Honors Computer Science Python
Mr. Clausen
Programs 4A, 4B, 4C, 4D, 4E, 4F

PROGRAM 4A Full Names (25 points)

This program should ask the user for their full name: first name, a space, middle name, a space, and last name, then press return. If you don't have a middle name, make one up. This should be assigned to a string variable. Your job is to separate this full name into three separate variables: one for the first name, one for the middle name, and a third for the last name. Display these separate names on the screen. Also display the LastName, a comma, a space, and the FirstName. You will need to use Python's string manipulation features. Here's a hint, the spaces tell you where the first, middle and last names can be found. **DO NOT USE LISTS OR THE SPLIT COMMAND! We are practicing string manipulation using string methods in this program.**

1) Use a `DocString` at the beginning of the program for your comments. Type comments at the beginning of the program to display your name and other information just like those used for program 1A. **Be sure to change the program name, program number, and program description.**

2) Leave a blank line after the DocString.

3) Initialize all of the variables that are to be used in this program. Initialize each variable on a separate line. (Initialize integers to 0 (zero), decimal numbers to 0.0, strings to "" (double quotes with nothing in between them, and Boolean variables to False). You will need variables for all of the following: full_name, first_name, middle_name, last_name, last_first, fullname_length, first_space_location, and second_space_location. You might need more variables. Make sure that you use descriptive identifiers for all of your variables to model "self-documenting code".

4) Leave a blank line after the variable initialization statements.

5) Type the following comment:
   #----------Display My Information----------
   Follow this comment with print statements to display your name and period output just like those used for program 1A. **Be sure to change the program name, and program number.**

6) Leave a blank line after the print statements listed above.

7) For the Input section, type the following comment:
   #----------Input----------
   Ask the user to enter their full name (first middle last). **Do not type cast the name since the input statement automatically inputs string values.**

8) For the Calculations section, type the following comment:
   #----------Calculations----------

DO NOT USE LISTS OR THE SPLIT COMMAND! We are practicing string manipulation using string methods in this program.
These calculations are going to be a little different than just crunching numbers. Perform all of the string processing calculations while assigning the results to variables. Find the number of characters in the full name including the spaces. Find the location of the 1st space. Find the location of the 2nd space. Use string "slicing" to find the first name, middle name and last name. Concatenate the last name, a comma and a space and the first name and assign this to a variable.

9) Leave a blank line after the calculations listed above.

10) For the Output section, type the following comment:
#---------- Output----------
Echo (print) the full name, then the first name, middle name, last name, and the LastName, FirstName.

11) Finish your program with these last 2 lines of code.
print ("")
input("Press enter to quit the program")

12) Save your program as LastNameFirstNameP4A.py.

13) When you are finished with your program, have tested it thoroughly to make sure that your program is correct, and are sure that you don't need to make any changes, then save your program in the "T" network mapping, in the Program 4A folder.

Program 4B Adlibbed MadLibs (40 points)

Write a program that asks the user to enter 3 different nouns, verbs, adverbs, and adjectives.
Make your program user friendly by prompting them for these values.
(Parts of Speech: http://www.englishclub.com/grammar/parts-of-speech_1.htm)

1) Use a `DocString` at the beginning of the program for your comments. Type comments at the beginning of the program to display your name and other information just like those used for program 1A. **Be sure to change the program name, program number, and program description.**

2) Leave a blank line after the DocString.

3) import random

4) Leave a blank line after the import statement.

5) Initialize all of the variables that are to be used in this program. Initialize each variable on a separate line. (Initialize integers to 0 (zero), decimal numbers to 0.0, strings to "" (double quotes

with nothing in between them, and Boolean variables to False). You will need variables for all of the following: sentence1,noun1, verb1, adverb1, adjective1, etc. for all 3 sentences, and random_number. You might need more variables. Make sure that you use descriptive identifiers for all of your variables to model "self-documenting code".

6) Type the following comment:
   #-------------------------Display My Information-------------------------
   Follow this comment with print statements to display your name and period output just like those used for program 1A. **Be sure to change the program name, and program number.**

7) Leave a blank line after the print statements listed above.

8) For the Input section, type the following comment:
   #-----------------------------------Input-----------------------------------
   Ask the user to enter noun1, verb1, adverb1, adjective1, etc. **Do not type cast these since the input statement automatically inputs string values.**

9) For the Calculations section, type the following comment:
   #---------------------------------Calculations---------------------------------
   These calculations are going to be a little different than just crunching numbers. Calculate a random number from 1 to 3 and assign this to a variable. Perform all of the string processing calculations while assigning the results to variables. The variable sentence1 should be assigned string literals (your preselected words for the Madlib in quotes) inserted in between the variables noun1, verb1, adverb1, and adjective1 using concatenation. Continue this pattern for sentence2 and sentence3.

10) Leave a blank line after the calculations listed above.

11) For the Output section, type the following comment:
   #--------------------------------- Output---------------------------------
   Use the random number as a condition in "if, elif, else statements to determine which one of three sentences representing the "Adlibbed" portion of our Madlib to print.

12) Finish your program with these last 2 lines of code.
   print ("")
   input("Press enter to quit the program")

13) Save your program as LastNameFirstNameP4B.py.

14) When you are finished with your program, have tested it thoroughly to make sure that your program is correct, and are sure that you don't need to make any changes, then save your program in the "T" network mapping, in the Program 4B folder.

PROGRAM 4C Be My Pal-indrome (25 points)

This program should ask the user for a word and test it to see if it is a palindrome. We are going to "Error Trap" the user's input using string methods before deciding whether the word is a palindrome or not.

1) Use a `DocString` at the beginning of the program for your comments. Type comments at the beginning of the program to display your name and other information just like those used for program 1A. **Be sure to change the program name, program number, and program description.**

2) Leave a blank line after the DocString.

3) Initialize all of the variables that are to be used in this program. Initialize each variable on a separate line. (Initialize integers to 0 (zero), decimal numbers to 0.0, strings to "" (double quotes with nothing in between them, and Boolean variables to False). Make sure that you use descriptive identifiers for all of your variables to model "self-documenting code".

4) Leave a blank line after the variable initialization statements.

5) Type the following comment:
   #----------Display My Information----------
   Follow this comment with print statements to display your name and period output just like those used for program 1A. **Be sure to change the program name, and program number.**

6) Leave a blank line after the print statements listed above.

7) For the Input section, type the following comment:
   #----------Input And Check for Valid Input----------
   Ask the user to enter their word. **Do not type cast the name since the input statement automatically inputs string values.**
   Use Python's string methods to remove all leading and trailing spaces from the word.
   Make the word all lower case so capitalization won't interfere with checking to see if the word is a palindrome.
   Use a primed while loop with a compound Boolean expression to make sure that the word doesn't contain any spaces, uses only letters of the alphabet (no numbers or symbols) and is not less than 3 characters long (there aren't any 2 letter palindromes). Respond with appropriate messages to the user depending on what they may have typed.
   Use another while loop, or add another Boolean condition to the previous loop to make sure that the user isn't entering the same letter for all of the letters in the "word".

8) For the Calculations section, type the following comment:
   #----------Calculations----------
   These calculations are going to be a little different than just crunching numbers. There is more

than one algorithm to determine whether the word is a palindrome or not. Feel free to choose whichever algorithm you wish.

9) Leave a blank line after the calculations listed above.

10) For the Output section, type the following comment:
#---------- Output----------
Echo (print) the word, then tell the user whether the word is a palindrome or not.

11) Finish your program with these last 2 lines of code.
print ("")
input("Press enter to quit the program")

12) Save your program as LastNameFirstNameP4C.py.

13) When you are finished with your program, have tested it thoroughly to make sure that your program is correct, and are sure that you don't need to make any changes, then save your program in the "T" network mapping, in the Program 4C folder.

Program 4D Word Jumble (50 points)

Write a program that reads a text file of vocabulary words (VocabList.txt). Select one of the words at random, and turn the word into a "word jumble" by scrambling all the letters of the word. Allow the user as many guesses as there are letters in the word. Keep track of the number of guesses and give the user feedback as to how many guesses remain and/or if they guessed the word correctly. If they guess the word, tell them how many guesses it took. If they use up all of their guesses and don't guess the word, please tell them what the word was.
Make your program user friendly by prompting them for their guess. And PLEASE print the word for me (cheat code) so I don't really have to guess the word while I am grading your program.
NOTE: When I grade these programs, I will use a different text file (with different words and a different number of words), so make sure your program isn't written specifically for this text file.

1) Use a `DocString` at the beginning of the program for your comments. Type comments at the beginning of the program to display your name and other information just like those used for program 1A. **Be sure to change the program name, program number, and program description.**

2) Leave a blank line after the DocString.

3) import random
import os

4) Leave a blank line after the import statement.

5) Initialize all of the variables that are to be used in this program. Initialize each variable on a separate line. (Initialize integers to 0 (zero), decimal numbers to 0.0, strings to "" (double quotes with nothing in between them, and Boolean variables to False). **Initialize lists to [ ] and make sure that the word "list" is part of the identifier name for the list (ie. wordList).** Make sure that you use descriptive identifiers for all of your variables to model "self-documenting code", and that all variables are initialized at this place in the program.

6) Leave a blank line after the variable initialization statements.

7) Type the following comment:
#----------Display My Information----------
Follow this comment with print statements to display your name and period output just like those used for program 1A. **Be sure to change the program name, and program number.**

8) Leave a blank line after the print statements listed above.

9) This program does not follow the input, calculations, output model.

10) Type the following comment:
#----------Section 1 Read File and Choose Word----------
Open the text file for reading.
Read the contents of a text file named "VocabList.txt" from the current working directory and assign the contents of the entire text file to a single string variable.
Split the string variable by the spaces and assign all of the words to a list (remember to include the word "list" in your variable name for your list).
Generate a random number from 0 to the length of the list and assign that to a variable.
Use this random number to "choose" a random word from the list.
Make a copy of this word.
Assign the length of this word to a variable. This is the number of guesses the user gets to guess the word.

11) Leave a blank line after these lines of code listed above.

12) Type the following comment:
#----------Section 2 Scramble The Word----------
Scramble the letters of the selected word into a "jumbled word". Use a while loop to do this and use the variable name "index" as the loop counter name. For the algorithm, choose a letter from the word at random and concatenate this letter to a variable that represents the scrambled word. Since strings are immutable, I will give you one line of code in class to "remove" the letter you just concatenated, so that you can't accidently choose that letter again. This means that the length of the word you chose at random will be getting smaller with every letter we "remove".

13) Leave a blank line after these lines of code listed above.

14) Type the following comment:
#----------Section 3 Guess The Word----------

Tell the user what the unscrambled word is (cheat code).
Tell the user what the scrambled word is.
Tell the user how many guesses they have to guess the word. This is the same as the number of letters in the word.
Use a while loop with a compound Boolean expression. One expression should be a Boolean variable representing whether the word has been guessed correctly. The other expression should compare the number of guesses that the user has guessed compared to the number of guesses allowed.
Ask the user to enter their guess, check this against the word chosen from the text file and tell the user whether they guessed the word correctly or not. If incorrect, tell the user how many guesses are left. If the user guesses the word, tell them how many guesses it took. Limit the number of guesses to the number of letters in the word.

15) If they use up all of their guesses and don't guess the word correctly, please tell them what the word was.

16) Finish your program with these last 2 lines of code.
```
print ("")
input("Press enter to quit the program")
```

17) Save your program as LastNameFirstNameP4D.py.

18) When you are finished with your program, have tested it thoroughly to make sure that your program is correct, and are sure that you don't need to make any changes, then save your program in the "T" network mapping, in the Program 4D folder.

Program 4E Play Don't Pay, You Won't Win Anyway (40 points)

Write a program that asks the user which lottery game they wish to play and how many "quick pick" tickets for each game they want. Your program should allow the user to play more than one game and choose a different number of quick picks for each of the games. Display the numbers on the monitor screen and save them to a text file named, "QuickPicks.txt". Below is a list of the games, the number choices and the odds of winning.

Mega Millions: Pick five lucky numbers from 1 to 75 and one MEGA number from 1 to 15: Odds 1 in 258,890,850.
Powerball: Pick five lucky numbers from 1 to 69 and one POWERBALL number from 1 to 26: Odds 1 in 292,201,338.
SuperLotto Plus: Pick five lucky numbers from 1 to 47 and one MEGA number from 1 to 27: Odds 1 in 41,416,353
Make your program user friendly by prompting them for their menu choice.

1) Use a `DocString` at the beginning of the program for your comments. Type comments at the beginning of the program to display your name and other information just like those used for program 1A. **Be sure to change the program name, program number, and program description.**

2) Leave a blank line after the DocString.

3) import random
   import os

4) Initialize constants for the largest number choice for each of the five numbers in each of the three games AND for the largest sixth number for each game (The state did change these all of these numbers in the past).

5) Initialize all of the variables that are to be used in this program. Initialize each variable on a separate line. (Initialize integers to 0 (zero), decimal numbers to 0.0, strings to "" (double quotes with nothing in between them, and Boolean variables to False). **Initialize lists to [ ] and make sure that the word "list" is part of the identifier name for the list (ie. wordList).** Make sure that you use descriptive identifiers for all of your variables to model "self-documenting code", and that all variables are initialized at this place in the program.

6) Leave a blank line after the variable initialization statements.

7) Type the following comment:
   #----------Display My Information----------
   Follow this comment with print statements to display your name and period output just like those used for program 1A. **Be sure to change the program name, and program number.**

8) Leave a blank line after the print statements listed above.

9) This program does not follow the input, calculations, output model.

10) Open the file "QuickPicks.txt" for "writing".

11) Write the line of code for a "while loop" that will continue repeating while the menuChoice is not a "q" or a "Q" (An upper or lowercase Q should quit the loop).

12) Display the menu using print statements for the user to select which game they wish to play **after** the "while loop header" (We want the menu choices to appear every time we want to choose another option.) Use an input statement to ask them which menu choice they want (Do not type cast the user's input, leave their answer as a string. Don't type cast it to a string either, the input command automatically enters information as a string.

13) For each of the "if statement" menu choices print to the screen and write to the text file one time which game's numbers you will be displaying on the screen and writing to the text file. Also for each "if statement" menu choice, ask the user how many quick picks they want. The user should

be able to ask for a different number of tickets for each game that they wish to play.

14) In each if statement path use nested loops to generate the five numbers plus the sixth number and turn this into a string for each Quick Pick they requested. If a number is less than 10, concatenate an empty space before the number whether it's one of the 5 regular numbers or the sixth "special number". The inner loop should generate one choice containing 5 regular numbers and one special number. Use string concatenation to append each random number generated to a string variable. The outer loop is responsible for generating as many "picks" as the user requested. Display each set of numbers to the screen and write them to the text file as each of the complete set of six numbers is generated.

15) Display the numbers on the screen and write them to a text file named "QuickPicks.txt".

16) Close the text file after the user enters a 'q' or 'Q' to quit the menu. This way all of the quick picks will be saved to the text file.

17) Finish your program with these last 2 lines of code.
```
print ("")
input("Press enter to quit the program")
```

18) Save your program as LastNameFirstNameP4E.py.

19) When you are finished with your program, have tested it thoroughly to make sure that your program is correct, and are sure that you don't need to make any changes, then save your program in the "T" network mapping, in the Program 4E folder.

Program 4F Cryptic Code (~~40 points~~)

I am not requiring students to write this program. If you finish all the other programs early, you may have fun writing this program. Therefore, I am leaving the directions less detailed.
Write a program that reads a text file named "OriginalMessage.txt", encrypts it using any encryption algorithm you wish to use. Save the encrypted file in a text file named "EncryptedMessage.txt". Next open the encrypted file and decrypt it and save it in a file named "DecryptedMessage.txt". Use a menu to let the user choose each of these actions.
Make your program user friendly by prompting them for their choice. Use one line comments to separate this program into its parts: input, and "calculations and output" which will be merged into one section.

1) Use a `DocString` at the beginning of the program for your comments.

2) Initialize all of the variables that are to be used in this program.

3) Use print statements to display your name and period output just like those used for program 1A.

4) Display a menu so the user can choose each of the actions for this program.

5) Open the original message and encrypt it.

6) Save the encrypted file as a text file.

7) Read the encrypted file, decrypt it and save it as a text file.

8) Make sure that you use descriptive identifiers for all of your variables.

9) Save your program as LastNameFirstNameP4F.py.

10) When you are finished with your program, have tested it thoroughly to make sure that your program is correct, and are sure that you don't need to make any changes, then save your program in the "T" network mapping, in the Program 4F folder.