Data and Design

Release 64e0ddc

Jacob Frias Koehler

Mar 09, 2018

Contents

1 Data and Design with Python

OVERVIEW

This short course aims to introduce participants to the Python computing language. We will investigate the use of Python to perform data analysis, access and structure information from the web, and build and deploy applications like web pages and message boards using Django. Students will be expected to complete a small project for each weeks topics described below.

1.1 Important Material Locations

- **Course Documentation**: http://data-and-design.readthedocs.io/en/latest/
- **Github Repository**: https://github.com/jfkoehler/data-design/tree/master/source
- **Slack Channel**: https://datadesignpython.slack.com/
- **YouTube Channel**: https://www.youtube.com/playlist?list=PLUCTTwyv9AdUYtNeV5w-2xMX5O9cCcfkB

1.2 Topics

- **Introduction to Data and Visualizations**: The first class will focus on using Pandas and Seaborn to explore data in `.csv` files and through API's. We emphasize the use of the computer to explore the data and look for patterns and differences. Our first project involves writing an analysis of New York City's 8th grade mathematics scores.
- *Introduction to Pandas and Seaborn*
- *Pandas and Seaborn*

- *Assignment: Access and Analyze Data*
- **Introduction to Web Scraping**: Today, we investigate the use of webscraping to pull and clean data from websites. We will investigate some basics of HTML and CSS, and use the `requests` and `BeautifulSoup2` libraries to pull this information.
- *Introduction to webscraping*
- *Scraping Part II*
- **Natural Language Processing and Scraping**: Today, we extend our webscraping work to analyze the text of documents scraped. We will use the Natural Language Toolkit to analyze text. We will also introduce the use of regular expressions in navigating text on the computer.
- *Webscraping and Natural Language Processing*
- *Sentiment Analysis of Text*
- *More Machine Learning*
- **Web Design with Django**: In this workshop, we will use the Django framework to design and deploy a basic web application. Our assignment will be a basic website ready to display our earlier work with Jupyter notebooks. We discuss Django projects and applications to use Python to build a basic website.
- *Basic WebSite with Django*
- *Applications with Django*
- **Data and our Website**: The final class serves to connect our earlier work with data and Python through Django models, where we build a database for our website. We will add a Blog application to our site, post some information, and access these posts as data in the shell. Finally, we use the ListView and DetailView to display these posts together with template logic.
- *Databases and Django: A Basic Blog*

1.3 Lessons Learned

- **Student Computers**: A number of students experienced difficulties with their computers at different points during the semester. In the first weeks, students who lacked access to their own functioning laptops dropped from enrollment. Also, a few students who were unaware of the level of coding involved dropped the course. If we were able to identify an IT support person who is capable of helping students install and optimize their personal computers, this would be great.

1.4 Technology Work

Also, if we were able to provide a web-based coding environment this could alleviate many of these issues. Below are three such options:

- **OpenEdX**: A Learning Management system built by MIT and Harvard as part of their open-course initiatives. This is freely available, however we would need a person competent in full stack web development. Alternatively, third party companies will launch and manage these applications for a fee that based on my initial research would be in the $10,000 neighborhood.
- **CoCalc**: A collaborative computing platform that has many language capability. We should be able to launch some version of this ourselves, using the Jupyter notebook and text editor execution capabilities of the service. This would again require some support from an individual who understands servers and deploying interactive software applications on them.

- **JupyterHub**: There have been examples of institutions that integrate Jupyter notebooks and other code related interfaces into their Learning Management Systems through JupyterHub. The most popular example is the Data8 course at UC Berkeley.
- http://data8.org/

  This class integrates the JupyterHub with a virtual textbook. I am close to such things however I don't have full control over my JupyterHub.

  You can check it out at
- http://hub.dubmathematics.com

My goal is to integrate this within a website that students can access using some kind of login token.

1.5 Suggestions for Course

Despite some bumps in the road, many students were able to complete excellent work. Here are some examples of student github repositories that house three projects and a completed website built with Django:

- https://github.com/charmillz/datadesign_python
- https://github.com/jchan9/Data_Design_Python/tree/master/Data/Projects
- https://github.com/warpz785/data-design
- https://github.com/kyler-ross/git-test

If I were to do the course over again, I would keep the aim for work with both Data Analysis and Web Design as the focus. Ideally, the class would be a regular 3 or 4 hour class where we can spend more time on all three areas. I would also be interested in connecting with other instructors who work in web design and data visualization to normalize the use of specific technologies.

1.6 Hypothetical Semester Length Version

Here is a prospective outline for such a class:

Section I: Data Analysis and Machine Learning

- **Week I**: Introduction to Python

Base introduction to the Python language. Jupyter notebooks and plotting. Saving and reusing programs.

- **Week II**: Introduction to Pandas

Introduction to Data Structures and the Pandas library. Students will work with built in and external datasets.

- **Week III**: Introduction to Machine Learning

We introduce machine learning through the Regression and Clustering algorithms. We will see how to implement each of these algorithms on our data structured with Pandas.

- **Week IV**: Machine Learning with TensorFlow

In this week, we introduce applications of machine learning to visual and audio problems with the Google TensorFlow machine learning library. Here we will discuss neural networks and their use in solving computer vision problems.

Section II: Data and the Internet

- **Week VI**: Introduction to WebScraping

This week focuses on data accession from the web. To start, we will scrape numerical tables into a Pandas DataFrame and use our earlier work with visualization and data analysis to explore the web data. Next we will focus on accessing and structuring textual data from tables in Wikipedia articles.

- **Week VII**: WebCrawling

This week we will use Scrapy to set up a web crawler that will extract data from multiple websites with a similar structure.

- **Week VIII**: Natural Language Processing I

Building on our earlier work with data analysis, we start turn text into data using the NLTK library. We discuss some introductory Natural Language Processing techniques and visualize novels from Project Gutenberg.

- **Week IX**: Machine Learning and Text

This week we focus on using Machine Learning to understand the sentiment and important topics in a range of text. This will take place with reviews on Yelp and Amazon.com.

Section III: Web Design with Django

- **Week X**: Introduction to Django

Setup a basic static website using the Python web framework Django. We will discuss the basics of how the internet works and complete a basic website that contains static HTML files that include some basic p5.js animations.

- **Week XI**: Django and Models

The week we explore the use of databases with Django applications. We will build a blog for our site and begin to post entries based on our eariler projects. Next, we see how we can analyze this data using our Juptyer notebooks.

- **Week XII**: Serving our Site

This week we complete our work with styling the basic site and serve it live to the internet using the Heroku service.

- **Week XIII**: User Authentication and Site Access

Adding to our website, we build a user authentication interface that allows us to restrict access to all or part of our website.

- **Week XIV**: Packaging your site as a reusable application

Finally, we will package our site for public use. We will use the Python standards to share our work with the larger world, including the launching of our frameworks on their own computer using a simple `pip install`.

2 Data and Design with Python

LMTH 2075: Spring 2018

Fridays 12:10 - 2:45

65 West 11th St, Room 458

3 Overview

This course covers some introductory ideas for data acquisition, analysis, and deployment on the web using the Python computer language. We cover basic data analysis and visualization, web scraping and crawling, some natural language processing and text analysis, web applications with Django, and game design with PyGame. By the end, students should feel comfortable pursuing further advanced work with Python and design.

3.1 Course Requirements

- Participation/Attendance 20%
- Lab Projects 80%

3.2 Course Materials

All available online through our github repository at https://github.com/jfkoehler/data-design/. This will be updated weekly as we move through our weeks together.

Also, you are to download and install Anaconda[1], making sure that you are able to run Jupyter notebooks on your computer. We will install additional software as we go.

3.3 Learning Outcomes

1. Use Python to perform basic data analysis
2. Use Matplotlib and Seaborn to visualize data
3. Use webscraping to access numerical and textual information
4. Use NLTK to investigate the text of scraped documents
5. Scrape multiple sites using a web crawlers and spiders
6. Deploy basic website and applications with Django

3.4 Resources

The university provides many resources to help students achieve academic and artistic excellence. These resources include: - University Libraries: http://library.newschool.edu - University Learning Center: http://www.newschool.edu/learning-center - University Disabilities Service: www.newschool.edu/student-disability-services/ In keeping with the university's policy of providing equal access for students with disabilities, any student with a disability who needs academic

[1] https://www.anaconda.com/download/#macos

accommodations is welcome to meet with me privately. All conversations will be kept confidential. Students requesting any accommodations will also need to contact Student Disability Service (SDS). SDS will conduct an intake and, if appropriate, the Director will provide an academic accommodation notification letter for you to bring to me. At that point, I will review the letter with you and discuss these accommodations in relation to this course. Student Ombuds: http://www.newschool.edu/intercultural-support/ombuds/ The Student Ombuds office provides students assistance in resolving conflicts, disputes or complaints on an informal basis. This office is independent, neutral, and confidential.

3.5 University Policies

University, College/School, and Program Policies [Faculty must include policies on academic honesty and attendance, as well as any required college/program policies]

Academic Honesty and Integrity Compromising your academic integrity may lead to serious consequences, including (but not limited to) one or more of the following: failure of the assignment, failure of the course, academic warning, disciplinary probation, suspension from the university, or dismissal from the university.

Students are responsible for understanding the University's policy on academic honesty and integrity and must make use of proper citations of sources for writing papers, creating, presenting, and performing their work, taking examinations, and doing research. It is the responsibility of students to learn the procedures specific to their discipline for correctly and appropriately differentiating their own work from that of others. The full text of the policy, including adjudication procedures, is found at http://www.newschool.edu/policies/

Resources regarding what plagiarism is and how to avoid it can be found on the Learning Center's website: http://www.newschool.edu/university-learning-center/avoiding-plagiarism.pdf [Additional college-specific standards for what constitutes academic dishonesty may be included here.]

Intellectual Property Rights: http://www.newschool.edu/provost/accreditation-policies/ Grade Policies: http://www.newschool.edu/registrar/academic-policies/

3.6 Attendance

"Absences may justify some grade reduction and a total of four absences mandate a reduction of one letter grade for the course. More than four absences mandate a failing grade for the course, unless there are extenuating circumstances, such as the following: an extended illness requiring hospitalization or visit to a physician (with documentation); a family emergency, e.g. serious illness (with written explanation); observance of a religious holiday.

The attendance and lateness policies are enforced as of the first day of classes for all registered students. If registered during the first week of the add/drop period, the student is responsible for any missed assignments and coursework.

For significant lateness, the instructor may consider the tardiness as an absence for the day. Students failing a course due to attendance should consult with an academic advisor to discuss options. Divisional and/or departmental/program policies serve as minimal guidelines, but policies may contain additional elements determined by the faculty member."

3.7 Student Course Ratings

During the last two weeks of the semester, students are asked to provide feedback for each of their courses through an online survey. They cannot view grades until providing feedback or officially declining to do so. Course evaluations are a vital space where students can speak about

the learning experience. It is an important process which provides valuable data about the successful delivery and support of a course or topic to both the faculty and administrators. Instructors rely on course rating surveys for feedback on the course and teaching methods, so they can understand what aspects of the class are most successful in teaching students, and what aspects might be improved or changed in future. Without this information, it can be difficult for an instructor to reflect upon and improve teaching methods and course design. In addition, program/department chairs and other administrators review course surveys. Instructions are available online at http://www.newschool.edu/provost/course-evaluations-student-instructions.pdf.

3.8 Workshop Outline

1. Data Analysis and Visualization with Pandas and Seaborn.
- Basic Overview of Python for Data Analysis and Visualization
- Access and investigate data from the World Bank using API
2. Data Acquisition and Web Scraping.
    - How to access and structure data from the web
    - Scrape and structure data from web sources
    - Introduce basic NLP tasks (tokenize, frequency distributions, stop word removal)
3. Natural Language Processing and Social Media Analysis with NLTK.
    - Use scraping knowledge to pull and structure text from web
    - Use additional Natural Language Processing techniques to investigate text from scraped sites
4. Web Application Development with Django
    - Basic Web Design Overview
    - Web Applications with DJango
5. Game Design with PyGame
    - Motion and Movement
    - Basic Games with PyGame

4 Exploring Data with Python

```
In [47]: %%HTML
         <iframe width="560" height="315" src="https://www.youtube.com/embed/WHdblAQHBms" frameborder="0"
```

```
<IPython.core.display.HTML object>
```

MATHEMATICAL GOALS

- Explore data based on a single variable
- Use summary descriptive statistics to understand distributions
- Introduce basic exploratory data analysis

PYTHON GOALS

- Introduce basic functionality of Pandas DataFrame
- Use Seaborn to visualize data
- Use Markdown cells to write and format text and images

MATERIALS

- Pandas Cheatsheet[2]
- Markdown Cheatsheet[3]
- Seaborn Tutorials and Documentation[4]

4.1 Introduction to the Jupyter Notebook

The Jupyter notebook has cells that can be used either as code cells or as markdown cells. Code cells will be where we execute Python code and commands. Markdown cells allow us to write and type, in order to further explain our work and produce reports.

Markdown

Markdown is a simplified markup language for formating text. For example, to make something bold, we would write `**bold**`. We can produce headers, insert images, and perform most standard formatting operations using markdown. Here is a markdown cheatsheet[5]. We can change a cell to a markdown cell with the toolbar, or with the keyboard shortcut `ctrl + m + m`. Create some markdown cells below, using the cheatsheet that has:

1. Your first and last name as a header
2. An ordered list of the reasons you want to learn Python
3. A blockquote embodying your feelings about mathematics

4.2 Libraries and Jupyter Notebook

Starting with Python it's important to understand how the notebook and Python work together. For the most part, we will not be writing all our code from scratch. There are powerful existing libraries that we can make use of with ready made functions that can accomplish most everything we'd want to do. When using a Jupyter notebook with with Python, we have to import any library that will be used. Each of the libraries we use today has a standard range of applications:

- **"pandas"**: Data Structure library, structures information in rows and columns and helps you rearrange and navigate the data.
- **"numpy"**: Numerical library, performs many mathematical operations and handles arrays. Pandas is actually built on top of `numpy`, we will use it primarily for generates arrays of numbers and basic mathematical operations.
- **"matplotlib"**: Plotting Library, makes plots for many situations and has deep customization possibilities. Useful in wide variety of contexts.
- **"seaborn"**: Statistical plotting library. Similar to `matplotlib` in that it is a plotting library, `seaborn` produces nice visualizations eliminating much of the work necessary for producing similar visualizations with `matplotlib`.

To import the libraries, we will write

```
import numpy as np
```

[2] https://github.com/pandas-dev/pandas/blob/master/doc/cheatsheet/Pandas_Cheat_Sheet.pdf
[3] https://guides.github.com/pdfs/markdown-cheatsheet-online.pdf
[4] https://seaborn.pydata.org/tutorial
[5] https://guides.github.com/pdfs/markdown-cheatsheet-online.pdf

and hit `shift + enter` to execute the cell. This code tells the notebook we want to have the `numpy` library loaded, and when we want to refer to a method from `numpy` we will preface it with `np`. For example, if we wanted to find the cosine of 10, `numpy` has a cosine function, and we write:

```
np.cos(10)
```

If we have questions about the function itself, we can use the help function by including a question mark at the end of the function.

```
np.cos?
```

A second example from `seaborn` involves loading a dataset that is part of the library call "tips".

```
sns.load_dataset("tips")
```

Here, we are calling something from the Seaborn package (`sns`), using the `load_dataset` function, and the dataset we want it to load is contained in the parenthesis.`("tips")`

```
In [1]: %matplotlib inline
        import matplotlib.pyplot as plt
        import numpy as np
        import pandas as pd
        import seaborn as sns

In [2]: np.cos(10)

Out[2]: -0.83907152907645244

In [3]: np.cos?

In [4]: tips = sns.load_dataset("tips")

In [ ]: #save the dataset as a csv file
        tips.to_csv('data/tips.csv')
```

4.3 Pandas Dataframe

The Pandas library is the standard Python data structure library. A `DataFrame` is an object similar to that of an excel spreadsheet, where there is a collection of data arranged in rows and columns. The datasets from the `Seaborn` package are loaded as `Pandas DataFrame` objects. We can see this by calling the `type` function. Further, we can investigate the data by looking at the first few rows with the `head()` function.

This is an application of a function to a pandas object, so we will write

```
tips.head()
```

If we wanted a different number of rows displayed, we could input this in the `()`. Further, there is a similar function `tail()` to display the end of the `DataFrame`.

```
In [5]: type(tips)

Out[5]: pandas.core.frame.DataFrame

In [6]: #look at first five rows of data
        tips.head()

Out[6]: total_bill   tip     sex smoker  day    time  size
        0       16.99  1.01  Female     No  Sun  Dinner     2
        1       10.34  1.66    Male     No  Sun  Dinner     3
        2       21.01  3.50    Male     No  Sun  Dinner     3
        3       23.68  3.31    Male     No  Sun  Dinner     2
        4       24.59  3.61  Female     No  Sun  Dinner     4
```

```
In [7]: #look at first five rows of total bill column
        tips["total_bill"]
```

```
Out[7]: 0      16.99
        1      10.34
        2      21.01
        3      23.68
        4      24.59
        5      25.29
        6       8.77
        7      26.88
        8      15.04
        9      14.78
        10     10.27
        11     35.26
        12     15.42
        13     18.43
        14     14.83
        15     21.58
        16     10.33
        17     16.29
        18     16.97
        19     20.65
        20     17.92
        21     20.29
        22     15.77
        23     39.42
        24     19.82
        25     17.81
        26     13.37
        27     12.69
        28     21.70
        29     19.65
               ...
        214    28.17
        215    12.90
        216    28.15
        217    11.59
        218     7.74
        219    30.14
        220    12.16
        221    13.42
        222     8.58
        223    15.98
        224    13.42
        225    16.27
        226    10.09
        227    20.45
        228    13.28
        229    22.12
        230    24.01
        231    15.69
        232    11.61
        233    10.77
        234    15.53
        235    10.07
        236    12.60
        237    32.83
        238    35.83
        239    29.03
        240    27.18
```

```
241    22.67
242    17.82
243    18.78
Name: total_bill, Length: 244, dtype: float64
```

In [8]:
```
tips["total_bill"].head()
```

Out[8]:
```
0    16.99
1    10.34
2    21.01
3    23.68
4    24.59
Name: total_bill, dtype: float64
```

In [9]:
```
#find the mean of the tips column
tips["tip"].mean()
```

Out[9]:
```
2.9982786885245902
```

In [10]:
```
tips["tip"].median()
```

Out[10]:
```
2.9
```

In [11]:
```
tips["tip"].mode()
```

Out[11]:
```
0    2.0
dtype: float64
```

In [12]:
```
tips["smoker"].unique()
```

Out[12]:
```
[No, Yes]
Categories (2, object): [No, Yes]
```

In [13]:
```
#groups the dataset by the sex column
group = tips.groupby("sex")
```

In [14]:
```
group.head()
```

Out[14]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4
5	25.29	4.71	Male	No	Sun	Dinner	4
6	8.77	2.00	Male	No	Sun	Dinner	2
11	35.26	5.00	Female	No	Sun	Dinner	4
14	14.83	3.02	Female	No	Sun	Dinner	2
16	10.33	1.67	Female	No	Sun	Dinner	3

In [15]:
```
group.first()
```

Out[15]:

sex	total_bill	tip	smoker	day	time	size
Male	10.34	1.66	No	Sun	Dinner	3
Female	16.99	1.01	No	Sun	Dinner	2

In [16]:
```
smoker = tips.groupby("smoker")
```

In [17]:
```
smoker.first()
```

Out[17]:

smoker	total_bill	tip	sex	day	time	size
Yes	38.01	3.00	Male	Sat	Dinner	4
No	16.99	1.01	Female	Sun	Dinner	2

In [18]:
```
group.last()
```

Out[18]:

sex	total_bill	tip	smoker	day	time	size

```
Male          17.82  1.75     No   Sat  Dinner     2
Female        18.78  3.00     No  Thur  Dinner     2
```

```
In [19]: group.sum()
```

```
Out[19]: total_bill     tip  size
         sex
         Male       3256.82  485.07   413
         Female     1570.95  246.51   214
```

```
In [20]: group.mean()
```

```
Out[20]: total_bill       tip      size
         sex
         Male     20.744076  3.089618  2.630573
         Female   18.056897  2.833448  2.459770
```

As shown above, we can refer to specific elements of a DataFrame in a variety of ways. For more information on this, please consult the Pandas Cheatsheet here[6]. Use the cheatsheet, google, and the help functions to perform the following operations.

PROBLEMS: SLICE AND DICE DATAFRAME

1. Select Column: Create a variable named `size` that contains the size column from the tips dataset. Use Pandas to determine how many unique values are in the column, i.e. how many different sized dining parties are a part of this dataset.
2. Select Row: Investigate how the `pd.loc` and `pd.iloc` methods work to select rows. Use each to select a single row, and a range of rows from the tips dataset.
3. Groupby: As shown above, we can group data based on labels, and perform statistical operations within these groups. Use the `groupby` function to determine whether smokers or non-smokers gave better tips on average.
4. Pivot Table: A Pivot Table takes rows and spreads them into columns. Try entering:

```
tips.pivot(columns='smoker', values='tip').describe()
```

What other way might you split rows in the data to make comparisons?

```
In [21]: size = tips["size"]
```

```
In [22]: size.head()
```

```
Out[22]: 0    2
         1    3
         2    3
         3    2
         4    4
         Name: size, dtype: int64
```

```
In [23]: tips.iloc[4:10]
```

```
Out[23]: total_bill   tip     sex smoker  day    time  size
         4       24.59  3.61  Female     No  Sun  Dinner     4
         5       25.29  4.71    Male     No  Sun  Dinner     4
         6        8.77  2.00    Male     No  Sun  Dinner     2
         7       26.88  3.12    Male     No  Sun  Dinner     4
         8       15.04  1.96    Male     No  Sun  Dinner     2
         9       14.78  3.23    Male     No  Sun  Dinner     2
```

```
In [24]: tips.loc[tips["smoker"]=="Yes"].mean()
```

```
Out[24]: total_bill    20.756344
         tip            3.008710
```

[6] https://github.com/pandas-dev/pandas/blob/master/doc/cheatsheet/Pandas_Cheat_Sheet.pdf

```
size           2.408602
dtype: float64
```

```
In [25]: tips.pivot(columns='smoker', values='tip').describe()
```

```
Out[25]: smoker        Yes          No
         count   93.000000  151.000000
         mean     3.008710    2.991854
         std      1.401468    1.377190
         min      1.000000    1.000000
         25%      2.000000    2.000000
         50%      3.000000    2.740000
         75%      3.680000    3.505000
         max     10.000000    9.000000
```

```
In [26]: tips.describe()
```

```
Out[26]: total_bill         tip        size
         count  244.000000  244.000000  244.000000
         mean    19.785943    2.998279    2.569672
         std      8.902412    1.383638    0.951100
         min      3.070000    1.000000    1.000000
         25%     13.347500    2.000000    2.000000
         50%     17.795000    2.900000    2.000000
         75%     24.127500    3.562500    3.000000
         max     50.810000   10.000000    6.000000
```

4.4 Vizualizing Data with Seaborn

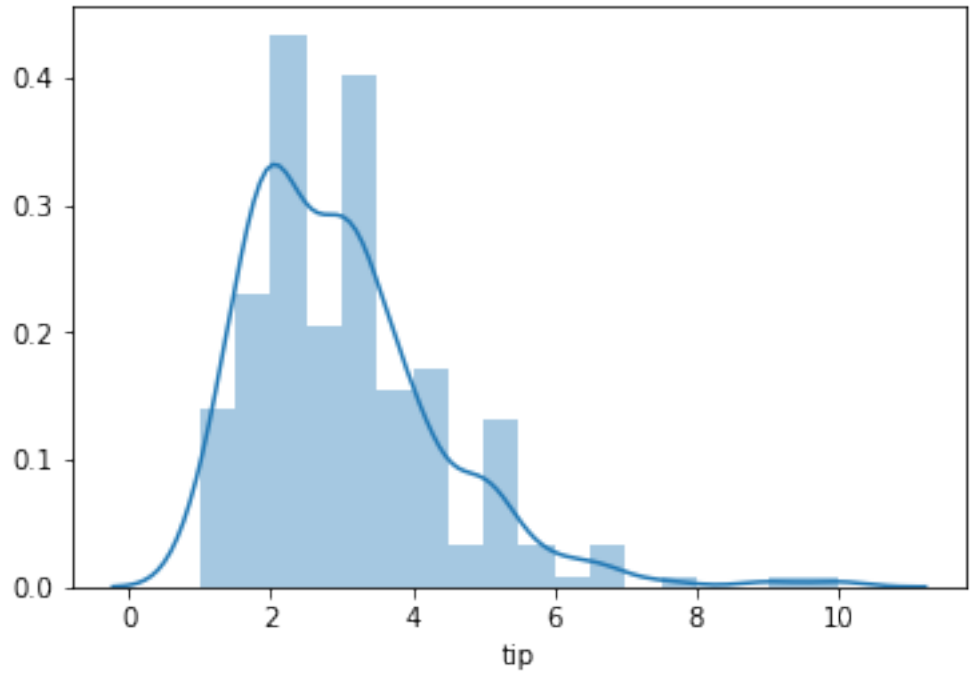
Visualizing the data will help us to see larger patterns and structure within a dataset. We begin by examining the distribution of a single variable. It is important to note the difference between a **quantitative** and **categorical** variable here. One of our first strategies for exploring data will be to look at a quantitative variable grouped by some category. For example, we may ask the questions:

- What is the distribution of tips?
- Is the distribution of tips different across the category gender?
- Is the distribution of tip amounts different across the category smoker or non-smoker?

We will use the **"seaborn"** library to visualize these distributions. To explore a single distribution we can use the `distplot` function. For example, below we visualize the tip amounts from our tips data set.

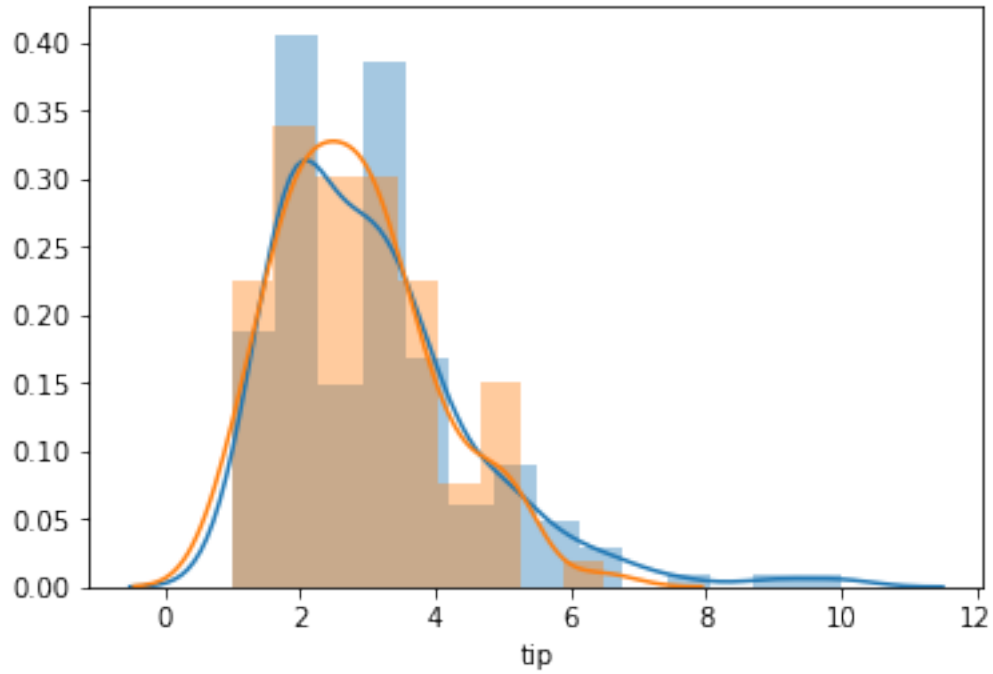
```
In [28]: sns.distplot(tips["tip"])
```

```
Out[28]: <matplotlib.axes._subplots.AxesSubplot at 0x1a09d96cc0>
```



We can now explore the second question, realizing that we will need to structure our data to plot accordingly. For this distribution plot, we will call two plots.

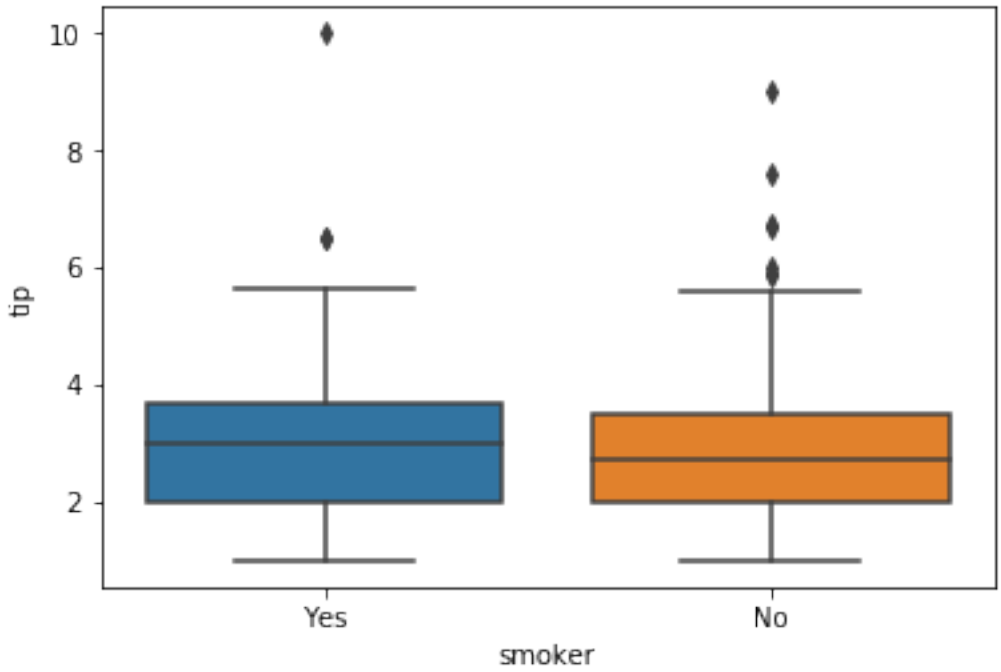
```
In [29]: male = tips.loc[tips["sex"] == "Male", ["sex", "tip"]]
         female = tips.loc[tips["sex"] == "Female", ["sex", "tip"]]

In [30]: male.head()

Out[30]: sex   tip
         1  Male  1.66
         2  Male  3.50
         3  Male  3.31
         5  Male  4.71
         6  Male  2.00

In [31]: sns.distplot(male["tip"])
         sns.distplot(female["tip"])

Out[31]: <matplotlib.axes._subplots.AxesSubplot at 0x1a12404f98>
```



Another way to compare two or more categories is with a `boxplot`. Here, we can answer our third question without having to rearannge the original data.

```
In [32]: sns.boxplot(x = "smoker",y = "tip", data = tips )
```

```
Out[32]: <matplotlib.axes._subplots.AxesSubplot at 0x1a12547080>
```



This is a visual display of the data produced by splitting on the smoker category, and comparing the median and quartiles of the two groups. We can see this numerically with the following code that chains together three methods: `groupby`(groups smokers), `describe`(summary statistics for data), `.T`(transpose–swaps the rows and columns of the output to familiar form).

```
In [33]: tips.groupby(by = "smoker")["tip"].describe()
```

```
Out[33]: count      mean       std  min  25%   50%    75%   max
         smoker
         Yes      93.0  3.008710  1.401468  1.0  2.0  3.00  3.680  10.0
         No      151.0  2.991854  1.377190  1.0  2.0  2.74  3.505   9.0
```

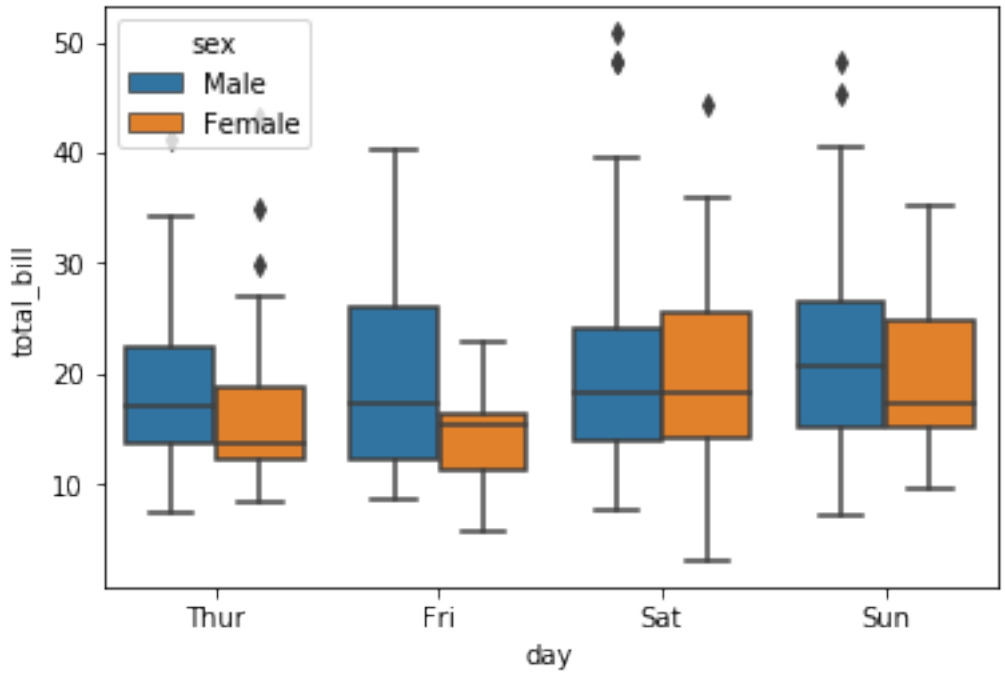
```
In [34]: tips.groupby(by = "smoker")["tip"].describe().T
```

```
Out[34]: smoker        Yes          No
         count   93.000000  151.000000
         mean     3.008710    2.991854
         std      1.401468    1.377190
         min      1.000000    1.000000
         25%      2.000000    2.000000
         50%      3.000000    2.740000
         75%      3.680000    3.505000
         max     10.000000    9.000000
```

What days do men seem to spend more money than women? Are these the same as when men tip better than women?

```
In [36]: sns.boxplot(x = "day", y = "total_bill", hue = "sex", data = tips)
```

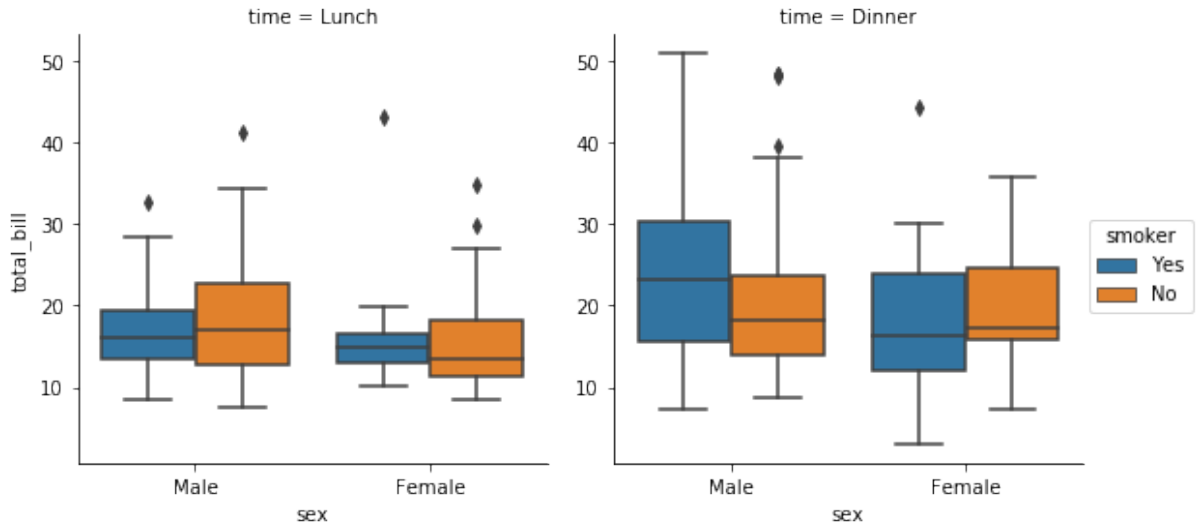
```
Out[36]: <matplotlib.axes._subplots.AxesSubplot at 0x1a12646d30>
```



To group the data even further, we can use a `factorplot`. For example, we break the plots for gender and total bill apart creating a plot for Dinner and Lunch that break the genders by smoking categories. Can you think of a different way to combine categories from the tips data?

```
In [37]: sns.factorplot(x="sex", y="total_bill",
                        hue="smoker", col="time",
                        data=tips, kind="box")
```

```
Out[37]: <seaborn.axisgrid.FacetGrid at 0x1a12769c18>
```



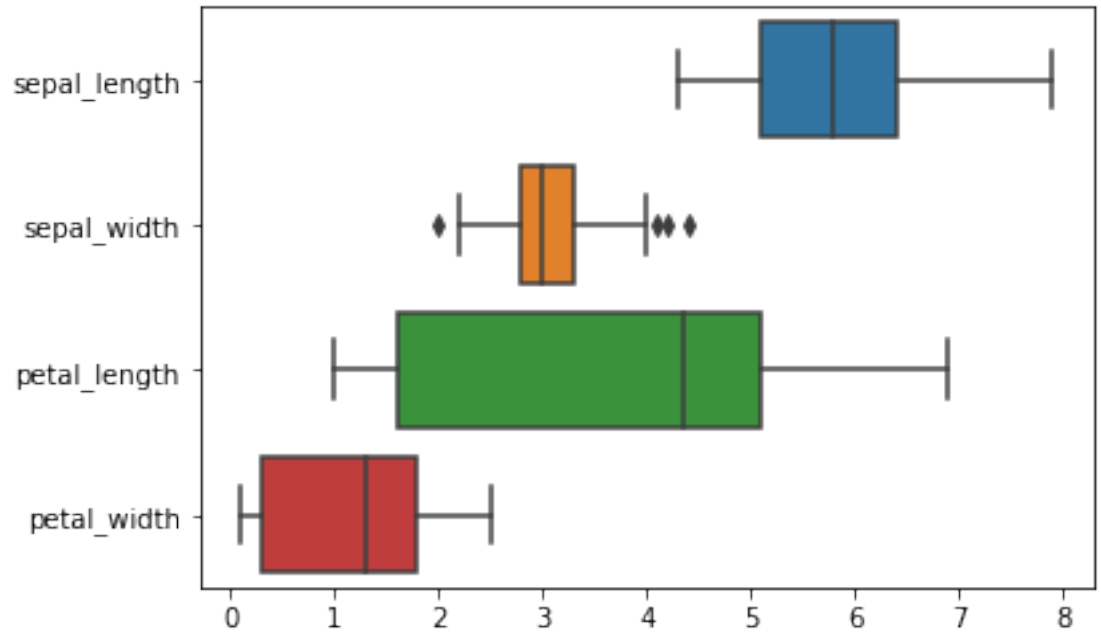
4.5 Playing with More Data

Below, we load two other built-in datasets; the iris and titanic datasets. Use seaborn to explore distributions of quantitative variables and within groups of categories. Use the notebook and a markdown cell to write a clear question about both the `iris` and `titanic` datasets. Write a response to these questions that contains both a visualization, and a written response that uses complete sentences to help understand what you see within the data relevant to your questions.

Iris Data Dataset with information about three different species of flowers, and corresponding measurements of `sepal_length`, `sepal_width`, `petal_length`, and `petal_width`.

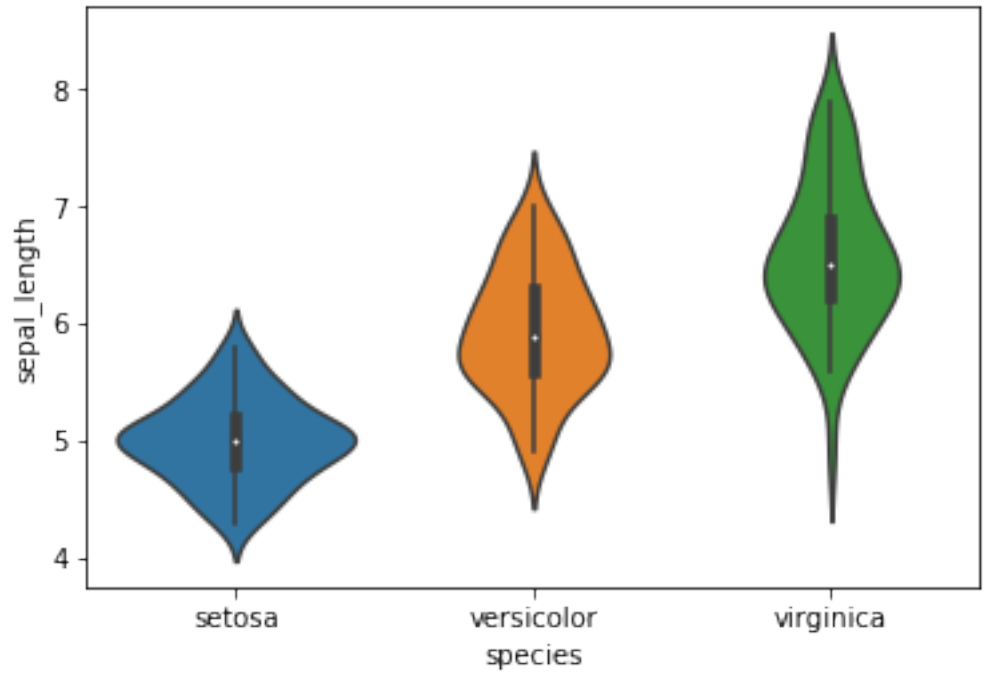
Titanic Data Data with information about the passengers on the famed titanic cruise ship including whether or not they survived the crash, how old they were, what class they were in, etc.

```
In [38]: iris = sns.load_dataset('iris')

In [39]: iris.head()

Out[39]: sepal_length  sepal_width  petal_length  petal_width species
         0           5.1          3.5           1.4          0.2  setosa
         1           4.9          3.0           1.4          0.2  setosa
         2           4.7          3.2           1.3          0.2  setosa
         3           4.6          3.1           1.5          0.2  setosa
         4           5.0          3.6           1.4          0.2  setosa

In [40]: sns.boxplot(data=iris, orient="h")

Out[40]: <matplotlib.axes._subplots.AxesSubplot at 0x1a12a7d6a0>
```



In [41]:
```
sns.violinplot(x=iris.species, y=iris.sepal_length)
```

Out[41]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a128e9fd0>
```



In [42]:
```
titanic = sns.load_dataset('titanic')
```

In [43]:
```
titanic.head()
```

Out[43]:

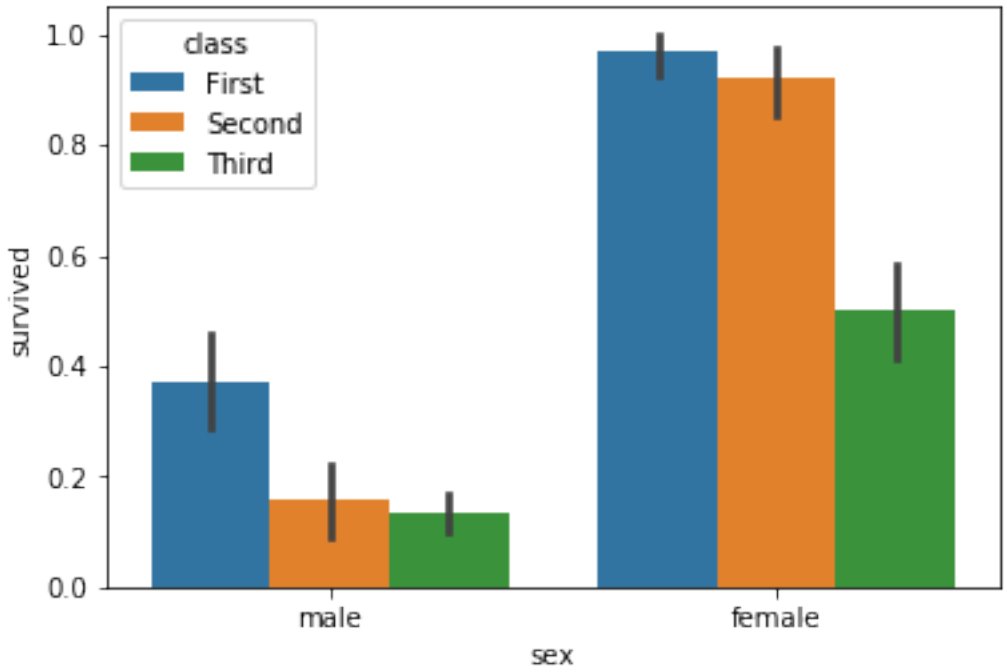
	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	\
0	0	3	male	22.0	1	0	7.2500	S	Third	
1	1	1	female	38.0	1	0	71.2833	C	First	
2	1	3	female	26.0	0	0	7.9250	S	Third	
3	1	1	female	35.0	1	0	53.1000	S	First	
4	0	3	male	35.0	0	0	8.0500	S	Third	

	who	adult_male	deck	embark_town	alive	alone

```
     0    man        True  NaN  Southampton    no  False
     1  woman       False    C    Cherbourg   yes  False
     2  woman       False  NaN  Southampton   yes   True
     3  woman       False    C  Southampton   yes  False
     4    man        True  NaN  Southampton    no   True
```

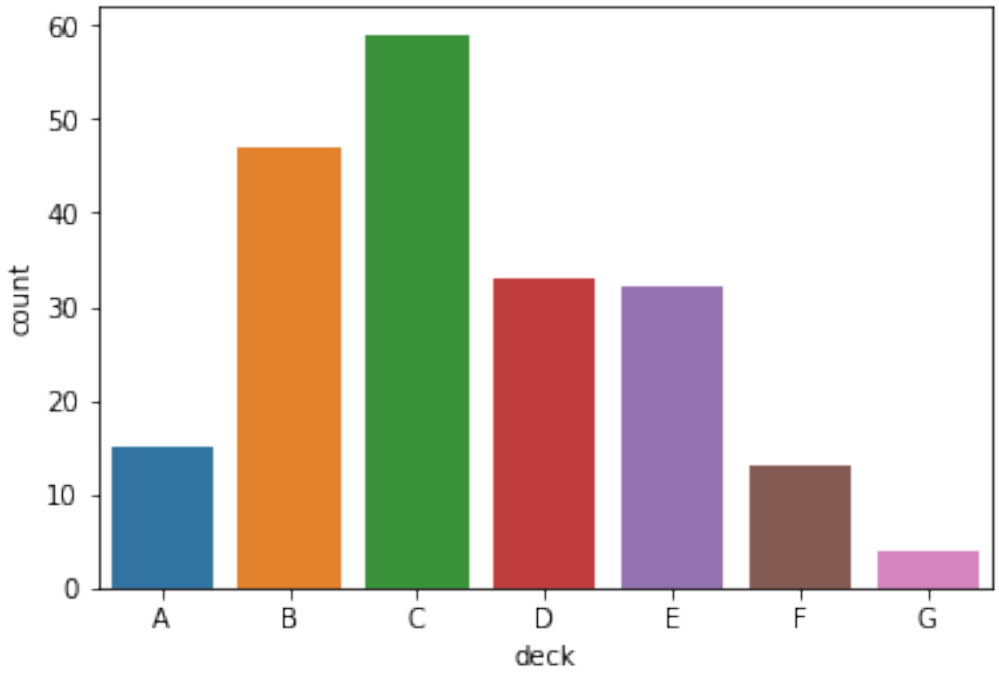
In [44]:
```
sns.barplot(x="sex", y="survived", hue="class", data=titanic)
```

Out[44]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a12cf9cc0>
```



In [45]:
```
sns.countplot(x="deck", data=titanic)
```

Out[45]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a12ce0358>
```



5 Data Accession

For today's workshop we will be using the `pandas` library, the `matplotlib` library, and the `seaborn` library. Also, we will read data from the web with the `pandas-datareader`. By the end of the workshop, participants should be able to use Python to tell a story about a dataset they build from an open data source.

GOALS:

- Understand how to load data as `.csv` files into Pandas
- Import data from web with `pandas-datareader` and compare development indicators from the World Bank
- Use API's and requests to pull data from web

5.1 `.csv` files

In the first session, we explored built-in datasets. Typically, we would want to use our own data for analysis. A common filetype is the `.csv` or comma separated values type. You have probably used a spreadsheet program before, something like Microsoft Excel or Google Sheets. These programs allow you to save the data as a universally recognized formats, including the `.csv` extension. This is important as the `.csv` filetype can be understood and read by most data analysis languages including Python and R.

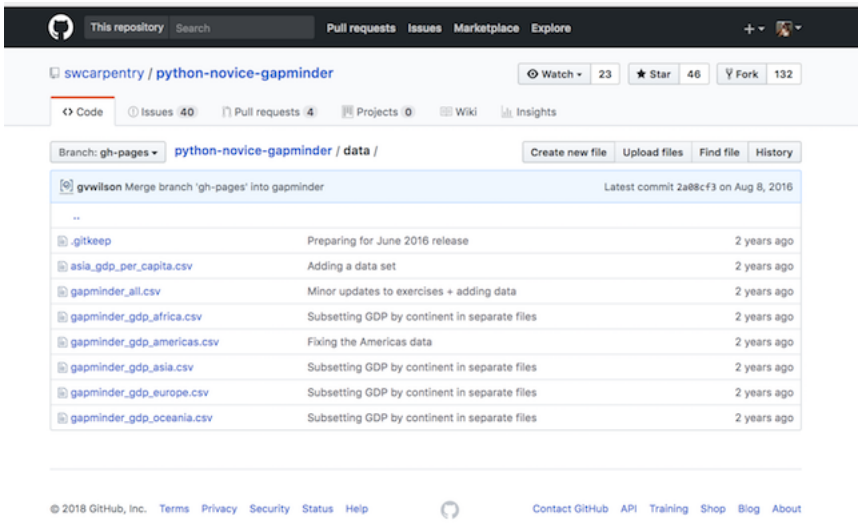
To begin, we will use Python to load a `.csv` file. Starting with the tips dataset from last lesson, we will save this data as a csv file in our data folder. Then, we can read the data in using Pandas `read_csv` method.

```
In [1]: %matplotlib notebook
        import matplotlib.pyplot as plt
        import numpy as np
        import pandas as pd
        import seaborn as sns

In [2]: tips = sns.load_dataset("tips")

In [3]: tips.head()

Out[3]: total_bill   tip     sex smoker  day    time  size
        0       16.99  1.01  Female     No  Sun  Dinner     2
        1       10.34  1.66    Male     No  Sun  Dinner     3
        2       21.01  3.50    Male     No  Sun  Dinner     3
        3       23.68  3.31    Male     No  Sun  Dinner     2
        4       24.59  3.61  Female     No  Sun  Dinner     4

In [4]: tips.to_csv('data/tips.csv')

In [5]: tips = pd.read_csv('data/tips.csv')

In [6]: tips.head()

Out[6]: Unnamed: 0  total_bill   tip     sex smoker  day    time  size
        0           0       16.99  1.01  Female     No  Sun  Dinner     2
        1           1       10.34  1.66    Male     No  Sun  Dinner     3
        2           2       21.01  3.50    Male     No  Sun  Dinner     3
        3           3       23.68  3.31    Male     No  Sun  Dinner     2
        4           4       24.59  3.61  Female     No  Sun  Dinner     4

In [13]: # add a column for tip percent
         tips['tip_pct'] = tips['tip']/tips['total_bill']

In [14]: # create variable grouped that groups the tips by sex and smoker
         grouped = tips.groupby(['sex', 'smoker'])
```

```
In [15]: # create variable grouped_pct that contains the tip_pct column from grouped
         grouped_pct = grouped['tip_pct']

In [16]: #what does executing this cell show?  Explain the .agg method.
         grouped_pct.agg('mean')

Out[16]: sex     smoker
         Female  No        0.156921
                 Yes       0.182150
         Male    No        0.160669
                 Yes       0.152771
         Name: tip_pct, dtype: float64

In [19]: # What other options can you pass to the .agg function?
         grouped_pct.agg(['mean', 'std'])

Out[19]:          mean       std
         sex    smoker
         Female No      0.156921  0.036421
                Yes     0.182150  0.071595
         Male   No      0.160669  0.041849
                Yes     0.152771  0.090588

In [20]: grouped_pct.agg?
```

5.2 Reading `.csv` files from web

If we have access to the file as a url, we can use the Pandas `read_csv` method to pass the url of the csv file instead of loading it from our local machine. For example, the Data and Software Carpentry organizations have a `.csv` file located in their github repository as seen below.



The first file on `asia_gdp_per_capita` can be loaded by using the link to the raw file on github:

https://raw.githubusercontent.com/swcarpentry/python-novice-gapminder/gh-pages/data/asia_gdp_per_capita.csv

hence, we pass this url to the `read_csv` function and have a new dataframe.

```
In [7]: asia_gdp = pd.read_csv('https://raw.githubusercontent.com/swcarpentry/python-novice-gapminder/gh-p

In [8]: asia_gdp.head()

Out[8]: 'year'  'Afghanistan'     'Bahrain'  'Bangladesh'  'Cambodia'     'China'  \
        0    1952     779.445314   9867.084765    684.244172  368.469286  400.448611
```

```
1    1957     820.853030  11635.799450    661.637458  434.038336  575.987001
2    1962     853.100710  12753.275140    686.341554  496.913648  487.674018
3    1967     836.197138  14804.672700    721.186086  523.432314  612.705693
4    1972     739.981106  18268.658390    630.233627  421.624026  676.900092

   'Hong Kong China'     'India'  'Indonesia'       'Iran'      ...       \
0        3054.421209  546.565749   749.681655  3035.326002      ...
1        3629.076457  590.061996   858.900271  3290.257643      ...
2        4692.648272  658.347151   849.289770  4187.329802      ...
3        6197.962814  700.770611   762.431772  5906.731805      ...
4        8315.928145  724.032527  1111.107907  9613.818607      ...

   'Philippines'  'Saudi Arabia'  'Singapore'  'Sri Lanka'      'Syria'  \
0    1272.880995     6459.554823  2315.138227  1083.532030  1643.485354
1    1547.944844     8157.591248  2843.104409  1072.546602  2117.234893
2    1649.552153    11626.419750  3674.735572  1074.471960  2193.037133
3    1814.127430    16903.048860  4977.418540  1135.514326  1881.923632
4    1989.374070    24837.428650  8597.756202  1213.395530  2571.423014

      'Taiwan'   'Thailand'   'Vietnam'  'West Bank and Gaza'  'Yemen Rep.'
0  1206.947913   757.797418  605.066492           1515.592329    781.717576
1  1507.861290   793.577415  676.285448           1827.067742    804.830455
2  1822.879028  1002.199172  772.049160           2198.956312    825.623201
3  2643.858681  1295.460660  637.123289           2649.715007    862.442146
4  4062.523897  1524.358936  699.501644           3133.409277   1265.047031

[5 rows x 34 columns]
```

5.3 Problems

Try to locate and load some `.csv` files using the internet. There are many great resources out there. Also, I want you to try the `pd.read_clipboard` method, where you've copied a data table from the internet. In both cases create a brief exploratory notebook for the data that contains the following:

- Jupyter notebook with analysis and discussion
- Data folder with relevant `.csv` files
- Images folder with at least one image loaded into the notebook

5.4 Accessing data through API

Pandas has the functionality to access certain data through a datareader. We will use the `pandas_datareader` to investigate information about the World Bank. For more information, please see the documentation:

http://pandas-datareader.readthedocs.io/en/latest/remote_data.html

We will explore other examples with the datareader later, but to start let's access the World Bank's data. For a full description of the available data, look over the source from the World Bank.

https://data.worldbank.org/indicator

```
In [38]: from pandas_datareader import wb

In [39]: import datetime

In [40]: wb.search('gdp.*capita.*const').iloc[:,:2]

Out[40]: id                                              name
         646     6.0.GDPpc_constant  GDP per capita, PPP (constant 2011 internation...
         8064        NY.GDP.PCAP.KD                 GDP per capita (constant 2010 US$)
```

```
8066        NY.GDP.PCAP.KN                      GDP per capita (constant LCU)
8068     NY.GDP.PCAP.PP.KD  GDP per capita, PPP (constant 2011 internation...
8069  NY.GDP.PCAP.PP.KD.87  GDP per capita, PPP (constant 1987 internation...
```

In [41]:
```
dat = wb.download(indicator='NY.GDP.PCAP.KD', country=['US','CA','MX'], start = 2005, end = 2016)
```

In [43]:
```
dat['NY.GDP.PCAP.KD'].groupby(level=0).mean()
```

Out[43]:
```
country
Canada           48601.353408
Mexico            9236.997678
United States    49731.965366
Name: NY.GDP.PCAP.KD, dtype: float64
```

In [44]:
```
wb.search('cell.*%').iloc[:,:2]
```

Out[44]:
```
id                                                 name
6339  IT.CEL.COVR.ZS  Population covered by mobile cellular network (%)
6394   IT.MOB.COV.ZS  Population coverage of mobile cellular telepho...
```

In [45]:
```
ind = ['NY.GDP.PCAP.KD', 'IT.MOB.COV.ZS']
```

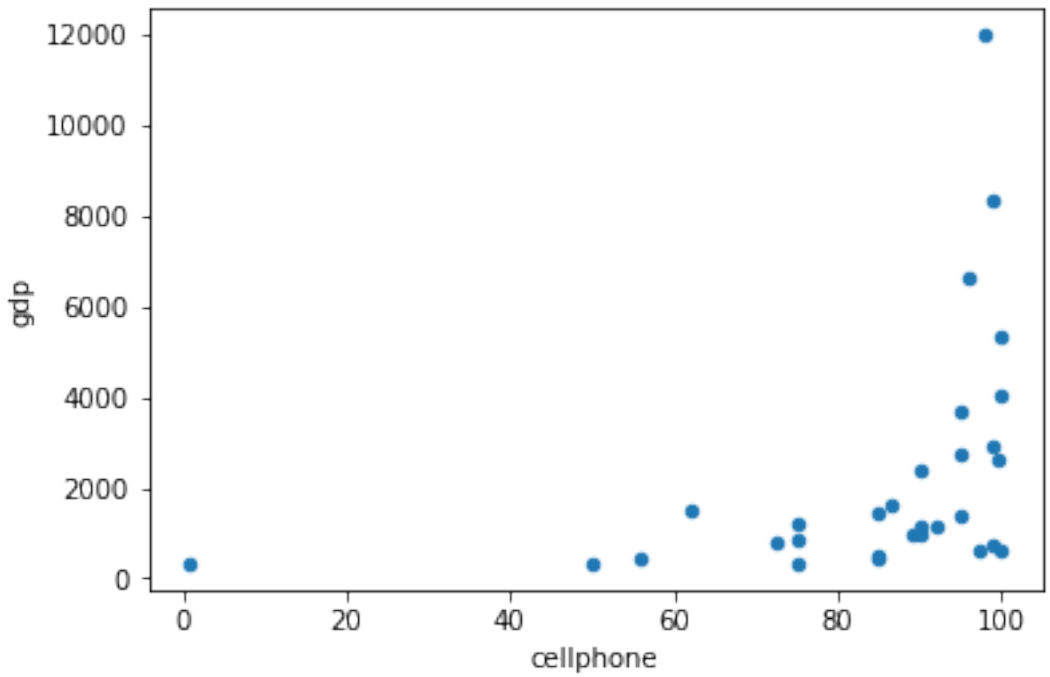
In [46]:
```
dat = wb.download(indicator=ind, country = 'all', start = 2011, end = 2011).dropna()
```

In [47]:
```
dat.columns = ['gdp', 'cellphone']
dat.tail()
```

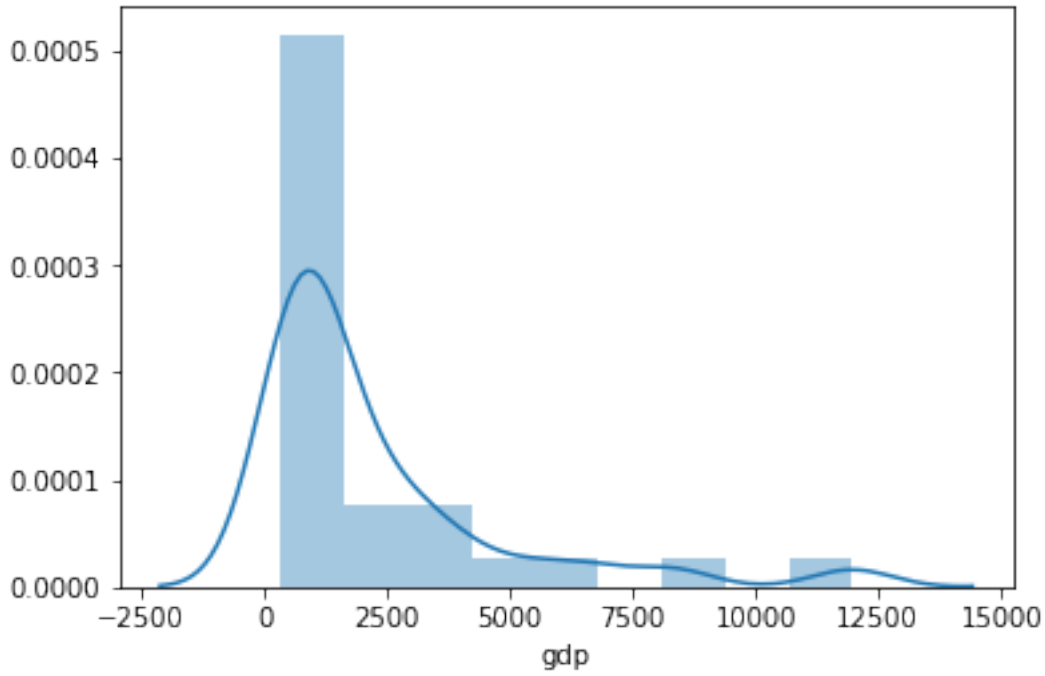
Out[47]:
```
gdp  cellphone
country   year
Swaziland 2011  3704.140658       94.9
Tunisia   2011  4014.916793      100.0
Uganda    2011   629.240447      100.0
Zambia    2011  1499.728311       62.0
Zimbabwe  2011   813.834010       72.4
```

In [48]:
```
dat.plot(x ='cellphone', y = 'gdp', kind = 'scatter')
```

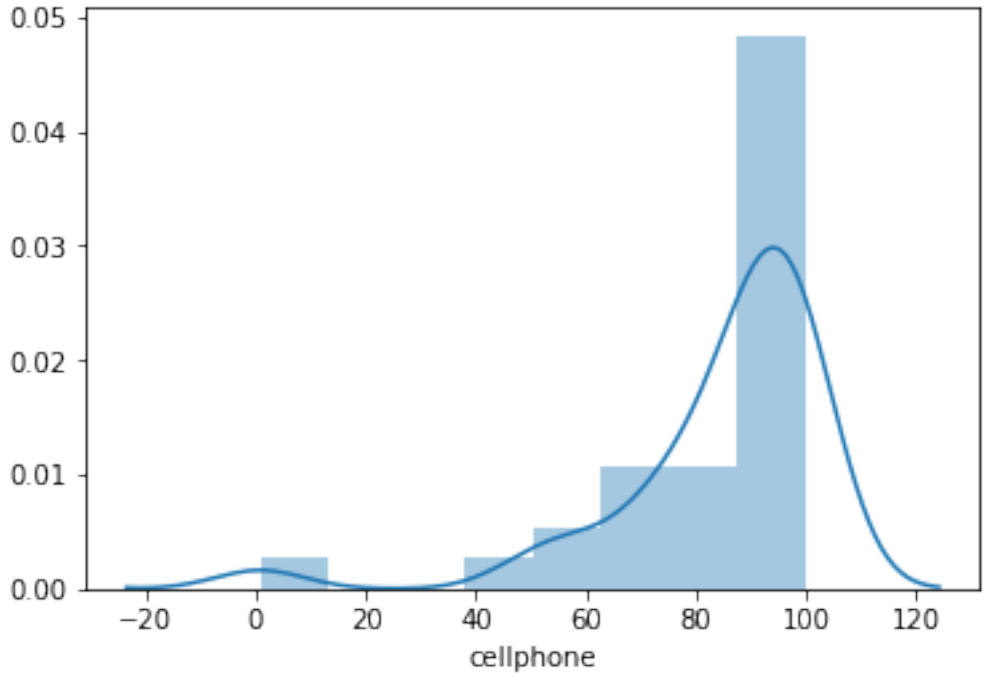
Out[48]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a2215fe80>
```



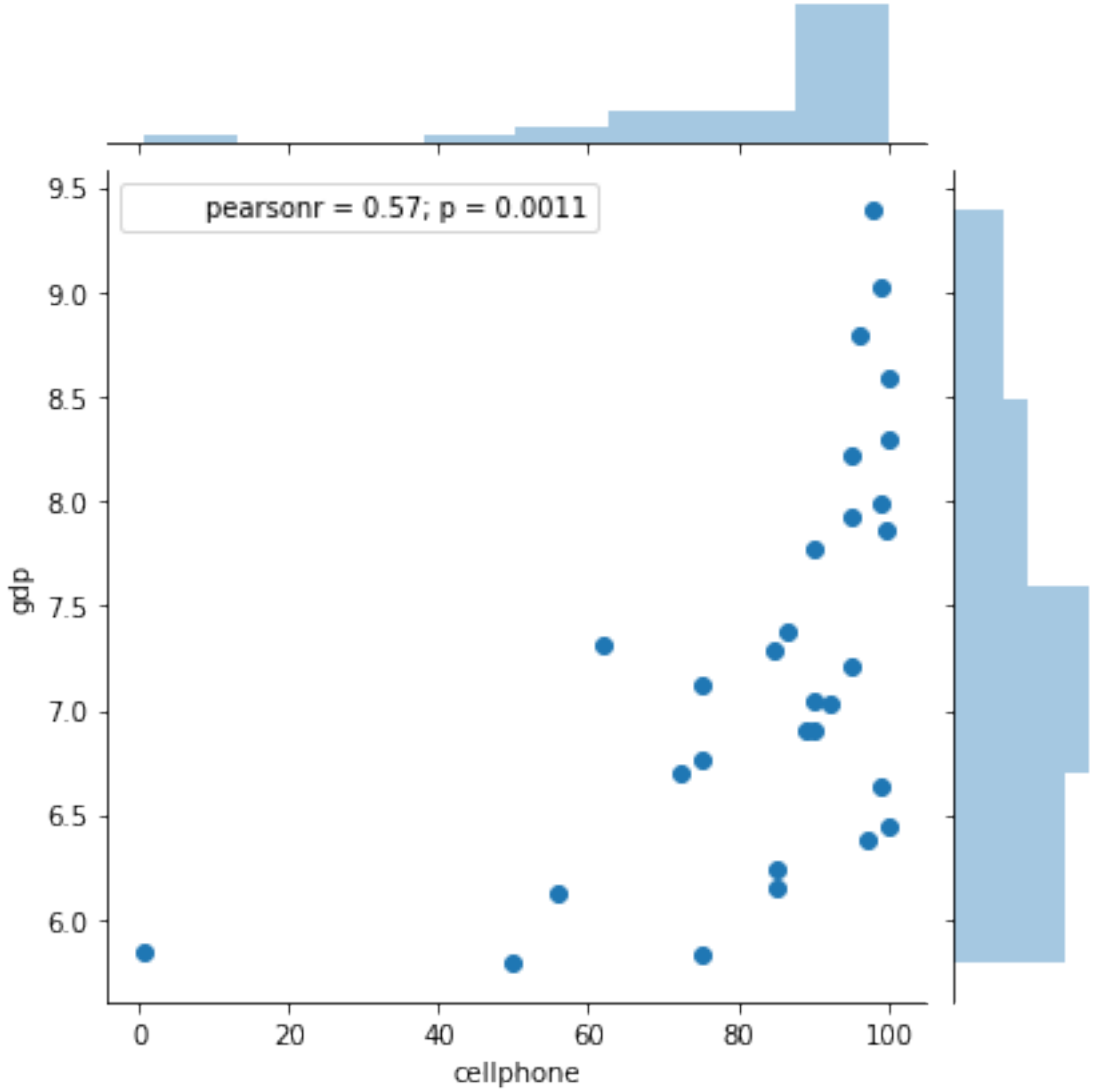
In [49]:
```
sns.distplot(dat['gdp']);
```




```
In [50]: sns.distplot(dat['cellphone']);
```

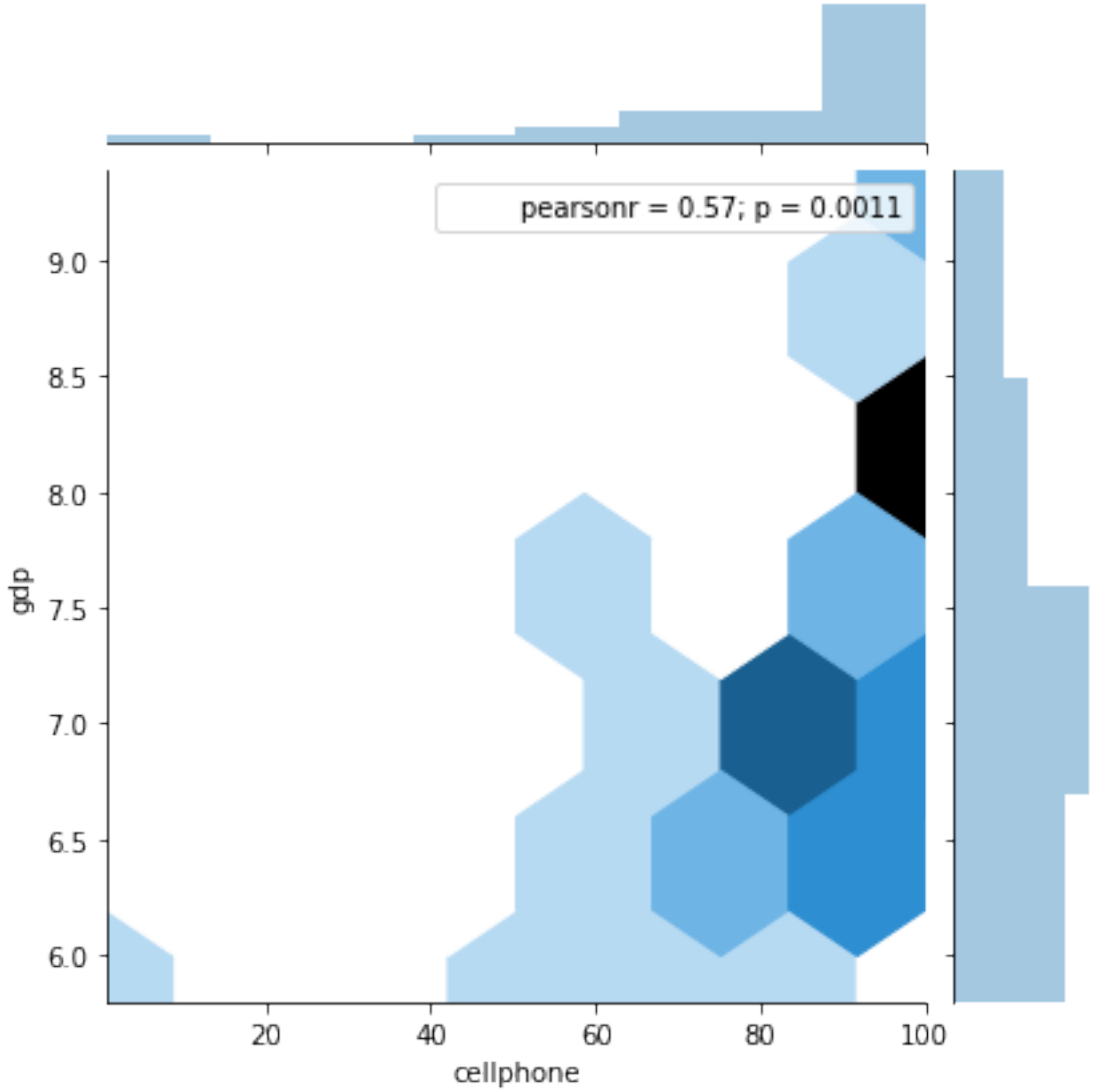



```
In [51]: sns.jointplot(dat['cellphone'], np.log(dat['gdp']))
Out[51]: <seaborn.axisgrid.JointGrid at 0x1a22099cf8>
```

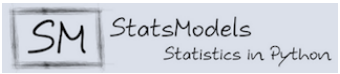



```
In [52]: sns.jointplot(dat['cellphone'], np.log(dat['gdp']), kind = 'hex')
```

```
Out[52]: <seaborn.axisgrid.JointGrid at 0x1a2144fc88>
```



5.5 StatsModels



StatsModels is a library that contains a wealth of classical statistical techniques. Depending on your comfort or interest in deeper use of classical statistics, you can consult the documentation at http://www.statsmodels.org/stable/index.html . Below, we show how to use statsmodels to perform a basic ordinary least squares fit with our y or dependent variable as `cellphone` and x or independent variable as `log(gdp)`.

```
In [53]: import numpy as np
         import statsmodels.formula.api as smf
         mod = smf.ols("cellphone ~ np.log(gdp)", dat).fit()

In [54]: mod.summary()

Out[54]: <class 'statsmodels.iolib.summary.Summary'>
         """
                                    OLS Regression Results
         ==============================================================================
         Dep. Variable:              cellphone   R-squared:                       0.321
```

```
Model:                            OLS   Adj. R-squared:                  0.296
Method:                 Least Squares   F-statistic:                     13.21
Date:                Sat, 13 Jan 2018   Prob (F-statistic):            0.00111
Time:                        12:28:27   Log-Likelihood:                -127.26
No. Observations:                  30   AIC:                             258.5
Df Residuals:                      28   BIC:                             261.3
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      -2.3708     24.082     -0.098      0.922     -51.700      46.959
np.log(gdp)    11.9971      3.301      3.635      0.001       5.236      18.758
==============================================================================
Omnibus:                       27.737   Durbin-Watson:                   2.064
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               62.978
Skew:                          -1.931   Prob(JB):                     2.11e-14
Kurtosis:                       8.956   Cond. No.                         56.3
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
"""
```

6 Project A

Using the data files loaded in notebook 4, or other data that you've located using a .csv or other, please do the following:

Start a new notebook and create the first cell as a markdown cell. Write your name as a header, and a paragraph of text that describes the dataset, where you found it, and what the different columns represent, i.e. what are your variables. 1. Slice the data by column 2. Slice the data by row using both `.loc` and `.iloc` 3. Use the `.groupby` method to create a grouped set of data 4. Create the following visualizations use the Seaborn Tutorial for help
(https://seaborn.pydata.org/tutorial/categorical.html)
- `distplot` - `boxplot` - `violin_plot` - `barplot`

Write a brief summary of any patterns noticed and differences between categorical distributions.

6.1 Additional API Examples

In our first project, we will use datasets obtained through web API's to write a nice report that includes visualizations, and reproducible code including data. Our options involve using the `NYCOpenData` portal API or the World Bank Climate Data API.

6.2 NYC Open Data

NYC OpenData

Below, we load a dataset from the NYC Open Data site. You can search for other datasets if you would like, or you may use the city's recent data on mathematics performance in grades 3 - 8. To begin, we

load the `requests` library, and enter the API Endpoint url from the site. This comes as a JSON or javascript file, so we need to use the `read_json` method to change this to a Pandas DataFrame.

```
In [1]: import requests

In [2]: math = requests.get('https://data.cityofnewyork.us/resource/uqrh-uk4g.json')

In [5]: math

Out[5]: <Response [200]>

In [7]: math.text[:300]

Out[7]: '[{"dbn":"01M015","demographic":"Asian","grade":"3","mean_scale_score":"s","num_level_1":"s","num_

In [8]: import pandas as pd

In [10]: math = pd.read_json(math.text)

In [11]: math.head()

Out[11]: dbn demographic grade mean_scale_score num_level_1 num_level_2  \
         0  01M015       Asian     3                s           s          s
         1  01M015       Black     3              662           0          3
         2  01M015    Hispanic     3              670           1          8
         3  01M015       Asian     3                s           s          s
         4  01M015       Black     3                s           s          s

           num_level_3 num_level_3_and_4 num_level_4  number_tested pct_level_1  \
         0           s                 s           s              3           s
         1           9                 9           0             12           0
         2          10                15           5             24         4.2
         3           s                 s           s              3           s
         4           s                 s           s              4           s

           pct_level_2 pct_level_3 pct_level_3_and_4 pct_level_4  year
         0           s           s                 s           s  2006
         1          25          75                75           0  2006
         2        33.3        41.7              62.5        20.8  2006
         3           s           s                 s           s  2007
         4           s           s                 s           s  2007
```

6.3 Climate Data

The World Bank has an API that allows access to a large amount of climate data. Here is a snippet from the documentation:

About the Climate Data API

> The Climate Data API provides programmatic access to most of the climate data used on the World Bank's Climate Change Knowledge Portal. Web developers can use this API to access the knowledge portal's data in real time to support their own applications, so long as they abide by the World Bank's Terms of Use.

```
In [12]: url = 'http://climatedataapi.worldbank.org/climateweb/rest/v1/country/cru/tas/year/CAN.csv'

In [13]: canada = requests.get(url)

In [18]: canada

Out[18]: <Response [200]>

In [19]: canada.text[:199]

Out[19]: 'year,data\n1901,-7.67241907119751\n1902,-7.862711429595947\n1903,-7.910782814025879\n1904,-8.155
```

```
In [25]: df = pd.read_(canada.text)

---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-25-009399ea74d6> in <module>()
----> 1 df = pd.read_table(canada.text)

~/anaconda3/lib/python3.6/site-packages/pandas/io/parsers.py in parser_f(filepath_or_buffer, sep, delimite
    653                     skip_blank_lines=skip_blank_lines)
    654
--> 655         return _read(filepath_or_buffer, kwds)
    656
    657     parser_f.__name__ = name

~/anaconda3/lib/python3.6/site-packages/pandas/io/parsers.py in _read(filepath_or_buffer, kwds)
    403
    404     # Create the parser.
--> 405     parser = TextFileReader(filepath_or_buffer, **kwds)
    406
    407     if chunksize or iterator:

~/anaconda3/lib/python3.6/site-packages/pandas/io/parsers.py in __init__(self, f, engine, **kwds)
    762             self.options['has_index_names'] = kwds['has_index_names']
    763
--> 764         self._make_engine(self.engine)
    765
    766     def close(self):

~/anaconda3/lib/python3.6/site-packages/pandas/io/parsers.py in _make_engine(self, engine)
    983     def _make_engine(self, engine='c'):
    984         if engine == 'c':
--> 985             self._engine = CParserWrapper(self.f, **self.options)
    986         else:
    987             if engine == 'python':

~/anaconda3/lib/python3.6/site-packages/pandas/io/parsers.py in __init__(self, src, **kwds)
   1603         kwds['allow_leading_cols'] = self.index_col is not False
   1604
-> 1605         self._reader = parsers.TextReader(src, **kwds)
   1606
   1607         # XXX

pandas/_libs/parsers.pyx in pandas._libs.parsers.TextReader.__cinit__ (pandas/_libs/parsers.c:4209)()

pandas/_libs/parsers.pyx in pandas._libs.parsers.TextReader._setup_parser_source (pandas/_libs/parsers.c:8

FileNotFoundError: File b'year,data\n1901,-7.67241907119751\n1902,-7.862711429595947\n1903,-7.910782814025
```

```
In [22]: df.head()
```

Out[22]:

	year	data
0	1901	-7.672419
1	1902	-7.862711
2	1903	-7.910783
3	1904	-8.155729
4	1905	-7.547311

```
In [26]: frame = pd.DataFrame(canada.text)

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-26-9d5b746d4789> in <module>()
----> 1 frame = pd.DataFrame(canada.text)
```

```
~/anaconda3/lib/python3.6/site-packages/pandas/core/frame.py in __init__(self, data, index, columns, dtype
    352                                         copy=False)
    353             else:
--> 354                 raise ValueError('DataFrame constructor not properly called!')
    355
    356         NDFrame.__init__(self, mgr, fastpath=True)

ValueError: DataFrame constructor not properly called!
```

```
In [29]: canada.text
```

```
Out[29]: 'year,data\n1901,-7.67241907119751\n1902,-7.862711429595947\n1903,-7.910782814025879\n1904,-8.155
```

6.4 Using the Documentation

Seems this is not so easy. Luckily, the climate data is also available as part of the `wbdata` package. Use the documentation to pull and analyze data related to Climate indicators, or a different choice using the documentation at: http://wbdata.readthedocs.io/en/latest/.

7 Introduction to Web Scraping

GOALS:

- Introduce structure of webpage
- Use requests to get website data
- Use Beautiful Soup to parse basic HTML page

```
In [20]: %%HTML
         <iframe width="560" height="315" src="https://www.youtube.com/embed/dFKwcFJHLhE?ecver=1" framebor
```

```
<IPython.core.display.HTML object>
```

7.1 What is a website

Behind every website is HTML code. This HTML code is accessible to you on the internet. If we navigate to a website that contains 50 interesting facts about Kanye West (http://www.boomsbeat.com/articles/2192/20140403/

50-interesting-facts-about-kanye-west-had-a-near-death-experience-in-2002-his-stylist-went-to-yale.htm), we can view the HTML behind it using the source code.

I'm using a macintosh computer and browsing with chrome. To get the source code I hit `control` and click on the page to see the page source option. Other browsers are similar. The result will be a new tab containing HTML code. Both are shown below.

7.2 HTML Tags

Tags are used to identify different objects on a website, and every tag has the same structure. For example, to write a paragraph on a webpage we would use the paragraph tags and put our text between the tags, as shown below.

```
<p>
This is where my text would go.
</p>
```

Here, the starts the paragraph and the ends the paragraph. Tags can be embedded within other tags. If we wanted to make a word bold and insert an image within the paragraph, we could write the following HTML code.

```
<p>
This is a <strong>heavy</strong> paragraph.  Here's a heavy picture.
<img src="images/heavy_pic.jpg"/img>
</p>
```

Also, tags may be given attributes. This may be used to apply a style using CSS. For example, the first fact about Kanye uses the `dir` attribute, and it was named `ltr`. This differentiates it from the opening paragraph that uses no attribute.

```
<div class="caption">Source: Flickr</div>
</div>
<p>Kanye West is a Grammy-winning rapper who is currently engaged to Kim Kardashian and he␣
↪is well known for his outrageous statements and for his broad musical palette.</p>
<ol>
<li dir="ltr">
<p dir="ltr">Kanye Omari West was born June 8, 1977 in Atlanta.</p>
```

We can use Python to pull the HTML of a webpage into a Jupyter notebook, and then use libraries with functions that know how to read HTML. We will use the attributes to further fine tune parsing the pieces of interest on the webpage.

7.3 Getting the HTML with Requests

The requests library can be used to fetch the HTML content of our website. We will assign the content of the webpage to a variable k. We can peek at this after, printing the first 400 characters of the request.

```
In [1]: import requests
        k = requests.get('http://www.boomsbeat.com/articles/2192/20140403/50-interesting-facts-about-kanye
```

```
In [2]: print(k.text[:400])
```

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>50 interesting facts about Kanye West: Had a near death-experience in 2002, his stylist went to Yal
<meta content="width=device-width" name="viewport">
```

```
<meta name="Keywords" content="Kanye West, Kanye West facts, Kanye West net worth, Kanye West full name" /
<meta name="Description" content="Kanye West is a
```

As we wanted, we have all the HTML content that we saw in our source view.

Parsing HTML with Beautiful Soup

Now, we will use the Beautiful Soup library to parse the HTML. Beautiful soup knows how to read the HTML and has many functions we can use to pull specific pieces of interest out. To begin, we turn our request object into a beautiful soup object named `soup`.

```
In [3]: from bs4 import BeautifulSoup
        soup = BeautifulSoup(k.text, 'html.parser')
```

Now, let us take a look at the source again and locate the structure surrounding the interesting facts. By searching on the source page for the first fact, I find the following.

```
alt="Source: Flickr" id="36455"
src="http://images.boomsbeat.com/data/images/full/36455/kanye-jpg.jpg" style="font-
family: Arial, Tahoma, Helvetica, sans-serif; font-size: 14px;" /></p>
<div class="imageNone">
<div class="caption">Source: Flickr</div>
</div>
<p>Kanye West is a Grammy-winning rapper who is currently engaged to Kim Kardashian an
he is well known for his outrageous statements and for his broad musical palette.</p>
<ol>
<li dir="ltr">
<p dir="ltr">Kanye Omari West was born June 8, 1977 in Atlanta.</p>
<!-- article_center_middle1_computer start-->
<div id='adunit_article_center_middle1_computer' class='adunit_rectangle'>
<div class='ad-sample advertisement_article_center_middle1_computer'>
<span>Advertisement</span></div>
<div id='article_center_middle1_computer'>
 <script type='text/javascript'>
  googletag.cmd.push(function() { googletag.display('article_center_middle1_computer');
```

Here, it's important to notice that the facts lie inside <p> paragraph tags. These tags also have an attribute `dir = "ltr"`. We can use beautiful soup to locate all these instances. If we are correct, we should have 50 interesting facts.

```
In [4]: facts = soup.find_all('p')

In [5]: len(facts)

Out[5]: 89

In [6]: facts[0]

Out[6]: <p class="art-date">Apr 03, 2014 11:57 AM EDT</p>

In [7]: facts[0].text

Out[7]: 'Apr 03, 2014 11:57 AM EDT'

In [8]: facts[2:53]

Out[8]: [<p>Kanye West is a Grammy-winning rapper who is currently engaged to Kim Kardashian and he is wel
         <p>Kanye Omari West was born June 8, 1977 in Atlanta.</p>,
         <p>His father Ray West was a black panther in the 60s and 70s and he later became one of the firs
         <p>The name Kanye means "the only one" in Swahilli.</p>,
         <p>Kanye lived in China for more than a year with his mother when he was in fifth grade. His moth
         <p>Kanye attended Chicago State University/Columbia College in Chicago.  He dropped out to pursue
         <p>Kanye's struggle to transition from producer to MC is well documented throughout his music. Ho
         <p>At the start of his music career, Kanye apparently kept his business dealings all in the famil
         <p>He sold his first beat to local Chicago rapper Gravity for $8,800.</p>,
         <p>He got his first big break through No I.D. (born Dion Ernest Wilson) is a veteran hip hop prod
         <p>No I.D.'s mother convinced him to meet this "energetic" kid, and the lessons paid off: "At fir
```

```
<p>He initially rose to fame as a producer for Roc-A-Fella Records. He is was influential on Jay-
<p>He dropped out of college and had a slew of random jobs. He worked as a telemarketer and sold
<p>Kanye was in a near fatal car accident while he was driving home from the studio in October 20
<p>While he was recovering in hospital, he didn't want to stop recording music so he asked for an
<p>He admits that the idea of becoming a male porn star crossed his mind once or twice before.</p
<p>His single debut is "Through the Wire" because he recorded it while he was still wearing the m
<p>Chaka Khan initially refused to grant Kanye permission to use the pitched-up sample of her voc
<p><iframe class="videocontent" height="480" src="http://www.youtube.com/embed/uvb-1wjAtk4" width
<p>He is a huge fan of Fiona Apple and her music. Yeezy told Apple she was "possibly [his] favori
<p>'College Dropout' was album of the year by almost every publication (New York Times, Time Maga
<p>West was the most nominated artist at the 47th Annual Grammy Awards with 10 nods, and he took
<p>Following the success of his The College Dropout album, he treated himself by purchasing an 18
<p>With the headline "Hip-Hop's Class Act," West becomes one of the rare entertainers to appear o
<p>He used the money from the "Diamonds from Sierra Leone" music video to raise awareness about b
<p>He caused controversy when he strayed  from his scripted monologue at the live televised Conce
<p><iframe class="videocontent" height="480" src="http://www.youtube.com/embed/zIUzLpO1kxI" width
<p>His nicknames include Ye, The Louis Vuitton Don, Yeezy or konman.</p>,
<p>Even after being named Best Hip-Hop Artist at the MTV Europe Music Awards in Copenhagen, a fum
<p><iframe class="videocontent" height="480" src="http://www.youtube.com/embed/YkwQbuAGLj4" width
<p>Kanye was named International Man of the Year by GQ in 2007 at a ceremony at Covent Garden's O
<p>Unfortunately, his mother died that same year following complications while getting plastic su
<p>Kanye says he realizes, "Nothing is promised in life except for death." and he lives everyday
<p>Kanye broke down at a concert in Paris, a week after the passing of his mother, Dr. Donda West
<p><iframe class="videocontent" height="480" src="http://www.youtube.com/embed/2ZXlnJ5o63g" width
<p>He launched an online travel company called "Kanye Travel Ventures" (KTV) through his official
<p>After the infamous  Taylor Swift Gate VMAs incident in 2009, he decided to leave the country f
<p><iframe class="videocontent" height="480" src="http://www.youtube.com/embed/UhL2LoYaZ90" width
<p>In addition to avoiding the VMAs backlash, 'Ye was able to slow down and spend time reflecting
<p>The Eternal Sunshine of the Spotless Mind director visited the studio on the same the day West
<p>He said once in an interview that he prefers finalizing a song in post production more than ha
<p>One of his favorite bands is  Scottish rock group Franz Ferdinand.</p>,
<p>The song, 'Stronger', famously used a sample of Daft Punk's 'Harder, Better, Faster, Stronger'
<p>Kanye was engaged to designer Alexis  Phifer for 18 months before he began a relationship Ambe
<p>For Kanye, being famous has always been an unbearable drain. In his new track 'New Slaves', he
<p>At one point in his career-circa the release of Graduation in 2007-Kanye was slated to star in
<p>He is a sensitive person at heart. An episode of South Park depicting Kanye as an egomaniac is
<p>Kanye and Royce have a long-standing feud stemming from a 2003 song that West produced for the
<p>Although 'Ye has a penchant for left field collaborations-most notably Chris Martin of Coldpla
<p>He is a budding fashion designer and he and he collaborated with French brand A.P.C. He garner
<p>Kanye opened up a burger chain in Chicago called Fatburger in 2008. When he opened it, he said
```

```
In [9]: facts[2].text
```

```
Out[9]: 'Kanye West is a Grammy-winning rapper who is currently engaged to Kim Kardashian and he is well k
```

```
In [10]: facts = facts[3:53]
```

Creating a Table of Facts

Now, we can create a table that contains each interesting fact. To do so, we will start with an empty list and append each interesting fact using our above syntax and a for loop.

```
In [11]: table = []
         for i in facts:
             fact = i.text
             table.append(fact)
```

```
In [12]: len(table)
```

```
Out[12]: 50
```

```
In [13]: table[:5]
```

```
Out[13]: ['Kanye Omari West was born June 8, 1977 in Atlanta.',
          'His father Ray West was a black panther in the 60s and 70s and he later became one of the first
          'The name Kanye means "the only one" in Swahilli.',
          'Kanye lived in China for more than a year with his mother when he was in fifth grade. His mothe
          'Kanye attended Chicago State University/Columbia College in Chicago.  He dropped out to pursue
```

Pandas and DataFrames

The standard library for data analysis in Python is Pandas. Here, the typical row and column format for data used is called a DataFrame. We can convert our table data to a dataframe as follows.

```
In [14]: import pandas as pd
         df = pd.DataFrame(table, columns=['Interesting Facts'])
```

We can use the `head()` function to examine the top 5 rows of our new DataFrame.

```
In [15]: df.head()
```

```
Out[15]: Interesting Facts
         0  Kanye Omari West was born June 8, 1977 in Atla...
         1  His father Ray West was a black panther in the...
         2   The name Kanye means "the only one" in Swahilli.
         3  Kanye lived in China for more than a year with...
         4  Kanye attended Chicago State University/Columb...
```

Save our Data

Now, we can convert the dataframe to a comma separated value file on our computer. We could read this back in at any time as shown with the `read_csv` file.

```
In [17]: df.to_csv('kanye_facts.csv', index=False, encoding='utf-8')
```

```
In [18]: df = pd.read_csv('kanye_facts.csv', encoding='utf-8')
```

```
In [19]: df.head(7)
```

```
Out[19]: Interesting Facts
         0  Kanye Omari West was born June 8, 1977 in Atla...
         1  His father Ray West was a black panther in the...
         2   The name Kanye means "the only one" in Swahilli.
         3  Kanye lived in China for more than a year with...
         4  Kanye attended Chicago State University/Columb...
         5  Kanye's struggle to transition from producer t...
         6  At the start of his music career, Kanye appare...
```

8 Scraping the Street

```
In [1]: %%HTML

        <iframe width="560" height="315" src="https://www.youtube.com/embed/5GcOXA_41MU" frameborder="0" a
```

```
<IPython.core.display.HTML object>
```

One of the most important television shows of all time was **21 Jump Street**. The show gave birth to stars like Johnny Depp, Richard Greico, and Holly Robinson Peete. The show also spoke to the youth of the late 80's and early 90's with a crew of undercover cops tackling law breakers.

8.1 Wikipedia List of Guest Stars

Wikipedia has a page containing information on the list of guest stars for five seasons of 21 Jump Street. Our goal is to create a table with the information on all the guest stars.

```
In [1]: l = [1, 2, 3, 4, 5]

In [2]: l[0]

Out[2]: 1

In [3]: l[:3]

Out[3]: [1, 2, 3]

In [4]: l[::2]

Out[4]: [1, 3, 5]

In [5]: %%HTML

        <h1>This a header</h1>
        <p>This is a paragraph</p>
        <p class = "special">This is a paragraph with an attribute</p>

<IPython.core.display.HTML object>

In [6]: import requests
        from bs4 import BeautifulSoup

In [7]: url = 'https://en.wikipedia.org/wiki/List_of_guest_stars_on_21_Jump_Street'

In [8]: page = requests.get(url)

In [9]: page

Out[9]: <Response [200]>

In [10]: soup = BeautifulSoup(page.text, 'html.parser')

In [11]: soup.title.text

Out[11]: 'List of guest stars on 21 Jump Street - Wikipedia'

In [12]: soup.title.string
```

```
Out[12]: 'List of guest stars on 21 Jump Street - Wikipedia'

In [13]: soup.a

Out[13]: <a id="top"></a>

In [14]: soup.div

Out[14]: <div class="noprint" id="mw-page-base"></div>

In [15]: soup.find_all('a')

Out[15]: [<a id="top"></a>,
          <a href="#mw-head">navigation</a>,
          <a href="#p-search">search</a>,
          <a class="mw-redirect" href="/wiki/Television_personality" title="Television personality">televi
          <a href="/wiki/Fox_Broadcasting_Company" title="Fox Broadcasting Company">Fox</a>,
          <a href="/wiki/21_Jump_Street" title="21 Jump Street">21 Jump Street</a>,
          <a href="#Season_1"><span class="tocnumber">1</span> <span class="toctext">Season 1</span></a>,
          <a href="#Season_2"><span class="tocnumber">2</span> <span class="toctext">Season 2</span></a>,
          <a href="#Season_3"><span class="tocnumber">3</span> <span class="toctext">Season 3</span></a>,
          <a href="#Season_4"><span class="tocnumber">4</span> <span class="toctext">Season 4</span></a>,
          <a href="#Season_5"><span class="tocnumber">5</span> <span class="toctext">Season 5</span></a>,
          <a href="#See_also"><span class="tocnumber">6</span> <span class="toctext">See also</span></a>,
          <a href="#References"><span class="tocnumber">7</span> <span class="toctext">References</span></
          <a href="/w/index.php?title=List_of_guest_stars_on_21_Jump_Street&amp;action=edit&amp;section=1"
          <a href="/wiki/Barney_Martin" title="Barney Martin">Barney Martin</a>,
          <a href="/wiki/Brandon_Douglas" title="Brandon Douglas">Brandon Douglas</a>,
          <a class="new" href="/w/index.php?title=Reginald_T._Dorsey&amp;action=edit&amp;redlink=1" title=
          <a href="/wiki/Billy_Jayne" title="Billy Jayne">Billy Jayne</a>,
          <a href="/wiki/Steve_Antin" title="Steve Antin">Steve Antin</a>,
          <a href="/wiki/Traci_Lind" title="Traci Lind">Traci Lind</a>,
          <a href="/wiki/Leah_Ayres" title="Leah Ayres">Leah Ayres</a>,
          <a href="/wiki/Geoffrey_Blake_(actor)" title="Geoffrey Blake (actor)">Geoffrey Blake</a>,
          <a href="/wiki/Josh_Brolin" title="Josh Brolin">Josh Brolin</a>,
          <a class="new" href="/w/index.php?title=Jamie_Bozian&amp;action=edit&amp;redlink=1" title="Jamie
          <a href="/wiki/John_D%27Aquino" title="John D'Aquino">John D'Aquino</a>,
          <a class="new" href="/w/index.php?title=Troy_Byer&amp;action=edit&amp;redlink=1" title="Troy Bye
          <a href="/wiki/Lezlie_Deane" title="Lezlie Deane">Lezlie Deane</a>,
          <a href="/wiki/Blair_Underwood" title="Blair Underwood">Blair Underwood</a>,
          <a href="/wiki/Robert_Picardo" title="Robert Picardo">Robert Picardo</a>,
          <a href="/wiki/Scott_Schwartz" title="Scott Schwartz">Scott Schwartz</a>,
          <a href="/wiki/Liane_Curtis" title="Liane Curtis">Liane Curtis</a>,
          <a href="/wiki/Byron_Thames" title="Byron Thames">Byron Thames</a>,
          <a href="/wiki/Sherilyn_Fenn" title="Sherilyn Fenn">Sherilyn Fenn</a>,
          <a href="/wiki/Christopher_Heyerdahl" title="Christopher Heyerdahl">Christopher Heyerdahl</a>,
          <a href="/wiki/Kurtwood_Smith" title="Kurtwood Smith">Kurtwood Smith</a>,
          <a href="/wiki/Sarah_G._Buxton" title="Sarah G. Buxton">Sarah G. Buxton</a>,
          <a href="/wiki/Jason_Priestley" title="Jason Priestley">Jason Priestley</a>,
          <a href="/w/index.php?title=List_of_guest_stars_on_21_Jump_Street&amp;action=edit&amp;section=2"
          <a href="/wiki/Kurtwood_Smith" title="Kurtwood Smith">Kurtwood Smith</a>,
          <a href="/wiki/Ray_Walston" title="Ray Walston">Ray Walston</a>,
          <a href="/wiki/Pauly_Shore" title="Pauly Shore">Pauly Shore</a>,
          <a href="/wiki/Shannon_Tweed" title="Shannon Tweed">Shannon Tweed</a>,
          <a href="/wiki/Lochlyn_Munro" title="Lochlyn Munro">Lochlyn Munro</a>,
          <a href="/wiki/Mindy_Cohn" title="Mindy Cohn">Mindy Cohn</a>,
          <a href="/wiki/Kent_McCord" title="Kent McCord">Kent McCord</a>,
          <a href="/wiki/Don_S._Davis" title="Don S. Davis">Don S. Davis</a>,
          <a class="mw-redirect" href="/wiki/Tom_Wright_(actor)" title="Tom Wright (actor)">Tom Wright</a>
          <a href="/wiki/Jean_Sagal" title="Jean Sagal">Jean Sagal</a>,
          <a href="/wiki/Liz_Sagal" title="Liz Sagal">Liz Sagal</a>,
          <a href="/wiki/Deborah_Lacey" title="Deborah Lacey">Deborah Lacey</a>,
          <a href="/wiki/Bradford_English" title="Bradford English">Bradford English</a>,
```

Christina Applegate,
Peter Berg,
Gabriel Jarret,
Bruce French,
Dann Florek,
Gregory Itzin,
Brad Pitt,
Don S. Davis,
Sam Anderson,
<a href="/w/index.php?title=List_of_guest_stars_on_21_Jump_Street&action=edit&section=3"
Leo Rossi,
Peri Gilpin,
Kelly Hu,
Russell Wong,
Christopher Titus,
Dom DeLuise,
<a class="new" href="/w/index.php?title=Kehli_O%27Byrne&action=edit&redlink=1" title="Ke
Larenz Tate,
Maia Brewton,
Michael DeLuise,
Mario Van Peebles,
*,
Bridget Fonda,
Conor O'Farrell,
Andrew Lauer,
<a class="new" href="/w/index.php?title=Claude_Brooks&action=edit&redlink=1" title="Clau
Margot Rose,
<a href="/w/index.php?title=List_of_guest_stars_on_21_Jump_Street&action=edit&section=4"
Don S. Davis,
Robert Romanus,
Rob Estes,
Stu Nahan,
Mario Van Peebles,
*,
Thomas Haden Church,
Billy Warlock,
Tony Plana,
Julie Warner,
Barbara Tarbuck,
Kamala Lopez,
Pamela Adlon,
Christine Elise,
Robyn Lively,
Vince Vaughn,
Mickey Jones,
Ray Baker,
Keith Coogan,
Shannen Doherty,
Wallace Langham,
Rosie Perez,
Don S. Davis,
Chick Hearn,
Kareem Abdul-Jabbar,
<a class="mw-redirect" href="/wiki/John_Waters_(filmmaker)" title="John Waters (filmmaker)">John
John Pyper-Ferguson,
Diedrich Bader,
Kelly Perine,
Kristin Dattilo,
<a href="/w/index.php?title=List_of_guest_stars_on_21_Jump_Street&action=edit&section=5"
Lisa Dean Ryan,

Scott Grimes,
Brigitta Dau,
Tony Dakota,
Perrey Reeves,
<a class="new" href="/w/index.php?title=Johannah_Newmarch&action=edit&redlink=1" title="
Richard Leacock,
<a class="new" href="/w/index.php?title=Pat_Bermel&action=edit&redlink=1" title="Pat Ber
Deanna Milligan,
Peter Outerbridge,
<a class="new" href="/w/index.php?title=Don_MacKay&action=edit&redlink=1" title="Don Mac
Terence Kelly,
Merrilyn Gann,
Ocean Hellman,
Lochlyn Munro,
Leslie Carlson,
David DeLuise,
Kamala Lopez,
Don S. Davis,
Jada Pinkett Smith,
^*,
<a href="/w/index.php?title=List_of_guest_stars_on_21_Jump_Street&action=edit&section=6"
Jump Street,
<a href="/w/index.php?title=List_of_guest_stars_on_21_Jump_Street&action=edit&section=7"
<a class="external text" href="http://www.imdb.com/title/tt0092312/" rel="nofollow"><i>21 Jump S
IMDb,
<a href="/wiki/Template:21_Jump_Street" title="Template:21 Jump Street"><abbr style=";;backgroun
<a href="/wiki/Template_talk:21_Jump_Street" title="Template talk:21 Jump Street"><abbr style=";
<a class="external text" href="//en.wikipedia.org/w/index.php?title=Template:21_Jump_Street&
Jump Street,
21 Jump Street,
<a href="/wiki/List_of_21_Jump_Street_episodes" title="List of 21 Jump Street episodes">episodes
guest stars,
Booker,
21 Jump Street,
22 Jump Street,
<a href="/wiki/22_Jump_Street_(Original_Motion_Picture_Score)" title="22 Jump Street (Original M
<a dir="ltr" href="https://en.wikipedia.org/w/index.php?title=List_of_guest_stars_on_21_Jump_Str
Categories,
21 Jump Street,
<a href="/wiki/Category:Lists_of_actors_by_role" title="Category:Lists of actors by role">Lists
<a href="/wiki/Category:Lists_of_American_television_series_characters" title="Category:Lists of
<a href="/wiki/Category:Lists_of_drama_television_characters" title="Category:Lists of drama tel
<a href="/wiki/Category:Lists_of_guest_appearances_in_television" title="Category:Lists of guest
<a accesskey="n" href="/wiki/Special:MyTalk" title="Discussion about edits from this IP address
<a accesskey="y" href="/wiki/Special:MyContributions" title="A list of edits made from this IP a
<a href="/w/index.php?title=Special:CreateAccount&returnto=List+of+guest+stars+on+21+Jump+St
<a accesskey="o" href="/w/index.php?title=Special:UserLogin&returnto=List+of+guest+stars+on+
<a accesskey="c" href="/wiki/List_of_guest_stars_on_21_Jump_Street" title="View the content page
<a accesskey="t" href="/wiki/Talk:List_of_guest_stars_on_21_Jump_Street" rel="discussion" title=
Read,
<a accesskey="e" href="/w/index.php?title=List_of_guest_stars_on_21_Jump_Street&action=edit"
<a accesskey="h" href="/w/index.php?title=List_of_guest_stars_on_21_Jump_Street&action=histo
,
Main page,
Contents,
<a href="/wiki/Portal:Featured_content" title="Featured content - the best of Wikipedia">Feature
<a href="/wiki/Portal:Current_events" title="Find background information on current events">Curr
<a accesskey="x" href="/wiki/Special:Random" title="Load a random article [x]">Random article</a
<a href="https://donate.wikimedia.org/wiki/Special:FundraiserRedirector?utm_source=donate&ut
Wikipedia store,

```
<a href="/wiki/Help:Contents" title="Guidance on how to use and edit Wikipedia">Help</a>,
<a href="/wiki/Wikipedia:About" title="Find out about Wikipedia">About Wikipedia</a>,
<a href="/wiki/Wikipedia:Community_portal" title="About the project, what you can do, where to f
<a accesskey="r" href="/wiki/Special:RecentChanges" title="A list of recent changes in the wiki
<a href="//en.wikipedia.org/wiki/Wikipedia:Contact_us" title="How to contact Wikipedia">Contact
<a accesskey="j" href="/wiki/Special:WhatLinksHere/List_of_guest_stars_on_21_Jump_Street" title=
<a accesskey="k" href="/wiki/Special:RecentChangesLinked/List_of_guest_stars_on_21_Jump_Street"
<a accesskey="u" href="/wiki/Wikipedia:File_Upload_Wizard" title="Upload files [u]">Upload file<
<a accesskey="q" href="/wiki/Special:SpecialPages" title="A list of all special pages [q]">Speci
<a href="/w/index.php?title=List_of_guest_stars_on_21_Jump_Street&amp;oldid=820109238" title="Pe
<a href="/w/index.php?title=List_of_guest_stars_on_21_Jump_Street&amp;action=info" title="More i
<a accesskey="g" href="https://www.wikidata.org/wiki/Special:EntityPage/Q6621947" title="Link to
<a href="/w/index.php?title=Special:CiteThisPage&amp;page=List_of_guest_stars_on_21_Jump_Street&
<a href="/w/index.php?title=Special:Book&amp;bookcmd=book_creator&amp;referer=List+of+guest+star
<a href="/w/index.php?title=Special:ElectronPdf&amp;page=List+of+guest+stars+on+21+Jump+Street&a
<a accesskey="p" href="/w/index.php?title=List_of_guest_stars_on_21_Jump_Street&amp;printable=ye
<a class="wbc-editpage" href="https://www.wikidata.org/wiki/Special:EntityPage/Q6621947#sitelink
<a href="//en.wikipedia.org/wiki/Wikipedia:Text_of_Creative_Commons_Attribution-ShareAlike_3.0_U
<a href="//creativecommons.org/licenses/by-sa/3.0/" rel="license" style="display:none;"></a>,
<a href="//wikimediafoundation.org/wiki/Terms_of_Use">Terms of Use</a>,
<a href="//wikimediafoundation.org/wiki/Privacy_policy">Privacy Policy</a>,
<a href="//www.wikimediafoundation.org/">Wikimedia Foundation, Inc.</a>,
<a class="extiw" href="https://wikimediafoundation.org/wiki/Privacy_policy" title="wmf:Privacy p
<a href="/wiki/Wikipedia:About" title="Wikipedia:About">About Wikipedia</a>,
<a href="/wiki/Wikipedia:General_disclaimer" title="Wikipedia:General disclaimer">Disclaimers</a
<a href="//en.wikipedia.org/wiki/Wikipedia:Contact_us">Contact Wikipedia</a>,
<a href="https://www.mediawiki.org/wiki/Special:MyLanguage/How_to_contribute">Developers</a>,
<a href="https://wikimediafoundation.org/wiki/Cookie_statement">Cookie statement</a>,
<a class="noprint stopMobileRedirectToggle" href="//en.m.wikipedia.org/w/index.php?title=List_of
<a href="https://wikimediafoundation.org/"><img alt="Wikimedia Foundation" height="31" src="/sta
<a href="//www.mediawiki.org/"><img alt="Powered by MediaWiki" height="31" src="/static/images/p
```

In [16]:
```
all_links = soup.find_all("a")
for link in all_links:
    print(link.get("href"))
```

```
None
#mw-head
#p-search
/wiki/Television_personality
/wiki/Fox_Broadcasting_Company
/wiki/21_Jump_Street
#Season_1
#Season_2
#Season_3
#Season_4
#Season_5
#See_also
#References
/w/index.php?title=List_of_guest_stars_on_21_Jump_Street&action=edit&section=1
/wiki/Barney_Martin
/wiki/Brandon_Douglas
/w/index.php?title=Reginald_T._Dorsey&action=edit&redlink=1
/wiki/Billy_Jayne
/wiki/Steve_Antin
/wiki/Traci_Lind
/wiki/Leah_Ayres
/wiki/Geoffrey_Blake_(actor)
/wiki/Josh_Brolin
/w/index.php?title=Jamie_Bozian&action=edit&redlink=1
/wiki/John_D%27Aquino
```

/w/index.php?title=Troy_Byer&action=edit&redlink=1
/wiki/Lezlie_Deane
/wiki/Blair_Underwood
/wiki/Robert_Picardo
/wiki/Scott_Schwartz
/wiki/Liane_Curtis
/wiki/Byron_Thames
/wiki/Sherilyn_Fenn
/wiki/Christopher_Heyerdahl
/wiki/Kurtwood_Smith
/wiki/Sarah_G._Buxton
/wiki/Jason_Priestley
/w/index.php?title=List_of_guest_stars_on_21_Jump_Street&action=edit&section=2
/wiki/Kurtwood_Smith
/wiki/Ray_Walston
/wiki/Pauly_Shore
/wiki/Shannon_Tweed
/wiki/Lochlyn_Munro
/wiki/Mindy_Cohn
/wiki/Kent_McCord
/wiki/Don_S._Davis
/wiki/Tom_Wright_(actor)
/wiki/Jean_Sagal
/wiki/Liz_Sagal
/wiki/Deborah_Lacey
/wiki/Bradford_English
/wiki/Christina_Applegate
/wiki/Peter_Berg
/wiki/Gabriel_Jarret
/wiki/Bruce_French_(actor)
/wiki/Dann_Florek
/wiki/Gregory_Itzin
/wiki/Brad_Pitt
/wiki/Don_S._Davis
/wiki/Sam_Anderson
/w/index.php?title=List_of_guest_stars_on_21_Jump_Street&action=edit&section=3
/wiki/Leo_Rossi
/wiki/Peri_Gilpin
/wiki/Kelly_Hu
/wiki/Russell_Wong
/wiki/Christopher_Titus
/wiki/Dom_DeLuise
/w/index.php?title=Kehli_O%27Byrne&action=edit&redlink=1
/wiki/Larenz_Tate
/wiki/Maia_Brewton
/wiki/Michael_DeLuise
/wiki/Mario_Van_Peebles
#endnote_reference_name_A
/wiki/Bridget_Fonda
/wiki/Conor_O%27Farrell
/wiki/Andrew_Lauer
/w/index.php?title=Claude_Brooks&action=edit&redlink=1
/wiki/Margot_Rose
/w/index.php?title=List_of_guest_stars_on_21_Jump_Street&action=edit&section=4
/wiki/Don_S._Davis
/wiki/Robert_Romanus
/wiki/Rob_Estes
/wiki/Stu_Nahan
/wiki/Mario_Van_Peebles
#endnote_reference_name_A

/wiki/Thomas_Haden_Church
/wiki/Billy_Warlock
/wiki/Tony_Plana
/wiki/Julie_Warner
/wiki/Barbara_Tarbuck
/wiki/Kamala_Lopez
/wiki/Pamela_Adlon
/wiki/Christine_Elise
/wiki/Robyn_Lively
/wiki/Vince_Vaughn
/wiki/Mickey_Jones
/wiki/Ray_Baker_(actor)
/wiki/Keith_Coogan
/wiki/Shannen_Doherty
/wiki/Wallace_Langham
/wiki/Rosie_Perez
/wiki/Don_S._Davis
/wiki/Chick_Hearn
/wiki/Kareem_Abdul-Jabbar
/wiki/John_Waters_(filmmaker)
/wiki/John_Pyper-Ferguson
/wiki/Diedrich_Bader
/wiki/Kelly_Perine
/wiki/Kristin_Dattilo
/w/index.php?title=List_of_guest_stars_on_21_Jump_Street&action=edit&section=5
/wiki/Lisa_Dean_Ryan
/wiki/Scott_Grimes
/wiki/Brigitta_Dau
/wiki/Tony_Dakota
/wiki/Perrey_Reeves
/w/index.php?title=Johannah_Newmarch&action=edit&redlink=1
/wiki/Richard_Leacock
/w/index.php?title=Pat_Bermel&action=edit&redlink=1
/wiki/Deanna_Milligan
/wiki/Peter_Outerbridge
/w/index.php?title=Don_MacKay&action=edit&redlink=1
/wiki/Terence_Kelly_(actor)
/wiki/Merrilyn_Gann
/wiki/Ocean_Hellman
/wiki/Lochlyn_Munro
/wiki/Leslie_Carlson
/wiki/David_DeLuise
/wiki/Kamala_Lopez
/wiki/Don_S._Davis
/wiki/Jada_Pinkett_Smith
#ref_reference_name_A
/w/index.php?title=List_of_guest_stars_on_21_Jump_Street&action=edit&section=6
/wiki/Jump_Street_(franchise)
/w/index.php?title=List_of_guest_stars_on_21_Jump_Street&action=edit&section=7
http://www.imdb.com/title/tt0092312/
/wiki/IMDb
/wiki/Template:21_Jump_Street
/wiki/Template_talk:21_Jump_Street
//en.wikipedia.org/w/index.php?title=Template:21_Jump_Street&action=edit
/wiki/Jump_Street_(franchise)
/wiki/21_Jump_Street
/wiki/List_of_21_Jump_Street_episodes
None
/wiki/Booker_(TV_series)
/wiki/21_Jump_Street_(film)

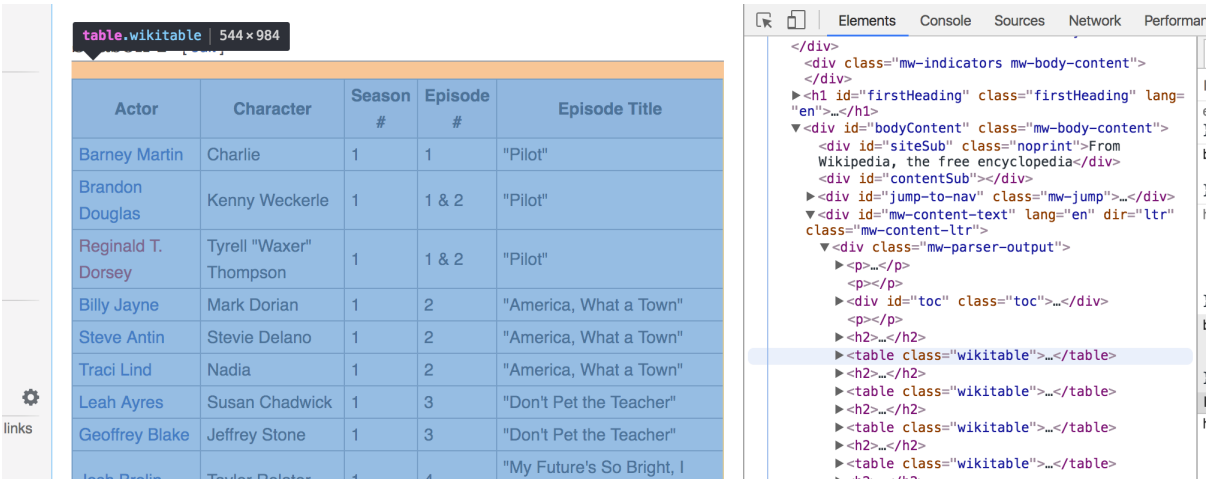
```
/wiki/22_Jump_Street
/wiki/22_Jump_Street_(Original_Motion_Picture_Score)
https://en.wikipedia.org/w/index.php?title=List_of_guest_stars_on_21_Jump_Street&oldid=820109238
/wiki/Help:Category
/wiki/Category:21_Jump_Street
/wiki/Category:Lists_of_actors_by_role
/wiki/Category:Lists_of_American_television_series_characters
/wiki/Category:Lists_of_drama_television_characters
/wiki/Category:Lists_of_guest_appearances_in_television
/wiki/Special:MyTalk
/wiki/Special:MyContributions
/w/index.php?title=Special:CreateAccount&returnto=List+of+guest+stars+on+21+Jump+Street
/w/index.php?title=Special:UserLogin&returnto=List+of+guest+stars+on+21+Jump+Street
/wiki/List_of_guest_stars_on_21_Jump_Street
/wiki/Talk:List_of_guest_stars_on_21_Jump_Street
/wiki/List_of_guest_stars_on_21_Jump_Street
/w/index.php?title=List_of_guest_stars_on_21_Jump_Street&action=edit
/w/index.php?title=List_of_guest_stars_on_21_Jump_Street&action=history
/wiki/Main_Page
/wiki/Main_Page
/wiki/Portal:Contents
/wiki/Portal:Featured_content
/wiki/Portal:Current_events
/wiki/Special:Random
https://donate.wikimedia.org/wiki/Special:FundraiserRedirector?utm_source=donate&utm_medium=sidebar&utm_ca
//shop.wikimedia.org
/wiki/Help:Contents
/wiki/Wikipedia:About
/wiki/Wikipedia:Community_portal
/wiki/Special:RecentChanges
//en.wikipedia.org/wiki/Wikipedia:Contact_us
/wiki/Special:WhatLinksHere/List_of_guest_stars_on_21_Jump_Street
/wiki/Special:RecentChangesLinked/List_of_guest_stars_on_21_Jump_Street
/wiki/Wikipedia:File_Upload_Wizard
/wiki/Special:SpecialPages
/w/index.php?title=List_of_guest_stars_on_21_Jump_Street&oldid=820109238
/w/index.php?title=List_of_guest_stars_on_21_Jump_Street&action=info
https://www.wikidata.org/wiki/Special:EntityPage/Q6621947
/w/index.php?title=Special:CiteThisPage&page=List_of_guest_stars_on_21_Jump_Street&id=820109238
/w/index.php?title=Special:Book&bookcmd=book_creator&referer=List+of+guest+stars+on+21+Jump+Street
/w/index.php?title=Special:ElectronPdf&page=List+of+guest+stars+on+21+Jump+Street&action=show-download-scr
/w/index.php?title=List_of_guest_stars_on_21_Jump_Street&printable=yes
https://www.wikidata.org/wiki/Special:EntityPage/Q6621947#sitelinks-wikipedia
//en.wikipedia.org/wiki/Wikipedia:Text_of_Creative_Commons_Attribution-ShareAlike_3.0_Unported_License
//creativecommons.org/licenses/by-sa/3.0/
//wikimediafoundation.org/wiki/Terms_of_Use
//wikimediafoundation.org/wiki/Privacy_policy
//www.wikimediafoundation.org/
https://wikimediafoundation.org/wiki/Privacy_policy
/wiki/Wikipedia:About
/wiki/Wikipedia:General_disclaimer
//en.wikipedia.org/wiki/Wikipedia:Contact_us
https://www.mediawiki.org/wiki/Special:MyLanguage/How_to_contribute
https://wikimediafoundation.org/wiki/Cookie_statement
//en.m.wikipedia.org/w/index.php?title=List_of_guest_stars_on_21_Jump_Street&mobileaction=toggle_view_mobi
https://wikimediafoundation.org/
//www.mediawiki.org/
```

```
In [18]: all_tables = soup.find_all('table')
```

```
In [19]: len(all_tables)
```

```
Out[19]: 6

In [20]: all_tables[0].text

Out[20]: '\n\nActor\nCharacter\nSeason #\nEpisode #\nEpisode Title\n\n\nBarney Martin\nCharlie\n1\n1\n"Pil
```

8.2 Using Attributes



Actor	Character	Season #	Episode #	Episode Title
Barney Martin	Charlie	1	1	"Pilot"
Brandon Douglas	Kenny Weckerle	1	1 & 2	"Pilot"
Reginald T. Dorsey	Tyrell "Waxer" Thompson	1	1 & 2	"Pilot"
Billy Jayne	Mark Dorian	1	2	"America, What a Town"
Steve Antin	Stevie Delano	1	2	"America, What a Town"
Traci Lind	Nadia	1	2	"America, What a Town"
Leah Ayres	Susan Chadwick	1	3	"Don't Pet the Teacher"
Geoffrey Blake	Jeffrey Stone	1	3	"Don't Pet the Teacher"
				"My Future's So Bright, I

```
In [21]: right_tables = soup.find_all('table', class_='wikitable')

In [22]: len(right_tables)

Out[22]: 5

In [23]: type(right_tables)

Out[23]: bs4.element.ResultSet

In [24]: right_tables[0]

Out[24]: <table class="wikitable">
         <tr>
         <th>Actor</th>
         <th>Character</th>
         <th>Season #</th>
         <th>Episode #</th>
         <th>Episode Title</th>
         </tr>
         <tr>
         <td><a href="/wiki/Barney_Martin" title="Barney Martin">Barney Martin</a></td>
         <td>Charlie</td>
         <td>1</td>
         <td>1</td>
         <td>"Pilot"</td>
         </tr>
         <tr>
         <td><a href="/wiki/Brandon_Douglas" title="Brandon Douglas">Brandon Douglas</a></td>
         <td>Kenny Weckerle</td>
         <td>1</td>
         <td>1 &amp;amp; 2</td>
         <td>"Pilot"</td>
         </tr>
         <tr>
         <td><a class="new" href="/w/index.php?title=Reginald_T._Dorsey&amp;action=edit&amp;redlink=1" tit
         <td>Tyrell "Waxer" Thompson</td>
         <td>1</td>
```

```
<td>1 &amp;amp; 2</td>
<td>"Pilot"</td>
</tr>
<tr>
<td><a href="/wiki/Billy_Jayne" title="Billy Jayne">Billy Jayne</a></td>
<td>Mark Dorian</td>
<td>1</td>
<td>2</td>
<td>"America, What a Town"</td>
</tr>
<tr>
<td><a href="/wiki/Steve_Antin" title="Steve Antin">Steve Antin</a></td>
<td>Stevie Delano</td>
<td>1</td>
<td>2</td>
<td>"America, What a Town"</td>
</tr>
<tr>
<td><a href="/wiki/Traci_Lind" title="Traci Lind">Traci Lind</a></td>
<td>Nadia</td>
<td>1</td>
<td>2</td>
<td>"America, What a Town"</td>
</tr>
<tr>
<td><a href="/wiki/Leah_Ayres" title="Leah Ayres">Leah Ayres</a></td>
<td>Susan Chadwick</td>
<td>1</td>
<td>3</td>
<td>"Don't Pet the Teacher"</td>
</tr>
<tr>
<td><a href="/wiki/Geoffrey_Blake_(actor)" title="Geoffrey Blake (actor)">Geoffrey Blake</a></td>
<td>Jeffrey Stone</td>
<td>1</td>
<td>3</td>
<td>"Don't Pet the Teacher"</td>
</tr>
<tr>
<td><a href="/wiki/Josh_Brolin" title="Josh Brolin">Josh Brolin</a></td>
<td>Taylor Rolator</td>
<td>1</td>
<td>4</td>
<td>"My Future's So Bright, I Gotta Wear Shades"</td>
</tr>
<tr>
<td><a class="new" href="/w/index.php?title=Jamie_Bozian&amp;amp;action=edit&amp;amp;redlink=1" title="Ja
<td>Kurt Niles</td>
<td>1</td>
<td>4</td>
<td>"My Future's So Bright, I Gotta Wear Shades"</td>
</tr>
<tr>
<td><a href="/wiki/John_D%27Aquino" title="John D'Aquino">John D'Aquino</a></td>
<td>Vinny Morgan</td>
<td>1</td>
<td>4</td>
<td>"My Future's So Bright, I Gotta Wear Shades"</td>
</tr>
<tr>
```

```
<td><a class="new" href="/w/index.php?title=Troy_Byer&amp;action=edit&amp;redlink=1" title="Troy
<td>Patty Blatcher</td>
<td>1</td>
<td>5</td>
<td>"The Worst Night of Your Life"</td>
</tr>
<tr>
<td><a href="/wiki/Lezlie_Deane" title="Lezlie Deane">Lezlie Deane</a></td>
<td>Jane Kinney</td>
<td>1</td>
<td>5</td>
<td>"The Worst Night of Your Life"</td>
</tr>
<tr>
<td><a href="/wiki/Blair_Underwood" title="Blair Underwood">Blair Underwood</a></td>
<td>Reginald Brooks</td>
<td>1</td>
<td>6</td>
<td>"Gotta Finish the Riff"</td>
</tr>
<tr>
<td><a href="/wiki/Robert_Picardo" title="Robert Picardo">Robert Picardo</a></td>
<td>Ralph Buckley</td>
<td>1</td>
<td>6</td>
<td>"Gotta Finish the Riff"</td>
</tr>
<tr>
<td><a href="/wiki/Scott_Schwartz" title="Scott Schwartz">Scott Schwartz</a></td>
<td>Jordan Simms</td>
<td>1</td>
<td>7</td>
<td>"Bad Influence"</td>
</tr>
<tr>
<td><a href="/wiki/Liane_Curtis" title="Liane Curtis">Liane Curtis</a></td>
<td>Lauren Carlson</td>
<td>1</td>
<td>7</td>
<td>"Bad Influence"</td>
</tr>
<tr>
<td><a href="/wiki/Byron_Thames" title="Byron Thames">Byron Thames</a></td>
<td>Dylan Taylor</td>
<td>1</td>
<td>7</td>
<td>"Bad Influence"</td>
</tr>
<tr>
<td><a href="/wiki/Sherilyn_Fenn" title="Sherilyn Fenn">Sherilyn Fenn</a></td>
<td>Diane Nelson</td>
<td>1</td>
<td>8</td>
<td>"Blindsided"</td>
</tr>
<tr>
<td><a href="/wiki/Christopher_Heyerdahl" title="Christopher Heyerdahl">Christopher Heyerdahl</a>
<td>Jake</td>
<td>1</td>
<td>9</td>
```

```
<td>"Next Generation"</td>
</tr>
<tr>
<td><a href="/wiki/Kurtwood_Smith" title="Kurtwood Smith">Kurtwood Smith</a></td>
<td>Spencer Phillips</td>
<td>1</td>
<td>10</td>
<td>"Low and Away"</td>
</tr>
<tr>
<td>David Raynr</td>
<td>Kipling "Kip" Fuller</td>
<td>1</td>
<td>11</td>
<td>"16 Blown to 35"</td>
</tr>
<tr>
<td><a href="/wiki/Sarah_G._Buxton" title="Sarah G. Buxton">Sarah G. Buxton</a></td>
<td>Katrina</td>
<td>1</td>
<td>11</td>
<td>"16 Blown to 35"</td>
</tr>
<tr>
<td><a href="/wiki/Jason_Priestley" title="Jason Priestley">Jason Priestley</a></td>
<td>Tober</td>
<td>1</td>
<td>12</td>
<td>"Mean Streets and Pastel Houses"</td>
</tr>
</table>
```

```
In [25]: right_tables[0].find_all('tr')[0].text
```

```
Out[25]: '\nActor\nCharacter\nSeason #\nEpisode #\nEpisode Title\n'
```

```
In [26]: right_tables[0].find_all('tr')[3].text
```

```
Out[26]: '\nReginald T. Dorsey\nTyrell "Waxer" Thompson\n1\n1 & 2\n"Pilot"\n'
```

```
In [27]: for row in right_tables[0].find_all('tr'):
             cells = row.find_all('td')
```

```
In [28]: cells
```

```
Out[28]: [<td><a href="/wiki/Jason_Priestley" title="Jason Priestley">Jason Priestley</a></td>,
          <td>Tober</td>,
          <td>1</td>,
          <td>12</td>,
          <td>"Mean Streets and Pastel Houses"</td>]
```

```
In [29]: for i in range(5):
             for row in right_tables[i].find_all('tr'):
                 cells = row.find_all('td')
```

```
In [30]: cells[0].text
```

```
Out[30]: 'Jada Pinkett Smith'
```

```
In [31]: cells[1].text
```

```
Out[31]: 'Nicole'
```

```
In [32]: right_tables[0].find_all('td')[0].text
```

```
Out[32]: 'Barney Martin'
```

```
In [33]: right_tables[0].find_all('td')[1].text

Out[33]: 'Charlie'

In [34]: right_tables[0].find_all('td')[2].text

Out[34]: '1'

In [35]: right_tables[0].find_all('td')[3].text

Out[35]: '1'

In [36]: right_tables[0].find_all('td')[4].text

Out[36]: '"Pilot"'

In [37]: right_tables[0].find_all('td')[5].text

Out[37]: 'Brandon Douglas'

In [38]: len(right_tables[0].find_all('td'))

Out[38]: 120

In [39]: len(right_tables[1].find_all('td'))

Out[39]: 135

In [40]: a = []
         for j in range(120):
             items = right_tables[0].find_all('td')[j].text
             a.append(items)

In [41]: b = []
         for j in range(135):
             items = right_tables[1].find_all('td')[j].text
             b.append(items)

In [42]: len(right_tables[2].find_all('td'))

Out[42]: 105

In [43]: c = []
         for j in range(len(right_tables[2].find_all('td'))):
             items = right_tables[2].find_all('td')[j].text
             c.append(items)

In [44]: d = []
         for j in range(len(right_tables[3].find_all('td'))):
             items = right_tables[3].find_all('td')[j].text
             d.append(items)

In [45]: e = []
         for j in range(len(right_tables[4].find_all('td'))):
             items = right_tables[4].find_all('td')[j].text
             e.append(items)

In [46]: a[-1], b[-1], c[-1], d[-1], e[-1]

Out[46]: ('"Mean Streets and Pastel Houses"',
          '"School\'s Out"',
          '"Loc\'d Out Part 2"',
          '"Blackout"',
          '"Homegirls"')

In [47]: a[130]

---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-47-b47504dfcc6c> in <module>()
----> 1 a[130]
```

```
IndexError: list index out of range
```

```
In [48]: a[131]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-48-8718084a62c9> in <module>()
----> 1 a[131]

IndexError: list index out of range
```

```
In [49]: a[:20]
```

```
Out[49]: ['Barney Martin',
          'Charlie',
          '1',
          '1',
          '"Pilot"',
          'Brandon Douglas',
          'Kenny Weckerle',
          '1',
          '1 & 2',
          '"Pilot"',
          'Reginald T. Dorsey',
          'Tyrell "Waxer" Thompson',
          '1',
          '1 & 2',
          '"Pilot"',
          'Billy Jayne',
          'Mark Dorian',
          '1',
          '2',
          '"America, What a Town"']
```

```
In [50]: a[::5]
```

```
Out[50]: ['Barney Martin',
          'Brandon Douglas',
          'Reginald T. Dorsey',
          'Billy Jayne',
          'Steve Antin',
          'Traci Lind',
          'Leah Ayres',
          'Geoffrey Blake',
          'Josh Brolin',
          'Jamie Bozian',
          "John D'Aquino",
          'Troy Byer',
          'Lezlie Deane',
          'Blair Underwood',
          'Robert Picardo',
          'Scott Schwartz',
          'Liane Curtis',
          'Byron Thames',
          'Sherilyn Fenn',
          'Christopher Heyerdahl',
          'Kurtwood Smith',
          'David Raynr',
          'Sarah G. Buxton',
          'Jason Priestley']
```

```
In [51]: actors = a[::5] + b[::5] + c[::5] + d[::5] + e[::5]
         character = a[1::5] + b[1::5] + c[1::5] + d[1::5] + e[1::5]
```

```
season = a[2::5] + b[2::5] + c[2::5] + d[2::5] + e[2::5]
episode = a[3::5] + b[3::5] + c[3::5] + d[3::5] + e[3::5]
title = a[4::5] + b[4::5] + c[4::5] + d[4::5] + e[4::5]
```

In [52]:
```
actors[:4]
```

Out[52]:
```
['Barney Martin', 'Brandon Douglas', 'Reginald T. Dorsey', 'Billy Jayne']
```

In [53]:
```
import pandas as pd
```

In [54]:
```
df = pd.DataFrame()
```

In [55]:
```
df['Actors'] = actors
df['Character'] = character
df['Season'] = season
df['Episode'] = episode
df['Title'] = title
```

In [56]:
```
df.head()
```

Out[56]:

	Actors	Character	Season	Episode	Title
0	Barney Martin	Charlie	1	1	"Pilot"
1	Brandon Douglas	Kenny Weckerle	1	1 & 2	"Pilot"
2	Reginald T. Dorsey	Tyrell "Waxer" Thompson	1	1 & 2	"Pilot"
3	Billy Jayne	Mark Dorian	1	2	"America, What a Town"
4	Steve Antin	Stevie Delano	1	2	"America, What a Town"

In [57]:
```
df.shape
```

Out[57]:
```
(129, 5)
```

In [58]:
```
df.to_csv('data/jumpstreet.csv')
```

9 Webscraping and Natural Language Processing

In [1]:
```
%%HTML

<iframe width="560" height="315" src="https://www.youtube.com/embed/q7AM9QjCRrI" frameborder="0" a
```

```
<IPython.core.display.HTML object>
```

9.1 Investigating texts from Project Gutenberg

9.2 List Review

```
In [1]: a = [i for i in ['Uncle', 'Stever', 'has', 'a', 'gun']]

In [2]: a

Out[2]: ['Uncle', 'Stever', 'has', 'a', 'gun']

In [3]: a[0]

Out[3]: 'Uncle'

In [4]: b = [i.lower() for i in a]

In [5]: b

Out[5]: ['uncle', 'stever', 'has', 'a', 'gun']
```

9.3 Scraping the Text

```
In [6]: %matplotlib inline
        import matplotlib.pyplot as plt
        import requests
        from bs4 import BeautifulSoup

In [8]: url = "http://www.gutenberg.org/files/15784/15784-0.txt"

In [9]: response = requests.get(url)

In [10]: type(response)

Out[10]: requests.models.Response

In [11]: response

Out[11]: <Response [200]>

In [12]: soup_dos = BeautifulSoup(response.content, "html.parser")

In [13]: len(soup_dos)

Out[13]: 1

In [14]: dos_text = soup_dos.get_text()

In [15]: type(dos_text)

Out[15]: str

In [16]: len(dos_text)

Out[16]: 550924

In [17]: dos_text[:100]

Out[17]: 'The Project Gutenberg EBook of The Chronology of Ancient Kingdoms Amended\r\nby Isaac Newton\r\n
```

9.4 Using Regular Expressions

Regular expressions are a way to parse text using symbols to represent different kinds of textual characters. For example, in the above sentence, notice that we have some symbols that are only there to impart formatting. If we want to remove these, and only have the textual pieces, we can use a regular expression to find only words.

```
In [18]: import re

In [19]: a = 'Who knew Johnny Depp was an undercover police officer (with Richard Greico)!'

In [20]: ds = 'd\w+'

In [21]: re.findall(ds, a)
```


WHENEVER I LEARN A NEW SKILL I CONCOCT ELABORATE FANTASY SCENARIOS WHERE IT LETS ME SAVE THE DAY.
OH NO! THE KILLER MUST HAVE FOLLOWED HER ON VACATION!
BUT TO FIND THEM WE'D HAVE TO SEARCH THROUGH 200 MB OF EMAILS LOOKING FOR SOMETHING FORMATTED LIKE AN ADDRESS!
IT'S HOPELESS!
EVERYBODY STAND BACK.
I KNOW REGULAR EXPRESSIONS.
PERL!
TAP TAP

```
Out[21]: ['dercover']

In [22]: ds = 'D\w+'

In [23]: re.findall(ds, a)

Out[23]: ['Depp']

In [24]: ds = '[dD]\w+'

In [25]: re.findall(ds, a)

Out[25]: ['Depp', 'dercover']

In [26]: words = re.findall('\w+', dos_text)

In [27]: words[:10]

Out[27]: ['The',
          'Project',
          'Gutenberg',
          'EBook',
          'of',
          'The',
          'Chronology',
          'of',
          'Ancient',
          'Kingdoms']
```

9.5 Tokenization

Turning the document into a collection of individual items – words.

```
In [28]: from nltk.tokenize import RegexpTokenizer

In [29]: tokenizer = RegexpTokenizer('\w+')

In [30]: tokens = tokenizer.tokenize(dos_text)

In [31]: tokens[:8]

Out[31]: ['The', 'Project', 'Gutenberg', 'EBook', 'of', 'The', 'Chronology', 'of']

In [32]: words = []
         for word in tokens:
             words.append(word.lower())

In [33]: words[:10]

Out[33]: ['the',
          'project',
          'gutenberg',
          'ebook',
          'of',
          'the',
          'chronology',
          'of',
          'ancient',
          'kingdoms']
```

9.6 Stopwords

```
In [34]: from nltk.corpus import stopwords

In [35]: set(stopwords.words('english'))
```

Out[35]: {'a',
 'about',
 'above',
 'after',
 'again',
 'against',
 'ain',
 'all',
 'am',
 'an',
 'and',
 'any',
 'are',
 'aren',
 "aren't",
 'as',
 'at',
 'be',
 'because',
 'been',
 'before',
 'being',
 'below',
 'between',
 'both',
 'but',
 'by',
 'can',
 'couldn',
 "couldn't",
 'd',
 'did',
 'didn',
 "didn't",
 'do',
 'does',
 'doesn',
 "doesn't",
 'doing',
 'don',
 "don't",
 'down',
 'during',
 'each',
 'few',
 'for',
 'from',
 'further',
 'had',
 'hadn',
 "hadn't",
 'has',
 'hasn',
 "hasn't",
 'have',
 'haven',
 "haven't",
 'having',
 'he',
 'her',

```
'here',
'hers',
'herself',
'him',
'himself',
'his',
'how',
'i',
'if',
'in',
'into',
'is',
'isn',
"isn't",
'it',
"it's",
'its',
'itself',
'just',
'll',
'm',
'ma',
'me',
'mightn',
"mightn't",
'more',
'most',
'mustn',
"mustn't",
'my',
'myself',
'needn',
"needn't",
'no',
'nor',
'not',
'now',
'o',
'of',
'off',
'on',
'once',
'only',
'or',
'other',
'our',
'ours',
'ourselves',
'out',
'over',
'own',
're',
's',
'same',
'shan',
"shan't",
'she',
"she's",
'should',
"should've",
```

```
 'shouldn',
 "shouldn't",
 'so',
 'some',
 'such',
 't',
 'than',
 'that',
 "that'll",
 'the',
 'their',
 'theirs',
 'them',
 'themselves',
 'then',
 'there',
 'these',
 'they',
 'this',
 'those',
 'through',
 'to',
 'too',
 'under',
 'until',
 'up',
 've',
 'very',
 'was',
 'wasn',
 "wasn't",
 'we',
 'were',
 'weren',
 "weren't",
 'what',
 'when',
 'where',
 'which',
 'while',
 'who',
 'whom',
 'why',
 'will',
 'with',
 'won',
 "won't",
 'wouldn',
 "wouldn't",
 'y',
 'you',
 "you'd",
 "you'll",
 "you're",
 "you've",
 'your',
 'yours',
 'yourself',
 'yourselves'}
```

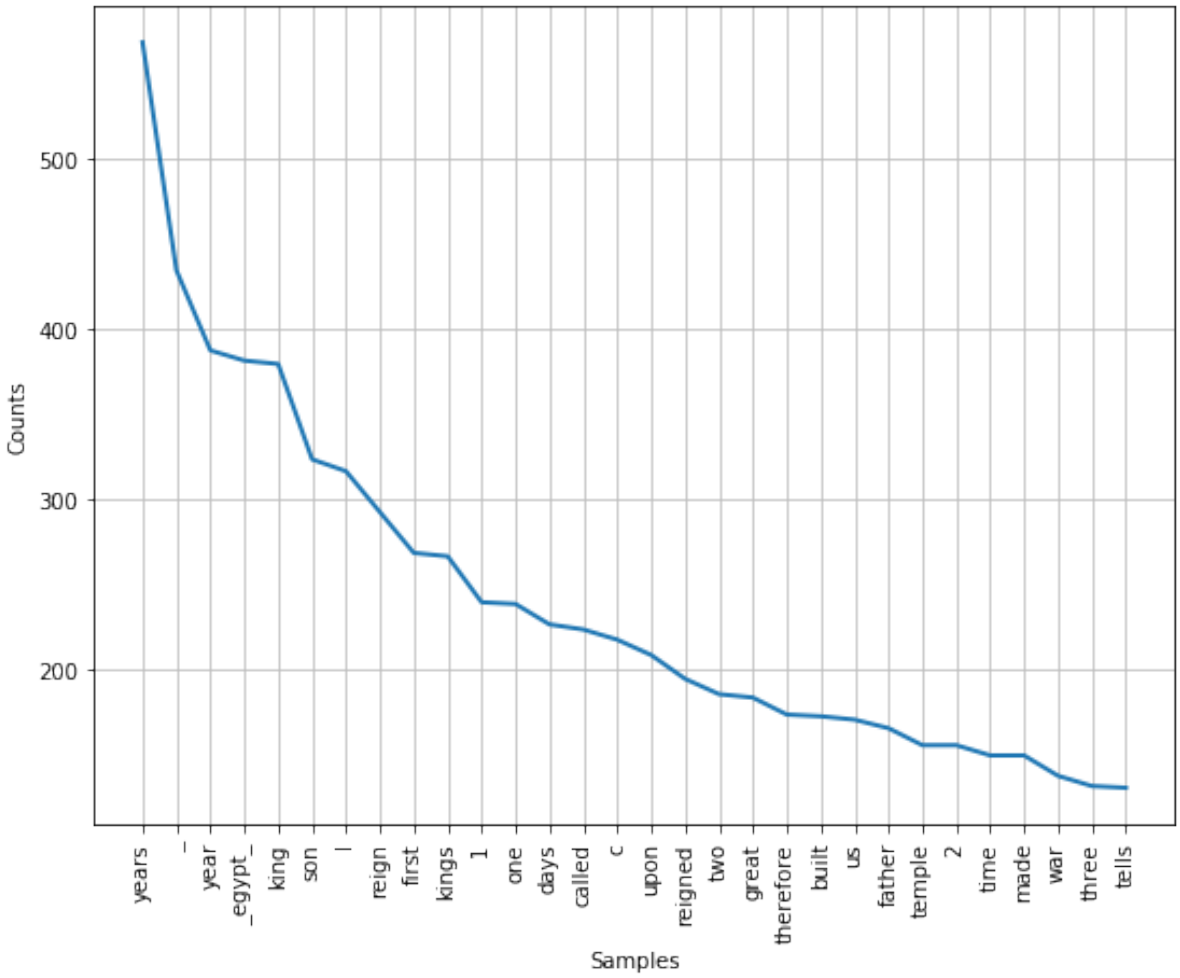
In [36]:
```
stop_words = set(stopwords.words('english'))
```

```
In [37]: filter_text = [word for word in words if not word in stop_words ]

In [38]: filter_text[:10]

Out[38]: ['project',
          'gutenberg',
          'ebook',
          'chronology',
          'ancient',
          'kingdoms',
          'amended',
          'isaac',
          'newton',
          'ebook']
```

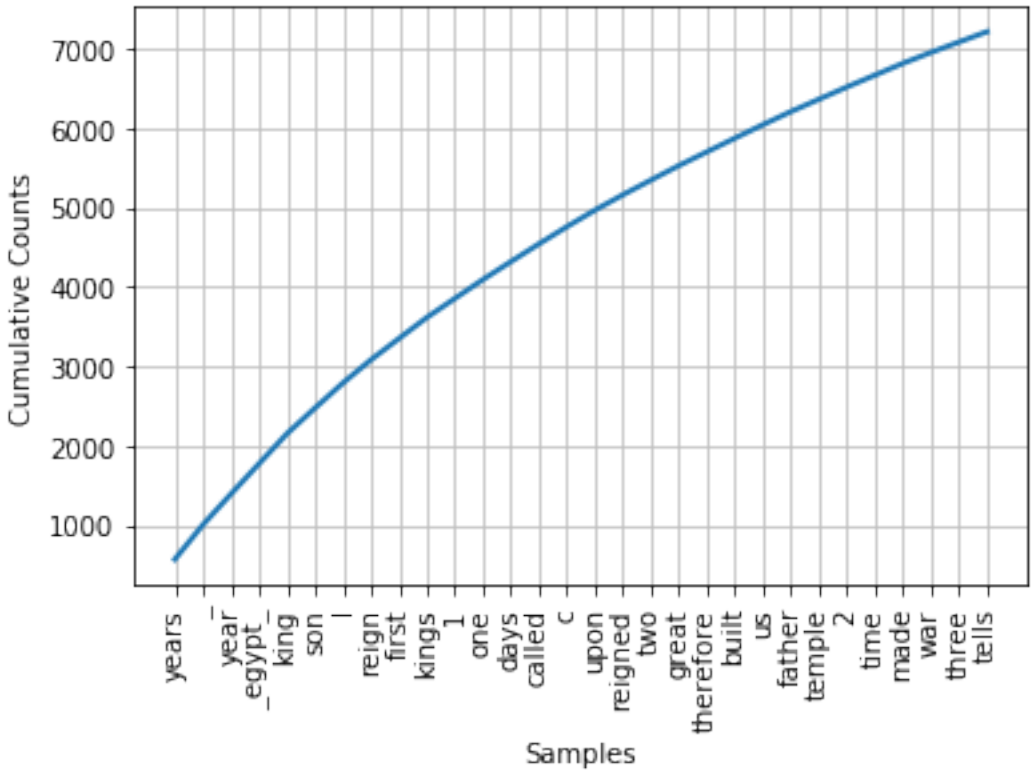
9.7 Analyzing the Text with NLTK

The Natural Language Toolkit is a popular Python library for text analysis. We will use it to split the text into individual words(tokens), and create a plot of the frequency distribution of the tokens.

```
In [39]: import nltk

In [40]: text = nltk.Text(filter_text)

In [41]: text[:10]

Out[41]: ['project',
          'gutenberg',
          'ebook',
          'chronology',
          'ancient',
          'kingdoms',
          'amended',
          'isaac',
          'newton',
          'ebook']

In [42]: fdist = nltk.FreqDist(text)

In [43]: type(fdist)

Out[43]: nltk.probability.FreqDist

In [44]: fdist.most_common(10)

Out[44]: [('years', 568),
          ('_', 434),
          ('year', 387),
          ('_egypt_', 381),
          ('king', 379),
          ('son', 323),
          ('1', 316),
          ('reign', 292),
          ('first', 268),
          ('kings', 266)]

In [45]: fdist['blood']

Out[45]: 5

In [46]: plt.figure(figsize = (9, 7))
         fdist.plot(30)
```

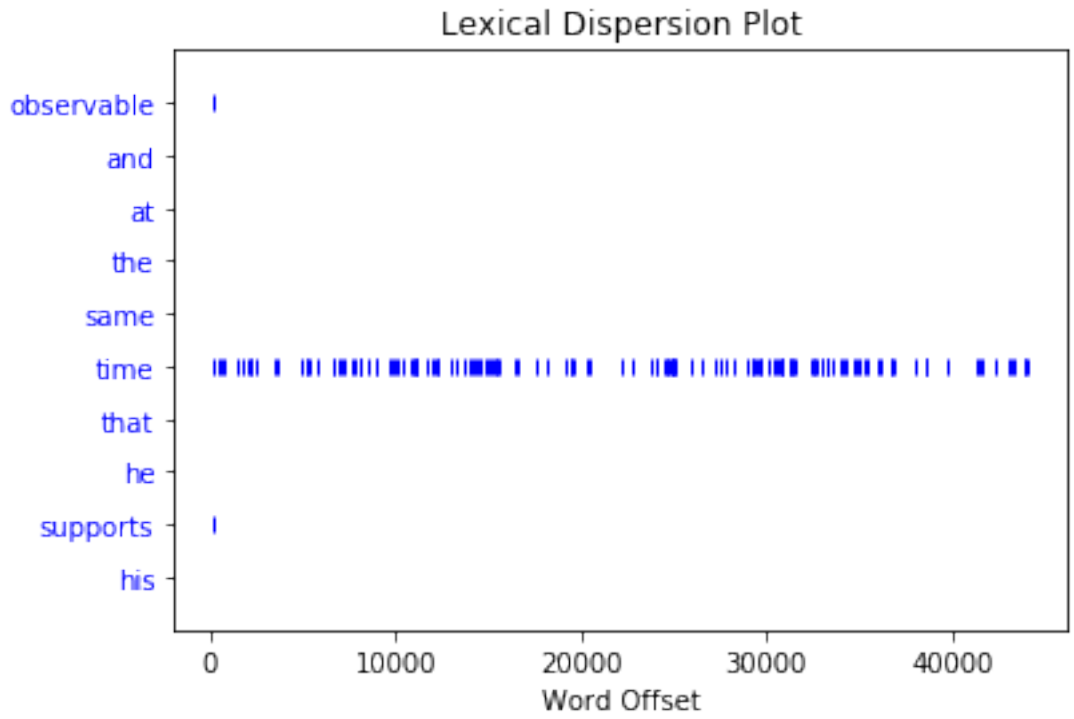


In [47]:
```
plt.figure()
fdist.plot(30, cumulative=True)
```



9.8 Part of Speech Tagging

```
In [48]: tagged = nltk.pos_tag(text)

In [49]: tagged[:10]

Out[49]: [('project', 'NN'),
          ('gutenberg', 'NN'),
          ('ebook', 'NN'),
          ('chronology', 'NN'),
          ('ancient', 'NN'),
          ('kingdoms', 'NNS'),
          ('amended', 'VBD'),
          ('isaac', 'JJ'),
          ('newton', 'NN'),
          ('ebook', 'NN')]

In [50]: text.similar("king")

reign son kings brother last one therefore year father according
called great years began war within man grandfather nabonass conquest

In [51]: text.common_contexts(["king", "brother"])

days_king son_father year_king kingdom_upon

In [52]: text.dispersion_plot(words[500:510])
```



9.9 Lexical Richness of Text

```
In [53]: len(text)
```

```
Out[53]: 49368
```

```
In [54]: len(set(text))/len(text)
```

```
Out[54]: 0.1888267703775725
```

```
In [55]: text.count("kings")
```

```
Out[55]: 266
```

```
In [56]: 100*text.count("kings")/len(text)
```

```
Out[56]: 0.5388105655485335
```

9.10 Long Words, Bigrams, Collacations

```
In [57]: long_words = [w for w in words if len(w)>10]
```

```
In [58]: long_words[:10]
```

```
Out[58]: ['restrictions',
          'distributed',
          'proofreading',
          '_alexander_',
          'encouragement',
          'extraordinary',
          'productions',
          '_chronology_',
          'demonstration',
          'judiciousness']
```

```
In [59]: list(nltk.bigrams(['more', 'is', 'said', 'than', 'done']))
```

```
Out[59]: [('more', 'is'), ('is', 'said'), ('said', 'than'), ('than', 'done')]
```

```
In [60]: text.collocations()
```

```
project gutenberg; _argonautic_ expedition; _red sea_; _anno nabonass;
_trojan_ war; year _nabonassar_; return _heraclides_; death _solomon_;
years piece; hundred years; one another; _darius hystaspis_; years
death; _heraclides_ _peloponnesus_; _alexander_ great; _assyrian_
empire; literary archive; high priest; _darius nothus_; _asia minor_
```

9.11 WordClouds

Another way to visualize text is using a wordcloud. I'll create a visualization using our earlier dataframe with guest stars on 21 Jump Street. We will visualize the titles with a wordcloud.

You may need to install wordcloud using

```
pip install wordcloud
```

```
In [61]: import pandas as pd
         from wordcloud import WordCloud, STOPWORDS

In [62]: df = pd.read_csv('data/jumpstreet.csv')

In [63]: df.head()
```

Out[63]:

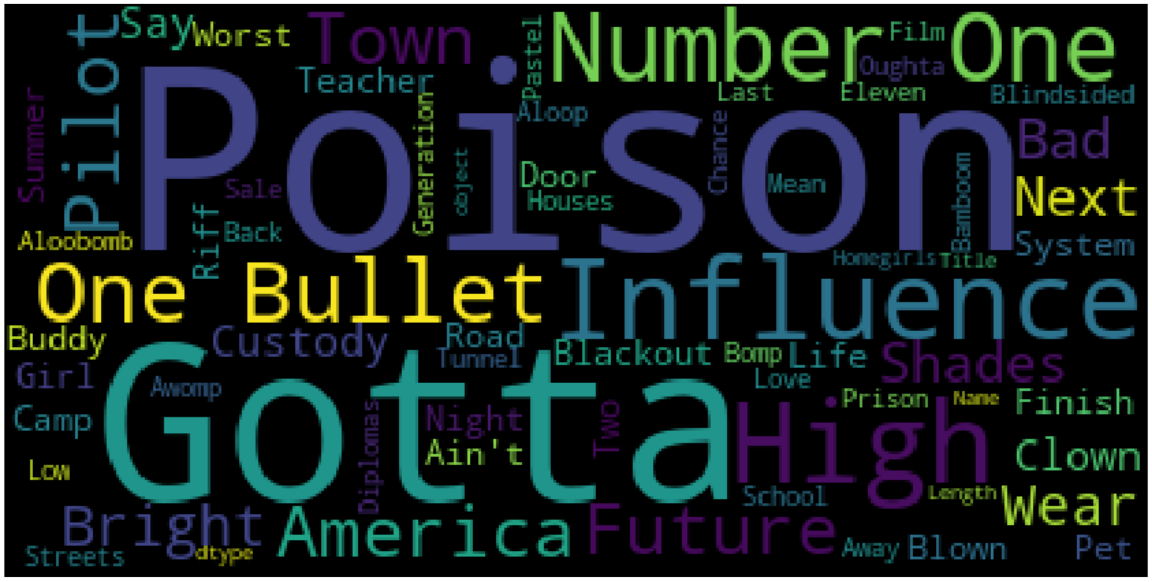
	Unnamed: 0	Actors	Character	Season	Episode	Title
0	0	Barney Martin	Charlie	1	1	"Pilot"
1	1	Brandon Douglas	Kenny Weckerle	1	1 & 2	"Pilot"
2	2	Reginald T. Dorsey	Tyrell "Waxer" Thompson	1	1 & 2	"Pilot"
3	3	Billy Jayne	Mark Dorian	1	2	"America, What a Town"
4	4	Steve Antin	Stevie Delano	1	2	"America, What a Town"

```
In [64]: wordcloud = WordCloud(background_color = 'black').generate(str(df['Title']))

In [65]: print(wordcloud)
```

```
<wordcloud.wordcloud.WordCloud object at 0x1a269cb4a8>
```

```
In [66]: plt.figure(figsize = (15, 23))
         plt.imshow(wordcloud)
         plt.axis('off')
```

```
Out[66]: (-0.5, 399.5, 199.5, -0.5)
```



9.12 Task

1. Scrape and tokenize a text from project Gutenberg.
2. Compare the most frequent occurring words with and without stopwords removed.
3. Examine the top bigrams. Create a barplot of the top 10 bigrams.
4. Create a wordcloud for the text.

Further Reading: http://www.nltk.org/book/

```
In [2]: from nltk.classify import NaiveBayesClassifier
        from nltk.corpus import subjectivity
        from nltk.sentiment import SentimentAnalyzer
        from nltk.sentiment.util import *
```

10 Introduction to Machine Learning

10.1 Sentiment Analysis with NLTK

http://www.nltk.org/api/nltk.sentiment.html

https://www.kaggle.com/ngyptr/python-nltk-sentiment-analysis

```
In [3]: n_instances = 100
```

```
In [4]: subj_docs = [(sent, 'subj') for sent in subjectivity.sents(categories='subj')[:n_instances]]
```

```
In [6]: obj_docs = [(sent, 'obj') for sent in subjectivity.sents(categories='obj')[:n_instances]]
```

```
In [7]: len(subj_docs), len(obj_docs)
```

```
Out[7]: (100, 100)
```

```
In [9]: subj_docs[1]
```

```
Out[9]: (['color',
          ',',
          'musical',
          'bounce',
```

```
          'and',
          'warm',
          'seas',
          'lapping',
          'on',
          'island',
          'shores',
          '.',
          'and',
          'just',
          'enough',
          'science',
          'to',
          'send',
          'you',
          'home',
          'thinking',
          '.'],
         'subj')
```

```
In [10]: train_subj_docs = subj_docs[:80]
         test_subj_docs = subj_docs[80:]
         train_obj_docs = obj_docs[:80]
         test_obj_docs = obj_docs[80:]
```

```
In [11]: train_docs = train_subj_docs + train_obj_docs
         test_docs = test_obj_docs + test_subj_docs
```

```
In [12]: clf = SentimentAnalyzer()
```

```
In [13]: all_words_neg = clf.all_words([mark_negation(doc) for doc in train_docs])
```

```
In [14]: unigram_features = clf.unigram_word_feats(all_words_neg, min_freq = 4)
```

```
In [15]: len(unigram_features)
```

```
Out[15]: 83
```

```
In [16]: clf.add_feat_extractor(extract_unigram_feats, unigrams = unigram_features)
```

```
In [17]: train_set = clf.apply_features(train_docs)
         test_set = clf.apply_features(test_docs)
```

```
In [18]: trainer = NaiveBayesClassifier.train
```

```
In [21]: classifier = clf.train(trainer, train_set)
```

```
Training classifier
```

```
In [23]: for key,value in sorted(clf.evaluate(test_set).items()):
             print('{0}: {1}'.format(key, value))
```

```
Evaluating NaiveBayesClassifier results...
Accuracy: 0.8
F-measure [obj]: 0.8
F-measure [subj]: 0.8
Precision [obj]: 0.8
Precision [subj]: 0.8
Recall [obj]: 0.8
Recall [subj]: 0.8
```

10.2 Basic Example

Below is a similar problem with some food review data.

```
In [40]: from nltk.tokenize import word_tokenize
```

```
In [41]: train = [("Great place to be when you are in Bangalore.", "pos"),
           ("The place was being renovated when I visited so the seating was limited.", "neg"),
           ("Loved the ambience, loved the food", "pos"),
           ("The food is delicious but not over the top.", "neg"),
           ("Service - Little slow, probably because too many people.", "neg"),
           ("The place is not easy to locate", "neg"),
           ("Mushroom fried rice was spicy", "pos"),
         ]

In [42]: dictionary = set(word.lower() for passage in train for word in word_tokenize(passage[0]))

In [43]: dictionary = set(word.lower() for passage in train for word in word_tokenize(passage[0]))

In [44]: t = [({word: (word in word_tokenize(x[0])) for word in dictionary}, x[1]) for x in train]

In [45]: classifier = nltk.NaiveBayesClassifier.train(t)

In [46]: test_data = "Manchurian was hot and spicy"
         test_data_features = {word.lower(): (word in word_tokenize(test_data.lower())) for word in dictio
         print (classifier.classify(test_data_features))
```

```
pos
```

10.3 Using Vader

There is an additional tool for sentiment analysis built in to nltk that includes another sentiment analysis analyzer.

```
In [24]: from nltk.sentiment.vader import SentimentIntensityAnalyzer

In [25]: paragraph = "It was one of the worst movies I've seen, despite good reviews. Unbelievably bad act


In [26]: from nltk import tokenize

In [27]: lines_list = tokenize.sent_tokenize(paragraph)

In [28]: lines_list

Out[28]: ["It was one of the worst movies I've seen, despite good reviews.",
          'Unbelievably bad acting!!',
          'Poor direction.',
          'VERY poor production.',
          'The movie was bad.',
          'Very bad movie.',
          'VERY bad movie.',
          'VERY BAD movie.',
          'VERY BAD movie!']

In [29]: sid = SentimentIntensityAnalyzer()
         for sent in lines_list:
             print(sent)
             ss = sid.polarity_scores(sent)
             for k in sorted(ss):
                 print('{0}: {1}, '.format(k, ss[k]), end = '')
             print()
```

```
It was one of the worst movies I've seen, despite good reviews.
compound: -0.7584, neg: 0.394, neu: 0.606, pos: 0.0,
Unbelievably bad acting!!
compound: -0.6572, neg: 0.686, neu: 0.314, pos: 0.0,
Poor direction.
compound: -0.4767, neg: 0.756, neu: 0.244, pos: 0.0,
VERY poor production.
compound: -0.6281, neg: 0.674, neu: 0.326, pos: 0.0,
The movie was bad.
```

```
compound: -0.5423, neg: 0.538, neu: 0.462, pos: 0.0,
Very bad movie.
compound: -0.5849, neg: 0.655, neu: 0.345, pos: 0.0,
VERY bad movie.
compound: -0.6732, neg: 0.694, neu: 0.306, pos: 0.0,
VERY BAD movie.
compound: -0.7398, neg: 0.724, neu: 0.276, pos: 0.0,
VERY BAD movie!
compound: -0.7616, neg: 0.735, neu: 0.265, pos: 0.0,
```

In [1]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
```

11 Intro to Machine Learning

One of the main ideas of machine learning, is to split data into testing and training sets. These sets are used to develop the model, and subsequently test its accuracy. Later, we will repeat this process a number of times to get an even better model. Machine learning can be thought of as representing a philosophy to model building, where we improve our models by iteratively building the model and testing it's performance on held out data.

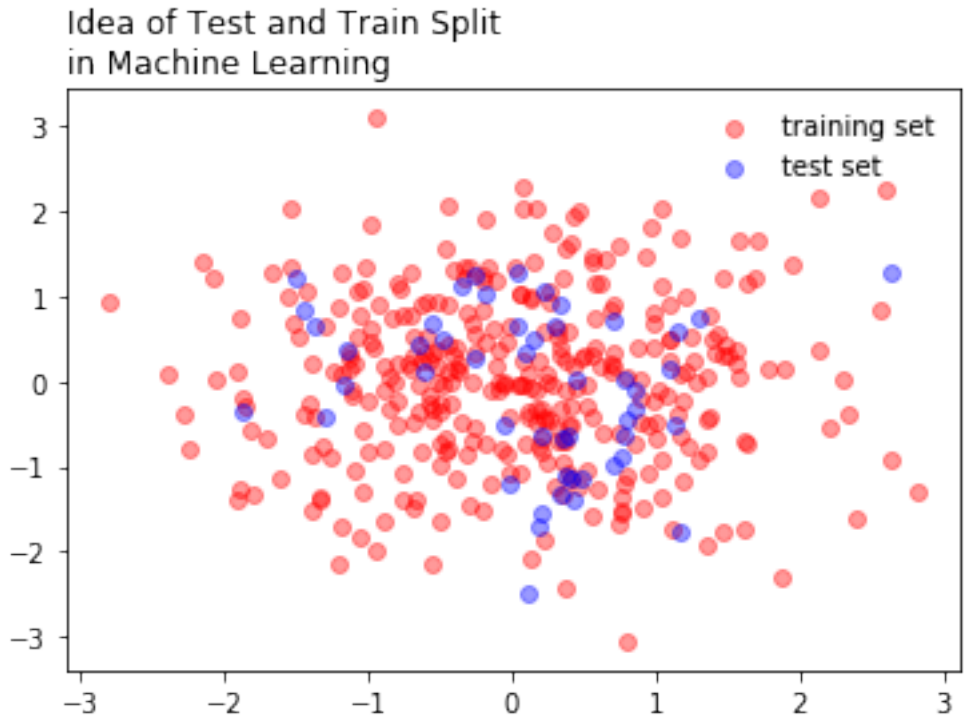
In [2]:
```
x = np.random.randn(400)
y = np.random.randn(400)
```

In [3]:
```
x.shape
```

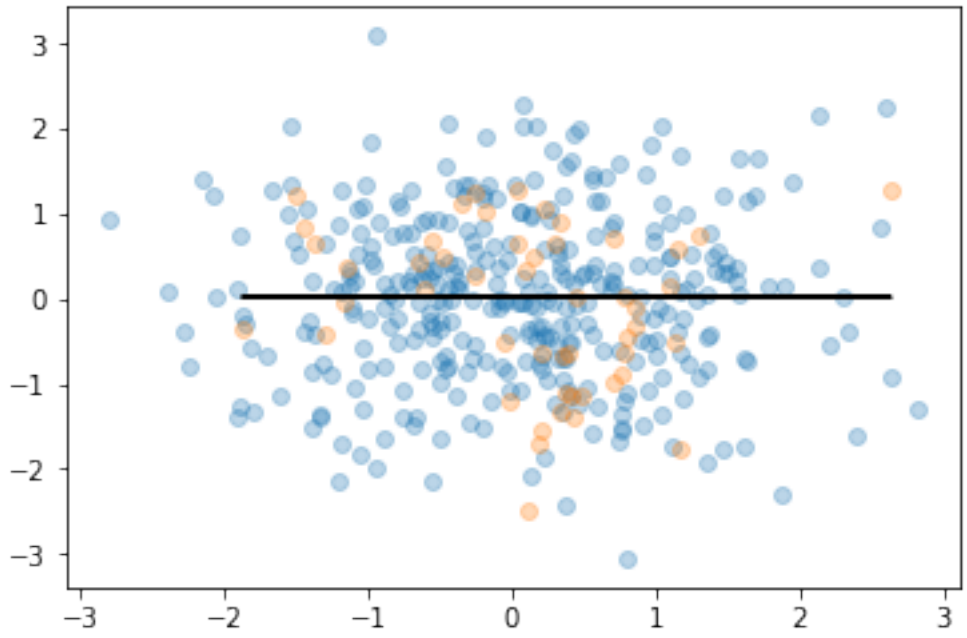
Out[3]:
```
(400,)
```

In [4]:
```
plt.scatter(x[:350], y[:350], color = 'red', alpha = 0.4, label = 'training set')
plt.scatter(x[350:], y[350:], color = 'blue', alpha = 0.4, label = 'test set')
plt.legend(loc = 'best', frameon = False)
plt.title("Idea of Test and Train Split \nin Machine Learning", loc = 'left')
```

Out[4]:
```
Text(0,1,'Idea of Test and Train Split \nin Machine Learning')
```



```
In [5]: X_train, x_test, y_train, y_test = x[:350].reshape(-1,1), x[350:].reshape(-1,1), y[:350].reshape(-

In [6]: X_train.shape

Out[6]: (350, 1)

In [7]: from sklearn import linear_model

In [8]: reg = linear_model.LinearRegression()
        reg.fit(X_train, y_train)

Out[8]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)

In [9]: reg.coef_

Out[9]: array([[-0.0010095]])

In [10]: y_predict = reg.predict(x_test.reshape(-1,1))

In [11]: plt.scatter(X_train, y_train, alpha = 0.3)
         plt.scatter(x_test, y_test, alpha = 0.3)
         plt.plot(x_test, y_predict, color = 'black')

Out[11]: [<matplotlib.lines.Line2D at 0x1a159ebcf8>]
```



11.1 Regression Example: Loading and Structuring Data

Predicting level of diabetes based on body mass index measures.

```
In [16]: %matplotlib inline
         import matplotlib.pyplot as plt
         import numpy as np
         from sklearn import datasets, linear_model
         from sklearn.metrics import mean_squared_error, r2_score

In [17]: diabetes = datasets.load_diabetes()

In [18]: diabetes

Out[18]: {'DESCR': 'Diabetes dataset\n================\n\nNotes\n-----\n\nTen baseline variables, age, sex
          'data': array([[ 0.03807591,  0.05068012,  0.06169621, ..., -0.00259226,
                   0.01990842, -0.01764613],
                 [-0.00188202, -0.04464164, -0.05147406, ..., -0.03949338,
```

```
        -0.06832974, -0.09220405],
       [ 0.08529891,  0.05068012,  0.04445121, ..., -0.00259226,
         0.00286377, -0.02593034],
       ...,
       [ 0.04170844,  0.05068012, -0.01590626, ..., -0.01107952,
        -0.04687948,  0.01549073],
       [-0.04547248, -0.04464164,  0.03906215, ...,  0.02655962,
         0.04452837, -0.02593034],
       [-0.04547248, -0.04464164, -0.0730303 , ..., -0.03949338,
        -0.00421986,  0.00306441]]),
'feature_names': ['age',
 'sex',
 'bmi',
 'bp',
 's1',
 's2',
 's3',
 's4',
 's5',
 's6'],
'target': array([ 151.,   75.,  141.,  206.,  135.,   97.,  138.,   63.,  110.,
        310.,  101.,   69.,  179.,  185.,  118.,  171.,  166.,  144.,
         97.,  168.,   68.,   49.,   68.,  245.,  184.,  202.,  137.,
         85.,  131.,  283.,  129.,   59.,  341.,   87.,   65.,  102.,
        265.,  276.,  252.,   90.,  100.,   55.,   61.,   92.,  259.,
         53.,  190.,  142.,   75.,  142.,  155.,  225.,   59.,  104.,
        182.,  128.,   52.,   37.,  170.,  170.,   61.,  144.,   52.,
        128.,   71.,  163.,  150.,   97.,  160.,  178.,   48.,  270.,
        202.,  111.,   85.,   42.,  170.,  200.,  252.,  113.,  143.,
         51.,   52.,  210.,   65.,  141.,   55.,  134.,   42.,  111.,
         98.,  164.,   48.,   96.,   90.,  162.,  150.,  279.,   92.,
         83.,  128.,  102.,  302.,  198.,   95.,   53.,  134.,  144.,
        232.,   81.,  104.,   59.,  246.,  297.,  258.,  229.,  275.,
        281.,  179.,  200.,  200.,  173.,  180.,   84.,  121.,  161.,
         99.,  109.,  115.,  268.,  274.,  158.,  107.,   83.,  103.,
        272.,   85.,  280.,  336.,  281.,  118.,  317.,  235.,   60.,
        174.,  259.,  178.,  128.,   96.,  126.,  288.,   88.,  292.,
         71.,  197.,  186.,   25.,   84.,   96.,  195.,   53.,  217.,
        172.,  131.,  214.,   59.,   70.,  220.,  268.,  152.,   47.,
         74.,  295.,  101.,  151.,  127.,  237.,  225.,   81.,  151.,
        107.,   64.,  138.,  185.,  265.,  101.,  137.,  143.,  141.,
         79.,  292.,  178.,   91.,  116.,   86.,  122.,   72.,  129.,
        142.,   90.,  158.,   39.,  196.,  222.,  277.,   99.,  196.,
        202.,  155.,   77.,  191.,   70.,   73.,   49.,   65.,  263.,
        248.,  296.,  214.,  185.,   78.,   93.,  252.,  150.,   77.,
        208.,   77.,  108.,  160.,   53.,  220.,  154.,  259.,   90.,
        246.,  124.,   67.,   72.,  257.,  262.,  275.,  177.,   71.,
         47.,  187.,  125.,   78.,   51.,  258.,  215.,  303.,  243.,
         91.,  150.,  310.,  153.,  346.,   63.,   89.,   50.,   39.,
        103.,  308.,  116.,  145.,   74.,   45.,  115.,  264.,   87.,
        202.,  127.,  182.,  241.,   66.,   94.,  283.,   64.,  102.,
        200.,  265.,   94.,  230.,  181.,  156.,  233.,   60.,  219.,
         80.,   68.,  332.,  248.,   84.,  200.,   55.,   85.,   89.,
         31.,  129.,   83.,  275.,   65.,  198.,  236.,  253.,  124.,
         44.,  172.,  114.,  142.,  109.,  180.,  144.,  163.,  147.,
         97.,  220.,  190.,  109.,  191.,  122.,  230.,  242.,  248.,
        249.,  192.,  131.,  237.,   78.,  135.,  244.,  199.,  270.,
        164.,   72.,   96.,  306.,   91.,  214.,   95.,  216.,  263.,
        178.,  113.,  200.,  139.,  139.,   88.,  148.,   88.,  243.,
         71.,   77.,  109.,  272.,   60.,   54.,  221.,   90.,  311.,
```

```
        281.,  182.,  321.,   58.,  262.,  206.,  233.,  242.,  123.,
        167.,   63.,  197.,   71.,  168.,  140.,  217.,  121.,  235.,
        245.,   40.,   52.,  104.,  132.,   88.,   69.,  219.,   72.,
        201.,  110.,   51.,  277.,   63.,  118.,   69.,  273.,  258.,
         43.,  198.,  242.,  232.,  175.,   93.,  168.,  275.,  293.,
        281.,   72.,  140.,  189.,  181.,  209.,  136.,  261.,  113.,
        131.,  174.,  257.,   55.,   84.,   42.,  146.,  212.,  233.,
         91.,  111.,  152.,  120.,   67.,  310.,   94.,  183.,   66.,
        173.,   72.,   49.,   64.,   48.,  178.,  104.,  132.,  220.,   57.])}
```

In [19]:
```
diabetes.DESCR
```

Out[19]:
```
'Diabetes dataset\n================\n\nNotes\n-----\n\nTen baseline variables, age, sex, body mas
```

In [20]:
```
diabetes.data
```

Out[20]:
```
array([[ 0.03807591,  0.05068012,  0.06169621, ..., -0.00259226,
         0.01990842, -0.01764613],
       [-0.00188202, -0.04464164, -0.05147406, ..., -0.03949338,
        -0.06832974, -0.09220405],
       [ 0.08529891,  0.05068012,  0.04445121, ..., -0.00259226,
         0.00286377, -0.02593034],
       ...,
       [ 0.04170844,  0.05068012, -0.01590626, ..., -0.01107952,
        -0.04687948,  0.01549073],
       [-0.04547248, -0.04464164,  0.03906215, ...,  0.02655962,
         0.04452837, -0.02593034],
       [-0.04547248, -0.04464164, -0.0730303 , ..., -0.03949338,
        -0.00421986,  0.00306441]])
```

In [34]:
```
diabetes.feature_names[2]
```

Out[34]:
```
'bmi'
```

In [21]:
```
diabetes.data[:, np.newaxis, 2]
```

Out[21]:
```
array([[ 0.06169621],
       [-0.05147406],
       [ 0.04445121],
       [-0.01159501],
       [-0.03638469],
       [-0.04069594],
       [-0.04716281],
       [-0.00189471],
       [ 0.06169621],
       [ 0.03906215],
       [-0.08380842],
       [ 0.01750591],
       [-0.02884001],
       [-0.00189471],
       [-0.02560657],
       [-0.01806189],
       [ 0.04229559],
       [ 0.01211685],
       [-0.0105172 ],
       [-0.01806189],
       [-0.05686312],
       [-0.02237314],
       [-0.00405033],
       [ 0.06061839],
       [ 0.03582872],
       [-0.01267283],
       [-0.07734155],
       [ 0.05954058],
```

```
[-0.02129532],
[-0.00620595],
[ 0.04445121],
[-0.06548562],
[ 0.12528712],
[-0.05039625],
[-0.06332999],
[-0.03099563],
[ 0.02289497],
[ 0.01103904],
[ 0.07139652],
[ 0.01427248],
[-0.00836158],
[-0.06764124],
[-0.0105172 ],
[-0.02345095],
[ 0.06816308],
[-0.03530688],
[-0.01159501],
[-0.0730303 ],
[-0.04177375],
[ 0.01427248],
[-0.00728377],
[ 0.0164281 ],
[-0.00943939],
[-0.01590626],
[ 0.0250506 ],
[-0.04931844],
[ 0.04121778],
[-0.06332999],
[-0.06440781],
[-0.02560657],
[-0.00405033],
[ 0.00457217],
[-0.00728377],
[-0.0374625 ],
[-0.02560657],
[-0.02452876],
[-0.01806189],
[-0.01482845],
[-0.02991782],
[-0.046085  ],
[-0.06979687],
[ 0.03367309],
[-0.00405033],
[-0.02021751],
[ 0.00241654],
[-0.03099563],
[ 0.02828403],
[-0.03638469],
[-0.05794093],
[-0.0374625 ],
[ 0.01211685],
[-0.02237314],
[-0.03530688],
[ 0.00996123],
[-0.03961813],
[ 0.07139652],
[-0.07518593],
[-0.00620595],
```

```
[-0.04069594],
[-0.04824063],
[-0.02560657],
[ 0.0519959 ],
[ 0.00457217],
[-0.06440781],
[-0.01698407],
[-0.05794093],
[ 0.00996123],
[ 0.08864151],
[-0.00512814],
[-0.06440781],
[ 0.01750591],
[-0.04500719],
[ 0.02828403],
[ 0.04121778],
[ 0.06492964],
[-0.03207344],
[-0.07626374],
[ 0.04984027],
[ 0.04552903],
[-0.00943939],
[-0.03207344],
[ 0.00457217],
[ 0.02073935],
[ 0.01427248],
[ 0.11019775],
[ 0.00133873],
[ 0.05846277],
[-0.02129532],
[-0.0105172 ],
[-0.04716281],
[ 0.00457217],
[ 0.01750591],
[ 0.08109682],
[ 0.0347509 ],
[ 0.02397278],
[-0.00836158],
[-0.06117437],
[-0.00189471],
[-0.06225218],
[ 0.0164281 ],
[ 0.09618619],
[-0.06979687],
[-0.02129532],
[-0.05362969],
[ 0.0433734 ],
[ 0.05630715],
[-0.0816528 ],
[ 0.04984027],
[ 0.11127556],
[ 0.06169621],
[ 0.01427248],
[ 0.04768465],
[ 0.01211685],
[ 0.00564998],
[ 0.04660684],
[ 0.12852056],
[ 0.05954058],
[ 0.09295276],
```

```
[ 0.01535029],
[-0.00512814],
[ 0.0703187 ],
[-0.00405033],
[-0.00081689],
[-0.04392938],
[ 0.02073935],
[ 0.06061839],
[-0.0105172 ],
[-0.03315126],
[-0.06548562],
[ 0.0433734 ],
[-0.06225218],
[ 0.06385183],
[ 0.03043966],
[ 0.07247433],
[-0.0191397 ],
[-0.06656343],
[-0.06009656],
[ 0.06924089],
[ 0.05954058],
[-0.02668438],
[-0.02021751],
[-0.046085  ],
[ 0.07139652],
[-0.07949718],
[ 0.00996123],
[-0.03854032],
[ 0.01966154],
[ 0.02720622],
[-0.00836158],
[-0.01590626],
[ 0.00457217],
[-0.04285156],
[ 0.00564998],
[-0.03530688],
[ 0.02397278],
[-0.01806189],
[ 0.04229559],
[-0.0547075 ],
[-0.00297252],
[-0.06656343],
[-0.01267283],
[-0.04177375],
[-0.03099563],
[-0.00512814],
[-0.05901875],
[ 0.0250506 ],
[-0.046085  ],
[ 0.00349435],
[ 0.05415152],
[-0.04500719],
[-0.05794093],
[-0.05578531],
[ 0.00133873],
[ 0.03043966],
[ 0.00672779],
[ 0.04660684],
[ 0.02612841],
[ 0.04552903],
```

```
[ 0.04013997],
[-0.01806189],
[ 0.01427248],
[ 0.03690653],
[ 0.00349435],
[-0.07087468],
[-0.03315126],
[ 0.09403057],
[ 0.03582872],
[ 0.03151747],
[-0.06548562],
[-0.04177375],
[-0.03961813],
[-0.03854032],
[-0.02560657],
[-0.02345095],
[-0.06656343],
[ 0.03259528],
[-0.046085  ],
[-0.02991782],
[-0.01267283],
[-0.01590626],
[ 0.07139652],
[-0.03099563],
[ 0.00026092],
[ 0.03690653],
[ 0.03906215],
[-0.01482845],
[ 0.00672779],
[-0.06871905],
[-0.00943939],
[ 0.01966154],
[ 0.07462995],
[-0.00836158],
[-0.02345095],
[-0.046085  ],
[ 0.05415152],
[-0.03530688],
[-0.03207344],
[-0.0816528 ],
[ 0.04768465],
[ 0.06061839],
[ 0.05630715],
[ 0.09834182],
[ 0.05954058],
[ 0.03367309],
[ 0.05630715],
[-0.06548562],
[ 0.16085492],
[-0.05578531],
[-0.02452876],
[-0.03638469],
[-0.00836158],
[-0.04177375],
[ 0.12744274],
[-0.07734155],
[ 0.02828403],
[-0.02560657],
[-0.06225218],
[-0.00081689],
```

```
[ 0.08864151],
[-0.03207344],
[ 0.03043966],
[ 0.00888341],
[ 0.00672779],
[-0.02021751],
[-0.02452876],
[-0.01159501],
[ 0.02612841],
[-0.05901875],
[-0.03638469],
[-0.02452876],
[ 0.01858372],
[-0.0902753 ],
[-0.00512814],
[-0.05255187],
[-0.02237314],
[-0.02021751],
[-0.0547075 ],
[-0.00620595],
[-0.01698407],
[ 0.05522933],
[ 0.07678558],
[ 0.01858372],
[-0.02237314],
[ 0.09295276],
[-0.03099563],
[ 0.03906215],
[-0.06117437],
[-0.00836158],
[-0.0374625 ],
[-0.01375064],
[ 0.07355214],
[-0.02452876],
[ 0.03367309],
[ 0.0347509 ],
[-0.03854032],
[-0.03961813],
[-0.00189471],
[-0.03099563],
[-0.046085  ],
[ 0.00133873],
[ 0.06492964],
[ 0.04013997],
[-0.02345095],
[ 0.05307371],
[ 0.04013997],
[-0.02021751],
[ 0.01427248],
[-0.03422907],
[ 0.00672779],
[ 0.00457217],
[ 0.03043966],
[ 0.0519959 ],
[ 0.06169621],
[-0.00728377],
[ 0.00564998],
[ 0.05415152],
[-0.00836158],
[ 0.114509  ],
```

```
[ 0.06708527],
[-0.05578531],
[ 0.03043966],
[-0.02560657],
[ 0.10480869],
[-0.00620595],
[-0.04716281],
[-0.04824063],
[ 0.08540807],
[-0.01267283],
[-0.03315126],
[-0.00728377],
[-0.01375064],
[ 0.05954058],
[ 0.02181716],
[ 0.01858372],
[-0.01159501],
[-0.00297252],
[ 0.01750591],
[-0.02991782],
[-0.02021751],
[-0.05794093],
[ 0.06061839],
[-0.04069594],
[-0.07195249],
[-0.05578531],
[ 0.04552903],
[-0.00943939],
[-0.03315126],
[ 0.04984027],
[-0.08488624],
[ 0.00564998],
[ 0.02073935],
[-0.00728377],
[ 0.10480869],
[-0.02452876],
[-0.00620595],
[-0.03854032],
[ 0.13714305],
[ 0.17055523],
[ 0.00241654],
[ 0.03798434],
[-0.05794093],
[-0.00943939],
[-0.02345095],
[-0.0105172 ],
[-0.03422907],
[-0.00297252],
[ 0.06816308],
[ 0.00996123],
[ 0.00241654],
[-0.03854032],
[ 0.02612841],
[-0.08919748],
[ 0.06061839],
[-0.02884001],
[-0.02991782],
[-0.0191397 ],
[-0.04069594],
[ 0.01535029],
```

```
                [-0.02452876],
                [ 0.00133873],
                [ 0.06924089],
                [-0.06979687],
                [-0.02991782],
                [-0.046085  ],
                [ 0.01858372],
                [ 0.00133873],
                [-0.03099563],
                [-0.00405033],
                [ 0.01535029],
                [ 0.02289497],
                [ 0.04552903],
                [-0.04500719],
                [-0.03315126],
                [ 0.097264  ],
                [ 0.05415152],
                [ 0.12313149],
                [-0.08057499],
                [ 0.09295276],
                [-0.05039625],
                [-0.01159501],
                [-0.0277622 ],
                [ 0.05846277],
                [ 0.08540807],
                [-0.00081689],
                [ 0.00672779],
                [ 0.00888341],
                [ 0.08001901],
                [ 0.07139652],
                [-0.02452876],
                [-0.0547075 ],
                [-0.03638469],
                [ 0.0164281 ],
                [ 0.07786339],
                [-0.03961813],
                [ 0.01103904],
                [-0.04069594],
                [-0.03422907],
                [ 0.00564998],
                [ 0.08864151],
                [-0.03315126],
                [-0.05686312],
                [-0.03099563],
                [ 0.05522933],
                [-0.06009656],
                [ 0.00133873],
                [-0.02345095],
                [-0.07410811],
                [ 0.01966154],
                [-0.01590626],
                [-0.01590626],
                [ 0.03906215],
                [-0.0730303 ]])
```

In [22]:
```
diabetes_X = diabetes.data[:, np.newaxis, 2]
```

In [23]:
```
diabetes.target
```

Out[23]:
```
array([ 151.,   75.,  141.,  206.,  135.,   97.,  138.,   63.,  110.,
        310.,  101.,   69.,  179.,  185.,  118.,  171.,  166.,  144.,
         97.,  168.,   68.,   49.,   68.,  245.,  184.,  202.,  137.,
```

```
 85.,  131.,  283.,  129.,   59.,  341.,   87.,   65.,  102.,
265.,  276.,  252.,   90.,  100.,   55.,   61.,   92.,  259.,
 53.,  190.,  142.,   75.,  142.,  155.,  225.,   59.,  104.,
182.,  128.,   52.,   37.,  170.,  170.,   61.,  144.,   52.,
128.,   71.,  163.,  150.,   97.,  160.,  178.,   48.,  270.,
202.,  111.,   85.,   42.,  170.,  200.,  252.,  113.,  143.,
 51.,   52.,  210.,   65.,  141.,   55.,  134.,   42.,  111.,
 98.,  164.,   48.,   96.,   90.,  162.,  150.,  279.,   92.,
 83.,  128.,  102.,  302.,  198.,   95.,   53.,  134.,  144.,
232.,   81.,  104.,   59.,  246.,  297.,  258.,  229.,  275.,
281.,  179.,  200.,  200.,  173.,  180.,   84.,  121.,  161.,
 99.,  109.,  115.,  268.,  274.,  158.,  107.,   83.,  103.,
272.,   85.,  280.,  336.,  281.,  118.,  317.,  235.,   60.,
174.,  259.,  178.,  128.,   96.,  126.,  288.,   88.,  292.,
 71.,  197.,  186.,   25.,   84.,   96.,  195.,   53.,  217.,
172.,  131.,  214.,   59.,   70.,  220.,  268.,  152.,   47.,
 74.,  295.,  101.,  151.,  127.,  237.,  225.,   81.,  151.,
107.,   64.,  138.,  185.,  265.,  101.,  137.,  143.,  141.,
 79.,  292.,  178.,   91.,  116.,   86.,  122.,   72.,  129.,
142.,   90.,  158.,   39.,  196.,  222.,  277.,   99.,  196.,
202.,  155.,   77.,  191.,   70.,   73.,   49.,   65.,  263.,
248.,  296.,  214.,  185.,   78.,   93.,  252.,  150.,   77.,
208.,   77.,  108.,  160.,   53.,  220.,  154.,  259.,   90.,
246.,  124.,   67.,   72.,  257.,  262.,  275.,  177.,   71.,
 47.,  187.,  125.,   78.,   51.,  258.,  215.,  303.,  243.,
 91.,  150.,  310.,  153.,  346.,   63.,   89.,   50.,   39.,
103.,  308.,  116.,  145.,   74.,   45.,  115.,  264.,   87.,
202.,  127.,  182.,  241.,   66.,   94.,  283.,   64.,  102.,
200.,  265.,   94.,  230.,  181.,  156.,  233.,   60.,  219.,
 80.,   68.,  332.,  248.,   84.,  200.,   55.,   85.,   89.,
 31.,  129.,   83.,  275.,   65.,  198.,  236.,  253.,  124.,
 44.,  172.,  114.,  142.,  109.,  180.,  144.,  163.,  147.,
 97.,  220.,  190.,  109.,  191.,  122.,  230.,  242.,  248.,
249.,  192.,  131.,  237.,   78.,  135.,  244.,  199.,  270.,
164.,   72.,   96.,  306.,   91.,  214.,   95.,  216.,  263.,
178.,  113.,  200.,  139.,  139.,   88.,  148.,   88.,  243.,
 71.,   77.,  109.,  272.,   60.,   54.,  221.,   90.,  311.,
281.,  182.,  321.,   58.,  262.,  206.,  233.,  242.,  123.,
167.,   63.,  197.,   71.,  168.,  140.,  217.,  121.,  235.,
245.,   40.,   52.,  104.,  132.,   88.,   69.,  219.,   72.,
201.,  110.,   51.,  277.,   63.,  118.,   69.,  273.,  258.,
 43.,  198.,  242.,  232.,  175.,   93.,  168.,  275.,  293.,
281.,   72.,  140.,  189.,  181.,  209.,  136.,  261.,  113.,
131.,  174.,  257.,   55.,   84.,   42.,  146.,  212.,  233.,
 91.,  111.,  152.,  120.,   67.,  310.,   94.,  183.,   66.,
173.,   72.,   49.,   64.,   48.,  178.,  104.,  132.,  220.,   57.])
```

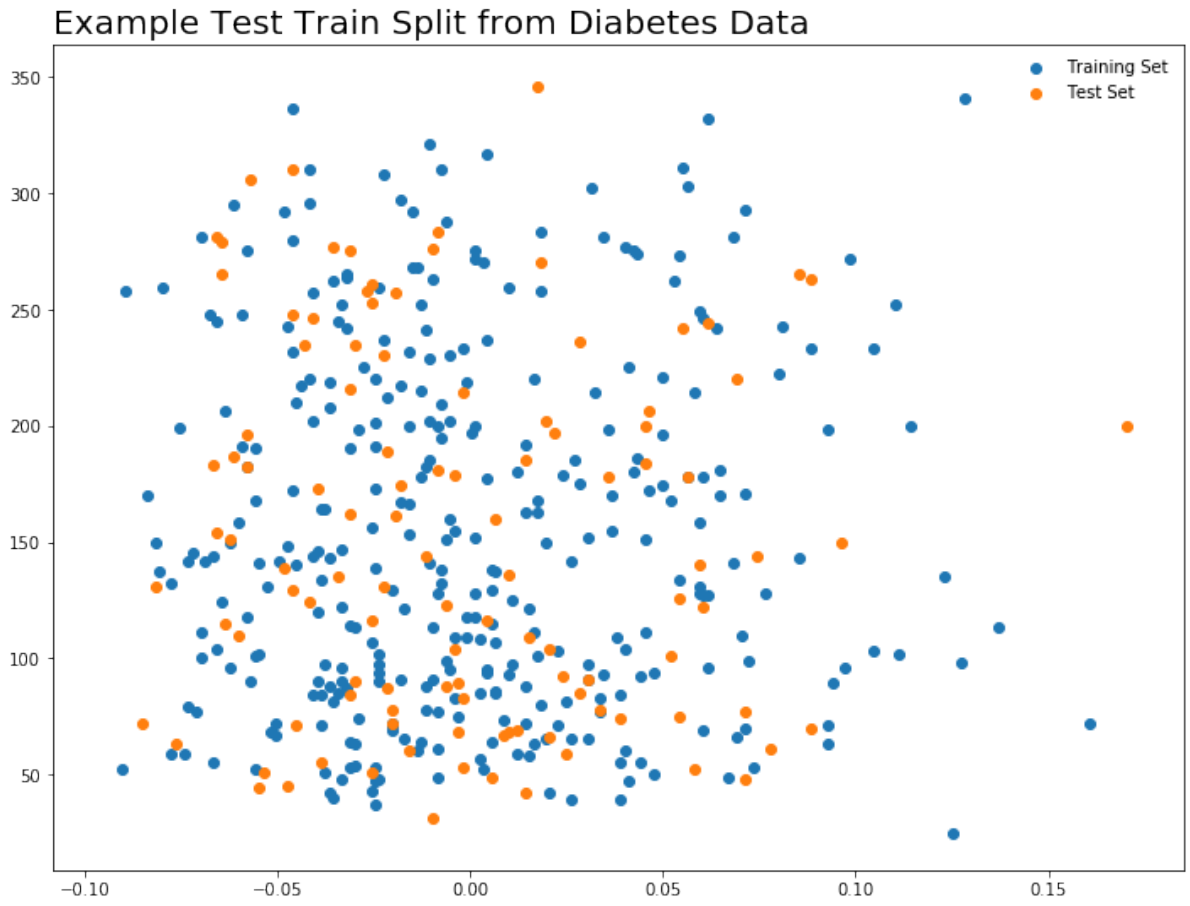
In [24]:
```
diabetes_y = diabetes.target
```

In [25]:
```
from sklearn.model_selection import train_test_split
```

In [26]:
```
X_train, x_test = train_test_split(diabetes_X)
y_train, y_test = train_test_split(diabetes_y)
```

In [32]:
```
plt.figure(figsize = (12, 9))
plt.scatter(X_train, y_train, label = 'Training Set')
plt.scatter(x_test, y_test, label = 'Test Set')
plt.legend(frameon = False)
plt.title("Example Test Train Split from Diabetes Data", loc = 'left', size = 20)
```

Out[32]:
```
Text(0,1,'Example Test Train Split from Diabetes Data')
```

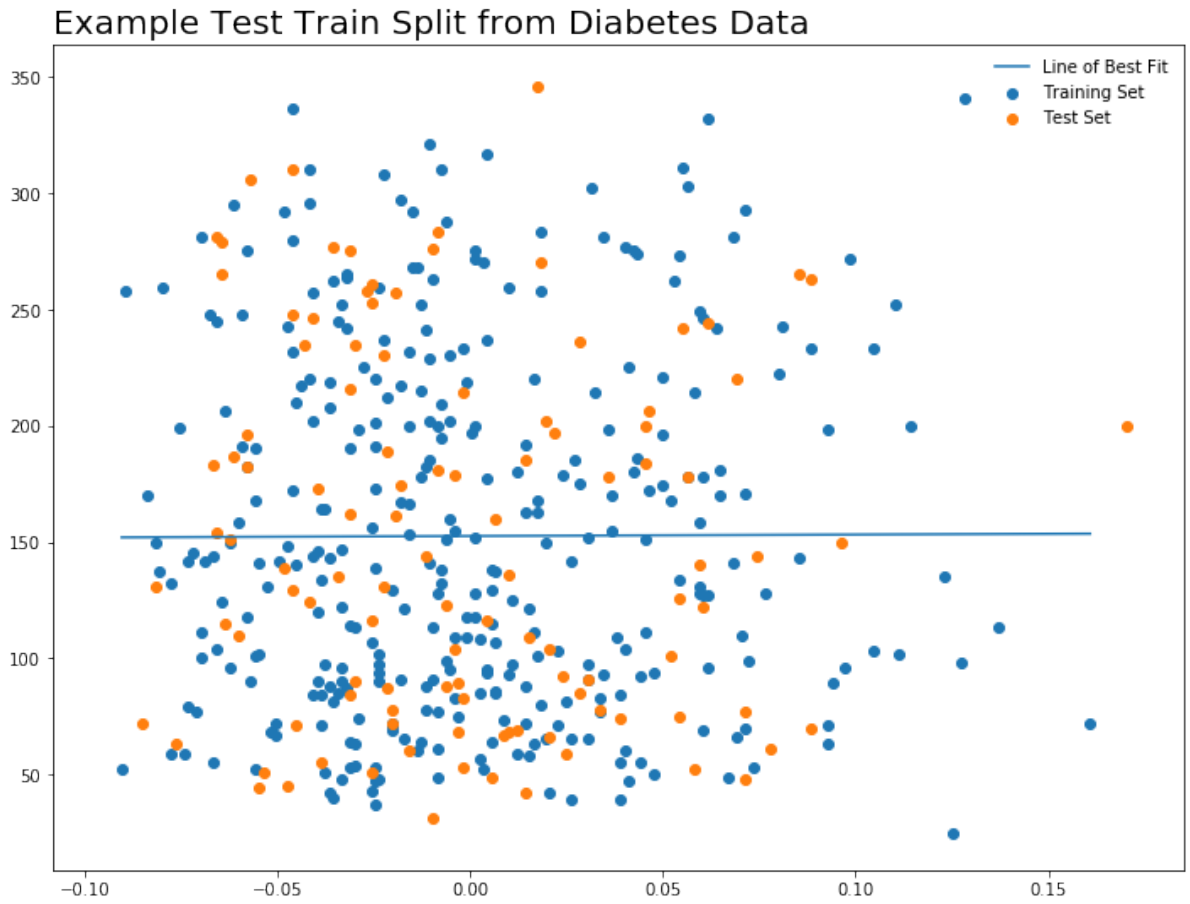


11.2 Linear Regression: Fitting and Evaluating the Model

```
In [35]: regr = linear_model.LinearRegression()

In [36]: regr.fit(X_train, y_train)

Out[36]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)

In [38]: predictions = regr.predict(x_test)

In [40]: print("The coefficients of the model are: \n", regr.coef_)

The coefficients of the model are:
 [ 6.29641819]

In [41]: print("The intercept of the model are: \n", regr.intercept_)

The intercept of the model are:
 152.512205614

In [43]: print("The Equation for the Line of Best Fit is \n y = ", regr.coef_, 'x +', regr.intercept_)

The Equation for the Line of Best Fit is
 y =  [ 6.29641819] x + 152.512205614

In [44]: def l(x):
             return regr.coef_*x + regr.intercept_

In [45]: l(30)

Out[45]: array([ 341.40475121])

In [46]: x = np.linspace(min(X_train), max(X_train), 1000)
```

```
In [47]: plt.figure(figsize = (12, 9))
         plt.scatter(X_train, y_train, label = 'Training Set')
         plt.scatter(x_test, y_test, label = 'Test Set')
         plt.plot(x, l(x), label = 'Line of Best Fit')
         plt.legend(frameon = False)
         plt.title("Example Test Train Split from Diabetes Data", loc = 'left', size = 20)
```

```
Out[47]: Text(0,1,'Example Test Train Split from Diabetes Data')
```




```
In [48]: print("The Mean Squared Error of the model is", mean_squared_error(y_test, predictions))
```

```
The Mean Squared Error of the model is 6126.13411338
```

```
In [49]: print("The Variance Score is ", r2_score(y_test, predictions))
```

```
The Variance Score is  -0.000950748287665
```

```
In [51]: regr.get_params
```

```
Out[51]: <bound method BaseEstimator.get_params of LinearRegression(copy_X=True, fit_intercept=True, n_job
```

11.3 Using StatsModels and Seaborn

```
In [57]: import statsmodels.api as sm
         import statsmodels.formula.api as smf
         import pandas as pd
```

```
In [60]: df = pd.DataFrame()
```

```
In [67]: df['bmi'] = diabetes.data[:, 2]
```

```
In [68]: df['disease'] = diabetes.target
```

```
In [69]: df.head()
```

Out[69]:
	bmi	disease
0	0.061696	151.0
1	-0.051474	75.0
2	0.044451	141.0
3	-0.011595	206.0
4	-0.036385	135.0

In [73]:
```
len(df['bmi'])
```

Out[73]: 442

In [75]:
```
results = smf.ols('disease ~ bmi', data = df).fit()
```

In [76]:
```
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                 disease   R-squared:                       0.344
Model:                             OLS   Adj. R-squared:                  0.342
Method:                  Least Squares   F-statistic:                     230.7
Date:                 Sat, 10 Feb 2018   Prob (F-statistic):           3.47e-42
Time:                         14:16:19   Log-Likelihood:                -2454.0
No. Observations:                 442   AIC:                             4912.
Df Residuals:                     440   BIC:                             4920.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept    152.1335      2.974     51.162      0.000     146.289     157.978
bmi          949.4353     62.515     15.187      0.000     826.570    1072.301
==============================================================================
Omnibus:                       11.674   Durbin-Watson:                   1.848
Prob(Omnibus):                  0.003   Jarque-Bera (JB):                7.310
Skew:                           0.156   Prob(JB):                       0.0259
Kurtosis:                       2.453   Cond. No.                         21.0
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

In [77]:
```
df2 = df[:300]
```

In [78]:
```
df2.head()
```

Out[78]:
	bmi	disease
0	0.061696	151.0
1	-0.051474	75.0
2	0.044451	141.0
3	-0.011595	206.0
4	-0.036385	135.0

In [79]:
```
df2b = df[300:]
```

In [80]:
```
df2b.head()
```

Out[80]:
	bmi	disease
300	0.073552	275.0
301	-0.024529	65.0
302	0.033673	198.0
303	0.034751	236.0
304	-0.038540	253.0

In [83]:
```
split_results = smf.ols('disease ~ bmi', data = df2).fit()
```

In [84]:
```
print(split_results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                 disease   R-squared:                       0.342
Model:                             OLS   Adj. R-squared:                  0.340
Method:                  Least Squares   F-statistic:                     154.8
Date:                 Sat, 10 Feb 2018   Prob (F-statistic):           6.61e-29
Time:                         14:18:03   Log-Likelihood:                -1668.4
No. Observations:                  300   AIC:                             3341.
Df Residuals:                      298   BIC:                             3348.
Df Model:                            1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept    151.0306      3.651     41.372      0.000     143.846     158.215
bmi          975.5736     78.405     12.443      0.000     821.276    1129.872
==============================================================================
Omnibus:                        9.498   Durbin-Watson:                   1.764
Prob(Omnibus):                  0.009   Jarque-Bera (JB):                6.672
Skew:                           0.238   Prob(JB):                       0.0356
Kurtosis:                       2.446   Cond. No.                        21.5
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

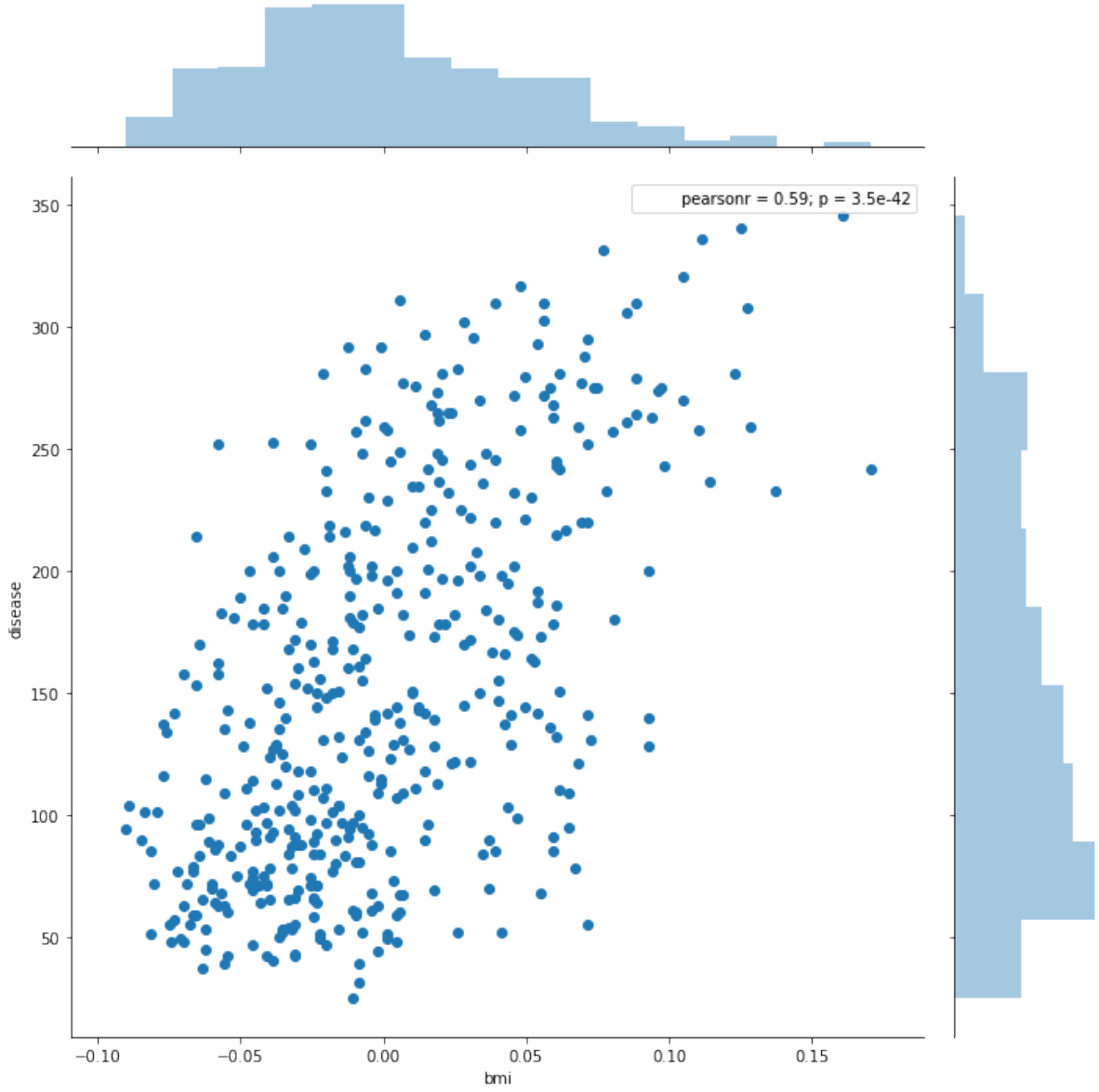
In [87]:
```
predictions = split_results.predict(df2b['bmi'])
```

In [88]:
```
predictions[:10]
```

Out[88]:
```
300    222.786110
301    127.100973
302    183.881164
303    184.932649
304    113.431668
305    112.380183
306    149.182158
307    120.792063
308    106.071273
309    152.336613
dtype: float64
```

In [95]:
```
import seaborn as sns
sns.jointplot('bmi', 'disease', data = df, size = 10)
```

Out[95]:
```
<seaborn.axisgrid.JointGrid at 0x1c216fc438>
```



11.4 Other Examples of Machine Learning

- What category does this belong to?
- What is this a picture of?

```
In [12]: from sklearn import datasets

In [13]: iris = datasets.load_iris()
         digits = datasets.load_digits()

In [14]: print(digits.data)

[[  0.   0.   5. ...,   0.   0.   0.]
 [  0.   0.   0. ...,  10.   0.   0.]
 [  0.   0.   0. ...,  16.   9.   0.]
 ...,
 [  0.   0.   1. ...,   6.   0.   0.]
 [  0.   0.   2. ...,  12.   0.   0.]
 [  0.   0.  10. ...,  12.   1.   0.]]

In [15]: digits.target
```

```
Out[15]: array([0, 1, 2, ..., 8, 9, 8])
```

```
In [16]: digits.images[0]
```

```
Out[16]: array([[  0.,   0.,   5.,  13.,   9.,   1.,   0.,   0.],
                [  0.,   0.,  13.,  15.,  10.,  15.,   5.,   0.],
                [  0.,   3.,  15.,   2.,   0.,  11.,   8.,   0.],
                [  0.,   4.,  12.,   0.,   0.,   8.,   8.,   0.],
                [  0.,   5.,   8.,   0.,   0.,   9.,   8.,   0.],
                [  0.,   4.,  11.,   0.,   1.,  12.,   7.,   0.],
                [  0.,   2.,  14.,   5.,  10.,  12.,   0.,   0.],
                [  0.,   0.,   6.,  13.,  10.,   0.,   0.,   0.]])
```

```
In [17]: iris.data[:5]
```

```
Out[17]: array([[ 5.1,  3.5,  1.4,  0.2],
                [ 4.9,  3. ,  1.4,  0.2],
                [ 4.7,  3.2,  1.3,  0.2],
                [ 4.6,  3.1,  1.5,  0.2],
                [ 5. ,  3.6,  1.4,  0.2]])
```

```
In [18]: iris.target
```

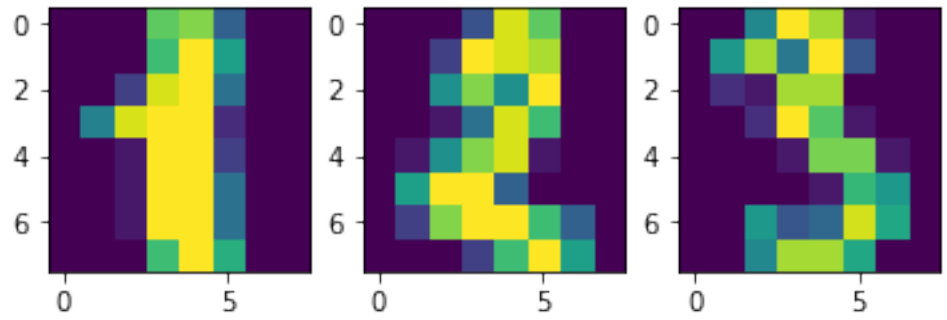
```
Out[18]: array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
                2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
                2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

11.5 What kind of Flower is This?

- **K-Means Clustering**
- **Naive Bayes Classifier**
- **Decision Tree**

```
In [19]: plt.subplot(1, 3, 1)
         plt.imshow(digits.images[1])

         plt.subplot(1, 3, 2)
         plt.imshow(digits.images[2])

         plt.subplot(1, 3, 3)
         plt.imshow(digits.images[3])
```

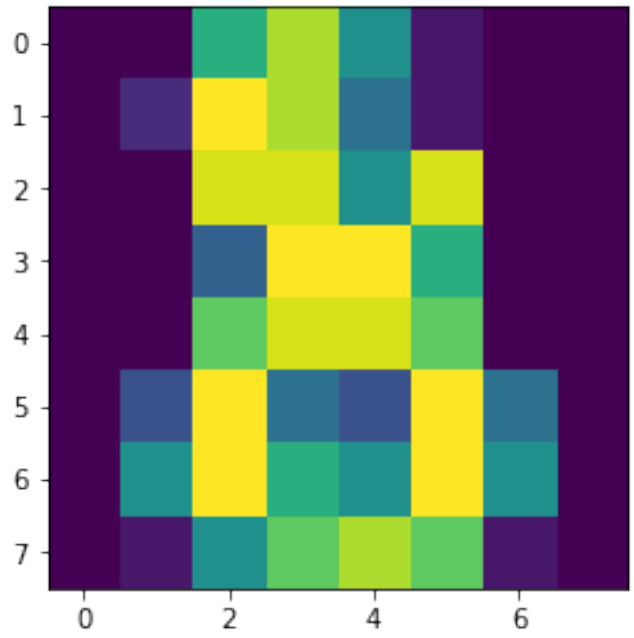
```
Out[19]: <matplotlib.image.AxesImage at 0x1a15e43630>
```



11.6 Learning and Predicting with Digits

Given an image, which digit does it represent? Here, we will *fit* an estimator to *predict* which class unknown images belong to. To do this, we will use the support vector classifier.

```
In [20]: from sklearn import svm

In [21]: clf = svm.SVC(gamma = 0.001, C = 100)

In [22]: #fit on all but last data point
         clf.fit(digits.data[:-1], digits.target[:-1])

Out[22]: SVC(C=100, cache_size=200, class_weight=None, coef0=0.0,
           decision_function_shape='ovr', degree=3, gamma=0.001, kernel='rbf',
           max_iter=-1, probability=False, random_state=None, shrinking=True,
           tol=0.001, verbose=False)

In [23]: clf.predict(digits.data[-1:])

Out[23]: array([8])

In [24]: plt.imshow(digits.images[-1])

Out[24]: <matplotlib.image.AxesImage at 0x1a15db4278>
```



12 Decision Tree Classifiers

12.1 Example

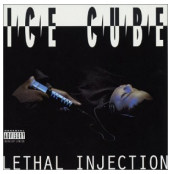
12.2 Important Considerations

PROS	CONS
Easy to visualize and Interpret	Prone to overfitting
No normalization of Data Necessary	Ensemble needed for better performance
Handles mixed feature types	

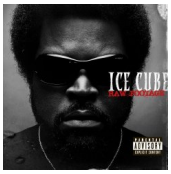
Was it good?

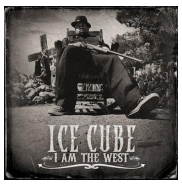
Named after TV Show?

Came out in 90's?



Lil John was involved?





Good Day?

12.3 Iris Example

Use measurements to predict species

Iris Setosa | Iris Virginica | Iris Versicolor

```
In [2]: %matplotlib inline
        import matplotlib.pyplot as plt
        from sklearn.datasets import load_iris
        from sklearn import tree
        from sklearn.datasets import load_iris
        from sklearn.tree import DecisionTreeClassifier
        from sklearn.model_selection import train_test_split
```

```
In [3]: import seaborn as sns
        iris = sns.load_dataset('iris')
        iris.head()
```

```
Out[3]: sepal_length  sepal_width  petal_length  petal_width species
        0           5.1          3.5           1.4          0.2  setosa
        1           4.9          3.0           1.4          0.2  setosa
        2           4.7          3.2           1.3          0.2  setosa
        3           4.6          3.1           1.5          0.2  setosa
        4           5.0          3.6           1.4          0.2  setosa
```

```
In [4]: #split the data
        iris = load_iris()
        X_train, X_test, y_train, y_test = train_test_split(iris.data, iris.target)
```

```
In [5]: len(X_test)
```

```
Out[5]: 38
```

```
In [6]: #load classifier
        clf = tree.DecisionTreeClassifier()
```

```
In [7]: #fit train data
        clf = clf.fit(X_train, y_train)
```

```
In [8]: #examine score
        clf.score(X_train, y_train)
```

```
Out[8]: 1.0
```

```
In [9]: #against test set
        clf.score(X_test, y_test)
```

```
Out[9]: 0.92105263157894735
```

12.4 How would specific flower be classified?

If we have a flower that has:

- Sepal.Length = 1.0

- Sepal.Width = 0.3
- Petal.Length = 1.4
- Petal.Width = 2.1

```
In [10]: clf.predict_proba([[1.0, 0.3, 1.4, 2.1]])
```

```
Out[10]: array([[ 0.,  1.,  0.]])
```

```
In [11]: #cross validation
         from sklearn.model_selection import cross_val_score
         cross_val_score(clf, X_train, y_train, cv=10)
```

```
Out[11]: array([ 0.83333333,  1.        ,  1.        ,  0.91666667,  0.91666667,
                 1.        ,  0.90909091,  1.        ,  1.        ,  0.9       ])
```

12.5 How important are different features?

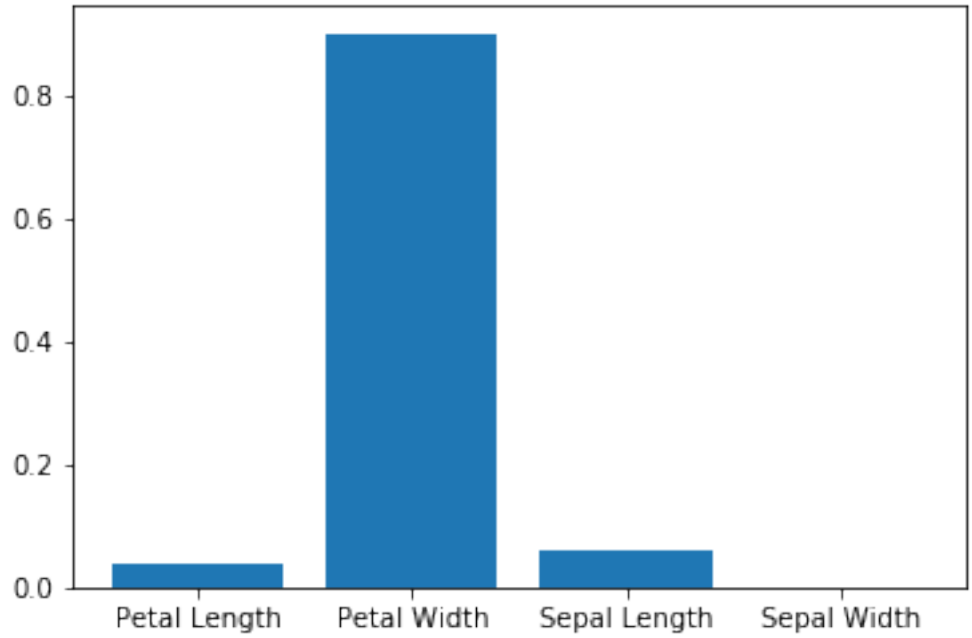
```
In [12]: #list of feature importance
         clf.feature_importances_
```

```
Out[12]: array([ 0.06184963,  0.        ,  0.03845214,  0.89969823])
```

```
In [13]: imp = clf.feature_importances_
```

```
In [14]: plt.bar(['Sepal Length', 'Sepal Width', 'Petal Length', 'Petal Width'], imp)
```

```
Out[14]: <Container object of 4 artists>
```

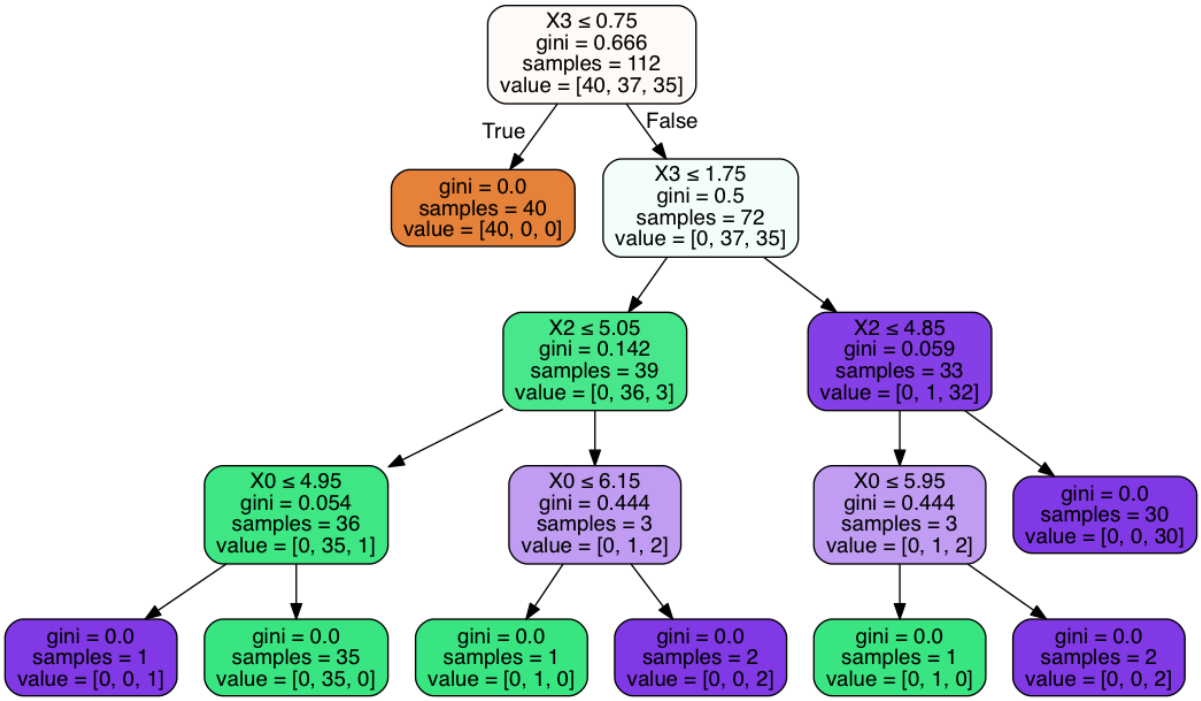


12.6 Visualizing Decision Tree

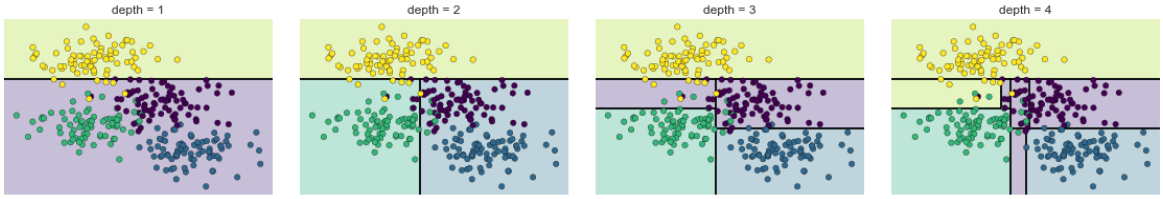
```
pip install pydotplus
```

```
In [15]: from sklearn.externals.six import StringIO
         from IPython.display import Image
         from sklearn.tree import export_graphviz
         import pydotplus

         dot_data = StringIO()
```

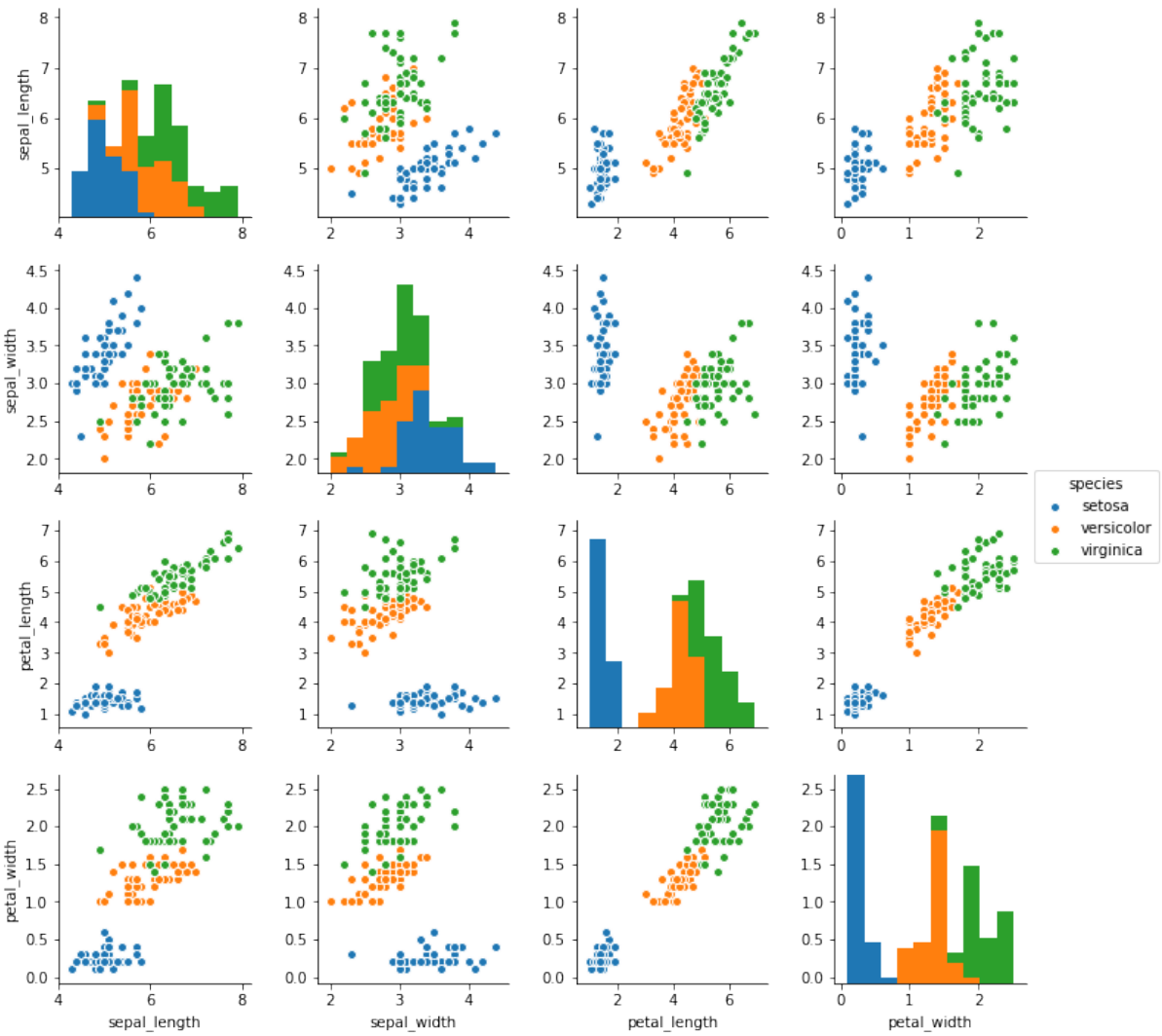
```
export_graphviz(clf, out_file=dot_data,  filled=True, rounded=True, special_characters=True)
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
```



12.7 What's Happening with Decision Tree



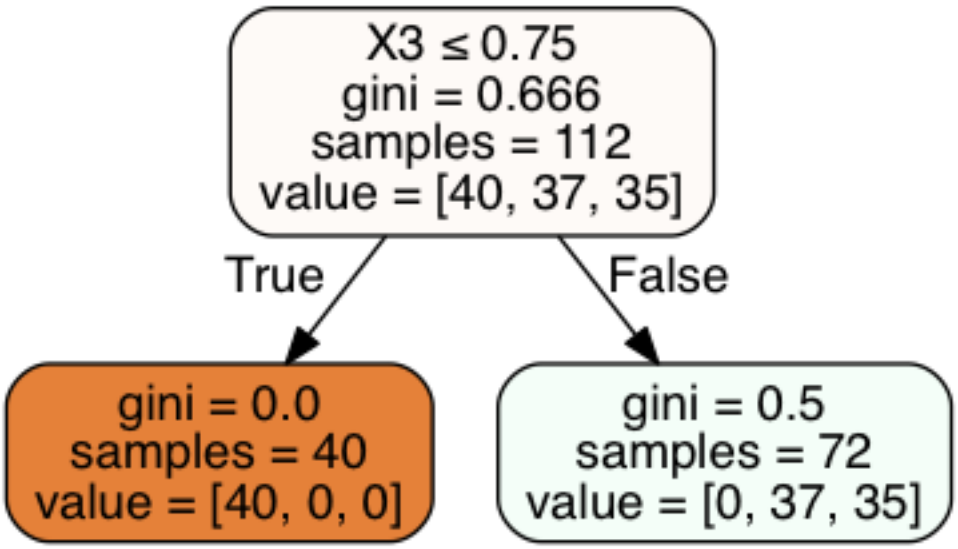
In [16]:
```
import seaborn as sns
iris = sns.load_dataset('iris')
sns.pairplot(data = iris, hue = 'species');
```



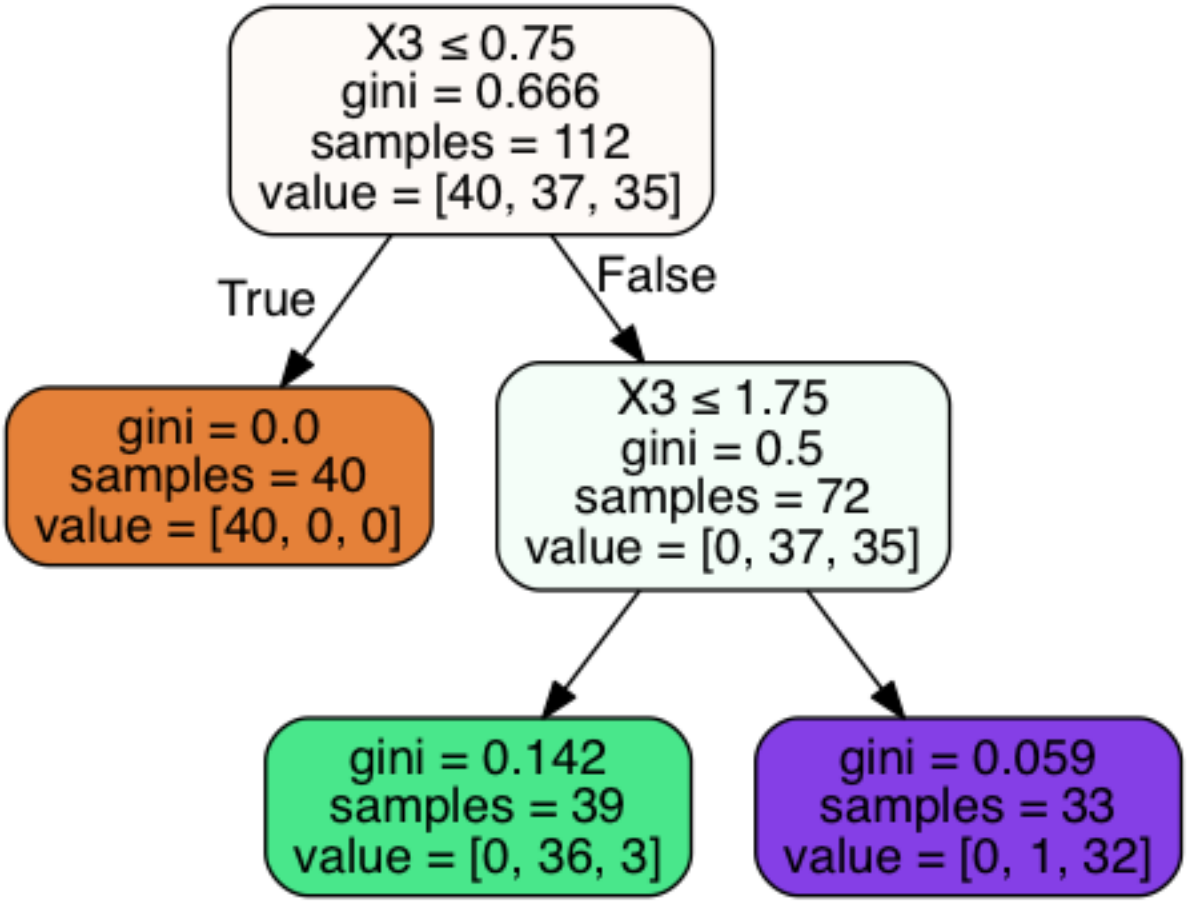
12.8 Pre-pruning: Avoiding Over-fitting

- `max_depth`: limits depth of tree
- `max_leaf_nodes`: limits how many leafs
- `min_samples_leaf`: limits splits to happen when only certain number of samples exist

```
In [17]: clf = DecisionTreeClassifier(max_depth = 1).fit(X_train, y_train)

In [18]: clf.score(X_train, y_train)

Out[18]: 0.6875

In [19]: clf.score(X_test, y_test)

Out[19]: 0.60526315789473684

In [20]: dot_data = StringIO()
         export_graphviz(clf, out_file=dot_data,  filled=True, rounded=True, special_characters=True)
         graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
         Image(graph.create_png())
```




```
In [21]: clf = DecisionTreeClassifier(max_depth = 2).fit(X_train, y_train)

In [22]: clf.score(X_train, y_train)

Out[22]: 0.9642857142857143

In [23]: clf.score(X_test, y_test)

Out[23]: 0.94736842105263153

In [24]: dot_data = StringIO()
         export_graphviz(clf, out_file=dot_data,  filled=True, rounded=True, special_characters=True)
         graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
         Image(graph.create_png())
```



In [25]:
```
clf = DecisionTreeClassifier(max_depth = 3).fit(X_train, y_train)
clf.score(X_train, y_train)
```

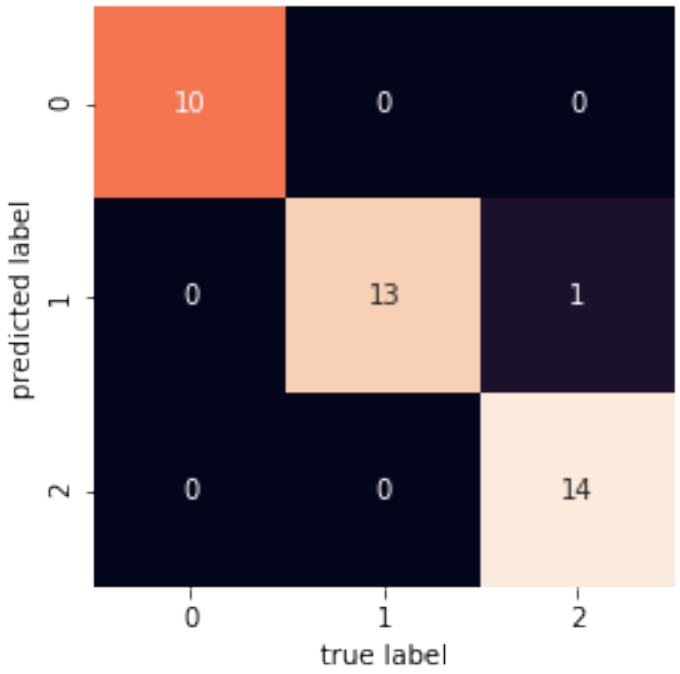
Out[25]: 0.9732142857142857

In [26]:
```
clf.score(X_test, y_test)
```

Out[26]: 0.97368421052631582

12.9 Confusion Matrix

In [29]:
```
from sklearn.metrics import classification_report
import sklearn.metrics
from sklearn.metrics import confusion_matrix

classifier=clf.fit(X_train,y_train)

predictions=clf.predict(X_test)

mat = confusion_matrix(y_test, predictions)
sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=False)
plt.xlabel('true label')
plt.ylabel('predicted label');
```



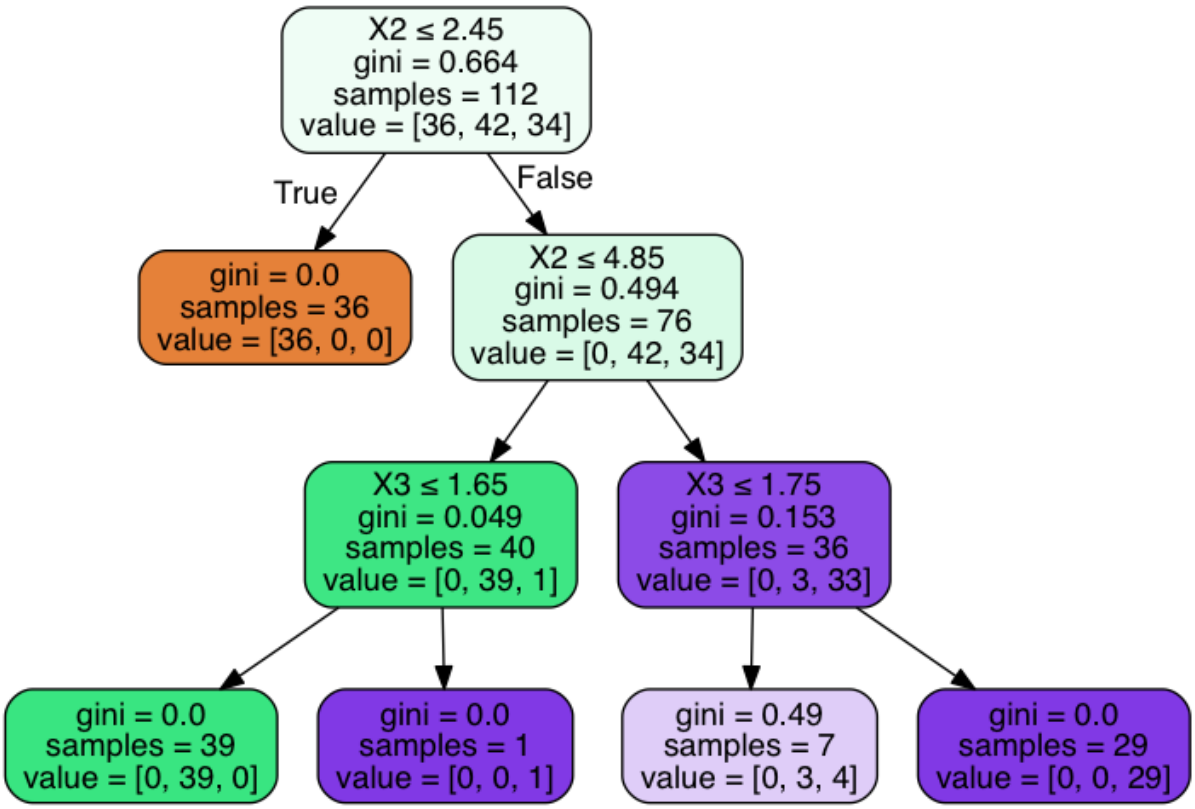
In [30]:
```
sklearn.metrics.confusion_matrix(y_test, predictions)
```

Out[30]:
```
array([[10,  0,  0],
       [ 0, 13,  0],
       [ 0,  1, 14]])
```

In [27]:
```
sklearn.metrics.accuracy_score(y_test, predictions)
```

Out[27]:
```
0.94736842105263153
```

In [28]:
```
dot_data2 = StringIO()
export_graphviz(clf, out_file=dot_data2,
                filled=True, rounded=True,
                special_characters=True)
graph2 = pydotplus.graph_from_dot_data(dot_data2.getvalue())
Image(graph2.create_png())
```



In [29]:
```
sklearn.metrics.accuracy_score(y_test, predictions)
```

Out[29]:
```
0.94736842105263153
```

12.10 Example with Adolescent Health Data

2. During your life, how many times have you used marijuana (also called COUNTRY SPECIFIC SLANG TERMS FOR MARIJUANA)?

A 0 times
B 1 or 2 times
C 3 to 9 times
D 10 to 19 times
E 20 or more times

In [33]:
```
from pandas import Series, DataFrame
import pandas as pd
import numpy as np
import matplotlib.pylab as plt

from sklearn.metrics import classification_report
import sklearn.metrics
```

In [34]:
```
AH_data = pd.read_csv("data/tree_addhealth.csv")
data_clean = AH_data.dropna()
data_clean.dtypes
```

Out[34]:
```
BIO_SEX      float64
HISPANIC     float64
WHITE        float64
BLACK        float64
NAMERICAN    float64
ASIAN        float64
age          float64
TREG1        float64
ALCEVR1      float64
ALCPROBS1      int64
marever1       int64
cocever1       int64
inhever1       int64
cigavail     float64
DEP1         float64
ESTEEM1      float64
VIOL1        float64
PASSIST        int64
DEVIANT1     float64
SCHCONN1     float64
GPA1         float64
EXPEL1       float64
FAMCONCT     float64
PARACTV      float64
PARPRES      float64
dtype: object
```

In [35]:
```
data_clean.describe()
```

Out[35]:

	BIO_SEX	HISPANIC	WHITE	BLACK	NAMERICAN	\
count	4575.000000	4575.000000	4575.000000	4575.000000	4575.000000	
mean	1.521093	0.111038	0.683279	0.236066	0.036284	
std	0.499609	0.314214	0.465249	0.424709	0.187017	
min	1.000000	0.000000	0.000000	0.000000	0.000000	
25%	1.000000	0.000000	0.000000	0.000000	0.000000	
50%	2.000000	0.000000	1.000000	0.000000	0.000000	
75%	2.000000	0.000000	1.000000	0.000000	0.000000	
max	2.000000	1.000000	1.000000	1.000000	1.000000	

	ASIAN	age	TREG1	ALCEVR1	ALCPROBS1	\
count	4575.000000	4575.000000	4575.000000	4575.000000	4575.000000	
mean	0.040437	16.493052	0.176393	0.527432	0.369180	
std	0.197004	1.552174	0.381196	0.499302	0.894947	
min	0.000000	12.676712	0.000000	0.000000	0.000000	
25%	0.000000	15.254795	0.000000	0.000000	0.000000	
50%	0.000000	16.509589	0.000000	1.000000	0.000000	
75%	0.000000	17.679452	0.000000	1.000000	0.000000	
max	1.000000	21.512329	1.000000	1.000000	6.000000	

	...	ESTEEM1	VIOL1	PASSIST	DEVIANT1	\
count	...	4575.000000	4575.000000	4575.000000	4575.000000	
mean	...	40.952131	1.618579	0.102514	2.645027	
std	...	5.381439	2.593230	0.303356	3.520554	
min	...	18.000000	0.000000	0.000000	0.000000	
25%	...	38.000000	0.000000	0.000000	0.000000	
50%	...	40.000000	0.000000	0.000000	1.000000	
75%	...	45.000000	2.000000	0.000000	4.000000	
max	...	50.000000	19.000000	1.000000	27.000000	

	SCHCONN1	GPA1	EXPEL1	FAMCONCT	PARACTV	\
count	4575.000000	4575.000000	4575.000000	4575.000000	4575.000000	

```
mean     28.360656     2.815647     0.040219    22.570557     6.290710
std       5.156385     0.770167     0.196493     2.614754     3.360219
min       6.000000     1.000000     0.000000     6.300000     0.000000
25%      25.000000     2.250000     0.000000    21.700000     4.000000
50%      29.000000     2.750000     0.000000    23.700000     6.000000
75%      32.000000     3.500000     0.000000    24.300000     9.000000
max      38.000000     4.000000     1.000000    25.000000    18.000000

           PARPRES
count  4575.000000
mean     13.398033
std       2.085837
min       3.000000
25%      12.000000
50%      14.000000
75%      15.000000
max      15.000000

[8 rows x 25 columns]
```

In [36]:
```
predictors = data_clean[['BIO_SEX','HISPANIC','WHITE','BLACK','NAMERICAN','ASIAN',
'age','ALCEVR1','ALCPROBS1','marever1','cocever1','inhever1','cigavail','DEP1',
'ESTEEM1','VIOL1','PASSIST','DEVIANT1','SCHCONN1','GPA1','EXPEL1','FAMCONCT','PARACTV',
'PARPRES']]

targets = data_clean.TREG1

pred_train, pred_test, tar_train, tar_test  =   train_test_split(predictors, targets, test_size=.

print(pred_train.shape, pred_test.shape, tar_train.shape, tar_test.shape)
```

```
(2745, 24) (1830, 24) (2745,) (1830,)
```

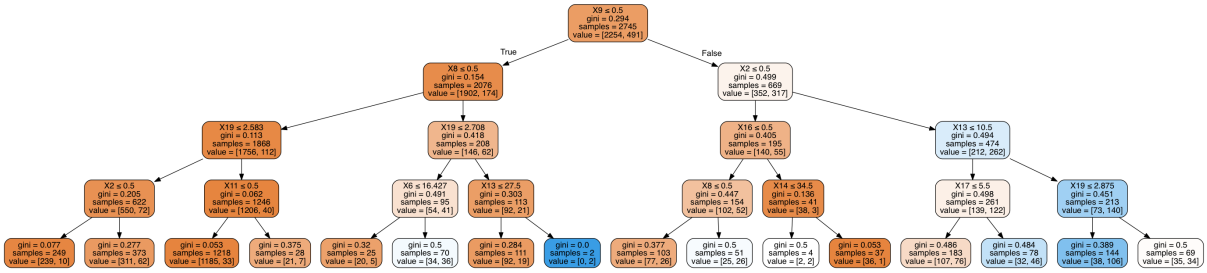
In [37]:
```
#Build model on training data
classifier=DecisionTreeClassifier(max_depth = 4)
classifier=classifier.fit(pred_train,tar_train)
predictions=classifier.predict(pred_test)

sklearn.metrics.confusion_matrix(tar_test,predictions)
```

Out[37]:
```
array([[1415,   99],
       [ 193,  123]])
```

In [38]:
```
sklearn.metrics.accuracy_score(tar_test, predictions)
```

Out[38]:
```
0.84043715846994538
```

In [39]:
```
from sklearn.externals.six import StringIO
from IPython.display import Image
from sklearn.tree import export_graphviz
import pydotplus
dot_data2 = StringIO()
export_graphviz(classifier, out_file=dot_data2,
                filled=True, rounded=True,
                special_characters=True)
graph2 = pydotplus.graph_from_dot_data(dot_data2.getvalue())
Image(graph2.create_png())
```




```
In [40]: sklearn.metrics.accuracy_score(tar_test, predictions)
```

```
Out[40]: 0.84043715846994538
```

13 Django Introduction

In this initial project, we will be introduced to formal use of the shell to create our first Django Project. As discussed, if you are using a Mac, you have a terminal available by looking in the search bar. The terminal is a place to interact with and create and edit programs. We will use the following commands:

- `cd` (change directory)
- `pwd` (print working directory)
- `ls` (list files)
- `mkdir` (make directory)
- `touch` (create file)

For example, I have a folder on my desktop where I keep all my files for this semester. I can open a new terminal and type

```
cd Desktop spring_18
```

and I will now be located in this folder. If I wanted to see the files here, I can write

```
ls -F
```

where the `-F` flags directories.

If we wanted to make a new directory named `images`, we can use

```
mkdir images
```

To create a new file, for example `home.html`, we would use

```
touch home.html
```

Finally, we will be using virtual environments for this project and will use `pipenv` to do this. In our terminal we can type

```
pip install pipenv
```

13.1 A First Django Project

To begin, we will create an empty project with Django to get a feel for using the virtual environment in the shell. We need to check that we have git working, and that we can open SublimeText (or another text editor) on our computers.

Set up Directory and Virtual Environment

Let's create a folder for our project on our desktop called django, and navigate into this folder by typing:

```
mkdir django
cd django
```

Now we will create a virtual environment where we install django.

```
pipenv install django
```

and activate it with

```
pipenv shell
```

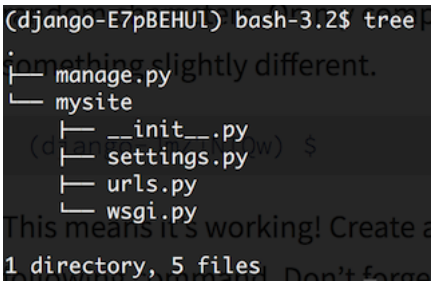
Start a Django Project

Now, we can start a project to test our new django installation. Let's create a project called `mysite` by typing the following in the terminal:

```
django-admin startproject mysite .
```

We can now examine the structure of the directory we have created with

```
tree
```



This is the standard django project structure. A `manage.py` python file, and a directory named `mysite` containing four files: `__init__.py`, `settings.py`, `urls.py`, and `wsgi.py`. To see the blank project in action, we will use the built in server, located in the `manage.py` file. To use this, we write

```
python manage.py runserver
```

We should see the project launched on our local computer at http://127.0.0.1:8000/. When we go to this page, we should see the following:

Now that we've started our project, we will add some content to it.

Starting an App

Similar to how we made use of the default Django project structure, within our project we will create an app named pages with the command

```
python manage.py startapp pages
```

django

View release notes for Django 2.0

The install worked successfully! Congratulations!

You are seeing this page because DEBUG=True is in your settings file and you have not configured any URLs.

Django Documentation

Topics, references, & how-to's

Tutorial: A Polling App

Get started with Django

```
(django-E7pBEHUl) bash-3.2$ tree
.
├── __init__.py
├── admin.py
├── apps.py
├── migrations
│   └── __init__.py
├── models.py
├── tests.py
└── views.py
1 directory, 7 files
```

Now, we have a directory with the following structure

We now will link this application to the Django project by opening the `settings.py` file located in the main `mysite` directory in a text editor. Find `INSTALLED_APPS`, and we will add our app `pages` to the list as shown.

```
32
33 ▼ INSTALLED_APPS = [
34       'pages',
35       'django.contrib.admin',
36       'django.contrib.auth',
37       'django.contrib.contenttypes',
38       'django.contrib.sessions',
39       'django.contrib.messages',
40       'django.contrib.staticfiles',
41   ]
```

Django Views

Now, we want to add some content to our app, and establish some connections that allow the content to be seen. In Django, the views determing the content displayed. We then have to use the urlconfs to decide where the content goes.

Starting with the views file, lets add the following code:

```
from django.shortcuts import render
from django.http import HttpResponse

# Create your views here.
def homepageview(request):
    return HttpResponse("<h3>It's So Wonderful to see you Jacob!</h3>")
```

This view will accept a request, and return the HTML header that I've placed in `HttpResponse()`. Now, we have to establish the location for the file using a `urls` file. We create a new file in our pages directory named `urls.py`. Here, will use the urlpatterns call and provide a path to our page. If we want it to be at a page called `home`, we could write the following:

```
from django.urls import path

from . import views

urlpatterns = [
    path('', views.homepageview, name = 'home')
]
```

This establishes the link within the application, and we need to connect this to the larger project within the base `urls.py` file. This was already created with our project, and we want it to read as follows:

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

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('pages.urls')),
]
```

Now, if we run our server again and navigate to http://127.0.0.1:8000/ we should see the results of our work.

It's So Wonderful to see you Jacob!

14 Templates and Bootstrap

Now, we will use Django's built in templating to style our home page. Django will look within each app for templates or for a base folder call `templates`. We will create a folder in the main project to house our templates, and a file called `home` to place our styling in.

```
mkdir templates
touch templates/home.html
```

Now, we will use the `settings.py` file to establish our template and tell Django where it is located. Adding the line:

```
TEMPLATES = [
    {
        'DIRS': [os.path.join(BASE_DIR, 'templates')],

},
]
```

Let's add some HTML to our home.html file as well.

```
<h1>Hello again.</h1>
```

14.1 Updating Views

There is a built-in TemplateView method that we will use in the `views.py` file. Here, we follow a similar approach to our last example in terms of mapping urls. In our `views.py` file we will add

```
from django.views.generic import TemplateView

class HomePageView(TemplateView):
    template_name = 'home.html'
```

In the app level `urls.py`, we just need to change the line in our `urlpatterns` list:

```
path('', views.HomePageView.as_view(), name = 'home')
```

Now, if we restart the server we will have our new home page rendered.

14.2 Adding a Page

To add a page, we will create a new template, view, and url route just as above. We can call this page our about page.

```
touch templates/about.html
```

Add HTML to the about page.

```
<h1>This is about me.</h1>
```

We create a view in our `views.py` file, we will create an aboutpageview class.

```
class aboutpageview(TemplateView):
    template_name = 'about.html'
```

In our `urls.py` file, we add a line to our urlpatterns to direct visitors to the about page.

```
path('about/', views.aboutpageview.as_view(), name = 'about'),
```

14.3 Extending the Template

Now, we will create a base file that we can use to extend a style across multiple pages. To do so, we will create a base file, and then use the Django minimal templating language to pull the formatting in to the additional pages.

```
touch templates/base.html
```

Here, we can add a minimal header to see how this can be applied to all pages. In the new `base.html` file, write the following:

```
<header>
  <a href="{% url 'home' %}">Home</a> | <a href="{% url 'about' %}">About</a>
</header>

{% block content %}
{% endblock %}
```

Now, we alter the `home.html` and `about.html` files to extend the `base.html` file. In each, we will add the line

```
{% extends 'base.html' %}
```

Finally, we wrap the content of both pages with `{% block content %}{% endblock %}`. Thus, in our `home.html` file, we have:

```
{% extends 'base.html' %}

{% block content %}
<h1>Welcome Home Jacob.</h1>
{% endblock %}
```

Same thing in our `about.html` file. Restart the server and you should see the header appear.

14.4 Using Bootstrap

The bootstrap framework is a way around developing all our own CSS. We can either directly download the files, or use the CDN link. We will follow this approach by copying the CDN information from the Bootstrap getting started page.

https://getbootstrap.com/docs/3.3/getting-started/

Go to our `base.html` file, and add the link in a `<head></head>` tag.


```
<head>
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.
↪min.css" integrity="sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/
↪K68vbdEjh4u" crossorigin="anonymous">
</head>
```

Fire up the server and you should notice a slight formatting change.

14.5 Tests

A major part of development in Django is the use of tests to assure everything is working. While our page is extremely simple, we still want to make sure that our home and about pages are functioning to return responses. In the `tests.py` file, we will place two simple tests that verify these pages are returning a 200 response code.

```
from django.test import TestCase

# Create your tests here.
from django.test import SimpleTestCase


class SimpleTests(SimpleTestCase):
    def test_home_page_status_code(self):
        response = self.client.get('/')
        self.assertEqual(response.status_code, 200)

    def test_about_page_status_code(self):
        response = self.client.get('/about/')
        self.assertEqual(response.status_code, 200)
```

14.6 Problem

Remember that our goal is to put together a website to share our Python Data projects that we've been making in Jupyter notebooks. To do so, let's consider taking a notebook, converting it to an HTML file,

and adding this file to a a page called **Projects** where users can see our different projects for visitors to see.

To create the HTML files of the notebooks, we will use Jupyter's `nbconvert` functionality. To start, navigate to the directory where your notebooks are housed with the terminal and `cd` command. Now, whatever notebook you would like to convert, enter

```
jupyter nbconvert notebook.ipynb
```

and you will have a new HTML file in the same folder. If you want, you can enter a directory to place this new file, for example

```
jupyter nbconvert notebook.ipynb htmls/
```

assuming we have an htmls directory.

Your goal is to play around with the different bootstrap features to style your home and about pages, and to add a projects page that contains your first Python projects from the Jupyter notebooks. You should explore a nicer navbar that includes a logo image.

15 Django Models: Building a Blog

GOALS:

- Introduce Django Models and Databases
- Add a Blog to our Site
- Use Python to analyze Blog Data

15.1 Starting the Blog

We will add a blog app to our site in the familiar manner. Be sure that you start by navigating to your project directory and activate the existing virtual environment (`pipenv shell`). Now, we create the new application with

```
python manage.py startapp blog
```

Next, be sure to add this app to your `settings.py` file in the main project directory.

15.2 Django Models

As we saw in our earlier applications, we had a default `models.py` file. The `models` are Django's place to structure database elements. We will see how to use the admin console to produce entries to this database for our blog. For example, suppose we want to be able to parse *Title*, *Author*, *Body*, *Created Date*, and *Published Date*. We will create fields for these that are then stored as data in a default SQLite database.

To begin, open the `models.py` file. There are a variety of kinds of fields that we can use, but we will start with some basics. To see more refer to the Django **Field** documentation:

https://docs.djangoproject.com/en/2.0/ref/models/fields/#common-model-field-options

```
from django.db import models
from django.contrib.auth.models import User
from django.utils import timezone
from django.urls import reverse
```

```
# Create your models here.
class Post(models.Model):
    title = models.CharField(max_length = 200)
    author = models.ForeignKey(User,on_delete = models.CASCADE, related_name = 'author')
    body = models.TextField()
    created_date = models.DateTimeField(blank = True, null = True)
    published_date = models.DateTimeField(blank=True, null=True)
```

This will allow us to login to the website and directly enter new blog posts with each of these fields. Notice that the title is a `CharField` whose length has been limited. The `author` is a `ForeignKey` that maps to user. This is a many to one element, that allows the user to create multiple posts. The `body` is a `TextField` and our `created_date` and `published_date` are `DateTimeField` types.

These will make more sense once we see the administration side which we will activate now.

15.3 Django Administration

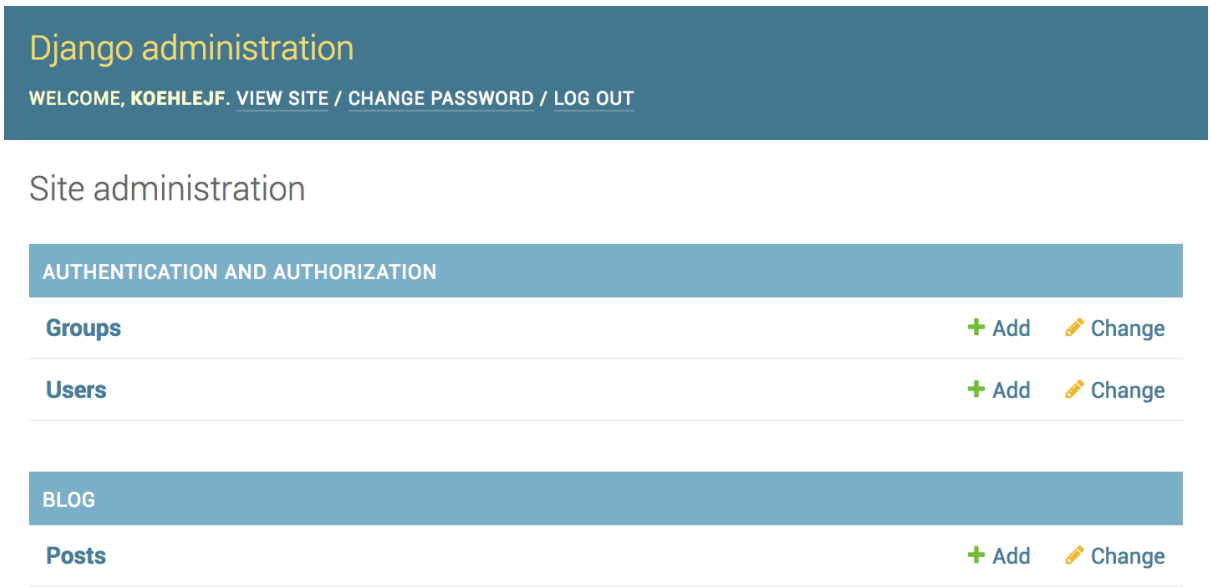
The admin side of Django allows us to login to the site and work in a friendly browser view. We start with creating a login for the admin in the terminal with:

```
python manage.py createsuperuser
```

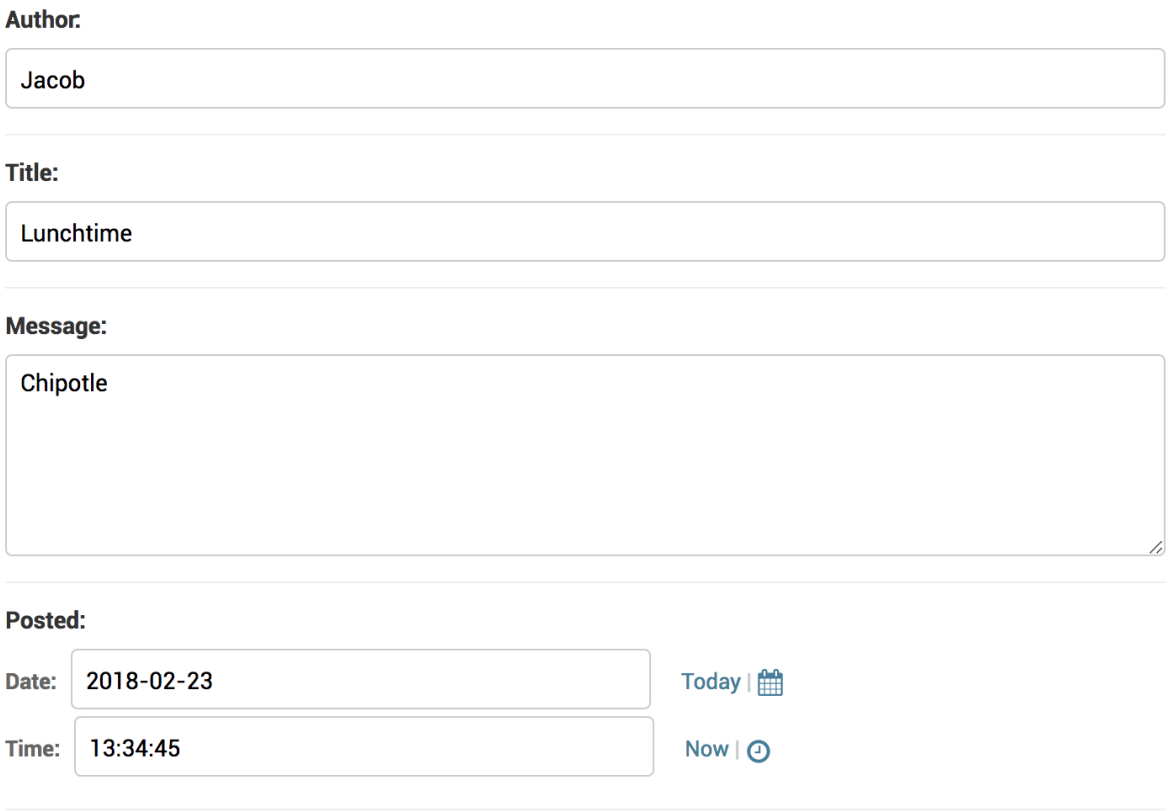
You will be promted to enter a username, email, and password. Remember these, as you will be using them in just a minute. Before being able to login, we register the model class we've created in our `admin.py` file as follows.

```
from django.contrib import admin
from .models import Post

admin.site.register(Post)
```

Now, run our server and head to 127.0.0.1:8000/admin. Hopefully after logging in, you will see the following:



Go ahead and add a few posts with arbitrary information such as:



15.4 Accessing our Data: QuerySets

Once you have a few data fields entered, you can go access this information in the shell. Shut your server down, and install IPython into your virtual enviornment. Next, start IPython up in the terminal running:

```
python manage.py shell
```

Now, we are using python just as we have in a Jupyter notebook. We want to load our model to examine, just as we've imported other objects in the past.

```
from blog.models import Post
```

Now we have access to all the attributes of the Post. Recall that when we defined the Post class, we gave it attributes named `title`, `author`, and `body`. We can display these looping through the Post objects.

```
for title in Post.objects.all():
    print title.title
```

15.5 Blog View

Much like we used the `TemplateView` for our earlier applications, we will use two additional view types that Django has for typical viewing behavior. First, is the `ListView`. This will connect with our data and allow us to list specific pieces of it. Makes sense for a blog homepage.

Create a new view, import the `ListView`, and a blank `base.html` and `home.html` file.

```
from django.views.generic import ListView

from . models import Post

class BlogListView(ListView):
    model = Post
    template_name = 'home.html'
```

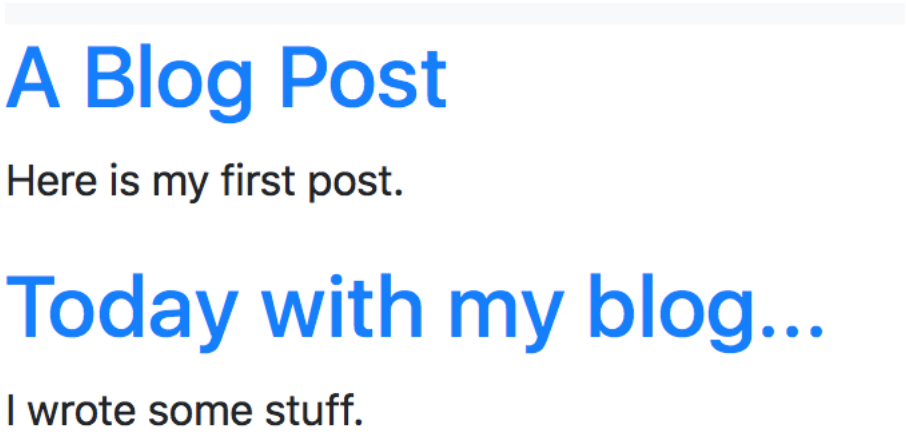
Create the base much as our earlier example, but place the content inside of a `<div>` tag as follows:

```
<div class = "container">
    {% block content %}
    {% endblock content %}
</div>
```

The `ListView` contains an `object_list` that we can use to access the elements of the model in a view, similar to how we accessed them in the shell before. We will do this by looping through the blog entries and displaying the **title** and **body** of the entries.

```
{% block content %}
  {% for post in object_list %}
    <div class="post-entry">
      <h2><a href="">{{ post.title }}</a></h2>
      <p>{{ post.body }}</p>
    </div>
  {% endfor %}
{% endblock content %}
```

Finally, we create a url to our blog, add this to our navigation, and fire up the server. We should see something that looks like a list of our entries with the title and body of the post.



15.6 Adding Individual Blog Pages

While our blog pages now have a home, we would like to link to these pages and see the entire blog entry. To do so, we will create a template named `blog_detail.html` and use a `DetailView` to display the details of the blog content. We need three things here; a view for the detail pages, a template for them, and a url that maps to these.

The view for the individual blogs should feel familiar. We import the `DetailView` and create a class based view with the template named `blog_detail.html`.

```
class BlogDetailView(DetailView):
    model = Post
    template_name = 'blog_detail.html'
```

Next, we can create our template in the templates folder named `blog_detail.html`. We will ask for the detail `object_list` containing the model elements and return the **title** and **body** of the blog.

```
{% block content %}

    <div class="post-entry">
      <h2>{{ post.title }}</h2>
      <p>{{ post.body }}</p>
    </div>

{% endblock content %}
```

Finally, we create the urls. We should recognize that now we are creating a list of urls, unlike our earlier work. We will make use of the fact that Django provides each entry in the database with an index called a **primary key**. In other words, my first blog post has primary key 1, my second 2, and so on. Thus, we can create urls based on these indicies as follows.

```
from django.urls import path, include

from . import views

urlpatterns = [
    path('blog/', views.BlogListView.as_view(), name = 'blog'),
    path('blog/<int:pk>/', views.BlogDetailView.as_view(),
        ]
```

In a similar manner, we can head over to our templates and attach href values to these titles based on the primary key as follows:

`html <a href="{% url 'blog_detail' post.pk %}">Title</a>`