Hangman

psp-08-01:

1 Overview

In this lab you will be creating the game **Hangman.** The object of the game is to guess the word before ... well you know. This lab is intended to give you practice writing functions, reading in data from files, and using looping constructs.

2 Learning Outcomes

By the end of this project students should be able to:

- read and write programs that use functions;
- read and write programs that effectively use lists;
- read and write programs that use user input;
- read and write programs that read from files;
- read and write programs that display graphical output;
- work effectively with a partner using pair-programming;
- write an effective report that describes the students' problem solving process.

3 Pre-Lab Instructions

Do this part before you come to lab:

- Read Problem Space Chapter 8: Input and Output.
- Go to http://www.mieliestronk.com/corncob_lowercase.txt and save the word list as word_list.txt. Alternatively, this file is available in BBLearn in your lab section under lab files.
- Write pseudo code for the functions you will use to implement your game.

- Read through Python's turtle graphics api.
  https://docs.python.org/2/library/turtle.html
- Practice drawing with turtle graphics by making some basic shapes;
  Here is an example:

```
import turtle

# A simple right-sided triangle
turtle.forward(90)
turtle.left(135)
turtle.forward(90)
turtle.left(90)
turtle.forward(90)
turtle.left(135)
turtle.forward(90)
turtle.hideturtle()
turtle.exitonclick()
```

- Search google for an image to use as your game's background. Resize it to 1024 x 682 and save it somewhere that will be accessible in lab.
- Be prepared at the beginning of class to show your pseudo code to the lab aide.

4 Lab Instructions

Do this part in lab:

Part 1

The Hangman game you are going to implement has the following requirements:

1. The game must randomly choose words from the word_list.txt file.
2. The game must keep track of wrong choices and tell the user if the letter has already been guessed.
3. The game should only allow the user as many turns as it takes to hang the man.
4. The game must let one correct letter choice fill in all letters in the word that match.
5. If the user wins display "You win!" and the word guessed.
6. If the user loses display "You lose!" and the word guessed.
7. Whether they win or lose, ask if the user wants to play again.

Once your game is working and you think you have all cases covered demonstrate it to the lab aide.

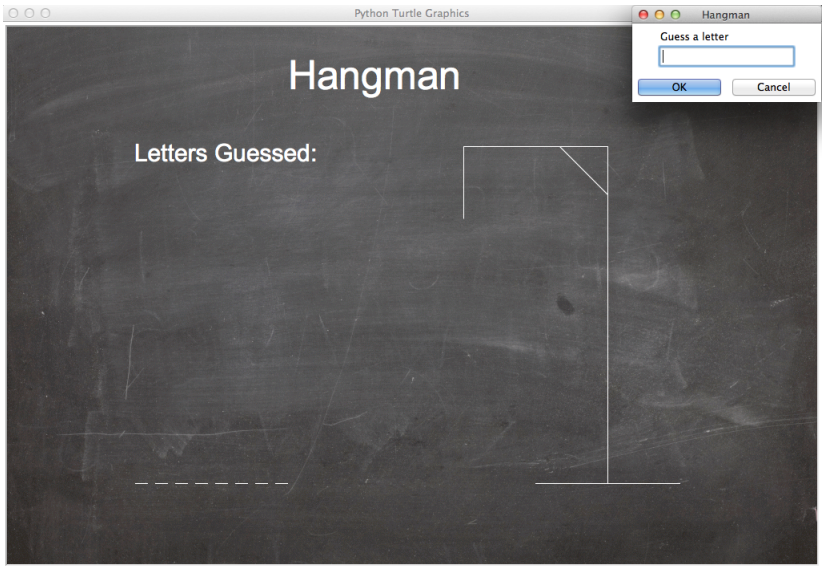
Part 2

Now that you have a working Hangman game we are going to use Python's turtle graphics to draw your game.
Here are some useful turtle functions to get you started:

```
turtle.pencolor('color')
#sets background picture
turtle.bgpic("{path/to/your/picture}.gif")
# sets the window size
turtle.setup(1024, 682)
turtle.up()
turtle.down()
turtle.home()
# returns (x,y) position of turtle
turtle.postion()
# sets the turtle position
turtle.setposition(x_coord, y_coord)
 #write to the screen, change alignment and font
turtle.write(letter, align="left", font=("Arial", 30))
# cretae pop up box to get input from the user
turtle.textinput("Hangman", "Guess a letter")
```

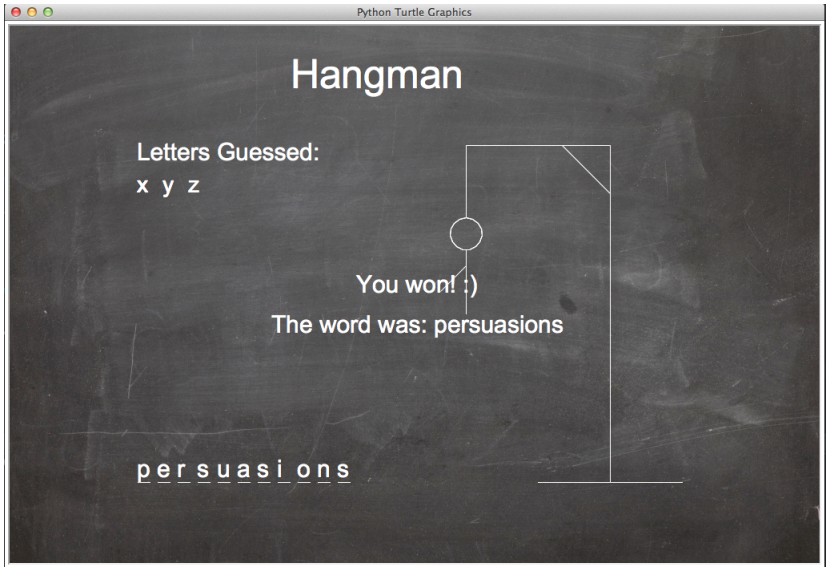
Your graphical hangman game has the following requirements:

1. Set the window size of your game to (1024, 682)
2. Set the background image for your game.
3. Give your hangman game a title and write it in the top-center of the window.
4. Draw the hanging structure.
5. Draw blanks for each letter in the word to be guessed.
6. Display each letter guessed correctly in the appropriate blank.
7. Display wrong letters guessed in the game. The position is up to you.



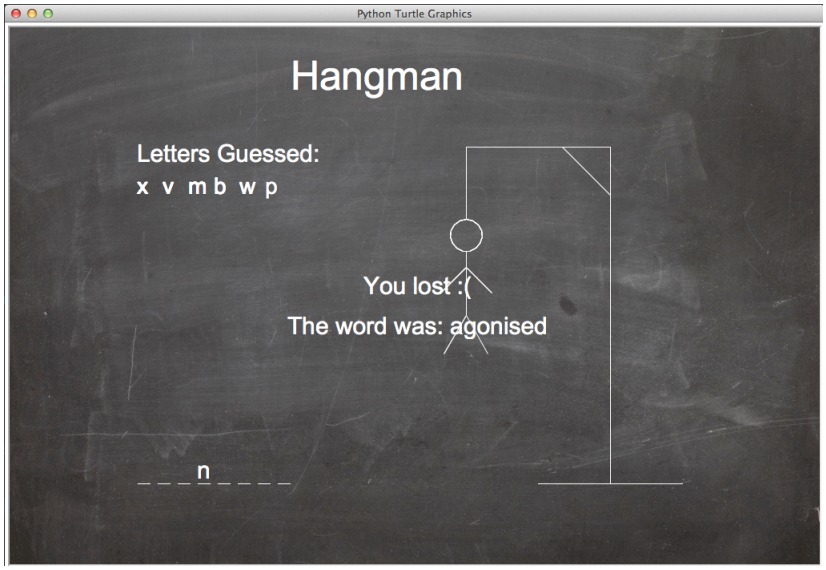
8. Draw each part of the hangman body

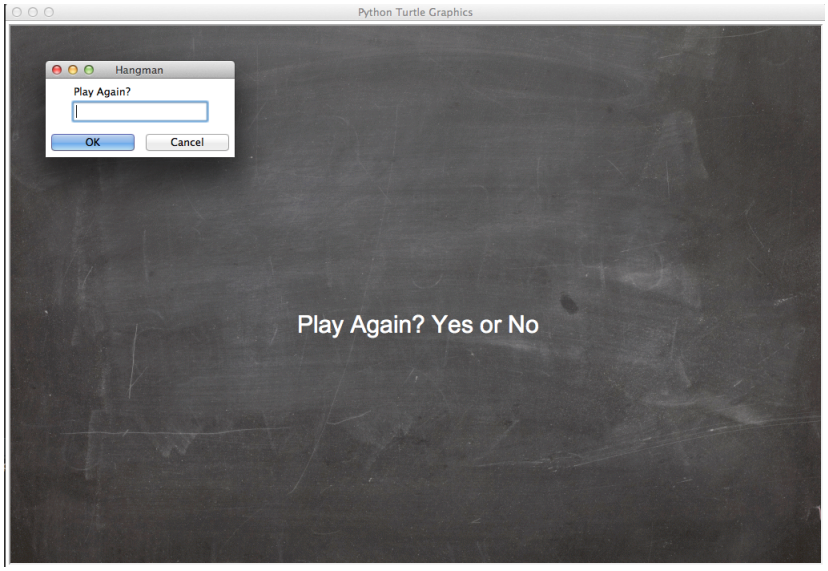
9. If the user wins display "You win!" and the word guessed in the center of the window. Have your program wait 2 seconds (import time might be useful here), then ask the player if they want to play again. Play another round if the answer is yes, or quit if the answer is no.



10. If the user loses display "You lose!" and the word guessed in the center of the window. Have your program wait 2 seconds (import time might be

useful here) , then ask the player if they want to play again. Play another round if the answer is yes, or quit if the answer is no.





Once your game is working and you think you have all cases covered demonstrate it to the lab aide.

When you have completed the lab run pep8 against your code until all formatting errors have been corrected and your code is PEP 8 compliant. See

the Getting Started lab if you need instructions on running the program, or the pep8 documentation found here.

5 Lab Report

Each pair of students will write a single lab report together and each student will turn in that same lab report on BBLearn. Submissions from each student on a pair should be identical.

Your lab report should begin with a preamble that contains:

- The lab assignment number and name
- Your name(s)
- The date
- The lab section

It should then be followed by four numbered sections:

1. Problem Statement

In this section you should describe the problem in **your** own words. The problem statement should answer questions like:

- What are the important features of the problem?
- What are the problem requirements?

This section should also include a reasonably complete list of requirements in the assignment. Following your description of the problem, include a bulleted list of specific features to implement. If there are any specific funtions, classes or numeric requirements given to you, they should be represented in this bulleted list.

2. Planning

In the second section you should describe what planning you did in order to solve the problem. You should include planning artifacts like sketches, diagrams, or pseudocode you may have used. You should also describe your planning process. List the specific data structures or techniques you plan on using, and why.

3. Implementation and Testing

In the third section you should describe how you implemented your plan. As directed by the lab instructor you should (as appropriate) include:

- a copy of your source code (Submitted in BBLearn as a .py file)
- a screen shot of your running application / solution
- results from testing

4. Reflection

In the last section you should reflect on the project. Consider different things you could have done to make your solution better. This might include code organization improvements, design improvements, etc.

You should also ask yourself what were the key insights or features of your solution? Were there alternative approaches or techniques you could have employed? How would these alternatives have impacted a different solution?

5. Partner Rating

Every assignment you are required to rate your partner with a score -1, 0 or 1. This should be submitted in the comment section of the BBLearn submission, and not in the report document. You do not have to tell your partner the rating you assign them. A rating of 1 indicates that your partner was particularly helpful or contributed exceptional effort. A rating of 0 indcates that your partner met the class expectations of them. Rating your partner at -1 means that they refused contribute to the project, failed to put in a resonable effort or actively blocked you from participating. If a student recieves three ratings of -1 they must attend a mandatory meeting with the instructor to dicuss the situation, and recieving additional -1 ratings beyond that, the student risks losing a letter grade, or even failing the course.

Colophon