# Homework 1: Intro to Python

Welcome to homework 1! Please complete this notebook by filling in the cells provided.

**Recommended Reading:**

- Chapter 1: What is Data Science
- Chapter 2: Programming in Python

For all problems that you must write explanations and sentences for, please provide your answer in the designated space.

**Deadline:**

This assignment is due **Wednesday, July 02 at 10pm in Gradescope**.

Directly sharing answers is not okay, but discussing problems with the course staff or with other students is encouraged. Refer to the policies page to learn more about how to learn cooperatively.

You should start early so that you have time to get help if you're stuck. The drop-in office hours schedule can be found on class_01_lot_1 slides posted on Canvas. You can also post questions or start discussions on Ed Discussion.

## Getting started

In order to complete the homework, it is necessary to download a few files. Please run the code below **only once** to download data needed to complete the homework. To run the code, click in the cell below and press the play buttom (or press shift-enter).

In [1]:
```
# If you are running this notebook in Google Colab, please uncommen
# (if you are using the YCRC JupyterHub server or Anaconda on your 

# !pip install https://github.com/lederman/YData_package/tarball/ma
```

In [2]:
```
# Please run this code once to download the data files you will nee

import YData

YData.download.download_data("moby_dick.txt")
YData.download.download_data("okc_self_summaries.csv")
```

## 0. Quote and reaction

One of this week's reading is a blog post from the dating website OkCupid about the "The Big Lies People Tell In Online Dating". Please read the blog post, and write down a quote that you find interesting. In the space below, write down the quote as well as one paragraph description for why you thought the quote was interesting.

There are other posts from this blog you might also find interesting. I would recommend you look at the original blog from ~2010 rather than the more recent blog posts as the older posts have a more data science focus, even if they are a bit dated (pun intended).

**Question 0.1 (5 points)** Please write down your "quote and reaction" here.

*Quote:* Time waits for no man, unless that man doesn't update his personal information.

Reaction: This sentence make a good conclusion about the blog with humorous, which attracts me a lot. I really like those people who make their research easy and enjoyable to read.

# 2. Names, Assignment Statements and Error Messages

Let's now start looking at Python programming!

When programming in Python one often makes mistakes which leads to the Python interpreter returning an error message. While at first these might look scary, soon you should be able to make sense of what the message is telling you and learn how to fix your code to remove the mistakes.

Let's start by exploring some error messages! Please look at the code below and answer the question about what is causing the error message.

**Question 2.1 (3 points)** When you run the following cell, Python produces a cryptic error message.

In [1]:
```
4 = 2 + 2
```

```
  Cell In[1], line 1
    4 = 2 + 2
    ^
SyntaxError: cannot assign to literal here. Maybe you meant '==' ins
tead of '='?
```

Choose the best explanation of what's wrong with the code, and then assign 1, 2, 3, or 4 to `names_q1` below to indicate your answer.

1. Python is smart and already knows `4 = 2 + 2`.
2. `4` is already a defined number, and it doesn't make sense to make a number be a name for something else. In Python, "`x = 2 + 2`" means "assign `x` as the name for the value of `2 + 2`."
3. It should be `2 + 2 = 4`.
4. I don't get an error message. This is a trick question.

In [ ]:
```
# Answer
names_q1 = 2
```

**Question 2.2 (3 points)** When you run the following cell, Python will produce another cryptic error message.

In [2]:
```
two = 3
six = two plus two
```

```
  Cell In[2], line 2
    six = two plus two
                  ^
SyntaxError: invalid syntax
```

Choose the best explanation of what's wrong with the code and assign 1, 2, 3, or 4 to `names_q2` below to indicate your answer.

1. The `plus` operation only applies to numbers, not the word "two".
2. The name "two" cannot be assigned to the number 3.
3. Two plus two is four, not six.
4. The name `plus` is not defined as an operator here.

In [ ]:
```
# Answer
names_q2 = 4
```

**Question 2.3 (3 point)** When you run the following cell, Python will, yet again, produce another cryptic error message.

In [3]:
```
x = print(5)
y = x + 2
```

```
5
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[3], line 2
      1 x = print(5)
----> 2 y = x + 2

TypeError: unsupported operand type(s) for +: 'NoneType' and 'int'
```

Choose the best explanation of what's wrong with the code and assign 1, 2, 3, or 4 to `names_q3` below to indicate your answer.

1. You cannot add the letter x with 2.
2. The `print` operation is meant for displaying values to the programmer, not for assigning values.
3. Python doesn't want `y` to be assigned.
4. What error message?

In [ ]:
```
# Answer
names_q3 = 2
```

# 3. Basic statistics: Differences between Universities

**Question 3.1 (8 points)** Suppose you'd like to *quantify* how *dissimilar* two universities are, using three quantitative characteristics. The US Department of Education data on Yale and Cal describes the following three traits (among many others):

| Trait | Yale | Cal |
|---|---|---|
| Average annual cost to attend ($) | 16,341 | 15,240 |
| Graduation rate (percentage) | 97 | 94 |
| Socioeconomic Diversity (percentage) | 20 | 23 |

You decide to define the dissimilarity between two universities as the maximum of the absolute values of the 3 differences in their respective trait values.

Using this method, compute the dissimilarity between Yale and CAL. Name the result `dissimilarity`. Use a single expression (a single line of code) to compute the answer. Let Python perform all the arithmetic (like subtracting 97 from 93) rather than simplifying the expression yourself. Be sure that the final

answer is printed out in the .pdf file you upload to show that you have the correct answer.

Hint: The built-in `abs()` function takes absolute values.

In [4]:
```
# Answer
dissimilarity = max(abs(16341-15240),abs(97-94),abs(20-23))
dissimilarity
```

Out[4]:
```
1101
```

# 4. Lists

One of the most widely use data structure in Python is the list. As we discussed in class, lists can hold any type of item, and we can use the square brackets `[]` to create, set, and extract elements from lists.

Let's explore a few simple practice exercieses using lists now!

**Question 4.1 (3 points)**

To start, create a list called `colleges` that has the following 5 character strings in the order specified here: Branford, Davenport, Morse, Saybrook, and Trumbull. Also, check print out that your list has 5 elements by using the `len()` function.

In [17]:
```
# Answer:
colleges = ['Branford','Davenport','Morse','Saybrook','Trumbull']
len(colleges)
```

Out[17]:
```
5
```

**Question 4.2 (3 points)** Now write one line of code that prints out the 3rd string in this list.

In [18]:
```
# Answer:
colleges[2]
```

Out[18]:
```
'Morse'
```

**Question 4.3 (3 points)** Next, write one line of code that prints out a list that has every other item in the `colleges` list, starting with the first item; i.e., it should return the list that has Branford, Morse and Trumbull.

In [19]:
```
# Answer:
colleges[0:5:2]
```

Out[19]:
```
['Branford', 'Morse', 'Trumbull']
```

**Question 4.4 (3 points)** Finally, write one line of code that replaces the string "Davenport" with the string "Silliman", and print out this modified `colleges` list.

In the answer section, report if this list of colleges seems better to you than the original list (or let us know if you don't have an opinion on this matter).

In [20]:
```
# Answer:
colleges[colleges.index("Davenport")] = "Silliman"
print(colleges)
```

```
['Branford', 'Silliman', 'Morse', 'Saybrook', 'Trumbull']
```

**ANSWER**: I do not have any opinion on this matter because I am not familiar with both colleges.

# 5. Character strings and text manipulation

Many data science insights can be extracted from text. Additionally, frequently before one can analyze data, one needs to "clean the data" which means putting it into a more useable format, and text manipulation is often a large part of data cleaning.

As, we discussed in class, text in Python is held in character strings. Let's explore some of the basics of manipulating character strings by looking at the book "Moby-Dick" which was written by Herman Melville and first published in 1851.

The code below loads the whole book into Python as a single character string named `book_data`. It also prints the letters (i.e. characters) in the book using the `len()` function. You will analyze this `book_data` string in this part of the homework

In [21]:
```
file = open('moby_dick.txt', 'r', encoding="utf8")
book_data = file.read()

# print the number of letters (i.e. characters) in the book
len(book_data)
```

Out[21]: 1191853

**Question 5.1 (3 points)** To start, print out the first 100 characters (i.e., letters) in the book.

Hint: The syntax to do this is similar to printing out the first 100 items of a list.

In [26]:
```
# Answer:
book_data[0:100]
```

Out[26]:
```
'\n\nMOBY-DICK;\n\nor, THE WHALE.\n\nBy Herman Melville\n\n\n\nCHA
PTER 1. Loomings.\n\nCall me Ishmael. Some year'
```

**Question 5.2 (5 points)** Let's now examine how many chapters the book has. To do this, let's count how many times the substring "\nCHAPTER" occurs in the book. In particular, we can use the `my_string.count(my_substring)` method where:

- `my_string` : refers to the longer piece of text we want to seach through
- `count()` is the name of the method we are using to count how many times `my_substring` occurs
- `my_substring` is the string we are searching for and counting in the longer piece of text given by `my_string`

Please go ahead and write one line of code that prints out how many times the string "\nCHAPTER" occured in the text.

In [28]:
```
# Answer:
book_data.count('\nCHAPTER')
```

Out[28]: 135

**Question 5.3 (5 points)** The book Moby-Dick is a book about whaling. Please write code to count how often is the word 'whale' occurs in the book. Make sure your code counts the number of occurances of the string "whale" regardless of how it is capitalized; i.e., it should count "whale", "Whale", "whAle" etc.

Try to make your code as short as possible (e.g., only one line of code). Hint: Think about ways to transform the text so you can easily detect the world "whale" regardless of the case.

In [32]:
```
# Answer:
book_data.lower().count('whale')
```

Out[32]: 1586

**Question 5.4 (5 points)** Now split the book up into a list named `book_chapters` , where each item of the list consists of the text of one chapter of the book.

Then, print out the length of the number of elements in the `book_chapters` list, which should be close to the number of chapters in the book.

Hint: the `my_string.split(my_substring)` could be useful here.

In [36]:
```
# Answer:
book_chapters = book_data.split('\nCHAPTER')
```

```
len(book_chapters)
```

Out[36]: 136

**Question 5.5 (5 points)** Next extract the text from the first chapter and store it in the name `chapter_1`. Then print out the first 500 characters of chapter 1.

In [43]:
```
# Answer:
chapter_1 = book_chapters[1]
chapter_1[0:500]
```

Out[43]:
```
' 1. Loomings.\n\nCall me Ishmael. Some years ago—never mind how l
ong precisely—having\nlittle or no money in my purse, and nothing
particular to interest me\non shore, I thought I would sail about
a little and see the watery part\nof the world. It is a way I have
of driving off the spleen and\nregulating the circulation. Wheneve
r I find myself growing grim about\nthe mouth; whenever it is a da
mp, drizzly November in my soul; whenever\nI find myself involunta
rily pausing before coffin warehouses, and\nbri'
```

**Question 5.6 (5 points)** You will notice that when you print the first 500 characters there are a lot of "\n" characters which correspond to a new line of text. Please write one line of code that removes these "\n" characters from the text and saves the results back to the name `chapter_1`. Then print out the first 500 characters of chapter 1 again.

In [44]:
```
# Answer:
chapter_1 = chapter_1.replace('\n','')
chapter_1[0:500]
```

Out[44]:
```
' 1. Loomings.Call me Ishmael. Some years ago—never mind how long
precisely—havinglittle or no money in my purse, and nothing partic
ular to interest meon shore, I thought I would sail about a little
and see the watery partof the world. It is a way I have of driving
off the spleen andregulating the circulation. Whenever I find myse
lf growing grim aboutthe mouth; whenever it is a damp, drizzly Nov
ember in my soul; wheneverI find myself involuntarily pausing befo
re coffin warehouses, andbringing up '
```

# 6. Exploring text further (5 points)

In part 1 of this homework you read an article analyzing data from the OkCupid website. Let's now look at some of this data ourselves!

When filling out an OkCupid profile, users have to fill in a "My self summary" box where users describe themselves. An example of what an OkCupid profile looks like can be seen here).

The code below loads the respones that 54,456 OkCupid users wrote as their "My self summary" into a list of strings called `self_summaries`, where each

string corresponds to one OkCupid user's "My self summary" response. Please explore the data and see if you can find anything interesting in it! Then, in the answer section write down anything interesting you find.

**Note**: There is a fair amount freedom to the degree that you challenge yourself here, and we will be relatively relaxed on the grading of this question. If you find the questions in this homework were easy, feel free to read about additional features of Python and experiment with these features to challenge yourself and to come up with an more interesting answer. If you found the above homework challenging, then just come up with a simple analysis (e.g., a couple lines of code), and make sure you understand all the code we have covered in class so far. At the moment we are a bit limited in the types of analyses since we have just started learning the basics of Python, but soon we will learn more methods that will allow you to answer much more interesting questions.

**Tip**: It could also be useful to join strings in the `self_summaries` list into a single string called `summaries_string`. You can do this using the following line of code `summaries_string = " ".join(my_self_summaries)`.

In [45]:
```
import pandas as pd

self_summaries = pd.read_csv("okc_self_summaries.csv")["my_self_sum
```

In [55]:
```
# Put your analysis code here
# Answer:
summaries_string = "".join(self_summaries)
summaries_string = summaries_string.lower()
words = summaries_string.split(" ")
word_counts = {}
for word in words:
    if word in word_counts:
        word_counts[word] += 1
    else:
        word_counts[word] = 1
sorted_word_counts = sorted(word_counts.items(), key=lambda x: x[1]
sorted_word_counts[0:100]
```

Out[55]:
```
[('i', 231061),
 ('and', 213763),
 ('to', 182534),
 ('a', 171249),
 ('the', 138020),
 ('in', 95611),
 ('of', 88348),
 ('/>\n<br', 80231),
 ('my', 74097),
 ('am', 52678),
 ('for', 50953),
 ("i'm", 49574),
 ('is', 45519),
```

```
('have', 44111),
('that', 43860),
('with', 43556),
('/>\ni', 41487),
('but', 40997),
('love', 38680),
('like', 36998),
('you', 32503),
('on', 30385),
('be', 30051),
('me', 27687),
('as', 27545),
('it', 26980),
('not', 26040),
('at', 25059),
('or', 23636),
('who', 22085),
('an', 19711),
('out', 19635),
('are', 19476),
('can', 19450),
('about', 19311),
('so', 18494),
('just', 18326),
('new', 17838),
('if', 17538),
('from', 16539),
('good', 16477),
('<a', 16378),
('all', 15912),
('class="ilink"', 15790),
('up', 15606),
('people', 15578),
('do', 15179),
('this', 14829),
('life', 14104),
('enjoy', 13948),
("don't", 13844),
('time', 13782),
('get', 13322),
('also', 12976),
('what', 12710),
('very', 12357),
('more', 12110),
('things', 12011),
('when', 11947),
('looking', 11906),
('been', 11214),
('really', 11118),
('know', 11083),
('some', 11079),
('was', 11061),
("/>\ni'm", 10953),
('someone', 10722),
('would', 10241),
("i've", 10072),
```

```
 ('one', 9916),
 ('being', 9515),
 ('want', 9416),
 ('go', 9122),
 ('going', 9108),
 ('make', 9035),
 ('much', 8855),
 ('friends', 8851),
 ('think', 8848),
 ('will', 8777),
 ('work', 8649),
 ("it's", 8549),
 ('always', 8537),
 ('by', 8518),
 ('myself', 8159),
 ('most', 8151),
 ('we', 8124),
 ('find', 7981),
 ('than', 7662),
 ('pretty', 7630),
 ('–', 7532),
 ('try', 7401),
 ('live', 7395),
 ('has', 7029),
 ('into', 6993),
 ('bay', 6939),
 ('san', 6872),
 ('years', 6861),
 ('here', 6833),
 ('fun', 6737),
 ('great', 6679)]
```

**ANSWER**: Describe anything interesting that you found in your analysis here.

By reading the top 100 words, we can find out that people are more willing to tell others about what they are interested in or what they always do, rather than their shortcomings or what they do not like. It is easy to understand because people on the social media are always posting things to find people who has similar points with them, rather than find those they hate. Thus it is natual to concentrate on themselves' hobby and personality stuffs when writing the profile.

# 7. Reflection (3 points)

Please finish Homework 1 Reflection Quiz. You can find the Quiz under "Quizzes" in Canvas.

# 8. Submission

Once you're finished filling in and running all cells, you should submit your

assignment as a .pdf on Gradescope. You can access Gradescope through Canvas on the left-side of the class home page. The problems in each homework assignment are numbered. **NOTE:** When submitting on Gradescope, please select the correct pages of your pdf that correspond to each problem. Failure to mark pages correctly will result in points being deducted from your homework score.

If you are running Jupyter Notebooks through an Anaconda installation on your own computer, you can produce the .pdf by completing the following steps:

1. Go to "File" at the top-left of your Jupyter Notebook
2. Under "Download as" (or "Save and Export Notebook As...") and select "HTML (.html)"
3. After the .html has downloaded, open it and then select "File" and "Print" (note you will not actually be printing)
4. From the print window, select the option to save as a .pdf

If you are running the assignment in a Google Colabs, you can use the following instructions:

1. Go to "File" at the top-left of your Jupyter Notebook and select "File" and "Print" (note you will not actually be printing)
2. From the print window, select the option to save as a .pdf
3. Be sure to look over the pdf file to make sure all your code and written work is saved in a clear way.