CSC 401 ASSIGNMENT TW0

Due Date: Tuesday, Aug. 4th by 11:58 PM

The purpose of this assignment is to assess your understanding of

- Creating user defined functions that may take arguments (parameters)
- Using For loop iterations
- Reading and writing text files.

SUBMISSION

- **Include your full name as a comment in the first line of your Python program**
- **Include the problem number as a comment in the second line of our Python program**
- Save each program to a separate file labeled as YourName_hw2_1.py, YourName_hw2_2.py, YourName_hw2_3.py.
- Upload each file to Submissions folder in D2L.
- DO NOT UPLOAD FileStats.txt. This is the file you will create in Problem 2

PROBLEMS

Note: you may not use Python statements, functions, data types, etc. that where not discussed in the reading assignment or the lecture notes/videos for week 2 or previous week. This is a class for students who have not programmed before and I expect everyone to code on the same level. If you have a better way of writing the code, then upload two versions: one that codes according to the specifications and the other that demonstrates advanced programming techniques.

I encourage you to

- Use computational thinking to solve the problems. These are straight-forward solutions, but developing a good habit of analyzing the problem and describing the steps will serve you well as the problems get more complex
- Test your code thoroughly. You are **not** required to include your test cases as part of your programs. But, **I do require that you code according to the problem specifications.**

PROBLEM ONE (10 POINTS)

USER-DEFINED FUNCTION, LIST, FORMATTING, DECISION, ITERATION (FOR LOOP)

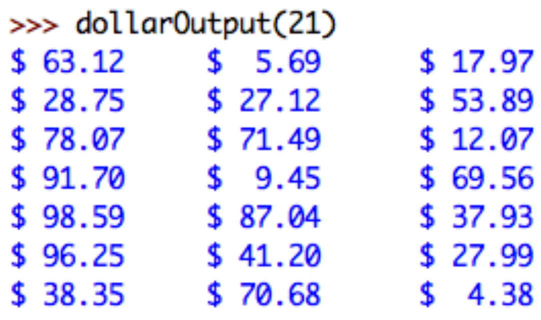
Write a Python user-defined function dollarOutput(n) that accepts an integer, n, prints the n values of dollar amounts as shown in the sample below. The output must have 3 columns of data. There should be 5 spaces between the columns. Each value must be right-aligned in each column. The values should be formatted as dollar and cents.

Start your function with the following code:

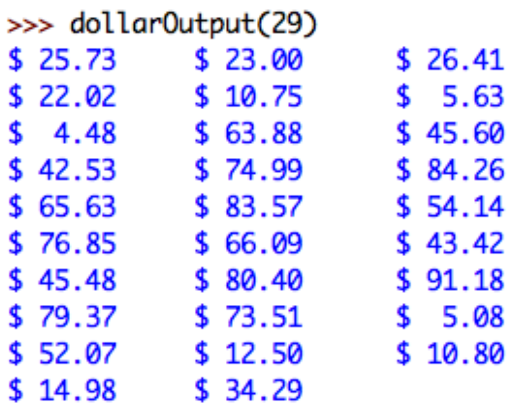
```
def dollarOutput(n):
    import random
    lst = []
    # lst is a list that containa 100 float numbers
    for i in range(100):
        lst.append(random.uniform(0,100))
```

This code will produce 100 random float numbers between 0 and 100 and store them in the list, lst. I suggest you print lst and examine its' contents. To complete this function, you are to write the code to get n numbers from lst and format as shown in the samples below.

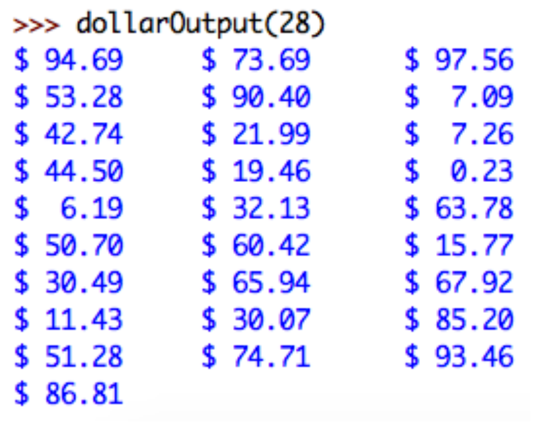
Samples:



```
>>> dollarOutput(21)
$ 63.12     $  5.69     $ 17.97
$ 28.75     $ 27.12     $ 53.89
$ 78.07     $ 71.49     $ 12.07
$ 91.70     $  9.45     $ 69.56
$ 98.59     $ 87.04     $ 37.93
$ 96.25     $ 41.20     $ 27.99
$ 38.35     $ 70.68     $  4.38
```



```
>>> dollarOutput(29)
$ 25.73     $ 23.00     $ 26.41
$ 22.02     $ 10.75     $  5.63
$  4.48     $ 63.88     $ 45.60
$ 42.53     $ 74.99     $ 84.26
$ 65.63     $ 83.57     $ 54.14
$ 76.85     $ 66.09     $ 43.42
$ 45.48     $ 80.40     $ 91.18
$ 79.37     $ 73.51     $  5.08
$ 52.07     $ 12.50     $ 10.80
$ 14.98     $ 34.29
```



```
>>> dollarOutput(28)
$ 94.69     $ 73.69     $ 97.56
$ 53.28     $ 90.40     $  7.09
$ 42.74     $ 21.99     $  7.26
$ 44.50     $ 19.46     $  0.23
$  6.19     $ 32.13     $ 63.78
$ 50.70     $ 60.42     $ 15.77
$ 30.49     $ 65.94     $ 67.92
$ 11.43     $ 30.07     $ 85.20
$ 51.28     $ 74.71     $ 93.46
$ 86.81
```

Note: There is only one function in this program. To run the function, type dollarOutput(n) replacing n with a positive integer, at the IDLE prompt. Your function should work correctly with any integer n greater than 0 and less then 101.

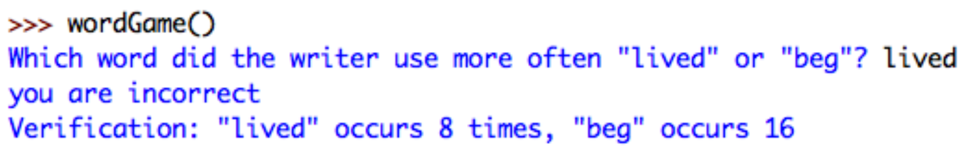
PROBLEM TWO (15 POINTS)
READ FILE, FOR LOOP, USER-DEFINED FUNCTION

Write a user-defined function wordGame() that reads the text file, Pride_and_Prejudice.txt. (Download the file Pride_and_Prejudice.txt from D2L in the assignment module. Save this file in the same folder in which you are saving your program.) Include code in this function that retrieves 2 random words from the file. Ask the user to guess which of the words the author used more often in the text.   If the user picks the word with the higher count, then they have guessed correctly.   Include a statement verifying the results.

See sample case for format. Enclose the chosen words in double quotes (") so they are easily distinguishable from the rest of the message.

To pick a random word, import random and use the random.choice () method. To determine how often a word occurs (use the count method). Use a logical condition to determine if the user guessed correctly. As shown in the sample cases, be sure to include the chosen words in double quotes.

Sample cases:



```
>>> wordGame()
Which word did the writer use more often "lived" or "beg"? lived
you are incorrect
Verification: "lived" occurs 8 times, "beg" occurs 16
```

```
>>> wordGame()
Which word did the writer use more often "than" or "down,"? down,
you are incorrect
Verification: "than" occurs 279 times, "down," occurs 7

>>> wordGame()
Which word did the writer use more often "parted;" or "had"? had
you are correct
Verification: "parted;" occurs 2 times, "had" occurs 1130
```

Note: There is only one function in this program. To run the function, type wordGame() at the IDLE prompt.

PROBLEM 3 (20 POINTS)
READ/WRITE FILE, ACCUMULATOR (FOR LOOP), USER-DEFINED FUNCTION

Write a function **getStats(fname)** that determines the following statistics for the input file, fname, and write the results to the text file, FileStats.txt.

- Number of occurrences of each day of the week in the file.
- Number of lines (determined by end of line character '\n')
- Number of words
- Numbers of characters (excludes spaces between words)

The output from getStats() should

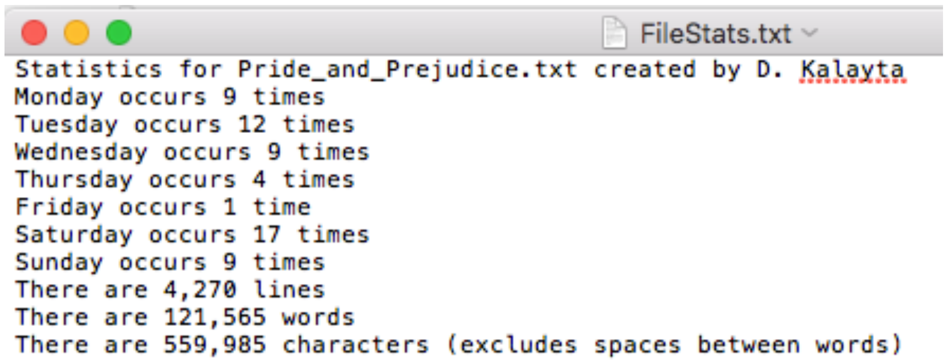
- print the message 'To view the results go to FileStats.txt'
- return 'Thanks for using the getStats() function'

Include the name of the file and your name in the first line of FileStats.txt. For example, I ran getStats("Pride_and Prejudice.txt") and got the following results:

```
>>> getStats('Pride_and_Prejudice.txt')
To view the results go to FileStats.txt

'Thanks for using the getStats() function'
```

To view the contents of FileStats.txt, go to the folder where your program is saved and, double click on FileStats.txt. Your file should be formatted as shown below.



```
Statistics for Pride_and_Prejudice.txt created by D. Kalayta
Monday occurs 9 times
Tuesday occurs 12 times
Wednesday occurs 9 times
Thursday occurs 4 times
Friday occurs 1 time
Saturday occurs 17 times
Sunday occurs 9 times
There are 4,270 lines
There are 121,565 words
There are 559,985 characters (excludes spaces between words)
```

The function should be able to handle any file the user specifies when calling the function. I will test with a different file and I expect to see the name of that file in the first line of FileStats.txt

Notes:

- Create a list to contain the days of the week, Monday, Tuesday, Wednesday, Thursday, Friday and Saturday.
- Use the days of the week as written in the sample output. Do NOT create seven separate write statements.
- Do not be concerned about case of the letters or punctuation. Count only days as shown in the file snapshot.
- The file read() method returns a string. Use this string to count the number of occurrences of each day of the week in your list.
- To determine the number of lines in a file, count the number of occurrences of the end of line character '\n' in the string.
- To determine the number of words and characters, use the string split() method to create a list of words. Use this list to accumulate the number of words and the number of characters in each word.
- This function should have only one read() statement.
- Close the input and output files.

Assignment Two Grading Rubric

Learning outcomes:

- Create user-defined functions
- Utilize string methods in text processing
- Read text file from within a Python program
- Write text file from within a Python program
- Use the String method format() to improve the readability of the output string
- Use iteration structure to repeat code
- Use methods from random library

Problem Numbers	Proficient 10 – 9	Nearing Proficiency 8 – 7	Needs Improvement 6 – 0
	Shows a comprehensive understanding of user-defined functions and parameter passing; string formatting, list methods, reading a file, writing to a file and iteration (for loops)	Shows an adequate understanding of user-defined functions and parameter passing, string formatting, list methods, reading a file, writing to a file and iteration (for loops)	Shows a minimal or no understanding of user-defined functions and parameter passing, string formatting, list methods, reading a file, writing to a file and iteration (for loops).
One	dollarOutput() accepts an integer value that represents the number of values to be displayed. User is not prompted for data. Iteration (for) structure used to iterate through the rows of requested output. Output is displayed in described format; 3 columns using the format statement. Each value has two digits after the decimal, is right-aligned and preceded	dollarOutput() does not accept any data User is prompted for data Iteration structure, other then 'for' is used to iterate through the rows of requested output. Some of the output is not displayed in described format and as shown in the example	dollarOutput() does not contain the supplied code. Iteration structure is not used to iterate through the rows. Output rows does not contain 3 columns of data. Output does not follow the described format or the example

	by a $ sign.		
Problem Number	Proficient 15 - 13	Nearing Proficiency 12 – 10	Needs Improvement 9 – 0
Two	wordGame() correctly opens, reads and closes the file wordGame() correctly retrieves two random words and displays the words enclosed in quotes as shown in the example wordGame() correctly compares word count and displays appropriate message as shown in examples	wordGame() correctly opens files, uses inappropriate read method and closes file wordGame() correctly retrieves the words but does not display them as shown in the example wordGame() does not compare all the possible word count conditions Messages are incomplete or incorrect	wordGame() is not defined correctly File is not correctly read and/or is not close Words are not randomly chosen as described Word count is incorrect Comparisons are not correctly coded Messages are incomplete or incorrect
Problem Number	Proficient 20 - 18	Nearing Proficiency 17 – 14	Needs Improvement 13 - 0
Three	Correctly defines getStats() as described Correctly passes file name as argument Correctly writes all results to output file Correctly performs calculations for	Adequately defines function as described Correctly passes file name as argument Some results are correctly written to output file Correctly performs some of the calculations for number of days	Function has minimal or no correct code Does not pass file name as argument Results are not written to output file Calculation for number of days is incorrect or incomplete

	number of days Correctly performs calculations for number of lines, number of words and number of characters Closed all the files No print statements in function (except for 'to view results go to FileStats.txt' Returns thanks message	Same calculations for number of lines, number of words and number of characters are correct Has more than one read() statement Closed one of the files Output is printed Prints message to go to output file Prints thanks message Return statement with no message	Calculations for number of lines, number of words and number of characters is incorrect Has more than one read() statement or no read() statement Did not close the files Output is printed No printed message to go to the output file No return statement
For all problems	Program (functions) do not use any data types, statement, methods or operators that have not been presented in week 2 or week 1 lectures or reading assignments. Complete thorough testing Program (functions) have no syntax errors Programs (functions) execute with no run time errors All problem specifications are	Program (functions) uses a data type, statement, method or operator that have not been presented in week 2 or week 1 lectures or reading assignments. Only tested with given data or partially tested Program (functions) have one or two syntax errors. Program (functions) do not execute because of a run time error Some deviations from specifications	Program (functions) uses more than one data type, statement, method or operator that have not been presented in week 2 or week 1 lectures or reading assignments. Minimal or no testing Programs (functions) have more than two syntax errors Program (functions) do not execute because of run-time

	correctly coded		errors Hardly follows the specifications