MIE 1513 Decision Support Systems
Assignment 1: Introduction to Python

January 10, 2022

This assignment involves basic python coding. The purpose of this assignment is to ensure the students have sufficient Python skills to complete the course' assignments.

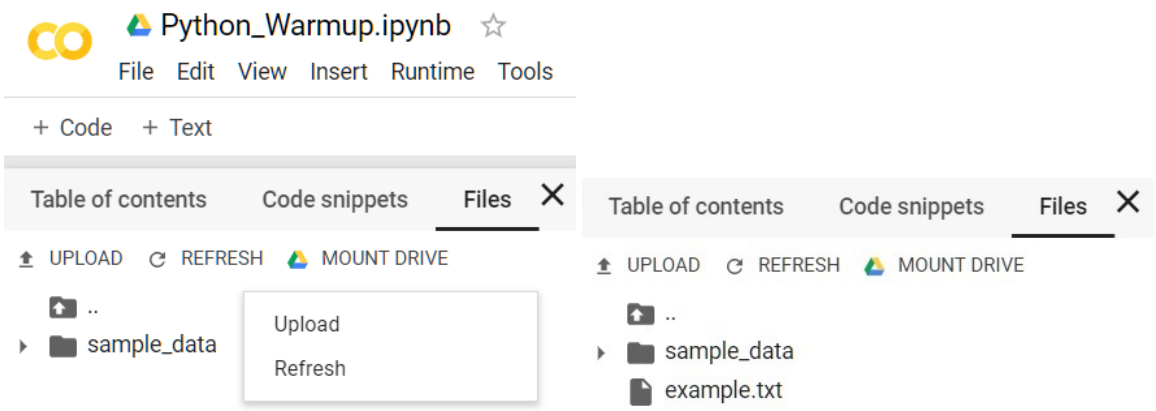
- Programming language: Python (Google Colaboratory)
- Due Date: Posted on Quercus

Marking scheme and requirements: Full marks will be given for (1) working, readable, reasonably efficient, documented code that achieves the assignment goals and (2) for providing appropriate answers to the questions in the provided Jupyter notebook (named `Python_Warmup.ipynb`) committed to the student's assignment repository (from the same place it was checked out).

Please note the plagiarism policy in the syllabus. If you borrow or modify any *multiline* snippets of code from the web, you are required to cite the URL in a comment above the code that you used. You do not need to cite tutorials or reference content that demonstrate how to use packages – you should certainly be making use of such content.

What/how to submit your work:

1. All your code should be written in the provided notebook named `Python_Warmup.ipynb`.
2. All functions must *return* the specified return type; the autograder **ignores** any content printed to the console/screen.
3. Your code should pass the provided validators. **Note: passing the validators does not mean your code is correct. The validators are very basic and are intended to make sure that the interface of your functions is correct.**
4. You will have to upload `example.txt` into Google CoLaboratory as shown below:



If you mount your drive, please comment out or delete the cell containing

```
from google.colab import drive
drive.mount('/content/drive')
```

5. Commit and push your work to your GitHub repository in order to submit it. Your last commit and push before the assignment deadline will be considered to be your submission. You can check your repository online to make sure that all required code has actually been committed and pushed to your repository.

Things to check when submitting your work:

1. Please make sure your notebook runs **end to end** without any errors.
2. The autograder will call your functions individually and verify its output. Therefore, please avoid defining **global variables** outside of the function for each exercise.
3. Please make sure your submitted functions **return** the desired output. Print lines will not be looked at.
4. If you need to use **external libraries** that are not provided in lab, please ask for approval on piazza.

1 Code the following functions:

1.1 Exercise 1:

Write a function `positionWord(filePath, position)` that takes as arguments the path of a file (`filePath`) and an integer(`position`). The function returns a list that includes the word in index `position` in each of the lines in the file. Note that:

1. You can assume that each line has more words than the required `position`.
2. The index of the first word is 0 (the second word is 1, etc).
3. The words in the returned list should be stripped of leading and trailing punctuation/symbols, etc.

1.1.1 Example code:

For the file `example.txt` that is provided in the repository, the following code:

```
positionWord("example.txt", 3)
```

Should return the following list:

```
['in', 'Ghost', 'humour', 'or', 'one']
```

1.2 Exercise 2:

Write a function `onlyUpperCase(text)` that takes in a string `text` as an argument, and returns a string that includes only the upper case characters from the original string.

1.2.1 Example code:

The following code:

```
onlyUpperCase("lorem ipsum Dolor Sit amet, conSectetur.")
```

Should return the following string:

```
'DSS'
```

1.3 Exercise 3:

Write a function `divisors(num1, num2)` that takes two numeric arguments: $num1$ and $num2$ (positive integer numbers). The function returns a list of all the numbers between 1 and $num1$ (including 1 and $num1$) such that $num2$ is a divisor of $num1$, i.e., all the numbers between 1 and $num1$ that can be divided by $num2$ with no remainder.

1.3.1 Example code:

The following code:

```
divisors(23, 6)
```

Should return the following list:

```
[6, 12, 18]
```

1.4 Exercise 4:

We are interested in counting the number of times each word appear in a list. To do so, write a function `countWords(wordList)` that takes a list of lowercase words `wordList` as an argument and returns a dictionary that stores the number of times each word appeared in the list.

1.4.1 Example code:

The following code:

```
countWords(["the", "seething", "sea", "ceaseth",
         "and", "thus", "the", "seething", "sea", "sufficeth", "us"])
```

Should produce the following output (**order of words is not important**):

```
{'and': 1,
'the': 2,
'sea': 2,
'thus': 1,
'ceaseth': 1,
'seething': 2,
'sufficeth': 1,
'us': 1}
```

1.5 Exercise 5:

We now want to make the previous function more organized by storing the word counts by the length of each word. To do so, write a function `organizedCountWords(wordList)` that takes a list of lowercase words `wordList` and returns a dictionary that maps from the length of a word to another dictionary with the word counts of all words of this length.

1.5.1 Example code:

The following code:

```
organizedCountWords(["the", "seething", "sea", "ceaseth",
         "and", "thus", "the", "seething", "sea", "sufficeth", "us"])
```

Should produce the following output (**order of words is not important**):

```
{2: {'us': 1},
3: {'the': 2, 'sea': 2, 'and': 1},
4: {'thus': 1},
7: {'ceaseth': 1},
8: {'seething': 2},
9: {'sufficeth': 1}}
```

1.6 Exercise 6:

Write a function `AvgNeighbourhoodListingPrice(neighbourhoodList, room_type, number_of_reviews, minimum_nights)` that takes in a list of `neighbourhood` string values, a `room_type` string value, as well as `number_of_reviews` and `minimum_nights` integer values. The function imports the following Airbnb listings data:

`https://raw.githubusercontent.com/MIE451-1513-2020/course-datasets/master/toronto_airbnb_listings_Aug2019.csv`

The function should only analyze listings with the given `room_type` value and exclude listings that have less than the given `number_of_reviews` value or more than the given `minimum_nights` value from the analysis. The function should return a one (not two) column Pandas dataframe containing the average listing `price` rounded to two decimal places for each neighbourhood. It should also be sorted by descending order (`neighbourhood` with the highest average listing `price` first).

Hint:

- Use the Pandas `groupby` function to obtain a dataframe that contains only one column of `price` that is indexed by the neighbourhoods: https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.groupby.html
- Use Pandas `read_csv` function to load the url directly without downloading the file with urllib first.

Note that:

1. You can assume that the `neighbourhood` list will contain only neighbourhoods in the dataset.
2. You can assume that the `room_type` string will contain a room type found in the dataset.
3. You can assume that the `number_of_reviews` and `minimum_nights` values will be non-negative integers.

1.6.1 Example code:

The following code:

```
AvgNeighbourhoodListingPrice(["Edenbridge-Humber Valley","Annex",
"The Beaches"],"Entire home/apt", 30, 3)
```

Should produce the following output:

neighbourhood	price
Edenbridge-Humber Valley	453.33
Annex	141.99
The Beaches	126.24

1.7 Exercise 7:

This course presumes that you have had a prerequisite that has covered basic machine learning (specifically, linear regression).

Assume that we are given a 3 column data matrix X with columns for x_1, x_2, x_3 and a target column vector y. For testing, you can use the following two URLS:

`https://raw.githubusercontent.com/MIE451-1513-2020/course-datasets/master/X.csv`
`https://raw.githubusercontent.com/MIE451-1513-2020/course-datasets/master/Y.csv`

For every row index i in this file corresponding to x_1, x_2, x_3, the corresponding row i of column vector y corresponds to the following linear computation: $y = c_0 + \sum_{j=1}^{3} c_j x_j$. We define $w = [c_0, c_1, c_2, c_3]^T$ where c_0 is a constant bias term and c_1, c_2, and c_3 are respectively linear coefficients for x_1, x_2, and x_3. We do not know w, but we can use linear regression to recover the best fitting coefficients w according to a least squares fit.

Write a function that takes as input *strings* for URLs for X and y. The function computes and returns just the best (least squares) fit linear regression coefficients $[c_1, c_2, c_3]^T$ as type Numpy ndarray column vector. The specific computation for this (with correctly defined X and y that you need to figure out) is $w = (X^T X)^{-1} X^T y$ (note that we do not use any regularization here). Note that you should *not* return the value of c_0 in the column vector.

1.7.1 Example code:

The following code:

```
leastSquaresFit(URLX,URLy)
```

Should return a NumPy ndarray with shape (3,1)

```
array([[78.5819959 ],
       [68.68353831],
       [66.40178688]])
```

2 Helpful Links

2.1 Python3

2.1.1 Python 3 official documentation

`https://docs.python.org/3/`

2.1.2 Python 3 official tutorial (with code examples)

https://docs.python.org/3/tutorial/

2.1.3 Free Python book "Dive Into Python 3"

http://www.diveintopython3.net/

2.2 Git

2.2.1 git - the simple guide

http://rogerdudler.github.io/git-guide/

2.2.2 Free Git book: "Pro Git"

https://git-scm.com/book/en/v2

2.3 Environment

2.3.1 Jupyter documentation

http://jupyter.readthedocs.io/en/latest/