**INSY 5336**

**Python Programming**

**Spring 2023**

**Homework 2 (100 points)**

**Deadline April 30th (No exceptions)**

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.
* 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 high-level algorithm of how you are solving the problem should be documented in the cell preceding the code in markdown language.
* The instructions for running of each cell and the expected results should be documented in the cell preceding the code using markdown language.
* 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.
* Error checking in your code is very important and differentiates a high-quality programmer from a low quality one. Hence you should account for invalid user inputs, infinite loops, out of range results, etc. and resolve them by appropriate error messages. **The homework will be graded for robustness of your code.**

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 robertfrost.txt is included in the homework files

Sample Run

*What is the filename: robertfrost.txt*

*Number of lines: 4*

*Number of words: 27*

*Number of characters: 132*

*Average length of a word: 3.8*

Note: If your file statistics are different from the answer I have given above, please explain in your notes/markdown script how you arrived at your answers. For example if you use the readlines() function then it will count the last line which does not end with a newline (\n) as a line, that is fine as long as you understand it and are able to explain.

2. (25 points) A string is an anagram of another if the second string is simply a scrambled version of the first. Write a python program to implement the following game:

1. Reads in a file that has words and their meanings in a text file. An example “words and their meanings” file is given in canvas. Note that your program needs to ask the user for the “words and their meanings” file to use.
2. The words and their meanings text file is of the csv (comma separated values) format. Use either notepad++ or notepad to create your file in the same format as the mywords.txt file given to you in canvas
3. Your program should then pick a word from the “words and their meanings” file, scramble the letters, and ask the user to unscramble it. Every run of your program should pick a word at random.
4. The user may type in the unscrambled word or may ask for the definition/meaning of the word by entering a question mark.
5. The game continues for the number of times as the number of letters in the word. For example if the word is “poor”, the program will allow four attempts.
6. The program will also provide the definition of the word (at the user’s request), only once. If the user enters “?” more than once, an error message will be given with the warning that the next input of a “?” will be counted as an attempt at the answer.
7. until the user says “no” to the question: “Do you want to continue?”. A sample run is shown below (user inputs are in red):

**Give the name of the “words and their meanings” file: mywords.txt**

**Do you want to play: Y**

**Unscramble the following letters to form a word. Type “?” for the meaning of the unscrambled word: ulecenop**

**Enter the answer [or ? for the meaning]: ?**

**The word means: great wealth**

**Enter the answer [or ? for the meaning]: rich**

**Wrong, try again**

**Enter the answer [or ? for the meaning]: rich**

**Wrong, try again**

**Enter the answer [or ? for the meaning]: rich**

**Wrong, try again**

**Enter the answer [or ? for the meaning]: rich**

**Wrong, try again**

**Enter the answer [or ? for the meaning]: rich**

**Wrong, try again**

**Enter the answer [or ? for the meaning]: rich**

**Wrong, try again**

**Enter the answer [or ? for the meaning]: rich**

**Wrong, try again**

**Enter the answer [or ? for the meaning]: rich**

**Wrong, you have exceeded the number of attempts. Bye!**

**Do you want to play: Y**

**Unscramble the following letters to form a word. Type “?” for the meaning of the unscrambled word: yrnuep**

**Enter the answer [or ? for the meaning]: injury**

**Wrong, try again**

**Enter the answer [or ? for the meaning]: ?**

**The word means: extremely poor**

**Enter the answer [or ? for the meaning]: ?**

**You have been given the meaning before. Next time you ask for the meaning it will count as an attempt!**

**Enter the answer [or ? for the meaning]: poverty**

**Wrong, try again**

**Enter the answer [or ? for the meaning]: penury**

**You got it!**

**Do you want to play: N**

**Goodbye!**

Hint: a) The easiest way to store the “words and their meanings” file would be in a dictionary. You may have a dictionary that looks like the one shown below:

{ “opulence”:”great wealth”, “penury”:”extremely poor”, …..}

b) Please create your own “words and their meanings” file to test your code. I will be using my own test file to test your homework submission. Do not hard code filenames in your program.

3. (25 points) The file poetry\_lines.txt is given to you (see Canvas for the file poetry\_lines.txt).

The file contains names of poets and an extract of their poetry. New lines in each poem are represented by a ‘/’. The format of a line is the following:

<Poet’s name>:<poetry delineated by ‘/’>\n

The first string in each line is the name of the author followed by a ‘:’, followed by the poetry which is delineated by ‘/’ to represent a new line in the poem.

The next line contains the next poem and so on.

You are required to input a few lines of your own poem to the python program (with lines separated by “/”) and compute the cosine distance (similarity score) between each line (of poetry from the file) and your own poem. Finally your program should display the following:

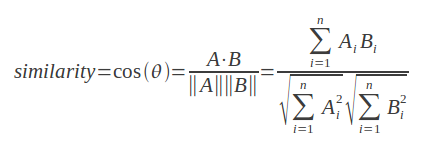
1. Each poet and the similarity score with your poem.
2. Finally display the poem that is closest to your input.

An example of how to compute the cosine distance between two lines of text is given below:

Line1: Hi Hi Hi how are you

Line2: Hi how are u u

1. The cosine similarity between two vectors (A and B) is given by the formula:



1. Create a dictionary of keys with the words and the values with the number of occurrences.
   1. Line1dict = {‘Hi’:3,’how’:1,’are’:1,’you’:1,’u’:0}
   2. Line2dict = {{‘Hi’:1,’how’:1,’are’:1,’you’:0,’u’:2}
2. The values in the two dictionaries form the two vectors to be compared. For example:
   1. Line1vector = (3,1,1,1,0)
   2. Line2vector = (1,1,1,0,2)
3. Now you can implement the cosine similarity formula above to compare Line1vector and Line2vector.
4. Note that the cosine distance should be between 0 and 1.

Your program should provide the following:

1. Ability to read in a user provided filename which contains names of poets and an extract of their poetry in the specified format.
2. Read in user’s own poem (using the input() method).
3. Functionality to compute the similarity (use cosine distant) between each poem and the user’s poem and display the results.
4. Finally choose and display the poem that is most similar to the user’s input.

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

**Give the name of the poetry file:** poetry\_lines.txt

**Input your poem delineated by “/” for each line:** Whose forest are these I know./He lives in the village/He has no idea I am looking at his property/and seeing the forest fill up with snow.

**Cosine distance results:**

William Wordsworth: xx

William Shakespeare: yy

Robert Frost: zz

The closest poem is:

Robert Frost: Whose woods these are I think I know./His house is in the village though;/He will not see me stopping here/To watch his woods fill up with snow.

Hints:

1. Use Python dictionaries to store the word and the number of times it occurs in the poetry or in the user poem.
2. Work with each line from the poetry\_lines.txt and the user input. Create one vector for the line of poem and another vector for the user input. Note that each vector should have the same words, but the number of times they appear in the poem and user input will be different.
3. Now compute the cosine distance between these two vectors. Use an ordered datatype such as lists to represent the final vectors for cosine calculations.
4. Repeat the steps for all the lines in the poetry\_lines.txt file and calculate the output values.

4. (25 points) This is a project to scrape data from the web and store the results in a text file and the SQLite database.

The website <https://finance.yahoo.com/trending-tickers> lists extensive finance data. You have to write Python scripts/programs to collect the current prices for the following commodities: Crude Oil, Gold and Silver. Your program should store the commodity name and its corresponding price in a text file called **commodity\_prices.txt**.

In addition to the commodity\_prices.txt file, the data should also be stored in an **SQLite** database called **CommodityDatabase** in the directory that your Jupyter Notebook code will be executed from. The CommodityDatabase should have a table called **CommodityTable** that contains the following columns and types:

Ticker TEXT

Price REAL

Every execution of your program should create a new commodity\_prices.txt and CommodityDatabase.db file in the directory (delete any existing files that you will create) that your Python script is located and run.

Testing Instructions:

Verify commodity\_prices.txt file is created with the commodity name and its corresponding price

Verify CommodityDatabase is created in SQLite database having table named CommodityTable and the following columns and types with correct data populated:

Ticker TEXT

Price REAL