38.902000427246094
- residue HOH has 1 atoms
  - atom: O x: 13.23799991607666 y:-16.785999298095703 z:38.619998931884766
- residue HOH has 1 atoms
  - atom: O x: 11.210000038146973 y:-8.220000267028809 z:18.395999908447266
- residue HOH has 1 atoms
  - atom: O x: -1.9579999446868896 y:10.479000091552734 z:54.09700012207031
- residue HOH has 1 atoms
  - atom: O x: 18.85700035095215 y:-12.720999717712402 z:17.263999938964844
```

Once we have the coordinates of an atom we can compute distances between atoms or angles. We can also align two structures rototranslating the two to minimize their distance. We will not cover this and many other features that are provided by Biopython, such as Pyhlogentic analysis tools, interface towards pathways in KEGG, clustering, etc. If you are interested can find all the features available in the Biopython tutorial[108].

12.7 Exercises

1. Write a python script that retrieves all the information present in SRA regarding PacBio sequencing performed on E.coli strain K12 (query term is "E.coli K12 wgs PacBio"). Print the number of results and for each id report the title, the accession id, the total number of spots and total number of bases sequenced.

Sample output:

```
Entries found: 10
Results for id: 6705337
 - acc="SRR8154667"
 - total_spots="163482"
 - total_bases="1561717136"
Results for id: 6705336
 - acc="SRR8154668"
 - total_spots="163482"
 - total_bases="897324802"
Results for id: 6705335
 - acc="SRR8154669"
 - total_spots="163482"
 - total_bases="1570240924"

 ...
 ...
```

Show/Hide Solution

[108] http://biopython.org/DIST/docs/tutorial/Tutorial.pdf

```
In [72]: %reset -s -f

         from Bio import Entrez

         Entrez.email = "my_email"
         handle = Entrez.esearch(db="sra", term="E.coli K12 wgs PacBio", retmax = 100)
         res = Entrez.read(handle)
         #uncomment to see all fields:
         #for el in res.keys():
         #    print(el , " : ", res[el])
         print("Entries found: {}".format(res["Count"]))


         for ids in res["IdList"]:
             print("Results for id:", ids)
             handle = Entrez.esummary(db="sra",  id = ids)
             res = Entrez.read(handle)
             #uncomment to see all info
             #print(res)
             for r in res:
                 info = r['ExpXml']
                 #print(info)
                 runs = r['Runs']
                 #some text parsing to do:
                 title = info.split("Title>")
                 #print(title)
                 print(title[1][:-2])
                 #print(runs)
                 r = runs.split(" ")
                 #print(r)
                 print(" - {}\n - {}\n - {}".format(r[1],
                                                    r[2],
                                                    r[3]))
```

```
Entries found: 10
Results for id: 6705337
PacBio RSII sequencing of E. coli K12
 - acc="SRR8154667"
 - total_spots="163482"
 - total_bases="1561717136"
Results for id: 6705336
PacBio RSII sequencing of E. coli K12
 - acc="SRR8154668"
 - total_spots="163482"
 - total_bases="897324802"
Results for id: 6705335
PacBio RSII sequencing of E. coli K12
 - acc="SRR8154669"
 - total_spots="163482"
 - total_bases="1570240924"
Results for id: 6705329
PacBio RSII sequencing of E. coli K12
 - acc="SRR8154675"
 - total_spots="163482"
 - total_bases="1765639407"
Results for id: 357838
E. coli K12 PacBio RS C2 CCS sequencing
 - acc="SRR801672"
 - total_spots="81741"
```

```
 - total_bases="247537688"
Results for id: 357018
E. coli K12 MG1655 PacBio RS C2 Sequencing
 - acc="SRR797943"
 - total_spots="81741"
 - total_bases="207734764"
Results for id: 357016
E. Coli K12 454 Sequencing To Use For PacBio RS Correction
 - acc="SRR797242"
 - total_spots="1385764"
 - total_bases="720878709"
Results for id: 285699
WGA E. coli genome DNA sequenced by C2 chemistry (Null)
 - acc="SRR631042"
 - total_spots="81741"
 - total_bases="174897723"
Results for id: 285698
Native E. coli genome DNA sequenced by C2 chemistry
 - acc="SRR631040"
 - total_spots="81741"
 - total_bases="109074629"
Results for id: 285697
WGA E. coli genome DNA sequenced by FCR chemistry
 - acc="SRR644306"
 - total_spots="60667"
 - total_bases="26862325"
```

2. Write a python function that reads all the entries of a blast alignment file in .xml format (like blast_res_apple.xml and outputs all the HSPs (see example below) having bitscore > B, alignment length > A and minimum percentage of identity > I, where B, A and I are input thresholds. Hint: implement a filtering function: *filterHSPs(align, minBitscore = 0, minAlignLen = 0, minPercIdent = 0.1)*.

```
Alignments of MDC020656.85
    MDC020656.85: 1939-2593
    gi|125995253|dbj|AB270792.1|: 201263-201917
    Score:820.917 AlignLen:579 Id/Len:0.8812785388127854
    MDC020656.85: 1446-1935
    gi|125995253|dbj|AB270792.1|: 306490-306017
    Score:582.873 AlignLen:428 Id/Len:0.8629032258064516
    ....
    ....
```

that is reporting the HSP with query start-end position, subject start-end position, score, alignment length and number of identities / alignment length.

Show/Hide Solution

```
In [55]: %reset -s -f

         from Bio.Blast import NCBIXML


         def filterHSPs(align, minBitscore = 0, minAlignLen = 0, minPercIdent = 0.1):
             ret = []

             for h in align.hsps:
                     b = h.bits
                     i = h.identities
                     al = h.align_length
                     toOut = ((b > minBitscore) and
```

```
                            (al > minAlignLen) and
                            (i/al > minPercIdent))



                    if(toOut):
                        qs = h.query_start
                        ss = h.sbjct_start
                        qe = h.query_end
                        se = h.sbjct_end
                        ret.append([qs,qe, ss,se, b, i, al])

            return ret

        result_handle = open("file_samples/blast_res_apple.xml")


        for res in NCBIXML.parse(result_handle):
            print("Alignments of {}".format(res.query))
            for align in res.alignments:
                filtered = filterHSPs(align, 300, 50, 0.8)
                if(len(filtered) > 0):
                    for h in filtered:
                        title = align.title.split( )[0]
                        print("\t{}: {}-{}".format(res.query,h[0],h[1]))
                        print("\t{}: {}-{}".format(title,h[2],h[3]))
                        print("\tScore:{} AlignLen:{} Id/Len:{}".format(
                                                                    h[4],
                                                                    h[5],
                                                                    h[5]/h[6]
                                                                   ))




        result_handle.close()
```

```
Alignments of MDC020656.85
        MDC020656.85: 1939-2593
        gi|125995253|dbj|AB270792.1|: 201263-201917
        Score:820.917 AlignLen:579 Id/Len:0.8812785388127854
        MDC020656.85: 1446-1935
        gi|125995253|dbj|AB270792.1|: 306490-306017
        Score:582.873 AlignLen:428 Id/Len:0.8629032258064516
        MDC020656.85: 1447-1935
        gi|125995253|dbj|AB270792.1|: 202375-202869
        Score:543.199 AlignLen:424 Id/Len:0.848
        MDC020656.85: 1031-1475
        gi|125995253|dbj|AB270792.1|: 201861-202300
        Score:417.864 AlignLen:370 Id/Len:0.8078602620087336
Alignments of MDC001115.177
        MDC001115.177: 93-1292
        gi|125995253|dbj|AB270792.1|: 116873-118061
        Score:1196.92 AlignLen:1007 Id/Len:0.8186991869918699
        MDC001115.177: 2777-3118
        gi|125995253|dbj|AB270792.1|: 124753-125093
        Score:419.668 AlignLen:301 Id/Len:0.8775510204081632
Alignments of MDC013284.379
```

```
        MDC013284.379: 821-1195
        gi|1632177573|ref|XM_008340665.3|: 1263-1637
        Score:677.55 AlignLen:375 Id/Len:1.0
        MDC013284.379: 450-723
        gi|1632177573|ref|XM_008340665.3|: 994-1267
        Score:495.409 AlignLen:274 Id/Len:1.0
        MDC013284.379: 96-348
        gi|1632177573|ref|XM_008340665.3|: 746-998
        Score:457.539 AlignLen:253 Id/Len:1.0
        MDC013284.379: 823-1121
        gi|1632246347|ref|XM_029093378.1|: 1290-1588
        Score:450.325 AlignLen:279 Id/Len:0.9331103678929766
        MDC013284.379: 456-723
        gi|1632246347|ref|XM_029093378.1|: 1025-1292
        Score:398.929 AlignLen:249 Id/Len:0.9291044776119403
        MDC013284.379: 96-358
        gi|1632246347|ref|XM_029093378.1|: 759-1021
        Score:376.387 AlignLen:241 Id/Len:0.9163498098859315
        MDC013284.379: 3264-3519
        gi|1632172110|ref|XM_029088641.1|: 3-257
        Score:442.21 AlignLen:252 Id/Len:0.984375
        MDC013284.379: 4187-4398
        gi|1632172110|ref|XM_029088641.1|: 255-466
        Score:383.6 AlignLen:212 Id/Len:1.0
        MDC013284.379: 4477-4657
        gi|1632172110|ref|XM_029088641.1|: 463-643
        Score:327.696 AlignLen:181 Id/Len:1.0
        MDC013284.379: 4187-4398
        gi|1632172108|ref|XM_029088637.1|: 205-416
        Score:383.6 AlignLen:212 Id/Len:1.0
        MDC013284.379: 3312-3519
        gi|1632172108|ref|XM_029088637.1|: 1-207
        Score:355.648 AlignLen:204 Id/Len:0.9807692307692307
        MDC013284.379: 4477-4657
        gi|1632172108|ref|XM_029088637.1|: 413-593
        Score:327.696 AlignLen:181 Id/Len:1.0
        MDC013284.379: 4180-4398
        gi|1632172509|ref|XM_029089075.1|: 346-564
        Score:342.123 AlignLen:207 Id/Len:0.9452054794520548
        MDC013284.379: 4477-4657
        gi|1632172509|ref|XM_029089075.1|: 561-741
        Score:309.662 AlignLen:177 Id/Len:0.9779005524861878
        MDC013284.379: 4180-4398
        gi|1632172507|ref|XM_029089065.1|: 366-584
        Score:342.123 AlignLen:207 Id/Len:0.9452054794520548
        MDC013284.379: 4477-4657
        gi|1632172507|ref|XM_029089065.1|: 581-761
        Score:309.662 AlignLen:177 Id/Len:0.9779005524861878
        MDC013284.379: 4188-4398
        gi|1632172505|ref|XM_029089057.1|: 252-462
        Score:341.221 AlignLen:202 Id/Len:0.957345971563981
        MDC013284.379: 4477-4657
        gi|1632172505|ref|XM_029089057.1|: 459-639
        Score:309.662 AlignLen:177 Id/Len:0.9779005524861878
Alignments of MDC018185.243
        MDC018185.243: 19988-22671
        gi|1330174482|gb|KF530876.2|: 292955-295629
        Score:3686.47 AlignLen:2431 Id/Len:0.9043898809523809
        MDC018185.243: 20998-22671
```

```
gi|1330174482|gb|KF530876.2|: 66778-65106
Score:2140.08 AlignLen:1479 Id/Len:0.8835125448028673
MDC018185.243: 19983-21003
gi|1330174482|gb|KF530876.2|: 69724-68700
Score:1150.03 AlignLen:873 Id/Len:0.8508771929824561
MDC018185.243: 19988-20502
gi|1330174482|gb|KF530876.2|: 301521-302035
Score:710.01 AlignLen:465 Id/Len:0.9029126213592233
MDC018185.243: 19988-20502
gi|1330174482|gb|KF530876.2|: 138308-138818
Score:704.6 AlignLen:465 Id/Len:0.9029126213592233
MDC018185.243: 19988-20502
gi|1330174482|gb|KF530876.2|: 55834-55319
Score:672.139 AlignLen:458 Id/Len:0.8875968992248062
MDC018185.243: 20211-20452
gi|1330174482|gb|KF530876.2|: 147775-147534
Score:370.977 AlignLen:227 Id/Len:0.9380165289256198
MDC018185.243: 20211-20452
gi|1330174482|gb|KF530876.2|: 342492-342251
Score:370.977 AlignLen:227 Id/Len:0.9380165289256198
MDC018185.243: 19988-22671
gi|530721664|gb|KF530875.1|: 284755-287429
Score:3686.47 AlignLen:2431 Id/Len:0.9043898809523809
MDC018185.243: 20998-22671
gi|530721664|gb|KF530875.1|: 66778-65106
Score:2140.08 AlignLen:1479 Id/Len:0.8835125448028673
MDC018185.243: 19983-21003
gi|530721664|gb|KF530875.1|: 69724-68700
Score:1150.03 AlignLen:873 Id/Len:0.8508771929824561
MDC018185.243: 19988-20502
gi|530721664|gb|KF530875.1|: 293321-293835
Score:710.01 AlignLen:465 Id/Len:0.9029126213592233
MDC018185.243: 19988-20502
gi|530721664|gb|KF530875.1|: 138308-138818
Score:706.404 AlignLen:465 Id/Len:0.9029126213592233
MDC018185.243: 19988-20502
gi|530721664|gb|KF530875.1|: 55834-55319
Score:672.139 AlignLen:458 Id/Len:0.8875968992248062
MDC018185.243: 20211-20452
gi|530721664|gb|KF530875.1|: 147775-147534
Score:372.78 AlignLen:227 Id/Len:0.9380165289256198
MDC018185.243: 20211-20452
gi|530721664|gb|KF530875.1|: 334292-334051
Score:370.977 AlignLen:227 Id/Len:0.9380165289256198
MDC018185.243: 19988-22671
gi|1183577511|dbj|LC050646.1|: 76338-79012
Score:3686.47 AlignLen:2431 Id/Len:0.9043898809523809
MDC018185.243: 19988-20502
gi|1183577511|dbj|LC050646.1|: 84903-85417
Score:710.01 AlignLen:465 Id/Len:0.9029126213592233
MDC018185.243: 20211-20452
gi|1183577511|dbj|LC050646.1|: 125887-125646
Score:370.977 AlignLen:227 Id/Len:0.9380165289256198
MDC018185.243: 19988-22671
gi|1183577509|dbj|LC050645.1|: 58043-60717
Score:3686.47 AlignLen:2431 Id/Len:0.9043898809523809
MDC018185.243: 19988-20502
gi|1183577509|dbj|LC050645.1|: 66608-67122
Score:710.01 AlignLen:465 Id/Len:0.9029126213592233
```

```
MDC018185.243: 20211-20452
gi|1183577509|dbj|LC050645.1|: 107592-107351
Score:370.977 AlignLen:227 Id/Len:0.9380165289256198
MDC018185.243: 19988-22671
gi|571026231|emb|HF968765.1|: 83128-85802
Score:3686.47 AlignLen:2431 Id/Len:0.9043898809523809
MDC018185.243: 19988-20502
gi|571026231|emb|HF968765.1|: 91693-92207
Score:710.01 AlignLen:465 Id/Len:0.9029126213592233
MDC018185.243: 20211-20452
gi|571026231|emb|HF968765.1|: 132677-132436
Score:370.977 AlignLen:227 Id/Len:0.9380165289256198
MDC018185.243: 19988-22671
gi|571026244|emb|HF968766.1|: 27097-29772
Score:3663.03 AlignLen:2425 Id/Len:0.9021577380952381
MDC018185.243: 19988-20502
gi|571026244|emb|HF968766.1|: 35546-36060
Score:710.01 AlignLen:465 Id/Len:0.9029126213592233
MDC018185.243: 20211-20452
gi|571026244|emb|HF968766.1|: 48521-48779
Score:340.32 AlignLen:228 Id/Len:0.8803088803088803
MDC018185.243: 19987-22187
gi|125995253|dbj|AB270792.1|: 219104-221312
Score:2290.66 AlignLen:1843 Id/Len:0.8324299909665763
MDC018185.243: 19988-20502
gi|125995253|dbj|AB270792.1|: 304541-304027
Score:691.977 AlignLen:461 Id/Len:0.8951456310679612
MDC018185.243: 19988-20502
gi|125995253|dbj|AB270792.1|: 231533-232049
Score:679.353 AlignLen:461 Id/Len:0.8916827852998066
MDC018185.243: 20211-20502
gi|125995253|dbj|AB270792.1|: 33979-34270
Score:425.078 AlignLen:269 Id/Len:0.9212328767123288
MDC018185.243: 20211-20452
gi|125995253|dbj|AB270792.1|: 46757-46998
Score:375.485 AlignLen:228 Id/Len:0.9421487603305785
MDC018185.243: 10123-11112
gi|1632238753|ref|XM_008392551.3|: 990-1
Score:1786.62 AlignLen:990 Id/Len:1.0
MDC018185.243: 7120-7915
gi|1632238753|ref|XM_008392551.3|: 829-30
Score:1171.67 AlignLen:746 Id/Len:0.9255583126550868
MDC018185.243: 6844-7250
gi|1632238753|ref|XM_008392551.3|: 990-577
Score:551.314 AlignLen:373 Id/Len:0.894484412470024
MDC018185.243: 10284-11129
gi|1632247443|ref|XM_008389659.3|: 835-1
Score:1227.58 AlignLen:782 Id/Len:0.9210836277974087
MDC018185.243: 7034-7920
gi|1632247443|ref|XM_008389659.3|: 921-43
Score:1176.18 AlignLen:799 Id/Len:0.8907469342251951
MDC018185.243: 10322-10536
gi|1632247443|ref|XM_008389659.3|: 921-707
Score:312.367 AlignLen:198 Id/Len:0.9209302325581395
MDC018185.243: 19987-20963
gi|117938437|gb|DQ886413.1|: 981-1
Score:1098.64 AlignLen:833 Id/Len:0.8491335372069317
MDC018185.243: 10413-11185
gi|1632262963|ref|XM_008363755.3|: 769-2
```

```
Score:946.252 AlignLen:680 Id/Len:0.8706786171574904
MDC018185.243: 7251-7918
gi|1632262963|ref|XM_008363755.3|: 770-104
Score:801.08 AlignLen:586 Id/Len:0.8643067846607669
MDC018185.243: 6277-6753
gi|1632238751|ref|XM_008392549.3|: 527-1003
Score:861.493 AlignLen:477 Id/Len:1.0
MDC018185.243: 9579-10031
gi|1632238751|ref|XM_008392549.3|: 543-1003
Score:736.159 AlignLen:440 Id/Len:0.9544468546637744
MDC018185.243: 4598-4897
gi|1632238751|ref|XM_008392549.3|: 1-300
Score:539.592 AlignLen:299 Id/Len:0.9966666666666667
MDC018185.243: 19987-20502
gi|313184264|emb|FN823234.1|: 66377-65862
Score:734.356 AlignLen:471 Id/Len:0.9127906976744186
MDC018185.243: 19990-20505
gi|313184264|emb|FN823234.1|: 64635-65166
Score:699.19 AlignLen:472 Id/Len:0.8838951310861424
MDC018185.243: 19988-20502
gi|114199045|gb|DQ886019.1|: 1826-2338
Score:711.814 AlignLen:466 Id/Len:0.9048543689320389
MDC018185.243: 19988-20502
gi|114199049|gb|DQ886021.1|: 1342-1870
Score:698.288 AlignLen:469 Id/Len:0.8865784499054821
MDC018185.243: 3417-3881
gi|1632263306|ref|XM_029096579.1|: 468-9
Score:628.859 AlignLen:420 Id/Len:0.8993576017130621
MDC018185.243: 19987-20453
gi|1227417682|gb|KX822762.1|: 468-1
Score:617.137 AlignLen:417 Id/Len:0.8910256410256411
MDC018185.243: 12968-13397
gi|1632255190|ref|XR_003769783.1|: 3194-2754
Score:499.016 AlignLen:379 Id/Len:0.8574660633484162
MDC018185.243: 17934-18409
gi|1632228214|ref|XR_003766910.1|: 1322-1795
Score:426.881 AlignLen:387 Id/Len:0.80625
MDC018185.243: 20211-20452
gi|313184297|emb|FN823235.1|: 121925-122166
Score:379.994 AlignLen:229 Id/Len:0.9462809917355371
MDC018185.243: 20211-20452
gi|313184297|emb|FN823235.1|: 5080-5321
Score:352.943 AlignLen:223 Id/Len:0.9214876033057852
MDC018185.243: 20211-20452
gi|339431357|gb|HQ445899.1|: 69560-69801
Score:370.977 AlignLen:227 Id/Len:0.9380165289256198
MDC018185.243: 20211-20452
gi|86438843|emb|AM167520.1|: 28866-29107
Score:370.977 AlignLen:227 Id/Len:0.9380165289256198
MDC018185.243: 20211-20452
gi|86438843|emb|AM167520.1|: 38530-38289
Score:370.977 AlignLen:227 Id/Len:0.9380165289256198
MDC018185.243: 18146-18425
gi|1632218186|ref|XR_003776530.1|: 344-624
Score:348.435 AlignLen:246 Id/Len:0.8754448398576512
MDC018185.243: 18131-18425
gi|1632219812|ref|XR_003776654.1|: 430-723
Score:336.713 AlignLen:254 Id/Len:0.8581081081081081
MDC018185.243: 18131-18425
```

```
        gi|1632219810|ref|XR_003776652.1|: 659-952
        Score:336.713 AlignLen:254 Id/Len:0.8581081081081081
        MDC018185.243: 20211-20452
        gi|541903978|gb|KF055410.1|: 30953-31211
        Score:335.811 AlignLen:227 Id/Len:0.8764478764478765
        MDC018185.243: 1035-1376
        gi|1632268443|ref|XR_001790922.2|: 431-84
        Score:324.089 AlignLen:286 Id/Len:0.8125
        MDC018185.243: 1035-1342
        gi|1632198832|ref|XR_003774309.1|: 355-49
        Score:317.778 AlignLen:258 Id/Len:0.8349514563106796
        MDC018185.243: 18130-18436
        gi|1632207113|ref|XR_003775191.1|: 469-777
        Score:306.957 AlignLen:255 Id/Len:0.8225806451612904
Alignments of MDC018185.241
        MDC018185.241: 1952-3265
        gi|1632261394|ref|XM_008342019.3|: 1314-1
        Score:2350.17 AlignLen:1311 Id/Len:0.996958174904943
        MDC018185.241: 1952-3399
        gi|1632240204|ref|XM_008391913.3|: 1448-22
        Score:1505.3 AlignLen:1236 Id/Len:0.8218085106382979
        MDC018185.241: 10688-11460
        gi|1632257016|ref|XM_017323798.2|: 929-160
        Score:1330.37 AlignLen:759 Id/Len:0.981888745148771
        MDC018185.241: 9977-10496
        gi|1632257016|ref|XM_017323798.2|: 1421-902
        Score:928.218 AlignLen:516 Id/Len:0.9923076923076923
        MDC018185.241: 10688-11454
        gi|1632257020|ref|XM_029095849.1|: 907-144
        Score:1319.55 AlignLen:753 Id/Len:0.9817470664928292
        MDC018185.241: 9977-10496
        gi|1632257020|ref|XM_029095849.1|: 1399-880
        Score:928.218 AlignLen:516 Id/Len:0.9923076923076923
        MDC018185.241: 10688-11409
        gi|1632257018|ref|XM_008342020.3|: 878-160
        Score:1241.1 AlignLen:709 Id/Len:0.981994459833795
        MDC018185.241: 9977-10496
        gi|1632257018|ref|XM_008342020.3|: 1370-851
        Score:928.218 AlignLen:516 Id/Len:0.9923076923076923
        MDC018185.241: 10686-11459
        gi|1632240978|ref|XM_008391689.3|: 886-123
        Score:936.333 AlignLen:674 Id/Len:0.8696774193548387
        MDC018185.241: 10214-10496
        gi|1632240978|ref|XM_008391689.3|: 1128-857
        Score:330.401 AlignLen:243 Id/Len:0.8526315789473684
        MDC018185.241: 22418-23085
        gi|1632220941|ref|XR_003776790.1|: 723-22
        Score:807.392 AlignLen:598 Id/Len:0.844632768361582
        MDC018185.241: 18138-18558
        gi|1632263230|ref|XM_008342179.2|: 351-771
        Score:746.078 AlignLen:417 Id/Len:0.9904988123515439
        MDC018185.241: 5323-5766
        gi|1632233144|ref|XR_003767438.1|: 458-902
        Score:630.662 AlignLen:408 Id/Len:0.912751677852349
        MDC018185.241: 12707-13182
        gi|1632266554|ref|XM_017328275.2|: 4-475
        Score:618.94 AlignLen:426 Id/Len:0.8930817610062893
        MDC018185.241: 12979-13399
        gi|1632238051|ref|XM_008391090.3|: 166-581
```

```
Score:426.881 AlignLen:350 Id/Len:0.8293838862559242
MDC018185.241: 21727-22114
gi|1632209355|ref|XR_003775372.1|: 1708-1317
Score:414.258 AlignLen:331 Id/Len:0.8275
MDC018185.241: 5340-5765
gi|1632267776|ref|XR_003770870.1|: 1717-1294
Score:396.224 AlignLen:355 Id/Len:0.8179723502304147
```

3. Write a python function `retrieve_sequences(search_term, number, outfile)` that retrieves the first `number` of sequences from NCBI's "nucleotide" database having a search term `term` (hint: use term and retmax parameters of Entrez.esearch) and stores them in a fasta file `outfile` (hint: use SeqIO.write). Test your code retrieving the first 5 entries having search term "starch AND Malus Domestica [Organism]"

Show/Hide Solution

```
In [56]: %reset -s -f
```

```
from Bio import Entrez
from Bio import SeqIO

def retrieve_sequences(search_term, number, filename):
    Entrez.email = "my_email"
    handle = Entrez.esearch(db="nucleotide",
                            term=search_term,
                            retmax=5)
    res = Entrez.read(handle)
    records = []
    for el in res["IdList"]:
        handle = Entrez.efetch(db="nucleotide",
                               id=el,
                               rettype = "gb",
                               retmode="text")
        my_seq = SeqIO.read(handle, format = "genbank")
        records.append(my_seq)
    N = SeqIO.write(records, filename, "fasta")
    print("Search term was: ", search_term)
    print("{} sequences written to {}".format(N,filename))

s_term = "starch AND Malus Domestica [Organism]"
retrieve_sequences(s_term, 5, "file_samples/starch_sequences.fasta")
```

```
Search term was:  starch AND Malus Domestica [Organism]
5 sequences written to file_samples/starch_sequences.fasta
```

4. Write a python function that aligns the sequences in the file created in exercise 3. (here you can find mine) against the NCBI nr database limiting the hits to the Malus Domestica organism (parameter entrez_query='"Malus Domestica" [Organism]' in qblast)and prints to screen the following info for each hsp:
   (a) The title;
   (b) Score and e-value;
   (c) The number of alignments on the same subject, the number of identities and positives and the alignment length;
   (d) The number of mismatches and the list of their positions (hint: you can use the match string and look for " ").

Show/Hide Solution

```
In [57]: from Bio.Blast import NCBIWWW
```

```
from Bio.Blast import NCBIXML

fasta_string = open("file_samples/starch_sequences.fasta").read()

res_handle = NCBIWWW.qblast("blastn", "nt", fasta_string,
                            entrez_query='"Malus Domestica" [Organism]'
                           )

for align in NCBIXML.parse(res_handle):

    for a in align.alignments:
            print("Align Title:{}".format(a.title))

            for h in a.hsps:
                s = h.score
                e = h.expect
                n = h.num_alignments
                i = h.identities
                p = h.positives
                m = h.match
                al = h.align_length
                misM = [str(x) for x in range(len(m)) if m[x] == " "]
                print("Score: {} E-val: {}".format(s,e))
                print("N.aligns:{} Ident:{} Pos.:{} Align len:{}".format(
                    n,i,p,al))
                if(len(misM)):
                    print("Num mismatches:",len(misM))
                    print("Mismatch pos:", ",".join(misM))
                else:
                    print("No mismatches")
                print("")


res_handle.close()
```

```
Align Title:gi|1040975204|ref|NM_001328845.1| Malus domestica glucose-1-phosphate adenylyltransferase
Score: 3630.0 E-val: 0.0
N.aligns:None Ident:1815 Pos.:1815 Align len:1815
No mismatches

Align Title:gi|1632214376|ref|XM_029108108.1| PREDICTED: Malus domestica glucose-1-phosphate adenylyl
Score: 3390.0 E-val: 0.0
N.aligns:None Ident:1767 Pos.:1767 Align len:1815
Num mismatches: 48
Mismatch pos: 8,16,17,18,19,21,28,31,47,51,56,58,71,72,74,91,104,106,110,128,134,162,167,180,187,190,

Align Title:gi|1632265869|ref|XM_008344054.3| PREDICTED: Malus domestica glucose-1-phosphate adenylyl
Score: 3290.0 E-val: 0.0
N.aligns:None Ident:1749 Pos.:1749 Align len:1818
Num mismatches: 69
Mismatch pos: 824,860,867,942,1031,1044,1122,1157,1190,1277,1295,1367,1371,1376,1382,1388,1394,1412,1

Align Title:gi|1632265871|ref|XR_003770676.1| PREDICTED: Malus domestica glucose-1-phosphate adenylyl
Score: 2455.0 E-val: 0.0
N.aligns:None Ident:1241 Pos.:1241 Align len:1250
Num mismatches: 9
Mismatch pos: 824,860,867,942,1031,1044,1122,1157,1190
```

```
Score: 843.0 E-val: 0.0
N.aligns:None Ident:512 Pos.:512 Align len:572
Num mismatches: 60
Mismatch pos: 31,49,121,125,130,136,142,148,166,199,211,231,337,338,339,343,364,368,369,371,376,386,

Align Title:gi|1632255889|ref|XM_008342710.3| PREDICTED: Malus domestica glucose-1-phosphate adenylyl
Score: 140.0 E-val: 9.46061e-28
N.aligns:None Ident:423 Pos.:423 Align len:651
Num mismatches: 228
Mismatch pos: 2,5,11,14,20,26,27,34,39,41,49,56,59,72,74,75,76,84,85,86,98,102,108,110,116,120,128,12

Score: 42.0 E-val: 0.3771
N.aligns:None Ident:33 Pos.:33 Align len:41
Num mismatches: 8
Mismatch pos: 8,10,11,12,16,17,29,35

Align Title:gi|1632239043|ref|XM_029091606.1| PREDICTED: Malus domestica glucose-1-phosphate adenylyl
Score: 127.0 E-val: 5.9703e-24
N.aligns:None Ident:131 Pos.:131 Align len:176
Num mismatches: 45
Mismatch pos: 2,5,11,14,23,26,27,34,39,41,49,56,59,72,74,79,81,82,83,95,99,101,105,107,113,117,125,12

Score: 68.0 E-val: 3.305e-08
N.aligns:None Ident:237 Pos.:237 Align len:366
Num mismatches: 129
Mismatch pos: 5,14,17,26,34,36,37,38,42,44,53,57,58,59,60,61,65,67,68,69,71,74,75,78,80,81,84,89,90,9

Align Title:gi|1632170348|ref|XM_029108064.1| PREDICTED: Malus domestica inactive glucose-1-phosphate
Score: 114.0 E-val: 1.07946e-20
N.aligns:None Ident:244 Pos.:244 Align len:368
Num mismatches: 124
Mismatch pos: 6,15,18,22,23,24,25,26,27,30,33,36,40,42,44,52,54,55,56,58,60,63,66,68,69,72,75,78,81,8

Score: 84.0 E-val: 1.50047e-12
N.aligns:None Ident:90 Pos.:90 Align len:122
Num mismatches: 32
Mismatch pos: 8,12,13,14,23,29,32,35,37,38,41,47,48,52,66,67,68,70,77,80,86,89,90,92,93,95,97,98,102,

Align Title:gi|1632253863|ref|XM_008389820.3| PREDICTED: Malus domestica glucose-1-phosphate adenylyl
Score: 105.0 E-val: 5.59172e-18
N.aligns:None Ident:497 Pos.:497 Align len:786
Num mismatches: 289
Mismatch pos: 4,5,9,11,14,20,23,29,30,32,37,38,39,40,41,53,63,65,68,75,77,83,84,85,87,89,90,91,92,98,

Align Title:gi|1632247379|ref|XM_008338947.3| PREDICTED: Malus domestica inactive glucose-1-phosphate
Score: 104.0 E-val: 5.59172e-18
N.aligns:None Ident:233 Pos.:233 Align len:354
Num mismatches: 121
Mismatch pos: 6,15,18,22,23,24,25,26,27,30,33,36,40,42,44,52,54,55,56,58,60,63,68,69,72,75,79,81,84,8

Score: 70.0 E-val: 9.46897e-09
N.aligns:None Ident:86 Pos.:86 Align len:120
Num mismatches: 34
Mismatch pos: 6,10,11,12,21,27,30,33,35,36,39,40,42,45,46,50,64,65,66,68,75,78,84,87,90,91,93,95,96,1

Align Title:gi|1632219716|ref|XM_008379059.3| PREDICTED: Malus domestica glucose-1-phosphate adenylyl
Score: 102.0 E-val: 1.9517e-17
N.aligns:None Ident:345 Pos.:345 Align len:528
```

```
Num mismatches: 183
Mismatch pos: 4,5,8,11,14,18,20,25,27,28,33,38,41,44,47,51,53,56,59,63,74,75,77,78,79,80,86,88,89,92,

Score: 59.0 E-val: 1.71203e-05
N.aligns:None Ident:133 Pos.:133 Align len:202
Num mismatches: 69
Mismatch pos: 2,10,13,19,22,23,25,31,32,37,39,40,41,42,43,46,49,50,51,54,56,58,61,63,64,67,73,82,84,8

Align Title:gi|1632219715|ref|XM_008379058.3| PREDICTED: Malus domestica glucose-1-phosphate adenylyl
Score: 102.0 E-val: 1.9517e-17
N.aligns:None Ident:345 Pos.:345 Align len:528
Num mismatches: 183
Mismatch pos: 4,5,8,11,14,18,20,25,27,28,33,38,41,44,47,51,53,56,59,63,74,75,77,78,79,80,86,88,89,92,

Score: 59.0 E-val: 1.71203e-05
N.aligns:None Ident:133 Pos.:133 Align len:202
Num mismatches: 69
Mismatch pos: 2,10,13,19,22,23,25,31,32,37,39,40,41,42,43,46,49,50,51,54,56,58,61,63,64,67,73,82,84,8

Align Title:gi|1632207701|ref|XM_029106623.1| PREDICTED: Malus domestica glucose-1-phosphate adenylyl
Score: 98.0 E-val: 2.37766e-16
N.aligns:None Ident:519 Pos.:519 Align len:823
Num mismatches: 304
Mismatch pos: 7,10,13,16,19,20,22,25,27,28,31,32,34,39,40,44,46,49,55,58,64,65,67,72,73,74,75,76,88,9

Align Title:gi|1632207699|ref|XM_029106622.1| PREDICTED: Malus domestica glucose-1-phosphate adenylyl
Score: 98.0 E-val: 2.37766e-16
N.aligns:None Ident:519 Pos.:519 Align len:823
Num mismatches: 304
Mismatch pos: 7,10,13,16,19,20,22,25,27,28,31,32,34,39,40,44,46,49,55,58,64,65,67,72,73,74,75,76,88,9

Align Title:gi|1632193374|ref|XM_029103016.1| PREDICTED: Malus domestica putative cyclin-B3-1 (LOC103
Score: 50.0 E-val: 0.00254088
N.aligns:None Ident:41 Pos.:41 Align len:51
Num mismatches: 10
Mismatch pos: 7,8,9,22,23,28,32,34,43,48

Align Title:gi|1632236059|ref|XM_008390912.3| PREDICTED: Malus domestica cyclin-J18 (LOC103451508), m
Score: 44.0 E-val: 0.108041
N.aligns:None Ident:50 Pos.:50 Align len:65
Num mismatches: 15
Mismatch pos: 11,12,13,20,23,27,29,31,37,39,46,47,49,54,55

Align Title:gi|1632255893|ref|XM_029095576.1| PREDICTED: Malus domestica DEAD-box ATP-dependent RNA h
Score: 43.0 E-val: 0.3771
N.aligns:None Ident:35 Pos.:35 Align len:44
Num mismatches: 9
Mismatch pos: 2,11,13,14,15,19,20,32,38

Score: 39.0 E-val: 4.59402
N.aligns:None Ident:30 Pos.:30 Align len:37
Num mismatches: 7
Mismatch pos: 2,7,8,10,13,26,28

Align Title:gi|1632255891|ref|XM_029095575.1| PREDICTED: Malus domestica DEAD-box ATP-dependent RNA h
Score: 43.0 E-val: 0.3771
N.aligns:None Ident:35 Pos.:35 Align len:44
Num mismatches: 9
Mismatch pos: 2,11,13,14,15,19,20,32,38
```

```
Align Title:gi|1632189520|ref|XM_029102026.1| PREDICTED: Malus domestica glucose-1-phosphate adenylyl
Score: 43.0 E-val: 0.3771
N.aligns:None Ident:132 Pos.:132 Align len:203
Num mismatches: 71
Mismatch pos: 2,10,17,19,22,23,25,28,31,32,37,39,40,41,42,43,45,49,50,51,56,58,62,65,68,74,77,83,85,8

Align Title:gi|1632189519|ref|XM_008387677.3| PREDICTED: Malus domestica glucose-1-phosphate adenylyl
Score: 43.0 E-val: 0.3771
N.aligns:None Ident:132 Pos.:132 Align len:203
Num mismatches: 71
Mismatch pos: 2,10,17,19,22,23,25,28,31,32,37,39,40,41,42,43,45,49,50,51,56,58,62,65,68,74,77,83,85,8

Align Title:gi|1632212049|ref|XM_029107614.1| PREDICTED: Malus domestica probable alpha-mannosidase A
Score: 42.0 E-val: 0.3771
N.aligns:None Ident:27 Pos.:27 Align len:31
Num mismatches: 4
Mismatch pos: 3,15,17,26

Align Title:gi|1632171654|ref|XM_008388982.3| PREDICTED: Malus domestica protein phosphatase 1 regula
Score: 42.0 E-val: 0.3771
N.aligns:None Ident:24 Pos.:24 Align len:26
Num mismatches: 2
Mismatch pos: 11,12

Align Title:gi|1632171652|ref|XM_008388975.3| PREDICTED: Malus domestica protein phosphatase 1 regula
Score: 42.0 E-val: 0.3771
N.aligns:None Ident:24 Pos.:24 Align len:26
Num mismatches: 2
Mismatch pos: 11,12

Align Title:gi|1632245134|ref|XM_008362507.2| PREDICTED: Malus domestica trihelix transcription facto
Score: 40.0 E-val: 1.31621
N.aligns:None Ident:29 Pos.:29 Align len:35
Num mismatches: 6
Mismatch pos: 2,9,11,12,20,23

Align Title:gi|1632245132|ref|XM_008362506.2| PREDICTED: Malus domestica trihelix transcription facto
Score: 40.0 E-val: 1.31621
N.aligns:None Ident:29 Pos.:29 Align len:35
Num mismatches: 6
Mismatch pos: 2,9,11,12,20,23

Align Title:gi|1632251859|ref|XM_029094697.1| PREDICTED: Malus domestica ankyrin repeat-containing pr
Score: 39.0 E-val: 4.59402
N.aligns:None Ident:21 Pos.:21 Align len:22
Num mismatches: 1
Mismatch pos: 14

Align Title:gi|1632226737|ref|XM_008387075.3| PREDICTED: Malus domestica beta-galactosidase 8-like (M
Score: 39.0 E-val: 4.59402
N.aligns:None Ident:29 Pos.:29 Align len:34
Num mismatches: 5
Mismatch pos: 3,8,26,27,28

Align Title:gi|1632215401|ref|XM_008362221.3| PREDICTED: Malus domestica dentin sialophosphoprotein
Score: 39.0 E-val: 4.59402
N.aligns:None Ident:27 Pos.:27 Align len:32
Num mismatches: 5
```

```
Mismatch pos: 4,6,9,23,26

Align Title:gi|1632209843|ref|XR_003775517.1| PREDICTED: Malus domestica uncharacterized LOC108170672
Score: 39.0 E-val: 4.59402
N.aligns:None Ident:21 Pos.:21 Align len:22
Num mismatches: 1
Mismatch pos: 5

Align Title:gi|1632209842|ref|XR_003775516.1| PREDICTED: Malus domestica uncharacterized LOC108170672
Score: 39.0 E-val: 4.59402
N.aligns:None Ident:21 Pos.:21 Align len:22
Num mismatches: 1
Mismatch pos: 5

Align Title:gi|1632209839|ref|XR_003775513.1| PREDICTED: Malus domestica uncharacterized LOC108170672
Score: 39.0 E-val: 4.59402
N.aligns:None Ident:21 Pos.:21 Align len:22
Num mismatches: 1
Mismatch pos: 5

Align Title:gi|1632209837|ref|XR_003775511.1| PREDICTED: Malus domestica uncharacterized LOC108170672
Score: 39.0 E-val: 4.59402
N.aligns:None Ident:21 Pos.:21 Align len:22
Num mismatches: 1
Mismatch pos: 5

Align Title:gi|1632209835|ref|XR_003775509.1| PREDICTED: Malus domestica uncharacterized LOC108170672
Score: 39.0 E-val: 4.59402
N.aligns:None Ident:21 Pos.:21 Align len:22
Num mismatches: 1
Mismatch pos: 5

Align Title:gi|1632209834|ref|XR_003775508.1| PREDICTED: Malus domestica uncharacterized LOC108170672
Score: 39.0 E-val: 4.59402
N.aligns:None Ident:21 Pos.:21 Align len:22
Num mismatches: 1
Mismatch pos: 5

Align Title:gi|1632209833|ref|XR_003775507.1| PREDICTED: Malus domestica uncharacterized LOC108170672
Score: 39.0 E-val: 4.59402
N.aligns:None Ident:21 Pos.:21 Align len:22
Num mismatches: 1
Mismatch pos: 5

Align Title:gi|1632209832|ref|XR_003775506.1| PREDICTED: Malus domestica uncharacterized LOC108170672
Score: 39.0 E-val: 4.59402
N.aligns:None Ident:21 Pos.:21 Align len:22
Num mismatches: 1
Mismatch pos: 5

Align Title:gi|1632209831|ref|XR_003775505.1| PREDICTED: Malus domestica uncharacterized LOC108170672
Score: 39.0 E-val: 4.59402
N.aligns:None Ident:21 Pos.:21 Align len:22
Num mismatches: 1
Mismatch pos: 5

Align Title:gi|1632209828|ref|XR_003775504.1| PREDICTED: Malus domestica uncharacterized LOC108170672
Score: 39.0 E-val: 4.59402
N.aligns:None Ident:21 Pos.:21 Align len:22
```

```
Num mismatches: 1
Mismatch pos: 5

Align Title:gi|1632209820|ref|XM_029107091.1| PREDICTED: Malus domestica uncharacterized LOC108170672
Score: 39.0 E-val: 4.59402
N.aligns:None Ident:21 Pos.:21 Align len:22
Num mismatches: 1
Mismatch pos: 5

Align Title:gi|1632209819|ref|XR_003775503.1| PREDICTED: Malus domestica uncharacterized LOC108170672
Score: 39.0 E-val: 4.59402
N.aligns:None Ident:21 Pos.:21 Align len:22
Num mismatches: 1
Mismatch pos: 5

Align Title:gi|1632209818|ref|XR_003775502.1| PREDICTED: Malus domestica uncharacterized LOC108170672
Score: 39.0 E-val: 4.59402
N.aligns:None Ident:21 Pos.:21 Align len:22
Num mismatches: 1
Mismatch pos: 5

Align Title:gi|1632209811|ref|XR_003775497.1| PREDICTED: Malus domestica uncharacterized LOC108170672
Score: 39.0 E-val: 4.59402
N.aligns:None Ident:21 Pos.:21 Align len:22
Num mismatches: 1
Mismatch pos: 5

Align Title:gi|1632209810|ref|XR_003775496.1| PREDICTED: Malus domestica uncharacterized LOC108170672
Score: 39.0 E-val: 4.59402
N.aligns:None Ident:21 Pos.:21 Align len:22
Num mismatches: 1
Mismatch pos: 5

Align Title:gi|1632209806|ref|XR_003775492.1| PREDICTED: Malus domestica uncharacterized LOC108170672
Score: 39.0 E-val: 4.59402
N.aligns:None Ident:21 Pos.:21 Align len:22
Num mismatches: 1
Mismatch pos: 5

Align Title:gi|1632257928|ref|XR_524259.3| PREDICTED: Malus domestica protein transport protein Sec61
Score: 38.0 E-val: 4.59402
N.aligns:None Ident:24 Pos.:24 Align len:26
Num mismatches: 2
Mismatch pos: 7,9

Align Title:gi|1632243939|ref|XR_003768479.1| PREDICTED: Malus domestica serine/threonine protein pho
Score: 38.0 E-val: 4.59402
N.aligns:None Ident:27 Pos.:27 Align len:31
Num mismatches: 4
Mismatch pos: 6,9,13,19

Align Title:gi|1632243938|ref|XR_003768478.1| PREDICTED: Malus domestica serine/threonine protein pho
Score: 38.0 E-val: 4.59402
N.aligns:None Ident:27 Pos.:27 Align len:31
Num mismatches: 4
Mismatch pos: 6,9,13,19

Align Title:gi|1632243936|ref|XM_008394794.3| PREDICTED: Malus domestica serine/threonine protein pho
Score: 38.0 E-val: 4.59402
```

```
N.aligns:None Ident:27 Pos.:27 Align len:31
Num mismatches: 4
Mismatch pos: 6,9,13,19

Align Title:gi|1632243934|ref|XM_008394793.3| PREDICTED: Malus domestica serine/threonine protein pho
Score: 38.0 E-val: 4.59402
N.aligns:None Ident:27 Pos.:27 Align len:31
Num mismatches: 4
Mismatch pos: 6,9,13,19

Align Title:gi|1632243932|ref|XM_008394792.3| PREDICTED: Malus domestica serine/threonine protein pho
Score: 38.0 E-val: 4.59402
N.aligns:None Ident:27 Pos.:27 Align len:31
Num mismatches: 4
Mismatch pos: 6,9,13,19

Align Title:gi|1632243930|ref|XM_008394791.3| PREDICTED: Malus domestica serine/threonine protein pho
Score: 38.0 E-val: 4.59402
N.aligns:None Ident:27 Pos.:27 Align len:31
Num mismatches: 4
Mismatch pos: 6,9,13,19

Align Title:gi|658309847|ref|NM_001293907.1| Malus domestica granule-bound starch synthase 1, chloro
Score: 3696.0 E-val: 0.0
N.aligns:None Ident:1848 Pos.:1848 Align len:1848
No mismatches

Align Title:gi|1632200625|ref|XM_029104747.1| PREDICTED: Malus domestica granule-bound starch synthas
Score: 3687.0 E-val: 0.0
N.aligns:None Ident:1847 Pos.:1847 Align len:1848
Num mismatches: 1
Mismatch pos: 1743

Align Title:gi|1632200623|ref|XM_008378000.3| PREDICTED: Malus domestica granule-bound starch synthas
Score: 3687.0 E-val: 0.0
N.aligns:None Ident:1847 Pos.:1847 Align len:1848
Num mismatches: 1
Mismatch pos: 1743

Align Title:gi|1632200627|ref|XM_029104748.1| PREDICTED: Malus domestica granule-bound starch synthas
Score: 3646.0 E-val: 0.0
N.aligns:None Ident:1838 Pos.:1838 Align len:1848
Num mismatches: 10
Mismatch pos: 1732,1733,1734,1735,1736,1737,1739,1740,1741,1743

Align Title:gi|357528265|gb|JN187083.1| Malus x domestica granule-bound starch synthase GBSS4 pseudog
Score: 1457.0 E-val: 0.0
N.aligns:None Ident:774 Pos.:774 Align len:804
Num mismatches: 30
Mismatch pos: 34,35,145,213,218,270,271,272,273,332,334,340,401,403,407,410,421,527,538,560,597,689,7

Align Title:gi|1632245385|ref|XM_008394428.2| PREDICTED: Malus domestica granule-bound starch synthas
Score: 1310.0 E-val: 0.0
N.aligns:None Ident:1226 Pos.:1226 Align len:1604
Num mismatches: 378
Mismatch pos: 6,12,14,18,24,27,34,39,40,42,45,60,72,93,111,114,123,126,129,136,138,144,147,150,162,17

Align Title:gi|1632245384|ref|XM_008394427.3| PREDICTED: Malus domestica granule-bound starch synthas
Score: 1310.0 E-val: 0.0
```

```
N.aligns:None Ident:1226 Pos.:1226 Align len:1604
Num mismatches: 378
Mismatch pos: 6,12,14,18,24,27,34,39,40,42,45,60,72,93,111,114,123,126,129,136,138,144,147,150,162,17

Align Title:gi|1632195925|ref|XM_008375918.3| PREDICTED: Malus domestica granule-bound starch synthas
Score: 1253.0 E-val: 0.0
N.aligns:None Ident:1214 Pos.:1214 Align len:1605
Num mismatches: 391
Mismatch pos: 5,11,17,23,26,33,38,39,44,59,71,92,95,104,110,113,116,122,125,128,134,135,137,143,146,1

Score: 40.0 E-val: 1.31621
N.aligns:None Ident:26 Pos.:26 Align len:30
Num mismatches: 4
Mismatch pos: 2,17,19,23

Align Title:gi|1632195924|ref|XM_008375917.3| PREDICTED: Malus domestica granule-bound starch synthas
Score: 1253.0 E-val: 0.0
N.aligns:None Ident:1214 Pos.:1214 Align len:1605
Num mismatches: 391
Mismatch pos: 5,11,17,23,26,33,38,39,44,59,71,92,95,104,110,113,116,122,125,128,134,135,137,143,146,1

Score: 40.0 E-val: 1.31621
N.aligns:None Ident:26 Pos.:26 Align len:30
Num mismatches: 4
Mismatch pos: 2,17,19,23

Align Title:gi|658309783|ref|NM_001293995.1| Malus domestica granule-bound starch synthase 1, chloropl
Score: 1253.0 E-val: 0.0
N.aligns:None Ident:1214 Pos.:1214 Align len:1605
Num mismatches: 391
Mismatch pos: 5,11,17,23,26,33,38,39,44,59,71,92,95,104,110,113,116,122,125,128,134,135,137,143,146,1

Score: 40.0 E-val: 1.31621
N.aligns:None Ident:26 Pos.:26 Align len:30
Num mismatches: 4
Mismatch pos: 2,17,19,23

Align Title:gi|357528261|gb|JN187081.1| Malus x domestica granule-bound starch synthase GBSS2 gene, c
Score: 717.0 E-val: 0.0
N.aligns:None Ident:360 Pos.:360 Align len:361
Num mismatches: 1
Mismatch pos: 358

Score: 495.0 E-val: 7.73152e-124
N.aligns:None Ident:249 Pos.:249 Align len:250
Num mismatches: 1
Mismatch pos: 2

Score: 430.0 E-val: 1.81989e-106
N.aligns:None Ident:215 Pos.:215 Align len:215
No mismatches

Score: 389.0 E-val: 4.57377e-95
N.aligns:None Ident:199 Pos.:199 Align len:201
Num mismatches: 2
Mismatch pos: 5,6

Score: 356.0 E-val: 2.21904e-86
N.aligns:None Ident:178 Pos.:178 Align len:178
```

```
No mismatches

Score: 260.0 E-val: 2.53415e-60
N.aligns:None Ident:130 Pos.:130 Align len:130
No mismatches

Score: 238.0 E-val: 2.37346e-54
N.aligns:None Ident:119 Pos.:119 Align len:119
No mismatches

Score: 209.0 E-val: 3.29916e-46
N.aligns:None Ident:106 Pos.:106 Align len:107
Num mismatches: 1
Mismatch pos: 103

Score: 188.0 E-val: 8.85288e-41
N.aligns:None Ident:94 Pos.:94 Align len:94
No mismatches

Score: 174.0 E-val: 5.58678e-37
N.aligns:None Ident:87 Pos.:87 Align len:87
No mismatches

Score: 162.0 E-val: 1.01011e-33
N.aligns:None Ident:81 Pos.:81 Align len:81
No mismatches

Score: 134.0 E-val: 4.02275e-26
N.aligns:None Ident:67 Pos.:67 Align len:67
No mismatches

Align Title:gi|357528259|gb|JN187080.1| Malus x domestica granule-bound starch synthase GBSS1 gene, c
Score: 196.0 E-val: 5.96503e-43
N.aligns:None Ident:188 Pos.:188 Align len:248
Num mismatches: 60
Mismatch pos: 2,5,13,16,19,22,25,34,43,46,52,53,55,61,66,70,71,73,79,95,97,110,111,114,115,118,124,12

Score: 150.0 E-val: 1.82633e-30
N.aligns:None Ident:96 Pos.:96 Align len:110
Num mismatches: 14
Mismatch pos: 5,11,17,23,26,33,38,39,44,59,71,92,95,104

Score: 148.0 E-val: 6.37451e-30
N.aligns:None Ident:146 Pos.:146 Align len:194
Num mismatches: 48
Mismatch pos: 2,5,11,15,16,17,18,25,30,32,33,41,44,46,47,48,53,56,58,59,62,65,66,67,72,74,77,91,92,93

Score: 132.0 E-val: 1.40408e-25
N.aligns:None Ident:105 Pos.:105 Align len:131
Num mismatches: 26
Mismatch pos: 12,14,20,23,26,29,41,44,47,50,53,56,59,60,68,71,72,74,77,86,92,95,113,119,120,128

Score: 111.0 E-val: 1.31505e-19
N.aligns:None Ident:72 Pos.:72 Align len:83
Num mismatches: 11
Mismatch pos: 5,11,14,17,32,47,50,63,68,71,74

Score: 107.0 E-val: 1.60205e-18
N.aligns:None Ident:94 Pos.:94 Align len:121
```

```
Num mismatches: 27
Mismatch pos: 10,11,18,19,20,32,35,40,41,47,48,50,52,55,62,68,71,74,76,83,86,87,88,89,92,101,110

Score: 89.0 E-val: 1.23166e-13
N.aligns:None Ident:70 Pos.:70 Align len:87
Num mismatches: 17
Mismatch pos: 5,6,14,20,23,26,27,29,32,41,50,53,65,73,74,80,83

Score: 77.0 E-val: 2.22689e-10
N.aligns:None Ident:73 Pos.:73 Align len:96
Num mismatches: 23
Mismatch pos: 11,17,19,29,35,39,40,41,42,44,49,50,53,55,58,65,68,69,72,74,81,83,86

Score: 61.0 E-val: 4.90505e-06
N.aligns:None Ident:53 Pos.:53 Align len:68
Num mismatches: 15
Mismatch pos: 5,9,11,13,14,23,26,29,32,33,35,36,43,50,53

Score: 60.0 E-val: 4.90505e-06
N.aligns:None Ident:51 Pos.:51 Align len:65
Num mismatches: 14
Mismatch pos: 2,5,11,14,17,23,24,26,32,35,38,41,50,62

Score: 53.0 E-val: 0.000727974
N.aligns:None Ident:82 Pos.:82 Align len:119
Num mismatches: 37
Mismatch pos: 4,8,10,13,14,16,22,23,25,26,32,37,38,42,43,46,49,50,51,54,56,60,61,63,64,65,70,73,76,79

Score: 40.0 E-val: 1.31621
N.aligns:None Ident:26 Pos.:26 Align len:30
Num mismatches: 4
Mismatch pos: 2,17,19,23

Align Title:gi|357528263|gb|JN187082.1| Malus x domestica granule-bound starch synthase GBSS3 gene, c
Score: 181.0 E-val: 1.31388e-38
N.aligns:None Ident:185 Pos.:185 Align len:248
Num mismatches: 63
Mismatch pos: 10,13,16,19,22,25,34,43,46,52,53,55,58,61,66,67,70,71,73,79,87,94,95,110,111,114,115,11

Score: 158.0 E-val: 1.23057e-32
N.aligns:None Ident:148 Pos.:148 Align len:194
Num mismatches: 46
Mismatch pos: 5,8,15,16,17,25,30,32,38,41,45,46,47,48,53,56,58,62,65,66,67,72,74,77,90,92,95,109,110,

Score: 152.0 E-val: 5.23252e-31
N.aligns:None Ident:97 Pos.:97 Align len:111
Num mismatches: 14
Mismatch pos: 6,12,14,18,24,27,34,39,40,42,45,60,72,93

Score: 137.0 E-val: 1.15254e-26
N.aligns:None Ident:131 Pos.:131 Align len:172
Num mismatches: 41
Mismatch pos: 13,14,19,22,25,28,40,43,44,49,52,55,58,59,70,73,76,85,91,94,103,106,112,118,119,127,130

Score: 107.0 E-val: 1.60205e-18
N.aligns:None Ident:94 Pos.:94 Align len:121
Num mismatches: 27
Mismatch pos: 10,11,18,19,20,29,32,35,40,47,48,50,52,55,59,62,68,71,74,76,83,86,87,88,89,101,110
```

```
Score: 99.0 E-val: 2.37766e-16
N.aligns:None Ident:72 Pos.:72 Align len:87
Num mismatches: 15
Mismatch pos: 5,14,20,23,26,27,29,32,41,50,53,65,74,80,83

Score: 98.0 E-val: 2.37766e-16
N.aligns:None Ident:64 Pos.:64 Align len:74
Num mismatches: 10
Mismatch pos: 5,11,14,17,32,47,50,64,68,71

Score: 97.0 E-val: 8.29884e-16
N.aligns:None Ident:77 Pos.:77 Align len:96
Num mismatches: 19
Mismatch pos: 11,19,29,35,39,40,41,44,49,50,55,65,69,72,74,81,83,84,86

Score: 90.0 E-val: 3.52876e-14
N.aligns:None Ident:72 Pos.:72 Align len:90
Num mismatches: 18
Mismatch pos: 2,5,14,18,23,26,41,44,50,57,59,65,68,74,77,80,83,86

Score: 89.0 E-val: 1.23166e-13
N.aligns:None Ident:85 Pos.:85 Align len:110
Num mismatches: 25
Mismatch pos: 8,17,26,36,38,39,45,46,49,52,53,56,61,65,71,73,74,76,77,79,83,85,88,91,94

Score: 82.0 E-val: 5.23714e-12
N.aligns:None Ident:65 Pos.:65 Align len:81
Num mismatches: 16
Mismatch pos: 2,11,14,17,24,26,32,35,38,50,62,65,67,68,71,72

Align Title:gi|1632212983|ref|XM_008367179.3| PREDICTED: Malus domestica granule-bound starch synthas
Score: 54.0 E-val: 0.000208568
N.aligns:None Ident:49 Pos.:49 Align len:63
Num mismatches: 14
Mismatch pos: 4,7,10,28,31,32,34,44,45,46,49,50,51,60

Align Title:gi|1632163374|ref|XM_029098420.1| PREDICTED: Malus domestica granule-bound starch synthas
Score: 54.0 E-val: 0.000208568
N.aligns:None Ident:49 Pos.:49 Align len:63
Num mismatches: 14
Mismatch pos: 4,7,10,28,31,32,34,44,45,46,48,50,51,60

Align Title:gi|1632223611|ref|XM_029087607.1| PREDICTED: Malus domestica starch synthase 3, chloropla
Score: 46.0 E-val: 0.0309542
N.aligns:None Ident:41 Pos.:41 Align len:53
Num mismatches: 12
Mismatch pos: 2,14,17,23,24,25,26,28,29,32,41,49

Align Title:gi|1632194644|ref|XM_029103312.1| PREDICTED: Malus domestica starch synthase 3, chloropla
Score: 44.0 E-val: 0.108041
N.aligns:None Ident:55 Pos.:55 Align len:77
Num mismatches: 22
Mismatch pos: 2,6,8,12,17,18,19,20,23,26,38,41,47,48,49,50,52,53,56,62,65,73

Align Title:gi|1632206082|ref|XM_008358830.3| PREDICTED: Malus domestica starch synthase 1, chloropla
Score: 42.0 E-val: 0.3771
N.aligns:None Ident:42 Pos.:42 Align len:56
Num mismatches: 14
Mismatch pos: 5,11,17,20,22,23,26,27,30,32,35,39,41,53
```

```
Align Title:gi|1632206081|ref|XM_008358828.3| PREDICTED: Malus domestica starch synthase 1, chloropla
Score: 42.0 E-val: 0.3771
N.aligns:None Ident:42 Pos.:42 Align len:56
Num mismatches: 14
Mismatch pos: 5,11,17,20,22,23,26,27,30,32,35,39,41,53

Align Title:gi|1632224947|ref|XM_029087990.1| PREDICTED: Malus domestica sufE-like protein 1, chloro
Score: 41.0 E-val: 1.31621
N.aligns:None Ident:22 Pos.:22 Align len:23
Num mismatches: 1
Mismatch pos: 18

Align Title:gi|1632263211|ref|XM_029096539.1| PREDICTED: Malus domestica vinorine synthase-like (LOC
Score: 40.0 E-val: 1.31621
N.aligns:None Ident:23 Pos.:23 Align len:25
Num mismatches: 2
Mismatch pos: 6,10

Align Title:gi|1632254475|ref|XR_003769651.1| PREDICTED: Malus domestica neuroguidin-like (LOC103425
Score: 39.0 E-val: 4.59402
N.aligns:None Ident:21 Pos.:21 Align len:22
Num mismatches: 1
Mismatch pos: 8

Align Title:gi|1632254473|ref|XR_003769649.1| PREDICTED: Malus domestica neuroguidin-like (LOC103425
Score: 39.0 E-val: 4.59402
N.aligns:None Ident:21 Pos.:21 Align len:22
Num mismatches: 1
Mismatch pos: 8

Align Title:gi|1632195180|ref|XM_029103459.1| PREDICTED: Malus domestica uncharacterized LOC114825112
Score: 39.0 E-val: 4.59402
N.aligns:None Ident:23 Pos.:23 Align len:24
Num mismatches: 1
Mismatch pos: 9

Align Title:gi|1632266859|ref|XM_008343789.3| PREDICTED: Malus domestica probable beta-1,3-galactosyl
Score: 38.0 E-val: 4.59402
N.aligns:None Ident:25 Pos.:25 Align len:29
Num mismatches: 4
Mismatch pos: 4,6,10,14

Align Title:gi|1632266858|ref|XM_008343788.3| PREDICTED: Malus domestica probable beta-1,3-galactosyl
Score: 38.0 E-val: 4.59402
N.aligns:None Ident:25 Pos.:25 Align len:29
Num mismatches: 4
Mismatch pos: 4,6,10,14

Align Title:gi|1632242149|ref|XM_008395235.3| PREDICTED: Malus domestica UV-B-induced protein At3g178
Score: 38.0 E-val: 4.59402
N.aligns:None Ident:28 Pos.:28 Align len:34
Num mismatches: 6
Mismatch pos: 2,7,8,25,26,31

Align Title:gi|1632242147|ref|XM_008395234.3| PREDICTED: Malus domestica UV-B-induced protein At3g178
Score: 38.0 E-val: 4.59402
N.aligns:None Ident:28 Pos.:28 Align len:34
Num mismatches: 6
```

```
Mismatch pos: 2,7,8,25,26,31

Align Title:gi|1632237734|ref|XM_008392593.3| PREDICTED: Malus domestica mediator of RNA polymerase
Score: 38.0 E-val: 4.59402
N.aligns:None Ident:22 Pos.:22 Align len:24
Num mismatches: 2
Mismatch pos: 8,9

Align Title:gi|1632203209|ref|XM_029105544.1| PREDICTED: Malus domestica lysine-specific demethylase
Score: 38.0 E-val: 4.59402
N.aligns:None Ident:22 Pos.:22 Align len:24
Num mismatches: 2
Mismatch pos: 11,13

Align Title:gi|1632203207|ref|XM_029105543.1| PREDICTED: Malus domestica lysine-specific demethylase
Score: 38.0 E-val: 4.59402
N.aligns:None Ident:22 Pos.:22 Align len:24
Num mismatches: 2
Mismatch pos: 11,13

Align Title:gi|1632182079|ref|XR_003772270.1| PREDICTED: Malus domestica uncharacterized LOC108172495
Score: 38.0 E-val: 4.59402
N.aligns:None Ident:27 Pos.:27 Align len:31
Num mismatches: 4
Mismatch pos: 5,8,23,27

Align Title:gi|1632195925|ref|XM_008375918.3| PREDICTED: Malus domestica granule-bound starch synthas
Score: 3690.0 E-val: 0.0
N.aligns:None Ident:1845 Pos.:1845 Align len:1845
No mismatches

Align Title:gi|1632195924|ref|XM_008375917.3| PREDICTED: Malus domestica granule-bound starch synthas
Score: 3690.0 E-val: 0.0
N.aligns:None Ident:1845 Pos.:1845 Align len:1845
No mismatches

Align Title:gi|658309783|ref|NM_001293995.1| Malus domestica granule-bound starch synthase 1, chloro
Score: 3690.0 E-val: 0.0
N.aligns:None Ident:1845 Pos.:1845 Align len:1845
No mismatches

Align Title:gi|1632245385|ref|XM_008394428.2| PREDICTED: Malus domestica granule-bound starch synthas
Score: 2961.0 E-val: 0.0
N.aligns:None Ident:1698 Pos.:1698 Align len:1843
Num mismatches: 145
Mismatch pos: 9,15,20,23,29,37,48,59,68,70,122,140,163,169,171,176,187,213,215,229,239,253,257,281,31

Align Title:gi|1632245384|ref|XM_008394427.3| PREDICTED: Malus domestica granule-bound starch synthas
Score: 2961.0 E-val: 0.0
N.aligns:None Ident:1698 Pos.:1698 Align len:1843
Num mismatches: 145
Mismatch pos: 9,15,20,23,29,37,48,59,68,70,122,140,163,169,171,176,187,213,215,229,239,253,257,281,31

Align Title:gi|658309847|ref|NM_001293907.1| Malus domestica granule-bound starch synthase 1, chloro
Score: 1253.0 E-val: 0.0
N.aligns:None Ident:1214 Pos.:1214 Align len:1605
Num mismatches: 391
Mismatch pos: 5,11,17,23,26,33,38,39,44,59,71,92,95,104,110,113,116,122,125,128,134,135,137,143,146,1
```

```
Score: 40.0 E-val: 1.31621
N.aligns:None Ident:26 Pos.:26 Align len:30
Num mismatches: 4
Mismatch pos: 2,17,19,23

Align Title:gi|1632200625|ref|XM_029104747.1| PREDICTED: Malus domestica granule-bound starch synthas
Score: 1249.0 E-val: 0.0
N.aligns:None Ident:1214 Pos.:1214 Align len:1605
Num mismatches: 391
Mismatch pos: 5,11,17,23,26,33,38,39,44,59,71,92,95,104,110,113,116,122,125,128,134,135,137,143,146,1

Score: 40.0 E-val: 1.31621
N.aligns:None Ident:26 Pos.:26 Align len:30
Num mismatches: 4
Mismatch pos: 2,17,19,23

Align Title:gi|1632200623|ref|XM_008378000.3| PREDICTED: Malus domestica granule-bound starch synthas
Score: 1249.0 E-val: 0.0
N.aligns:None Ident:1214 Pos.:1214 Align len:1605
Num mismatches: 391
Mismatch pos: 5,11,17,23,26,33,38,39,44,59,71,92,95,104,110,113,116,122,125,128,134,135,137,143,146,1

Score: 40.0 E-val: 1.31621
N.aligns:None Ident:26 Pos.:26 Align len:30
Num mismatches: 4
Mismatch pos: 2,17,19,23

Align Title:gi|1632200627|ref|XM_029104748.1| PREDICTED: Malus domestica granule-bound starch synthas
Score: 1235.0 E-val: 0.0
N.aligns:None Ident:1213 Pos.:1213 Align len:1608
Num mismatches: 395
Mismatch pos: 5,11,17,23,26,33,38,39,44,59,71,92,95,104,110,113,116,122,125,128,134,135,137,143,146,1

Score: 40.0 E-val: 1.31621
N.aligns:None Ident:26 Pos.:26 Align len:30
Num mismatches: 4
Mismatch pos: 2,17,19,23

Align Title:gi|357528259|gb|JN187080.1| Malus x domestica granule-bound starch synthase GBSS1 gene, c
Score: 705.0 E-val: 0.0
N.aligns:None Ident:354 Pos.:354 Align len:355
Num mismatches: 1
Mismatch pos: 352

Score: 494.0 E-val: 7.73152e-124
N.aligns:None Ident:247 Pos.:247 Align len:247
No mismatches

Score: 388.0 E-val: 4.57377e-95
N.aligns:None Ident:194 Pos.:194 Align len:194
No mismatches

Score: 363.0 E-val: 5.21867e-88
N.aligns:None Ident:183 Pos.:183 Align len:184
Num mismatches: 1
Mismatch pos: 181

Score: 260.0 E-val: 2.53415e-60
N.aligns:None Ident:130 Pos.:130 Align len:130
```

```
No mismatches

Score: 238.0 E-val: 2.37346e-54
N.aligns:None Ident:119 Pos.:119 Align len:119
No mismatches

Score: 229.0 E-val: 1.22948e-51
N.aligns:None Ident:116 Pos.:116 Align len:117
Num mismatches: 1
Mismatch pos: 113

Score: 212.0 E-val: 2.70812e-47
N.aligns:None Ident:106 Pos.:106 Align len:106
No mismatches

Score: 208.0 E-val: 3.29916e-46
N.aligns:None Ident:104 Pos.:104 Align len:104
No mismatches

Score: 193.0 E-val: 7.26689e-42
N.aligns:None Ident:98 Pos.:98 Align len:99
Num mismatches: 1
Mismatch pos: 2

Score: 176.0 E-val: 1.60064e-37
N.aligns:None Ident:88 Pos.:88 Align len:88
No mismatches

Score: 165.0 E-val: 2.89402e-34
N.aligns:None Ident:84 Pos.:84 Align len:85
Num mismatches: 1
Mismatch pos: 82

Score: 140.0 E-val: 9.46061e-28
N.aligns:None Ident:70 Pos.:70 Align len:70
No mismatches

Align Title:gi|357528265|gb|JN187083.1| Malus x domestica granule-bound starch synthase GBSS4 pseudog
Score: 603.0 E-val: 3.74445e-153
N.aligns:None Ident:582 Pos.:582 Align len:765
Num mismatches: 183
Mismatch pos: 5,11,17,23,26,33,38,39,44,59,71,92,95,110,113,116,122,125,128,134,135,137,143,146,149,1

Align Title:gi|357528263|gb|JN187082.1| Malus x domestica granule-bound starch synthase GBSS3 gene, c
Score: 570.0 E-val: 1.81668e-144
N.aligns:None Ident:327 Pos.:327 Align len:355
Num mismatches: 28
Mismatch pos: 9,15,20,23,29,37,48,59,68,70,122,140,163,169,171,176,187,213,215,229,239,253,257,281,31

Score: 384.0 E-val: 5.572e-94
N.aligns:None Ident:225 Pos.:225 Align len:247
Num mismatches: 22
Mismatch pos: 2,5,10,13,55,58,61,67,79,87,94,97,133,151,170,185,187,208,214,215,225,238

Score: 283.0 E-val: 2.70573e-66
N.aligns:None Ident:173 Pos.:173 Align len:194
Num mismatches: 21
Mismatch pos: 2,8,11,18,33,38,44,45,56,59,90,91,93,101,113,125,132,140,158,173,176
```

```
Score: 279.0 E-val: 3.29625e-65
N.aligns:None Ident:162 Pos.:162 Align len:177
Num mismatches: 15
Mismatch pos: 11,14,44,46,67,71,103,106,133,142,145,155,160,167,174

Score: 220.0 E-val: 1.82472e-49
N.aligns:None Ident:122 Pos.:122 Align len:130
Num mismatches: 8
Mismatch pos: 6,27,38,50,57,68,80,101

Score: 188.0 E-val: 8.85288e-41
N.aligns:None Ident:97 Pos.:97 Align len:99
Num mismatches: 2
Mismatch pos: 86,87

Score: 184.0 E-val: 1.0785e-39
N.aligns:None Ident:107 Pos.:107 Align len:117
Num mismatches: 10
Mismatch pos: 10,38,48,56,63,73,85,88,94,104

Score: 182.0 E-val: 3.76434e-39
N.aligns:None Ident:106 Pos.:106 Align len:116
Num mismatches: 10
Mismatch pos: 4,46,47,48,55,62,68,79,109,113

Score: 178.0 E-val: 4.58591e-38
N.aligns:None Ident:98 Pos.:98 Align len:104
Num mismatches: 6
Mismatch pos: 8,18,36,63,65,90

Score: 156.0 E-val: 4.29511e-32
N.aligns:None Ident:84 Pos.:84 Align len:88
Num mismatches: 4
Mismatch pos: 7,30,66,74

Score: 152.0 E-val: 5.23252e-31
N.aligns:None Ident:88 Pos.:88 Align len:96
Num mismatches: 8
Mismatch pos: 12,18,42,43,54,59,69,85

Score: 140.0 E-val: 9.46061e-28
N.aligns:None Ident:79 Pos.:79 Align len:85
Num mismatches: 6
Mismatch pos: 6,24,42,67,74,82

Score: 115.0 E-val: 1.07946e-20
N.aligns:None Ident:65 Pos.:65 Align len:70
Num mismatches: 5
Mismatch pos: 26,32,36,56,62

Align Title:gi|357528261|gb|JN187081.1| Malus x domestica granule-bound starch synthase GBSS2 gene, c
Score: 194.0 E-val: 2.082e-42
N.aligns:None Ident:187 Pos.:187 Align len:247
Num mismatches: 60
Mismatch pos: 2,5,13,16,19,22,25,34,43,46,52,53,55,61,66,70,71,73,79,95,97,110,111,114,115,118,124,12

Score: 150.0 E-val: 1.82633e-30
N.aligns:None Ident:96 Pos.:96 Align len:110
Num mismatches: 14
```

```
Mismatch pos: 5,11,17,23,26,33,38,39,44,59,71,92,95,104

Score: 148.0 E-val: 6.37451e-30
N.aligns:None Ident:146 Pos.:146 Align len:194
Num mismatches: 48
Mismatch pos: 2,5,11,15,16,17,18,25,30,32,33,41,44,46,47,48,53,56,58,59,62,65,66,67,72,74,77,91,92,93

Score: 132.0 E-val: 1.40408e-25
N.aligns:None Ident:105 Pos.:105 Align len:131
Num mismatches: 26
Mismatch pos: 12,14,20,23,26,29,41,44,47,50,53,56,59,60,68,71,72,74,77,86,92,95,113,119,120,128

Score: 108.0 E-val: 4.58996e-19
N.aligns:None Ident:72 Pos.:72 Align len:84
Num mismatches: 12
Mismatch pos: 5,11,14,17,32,47,50,63,68,71,74,80

Score: 107.0 E-val: 1.60205e-18
N.aligns:None Ident:94 Pos.:94 Align len:121
Num mismatches: 27
Mismatch pos: 10,11,18,19,20,32,35,40,41,47,48,50,52,55,62,68,71,74,76,83,86,87,88,89,92,101,110

Score: 91.0 E-val: 3.52876e-14
N.aligns:None Ident:71 Pos.:71 Align len:88
Num mismatches: 17
Mismatch pos: 6,7,15,21,24,27,28,30,33,42,51,54,66,74,75,81,84

Score: 73.0 E-val: 2.71291e-09
N.aligns:None Ident:71 Pos.:71 Align len:94
Num mismatches: 23
Mismatch pos: 11,17,19,29,35,39,40,41,42,44,49,50,53,55,58,65,68,69,72,74,81,83,86

Score: 62.0 E-val: 1.40532e-06
N.aligns:None Ident:52 Pos.:52 Align len:66
Num mismatches: 14
Mismatch pos: 3,6,12,15,18,24,25,27,33,36,39,42,51,63

Score: 61.0 E-val: 4.90505e-06
N.aligns:None Ident:53 Pos.:53 Align len:68
Num mismatches: 15
Mismatch pos: 4,8,10,12,13,22,25,28,31,32,34,35,42,49,52

Score: 53.0 E-val: 0.000727974
N.aligns:None Ident:82 Pos.:82 Align len:119
Num mismatches: 37
Mismatch pos: 4,8,10,13,14,16,22,23,25,26,32,37,38,42,43,46,49,50,51,54,56,60,61,63,64,65,70,73,76,79

Score: 40.0 E-val: 1.31621
N.aligns:None Ident:26 Pos.:26 Align len:30
Num mismatches: 4
Mismatch pos: 2,17,19,23

Align Title:gi|1632215725|ref|XM_029108378.1| PREDICTED: Malus domestica NEDD8-specific protease 1-li
Score: 79.0 E-val: 6.38014e-11
N.aligns:None Ident:53 Pos.:53 Align len:62
Num mismatches: 9
Mismatch pos: 2,11,26,28,29,36,45,50,58

Align Title:gi|1632206082|ref|XM_008358830.3| PREDICTED: Malus domestica starch synthase 1, chloropla
```

```
Score: 56.0 E-val: 5.97557e-05
N.aligns:None Ident:49 Pos.:49 Align len:63
Num mismatches: 14
Mismatch pos: 5,8,14,17,20,22,26,27,30,32,50,53,56,59

Align Title:gi|1632206081|ref|XM_008358828.3| PREDICTED: Malus domestica starch synthase 1, chloropla
Score: 56.0 E-val: 5.97557e-05
N.aligns:None Ident:49 Pos.:49 Align len:63
Num mismatches: 14
Mismatch pos: 5,8,14,17,20,22,26,27,30,32,50,53,56,59

Align Title:gi|1632223611|ref|XM_029087607.1| PREDICTED: Malus domestica starch synthase 3, chloropla
Score: 53.0 E-val: 0.000727974
N.aligns:None Ident:76 Pos.:76 Align len:109
Num mismatches: 33
Mismatch pos: 4,15,18,19,21,24,25,26,27,29,30,33,36,39,50,51,54,55,57,58,59,60,75,76,78,79,81,91,92,9

Align Title:gi|1632169330|ref|XM_008382548.3| PREDICTED: Malus domestica starch synthase 1, chloropla
Score: 51.0 E-val: 0.00254088
N.aligns:None Ident:48 Pos.:48 Align len:63
Num mismatches: 15
Mismatch pos: 2,8,11,14,17,20,22,26,30,32,44,50,53,56,59

Align Title:gi|1284883255|gb|KY524597.1| Malus domestica AT1G02020 mRNA, partial sequence
Score: 51.0 E-val: 0.00254088
N.aligns:None Ident:48 Pos.:48 Align len:63
Num mismatches: 15
Mismatch pos: 2,8,11,14,17,20,22,26,30,32,44,50,53,56,59

Align Title:gi|1632194644|ref|XM_029103312.1| PREDICTED: Malus domestica starch synthase 3, chloropla
Score: 48.0 E-val: 0.00886854
N.aligns:None Ident:75 Pos.:75 Align len:109
Num mismatches: 34
Mismatch pos: 4,15,18,19,21,24,25,26,27,29,30,33,36,39,50,51,54,55,57,58,59,60,75,76,78,79,81,91,92,9

Align Title:gi|1632261657|ref|XM_029096170.1| PREDICTED: Malus domestica chromatin structure-remodeli
Score: 41.0 E-val: 1.31621
N.aligns:None Ident:25 Pos.:25 Align len:28
Num mismatches: 3
Mismatch pos: 4,5,7

Align Title:gi|1632261655|ref|XM_029096169.1| PREDICTED: Malus domestica chromatin structure-remodeli
Score: 41.0 E-val: 1.31621
N.aligns:None Ident:25 Pos.:25 Align len:28
Num mismatches: 3
Mismatch pos: 4,5,7

Align Title:gi|1632241173|ref|XM_029092074.1| PREDICTED: Malus domestica cytochrome b561 and DOMON do
Score: 41.0 E-val: 1.31621
N.aligns:None Ident:25 Pos.:25 Align len:28
Num mismatches: 3
Mismatch pos: 18,19,21

Align Title:gi|1632241171|ref|XM_029092073.1| PREDICTED: Malus domestica cytochrome b561 and DOMON do
Score: 41.0 E-val: 1.31621
N.aligns:None Ident:25 Pos.:25 Align len:28
Num mismatches: 3
Mismatch pos: 18,19,21
```

```
Align Title:gi|1632241169|ref|XM_008343395.2| PREDICTED: Malus domestica cytochrome b561 and DOMON do
Score: 41.0 E-val: 1.31621
N.aligns:None Ident:25 Pos.:25 Align len:28
Num mismatches: 3
Mismatch pos: 18,19,21

Align Title:gi|1632238354|ref|XM_008392532.3| PREDICTED: Malus domestica chromatin structure-remodeli
Score: 41.0 E-val: 1.31621
N.aligns:None Ident:25 Pos.:25 Align len:28
Num mismatches: 3
Mismatch pos: 4,5,7

Align Title:gi|1632238350|ref|XM_008392530.3| PREDICTED: Malus domestica chromatin structure-remodeli
Score: 41.0 E-val: 1.31621
N.aligns:None Ident:25 Pos.:25 Align len:28
Num mismatches: 3
Mismatch pos: 4,5,7

Align Title:gi|1632238348|ref|XM_008392529.3| PREDICTED: Malus domestica chromatin structure-remodeli
Score: 41.0 E-val: 1.31621
N.aligns:None Ident:25 Pos.:25 Align len:28
Num mismatches: 3
Mismatch pos: 4,5,7

Align Title:gi|1632193684|ref|XM_008373360.3| PREDICTED: Malus domestica LRR receptor-like serine/thr
Score: 41.0 E-val: 1.31621
N.aligns:None Ident:30 Pos.:30 Align len:36
Num mismatches: 6
Mismatch pos: 7,11,16,17,18,19

Align Title:gi|1632166542|ref|XM_029091540.1| PREDICTED: Malus domestica plant UBX domain-containing
Score: 41.0 E-val: 1.31621
N.aligns:None Ident:32 Pos.:32 Align len:37
Num mismatches: 5
Mismatch pos: 7,11,15,20,32

Align Title:gi|1632166540|ref|XM_029091520.1| PREDICTED: Malus domestica plant UBX domain-containing
Score: 41.0 E-val: 1.31621
N.aligns:None Ident:32 Pos.:32 Align len:37
Num mismatches: 5
Mismatch pos: 7,11,15,20,32

Align Title:gi|1632166538|ref|XM_029091488.1| PREDICTED: Malus domestica plant UBX domain-containing
Score: 41.0 E-val: 1.31621
N.aligns:None Ident:32 Pos.:32 Align len:37
Num mismatches: 5
Mismatch pos: 7,11,15,20,32

Align Title:gi|1632166536|ref|XM_008345648.3| PREDICTED: Malus domestica plant UBX domain-containing
Score: 41.0 E-val: 1.31621
N.aligns:None Ident:32 Pos.:32 Align len:37
Num mismatches: 5
Mismatch pos: 7,11,15,20,32

Align Title:gi|1632166534|ref|XM_029091435.1| PREDICTED: Malus domestica plant UBX domain-containing
Score: 41.0 E-val: 1.31621
N.aligns:None Ident:32 Pos.:32 Align len:37
Num mismatches: 5
Mismatch pos: 7,11,15,20,32
```

```
Align Title:gi|1632192461|ref|XR_003773535.1| PREDICTED: Malus domestica uncharacterized LOC11482493
Score: 40.0 E-val: 1.31621
N.aligns:None Ident:23 Pos.:23 Align len:25
Num mismatches: 2
Mismatch pos: 18,19

Align Title:gi|1632170586|ref|XM_029108527.1| PREDICTED: Malus domestica glycerophosphodiester phosph
Score: 39.0 E-val: 4.59402
N.aligns:None Ident:21 Pos.:21 Align len:22
Num mismatches: 1
Mismatch pos: 11

Align Title:gi|1632170584|ref|XM_008345687.2| PREDICTED: Malus domestica glycerophosphodiester phosph
Score: 39.0 E-val: 4.59402
N.aligns:None Ident:21 Pos.:21 Align len:22
Num mismatches: 1
Mismatch pos: 11

Align Title:gi|1632170582|ref|XM_008345686.2| PREDICTED: Malus domestica glycerophosphodiester phosph
Score: 39.0 E-val: 4.59402
N.aligns:None Ident:21 Pos.:21 Align len:22
Num mismatches: 1
Mismatch pos: 11

Align Title:gi|1632214967|ref|XM_008382221.3| PREDICTED: Malus domestica uncharacterized LOC10344338
Score: 38.0 E-val: 4.59402
N.aligns:None Ident:27 Pos.:27 Align len:31
Num mismatches: 4
Mismatch pos: 11,15,21,24

Align Title:gi|1632213490|ref|XM_008381298.3| PREDICTED: Malus domestica uncharacterized LOC10344250
Score: 38.0 E-val: 4.59402
N.aligns:None Ident:22 Pos.:22 Align len:24
Num mismatches: 2
Mismatch pos: 11,21

Align Title:gi|1632213488|ref|XM_008381297.3| PREDICTED: Malus domestica uncharacterized LOC10344250
Score: 38.0 E-val: 4.59402
N.aligns:None Ident:22 Pos.:22 Align len:24
Num mismatches: 2
Mismatch pos: 11,21

Align Title:gi|1632192404|ref|XR_003773527.1| PREDICTED: Malus domestica uncharacterized LOC11482493
Score: 38.0 E-val: 4.59402
N.aligns:None Ident:22 Pos.:22 Align len:24
Num mismatches: 2
Mismatch pos: 18,19

Align Title:gi|1632168202|ref|XM_008375822.3| PREDICTED: Malus domestica dual specificity tyrosine-ph
Score: 38.0 E-val: 4.59402
N.aligns:None Ident:22 Pos.:22 Align len:24
Num mismatches: 2
Mismatch pos: 11,17

Align Title:gi|1632168200|ref|XM_017333399.2| PREDICTED: Malus domestica dual specificity tyrosine-ph
Score: 38.0 E-val: 4.59402
N.aligns:None Ident:22 Pos.:22 Align len:24
Num mismatches: 2
```

```
Mismatch pos: 11,17

Align Title:gi|658309773|ref|NM_001293931.1| Malus domestica probable alpha-amylase 2 (LOC103440221),
Score: 3172.0 E-val: 0.0
N.aligns:None Ident:1586 Pos.:1586 Align len:1586
No mismatches

Align Title:gi|1632206575|ref|XM_008378889.3| PREDICTED: Malus domestica probable alpha-amylase 2 (LO
Score: 3063.0 E-val: 0.0
N.aligns:None Ident:1559 Pos.:1559 Align len:1576
Num mismatches: 17
Mismatch pos: 318,384,440,462,463,572,655,738,759,946,952,1144,1171,1213,1335,1380,1573

Align Title:gi|1632206574|ref|XM_008378890.3| PREDICTED: Malus domestica probable alpha-amylase 2 (LO
Score: 3063.0 E-val: 0.0
N.aligns:None Ident:1559 Pos.:1559 Align len:1576
Num mismatches: 17
Mismatch pos: 318,384,440,462,463,572,655,738,759,946,952,1144,1171,1213,1335,1380,1573

Align Title:gi|1632206573|ref|XM_008378891.3| PREDICTED: Malus domestica probable alpha-amylase 2 (LO
Score: 3063.0 E-val: 0.0
N.aligns:None Ident:1559 Pos.:1559 Align len:1576
Num mismatches: 17
Mismatch pos: 318,384,440,462,463,572,655,738,759,946,952,1144,1171,1213,1335,1380,1573

Align Title:gi|1632252077|ref|XM_029094733.1| PREDICTED: Malus domestica probable alpha-amylase 2 (LO
Score: 1948.0 E-val: 0.0
N.aligns:None Ident:1203 Pos.:1203 Align len:1339
Num mismatches: 136
Mismatch pos: 10,39,50,51,53,54,55,58,87,103,130,165,166,167,168,170,184,188,197,201,229,230,231,233,

Score: 41.0 E-val: 1.31621
N.aligns:None Ident:30 Pos.:30 Align len:35
Num mismatches: 5
Mismatch pos: 15,18,23,26,28

Align Title:gi|60652322|gb|AY939872.1| Malus x domestica cytosolic alpha-amylase (AMY9) gene, complet
Score: 391.0 E-val: 1.31041e-95
N.aligns:None Ident:220 Pos.:220 Align len:235
Num mismatches: 15
Mismatch pos: 27,48,58,62,75,114,128,141,142,158,183,184,206,207,228

Score: 334.0 E-val: 2.07832e-80
N.aligns:None Ident:202 Pos.:202 Align len:225
Num mismatches: 23
Mismatch pos: 45,54,63,69,81,84,97,99,123,136,160,162,165,168,192,201,202,203,204,205,211,215,222

Score: 270.0 E-val: 4.89207e-63
N.aligns:None Ident:144 Pos.:144 Align len:150
Num mismatches: 6
Mismatch pos: 8,12,18,59,88,147

Score: 241.0 E-val: 6.80008e-55
N.aligns:None Ident:128 Pos.:128 Align len:133
Num mismatches: 5
Mismatch pos: 53,101,104,113,125

Score: 199.0 E-val: 1.70901e-43
N.aligns:None Ident:110 Pos.:110 Align len:117
```

```
Num mismatches: 7
Mismatch pos: 7,25,73,85,90,108,114

Score: 199.0 E-val: 1.70901e-43
N.aligns:None Ident:116 Pos.:116 Align len:127
Num mismatches: 11
Mismatch pos: 8,60,63,77,80,81,84,92,98,116,122

Score: 177.0 E-val: 1.60064e-37
N.aligns:None Ident:96 Pos.:96 Align len:101
Num mismatches: 5
Mismatch pos: 8,21,22,61,73

Score: 169.0 E-val: 2.37556e-35
N.aligns:None Ident:89 Pos.:89 Align len:92
Num mismatches: 3
Mismatch pos: 17,42,69

Score: 90.0 E-val: 3.52876e-14
N.aligns:None Ident:48 Pos.:48 Align len:50
Num mismatches: 2
Mismatch pos: 10,39

Score: 77.0 E-val: 2.22689e-10
N.aligns:None Ident:40 Pos.:40 Align len:41
Num mismatches: 1
Mismatch pos: 22

Score: 72.0 E-val: 2.71291e-09
N.aligns:None Ident:38 Pos.:38 Align len:40
Num mismatches: 2
Mismatch pos: 8,35

Score: 67.0 E-val: 1.15356e-07
N.aligns:None Ident:35 Pos.:35 Align len:36
Num mismatches: 1
Mismatch pos: 25

Align Title:gi|1632268526|ref|XM_008359555.3| PREDICTED: Malus domestica uncharacterized LOC103421513
Score: 172.0 E-val: 1.94998e-36
N.aligns:None Ident:377 Pos.:377 Align len:563
Num mismatches: 186
Mismatch pos: 6,8,9,17,18,20,21,24,37,38,39,40,42,43,44,45,48,51,52,57,60,63,65,70,71,77,78,79,81,87,

Align Title:gi|1632213193|ref|XM_008359554.3| PREDICTED: Malus domestica uncharacterized LOC103421512
Score: 164.0 E-val: 2.89402e-34
N.aligns:None Ident:375 Pos.:375 Align len:561
Num mismatches: 186
Mismatch pos: 6,8,9,17,20,21,24,37,38,39,40,42,43,44,45,51,52,57,63,65,70,74,76,77,79,85,88,92,95,104

Align Title:gi|1632235994|ref|XM_029090866.1| PREDICTED: Malus domestica alpha-amylase 3, chloroplast
Score: 122.0 E-val: 7.27331e-23
N.aligns:None Ident:129 Pos.:129 Align len:173
Num mismatches: 44
Mismatch pos: 5,11,17,20,21,35,41,42,50,54,59,62,71,77,80,85,93,94,95,98,101,103,104,107,108,110,111,

Align Title:gi|658309971|ref|NM_001294091.1| Malus domestica alpha-amylase 3, chloroplastic-like (AMY
Score: 122.0 E-val: 7.27331e-23
N.aligns:None Ident:129 Pos.:129 Align len:173
```

```
Num mismatches: 44
Mismatch pos: 5,11,17,20,21,35,41,42,50,54,59,62,71,77,80,85,93,94,95,98,101,103,104,107,108,110,111,

Align Title:gi|1632262880|ref|XM_008341710.3| PREDICTED: Malus domestica alpha-amylase 3, chloroplast
Score: 115.0 E-val: 1.07946e-20
N.aligns:None Ident:130 Pos.:130 Align len:175
Num mismatches: 45
Mismatch pos: 5,11,17,20,21,35,41,42,50,54,59,62,71,77,80,85,93,94,95,98,101,102,107,108,110,112,113,

Align Title:gi|1632174389|ref|XM_029092045.1| PREDICTED: Malus domestica uncharacterized LOC114820820
Score: 101.0 E-val: 6.8121e-17
N.aligns:None Ident:61 Pos.:61 Align len:68
Num mismatches: 7
Mismatch pos: 21,26,27,34,47,51,65

Align Title:gi|1632189813|ref|XM_008374926.3| PREDICTED: Malus domestica alpha-amylase (LOC103436492)
Score: 77.0 E-val: 2.22689e-10
N.aligns:None Ident:104 Pos.:104 Align len:145
Num mismatches: 41
Mismatch pos: 3,6,10,18,33,36,39,41,42,46,52,55,58,59,62,65,67,68,70,74,79,82,83,91,94,97,100,103,109

Align Title:gi|1632189811|ref|XM_008374925.3| PREDICTED: Malus domestica alpha-amylase-like (LOC10343
Score: 73.0 E-val: 2.71291e-09
N.aligns:None Ident:101 Pos.:101 Align len:144
Num mismatches: 43
Mismatch pos: 3,6,10,12,15,18,33,36,39,40,41,43,44,45,51,54,57,58,60,61,64,66,67,69,73,78,81,82,84,90

Align Title:gi|60652318|gb|AY939870.1| Malus x domestica secreted alpha-amylase (AMY2) mRNA, partial
Score: 71.0 E-val: 9.46897e-09
N.aligns:None Ident:83 Pos.:83 Align len:112
Num mismatches: 29
Mismatch pos: 3,6,10,18,33,36,39,41,42,46,52,55,58,59,62,65,67,68,70,74,79,82,83,91,94,97,100,103,109

Align Title:gi|1632211733|ref|XR_003775807.1| PREDICTED: Malus domestica uncharacterized LOC103424175
Score: 68.0 E-val: 3.305e-08
N.aligns:None Ident:37 Pos.:37 Align len:39
Num mismatches: 2
Mismatch pos: 7,34

Align Title:gi|1632211732|ref|XR_003775806.1| PREDICTED: Malus domestica uncharacterized LOC103424175
Score: 68.0 E-val: 3.305e-08
N.aligns:None Ident:37 Pos.:37 Align len:39
Num mismatches: 2
Mismatch pos: 7,34

Align Title:gi|1632223006|ref|XM_008360451.3| PREDICTED: Malus domestica protein DEHYDRATION-INDUCED
Score: 43.0 E-val: 0.3771
N.aligns:None Ident:34 Pos.:34 Align len:40
Num mismatches: 6
Mismatch pos: 14,15,16,20,28,34

Align Title:gi|1632223005|ref|XM_008360449.3| PREDICTED: Malus domestica protein DEHYDRATION-INDUCED
Score: 43.0 E-val: 0.3771
N.aligns:None Ident:34 Pos.:34 Align len:40
Num mismatches: 6
Mismatch pos: 14,15,16,20,28,34

Align Title:gi|1632171122|ref|XR_528151.3| PREDICTED: Malus domestica uncharacterized LOC103426622 (
Score: 43.0 E-val: 0.3771
```

```
N.aligns:None Ident:26 Pos.:26 Align len:29
Num mismatches: 3
Mismatch pos: 8,21,23

Align Title:gi|1632215018|ref|XM_008382491.3| PREDICTED: Malus domestica uncharacterized protein C630
Score: 40.0 E-val: 1.31621
N.aligns:None Ident:25 Pos.:25 Align len:27
Num mismatches: 2
Mismatch pos: 15,19

Align Title:gi|1632215017|ref|XM_008382490.3| PREDICTED: Malus domestica uncharacterized protein C630
Score: 40.0 E-val: 1.31621
N.aligns:None Ident:25 Pos.:25 Align len:27
Num mismatches: 2
Mismatch pos: 15,19

Align Title:gi|1632200206|ref|XM_008350248.3| PREDICTED: Malus domestica probable LRR receptor-like s
Score: 39.0 E-val: 4.59402
N.aligns:None Ident:27 Pos.:27 Align len:32
Num mismatches: 5
Mismatch pos: 4,9,12,16,18

Align Title:gi|1632245259|ref|XM_029093131.1| PREDICTED: Malus domestica dynamin-related protein 4C-l
Score: 38.0 E-val: 4.59402
N.aligns:None Ident:36 Pos.:36 Align len:45
Num mismatches: 9
Mismatch pos: 5,9,10,13,17,18,30,35,40

Align Title:gi|1632226752|ref|XM_029088520.1| PREDICTED: Malus domestica serine/threonine-protein kin
Score: 38.0 E-val: 4.59402
N.aligns:None Ident:28 Pos.:28 Align len:34
Num mismatches: 6
Mismatch pos: 4,6,9,24,26,27

Align Title:gi|1632226750|ref|XM_029088519.1| PREDICTED: Malus domestica serine/threonine-protein kin
Score: 38.0 E-val: 4.59402
N.aligns:None Ident:28 Pos.:28 Align len:34
Num mismatches: 6
Mismatch pos: 4,6,9,24,26,27

Align Title:gi|1632255559|ref|XM_029095487.1| PREDICTED: Malus domestica probable starch synthase 4,
Score: 4494.0 E-val: 0.0
N.aligns:None Ident:2247 Pos.:2247 Align len:2247
No mismatches

Align Title:gi|1632255557|ref|XM_029095486.1| PREDICTED: Malus domestica probable starch synthase 4,
Score: 4467.0 E-val: 0.0
N.aligns:None Ident:2247 Pos.:2247 Align len:2258
Num mismatches: 11
Mismatch pos: 240,241,242,243,244,245,246,247,248,249,250

Align Title:gi|1632166473|ref|XM_029090068.1| PREDICTED: Malus domestica probable starch synthase 4,
Score: 4116.0 E-val: 0.0
N.aligns:None Ident:2176 Pos.:2176 Align len:2252
Num mismatches: 76
Mismatch pos: 10,18,19,24,25,29,88,141,142,143,154,167,170,197,230,231,232,233,234,235,236,237,238,23

Align Title:gi|1632178903|ref|XM_008355920.3| PREDICTED: Malus domestica probable protein phosphatase
Score: 42.0 E-val: 0.3771
```

```
N.aligns:None Ident:35 Pos.:35 Align len:43
Num mismatches: 8
Mismatch pos: 13,14,15,16,17,19,32,34

Align Title:gi|1632162884|ref|XM_008377756.3| PREDICTED: Malus domestica mitogen-activated protein ki
Score: 42.0 E-val: 0.3771
N.aligns:None Ident:27 Pos.:27 Align len:31
Num mismatches: 4
Mismatch pos: 4,6,19,28

Align Title:gi|1632223925|ref|XM_029087690.1| PREDICTED: Malus domestica two-pore potassium channel 1
Score: 41.0 E-val: 1.31621
N.aligns:None Ident:25 Pos.:25 Align len:28
Num mismatches: 3
Mismatch pos: 2,16,20

Align Title:gi|1632205457|ref|XM_008378146.3| PREDICTED: Malus domestica oleosin 1-like (LOC10343958
Score: 40.0 E-val: 1.31621
N.aligns:None Ident:29 Pos.:29 Align len:35
Num mismatches: 6
Mismatch pos: 2,5,20,22,29,31

Align Title:gi|1632186274|ref|XM_008365690.3| PREDICTED: Malus domestica uncharacterized LOC10342762
Score: 39.0 E-val: 4.59402
N.aligns:None Ident:30 Pos.:30 Align len:37
Num mismatches: 7
Mismatch pos: 12,17,18,20,23,29,34

Align Title:gi|1632186272|ref|XM_008365689.3| PREDICTED: Malus domestica uncharacterized LOC10342762
Score: 39.0 E-val: 4.59402
N.aligns:None Ident:30 Pos.:30 Align len:37
Num mismatches: 7
Mismatch pos: 12,17,18,20,23,29,34

Align Title:gi|1632183600|ref|XM_029100637.1| PREDICTED: Malus domestica uncharacterized LOC11482420
Score: 39.0 E-val: 4.59402
N.aligns:None Ident:41 Pos.:41 Align len:53
Num mismatches: 12
Mismatch pos: 2,8,10,11,13,17,31,32,35,36,42,45

Align Title:gi|1632167864|ref|XM_008389523.3| PREDICTED: Malus domestica heat shock cognate 70 kDa pr
Score: 39.0 E-val: 4.59402
N.aligns:None Ident:27 Pos.:27 Align len:31
Num mismatches: 4
Mismatch pos: 5,7,19,20

Align Title:gi|1632167856|ref|XM_008364489.3| PREDICTED: Malus domestica heat shock cognate 70 kDa pr
Score: 39.0 E-val: 4.59402
N.aligns:None Ident:27 Pos.:27 Align len:31
Num mismatches: 4
Mismatch pos: 5,7,19,20

Align Title:gi|1632268257|ref|XR_003770907.1| PREDICTED: Malus domestica uncharacterized LOC10817011
Score: 38.0 E-val: 4.59402
N.aligns:None Ident:22 Pos.:22 Align len:24
Num mismatches: 2
Mismatch pos: 3,7

Align Title:gi|1632255849|ref|XM_008342676.3| PREDICTED: Malus domestica uncharacterized LOC10340382
```

```
Score: 38.0 E-val: 4.59402
N.aligns:None Ident:24 Pos.:24 Align len:26
Num mismatches: 2
Mismatch pos: 3,18

Align Title:gi|1632229560|ref|XM_008388384.3| PREDICTED: Malus domestica rho GTPase-activating protei
Score: 38.0 E-val: 4.59402
N.aligns:None Ident:22 Pos.:22 Align len:24
Num mismatches: 2
Mismatch pos: 2,21

Align Title:gi|1632219772|ref|XM_008379093.3| PREDICTED: Malus domestica probable receptor-like serin
Score: 38.0 E-val: 4.59402
N.aligns:None Ident:25 Pos.:25 Align len:29
Num mismatches: 4
Mismatch pos: 12,16,22,23

Align Title:gi|1632200352|ref|XM_029104682.1| PREDICTED: Malus domestica ubiquitin-conjugating enzyme
Score: 38.0 E-val: 4.59402
N.aligns:None Ident:19 Pos.:19 Align len:19
No mismatches

Align Title:gi|1632187865|ref|XR_003772816.1| PREDICTED: Malus domestica uncharacterized LOC10340867
Score: 38.0 E-val: 4.59402
N.aligns:None Ident:22 Pos.:22 Align len:24
Num mismatches: 2
Mismatch pos: 11,17

Align Title:gi|1632187864|ref|XR_003772815.1| PREDICTED: Malus domestica uncharacterized LOC10340867
Score: 38.0 E-val: 4.59402
N.aligns:None Ident:22 Pos.:22 Align len:24
Num mismatches: 2
Mismatch pos: 11,17

Align Title:gi|1632187863|ref|XR_003772814.1| PREDICTED: Malus domestica uncharacterized LOC10340867
Score: 38.0 E-val: 4.59402
N.aligns:None Ident:22 Pos.:22 Align len:24
Num mismatches: 2
Mismatch pos: 11,17

Align Title:gi|1632187861|ref|XR_003772812.1| PREDICTED: Malus domestica uncharacterized LOC10340867
Score: 38.0 E-val: 4.59402
N.aligns:None Ident:22 Pos.:22 Align len:24
Num mismatches: 2
Mismatch pos: 11,17

Align Title:gi|1632187860|ref|XR_003772811.1| PREDICTED: Malus domestica uncharacterized LOC10340867
Score: 38.0 E-val: 4.59402
N.aligns:None Ident:22 Pos.:22 Align len:24
Num mismatches: 2
Mismatch pos: 11,17

Align Title:gi|1632187859|ref|XR_003772810.1| PREDICTED: Malus domestica uncharacterized LOC10340867
Score: 38.0 E-val: 4.59402
N.aligns:None Ident:22 Pos.:22 Align len:24
Num mismatches: 2
Mismatch pos: 11,17

Align Title:gi|1632187857|ref|XR_003772808.1| PREDICTED: Malus domestica uncharacterized LOC10340867
```

```
Score: 38.0 E-val: 4.59402
N.aligns:None Ident:22 Pos.:22 Align len:24
Num mismatches: 2
Mismatch pos: 11,17

Align Title:gi|1632187855|ref|XR_003772806.1| PREDICTED: Malus domestica uncharacterized LOC10340867
Score: 38.0 E-val: 4.59402
N.aligns:None Ident:22 Pos.:22 Align len:24
Num mismatches: 2
Mismatch pos: 11,17

Align Title:gi|1632187854|ref|XR_003772805.1| PREDICTED: Malus domestica uncharacterized LOC10340867
Score: 38.0 E-val: 4.59402
N.aligns:None Ident:22 Pos.:22 Align len:24
Num mismatches: 2
Mismatch pos: 11,17

Align Title:gi|1632187852|ref|XR_003772803.1| PREDICTED: Malus domestica uncharacterized LOC10340867
Score: 38.0 E-val: 4.59402
N.aligns:None Ident:22 Pos.:22 Align len:24
Num mismatches: 2
Mismatch pos: 11,17

Align Title:gi|1632170394|ref|XM_008385860.3| PREDICTED: Malus domestica acyl-coenzyme A thioesterase
Score: 38.0 E-val: 4.59402
N.aligns:None Ident:22 Pos.:22 Align len:24
Num mismatches: 2
Mismatch pos: 3,19

Align Title:gi|1284951105|gb|KY538526.1| Malus domestica AT1G08610 mRNA, partial sequence
Score: 38.0 E-val: 4.59402
N.aligns:None Ident:25 Pos.:25 Align len:29
Num mismatches: 4
Mismatch pos: 12,16,22,23
```

5. Write a python function `getPublicationInfo(title_term,other_term)` that retrieves the first 20 pubmed publications having the `title_term` in the title and `other_term` somewhere else in the text (hint use: "Title" and "[Other Term]" as esearch parameter term). For each publication print:
   (a) the title
   (b) authors
   (c) journal
   (d) year of publication (hint: get and split properly the "PubDate" entry)
   (e) a link to the pubmed entry (hint: it is the string "https://www.ncbi.nlm.nih.gov/pubmed/" followed by the pubmed id ("eid" entry of the dictionary "ArticleIds"). es: https://www.ncbi.nlm.nih.gov/pubmed/26919684

Hint: to see how to combine search terms test them here: https://www.ncbi.nlm.nih.gov/pubmed/advanced.

Test your code calling `getPublicationInfo("apple","drought")`

Show/Hide Solution

In [58]:
```
from Bio import Entrez

def getPublicationInfo(title_term,other_term):
```

```
        Entrez.email = "my_email"
        s_term = title_term + " [Title] AND " + other_term + " [Other Term]"
        handle = Entrez.esearch(db="pubmed", term=s_term)
        res = Entrez.read(handle)
    #uncomment to see all info
    #    for el in res.keys():
    #        print(el , " : ", res[el])
    #
    #    print("")
        for ids in res["IdList"]:
            handle = Entrez.esummary(db="pubmed",  id = ids)
            res = Entrez.read(handle)
            #uncomment to see all info
            #print(res)
            for r in res:
                print(r["Title"])
                print(",".join(r["AuthorList"]))
                print(r["Source"])
                print(r["PubDate"].split()[0])
                print("https://www.ncbi.nlm.nih.gov/pubmed/" + r["ArticleIds"]["eid"])
                print("")

getPublicationInfo("apple","drought")
```

```
[Effects of PGPR inoculation on photosynthesis and physiological-ecological characteristics of apple
Xu XD,Zhang C,Qin C,Su YY,Zhou J,Zhang H,Zhang LX
Ying Yong Sheng Tai Xue Bao
2019
https://www.ncbi.nlm.nih.gov/pubmed/31621237

The ERF transcription factor MdERF38 promotes drought stress-induced anthocyanin biosynthesis in appl
An JP,Zhang XW,Bi SQ,You CX,Wang XF,Hao YJ
Plant J
2019
https://www.ncbi.nlm.nih.gov/pubmed/31571281

MdGRF11, an apple 14-3-3 protein, acts as a positive regulator of drought and salt tolerance.
Ren YR,Yang YY,Zhang R,You CX,Zhao Q,Hao YJ
Plant Sci
2019
https://www.ncbi.nlm.nih.gov/pubmed/31521216

Response of sugar metabolism in apple leaves subjected to short-term drought stress.
Yang J,Zhang J,Li C,Zhang Z,Ma F,Li M
Plant Physiol Biochem
2019
https://www.ncbi.nlm.nih.gov/pubmed/31170640

An apple (Malus domestica) NAC transcription factor enhances drought tolerance in transgenic apple pl
Jia D,Jiang Q,van Nocker S,Gong X,Ma F
Plant Physiol Biochem
2019
https://www.ncbi.nlm.nih.gov/pubmed/31015089

Impact of foliar fertilization on apple and pear trees in reconciling productivity and alleviation of
Zargar M,Tumanyan A,Ivanenko E,Dronik A,Tyutyuma N,Pakina E
Commun Integr Biol
2019
https://www.ncbi.nlm.nih.gov/pubmed/30891112
```

```
Apple AP2/EREBP transcription factor MdSHINE2 confers drought resistance by regulating wax biosynthes
Zhang YL,Zhang CL,Wang GL,Wang YX,Qi CH,You CX,Li YY,Hao YJ
Planta
2019
https://www.ncbi.nlm.nih.gov/pubmed/30826884

Overexpression of the RNA binding protein MhYTP1 in transgenic apple enhances drought tolerance and W
Guo T,Wang N,Xue Y,Guan Q,van Nocker S,Liu C,Ma F
Plant Sci
2019
https://www.ncbi.nlm.nih.gov/pubmed/30824018

Does fertilization explain the extraordinary hydraulic behaviour of apple trees?
Beikircher B,Losso A,Gemassmer M,Jansen S,Mayr S
J Exp Bot
2019
https://www.ncbi.nlm.nih.gov/pubmed/30793193

Neoformation and summer arrest are common sources of tree plasticity in response to water stress of a
Chen D,Pallas B,Martinez S,Wang Y,Costes E
Ann Bot
2019
https://www.ncbi.nlm.nih.gov/pubmed/30596816

An apple transcription factor, MdDREB76, confers salt and drought tolerance in transgenic tobacco by
Sharma V,Goel P,Kumar S,Singh AK
Plant Cell Rep
2019
https://www.ncbi.nlm.nih.gov/pubmed/30511183

Comprehensive genomic analysis of the TYROSINE AMINOTRANSFERASE (TAT) genes in apple (Malus domestica
Wang H,Dong Q,Duan D,Zhao S,Li M,van Nocker S,Ma F,Mao K
Plant Physiol Biochem
2018
https://www.ncbi.nlm.nih.gov/pubmed/30391815

Genome-wide analyses of genes encoding FK506-binding proteins reveal their involvement in abiotic str
Dong Q,Mao K,Duan D,Zhao S,Wang Y,Wang Q,Huang D,Li C,Liu C,Gong X,Ma F
BMC Genomics
2018
https://www.ncbi.nlm.nih.gov/pubmed/30253753

Genome-Wide Analysis and Cloning of the Apple Stress-Associated Protein Gene Family Reveals MdSAP15,
Dong Q,Duan D,Zhao S,Xu B,Luo J,Wang Q,Huang D,Liu C,Li C,Gong X,Mao K,Ma F
Int J Mol Sci
2018
https://www.ncbi.nlm.nih.gov/pubmed/30134640

An apple sucrose transporter MdSUT2.2 is a phosphorylation target for protein kinase MdCIPK22 in resp
Ma QJ,Sun MH,Lu J,Kang H,You CX,Hao YJ
Plant Biotechnol J
2019
https://www.ncbi.nlm.nih.gov/pubmed/30133123

Mapping QTLs for water-use efficiency reveals the potential candidate genes involved in regulating th
Wang H,Zhao S,Mao K,Dong Q,Liang B,Li C,Wei Z,Li M,Ma F
BMC Plant Biol
2018
```

```
https://www.ncbi.nlm.nih.gov/pubmed/29940853

Significant and unique changes in phosphorylation levels of four phosphoproteins in two apple rootsto
Ren J,Mao J,Zuo C,Calderón-Urrea A,Dawuda MM,Zhao X,Li X,Chen B
Mol Genet Genomics
2017
https://www.ncbi.nlm.nih.gov/pubmed/28710562

Improvement of drought tolerance by overexpressing MdATG18a is mediated by modified antioxidant syste
Sun X,Wang P,Jia X,Huo L,Che R,Ma F
Plant Biotechnol J
2018
https://www.ncbi.nlm.nih.gov/pubmed/28703378

Genome Wide Identification and Characterization of Apple bHLH Transcription Factors and Expression An
Mao K,Dong Q,Li C,Liu C,Ma F
Front Plant Sci
2017
https://www.ncbi.nlm.nih.gov/pubmed/28443104

Comprehensive Genomic Identification and Expression Analysis of the Phosphate Transporter (PHT) Gene
Sun T,Li M,Shao Y,Yu L,Ma F
Front Plant Sci
2017
https://www.ncbi.nlm.nih.gov/pubmed/28424713
```

6. Write some python code to retrieve the structure of two forms of the aspartate transcarbamoylase (PDB ids: 4FYW and 1D09). If you are interested, read more about the Aspartate Transcarbamoylase here[109]. Write a function that gets the .cif file name and prints:
   (a) the number of chains, residues and atoms present in the file;
   (b) a histogram of the residues (plotting it with matplotlib) that are not water (encoded as "HOH");
   (c) a link to an online tool to visualize the 3D structure. The link will be "http://www.rcsb.org/pdb/ngl/ngl.do?pdbid=" followed by the PDB id of the protein (e.g. 1d09).

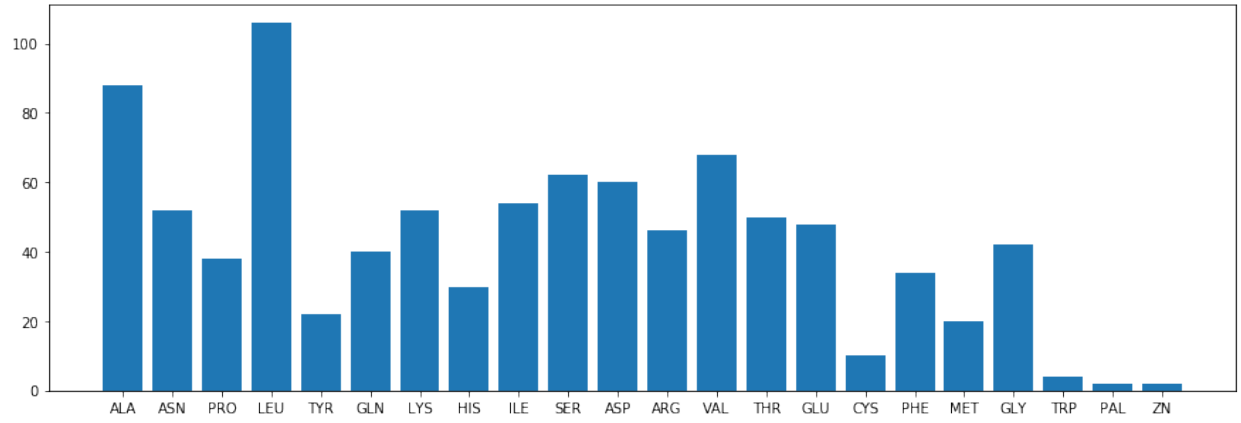
Show/Hide Solution

```
In [59]: from Bio.PDB import *
         import matplotlib.pyplot as plt

         def printCifInfo(filename):

             parser = MMCIFParser(QUIET=True) #To disable warnings
             id = filename.split("/")[1].split(".")[0]

             structure = parser.get_structure(id, filename)
             chains = structure.get_chains()
             residues = structure.get_residues()

             atoms = structure.get_atoms()
             res_histo = {}
             resCnt = 0 #need this because while reading the residues
                        #I am pulling stuff out of the iterator
             for res in residues:
                 rname = res.get_resname()
                 if(rname != "HOH"):
```

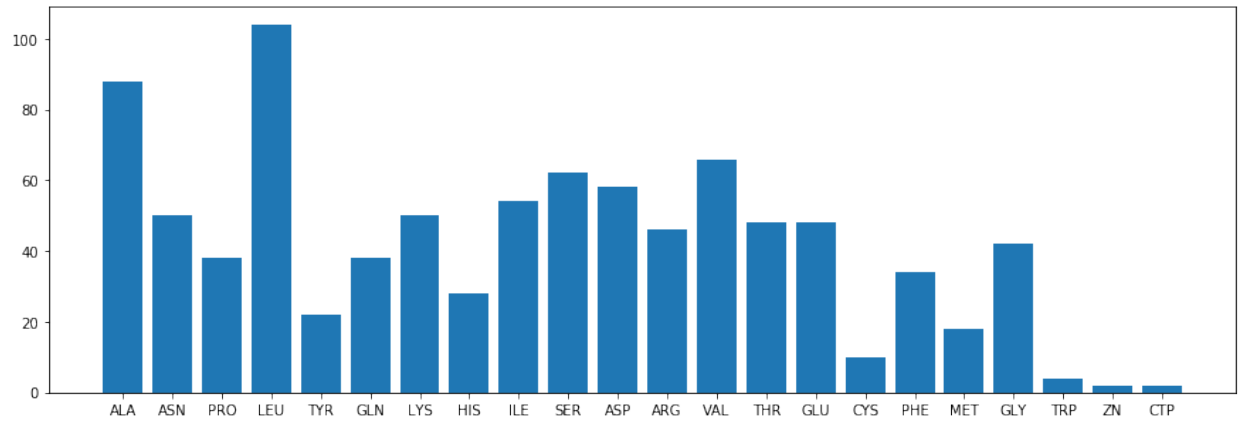
[109] http://pdb101.rcsb.org/motm/215

```
                    if( rname not in res_histo):
                        res_histo[rname] = 1
                    else:
                        res_histo[rname] += 1
                resCnt += 1
            plt.figure(figsize=(15,5))
            plt.bar(res_histo.keys(), res_histo.values())
            plt.show()
            print("Number of chains: {}".format(len(list(chains))))
            print("Number of residues: {}".format(resCnt))
            print("Number of atoms: {}".format(len(list(atoms))))
            print("http://www.rcsb.org/pdb/ngl/ngl.do?pdbid=" + id)

        pdbl = PDBList()
        structures = ["1D09", "4FYW"]
        el = pdbl.download_pdb_files(structures, file_format = "mmCif", pdir = "file_samples/")

        printCifInfo("file_samples/1d09.cif")
        printCifInfo("file_samples/4fyw.cif")
```

```
Structure exists: 'file_samples/1d09.cif'
Structure exists: 'file_samples/4fyw.cif'
```




```
Number of chains: 4
Number of residues: 1547
Number of atoms: 7883
http://www.rcsb.org/pdb/ngl/ngl.do?pdbid=1d09
```




```
Number of chains: 4
```

```
Number of residues: 1483
Number of atoms: 7714
http://www.rcsb.org/pdb/ngl/ngl.do?pdbid=4fyw
```

CHAPTER 13

Midterm test simulation

13.1 Before you start

Please write one single python script.

IMPORTANT: Add your name and ID (matricola) on top of each .py file!

13.2 Problem 1

Given a text string S (possibly on several lines), write a python function that takes the string S and a second string S_1 (**note: assumed to be shorter than S**) in inpu and returns a list with all positions at which S_1 starts in S. If no such positions exist a warning message should be printed.

Ex.

```
S = """TGAATACATATTATTCCGC
GTCCTACCAAAAAATATACATAGATAT
GATACTAGGGGACCGGTTCCGGGATGATGAT"""

print(findPositions(S, "TGA"))
print(findPositions(S, "GAT"))
print(findPositions(S, "none"))
```

should return:

```
String 'TGA' found at the following positions:
[0, 45, 70, 73]

String 'GAT' found at the following positions:
[41, 46, 68, 71, 74]

Warning: string 'none' not found. Returning None
None
```

13.3 Problem 2

The .tsv file `gene_models.tsv` is a compact representation of the gene models on the genome having sequence specified in `sequences.fasta`. The first few lines are reported below:

```
Chr     feature start   end     ID
Chr05   gene    50988   51101   gene:MD05G1000100
Chr05   ncRNA   50988   51101   ncRNA:MD05G1000100
Chr05   gene    83210   83329   gene:MD05G1000200
Chr05   ncRNA   83210   83329   ncRNA:MD05G1000200
Chr05   gene    87650   91333   gene:MD05G1000300
Chr05   mRNA    87650   91333   mRNA:MD05G1000300
Chr05   CDS     87650   87727   CDS:MD05G1000300.8
```

as the header (first line) describes, the first column is the chromosome containing the feature (that can be gene, ncRNA, exon, CDS, ...), the third and fourth columns are the start and end position of the feature on the chromosome and the fifth row is the identifier of the feature ID. **Note that, unlike python strings, features in ''gene_models'' start from coordinate 1 instead of 0.**

The `file sequences.fasta` stores sequence information in .fasta format. A mock entry is the following:

```
>Chr01
AGGCCTAGGTCTTCCAGAGTCGCTTTTTCCAGCTCCAGACCGATCTCTTCAG
AGGCCAATCGCCAGTTTACCACATACACCCAGACCGATCTCTTCAG
```

where the first line is the identifier of the read and starts with a ">". The sequence follows the line with the identifier and can be on multiple lines.

Implement the following python functions:

1. `computeGeneStats(filename)`: gets the `filename` of a .tsv file as explained above, stores its content in a suitable data structure of your choice (hint: pandas might help here), counts (and prints) the number of features of type **gene**, printing the minimum, maximum and average gene length in the whole file. The function should also produce a bar plot of the number of genes per chromosome.

Note: The function should return the data structure containing all the data.

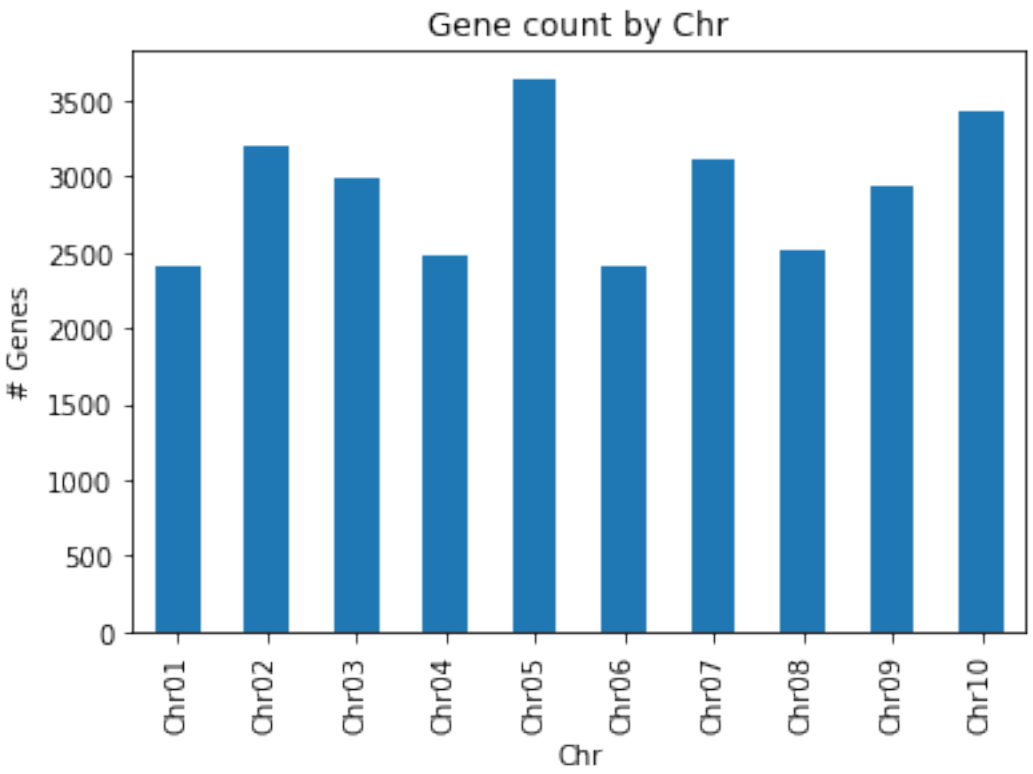
Calling:

```
fn = "gene_models.tsv"
seqFile = "sequences.fasta"
GenesDF = computeGeneStats(fn)
```

should give:

```
29136 genes present
Avg gene length: 2724.71
Min gene length: 45
Max gene length: 53052
```

2. `printSequence(geneInfo, geneID, sequenceFile)`: gets the `geneInfo` data structure **created by computeGeneStats**, a **gene identifier** `geneID` and **the filename of a fasta formatted file** `sequenceFile` and if `geneID` is present in `geneInfo` it **prints its length and a fasta-formatted string reporting the sequence of the gene** Note that chromosome, start and end position are saved in `geneInfo`. If the chromosome is not in the sequenceFile or geneID is not in geneInfo the function should display a corresponding message (see examples below). Optional: add a parameter `outFile` with default value None, that if set (it will contain the path to an output file) makes the function write the fasta entries to file `outFile` rather than to the screen.



Hint: you can use biophtyon to read a fasta file.

For example, assuming an entry:

```
ChrX   gene   3   7   gene:geneid1
```

in `geneInfo` and a sequence:

```
>ChrX
GATTACATAACACACTACA
```

in `sequenceFile`, calling `printSequence(geneInfo,"geneid1",sequenceFile)` should print:

```
Gene geneid1 is in ChrX and has length 5
>geneid1:ChrX:3_7
TTACA
```

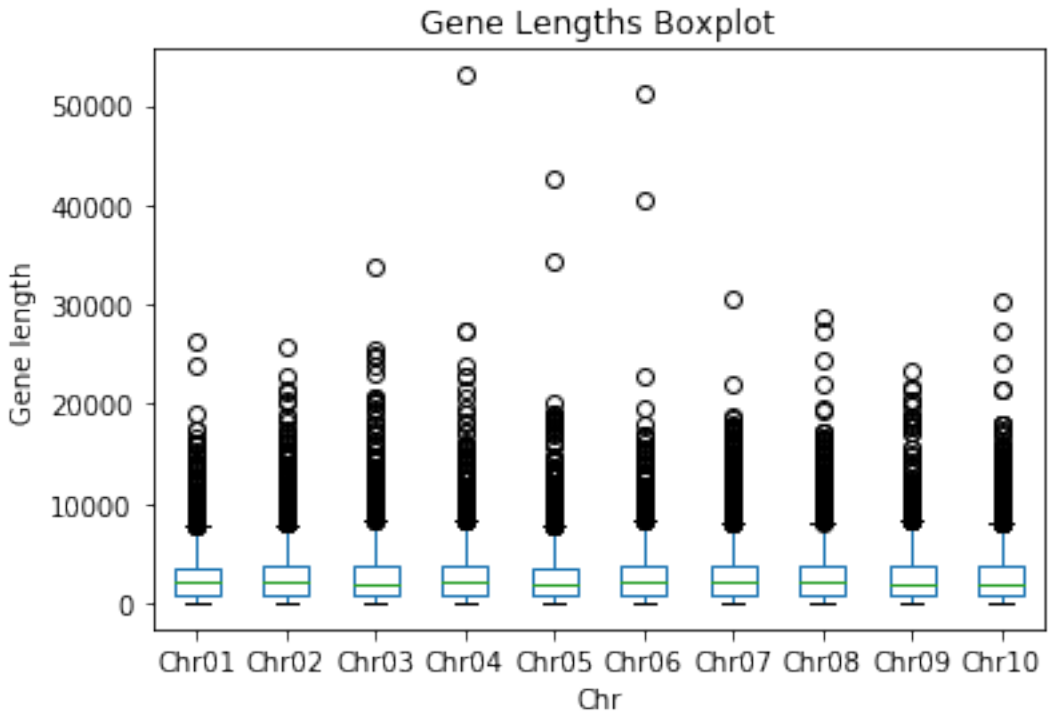
Given `GenesDF` as returned by computeGeneStats, calling:

```
printSequence(GenesDF,"MD05G1000100",seqFile)
print("")
printSequence(GenesDF,"MD03G1000400",seqFile)
print("")
printSequence(GenesDF,"MD08G1000100",seqFile)
print("")
printSequence(GenesDF,"MD08G100019191",seqFile)
```

should give:

```
Gene MD05G1000100 is in Chr05 and has length 114
>MD05G1000100:Chr05:50988_51101
CGCCCAAGGGAGTGGATCCTGTAAGCCTTATATGTATATTCCCATCTCTACCTAGCACGAGGCCTTTTGGGAGCTCATTGGCTTCGGGTTATTGTCACATC

Gene MD03G1000400 is in Chr03 and has length 189
>MD03G1000400:Chr03:21801_21989
CGTATTATAGGAATTCTTCTTCCATTGAAAGGATGGATTGAAAGGATGGATGGCATGTTGTCTATACCTCTCGTCATTAGGATGTACAATAATGGCAGTAT

Gene MD08G1000100 is in Chr08 and has length 5384
Chr08 is not present in the sequence file.

GeneID MD08G100019191 is not present in the input file.
Gene MD05G1000100 is in Chr05 and has length 114
```

3. (Optional) Modify the function `computeGeneStats(filename)` to make it also plot a boxplot of the gene lengths per chromosome.



Expected output:

13.3.1 Available material for exams

The course notes can be used as the slides of the theory and practicals. Documents on the libraries are here:

- matplotlib
- numpy
- pandas
- biopython

You can download all of them here: archive

13.3.2 Download the data

Create a "qcbsciprolab-2019-31-10-NAME-SURNAME-ID" folder on the desktop. Download the following data to the folder that you just created.

The gene models .tsv file: gene_models.tsv and the fasta file with the sequences: sequences.fasta. Note that the files are gzipped, so before using them gunzip them in a terminal with:

```
gunzip sequences.fasta.gz
gunzip gene_models.tsv.gz
```

13.3.3 A possible solution:

13.3.4 Problem 1

```
In [1]: %reset -s -f


        def findPositions(S, S1):
            s1len = len(S1)
            S = S.replace("\n","")
            ret = [x for x in range(0,len(S)) if S[x:x+s1len] == S1]
            if len(ret) > 0:
                print("\nString '{}' found at the following positions:".format(S1))
                return ret
            else:
                print("\nWarning: string '{}' not found. Returning None".format(S1,S))



        S = """TGAATACATATTATTCCGC
        GTCCTACCAAAAAATATACATAGATAT
        GATACTAGGGGACCGGTTCCGGGATGATGAT"""

        print(findPositions(S, "TGA"))
        print(findPositions(S, "GAT"))
        print(findPositions(S, "none"))


String 'TGA' found at the following positions:
[0, 45, 70, 73]

String 'GAT' found at the following positions:
[41, 46, 68, 71, 74]

Warning: string 'none' not found. Returning None
None
```

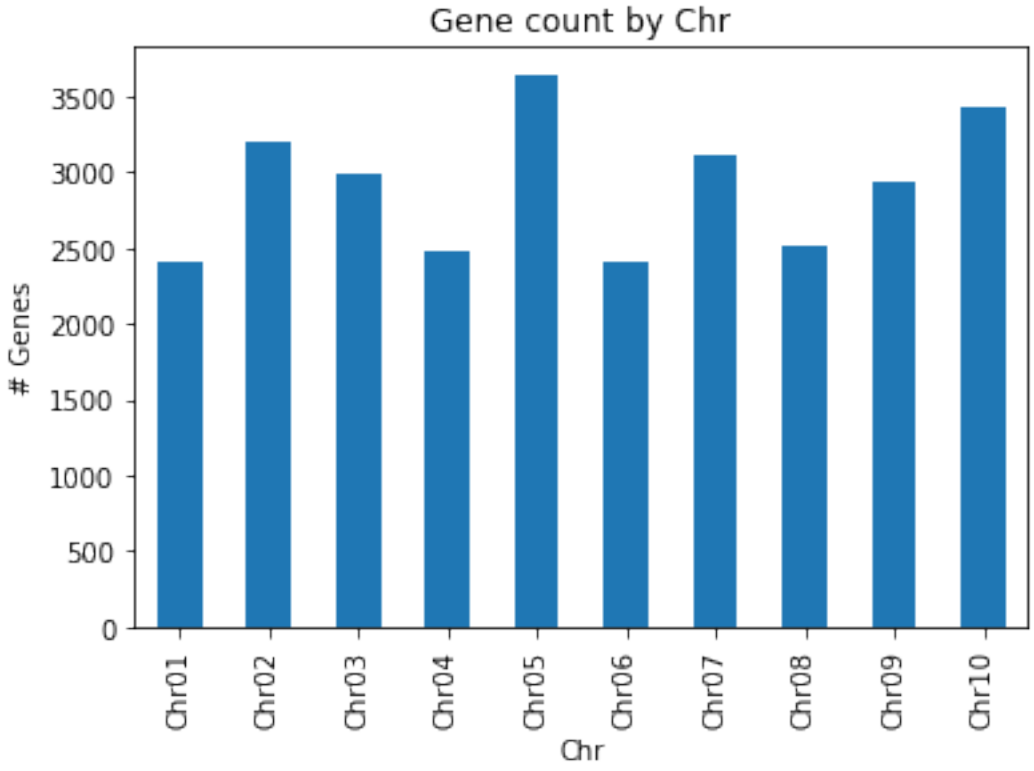
13.3.5 Problem 2

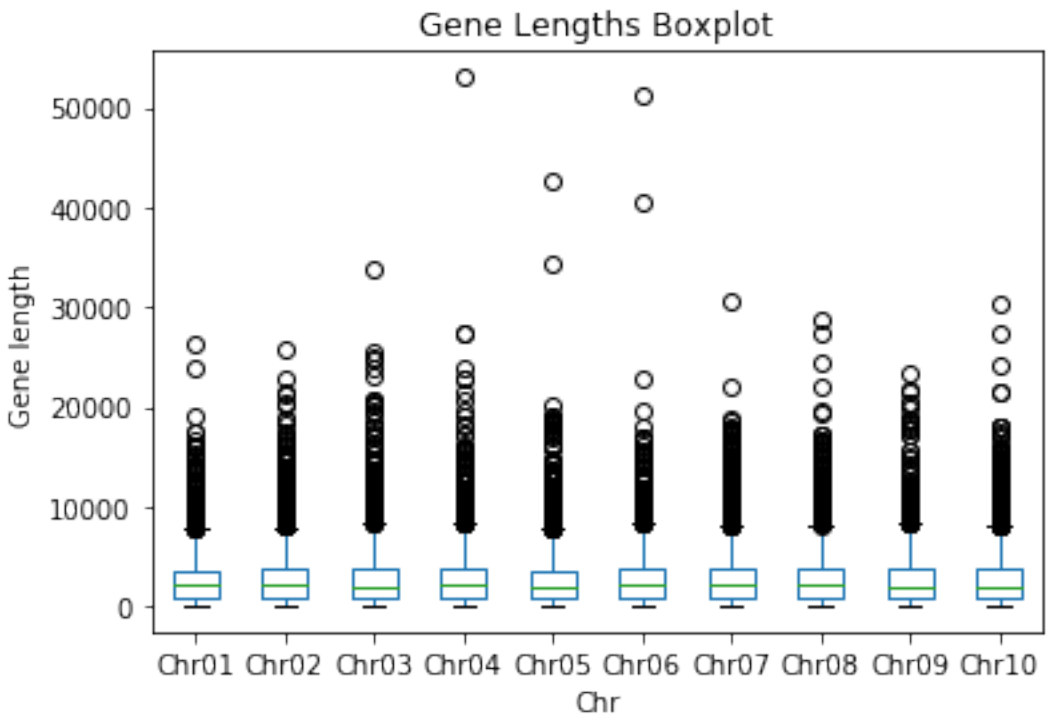
```
In [3]: %reset -s -f
        """exercise.py"""

        import matplotlib.pyplot as plt
        import pandas as pd
```

```
import numpy as np
from Bio import SeqIO


def computeGeneStats(filename):
    geneInfo = pd.read_csv(filename, sep="\t", header = 0)
    genes = geneInfo[ geneInfo["feature"] == "gene"]
    #print(genes.head())
    print("{} genes present".format(genes.shape[0]))
    #print(genes.head())
    avg = np.mean(genes["end"] - genes["start"] + 1)
    m = np.min(genes["end"] - genes["start"] + 1 )
    M = np.max(genes["end"] - genes["start"] + 1 )
    print("Avg gene length: {:.2f}\nMin gene length: {}\nMax gene length: {}".format(avg, m,

    grpd = genes.groupby(["Chr"])
    agg = grpd.aggregate(pd.DataFrame.count)["feature"]
    agg.plot(kind = 'bar')
    plt.title("Gene count by Chr")
    plt.ylabel("# Genes")
    plt.xlabel("Chr")
    plt.show()

    bpdata = {}
    for i, g in grpd:
        if(i not in bpdata):
            bpdata[i] = []
        bpdata[i] = g["end"] - g["start"] + 1

    pdData = pd.DataFrame(bpdata)
    pdData.plot(kind='box')
    plt.title("Gene Lengths Boxplot")
    plt.ylabel("Gene length")
    plt.xlabel("Chr")
    plt.show()

    return geneInfo

def printSequence(geneInfo, geneID, sequenceFile, outFile = None):
        gn = "gene:" + geneID
        gene = geneInfo[geneInfo["ID"] == gn]
        of = None
        if outFile != None:
            of = open(outFile, "w")

        if len(gene) > 0:
            seqDict = SeqIO.to_dict(SeqIO.parse(sequenceFile, "fasta"))
            chr = gene.iloc[0]["Chr"]
            start = gene.iloc[0]["start"]
            end = gene.iloc[0]["end"]
            print("Gene {} is in {} and has length {}".format(geneID,chr, end-start + 1))
            if chr in seqDict:
                s = seqDict[chr].seq[start-1:end ]
                if outFile != None:
                    of.write(">"+geneID + ":" + chr +":" + str(start) + "_" + str(end)+"\n")
                    of.write(str(s)+"\n")
                else:
                    print(">"+geneID + ":" + chr +":" + str(start) + "_" + str(end))
                    print(str(s))
```

```
            else:
                print(chr, "is not present in the sequence file.")

        else:
            print("GeneID {} is not present in the input file.".format(geneID))

        if outFile != None:
            of.close()


fn = "./midterm_sim/gene_models.tsv"
seqFile = "./midterm_sim/sequences.fasta"

GenesDF = computeGeneStats(fn)
printSequence(GenesDF,"MD05G1000100",seqFile)
print("")
printSequence(GenesDF,"MD03G1000400",seqFile)
print("")
printSequence(GenesDF,"MD08G1000100",seqFile)
print("")
printSequence(GenesDF,"MD08G100019191",seqFile)

printSequence(GenesDF,"MD05G1000100",seqFile, "MD05G1000100.fasta")
```

```
29136 genes present
Avg gene length: 2724.71
Min gene length: 45
Max gene length: 53052
```






```
Gene MD05G1000100 is in Chr05 and has length 114
>MD05G1000100:Chr05:50988_51101
CGCCCAAGGGAGTGGATCCTGTAAGCCTTATATGTATATTCCCATCTCTACCTAGCACGAGGCCTTTTGGGAGCTCATTGGCTTCGGGTTATTGTCACATC

Gene MD03G1000400 is in Chr03 and has length 189
>MD03G1000400:Chr03:21801_21989
CGTATTATAGGAATTCTTCTTCCATTGAAAGGATGGATTGAAAGGATGGATGGCATGTTGTCTATACCTCTCGTCATTAGGATGTACAATAATGGCAGTAT

Gene MD08G1000100 is in Chr08 and has length 5384
Chr08 is not present in the sequence file.

GeneID MD08G100019191 is not present in the input file.
Gene MD05G1000100 is in Chr05 and has length 114
```

CHAPTER 14

14.1 Index