ASSIGNMENT 2

Contents

[Contents 1](#_Toc460353060)

[Purpose 2](#_Toc460353061)

[Assignment Background 2](#_Toc460353062)

[Assignment Statement 2](#_Toc460353063)

[Requirements: 2](#_Toc460353064)

[Code/Comment Format 4](#_Toc460353065)

[What to Deliver 4](#_Toc460353066)

[Notes 5](#_Toc460353067)

# Purpose

In this assignment, you will be demonstrating your understanding and use of python constructs, as covered in module 2.

# Assignment Background

This assignment involves the game Mad Libs built in module 1. For this assignment, you are required to use the given .py file skeleton. The reason for this is to give everyone working code, so no one falls behind. We will aim to make our mad lib madder (!) as well as more persistent by saving the user data to files.

# Examples

Mad Lib Sentences

Man verb on a adjective noun.

Noun verb to the adjective ground.

All the king’s adjective horses and all the king’s dainty noun could not verb scrambled egg man back together again.

Example word lists:

NOUNS = ["time", "year", "people", "way", "day", "man", "thing", "woman"]

VERBS = ["pay", "put", "read", "run", "say", "see"]

ADJECTIVES = ["other", "new", "good", "high", "old"]

# Assignment Statement

* Like assignment 1, you can either choose to use the provided csv files or create your own csv files with your own data. The files provide the data for the lists used in the mad lib.
* The lists must be of different lengths.
* The lists should be < 10 items long.
* Each sentence should be missing 1 element of each type (noun, verb, adjective).

You are required to supply *Last Name\_First Name\_Assignment2.py.*

# Requirements:

NOTE: You are required to use the given skeleton to complete assignment 2.

1. All 4 lists must be loaded from CSV files at the beginning of the program. (Sentences, nouns verbs and adjectives etc.)
   1. Create a “resources” folder where the files can be located.
   2. Create a “load\_csv\_to\_list” function. Given the path to a file, it will return its content as a list. Please remember to validate the path and the file's content appropriately.
      1. HINT: This means the following: Given the path to a file it will return a list. This is algorithm verbiage for:
         1. def load\_csv\_to\_list(String:path\_to\_file): List
         2. i.e. input path to file, output the list (parameter of the function path to file, return value a list)
   3. Create a load\_mad\_lib\_resource function that:
      1. Calls the load\_csv\_to\_list function
         1. Verifies the list is valid
         2. Returns a tuple of the shuffled list by calling shuffle\_list function in step 4
   4. Since the data stored is a sequence or list, what we did with user input in assignment 1 left our program very predictable with high and low boundaries. To remedy this, please shuffle the content of the data lists. The shuffle should occur whenever the program is started, or continued. Create a new “shuffle\_list” function. This function will for each list:
      1. Shuffle the contents of the lists read from the CSV files.

NOTE: You are required to write your own shuffle list function. Do not call any built-in python modules that provide this functionality.

1. In this version of the program, we save the user's previous mad lib in a file.
   1. Prompt the user for his/her username.
   2. The mad lib file name should include the username. Use his/her username in conjunction with the “os.join” python function to create the path ( <https://docs.python.org/3/library/os.path.html#os.path.join> ) and locate the saved mad lib file, if it exits.
   3. Load the saved data to resume playing the game, if found
   4. Reuse any applicable code written for requirement #1
   5. Make sure to load the data in the right order. We suggest for you to store the values of the sentence list in the 1st column and their order in the 2nd column.
   6. Once loaded sort the list to ensure correct ordering, explain which method you used and why. HINT: If your code needs an explanation, or you just want to clarify your thoughts, use a doc-strings, or comments, within the code.
2. Create a “play\_game” function, the user input is the used for list item selection.
   1. Prompt user for a number..
   2. Validate user input, see assignment #1 for details
   3. Using the same method as for assignment #1 pick a random item:
      1. 1 sentence from the sentence list
      2. 1 noun from the noun list.
      3. 1 verb from the verb list.
      4. 1 adjective from the adjective list
   4. Replace the selected noun, verb, adjective in the selected sentence.
   5. Check that the resulting sentence is unique (has not been used before).
      1. If it has not been used before, save it in the list of completed sentences (your mad lib). Save (write) the list to file so it could be used for later use (see #2)
      2. If it has been used before, do not save it and print message letting the user know.
   6. Print your current mad lib to the user (list of all completed sentences so far)
   7. Ask user if he/she wants to keep playing (ans: y or n)
      1. Validate input
      2. If “y” go back to 1
      3. If “n” exit

# Code/Comment Format

Good code includes well named variables that are consistent from the beginning to the end of the program. Naming of objects should be self-explanatory. For instance, iterator\_for\_noun\_list is much better than i.

Every program consists of a sequence of paragraphs, each of which has objectives, and which builds on the previous paragraphs. We are mostly interested in objectives that are valid at the end of the program so we can verify the program's design. The following is a preferred form for such paragraph headings. The # sign is adequate when the comment is a single line.

#This is an in-line comment – used to document the code for you, or anyone else, that intends

#To extend the code

In-line comments are helpful when one has to go back to the code 6 months later to make changes.

For doc strings, python allows the use of triple quotes. The triple quotes can be either single or double quotes. A doc sting is generally used as user documentation. It does not need to include details of the implementation of the program, but instead it provides documentation as how to use the API for the program (input, output etc.)

For example:

“””

This is an example of a doc string

It allows multiple lines within the string.

“””

‘’’

This is an example of a doc string

It allows multiple lines within the string.

‘’’

This becomes significant when using functions, classes etc. as the triple quotes help to self-document the parameters and return values of the function.

# What to Deliver

Supply

The *Last Name\_First Name\_Assignment2.py* file. If you utilize your own csv files, you are required to upload them with the assignment.

# Notes

* Assignments can be submitted once. If extenuating circumstances exist, contact your facilitator.
* Note the statement in the syllabus on timeliness of submissions (the gist being that all assignments must observe the deadlines).
* Start by identifying and ordering the objectives.
* There are no testing requirements for this assignment. However, it would be prudent to make sure your program does not crash and all input validation is performed correctly.

# Grading

Requirement Points

1a. 5

1b. 15

1c. 10

1d. 15

2a n/a

2b 5

2c 10

2d None – Code should be working from skeleton

2e 15

2f 15

3 10 points