CSE 231 Spring 2017

Programming Project #4

This assignment is worth 40 points (4% of the course grade) and must be completed and turned in before 11:59 PM on Monday, February 6th.

Assignment Overview

Hangman is a popular word game. In this game, the player is given some number of blanks representing the name of a movie or an actor and he/she has to guess the name using at most K number of chances.

A good website to play this game is: http://www.hangman.no/. Select the UK flag, then click on "Start game" and finally choose a category in the list and get started. The number of chances you get is 10 (i.e. k = 10), for your project it will be only 6.

Task

Your task is to implement the Hangman game in Python. Before implementing the game, please play the game on the website mentioned above. It will help you understand the project.

Project Specifications:

1) Output a brief description of the game of hangman and how to play.
2) Ask the user to enter the word or phrase that will be guessed (have a friend enter the phrase for you if you want to be surprised). Error check: repeatedly ask for a word or phrase until the input is letters and spaces.
3) Output the appropriate number of dashes and spaces to represent the phrase. Dashes are placeholders for letters.
4) Continuously read guesses of a letter from the user and replace the corresponding dashes if the letter is in the word (replace all occurrences, upper and lower case, of the letter), otherwise report that the user has made an incorrect guess. Error check: only accept letters and spaces. An incorrect guess is counted as a guess, but a guess that is not letters and spaces is not counted as a guess. If a guess is more than one letter, it is assumed to be a guess of the whole phrase: if the match is perfect (independent of case), the game is won; if not, the game is over as a loss.
5) Each turn you will display the phrase as dashes but with any already-guessed letters filled in, as well as which letters have been incorrectly guessed so far and how many guesses the user has remaining. Case of original input must be preserved, e.g. an upper-case letter in word/phrase must always be displayed in upper case even if the user guessed using a lower-case letter.
6) Your program should allow the user to make a total of k=6 guesses. After the 6^{th} guess, the game is either won with a correct guess or the game ends as a loss.
7) One run of the program plays the game once. Do not have a loop to restart the game.

Assignment Notes:

1. Simplify:
    a. Leave error checking to last.

   b. Write your first version to handle a single word (not a phrase of multiple words).
2. To clarify the project specifications, sample output is appended to the end of this document.
3. You may not use advanced data structures such as lists, sets, or dictionaries.
4. Items 1-7 of the Coding Standard will be enforced for this project.
5. Review the syllabus with respect to collaboration. What you hand in must be your own work. Among other things, note this one: "If you show your code to another student, you are almost guaranteed a zero because most novice programmers will not be able to think of another way to do it and end up copying your code or sharing it with someone else who copies it."

Deliverables

The deliverable for this assignment is the following file:

`proj04.py` -- your source code solution

Be sure to use the specified file name and to submit it for grading via the **handin system** before the project deadline

Notes and Hints:

This project should all be done with strings, use help(str) in the python shell window to see all of the string methods that may be useful. The instructors used some of these in drafting a solution, but neither used all of them:

1) The string concatenation operator "+". In order to keep track of the incorrect guesses, you could initialize a blank string and use "+" to update the string with the incorrect guesses.

2) The membership operator "in" would be useful to check if a particular letter/digit has already been guessed (correctly or incorrectly).

3) for i,ch in enumerate(s):
can be useful when iterating through the characters (ch) of a string while wanting to keep track of indices (i) at the same time.

4) String slicing is useful to insert a letter in a string. For example if I have x = "hello" and I wanted to put a 'Z' in the middle of the word, I could write:
x = x[0:3] + 'Z' + x[3:]
or if I wanted to 'Z' to replace the 'e' I could write:
x = x[0:1] + 'Z' + x[2:]
remember string indexing using slices includes the start position but not the end position, so x[0:2] is "he" but does not include the 'l' at index 2.

5) lower() can be used to change a string to lowercase – make sure you find the letter the user enters whether it's lower or uppercase in the guess word.

6) find() returns the index at which the first instance of a substring is found in a string, for example if x=”hello”
x.find(‘e’) returns 1 and x.find(‘l’) returns 2.

7) isalpha() is useful for error checking to check for letters.

8) replace(old,new) is useful for dealing with spaces during error checking.

9) Try the example program with many different inputs and make sure your program behaves similarly!

Getting Started

- *Solve the problem using pencil and paper first.* You cannot write a program until you have figured out how to solve the problem. This first step can be done collaboratively with another student. However, once the discussion turns to Python specifics and the subsequent writing of Python, you must work on your own.

- Use Anaconda Spyder to create a new program. Use the required file name (`proj04.py`).

- Write a simple version of the program, e.g. handle single words with no errors. Run the program and track down any errors.

- Use the **handin** system to turn in the first version of your program.

- Cycle through the steps to incrementally develop your program:

    - Edit your program to add new capabilities.
    - Run the program and fix any errors.

- Use the **handin** system to submit your final version.

Sample output of the program:

Test 1

```
Hangman: guess letters until you can guess the whole word or phrase.
In this game you get six tries.

Enter a word or phrase: hello
phrase: hello
current: -----
```

```
0 guesses so far out of 6:

Guess a letter or whole word/phrase: h
current: h----
1 guesses so far out of 6: h

Guess a letter or whole word/phrase: l
current: h-ll-
2 guesses so far out of 6: hl

Guess a letter or whole word/phrase: x
Letter not in phrase.
current: h-ll-
3 guesses so far out of 6: hlx

Guess a letter or whole word/phrase: o
current: h-llo
4 guesses so far out of 6: hlxo

Guess a letter or whole word/phrase: e
current: hello
You won.
```

Test 2

```
Hangman: guess letters until you can guess the whole word or phrase.
In this game you get six tries.

Enter a word or phrase: hello
phrase: hello
current: -----
0 guesses so far out of 6:

Guess a letter or whole word/phrase: x
Letter not in phrase.
current: -----
1 guesses so far out of 6: x

Guess a letter or whole word/phrase: y
Letter not in phrase.
current: -----
2 guesses so far out of 6: xy

Guess a letter or whole word/phrase: z
Letter not in phrase.
current: -----
3 guesses so far out of 6: xyz

Guess a letter or whole word/phrase: a
Letter not in phrase.
current: -----
4 guesses so far out of 6: xyza
```

```
Guess a letter or whole word/phrase: b
Letter not in phrase.
current: -----
5 guesses so far out of 6: xyzab

Guess a letter or whole word/phrase: c
Letter not in phrase.
current: -----
You lost.
The word/phrase was: hello
```

Test 3

```
Hangman: guess letters until you can guess the whole word or phrase.
In this game you get six tries.



Enter a word or phrase: ab3x
Error: only letters are allowed as input.

Enter a word or phrase: Success
phrase: Success
current: -------
0 guesses so far out of 6:

Guess a letter or whole word/phrase: 8
Only letters and spaces are allowed as input.
0 guesses so far out of 6:

Guess a letter or whole word/phrase: s
current: S----ss
1 guesses so far out of 6: s

Guess a letter or whole word/phrase: C
current: S-cc-ss
2 guesses so far out of 6: sc

Guess a letter or whole word/phrase: success
You won.
```

Test 4

```
Hangman: guess letters until you can guess the whole word or phrase.
In this game you get six tries.
```

```
Enter a word or phrase: abc 2
Error: only letters are allowed as input.

Enter a word or phrase: to be or not to be
phrase: to be or not to be
current: -- -- -- --- -- --
0 guesses so far out of 6:

Guess a letter or whole word/phrase: O
current: -o -- o- -o- -o --
1 guesses so far out of 6: o

Guess a letter or whole word/phrase: t
current: to -- o- -ot to --
2 guesses so far out of 6: ot

Guess a letter or whole word/phrase: E
current: to -e o- -ot to -e
3 guesses so far out of 6: ote

Guess a letter or whole word/phrase: b
current: to be o- -ot to be
4 guesses so far out of 6: oteb

Guess a letter or whole word/phrase: r
current: to be or -ot to be
5 guesses so far out of 6: otebr

Guess a letter or whole word/phrase: xxxx yyyy
Wrong guess of whole word or phrase.
You lost.
The word/phrase was: to be or not to be
```

Grading Rubric

```
Computer Project #04                              Scoring Summary

General Requirements

_______     5 pts    Coding Standard 1-7
          (descriptive comments, function header, etc...)


Implementation:

__0__     (10 pts)  Pass test1: plays one-word, lower-case game with no error
checking
```

__0__ (5 pts) Pass test2: stops one-word, lower-case game after 6 incorrect guesses

__0__ (8 pts) Pass test3: one-word, multiple case game that handles errors in input word and guessed letters, preserves case, and fully-guessed word wins.

__0__ (8 pts) Pass test4: multi-word, multiple case game that handles errors in input word and guessed letters, preserves case, and fully-guessed word loses.

__0__ (4 pts) Output is clear and easy to read.

TA Comments:

Optional Testing

This test suite has a test program for each function and two for the entire program. They are handled slightly differently.

1. Test the entire program using `run_file.py` and input the corresponding digit for the appropriate test file. You need to have the files `test1.txt, test2.txt, test3.txt` and `test4.txt` from the project directory.
   Make sure that you have the following lines at the top of your program (only for testing):

```
import sys
def input( prompt=None ):
    if prompt != None:
        print( prompt, end="" )
    aaa_str = sys.stdin.readline()
    aaa_str = aaa_str.rstrip( "\n" )
    print( aaa_str )
    return aaa_str
```

Educational Research

When you have completed the project insert the 5-line comment specified below.

For each of the following statements, please respond with how much they apply to your experience completing the programming project, on the following scale:

1 = Strongly disagree / Not true of me at all
2
3
4 = Neither agree nor disagree / Somewhat true of me
5

6
7 = Strongly agree / Extremely true of me

****Please note that your responses to these questions will not affect your project grade, so please answer as honestly as possible.****

Q1: Upon completing the project, I felt proud/accomplished

Q2: While working on the project, I often felt frustrated/annoyed

Q3: While working on the project, I felt inadequate/stupid

Q4: Considering the difficulty of this course, the teacher, and my skills, I think I will do well in this course.

Q5: I ran the optional test cases (choose 7=Yes, 1=No)

Please insert your answers into the bottom of your project program as a comment, formatted exactly as follows (so we can write a program to extract them).

```
# Questions
# Q1: 5
# Q2: 3
# Q3: 4
# Q4: 6
# Q5: 7
```