

CS1300 Intro to Computer Science

Final Project: Word Jumble

Due date: Dec 6, 2019, 11:55pm

Objectives:

Demonstrate your ability to apply fundamental programming concepts that we have covered throughout the semester to write a program.

- Variables
- Iterative statements
- Select statements
- Functions
- Strings
- Lists
- Write a program given a general outline following basic design heuristics.
- Recognize and extract a function from existing requirements to simplify the code.
- Define functions with docStrings that list the description, input, and output.

Overview and Project Layout

The final project builds a word jumble game. The game will present a jumbled word to the player. The main objective of the game is for the player to simply unscramble the word and type it in for evaluation. The player types in a guess until the word is successfully unscrambled, at which point the player is congratulated and they are asked if they would like to play again. If, at any point, the player no longer wishes to play, they may hit enter at the prompt to quit the game.

The project divides into three parts (Part 1 is described in the paragraph above). Part 1 is worth 79 points. Part 2 is worth 10 points, and Part 3 is worth 11 points. You are not required to complete Parts 2 and 3, but your overall grade will still be out of 100 points. For example, if you complete only Part 1, then the maximum grade you can earn is 79 out of 100 points. Similarly, if you complete through part 2, the maximum grade you could potentially earn is 89/100 points.

Part 2 will be more challenging than Part 1, and in turn Part 3 will be more challenging than Part 2, mainly being that less detailed instructions are provided and require you to demonstrate a deeper understanding of the course objectives. Part 2 will not be graded if you do not earn 70 of the 79 points on Part 1. Part 3 will not be graded if you earn less than 7 of the 10 possible points on Part 2.

The specifications are given for each part below. Each part builds on the previous part. **Do not move onto the next part until you have successfully completed ALL requirements from the previous part.**

Getting Help

- You are able to seek help from the csX tutors for *Part 1 only*. TA's have been instructed to correct syntax errors and help with explaining and correcting runtime errors. If you have a question about how to implement a given task, the TA's are instructed to refer you to a related chapter in the book and/or discuss a lab that dealt with similar material. Do not seek their help on Parts 2 and 3.
- Do not discuss the project with anyone other than the csX tutors or your instructor. **I run all project submissions through a plagiarism detection service.**
- You may only seek help/ask questions from your instructor for Parts 2 and 3 of the project.

Code Format

The format of the code is very important. *Please do not mix program code and function definitions.* **It is a requirement** that your code file be organized into the following sections:

- **Import statements**
- **Function definitions** – all functions must have docStrings. Every docString *must have* a description, any input requirements or @input none if no input is required, and @return stating what the output value is or none if the function does not have an output.
- **Program code** – of which there will only be one line of code that falls into this category.

Additionally, name your variables and functions appropriately. By this I mean, give obvious names. For example, if your variable is pointing to a grade value, then call it `a_grade`, not `character`. Additionally, function names usually are stated as an action or a verb – e.g. `calculate_grade`; and variables are typically stated as nouns – e.g. `a_grade`, `size`.

Getting started

All code will go into a single python file in part I; part III is optional if you would like to include external files. Please name your file according to the following convention: `firstname_lastname_finalProject.py`, where `firstname` and `lastname` are you actual first and last names.

There is no starter file or additional module files for this assignment, however, you are free to use your lab exercises as examples. If you have any questions about the solutions to previous labs, please come and talk to me. I will be happy to go over them with you.

Special Notes

Here are some hints to get you started. Please keep in mind that the code referenced here are examples and you will use similar code in your project.

1. **Getting a random word from the list.** The `random` module provides methods that enable us to easily get *a random element from a non-empty sequence/list*. Reference the [random module documentation](#) for an appropriate function. In the case of this game, the list will be a list of words to scramble and guess. Usage of this method is very similar to the `random.randrange` function from assignment 2.
2. **List slices review.** Please remember the following about list slices: the start value is included in the range; the stop value is **not**. Consider the following:

```
fullname = "A|n|a| |P|e|r|a|l|t|a"  
(index      0|1|2|3|4|5|6|7|8|9|10) # These are the index values of the characters in the string.
```

```
space_position = 3          # index value of the space in the fullname variable value.  
f_name = name[ : space_position]    # f_name has the value 'Ana'; up to but does not include 3  
l_name = name[ space_position + 1 : ]# l_name has the value 'Peralta'; from position 4 to end
```

Note that the space in the name has been 'deleted'.

Academic Honesty

The following page contains detailed instructions on the requirements for the assignment. There is nothing unexpected or tricky about this assignment. It is a way for me to evaluate your understanding of the concepts presented in this course. If you have any questions, you can seek help from the csX lab or from myself. You may not ask fellow classmates, CS majors outside of the csX lab, or any other CS professional(s) for help on this project.

Please remember to **do your own work** and not to discuss the implementation detail (i.e. how you wrote your code) of your assignment with anyone. *If you are working in the lab or other public space, be aware of those beside you who may have wandering eyes. Furthermore, do not leave your work on any public computers. If someone is borrowing your computer, make sure that your assignment is not on the machine. Do not give your code to anyone else, even if they promise not to copy it.* **Any work flagged by the plagiarism software will be awarded a zero and all incidents will also be reported to the Office of Student Affairs so that they can determine if further disciplinary action is warranted.** Please keep in mind that this is 20% of your final grade in this class. A zero on this project will drop your grade a minimum of 2 letter grades.

Part I: Program Structure

The project breaks down into two basic steps listed below. In the following section, there will be more requirements listed in order to implement the functionality.

1. Play a game
2. Scramble chosen word

Code requirements

Scramble the Chosen Word

Define a function to scramble the word chosen.

Outline:

- The function requires one input representing the chosen word.
- Use a local variable to hold the jumbled word. The initial value should be an empty string.
- Use a loop create your jumbled word one letter at a time; continue looping while the word is not an empty string. (In the body of the loop, you will write code that will take away one letter at a time.)
- Get a random letter from the word by generating a random position/index in the word to scramble. **Hint:** The upper bound of the **randrange** function is the length of the word to scramble.
- Update the jumbled word by adding the random letter to the jumbled word.
- Create a new version of the word minus the one letter at the random position chosen. **Hint:** Use the slice operator to divide the word into two halves; the randomly chosen letter divides the word.
 - The first slice, is every letter up to, but not including **word[position]** (where word and position are variables in my function.)
 - The second slice starts at the letter *after* **word[position]** and goes to the end of the word. I gave you some hints under the Special Notes.
 - Update word by adding the two first and second slices back together. This is where you will ‘take away’ the letter you just added to the jumbled word.
- Return the jumbled word.

Test your code. Verify the function by calling the function and verifying that a scrambled version is returned.

Play the Game

Define a function to put all these pieces together to create a Word Scramble Game.

Program Outline:

- Introduce the game by providing a small paragraph about what the game is and how to play it. I used triple-quoted strings to maintain the formatting in the example provided on the last page.
- Use a local variable to hold a list of a minimum of 10 words. You may
 - Choose any words you wish as long as they are not offensive or derogatory in any way.

- Directly assign them as a list of words: meaning `word = [...]`.
- Select a random word from the list.
- Use a local variable to hold the scrambled version of the chosen word.
- Keep asking the player for a guess until they have successfully unscrambled the word.
- Notify the player of the status of their guess after each attempt.
- The player can exit the game at any time by hitting enter at the prompt. This will result in an empty string.
- Ask the user if they wish to play again.

Requirements:

- The function requires zero input.
- This function does not return a value.
- There are at least three possible functions that you could write to simplify the play game function. Identify what functionality could be broken out into a separate function and write it. You are required to implement one of the functions; however, you can earn an additional point for each function correctly identified and implemented.

Test your code. You should be able to play the word scramble game as many times as you wish. The player may also abort play at any time by hitting enter at the prompt.

This concludes part 1 of the project. **Once you are finished with Part I**, make a comment at the top of the file that you completed part I. Submit a copy of your project to Moodle. If you choose to continue with parts II and III, you can re-submit the project as many times as you like up to the submission deadline. *That way if you forget to upload the finished project, you will get partial credit for your work.*

NOTE: *I will not accept late submissions of any kind without a valid written excuse, such as a doctor's note. Please speak with me ahead of time to determine if an extension is allowed.* This project is 20% of your final grade.

If you would like, you can move on to part II.

Part II

In my version, I had some words in my list that contained spaces, e.g. 'computer science'. If this 'word' were to be chosen, there is no telling where the space would end up and which letters belong to which word. What typically happens in this situation, in paper versions of word scramble, the space is kept in the same position and each word is scrambled separately.

Currently: 'computer science' can be scrambled to become 'rcoetucimepec ns'
or another version could be 'ent roumicepeccs'

Goal: 'computer science' to become 'rouctemp eccnise' where the space maintained at its original location so that no matter how the individual words are jumbled, the space will always appear in the correct position.

You can assume that there will only be one space in any 'word'.

This concludes part 2 of the project. **Once you are finished with Part 2**, make a comment at the top of the file that you completed part 2. Submit a copy of your project to Moodle. If you choose to continue with part III, you can re-submit the project as many times as you like up to the submission deadline. *That way if you forget to upload the finished project, you will get partial credit for your work.*

Part III

In this class we worked on two games. For assignment 2 you created the guessing game and for the final project you created the word scramble game. It would be cool to be able to play either of these two games by choosing one of them from a menu.

For example, when the program runs, the player would be presented with two menu items

1. Play the guessing game
2. Play the word scramble game

If the player selected 1 from the menu list, then the guessing game would begin. Otherwise, if the player selected 2 from the menu, then the word scramble game would begin. Once either game has ended, if the player wants to play again, then they would be presented with the menu options again.

You may use and modify any code used in this class. This includes any code that I gave you or any work that you completed. How you organize this part of the project is up to you. You have several options – have all code in one file or spread the code over multiple files and use import statements to bring in the additional functionality. One suggestion is to re-name the play_game functions so that they identify the game they call, for example, rename the play_game function from the project to play_word_scramble. This will help make your code less confusing.

This concludes part 3 of the project. **Once you are finished with Part 3**, make a comment at the top of the file that you completed part 3. Submit a copy of your project to Moodle.

Submission

Submit your code file to Moodle, and any additional files you may need to submit, by the due date.

Grading (100 points)

- Part I (79 points)
 - Scramble the word
 - Play the game
 - Identification and implementation of one function from the play game function.
 - The structure of the code file, docStrings, function, and variable names
- Part II (10 points)
 - Incorporation of two-word jumbles.
- Part III (11 points)
 - Menu implementation
 - Incorporation of the guessing game code

Figure 1: Example execution of the Part I

```
>>>
RESTART: /Users/crolka/Google Drive/_UWG/Fall2019/CS13

      Welcome to Word Jumble!
      Unscramble the letters to make a word.

      (Press enter at the prompt to quit at any time.)

The jumble is: hlpoyoexn
Your guess: xylophone
That's it! Congratulations you won!
Do you want to play again?(y/n) y
The jumble is: rwsaen
Your guess: rensaw
Sorry, that's not it. Try again.
The jumble is: rwsaen
Your guess: answer
That's it! Congratulations you won!
Do you want to play again?(y/n) y
The jumble is: itflifcud
Your guess: ←
Thanks for playing
>>> |
```

I pressed
enter here