**INSY 5336**

**Python Programming**

**Fall 2020**

**Homework 3 (50 points)**

**Due Date: November 18, 2020, 11:59 pm CST (no late submission)**

The following guidelines should be followed and will be used to grade your homework:

* All code to be implemented and submitted as a jupyter notebook (.ipynb) file. Submit a single ipynb file.
* This is an individual homework assignment, no group submissions will be accepted. If you discuss in groups, please write your code individually and submit.
* Sample runs shown in the question should be used as a guide for implementation. However extensive testing needs to be done on your code to deal with all test cases that might possibly be executed.
* The logic of how you are solving the problem should be documented in the cell preceding the code in markdown language. In the case that your code is incorrect, your logic counts as effort points.
* Every code segment in the jupyter notebook cells should be well documented with comments. Use # in the code to provide comments and they should explain the algorithm step and what the code segment is doing. Follow the example in the notebook files provided in the lectures.
* Error checking in your code is **very important** and differentiates a high quality programmer from a low quality one. Use try/except blocks, if statements and other python code constructs to deal with unexpected errors and deal with them gracefully. **The homework will be graded for robustness of your code. You will lose 50% of the points if your code contains error/does not run! You will lose 10% of the points if your code runs but produces wrong result. In the second situation, you will gain some points back if your logic is clear and correct.**

1. (25 points) Write a program to request a file name from the user and calculate the following statistics of the contents of the file:

* Number of lines
* Number of words
* Number of characters
* Average length of a word

In this problem use the following definitions:

A line is a sequence of characters that end with a newline (\n) character

A word bounded by one or more spaces (or \n) on either side of it (or both sides)

A character is any single length string, e.g. ‘a’, ‘-‘, etc. but not a space (or white space)

An example file called julius\_ceasar.txt is included in the homework files

Sample Run

*What is the filename: julius\_ceasar.txt*

*Number of lines: 23*

*Number of words: 177*

*Number of characters: 802*

*Average length of a word: 4.531073446327683*

2. (25 points) An example file called movie\_reviews.txt is included in the homework files.

The file contains names of reviewers and their reviews of different movies in the comma separated format. The first string is the name of the reviewer followed by a name of a movie and its rating. You are required to write a Python program that computes a similarity score between any two reviewers using Euclidean distances. An example data from the movie\_reviews.txt file is given below:

Trevor Chappell’s reviews are:

'Lawrence of Arabia': 3.0,

'Gravity': 4.0,

'The Guns of Navarone': 3.0,

'Prometheus': 5.0,

'For a Few Dollars More': 3.5

Peter’s reviews are:

'Gravity':4.5,

'For a Few Dollars More':1.0,

'Prometheus':4.0

The Euclidean distance between Peter and Trevor Chappell is computed as follows:

1. Only consider the movies that both have reviewed.
2. Take the difference between corresponding reviews.
3. Sum the square of the differences
4. The square root of the sum of differences is the Euclidean score. The shorter the distance the closer the two reviewers.

For our example, this would be: (4.0 – 4.5)2  + (3.5 – 1.0)2 + (5.0 – 4.0)2 = 0.25 + 6.25 + 1 = 7.5

Euclidean distance is Square Root of 7.5 = 2.7386

Your program should provide the following:

1. Ability to read in a user provided filename which contains movie reviews in the comma separated format similar to the given file movie\_reviews.txt
2. Functionality to compute the similarity between two reviewers provided by the user
3. Functionality to compute the similarity between one user provided reviewer and all other reviewers in the database

A sample run for the two required functionalities is given below (user inputs are in red):

**Give the name of the movie reviews file: movie\_reviews.txt**

**What do you want to do? Input 1 for similarity between two reviewers, or Input 2 for similarity between one reviewer and all others in the database or 3 to quit: 1**

**Provide Reviewer1 name:** Peter

**Provide Reviewer2 name:** Trevor Chappell

**The similarity score between Peter and Trevor Chappell is:** 2.7386

**What do you want to do? Input 1 for similarity between two reviewers, or Input 2 for similarity between one reviewer and all others in the database or 3 to quit: 2**

**Provide Reviewer name:** Peter

**The Similarity Scores are:**

**Peter Nancy Pollock 1.87**

**Peter Jill Brown 1.50**

**Peter Jack Holmes 2.87**

**Peter Trevor Chappell 2.74**

**Peter Mary Doyle 1.58**

**Peter Doug Redpath 1.80**

**What do you want to do? Input 1 for similarity between two reviewers, or Input 2 for similarity between one reviewer and all others in the database or 3 to quit: 3**

**Goodbye!**

Hint:

1. A good way to convert the file contents to a Python data structure is to create a dictionary. For example the first two lines of the movie\_reviews.txt file can be stored as a dictionary entry such as the following:

{'Nancy Pollock': {'Lawrence of Arabia': 2.5, 'Gravity': 3.5,

'The Godfather': 3.0, 'Prometheus': 3.5, 'For a Few Dollars More': 2.5,

'The Guns of Navarone': 3.0},'Jack Holmes': {'Lawrence of Arabia': 3.0, 'Gravity': 3.5, 'The Godfather': 1.5, 'Prometheus': 5.0, 'The Guns of Navarone': 3.0,

'For a Few Dollars More': 3.5}}