python training en

Release 0.1

Kamil Pazik

Dec 03, 2019

CONTENTS:

CHAPTER

ONE

INTRO

1.1 Intro

1.1.1 About Python

- Open Source,
- Name comes from BBC show Flying Circus of Monty Python. Creator is fan of the series,
- Development of **Python** (interpreter) started in 1989,
- Guido van Rossum - was creator and dictator for python. He is like Linus Torvalds for Linux kernel,
- Created to create system tool for very specified Amoeba system (since there were no Coreutils and creating new applications using C/Assembler would take long time),
- Language between C and Shell (bash),
- Python 2.0 – October 2000,
- Python 3.0 - December 2008

1.1.2 Usage

- DevOps
  - Boto3,
  - Redhat - tools,
    - Installer **Anaconda**,
    - system-config-network-tui,
    - system-config-services,
    - others *system-config-*,
    - Package installers (**yum - python2, dnf - python3**),
    - OpenStack,
    - Ansible (management of configuration / deployment)
- Data Science / Machine Learning
  - sklearn,
  - Tensorflow,

  - pySpark
- Web Development
  - Django,
  - Flask

1.1.3 Who's using

- Google – as a main language (at least in the past), next to Java and C++. For processing great amount of data from the users,
- Netflix – for scaling infrastructure, alerts in case of change security settings,
- Instagram (framework Django), Facebook (Framework Tornado),
- Spotify (Big volume of data to process – Luigi),
- Nasa

1.2 Installation

1.2.1 Windows

- Installer exe available on the Python page

1.2.2 Mac

- Can be installed using brew,
- Eventually from page,
- Best would be using pyenv

Hint: You may use commands like:

```
pyenv versions # to show all available versions
pyenv global # to see global configuration
pyenv install 3.6.9 # to install specific version of python
pyenv global 3.6.9 # to setup global version of python
```

1.2.3 Linux

- On **debian** or **ubuntu** `sudo apt-get update && sudo apt-get install python python3-pip python3-venv`
- On **fedora** `sudo dnf install python` or specific version `sudo dnf install python37`

Hint: For checking current version we execute this command: `python --version` or `python -V`

1.2.4 Source code

- Clone of the repository from github

1.3 Python run environment

Python can be launched in couple ways.

1.3.1 Virtual Environment (venv)

- Keeps environment separate,
- Solves problem of dependencies, packages conflicts,
- Helps keeping different **python/libraries** in our projects

---

Note: In order to create virtual environment we do **one of below** commands - `python -m venv <DIR>` instead *DIR* usually we use *venv* or *env* - `virtualenv <KATALOG>` ex. `virtualenv venv`

Note: In order to use environments we need to execute: `source venv/bin/activate`

Hint: In case of **Windows** user is activating env withouth **source** command. `venv\Scripts\activate.bat`

1.3.2 Global environment

- All packages are global - there is no separation,
- There might problems with dependencies

1.3.3 Environment in container

- Python available within container,
- Good in case of testing solutions,
- Integral part of nowadays **CI/CD** environments

1.3.4 Creation of new environment

1. `python -m venv venv`,
2. `source venv/bin/activate`,

1.3.5 Installation of new packages inside of environment

1. *Optional step* `pip freeze`,
2. `pip install <package_name>`,
3. `pip freeze` Just to verify what has been installed,
4. *Optional step* `pip freeze > requirements.txt`

1.3.6 Exercise

This exercise will show you typical use case of virtual environments.

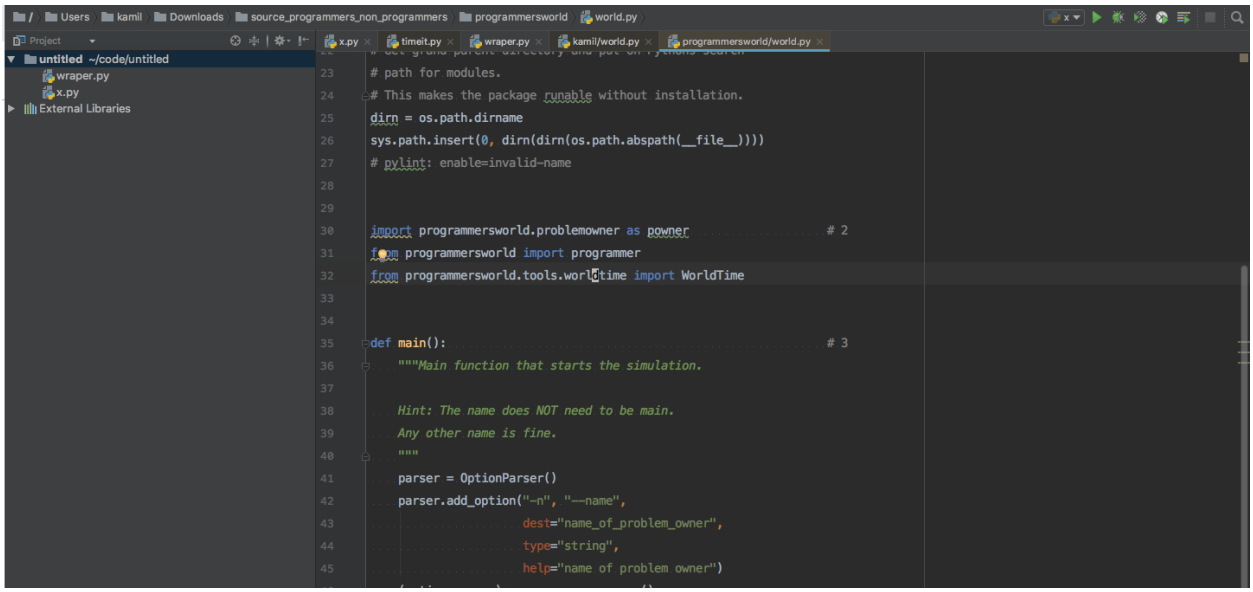
Attention:

1. Create new virtual environment,
2. Activate virtual env,
3. Look on `pip freeze`,
4. Install django,
5. Look on `pip freeze`,
6. Deactivate virual env,
7. Remove virtual env,
8. Start from scratch,
9. Install notebook

1.4 Ide

1.4.1 pyCharm

Paid solution from JetBrains * A lot of plugins, * Support of frameworks, * Integraion with Docker, database, console, cloud solutions



1.4.2 Visual Studio Code

- Free tool,
- You need to install plugins

1.4.3 Eclipse

- Free tool

1.4.4 Spyder

- Free toll
- Used in science purposes or data analysis. Niche, replaced by **Jupyter Notebook**

1.4.5 IDLE

- Free tool,
- Not recomended,
- Hard development proess

1.5 Python

1.5.1 Structured programming

- Its possible to write simple scripts

1.5.2 Object programming

- Everything is an object

1.5.3 Functional programming

```
>>> digits = [1, 2, 3, 4, 5]
>>> power_of_two = [2**n for n in digits]
>>> power_of_two
[2, 4, 8, 16, 32]
```

1.5.4 Dynamic typing

- Types are defined during program execution,
- On the one hand - freedom, on the other hand - slower execution,
- No compiling - errors are appearing after execution of line of code not during compilation

```
place  = 43 # int
place = "near window" # str
print(place)
```

The variable will be overwritten (also type will be changed)

```
near window
```

1.5.5 Garbage collector

- Manages of data cleaning,
- Based on algorithm counting occurrences of **references** to specific object

1.5.6 Passing by reference instead of by value

- As default reference of an object is passed (to safe memory),

1.6 Python shells

1.6.1 python

- Default **shell** delivered with python installation

1.6.2 ipython

- easy to install,
- command reverse search **ctrl-r**,
- has additional functionalities,

- colorful

Hint: %timeit # to measure time !ping # or whatever command

1.6.3 jupyter-notebook

- based on **ipython**,
- most often used within **data science** teams,
- additional functions like printing

CHAPTER

TWO

BASICS

2.1 Strings

2.1.1 Printing strings

```
>>> print('Hello World!')
Hello World!
```

Or using double **quotation** mark

```
>>> print("Hello World!")
Hello World!
```

Note: **Single ‘ quotation** and **Double ” quotation** works almost same. But if you want to put ' into quotation mark you need to mix quotations characters ex.

```
>>> print("It's a nice day")
It's a nice day
```

But if you use ' twice you will see syntax exception

```
>>> print('It's a nice day')
Traceback (most recent call last):
SyntaxError: invalid syntax
```

2.1.2 Defining strings

```
txt = "Hello World!"
print(txt)
```

```
Hello World!
```

2.1.3 Checking types

```
>>> type(txt)
<class 'str'>
```

2.1.4 Checking length of string

```
>>> len(txt)
12
```

2.1.5 Printing special characters

- New line special character

```
>>> print("Hello\nWorld!")
Hello
World!
```

- Tabulator character

```
>>> print("Hello\tWorld!")
```

2.1.6 String concatenation

```
>>> print('Hello ' +  'attendee')
Hello attendee
```

2.1.7 String concatenation - format

```
>>> print('Hello {}, have a great day'.format('Tomasz'))
Hello Tomasz, have a great day
```

2.1.8 Different representations - format

```
>>> '{:s}'.format('Some text') # in case of digit - exception
'Some text'
```

```
>>> '{:s}'.format(4) # in case of digit - exception
Traceback (most recent call last):
ValueError: Unknown format code 's' for object of type 'int
```

```
class Data:
    """Simple Data class"""

    def __init__(self, value):
        self.value = value

    def __str__(self):
        return '{}'.format(self.value)

    def __repr__(self):
        return '<{} object with value: {}>'.format(self.__class__.__name__, self.
↪value)
```

(continues on next page)

(continued from previous page)

```
print("{0!s}".format(Data(54), Data(54)))
print("{0!r}".format(Data(54), Data(54)))
print("{obj!s}".format(obj=Data(41)))
print("{obj!r}".format(obj=Data(41)))
```

```
54
<Data object with value: 54>
41
<Data object with value: 41>
```

```
>>> '{:>10}'.format('test')
'      test'
```

```
>>> '{:10}'.format('test')
'test      '
```

```
>>> '{:^10}'.format('test')
'   test   '
```

2.1.9 Functions available on srings

```
>>> 'Hello'.endswith('o')
True
```

```
>>> 'Hello'[-1] == 'o'
True
```

2.1.10 Substrings

```
>>> 'Hello'[-1]
'o'
```

```
>>> 'Hello'[0:6:2]
'Hlo'
```

```
names = 'Marta, Kasia, Monika, Tomek, Przemek, Janek, Marta, Malgosia'

print(names.count('Ma'))
```

In result we receive number of occurrences

```
3
```

Below we find an index of string.

```
>>> names.find('Kasia')
7
```

Hint: Letter **'K'** is at 8th position, which means index no. 7 (countring from 0)

2.1.11 Splitting strings

```
>>> names.split(',')
['Marta', ' Kasia', ' Monika', ' Tomek', ' Przemek', ' Janek', ' Marta', ' Malgosia']
```

We received list of strings

2.1.12 Operations of strings

```
>>> names = names.replace("Janek", "Adam")
>>> print(names)
Marta, Kasia, Monika, Tomek, Przemek, Adam, Marta, Malgosia
```

2.1.13 Checking if string is a digit

```
>>> names.isdigit()
False
```

```
>>> temperature = "34"
>>> print(temperature.isdigit())
True
```

2.1.14 String as uppercase

```
>>> print(names.upper())
MARTA, KASIA, MONIKA, TOMEK, PRZEMEK, ADAM, MARTA, MALGOSIA
```

2.1.15 String as lowercase

```
>>> print(names.lower())
marta, kasia, monika, tomek, przemek, adam, marta, malgosia
```

2.1.16 Exercise

1 Create program writing your name and surname 2 Print fallowing statement: "Test characters: ', /, " " 3 Create two attendees of this training (give them names which you like) *(both attendees are seperate variables)*

- first_attendee,
- second_attendee

4 Exchange places of attendees - `first_attendee` should have `second_attendee` content and other way round

- Print attendees,

- Is it possible to change places in different way ?

5 Let user put his name using keyboard **(Use google)**

2.2 Integer number

2.2.1 Defining

```
>>> net_salary = 8000
>>> print(net_salary)
8000
```

2.2.2 Checking types

```
>>> type(net_salary)
<class 'int'>
```

Comparing types: string and integer

```
>>> salary_str = '8000'
>>> salary_str == net_salary
False
```

2.2.3 Converting types

```
>>> salary_converted = int(salary_str)
>>> salary_converted == net_salary
True
```

2.2.4 Operations on digits

After salary increase we get 5% more money

```
>>> net_salary = net_salary*1.05
>>> print(net_salary)
8400.0
```

```
>>> type(net_salary) == float
True
```

```
>>> type(net_salary)
<class 'float'>
```

2.2.5 Adding

- we received 200 PLN monthly bonus

```
>>> net_salary += 200
>>> net_salary = int(net_salary)
>>> print(net_salary)
8600
```

2.2.6 Integer division

- We want to calculate net income **per person** in 3 persons family,
- We want to round the income to the 2nd decimal place (rounding)

```
>>> print(round(net_salary / 3, 2))
2866.67
```

```
>>> net_salary // 3
2866
```

```
>>> round(net_salary / 3)
2867
```

As we see we lost precision. Values after commas has been ignored

2.2.7 Modulo division

- We want to check if our salary is **even** *(divisible by two)*

```
>>> print(net_salary % 2)
0
```

There is no **reminder** so it's **even** number

2.3 Float numbers

2.3.1 Defining

```
>>> net_salary = 8000.63
>>> print(net_salary)
8000.63
```

2.3.2 Checking type

```
>>> type(net_salary)
<class 'float'>
```

2.3.3 Type conversion

```
>>> salary_converted = int(net_salary)
>>> salary_converted == net_salary
False
```

2.3.4 Interesting facts

```
>>> print(0.1 + 0.2)
0.30000000000000004
```

Hint: Answer on the page.

Additional wiki page about **IEEE 754**.

```
import decimal

ctx = decimal.getcontext()
print(ctx)

a = decimal.Decimal(0.2)
b = decimal.Decimal(0.1)

ctx.prec = 6
print(a + b)
```

2.3.5 Exercise

1. Calculate sum of 123, 321, 675 and print result on the screen,
2. Check if sum is multiples of the number **5**,
3. Calculate **income tax** (tax rate is 19%) user is giving the amount **(input)**. Assuming tax free allowance is 5000,
4. Calculate **area** of circle (with given **radius** by the user)

Hint: Use **input** function. You may also import additional module - search for it using **google**

2.4 Lists

2.4.1 Defining

```
>>> salary_list = [4000, 5000, 3000, 8000]
>>> print(salary_list)
[4000, 5000, 3000, 8000]
```

2.4.2 Checking type

```
>>> type(salary_list)
<class 'list'>
```

2.4.3 Operation on lists

- Checking list type

```
>>> len(salary_list)
4
```

- Checking occurrences withing list

```
>>> 3000 in salary_list
True
```

- Adding element into list

```
salary_list.append(12000)
print(salary_list)
```

```
[4000, 5000, 3000, 8000, 12000]
```

- Adding element into concrete place into list

```
salary_list.insert(1, 4500)
print(salary_list)
```

```
[4000, 4500, 5000, 3000, 8000, 12000]
```

- List sorting

```
salary_list = sorted(salary_list)
print(salary_list)
```

```
[3000, 4000, 4500, 5000, 8000, 12000]
```

```
print(sorted(salary_list, reverse=True))
```

```
[12000, 8000, 5000, 4500, 4000, 3000]
```

- List sorting **(in place)**

```
salary_list.extend([2800, 15000])
salary_list.sort()
print(salary_list)
```

```
[2800, 3000, 4000, 4500, 5000, 8000, 12000, 15000]
```

- Getting last element from the list

```
>>> retreived = salary_list.pop()
>>> print(retreived)
15000
```

Again we put element to the list

```
>>> salary_list.append(retreived)
>>> print(salary_list)
[2800, 3000, 4000, 4500, 5000, 8000, 12000, 15000]
```

Let's put element which is incorrect (negative salary)

```
>>> salary_list.append(-300)
>>> print(salary_list)
[2800, 3000, 4000, 4500, 5000, 8000, 12000, 15000, -300]
```

This negative is not needed - let's remove it:

```
>>> del salary_list[-1]
>>> print(salary_list)
[2800, 3000, 4000, 4500, 5000, 8000, 12000, 15000]
```

- Iterating over the list

```
for salary in salary_list:
    print(salary)
```

```
2800
3000
4000
4500
5000
8000
12000
15000
```

2.4.4 Type conversion

- into **tuple**

```
salary_tuple = tuple(salary_list)
print(salary_tuple)
```

```
(2800, 3000, 4000, 4500, 5000, 8000, 12000, 15000)
```

- into **set**

```
>>> set_plac = set(salary_list)
>>> print(set_plac)
```

2.4.5 Exercises part 1

1. Create list containing:

    - Audi
    - Bmw
    - Mercedes
    - Mazda
2. Replace **Audi** with **Mazda**
3. Remove last car
4. Print last car on the list

2.4.6 Exercised part 2

1. Create list of temperatures `[-5, -4, 0, -3, -2, 9, 10]`,
2. Sort descending - **in place**,
3. Sort ascending - **not in place**

2.5 Dictionaries

Type of data **key - value**

2.5.1 Defining

```
>>> workers = {1: 'Adam', 3: 'Tomasz', 4: 'Kasia'}
>>> print(workers)
```

2.5.2 Checking type

```
>>> type(workers)
<class 'dict'>
```

2.5.3 Operations on dictionary

- Checking length of dictionary

```
>>> len(workers)
3
```

- Checking element occurrences

```
>>> 3000 in workers
False
```

```
>>> 1 in workers
True
```

Employee with `id 1` exists inside of dictionary

- Adding element to the list

```
>>> workers[15] = "Marek"
>>> print(workers)
>>> print(len(workers))
4
```

2.5.4 Exercises

1. Create dictionary with capitals of:

- France,
- Germany,
- Poland,
- Czech republic

1. Get capital of **Uk** - in case of not having capital within dictionary print `"unknown capital`
2. Remove capital of Czech republic from dicionary,

Hint: Look for the method which is giving you some text in case of not having specific key inside of dict

---

2.6 Set

Type of data - same as in mathematics

2.6.1 Defining

```
>>> A = {1, 2, 3, 4, 5}
>>> B = {4, 5, 6, 7, 8}
```

2.6.2 Checking type

```
>>> type(A)
<class 'set'>
```

2.6.3 Operation on sets

- Checking set length

```
>>> len(A)
5
```

- Checking element occurrence

```
>>> 17 in A
False
```

```
>>> 3 in A
True
```

- Adding element to the set

```
>>> A.add(17)
>>> 17 in A
True
```

- Union

```
C = A | B
print(C)
```

```
{1, 2, 3, 4, 5, 6, 7, 8, 17}
```

```
>>> print(C.issuperset(A))
True
```

```
>>> print(C.issuperset(A))
True
```

- Intersection - Common set

```
>>> D = A & B
>>> print(D)
{4, 5}
```

- Difference

```
E = A - B
print(E)
```

```
{1, 2, 3, 17}
```

```
F = B - A
print(F)
```

```
{8, 6, 7}
```

- Symmetric difference

```
>>> print(A.symmetric_difference(B))
{1, 2, 3, 17, 6, 7, 8}
```

2.6.4 Immutable sets

```
>>> A = frozenset([1, 2, 3, 4])
```

2.6.5 Implementation

Code written in C to review.

2.6.6 Exercises - part 1

- having sets:
  - A = {'wp.pl', 'onet.pl', 'google.com', 'ing.pl', 'facebook.com'}
  - B = {'wp.pl', 'youtube.pl', 'wikipedia.org', 'ovh.com', 'facebook.com'}
- Find:
  - Instersection of domains
  - Domains existing in just one of the sets

2.6.7 Exercises - part 2

- Having list [1, 2, 4, 5, 7, 7, 7] print only unique values

CHAPTER

THREE

CODE FLOW

3.1 If else statements

- if

```
if True:
    print('True value')
```

Upper code is returning following text:

```
True value
```

```
if False:
    print('False value')
```

As you there is nothing printed.

3.1.1 Conversion of list into boolean

```
empty_list = []

if empty_list:
    print('List with content')
else:
    print('List empty')
```

Text above is returning following text:

```
List empty
```

Attention: What happened here is implicit conversion of type list into bool

```
>>> bool([])
False
```

```
>>> bool([1, 2, 3])
True
```

3.1.2 Checking bool values

```
>>> bool(-1)
True
```

```
>>> bool(0)
False
```

```
>>> bool(124)
True
```

```
>>> bool({})
False
```

Warning: Every number but not zero will return `True`. Number -1 if written in binary (U2) got got ones (1) on decimal positions

3.1.3 Checking ranges

```
temperature = 18

if 16 <= temperature < 24:
    print('Temperature good for biking')
else:
    print('Temperature not appropiate for biking')
```

This will give us

```
Temperature good for biking
```

If we change temperature to **below zero**

```
temperature = -3

if 16 <= temperature < 24:
    print('Temperature good for biking')
elif 3 <= temperature < 16:
    print('Temperature good for walk')
elif -5 <= temperature < 3:
    print('Temperature good for skiing')
else:
    print('Don`t know what to do :(')
```

```
Temperature good for skiing
```

3.1.4 Exercises - part 1

1. Let user put his age, check if he is adult,
2. Let user put number, check if the value is float or integer

Hint: There are many ways to do that. Find your own ;)

Exercises - part 2

1. Create simple BMI calculator, which will get all values from `input`. In result it should return status if person is:
   - Overweight,
   - Normal,
   - Underweight

Exercises - part 3

- Use library `os` function `system` for checking if host is active
  - Depending on a status print proper message,

Hint: Bear in mind that systems got own status codes after execution commands

3.2 Loops

3.2.1 for loop

```
for i in range(5):
    print(i)
```

```
0
1
2
3
4
```

```
for i in range(2, 6):
    print(i)
```

```
2
3
4
5
```

```
for i in range(2, 7, 2):
    print(i)
```

```
2
4
6
```

3.2.2 while

```
# i is our "iterator variable"
i = 0
while i < 10:
    print(i)
    i += 1
```

```
0
1
2
3
4
5
6
7
8
9
```

```
# i is our "iterator variable"
i = 6
while i < 10:
    if i == 7:
        print('Lucky 7')
        i += 1
        continue

    print(i)

    i += 1
```

```
6
Lucky 7
8
9
```

```
# i is our "iterator variable"
i = 6
while i < 10:
    if i == 7:
        print('Lucky 7')
        break

    print(i)

    i += 1
```

```
6
Lucky 7
```

3.2.3 iterate over iterables

```
for car in ['BMW', 'Audi', 'Mercedes']:
    print(car)
```

```
BMW
Audi
Mercedes
```

3.2.4 Exercise - part 1

1. Create `dictionary of hosts` where you store date of connection check and status if it went well
   - You can define list of hosts ex. `wp.pl, google.com, ing.pl, nonexisting.domain`

Hint: You may get date by using `datetime`, you may also get the data from system using `os.popen`

3.2.5 Exercise - part 2

- create list of even numbers from 0 to 100,
- print this list

3.3 List/Dict/Set comprehensions

Its used for code readability

Hint: At first, its better to create code withouth comprehenstion, later if you got experience you may try to make code with "comprehensions"

```
even_numbers = [element for element in range(2, 21, 2)]
print(even_numbers)
```

```
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
```

```
even_numbers2 = [element for element in range(2, 21) if (element % 2) == 0 ]
print(even_numbers2)
```

```
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
```

```
even_numbers3 = [element for element in range(2, 21) if not (element % 2)]
print(even_numbers3)
```

```
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
```

3.3.1 Dict comprehension

```
data_dict = {'Adam': 'Audi', 'Tomek': 'BMW', 'Kasia': 'Citroen'} # doctest: +SKIP
```

3.3.2 Set comprehension

```
data_set = {i**2 for i in range(5)}
print(data_set)
```

```
{0, 1, 4, 9, 16}
```

3.3.3 Exercise part 1

1. Find 20 numbers divisible by 2 or divisible by 5 (if you don't know how to make list comprehension, create normal list)

3.3.4 Exercise part 2

1. Create mapping (dict comprehension)

- **key** is number, values are letters from the alphabet A-Z,
- `{0: 'A', 1: 'B', 2: 'C', 3: 'D', ...}`

Hint: Take a look on ASCII table

- You can convert number to character,,
- You can use `chr()` function - check **google**

3.3.5 Exercise part 3

1. having **lista dat** in file

- Create list - result of processing file gettting just date (list comprehension). * Based on this list - create list, where event occured in 19th second

3.4 Functions

- Gives possibility to reuse code,
- Gives as chance to track code,
- Splitting is more logical than executing code line by line

Different definitions of functions

```
def function_a():
    """Docstring documenting function"""

    print('This is simple function')

# "Execution" of a function
function_a()
```

```
This is simple function
```

3.4.1 Function with parameteres

```
def sum_of_three_numbers(a, b, c):
    """Function calculating sum of three numbers"""

    print(a + b + c)

result = sum_of_three_numbers(3, 5, 8)
print(result)
```

```
16
None
```

```
def sum_of_four_numbers(a, b, c=0, d=0):
    """Simple function suming 4 numbers with default 4th param"""

    return (a + b + c + d)

print(sum_of_four_numbers(3, 5))
print(sum_of_four_numbers(3, 5, 8))
print(sum_of_four_numbers(3, 5, 8, 16))
```

```
8
16
32
```

3.4.2 Args

```
def sum_of_many(show, *nums):
    res_sum = 0

    for num in nums:
        res_sum += num

    if show:
        print('Sum equals to {}'.format(res_sum))
    return res_sum

res = sum_of_many(True, 1, 2, 3, 4, 5, 6, 7)
print(res)
```

```
Sum equals to 28
28
```

3.4.3 Kwargs

```
def res_salary_sum(**kwargs):
    """Sums all people"""

    res_sum = 0

    for person, salary in kwargs.items():
        res_sum += salary
    return res_sum

print(res_salary_sum(Adam=3000, Tomek=2500, Kasia=4320))
```

```
9820
```

3.4.4 Exercises part 1

1. Create function checking if person is adult
2. Function should use 2 arguments (name of person and age)

3.4.5 Exercises part 2

- Create function which would be checking strength of password (own algorithm)
- Password can have at least 6 characters
- Password is stronger, when:
  - It has uppercase letters,
  - Has numbers,
  - Has special character (you can define list of special characters on your own ex. `['_', '*', '&']`)

3.4.6 Exercises part 3

1. Modify code calculation BMI - now it should be function

- Takes additional parameter - name,

3.4.7 Exercises part 4

- Create function `report_salary(team, stats=True, *args)` which for specific team returns average salary of the team, round the result to the 2nd decimal place
- Additionaly if flag `stats` is on print statistics:
  - Average,
  - Median,
  - Minimal value,
  - Maximal value

Hint: To calculate **median** either you can create own function. But also you can create function form libraries.

3.5 Exceptions

- In case of execution illegal operation,
- In case of resource being unavailable for us - ex. no access rights / not enough memory / servers is unavailable.

3.5.1 Syntax Errors

```
>>> while True print('Hello world')
File "<stdin>", line 1
while True print('Hello world')
               ^
SyntaxError: invalid syntax
```

3.5.2 Key Errors

```
capitals = {"France": "Paris", "Germany":"Berlin", "Poland":"Warsaw", "Check-republic
↪":"Praga"}
capitals["USA"]

KeyError: 'USA'
```

3.5.3 Attribute error

- If operation not possible to be done

```
"Hello Wordl".append('!')
```

3.5.4 Indentation Error

```
def testfunc():
print('Hello ;)')
 print('My name is:')

File "<ipython-input-4-9cd3c6fb52a1>", line 3
 print('My name is:')
 ^
IndentationError: unexpected indent
```

3.5.5 ModuleNotFoundError

```
import not_existing_module

ModuleNotFoundError: No module named 'not_existing_module'
```

Table of exceptions hierarchy in Python.

3.5.6 IndexError

```
attendees = ['Kasia', 'Adam', 'Tomek']
attendees[6]

IndexError: list index out of range
```

3.5.7 Exception handling

```
for i in range(3, -3, -1):
 try:
     print('Try of division by {}'.format(i))
     3 / i
 except ZeroDivisionError:
     print('Skipping, illegal operation !!!')

 finally:
     print('End of handling')
```

```
Try of division by 3
End of handling
Try of division by 2
End of handling
Try of division by 1
End of handling
Try of division by 0
Skipping, illegal operation !!!
End of handling
Try of division by -1
End of handling
Try of division by -2
End of handling
```

3.5.8 Raising an exception

```
def generate_report(input_data, outputfile):
    raise NotImplementedError('Function development still in progress')

NotImplementedError: Function development still in progress
```

3.5.9 Exercises part 1

1. You got list of attendees

- attendees = [“Kasia”, “Adam”, “Tomek”]

1. Handle the situation when

- Element no. 5 is gathered,

1. Handle situation when trying to access to capital of **Italy**
    - use `capitals = {"France": "Paris", "Germany":"Berlin", "Poland":"Warsaw", "Check-republic":"Praga"}`

3.6 Iterators

- Lazy evaluation,
- Memory efficient,
- Used in many places
  - open,
  - zip,
  - enumerate,
  - reversed

```
from typing import Iterable
print(issubclass(range, Iterable))
```

```
True
```

Hint: You can check different types in same way ex. lists, strings

```
from typing import Iterable, Iterator

print(isinstance(range(10), Iterable))
print(hasattr(range(10),'__iter__'))
print(callable(range(10).__iter__))
print(isinstance(iter([1,2]) , Iterator))
```

```
True
True
True
True
```

3.6.1 Iterators vs lists

```
# Not using too much memory - iterating on the fly
for i in ( i ** 2 for i in range(10**8)):
    print(i)
```

```
# using a lot of memory

lista = [ i ** 2 for i in range(10**8)]
```

Hint: Compare proces for list and generator using `ps aux PID` Additionally you may use linux function `watch -d -n 0.1`

3.6.2 Defining iterators

```
class Numbers:
    def __iter__(self):
        self.value = 1
        return self

    def __next__(self):
        value = self.value
        self.value += 1
        return value

numbers = Numbers()
my_iter = iter(numbers)

print(next(my_iter))
print(next(my_iter))
print(next(my_iter))
```

```
1
2
3
```

3.6.3 Zip

```
from typing import Iterable, Iterator

za = zip([1,2,3], ['a', 'b', 'c'])
print(isinstance(za, Iterable))
print(isinstance(za, Iterator))
```

```
True
True
```

3.6.4 Exercises

1. We got list of expenses in specific days of the week

- expenses = [11.25, 18.0, 20.0, 10.75, 9.50]

1. Print all numbers (without using `range` / `len`) **google**

- if form like: **"parking cost 1: 11.25"**

Hint: You may use `enumberate`

3.7 Generators

- Lazy evaluation,
- Memory effective

```
import collections, types

print(issubclass(types.GeneratorType, collections.Iterator))
```

```
True
```

Note: Generator is **Iteratorem**, but **Iterator** is not **Generator** !

3.7.1 Expression as generator

```
g = (n for n in range(20) if not n % 2) # just even !

for x in g:
    print(x)
```

3.7.2 Function as generator

```
def simple_gen():
    yield 5
    yield 10
    yield 15

s = simple_gen()

print(next(s))
print(next(s))
print(next(s))
```

```
5
10
15
```

```
# There are no elements to iterate over
print(next(s))

StopIteration:
```

```
def new_range(start=0, stop=None, step=1):
    i = start

    if stop is None:
        while True:
            yield i
            i += step
    else:
        while i < stop:
            yield i
            i += step

g = new_range(2, 5)
```

(continues on next page)

(continued from previous page)

```
print(next(g))
print(next(g))
```

```
2
3
```

Note: Generators are functions generating next values. When iterator then we should have `next()` method.

3.7.3 Exercises part 1

- Create generator, which is generating values which are **3 times greater than values from 0 to 20** ex. 0, 3, 6, 9, 12 . . .

3.7.4 Exercises part 2

1. Use file from list comprehension exercise **list comprehension**
1. Get this file using python

Hint:

- You can use library: `urllib`
- You can use default function `open` and `readline`

1. Clean up the file,
2. Write generator converting date in text to date format,
3. Unfortunately, during logs creation we had our time set up badly. We need add **1 hour to the log hour**

Hint: In package `datetime` there is `timedelta`

CHAPTER

FOUR

STANDARD LIBRARY

4.1 Stdlib

4.1.1 Pickling

Process of serialization of binary data into the file

4.1.2 Dump

```
hours = ['Tue Mar 29 23:40:17 +0000 2016', 'Tue Mar 29 23:40:19 +0000 2016']

file_store = open('/Users/kamil/daty.pickle', 'wb') # write/binary
pickle.dump(hours, file_Store)
```

4.1.3 Load

Hint: Method `load` instead dump

Warning: In case **Linuxa** there are diffrent paths format than in case of **Windows**.

4.1.4 Exercise

1. Download following file
1. Process dates (strings) into `datetime` type

Hint: In order to setup date format look on docs *docs*

https://docs.python.org/3/library/datetime.html

1. Save pickle after processing
2. Read pickle into different variable - check

4.2 Files operations

4.2.1 Files reading

```
file = 'file.txt'
```

4.2.2 Reading line by line

```
with open(r'../plik.txt') as file_descriptor:
    lines = file_descriptor.readlines()
```

4.2.3 Saving

```
with open(r'/tmp/iris.csv', mode='w') as file_descriptor:
    file_descriptor.write('hello')
```

4.2.4 Context manager

```
with open(r'plik.txt') as file_descriptor:
     for linia in file_descriptor:
         print(linia)
```

4.3 Regular expressions

4.3.1 Import

```
>>> import re
```

4.3.2 Usage

- Text processing,
  - Finding patterns,
  - Data cleaning
- Data validation

4.3.3 Functions withing re package

Function	meaning and usage	Result
re.match	If match `re.match(r"(\d+)\.(\d+)", "24.1632")`	True/False
re.search	First occurrence `re.search('(?<=abc)def', 'abcdef')`	
re.split	Splitting by separator `re.split(r'\W+', 'a, b, c')`	List
re.findall	Find all occurrences `re.findall('\w+', "A B")`	List
re.finditer	Find all occurrences `re.finditer('\w+', "A B")`	Iterator

4.3.4 Characters classes

Class	Meaning	
.	Any character	
^	Beginning of the line	
$	End of line	
*	Zero or more occurrences	
+	One or more occurrences	
?	One or zero occurrences	
{n}	N of occurrences	
{n, m}	Number of occurrences in range `n` to `m`	
d	Number group - same as [0-9]	
D	Anti number group [^0-9]	
w	Group “characters” - same as **[a-zA-Z0-9_]**	
W	Anti group “characters” - same as **[^a-zA-Z0-9_]**	
s	Group of white characters - same as [\r\n\t\f\v]	
[abc]	Group of characters a, b or c	
[a-z]	Characters in range `a` to `z`	
()	Group	

4.3.5 Exercise - part 1

- Create function `check_ip`
  - Function will be checking if IP is correct,
  - Check function on dictionary of hosts

```
 {
     '127.0.0.1': {'correct': None},
     '8.8.8.8': {'correct': None},
     'x.x.x.x': {'correct': None}
 }

* In place of **x.x.x.x** put any address from your network,
* Amend **correct** flag
```

Hint: You may use following expression, or find / create more precisse `^(?:[0-9]{1,3}\.){3}[0-9]{1,3}$`

4.3.6 Exercise - part 2

- Create function `check_email`
  - Function will be checking if email is correct

4.3.7 Exercise - part 3

- Using library `requests`
  - Download content of the page
- Get all **html** tags,
- Get human readable words

4.3.8 Exercise - part 4

- Using library `collections`
  - Get number of occurrences of word from Ex. part 3 (second point),
  - Get top 10 of most frequent words ?,
  - Get top 70 of most frequent words ?,

4.3.9 Resources

- Regex Expressions 101 - regex101
- Regular Expressions Cookbook by Steven Levithan, Jan Goyvaerts - book
- Email regex
- Stack overflow disscussion
- PyRegex

CHAPTER

FIVE

OBJECT PROGRAMMING

5.1 Object oriented programming

5.1.1 Class creation

```
class Human:
    pass

adam = Human()
```

5.1.2 Constructor

- is explaining what values should be assigned during creation of an instance

```
class Human:
    def __init__(self, name):
        self.name = name

eve = Human('Eve')
print(eve.name)
```

```
Eve
```

5.1.3 Self

- `self` is like `this` in other languages like java/c#,
- Its pointing to our instance/object,
- It could be named different but `self` is convention

5.1.4 Instance vs class

- class Human is a class (just concept / definition),
- `adam = Human('Adam')` is creation of an object/instance,
- adam is an object (concrete - creation of concept)

```
class Human:

    def __init__(self, name):
        self.name = name

adam = Human('Adam')
print(adam.name)
```

```
Adam
```

5.1.5 Class variables

- variables which stay the same across different objects

```
class Human:
    species = 'homo-sapiens'

    def __init__(self, name):
        self.name = name

print(Human.species)
adam = Human('Adam')
print(adam.name)
print(adam.species)
```

```
homo-sapiens
Adam
homo-sapiens
```

5.1.6 Special methods

Method	Parameters	Operator	Meaning
add	(self, other)	+	Adding objects
sub	(self, other)	•	Subtracting objects
len	(self)	len	Getting length of an object
contains	(self, other)	in	Check if in
str	(self)	str	Convert object to str
repr	(self)	repr	Get representation of object

5.1.7 Composition

5.1.8 Aggregation

5.1.9 Composition vs Aggregation

Composition	Composition	Aggregation
Creation	Inside	Outside
Deletion	With main ob	Independent

5.1.10 Example

```
import random

class Car:
    colors = ['red', 'blue', 'black']

    def __init__(self, brand='', color=None):
        self.brand = brand

        if not color:
            self.color = random.choice(self.colors)

    def __repr__(self):
        return "<{class_name} of brand: {brand} and color: {color}>".format(
            class_name=self.__class__.__name__,
            brand=self.brand,
            color=self.color
        )

    def __str__(self):
        return "{color} {brand} car".format(color=self.color, brand=self.brand)

bmw = Car('Bmw')
repr(bmw)
str(bmw)
```

5.1.11 Checking types

Note: check this code here

5.1.12 Exercises - part 1

1. Create class **mechanical_vehicle**, which is inheriting after **vehicles**,
2. When we create **mechanical vehicle** we need to know its unique id - its called **VIN** number,
3. Add fields:

- Fuel consumption per 100 km

- Add properties (property decorator) - miles left

1. Add method `go(how_far)` - this should change `fuel_amount` state and `milage` state,

5.1.13 Exercises - part 2

1. Create class **Server** which got:

- Name,
- Ip,
- Create `ping` method (use `os` – execute ping command),
- Change **representation** and **conversion to string** methods,
- Store history of ping - date and status,
- Create list of hosts for **pinging** ['127.0.0.1', ...],
- Iterate over the list and print message for hosts if they are **pingable**

Hint:

- Use `os` library

Hint: Library __pathlib__ (std). Class PurePath:

Hint: Library ldap3:

5.1.14 Exercises - part 2

1. Create class **Cluster** which got:

- Location,
- Name

1. Its also to do `len` and add + on `Cluster` object

CHAPTER

SIX

NETWORK

6.1 Paramiko

6.1.1 Installation

```
pip install paramiko
```

6.1.2 Connection

```
adress = 'ec2-54-93-218-119.eu-central-1.compute.amazonaws.com' # adjust your address
username = 'username'
password = 'password'

client = paramiko.SSHClient()
client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
client.connect(adress, username= username, password=password) # we need to close␣
↪connection ourself
```

Hint: You can check if connection is still open using: `lsof -i@ec2-54-93-218-119.eu-central-1.compute.amazonaws.com`

6.1.3 Using sftp

```
sftp = client.open_sftp()

sftp.listdir('/var/log')
sftp.get('/var/log/access_log' ,'access_log')
```

6.2 Smtplib

6.2.1 Example code

```
import smtplib

gmail_user = 'user@gmail.com'
gmail_password = 'password'

sent_from = gmail_user

to = ['me@gmail.com', 'pazik.kamil@gmail.com']
subject = 'Message subject'
body = "Hey, what's up?\n\n- You"

email_text = """\
From: {}
To: {}
Subject: {}

{}
""".format(sent_from, ", ".join(to), subject, body)

try:
    server = smtplib.SMTP_SSL('smtp.gmail.com', 465)
    server.ehlo()
    server.login(gmail_user, gmail_password)
    server.sendmail(sent_from, to, email_text)
    server.close()

    print('Email sent!')
except:
    print('Something is wrong')
```

6.2.2 Yagmail

```
import yagmail

yag = yagmail.SMTP('user@gmail.com', 'password')

contents = [
    "This is the body, and here is just text http://somedomain/image.png",
    "You can find an audio file attached.", '/Users/kamil/code/cisco_python/apache_
↪logs.txt'
]
yag.send('pazik.kamil@gmail.com', 'subject', contents)
```

CHAPTER

SEVEN

PANDAS

7.1 Pandas

[1]:
```
!ls # magic command
```

```
Python basics - day 2.ipynb http.log
Python basics.ipynb         requirements.txt
Untitled.ipynb              venv
Untitled1.ipynb
```

7.1.1 Installing packages in jupyter-notebook

[53]:
```
!pip install pandas
!pip install matplotlib
```

```
Requirement already satisfied: pandas in ./venv/lib/python3.6/site-packages (0.25.1)
Requirement already satisfied: numpy>=1.13.3 in ./venv/lib/python3.6/site-packages␣
↪(from pandas) (1.17.2)
Requirement already satisfied: python-dateutil>=2.6.1 in ./venv/lib/python3.6/
↪site-packages (from pandas) (2.8.0)
Requirement already satisfied: pytz>=2017.2 in ./venv/lib/python3.6/site-packages␣
↪(from pandas) (2019.2)
Requirement already satisfied: six>=1.5 in ./venv/lib/python3.6/site-packages (from␣
↪python-dateutil>=2.6.1->pandas) (1.12.0)
You are using pip version 18.1, however version 19.2.3 is available.
You should consider upgrading via the 'pip install -upgrade pip' command.
Collecting matplotlib
  Downloading https://files.pythonhosted.org/packages/cf/a4/
↪d5387a74204542a60ad1baa84cd2d3353c330e59be8cf2d47c0b11d3cde8/matplotlib-3.1.
↪1-cp36-cp36m-macosx_10_6_intel.macosx_10_9_intel.macosx_10_9_x86_64.
↪macosx_10_10_intel.macosx_10_10_x86_64.whl (14.4MB)
    100% || 14.4MB 1.0MB/s ta 0:00:011   37% |                    | 5.4MB 1.8MB/s eta␣
↪0:00:05
Requirement already satisfied: numpy>=1.11 in ./venv/lib/python3.6/site-packages␣
↪(from matplotlib) (1.17.2)
Collecting cycler>=0.10 (from matplotlib)
  Using cached https://files.pythonhosted.org/packages/f7/d2/
↪e07d3ebb2bd7af696440ce7e754c59dd546ffe1bbe732c8ab68b9c834e61/cycler-0.10.0-py2.
↪py3-none-any.whl
Requirement already satisfied: python-dateutil>=2.1 in ./venv/lib/python3.6/
↪site-packages (from matplotlib) (2.8.0)
Collecting kiwisolver>=1.0.1 (from matplotlib)
```

(continues on next page)

(continued from previous page)

```
  Downloading https://files.pythonhosted.org/packages/49/5d/
↪d1726d2a2fd471a69ef5014ca42812e1ccb8a13085c42bfcb238a5611f39/kiwisolver-1.1.
↪0-cp36-cp36m-macosx_10_6_intel.macosx_10_9_intel.macosx_10_9_x86_64.
↪macosx_10_10_intel.macosx_10_10_x86_64.whl (113kB)
    100% || 122kB 2.1MB/s ta 0:00:01
Collecting pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 (from matplotlib)
  Using cached https://files.pythonhosted.org/packages/11/fa/
↪0160cd525c62d7abd076a070ff02b2b94de589f1a9789774f17d7c54058e/pyparsing-2.4.2-py2.
↪py3-none-any.whl
Requirement already satisfied: six in ./venv/lib/python3.6/site-packages (from␣
↪cycler>=0.10->matplotlib) (1.12.0)
Requirement already satisfied: setuptools in ./venv/lib/python3.6/site-packages (from␣
↪kiwisolver>=1.0.1->matplotlib) (40.6.2)
Installing collected packages: cycler, kiwisolver, pyparsing, matplotlib
Successfully installed cycler-0.10.0 kiwisolver-1.1.0 matplotlib-3.1.1 pyparsing-2.4.2
You are using pip version 18.1, however version 19.2.3 is available.
You should consider upgrading via the 'pip install -upgrade pip' command.
```

7.1.2 Importing pandas and matplotlib

[2]:
```
import pandas as pd
from matplotlib import pyplot as plt
```

7.1.3 File we will be working on

cleaned_access_log

7.1.4 Downloading using request

[31]:
```
!pip install requests
```

```
Collecting requests
  Using cached https://files.pythonhosted.org/packages/51/bd/
↪23c926cd341ea6b7dd0b2a00aba99ae0f828be89d72b2190f27c11d4b7fb/requests-2.22.0-py2.
↪py3-none-any.whl
Collecting chardet<3.1.0,>=3.0.2 (from requests)
  Using cached https://files.pythonhosted.org/packages/bc/a9/
↪01ffebfb562e4274b6487b4bb1ddec7ca55ec7510b22e4c51f14098443b8/chardet-3.0.4-py2.
↪py3-none-any.whl
Collecting urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 (from requests)
  Using cached https://files.pythonhosted.org/packages/e6/60/
↪247f23a7121ae632d62811ba7f273d0e58972d75e58a94d329d51550a47d/urllib3-1.25.3-py2.
↪py3-none-any.whl
Collecting idna<2.9,>=2.5 (from requests)
  Using cached https://files.pythonhosted.org/packages/14/2c/
↪cd551d81dbe15200be1cf41cd03869a46fe7226e7450af7a6545bfc474c9/idna-2.8-py2.
↪py3-none-any.whl
Collecting certifi>=2017.4.17 (from requests)
  Using cached https://files.pythonhosted.org/packages/18/b0/
↪8146a4f8dd402f60744fa380bc73ca47303cccf8b9190fd16a827281eac2/certifi-2019.9.11-py2.
↪py3-none-any.whl
Installing collected packages: chardet, urllib3, idna, certifi, requests
```

(continues on next page)

(continued from previous page)

```
Successfully installed certifi-2019.9.11 chardet-3.0.4 idna-2.8 requests-2.22.0␣
↪urllib3-1.25.3
You are using pip version 18.1, however version 19.2.3 is available.
You should consider upgrading via the 'pip install -upgrade pip' command.
```

[33]:
```
import requests

url = 'https://python.variantcore.com/cleaned_access_log'
r = requests.get(url, allow_redirects=True)
open('access_log', 'wb').write(r.content)
```

[33]:
```
2468755
```

[34]:
```
!ls
```

```
Python basics - day 2.ipynb http.log
Python basics.ipynb         pandas.ipynb
Untitled.ipynb              requirements.txt
access_log                  venv
apache_logs.txt             xx.log
cleaned_access_log
```

7.1.5 Reading text file

[7]:
```
data = pd.read_csv('cleaned_access_log')
```

[10]:
```
data.head(5)
```

[10]:

	ip	time	request	status	size	referer	user_agent
0	83.149.9.216	[17/May/2015:10:05:03 +0000]	"GET /presentations/logstash-monitorama-2013/i...	200	203023.0	"http://semicomplete.com/presentations/logstas...	"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_1...
1	83.149.9.216	[17/May/2015:10:05:43 +0000]	"GET /presentations/logstash-monitorama-2013/i...	200	171717.0	"http://semicomplete.com/presentations/logstas...	"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_1...
2	83.149.9.216	[17/May/2015:10:05:47 +0000]	"GET /presentations/logstash-monitorama-2013/p...	200	26185.0	"http://semicomplete.com/presentations/logstas...	"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_1...
3	83.149.9.216	[17/May/2015:10:05:12 +0000]	"GET /presentations/logstash-monitorama-2013/p...	200	7697.0	"http://semicomplete.com/presentations/logstas...	"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_1...
4	83.149.9.216	[17/May/2015:10:05:07 +0000]	"GET /presentations/logstash-monitorama-2013/p...	200	2892.0	"http://semicomplete.com/presentations/logstas...	"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_1...

7.1.6 Simple EDA

```
[8]: data.columns
```

```
[8]: Index(['ip', 'time', 'request', 'status', 'size', 'referer', 'user_agent'], dtype=
     ↪'object')
```

```
[57]: data['time'].describe()
```

```
[57]: count                           10000
      unique                           4363
      top       [19/May/2015:00:05:25 +0000]
      freq                                9
      Name: time, dtype: object
```

7.1.7 Clean up

```
[11]: from datetime import datetime
```

```
[67]: datetime.strptime('[20/May/2015:21:05:28 +0000]', '[%d/%b/%Y:%H:%M:%S %z]')
```

```
[67]: datetime.datetime(2015, 5, 20, 21, 5, 28, tzinfo=datetime.timezone.utc)
```

```
[15]: data['time'].apply(lambda x: datetime.strptime(x, '[%d/%b/%Y:%H:%M:%S %z]'))
```

```
---------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-15-f20f25333a1e> in <module>
---> 1 data['time'].apply(lambda x: datetime.strptime(x, '[%d/%b/%Y:%H:%M:%S %z]'))

~/code/cisco_python/venv/lib/python3.6/site-packages/pandas/core/series.py in␣
↪apply(self, func, convert_dtype, args, **kwds)
   4040             else:
   4041                 values = self.astype(object).values
-> 4042                 mapped = lib.map_infer(values, f, convert=convert_dtype)
   4043
   4044         if len(mapped) and isinstance(mapped[0], Series):

pandas/_libs/lib.pyx in pandas._libs.lib.map_infer()

<ipython-input-15-f20f25333a1e> in <lambda>(x)
---> 1 data['time'].apply(lambda x: datetime.strptime(x, '[%d/%b/%Y:%H:%M:%S %z]'))

~/.pyenv/versions/3.6.9/lib/python3.6/_strptime.py in _strptime_datetime(cls,␣
↪data_string, format)
    563     """Return a class cls instance based on the input string and the
    564     format string."""
-> 565     tt, fraction = _strptime(data_string, format)
    566     tzname, gmtoff = tt[-2:]
    567     args = tt[:6] + (fraction,)

~/.pyenv/versions/3.6.9/lib/python3.6/_strptime.py in _strptime(data_string, format)
    360     if not found:
    361         raise ValueError("time data %r does not match format %r" %
-> 362                          (data_string, format))
    363     if len(data_string) != found.end():
```

(continues on next page)

(continued from previous page)

```
    364         raise ValueError("unconverted data remains: %s" %

ValueError: time data '(compatible;' does not match format '[%d/%b/%Y:%H:%M:%S %z]'
```

[17]:
```
data['time'][0:8899]
```

[17]:
```
0       [17/May/2015:10:05:03 +0000]
1       [17/May/2015:10:05:43 +0000]
2       [17/May/2015:10:05:47 +0000]
3       [17/May/2015:10:05:12 +0000]
4       [17/May/2015:10:05:07 +0000]
                    ...
8894    [20/May/2015:12:05:35 +0000]
8895    [20/May/2015:12:05:34 +0000]
8896    [20/May/2015:12:05:26 +0000]
8897    [20/May/2015:12:05:48 +0000]
8898                     (compatible;
Name: time, Length: 8899, dtype: object
```

[12]:
```
data['time'][8899:].apply(lambda x: datetime.strptime(x, '[%d/%b/%Y:%H:%M:%S %z]'))
```

[12]:
```
8899   2015-05-20 12:05:25+00:00
8900   2015-05-20 12:05:59+00:00
8901   2015-05-20 12:05:16+00:00
8902   2015-05-20 12:05:54+00:00
8903   2015-05-20 12:05:39+00:00
                  ...
9995   2015-05-20 21:05:28+00:00
9996   2015-05-20 21:05:50+00:00
9997   2015-05-20 21:05:00+00:00
9998   2015-05-20 21:05:56+00:00
9999   2015-05-20 21:05:15+00:00
Name: time, Length: 1101, dtype: datetime64[ns, UTC]
```

[19]:
```
data['time'][8898]
```

[19]:
```
'(compatible;'
```

[24]:
```
data.drop([8898], inplace=True)
```

Check if deleted

[33]:
```
8898 in data.index
```

[33]:
```
True
```

[132]:
```
data['time'][8898]
```

```
---------------------------------------------------
KeyError                                  Traceback (most recent call last)
~/code/cisco_python/venv/lib/python3.6/site-packages/pandas/core/indexes/base.py in
↪get_loc(self, key, method, tolerance)
   2896             try:
-> 2897                 return self._engine.get_loc(key)
   2898             except KeyError:
```

(continues on next page)

(continued from previous page)

```
pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.
↪get_item()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.
↪get_item()

KeyError: 'time'

During handling of the above exception, another exception occurred:

KeyError                                  Traceback (most recent call last)
<ipython-input-132-9dcbd30e4101> in <module>
---> 1 data['time'][8898]

~/code/cisco_python/venv/lib/python3.6/site-packages/pandas/core/frame.py in␣
↪__getitem__(self, key)
   2978             if self.columns.nlevels > 1:
   2979                 return self._getitem_multilevel(key)
-> 2980             indexer = self.columns.get_loc(key)
   2981             if is_integer(indexer):
   2982                 indexer = [indexer]

~/code/cisco_python/venv/lib/python3.6/site-packages/pandas/core/indexes/base.py in␣
↪get_loc(self, key, method, tolerance)
   2897                 return self._engine.get_loc(key)
   2898             except KeyError:
-> 2899                 return self._engine.get_loc(self._maybe_cast_indexer(key))
   2900         indexer = self.get_indexer([key], method=method, tolerance=tolerance)
   2901         if indexer.ndim > 1 or indexer.size > 1:

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.
↪get_item()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.
↪get_item()

KeyError: 'time'
```

[38]:
```
data['time'] = data['time'].apply(lambda x: datetime.strptime(x, '[%d/%b/%Y:%H:%M:%S
↪%z]'))
```

[136]:
```
data.index = data.index.tz_localize(None)
```

[137]:
```
data.index
```

[137]:
```
DatetimeIndex(['2015-05-17 10:05:03', '2015-05-17 10:05:43',
               '2015-05-17 10:05:47', '2015-05-17 10:05:12',
```

(continues on next page)

(continued from previous page)

```
               '2015-05-17 10:05:07', '2015-05-17 10:05:34',
               '2015-05-17 10:05:57', '2015-05-17 10:05:50',
               '2015-05-17 10:05:24', '2015-05-17 10:05:50',
               ...
               '2015-05-20 21:05:11', '2015-05-20 21:05:29',
               '2015-05-20 21:05:34', '2015-05-20 21:05:15',
               '2015-05-20 21:05:01', '2015-05-20 21:05:28',
               '2015-05-20 21:05:50', '2015-05-20 21:05:00',
               '2015-05-20 21:05:56', '2015-05-20 21:05:15'],
              dtype='datetime64[ns]', name='time', length=9999, freq=None)
```

[39]:
```
data['time'][0]
```

[39]:
```
Timestamp('2015-05-17 10:05:03+0000', tz='UTC')
```

[49]:
```
data['user_agent'].apply(str.upper)
```

[49]:
```
0       "MOZILLA/5.0 (MACINTOSH; INTEL MAC OS X 10_9_1...
1       "MOZILLA/5.0 (MACINTOSH; INTEL MAC OS X 10_9_1...
2       "MOZILLA/5.0 (MACINTOSH; INTEL MAC OS X 10_9_1...
3       "MOZILLA/5.0 (MACINTOSH; INTEL MAC OS X 10_9_1...
4       "MOZILLA/5.0 (MACINTOSH; INTEL MAC OS X 10_9_1...
                              ...
9995            "TINY TINY RSS/1.11 (HTTP://TT-RSS.ORG/)"
9996            "TINY TINY RSS/1.11 (HTTP://TT-RSS.ORG/)"
9997    "MOZILLA/5.0 (COMPATIBLE; GOOGLEBOT/2.1; +HTTP...
9998    "MOZILLA/5.0 (WINDOWS NT 5.1; RV:6.0.2) GECKO/...
9999    "UNIVERSALFEEDPARSER/4.2-PRE-314-SVN +HTTP://F...
Name: user_agent, Length: 9999, dtype: object
```

7.1.8 Simple EDA

[51]:
```
import re
```

[47]:
```
data['user_agent'][911]
```

[47]:
```
'"Mozilla/5.0 (Windows NT 6.1; WOW64; rv:27.0) Gecko/20100101 Firefox/27.0"'
```

[59]:
```
data['status'] = data.status.astype(int)
```

[60]:
```
data.status
```

[60]:
```
0       200
1       200
2       200
3       200
4       200
       ...
9995    200
9996    200
9997    200
9998    200
9999    200
Name: status, Length: 9999, dtype: int64
```

[55]:
```
data[data.user_agent.str.contains('Linux', regex= True, na=False, flags=re.
↪IGNORECASE)]
```

[55]:
```
                  ip                        time  \
23     24.236.252.67 2015-05-17 10:05:40+00:00
24      93.114.45.13 2015-05-17 10:05:14+00:00
25      93.114.45.13 2015-05-17 10:05:04+00:00
26      93.114.45.13 2015-05-17 10:05:45+00:00
27      93.114.45.13 2015-05-17 10:05:14+00:00
...              ...                       ...
9949  91.151.182.109 2015-05-20 21:05:13+00:00
9950  91.151.182.109 2015-05-20 21:05:50+00:00
9953    63.140.98.80 2015-05-20 21:05:27+00:00
9954    63.140.98.80 2015-05-20 21:05:58+00:00
9955    63.140.98.80 2015-05-20 21:05:11+00:00

                                              request status     size  \
23                          "GET /favicon.ico HTTP/1.1"    200   3638.0
24      "GET /articles/dynamic-dns-with-dhcp/ HTTP/1.1"    200  18848.0
25                            "GET /reset.css HTTP/1.1"    200   1015.0
26                           "GET /style2.css HTTP/1.1"    200   4877.0
27                          "GET /favicon.ico HTTP/1.1"    200   3638.0
...                                                 ...    ...      ...
9949         "GET /images/web/2009/banner.png HTTP/1.1"    200  52315.0
9950                        "GET /favicon.ico HTTP/1.1"    200   3638.0
9953                  "GET /projects/xdotool/ HTTP/1.1"    200  12292.0
9954                "GET /images/jordan-80.png HTTP/1.1"    200   6146.0
9955  "GET /files/logstash/logstash-1.3.2-monolithic...    404    324.0

                                                referer  \
23                                                  "-"
24    "http://www.google.ro/url?sa=t&rct=j&q=&esrc=s...
25    "http://www.semicomplete.com/articles/dynamic-...
26    "http://www.semicomplete.com/articles/dynamic-...
27                                                  "-"
...                                                 ...
9949    "http://www.semicomplete.com/projects/xdotool/"
9950                                                "-"
9953  "http://stackoverflow.com/questions/3983946/ge...
9954    "http://www.semicomplete.com/projects/xdotool/"
9955                                                "-"

                                             user_agent
23    "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:26...
24    "Mozilla/5.0 (X11; Linux x86_64; rv:25.0) Geck...
25    "Mozilla/5.0 (X11; Linux x86_64; rv:25.0) Geck...
26    "Mozilla/5.0 (X11; Linux x86_64; rv:25.0) Geck...
27    "Mozilla/5.0 (X11; Linux x86_64; rv:25.0) Geck...
...                                                 ...
9949  "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/5...
9950  "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/5...
9953  "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/5...
9954  "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/5...
9955  "Chef Client/10.18.2 (ruby-1.9.3-p327; ohai-6....

[2314 rows x 7 columns]
```

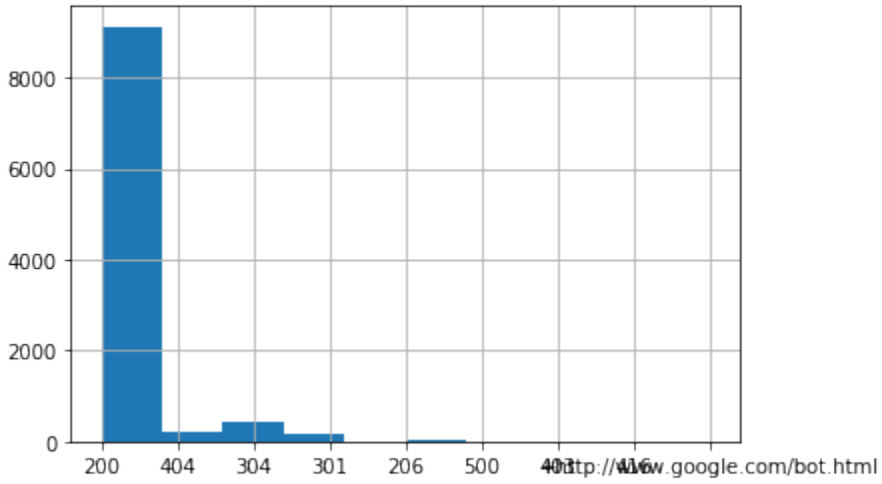
[47]:
```
data[['ip', 'status', 'user_agent']].describe()
```

[47]:
```
                    ip status  \
count            10000  10000
unique            1754      9
top     66.249.73.135    200
freq               482   9125

                                                user_agent
count                                                 9999
unique                                                 558
top     "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebK...
freq                                                  1044
```

[56]:
```
data['status'].hist()
```

[56]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x120f6a5f8>
```

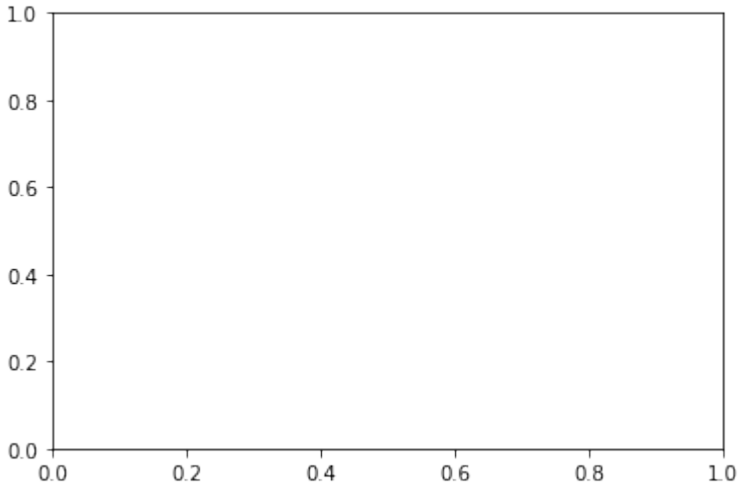


[66]:
```
data.set_index('time', inplace=True)
```

[75]:
```
min(data.index)
```

[75]:
```
Timestamp('2015-05-17 10:05:00+0000', tz='UTC')
```

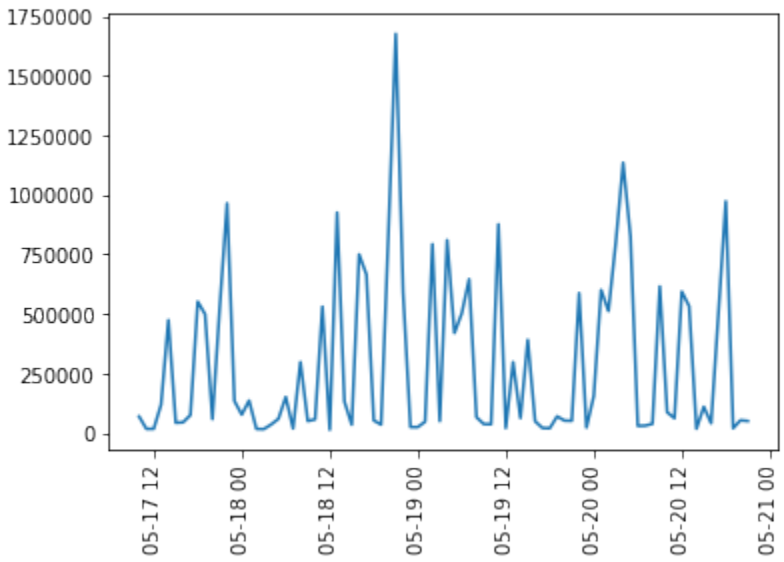
[103]:
```
fig, ax = plt.subplots( nrows=1, ncols=1 )
```



[107]:
```
xa = data['size'].resample('H').mean()
```

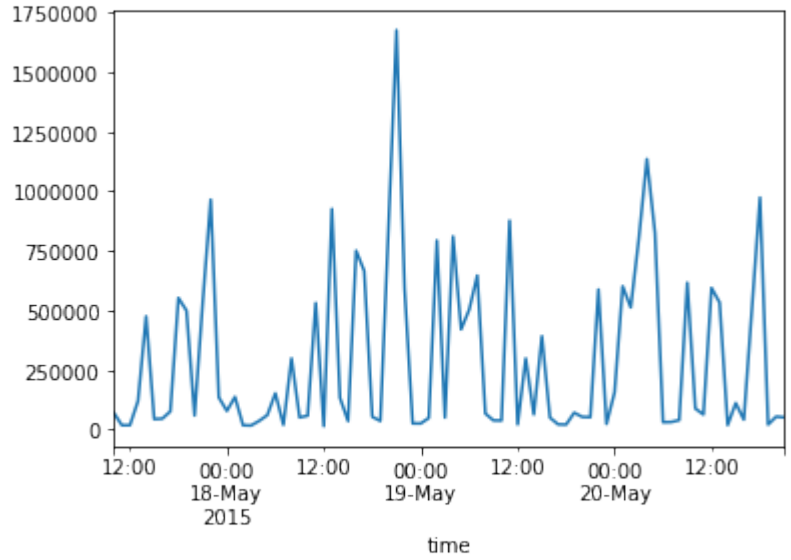
[90]:
```
from matplotlib import pyplot as plt
```

[110]:
```
plt.xticks(rotation=90)
plt.plot(xa)
plt.savefig('myfig')
```



[114]:
```
xa = data['size'].resample('H').mean().plot()
```

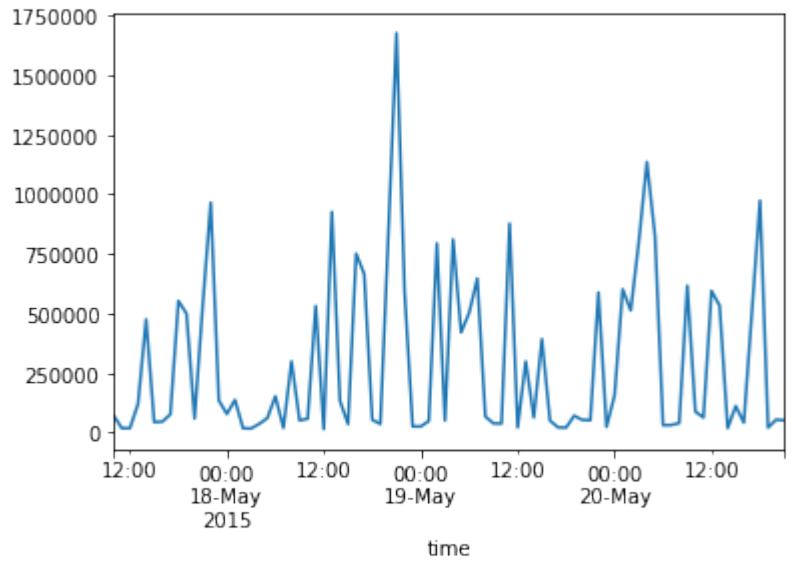
```
/Users/kamil/code/cisco_python/venv/lib/python3.6/site-packages/pandas/core/arrays/
↪datetimes.py:1269: UserWarning: Converting to PeriodArray/Index representation will␣
↪drop timezone information.
  UserWarning,
```



[112]:
```
xa.plot()
```

```
/Users/kamil/code/cisco_python/venv/lib/python3.6/site-packages/pandas/core/arrays/
↪datetimes.py:1269: UserWarning: Converting to PeriodArray/Index representation will␣
↪drop timezone information.
  UserWarning,
```

[112]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x122415320>
```



[ ]:
```
data[data.user_agent.str.contains('Linux', regex= True, na=False, flags=re.
↪IGNORECASE)]
```

[128]:
```
data.to_excel?
```

[129]:
```
!pip install openpyxl
```

```
Collecting openpyxl
  Downloading https://files.pythonhosted.org/packages/f5/39/
↪942a406621c1ff0de38d7e4782991b1bac046415bf54a66655c959ee66e8/openpyxl-2.6.3.tar.gz␣
↪(173kB)
    100% || 174kB 1.8MB/s ta 0:00:01
Collecting jdcal (from openpyxl)
  Downloading https://files.pythonhosted.org/packages/f0/da/
↪572cbc0bc582390480bbd7c4e93d14dc46079778ed915b505dc494b37c57/jdcal-1.4.1-py2.
↪py3-none-any.whl
Collecting et_xmlfile (from openpyxl)
  Downloading https://files.pythonhosted.org/packages/22/28/
↪a99c42aea746e18382ad9fb36f64c1c1f04216f41797f2f0fa567da11388/et_xmlfile-1.0.1.tar.gz
Installing collected packages: jdcal, et-xmlfile, openpyxl
  Running setup.py install for et-xmlfile ... done
  Running setup.py install for openpyxl ... done
Successfully installed et-xmlfile-1.0.1 jdcal-1.4.1 openpyxl-2.6.3
You are using pip version 18.1, however version 19.2.3 is available.
You should consider upgrading via the 'pip install -upgrade pip' command.
```

[138]:
```
data.to_excel('report.xlsx')
```

[139]:
```
!ls
```

```
cleaned_access_log myfig.png          report.xlsx
foo.png            pandas.ipynb
```

7.1.9 Exercise - report creation

- Report should say about different web browsers user are using,

CHAPTER

EIGHT

DJANGO

8.1 Docker

8.1.1 What for

- local development,
- deployment

8.1.2 How it works

- cgroups,
- native on Linux,
- non native on Mac(Hyperkit), Windows

8.1.3 Definition

```
FROM python:3.6

MANTAINER kamil pazik

ENV API_HOME /opt/api

RUN apt-get update
RUN apt-get install -y vim

RUN pip install --upgrade pip

WORKDIR $API_HOME
```

8.1.4 Exercise - part 1

- create `Dockerfile` and install there a **django** automatically,
- build the docker image, and tag it as `django_alpha_image`,
- print images - how many of them you got in your system,
- run the container,

- log into the container and try to `ping` some server

8.2 docker-compose

8.2.1 Why

- To manage multiple parts of system
  - Databases,
  - Web servers,
  - Other servers

8.2.2 Definition

```
version: '3'
services:
  web:
    build: .
    ports:
    - "5000:5000"
    volumes:
    - .:/opt/api
```

8.2.3 Exercise 1

- create `docker-compose.yml` in version 3.5,
- define web service - you should have port 8000 published,
- name container - whatever name you like,
- launch `docker-compose up`,
- Define `entrypoint.sh`,
- make `docker-compose up`

8.3 Web api - Django

8.3.1 Creation of virtual env

```
python -m venv venv
source venv/bin/activate
```

8.3.2 Instalation

```
pip install django
pip install djangorestframework # remember to add 'rest_framework', to settings.py -␣
↪INSTALLED_APPS
pip install django-extensions # remember to add 'django_extensions', to settings.py -␣
↪INSTALLED_APPS
pip install markdown       # Markdown support for the browsable API.
pip install django-filter  # Filtering support - remember to add to settings.py
↪'django_filters' - INSTALLED_APPS
```

8.3.3 Creation of dependencies

```
pip freeze > requirements.txt
```

8.3.4 Creation of new project

```
django-admin startproject servermonitoring
cd servermonitoring
python manage.py migrate
python manage.py runserver
```

8.3.5 Create of new app

```
django-admin startapp api
```

8.3.6 Check in browser

- Chrome/Postman address `http://127.0.0.1:8000/`
- Curl `http://127.0.0.1:8000/`

8.3.7 Adjust settings

Tip: look on `manage.py` ex. using command: `cat manage.py` there you got way how django is launched change setup in `settings.py`

```
vim servermonitoring/settings.py
```

Hint: you can do that using vim `vim servermonitoring/settings.py`

Or directly in **pyCharm**

```
# servermonitoring/settings.py

INSTALLED_APPS = [
 'django.contrib.admin',
 'django.contrib.auth',
 'django.contrib.contenttypes',
 'django.contrib.sessions',
 'django.contrib.messages',
 'django.contrib.staticfiles',
 'django_filters',
 'django_extensions',
 'rest_framework',
 'api'
]
```

8.3.8 Tests creation

```
# api/tests.py

from django.test import TestCase
from rest_framework import status

from api.models import Server

class ServerModelTestCase(TestCase):
    """Server model tests"""

    def setUp(self):
        """Definition of startup values"""

        self.server = Server(address='127.0.0.1')

    def test_model_repr(self):
        self.assertEqual("<Server address: 127.0.0.1>", repr(self.server))

    def test_model_str(self):
        self.assertEqual("Server o adresie: 127.0.0.1", str(self.server))
```

8.3.9 Test execution

```
python manage.py test
```

8.3.10 Model creation

```
from django.db import models

class Server(models.Model):
    created = models.DateTimeField(auto_now_add=True)
    location = models.CharField(max_length=100, null=True, blank=True, default='')
    available = models.BooleanField(default=False)
    address = models.GenericIPAddressField()
```

(continues on next page)

(continued from previous page)

```
    admin_contact = models.EmailField(max_length=70, null=True, blank=True)
    admin_phone = models.CharField(max_length=70, null=True, blank=True, default='')

    def __repr__(self):
        return "<{} address: {}>".format(self.__class__.__name__, self.address)

    def __str__(self):
        return "{} o adresie: {}".format(self.__class__.__name__, self.address)
```

8.3.11 Creation and execution of migations

```
python manage.py makemigrations
python manage.py migrate
```

8.3.12 Checking of sql migrations code

```
python manage.py sqlmigrate api 0001
```

8.3.13 Execution of test after migrations

```
python manage.py test
```

8.3.14 Addint view test

```
# api/tests.py
# ............

from django.test import TestCase
from rest_framework import status
from api.models import Server
from django.urls import reverse

class ViewServerTestCase(TestCase):
    """Test for server view"""

    def test_create_server(self):
        """We check if we can create server (post)"""

        url = reverse('servers')
        data = {"location": "office", "address": "127.0.0.1"}

        response = self.client.post(url, data, format="json")

        self.assertEqual(response.status_code, status.HTTP_201_CREATED)
        self.assertEqual(len(response.data), 1)

    def test_view_server_list(self):
        """We check if we get proper amount of servers"""
```

(continues on next page)

(continued from previous page)

```
        url = reverse('servers')

        response = self.client.get(url, format="json")
        self.assertEqual(response.status_code, status.HTTP_200_OK)
```

8.3.15 Serializer

```
# api/serializers.py

from rest_framework import serializers

from .models import Server

class ServerSerializer(serializers.ModelSerializer):
    class Meta:
        model = Server
        fields = '__all__' # albo fields = ('location', 'address',)
```

8.3.16 Adding view

```
# api/views.py

from rest_framework.views import APIView
from rest_framework.response import Response
from rest_framework import status

from .serializers import ServerSerializer
from .models import Server

class ServerList(APIView):
    """Lista serwerow"""

    serializer_class = ServerSerializer

    def get_queryset(self):
        queryset = Server.objects.all()
        location = self.request.query_params.get('location', None)

        if location is not None:
            queryset = queryset.filter(location__icontains=location)
        return queryset

    def get(self, request, format=None):

        servers = self.get_queryset()
        serializer = ServerSerializer(servers, many=True)
        return Response(serializer.data)

    def post(self, request, format=None):
        serializer = ServerSerializer(data=request.data)
```

(continues on next page)

(continued from previous page)

```
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data, status=status.HTTP_201_CREATED)
        return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
```

8.3.17 Add urls

```
from django.contrib import admin
from django.urls import path

from api import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('servers/', views.ServerList.as_view(), name='servers')
]
```

8.3.18 Methods

- Get
- Post
- Put
- Delete

8.3.19 HTTP codes

- `200` - success. Request correct. Response correct.
- `400` - fail **request**. Bad request / problems with authentication.
- `403` - access denied,
- `404` - no such page,
- `500` - internal error. Usually because of developer made mistake.

8.3.20 Requests

8.3.21 Responses

8.3.22 Settings

8.3.23 View

8.3.24 Migations

Hint: To see migrations we can execute `python manage.py showmigrations`

8.3.25 Orm

8.3.26 Django extensions

```
python manage.py show_urls
python manage.py shell_plus # better console/dev server- ipython
python manage.py runserver_plus # server
```

8.3.27 Practice

Zero

- Napraw test,
- Dodaj do testu sprawdzanie daty - stworzenia wpisu

First

- Create 4 hosts - each in different way
  - Using web page,
  - Using `request` from postman,
- Create model, which:
  - Would be storing `date`

Second

- Stwórz endpoint (POST), który:
  - Będzie

Third

CHAPTER

NINE

CONTACT

9.1 Contact

- Mail: kpazik@variantcore.com
- LinkedIn: https://www.linkedin.com/in/kamil-pazik

CHAPTER

TEN

INDICES AND TABLES

- genindex
- modindex
- search