Project 5

Start Assignment

Due Jul 16 by 11:59pm **Points** 50 **Submitting** a file upload **File Types** zip

Goals

At the completion of this assignment, you should:

- have gained additional experience working with Boolean expressions
- have gained additional experience working with relational operators
- have gained additional experience working with logical operators
- have gained additional experience working with conditions
- have gained additional experience working with keyboard input
- have gained additional experience working with len() and print() functions and concatenation
- have gained experience working with while loops
- have gained experience working with the in and not in operators
- have gained experience with escape characters (\t and \n)
- have gained experience with simple formatted output using escape characters
- have gained additional experience with guardian, traversal, search and counter patterns
- have practiced outlining the steps and including those steps as comments in your code
- have gained experience interpreting requirements and implementing those requirements in code

Expected Output and Grading Rubric

You can view the expected output and grading rubric for this lab at the end of the assignment.

Getting Started

You will not be required to submit an outline or pseudocode for this assignment; however, you should take the time to consider the steps involved in solving this assignment, and include those steps as single-line comments in your code to help guide you through implementing the coded solution.

You should not attempt to write an entire coded solution; instead, practice incremental development, adding small amounts of code that you can test and validate.

You are not required to create any user-defined functions for this assignment, but you may use functions as part of your solution.

Assignment

For this assignment you will be creating a "simple" guessing game. You will prompt the user for a word, then you will proceed to make letter "guesses" to try to guess the word. Yes, you already know the word, so the guessing should be easy.

Below are the game "rules". These rules represent the "what" and are typically referred to as functional specifications.

Functional Specifications:

- Prompt the user for a guess word
- Prompt the user for a letter
- Empty word or letter input will exit the program
- After each letter guess, if all letters not found, display the guess status (tabbed over on tab space):
  - Indicate if already guessed and already found
  - Indicate if already guessed and already not found
  - Indicate if guessed letter not found
  - Indicate if guessed letter found
  - Follow the guess status with the complete list of letters guessed so far
- Upon program exit, or when all letters have been guessed, display results if any guesses were made
  - Display original guess word
  - Display matched letters
  - Display unmatched letters
  - Display total number of guesses
- Guess letters need only be entered once even if letters are repeated in the guess word
- **Optional**: Letters are case-insensitive (i.e., guessed letters will match word regardless of case)

In addition to functional specifications to describe what the program should do, you also need to consider "how" the program will accomplish the required tasks. The "how" is typically called the design specifications.

Design Specifications:

- Implement your solution using a file called **p5-guess.py**
- Use only programming syntax we've learned so far in class
- Use the *in* and *not in* operators to determine if a letter is contained within a string
- Use the *while* loop to force continuous input
- Use the *break* statement to exit any loops
- Use conditionals to test input and determine actions
- Use accumulator variables for matched, unmatched and total guesses variables
- Use the tab escape character to indent guess status information (i.e. tabbed)
- Guess status information must be preceded by "> " and have the following format (where x is the guessed letter):
  - > x not found

    - > x already guessed and not found
    - > x already guessed and found
    - > x found
- After each guess status, display the complete list of guessed letters in the following format (where esta represents the matched and unmatched guessed letters combined):
    - > Guesses so far: esta
- If the letter entered is more than a single character in length, display the following error message tabbed, and re-prompt the user for another letter:
    - > Only enter a single letter
- Use the tab escape character to align final result information in columns
- Word prompt must be: "Enter a guess word (blank to quit): "
- Letter prompt must be: "Enter a guess letter (blank to quit): "
- Separate the final results with a blank line
- The first line of the results must be "*** Results ***"
- The second line must be "Word:" followed by the original guess word, tabbed to align with the subsequent result line values
- The third line must be "Matched:" followed by the matched guessed letters, tabbed to align with the subsequent result line values
- The fourth line must be "Unmatched:" followed by the unmatched guessed letters, tabbed to align with the subsequent result line values
- The fifth line must be "Guesses:" followed by the total number of letters guessed, tabbed to align with all the line values

How to Determine All Letters Guessed

The guess word may have duplicate letters, but guess letters can only be entered once. You will need to create a string of the unique letters in the guess word, and use this unique letters string when determining if all of the letters have been guessed.

Sample Output

The following sample output demonstrates the expected input and output.

Empty word guess:

```
Enter a guess word (blank to quit):
>>>
```

Empty letter guess:

```
Enter a guess word (blank to quit): test
Enter a guess letter (blank to quit):
>>>
```

Complete game with errors and repeated matched and unmatched guesses:

```
Enter a guess word (blank to quit): test
Enter a guess letter (blank to quit): a
        > a not found
        > Guesses so far: a
Enter a guess letter (blank to quit): a
        > a already guessed and not found
        > Guesses so far: a
Enter a guess letter (blank to quit): t
        >  t found
        > Guesses so far: ta
Enter a guess letter (blank to quit): e
        >  e found
        > Guesses so far: tea
Enter a guess letter (blank to quit): s
        >  s found
        > Guesses so far: tesa
        > All letters found

*** Results ***
Word:           test
Matched:        tes
Unmatched:      a
Guesses:        5
>>>
```

Submissions

For this assignment you will be uploading your work from Lab 5 as well as your work from this assignment.

Remember that all submitted files must include the header docstring at the top of each file.

Create a single compressed file that contains the Python files. To create the compressed file, select the deliverable Python files, right-mouse click on the selected files, and select Compress (Mac) or Compressed (Windows). A new file will be created that contains all of the deliverable Python files.

Rename the compressed file **p5.zip**. Use the Project 5 Canvas site to submit/upload the compressed/ZIP file.

The compressed file will contain the following deliverables (point values in parentheses).

- **(15) lab5-factorial.py**
- **(35) p5-guess.py**

Once uploaded, revisit the Project 5 page in Canvas and confirm the file was uploaded.

Remember, you are responsible to submit your assignments before the deadline. Late submissions will not be accepted.

Expected Output

To assist you with completing this project, below is the expected output.

```
Enter a guess word (blank to quit): oregon
Enter a guess letter (blank to quit): o
        > o found
```

```
        > Guesses so far: o
Enter a guess letter (blank to quit): o
        > o already guessed and found
        Guesses so far: o
Enter a guess letter (blank to quit): x
        > x not found
        > Guesses so far: ox
Enter a guess letter (blank to quit): x
        > x already guessed and not found
        > Guesses so far: ox
Enter a guess letter (blank to quit): r
        > r found
        > Guesses so far: orx
Enter a guess letter (blank to quit): e
        > e found
        > Guesses so far: orex
Enter a guess letter (blank to quit): n
        > n found
        > Guesses so far: orenx
Enter a guess letter (blank to quit): g
        > g found
        > Guesses so far: orengx
        > All letters found

*** Results ***
Word:           oregon
Matched:        oreng
Unmatched:      x
Guesses:        8
```

Grading Rubric

p5-guess.py

- Verify use of break
- Verify use of while
- Verify use of in
- Verify use of tab (\t)
- Verify use of >
- Verify use of at least two input()
- Validate letter prompt
- Validate valid guess detection
- Validate duplicate guess detection
- Validate invalidate guess detection
- Validate duplicate invalidate guess detection
- Validate already guess and found
- Validate successful guess results output
- Validate successful guess matched
- Validate successful guess unmatched
- Validate successful guess number of guesses