Lab 4 / Project 1

Due Sunday, Nov 17

Note: Turn in lab 4 and Project 1 separately in Sakai. Both are due Sunday, Nov 17.

For the lab:

The lab must be done using paired partners. One student should turn in the lab, but the lab must have both students' names on it. Pick a new (or old) partner. It's time to switch. If you're running out of new people, you can repair with someone you paired with previously.

Create a file called lab4.py. Include as comments at the very top of the file: your name, your partner's name, class time, TA's name(s), and lab due date. Then include the following comments and functions:

For the Project:

The project may or may not be done using paired partners. If you choose to work alone, that is your choice. If you choose to work with a partner, be aware that you are still responsible for the project being done and turned in on time regardless of your partner's contribution. One file should be turned in, with both of your names on it and your TAs names as well.

1. Write a function that takes as an input parameter a list of words and uses a for loop to remove all words that begin with the letter 's' from the list. The function returns that list.

2. When I was a kid, I used to speak a language we called "gibberish" to my friend when we didn't want adults to understand us. For this language, every vowel would be replaced with ithag and then the vowel. So, for instance,

- For instance, the word "cat" would become "cithagat"
- Monkey would become "mithagonkithagey"

Write the comments with input, output, and test cases for a function that uses the input function (no input parameters!) to ask the user to input a word to be translated. It then uses a for loop to create a new word in the "gibberish" language. So every time a vowel is encountered in a word, it is replaced with "ithag",then the vowel, then the rest of the word.

3a. First write a function that takes as an input parameter a number. It then uses a list of letters (the alphabet – I'm including it so you don't have to type) and a for loop to generate a new word of random letters that is the input parameter long. So, given the following:

```
alpha = ['a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z']
```

if the input parameter was 4 and the first random numbers were 7, the second 22, the third 18 and the forth 3, the output string would be: "ewsd"

3b. Now write a function that takes as input a string – a word. It then uses the function in 3a to get a random word that is the same length as the input word. It uses a while loop (this is a case where I don't think it's possible to write this as a for loop) to check if the random word is the same as the input word, and continues to generate new random words until the random word is the same as the input word. It counts the number of random words that had to be generated before the input word was matched. That count number is returned from the function. (While we are assuming that all letters are lower case, and that there are no special characters, this is a very basic password guessing algorithm and could easily be modified to include upper case letters and special characters. Start with a short word (3 or4 letters long) and note the count of the number of guesses. Then slowly make the word longer and note how the time changes and the number of guesses increases. This is why it's better to have a longer password.

4. Write a function that takes as input parameter an integer – a positive number. The function should use a for loop to create a list of random numbers between 1 and 1000. The length of the list should be the input integer. The function should then return the list of random numbers.

5. Write a function that takes as an input parameter an integer x. It calls the function you wrote in 4 to generate a list of numbers x in length. It then finds the largest number in the list and removes it from the list. It returns the index (the location) of that largest value in the list.

6. Write a function that takes as an input parameter a number x. It then generates a list of x lists, all x elements long of random numbers and returns it. So if the number was 3, you might get:

[[72,21,-4],[-38,2,29][-84,62,17]]

7. Now write a function that prints out the second value of each of the lists in the list you created in 6. So, in the example above, you'd print out 21,2,62.

8a. Write your own sort function. Yes, python has a sorting function built in for lists. But someone had to write that sorting function – why not you? The function should take as an input parameter an unsorted list of numbers, and it should return the sorted list of numbers. Now, be very clear in your comments how you intend to sort the list – if you can write clearly exactly how you intend to systematically go about sorting the list, it will make it a lot easier to write the function.

8b. Who has the fastest computer? Python lets you get the current date and time (up to the microsecond) using the datetime library module. To do this you must first import the datetime library module**.** Then you create a variable that holds the current date and time. Once you've done this you can access different parts of the variable to access the different parts of time. To use this module to see how fast your sorting algorithm works, first write a third function that takes as an input parameter an integer. It calls the function you wrote in 4 to create a list of random numbers that integer long. It then calls the function you wrote in 8a to sort that list created in 4. This function will return an integer – the time it took to sort the list. To calculate the time, do the following:

At the top of your python lab file, import the datetime library:

```
from datetime import *
```

Then, right before you call the sorting function (the 9b function), set a variable to the current time:

```
now = datetime.now()
```

This takes the current date and time and places it in the variable called now. You can then print out the different parts of the time variable, e.g., now.hour (holds the current hour), now.minute (holds the current minute, now.second (holds the current second), and now.microsecond (holds the current microsecond). Use this to print out the current time. (If you like, you can write a function that calls datetime.now() and then prints out the current time).

Then call the sorting function you wrote with the list you generated, and then call datetime.now() again and print out the time. See how fast your sorting algorithm worked?

Project 1: Hangman

For the project, you can either work with a partner, or you can work on your own.

Your project is to create an on-line version of the game, “Hangman” using turtle. At the end of this description I have given you a basic outline that I used for my version. However, you may add or delete functions and parameters as you see fit, as long as the required functionality is there.

The required functionality is that there be a list of words. I’ve given you a function that will read a file of word into a list (I’ve also included a file with a list of words on my web site), although you may create a long list of potential words if you prefer. The program should choose a word randomly from the list. It should then represent the word as blanks in turtle. It should interactively allow the user to guess a letter. If the letter is in the word, the blank should be filled in. If the user guesses incorrectly, a body part should be drawn. In both cases, the guessed letters should be displayed on the screen. (If the user guesses a letter they’ve already guessed, nothing happens). If the user fills in all the blanks, then s/he should get a message saying that they guessed the word correctly, and how many guesses it took. If all the body parts have been drawn, the user should be informed that they lost and shown the word they were unable to guess. In either case, they should be asked if they want to play again, and, if so, allowed to start over with a new word.

Some turtle commands that you will find useful:

```
letter = turtle.textinput("Title Window","Guess a letter")
```

This is the basic input command that we’ve been using in python. With Turtle, we first specify that we want to put text into the Title Window. Whatever the user types in goes into the variable letter, just like with python.

```
turtle.write("The word was: " + word)
```

This is your basic print command, only it writes inside the turtle window. Unlike python’s IDLE shell, in which every print is written on its own line, turtle will write wherever the pen happens to be at the moment. So you probably want to use turtle.goto(xcoordinate,ycoordinate) to position the pen first, and then use turtle.write. Also, turtle.write lets you control, among other things, the font family and the font size. So, for instance, if I wanted to write in an Arial font that’s 30 pixels in size, I’d do the following:

```
turtle.write("The word was: " + word, font = ("Arial",30))
```

Another command that you may find useful is:

```
turtle.setheading(90)
```

This sets the direction of the pen. So when rotate right a couple of times and go forward a couple of times and draw a circle, etc. you may eventually have no idea which way forward is when you use the command, “turtle.forward(30). You can reset the direction using the setheading command. 0 is East (to the right).

To clear everything out of the turtle window, you can use:

```
turtle.clear()
```

And finally, if you want a background image in your turtle window, you can use:

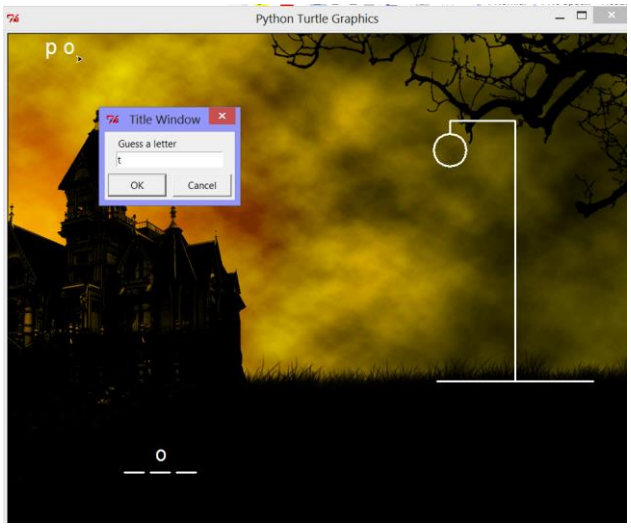
```
turtle.bgpic("spookybg.gif")
```

A couple of notes about the background picture. First, it has to be of type gif. Turtle doesn't take jpgs or pngs or anything else but gifs. I don't know why. That's just how it is.

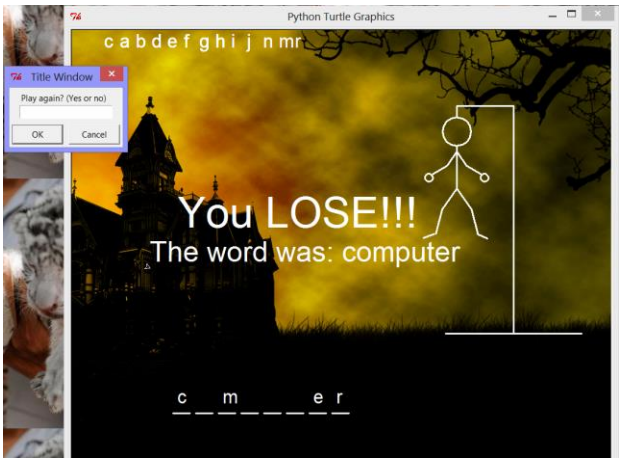
Second, if you choose to include a background image, make sure you put the background image in the same folder where you are saving your .py file. You can always specify the path, but you will be uploading this to Sakai along with your project code, and the TAs need to be able to run it, so just place the gif image in the same directory in the same folder as your project code.

Screenshots of code running:

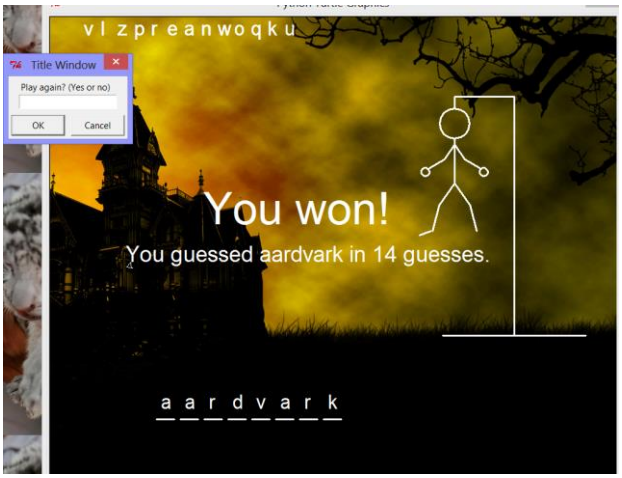
Here is a screenshot of the game in progress. In the upper left corner is a list of letters that have already been guessed. In the lower left is the blanks that represent the word to be guessed, the number of blanks equal to the number of letters in the word (in this case, it was "dog"). It has correct letters filled in. For each incorrect guess, a body part is drawn on the noose, starting with the head. A body part is not drawn when the user chooses a letter in the word. Nor is a body part drawn when the user chooses a letter they've already chosen.



Here is a losing screenshot. In this case, the user is told the word they were unable to guess. Notice that the game asks if you want to play again. If it does, you must clear the screen, choose a new word, clear out the chosen letters, and start over. If not, the game ends.



Here is a winning screenshot. Notice that it informs you of how many guesses it took for the user to guess the answer. Also, again, notice that the game asks the user if they want to play again.



Suggested Code Outline:

```
from random import *
import turtle

#takes as an input parameter a blank list.  Returns a list of words.
#reads in all the words in the file, 'wordlist.txt' (which you can download from
#my web page) and places all the words into a list, which is returned.
def readfile(listofwords):
    f = open('wordlist.txt','r') #opens file, 'wordlist.txt' for reading
                                 #reads file into the variable f
    for line in f:
        listofwords.append(line.strip())
    f.close()
    return(listofwords)


#takes as an input parameter a list of words
#returns a string – a random word chosen from the list of words
#Test case: getword(['cat','dog','happy','mother'])=> 'happy'
def getword(listofwords):


#takes two input parameters: a string (the word being guessed) and a list of letters.
#The function checks to see if each letter in the word is in the same place as in
#the list of letters.
#The function returns a Boolean value True if the word and the list of letters
#contain the same letters and False if they don't.
#Test cases: checkwin("dog",["d","o","g"] => True
#            checkwin("mother,["_","_","t","h","e","_"])=> False
#            checkwin("Happy",["h","a","p","p","y"])=> True
def checkwin(word,blank):
```

```
#takes as input parameters a string and a list: the word being guessed and the list
# of letters already guessed
#This function prints in the turtle window, “You Won!”  and then
# “You guessed in 13 guesses.”
#(Replace 13 with the number of guesses taken)
# This function returns nothing.
def Wongame(word,letters):


#takes as input a string – the word being guessed
#prints into the turtle window, “You Lose!!” and then
#”The word was happy” (or whatever the word was that was being guessed.
#returns nothing.
def Lostgame(word):


#takes as input a string – the word being guessed
#returns a list, with the ‘_’ character for every character in the word.
#test cases: makeblank(‘cat’) => [‘_’,’_’,’_’]
             makeblank(‘happy’) => [‘_’,’_’,’_’ ,’_’,’_’]
def makeblank(word):

#no input parameters and nothing returned.
#Clears the turtle window
#sets the pen color
#sets the window’s background image
#then draws the hangman structure, e.g.,
```

```
def drawstruct():
#takes as input parameters 3 parameters: a list of blanks (each blank representing
#a letter in the word being guessed), and two numbers: the length of the blank to be
#drawn, and the length of the space drawn in between each blank.
#Returns nothing.
#draws in the turtle window a blank for each blank in the list, with a space between
#each blank
def drawblanks(blank,size,space):

# This function is the heart of each round of hangman.
# It takes as input parameters:
#      a string: word – the word being guessed
#      a list: letters – the list of letters already guessed
#      an integer: wrong – the number of wrong guesses so far.
# It returns nothing.
#
```

```
# It first calls the drawstruct function to draw the basic hangman structure.  It
# then makes a list of blanks by calling the makeblank function.
# It draws the blanks on the turtle window by calling the drawblanks function.
# It then loops until either the word is successfully guessed or the hangman is
# hanged ( wrong == 10).
# In the loop it calls the writeguessed function to write out the letters
# already guessed.
# It uses the turtle.textinput function to ask the user to ‘Guess a letter’.
# If the letter is in the word being guessed, placeletter function is called, the
# letter is appended to the list of letters already guessed, and, if the checkwin
# function indicates the word has been successfully guessed, the Wongame function is
# called.
# Otherwise if the letter is not in the word, the drawbody function is called,
# the letter is appended to the list of guessed letters, and, if wrong guesses is up
# to 10, the Lostgame function is called.
def playround(word,letters,wrong):


#takes as an input parameter the list of letters that has been guessed and
#returns nothing.
#writes the guessed letters at the top left of the turtle screenusing a loop and
#turtle.write.
def writeguessed(letters):

################################################################################
# Functions that draw out all of the different body parts of the hangman
################################################################################
#takes as an input parameter an integer: the number of wrong guesses so far.
#based on that number, it calls the function that draws the appropriate body part.
#it returns an integer: the number wrong + 1.
def drawbody(wrong):

#No input parameter, nothing returned.  Draws the head in the turtle window.
def drawhead():

#No input parameter, nothing returned.  Draws the torso in the turtle window.
def drawtorso():

#No input parameter, nothing returned.  Draws the right arm in the turtle window.
def drawrightarm():

#No input parameter, nothing returned.  Draws the left arm in the turtle window.
def drawleftarm():

#No input parameter, nothing returned.  Draws the right leg in the turtle window.
def drawrightleg():

#No input parameter, nothing returned.  Draws the left leg in the turtle window.
def drawleftleg():

#No input parameter, nothing returned.  Draws the right hand in the turtle window.
def righthand():

#No input parameter, nothing returned.  Draws the left hand in the turtle window.
def lefthand():
```

```
#No input parameter, nothing returned.  Draws the right foot in the turtle window.
def rightfoot():

#No input parameter, nothing returned.  Draws the left foot in the turtle window.
def leftfoot():
################################################################################

#Takes as input parameters:
# letter: a string of one character - the letter guessed
# word: a string – the word being guessed
# blank: a list – the list of blanks that represents the word being guessed
# size: an integer – the length of the blank line being drawn in turtle
# space: an integer – the length of the spaces between blank lines drawn
#
# This function modifies the blank list, by substituting the letter for the
# '_' character in the list everywhere the letter occurs in the word.
# It also writes in turtle over the appropriate blank line the letter guessed at the
# appropriate places.
# It returns a string – the blank list with the new letters in place
# Test cases:
#     placeletter('p','puppy',['_','_','_','_','_'],30,10) => ['p','_','p','p','_']
def placeletter(letter,word,blank,size,space):

#takes as an input parameter a list of words.
#Asks the user if they want to, 'Play again?' and continues looping until the
#user enters, 'no'.
# For each new game, sets the list of letters back to an empty list, gets
# a new word by calling the getword function, and then plays a round of hangman
# by calling the playround function.
# returns nothing.
def game(listofwords):

#starts the game by calling the readfile function to read in the list of words, and
#then calling the game function with that list.
#no input parameters and no output parameters.
def main():
    ls = readfile([])
    game(ls)

main()
```