Spring 2019
Computer Science I
Section 4 – TU-TH – 9:00AM – Fulton 250
Section 5 – TU-TH - 10:30AM – Fulton 250
Office Hours Monday and-Friday -10:00AM – 12:00AM
St Mary - CS Department – Of. 281.

Maíra Marques Samary PhD

TA's Office Hours – all at Fulton 160
Molly Soja (sojam@bc.edu) - M 2PM-3PM
Yicheng Shen – Mark (shenvw@bc.edu) - TH 3PM-4PM
Anna Peterson (peterspm@bc.edu) - SU NOON-1PM
Luke Ruter (ruterl@bc.edu) - SA 2PM-3PM

Assignment 4

Due date – 21/03/19 11:59 PM

General Instructions

Create a folder named ***HW1_LASTNAME_FIRSTNAME***. You will populate the folder with **ALL** of the .py files you write for this homework. To submit the homework, verify the folder includes all your .py files, compress (zip) the folder then upload to Canvas. Remember to include the following comments at the **top of each** of your .py files:

```
# author:
# assignment:
# description:
```

What to submit in Canvas?

Make sure all your files are saved in the folder HW4_LASTNAME_FIRSTNAME, then compress (zip) the folder and upload to Canvas.

If you encounter any problems in completing the assignment or in the submission process, please don't hesitate to ask for help. The sooner, the better!

IMPORTANT

Now that you are already familiar with functions, almost all your code must be inside functions! Points will be taken if you put too many line of codes outside functions.

From now on, I will not say to you the amount of .py files you will create, you can create as many as you want.

Don't forget, if your game **does not run** and shows errors when we try to evaluate and correct, you will have an **automatic 0. No arguing, no complaints!!!**

HANGMAN- 10 points

1. **Goals of this homework:**

In this homework is asked for you to create a program of the popular hangman game. The simplest version of this game is to guess a word chosen by the opponent (in this case, the computer).
The player must try to guess the word by saying letters that seem to be part of the secret word. If it is correct, all the occurrences of the chosen letter must be written in the place of the secret word.
Otherwise, a part of the body of the hanged man should be drawn, starting with the head. The game ends when the player correctly guesses the secret word, or, when the five parts of the body are drawn: (1) the head, (2) the trunk, (3) the arms, (4) the left leg and (5) the right leg.

2. **Game functionality:**
   a. **Word Selection**

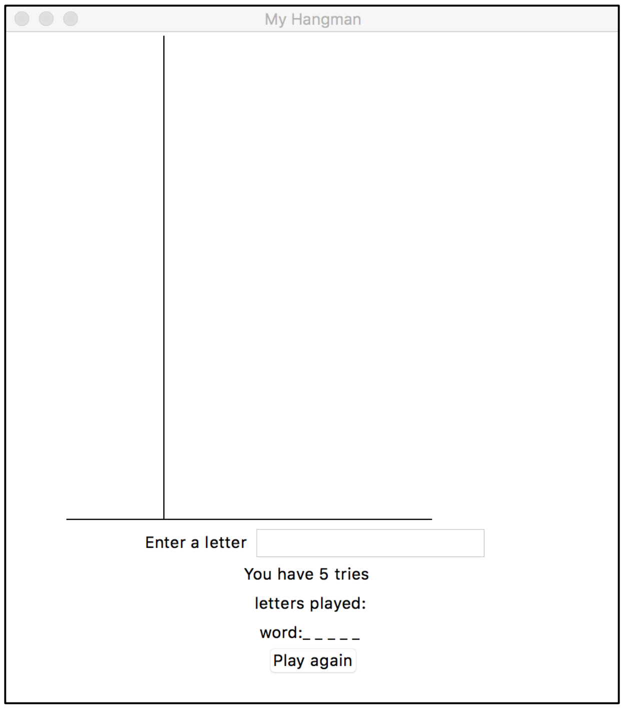
The game must be automatically select a word from a file. To do this, you must randomly select a word from those available in the words.txt file. To make your task easier, assume that all the words contained in the file are written in lowercase letters and do not contain accents or special characters.

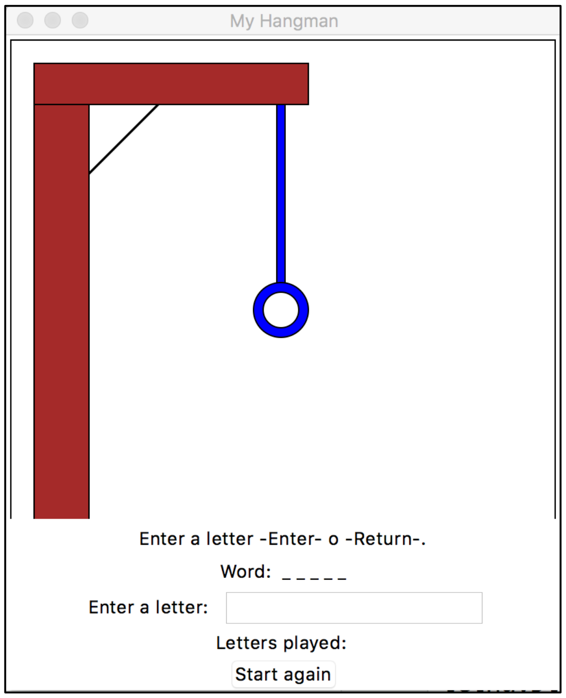
   b. **Graphical Interface**

Design and implement a graphical user interface (Window) to support the behavior of the game. You must use the library Tkinter that we saw on classes.
It is expected that the screen (window) program has, at least, the following components (widgets):

- A canvas to draw the hanged person
- A text field where the user enters the letter he wants to play
- A text label showing the secret word and the progress of the game
- A label where the letters played are shown
- A button to restart the game: the canvas must be **cleaned** and the secret word restarted

For example, when you start the game, its interface can have different aspects, such as the two below (one has just enough, and the other one has something extra):

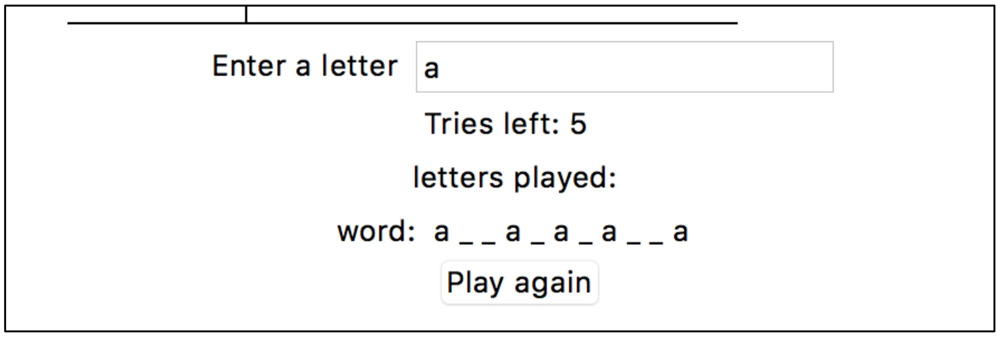




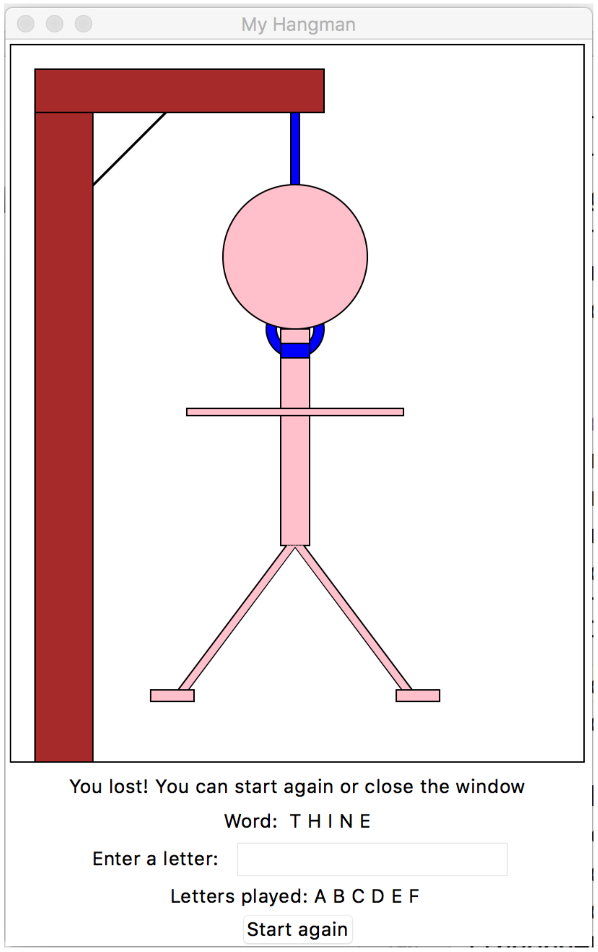
You are free to decide the appearance of your interface (screen). For example, you can draw the support where the body of the hanged person will be hanging, you can add colors to the text labels, improve the distribution of the components of the interface, among others.

3. Rules

Like the real game, if the user enters a letter that is repeated multiple times in the secret word, all the occurrences of it must be shown. For example, if the word to guess is: "abracadabra" and the player initially enters the letter "a", the interface should react as follows:



If the player does not guess a letter that is contained in the secret word (or, if it indicates one that was already shown as correct), the hanging body should be drawn on the canvas. To do this, first draw the head, then the trunk, then the two arms, then the left leg and finally the right leg. Make sure that the different elements to be drawn have an adequate size and are properly centered on the drawing canvas.



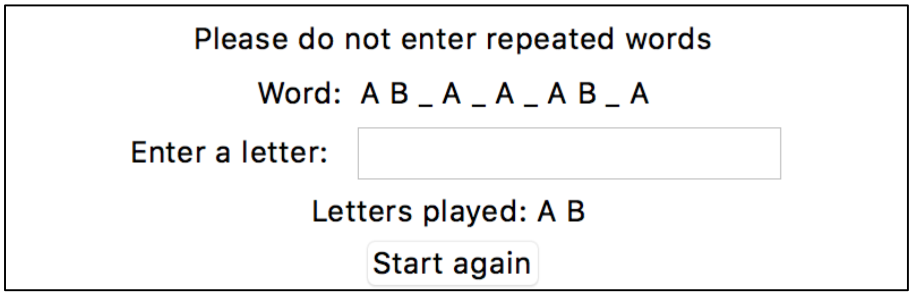
You lost! You can start again or close the window

Word: T H I N E

Enter a letter:

Letters played: A B C D E F

The game after every time the player enters a letter, must store and show the letters played to the user, and clear the space (the entry) where the player writes the letter.
If the player writes a letter that he already tried, the game should show a message saying that the player just tried this letter, as the example below:



Please do not enter repeated words

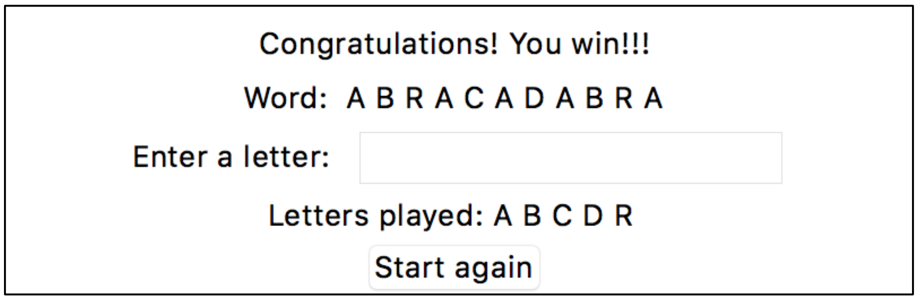
Word: A B _ A _ A _ A B _ A

Enter a letter:

Letters played: A B

If the player makes five mistakes, the body of the hangman must be completely drawn and the game must end, indicating the solution.

If the player correctly guesses the word, the game must end, and must indicate an appropriate message to the user (for example, CONGRATULATIONS !!).


Congratulations! You win!!!

Word: A B R A C A D A B R A

Enter a letter:

Letters played: A B C D R

Finally, if the user presses the button to restart the game, the drawing canvas must be cleaned and the computer must randomly select a new word from the file. After that, the program should allow the user to start a new session of the game.

TIPS (You don't have to do this way, but is a good approach to do it):

Don't let to do your homework a couple of days before submission deadline.
If you do that you will not be able to do it.
Divide your work. Suggestion:

1. Create a function that can read the file dictionary and put all the words inside a list, make your function to return this list.
2. Create a function that receives the list with all the words and chooses randomly one word from the list.
3. Create a function that draws the spaces to put the letters, an entry. You can create one or many, is up to you.
4. Create the other visual items from the screen.
5. Start getting the letters from the player, if it is one letter you show on the screen in another label (the one that shows the letters the player already tried) and you compare with the letters on the random word.
6. If the user got the letter you put the letters on the space of the secret word. If the user didn't get the letter you draw a part of the body. Remember that this game has only 5 body parts and that both arms are considered just one body part.
7. Create a button that clear all the lists, entry labels and all the variables needed and restart the game.

If you do a part of you game a day, it won't be too much and you will be able to see that you can do a lot. But don't try to do it last minute!!!

4. **Homework Rubrics**

If your code does not run you have an automatic 0

HW4 grade 10.0 need to have:

- Assignment been submitted on time
- Most of the code inside functions. You can use one or multiple .py files
- The words are randomly chosen from the file Dictionary.txt
- A screen game that have the elements:
    - a hangman (head, body, arms and two legs),
    - a place holder (entry) for the letters,
    - a place to display the letters already played,
    - a place to display messages,
    - a place to display the letters from the randomly chosen word.
    - a buttom to play again.
- If the player puts a letter and the word has the letter, it shows the letter as already played in a label, and show the letter in proper spaces on the word holder.
    - After writing a letter and pressing enter, the game must validate if it is only one letter, if it is, it shows the letter on the list, validates if the letter is on the random word, and clear the space on the letter.
    - If the player writes more than one letter, or a number, or anything else. The game must display a message saying that it is just one letter accepted and this does not count as a try.
- If the player makes 5 mistakes, the game says that the player lost and that he/she can try to play again pressing the button play again.
- When the player presses, the button play again, the game starts over. Another new word is randomly picked, the hangman should be cleared. All the labels must return to its initial state.

This time we will do the correction imagining that everybody has the 10 grade. And according to what is missing, we will take points. So, if your code **does not have** something (every line is considered "something") then the points at the end of the line will be taken:

- Most of the code inside functions. You can use one or multiple .py files **- 1.0 point less**
- The words are randomly chosen from the file Dictionary.txt **- 1.0 point less**
- A screen game that have the elements:
    - a hangman (head, body, arms and two legs), **- 0.5 point less**
    - a place holder (entry) for the letters, **- 0.5 point less**
    - a place to display the letters already played, **- 0.5 point less**
    - a place to display messages, **- 0.5 point less**

    - a place to display the letters from the randomly chosen word. **- 0.5 point less**
    - a buttom to play again. **- 0.5 point less**
- If the player puts a letter and the word has the letter, it shows the letter as already played in a label, and show the letter in proper spaces on the word holder.
    - After writing a letter and pressing enter, the game must validate if it is only one letter, if it is, it shows the letter on the list, validates if the letter is on the random word, and clear the space on the letter. **- 1.0 point less**
    - If the player writes more than one letter, or a number, or anything else. The game must display a message saying that it is just one letter accepted and this does not count as a try. **- 1.0 point less**
- If the player makes 5 mistakes, the game says that the player lost and that he/she can try to play again pressing the button play again. **- 1.0 point less**
- When the player presses, the button play again, the game starts over. Another new word is randomly picked, the hangman should be cleared. All the labels must return to its initial state. **- 2.0 point less**

You can create how many functions you want and how many .py files you want. But be careful to not lose yourself in the code.