

Lab 4: Hangman (Strings, Lists)

Due Wednesday, Apr 22

For the Project:

You may work with a partner on this lab

Hangman

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. You may add extra help functions if you so choose.

The project requires a list of potential words the user has to guess. 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.

To see how this works, view the video on my web site for an example

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:

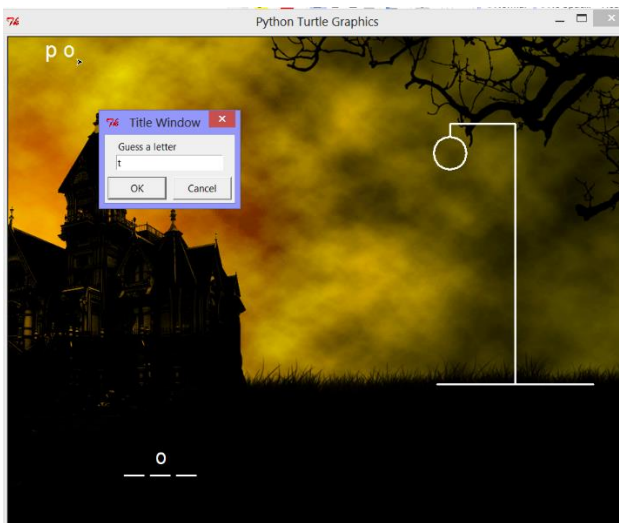
```
turtle.bgpic("spookybg.gif")
```

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.

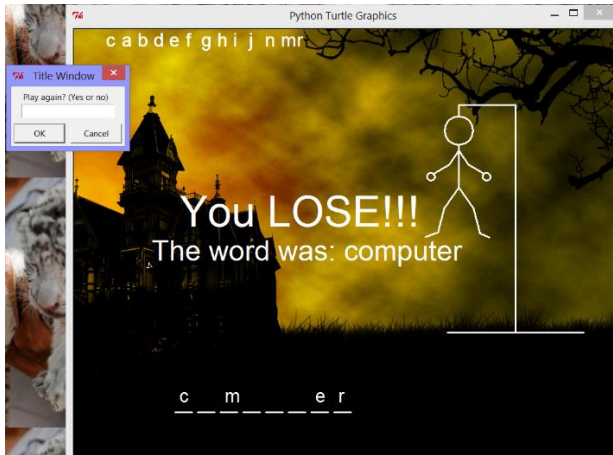
Second, if you choose to include a background image, make sure you put the background image in the same folder where you are saving your .py file. You can always specify the path, but you will be uploading this to Sakai along with your project code, and the TAs need to be able to run it, so just place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

Screenshots of code running:

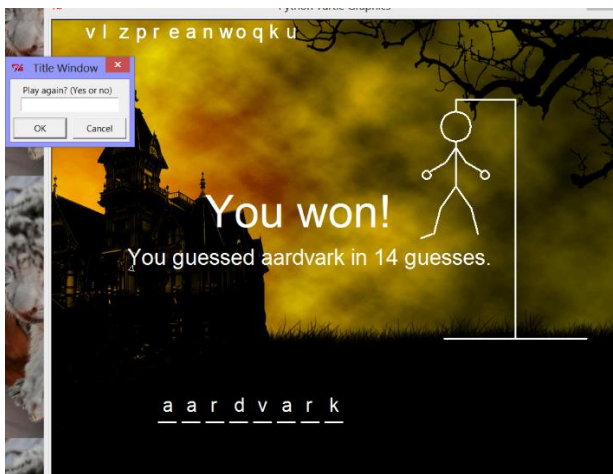
Here is a screenshot of the game in progress. In the upper left corner is a list of letters that have already been guessed. In the lower left is the blanks that represent the word to be guessed, the number of blanks equal to the number of letters in the word (in this case, it was "dog"). It has correct 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.



Functions you will need to write:

```
#Function 1:
#drawstruct()
#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.,
```



(3 pts) def drawstruct():

```
#####
```

```
# Functions 2-10:
```

```
# 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.
```

(2 pt)def drawbody(wrong):

```
#No input parameter, nothing returned.  Draws the head in the turtle window at the
```

```
#appropriate location (on the gallows)
```

(1 pt)def drawhead():

```
#No input parameter, nothing returned.  Draws the torso at the appropriate place in
```

```
#the turtle window.
```

(1 pt)def drawtorso():

```
#No input parameter, nothing returned.  Draws the right arm at the appropriate place
```

```
#in the turtle window.
```

(1 pt)def drawrightarm():

```
#No input parameter, nothing returned.  Draws the left arm at the appropriate place
```

```
#in the turtle window.
```

(1 pt)def drawleftarm():

```
#No input parameter, nothing returned.  Draws the right leg in the turtle window.
```

(1 pt)def drawrightleg():

```
#No input parameter, nothing returned.  Draws the left leg in the turtle window.
```

(1 pt)def drawleftleg():

```
#No input parameter, nothing returned.  Draws the right hand in the turtle window.
```

(1 pt)def righthand():

```
#No input parameter, nothing returned.  Draws the left hand in the turtle window.
```

(1 pt)def lefthand():

```
#No input parameter, nothing returned.  Draws the right foot in the turtle window.
```

(1 pt)def rightfoot():

```
#No input parameter, nothing returned.  Draws the left foot in the turtle window.
```

```

(1 pt)def leftfoot():
#####

#Function 11:
#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'
(3 pts)def getword(listofwords):

#Function 12:
#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.
(3 pts)def Wongame(word,letters):

#Function 13:
#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.
(3 pts)def Lostgame(word):

#Function 14:
#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
(7 pts)def checkwin(word,blank):

#Function 15:
# take as an input parameter:
# a string - the word being guessed
# It creates a new list of blanks, with the '_' character for every
# character in the word.
# It returns that list
#test cases: makeblank('cat') => ['_','_','_']
#             makeblank('happy') => ['_','_','_','_','_','_']
(5 pts)def makeblank(word):

#Function 16:
# takes as input parameters 3 parameters:
# a list of blanks (each blank representing a letter in the word being guessed),
# an integer: the length of a single blank to be drawn
# an integer: the length of the space drawn in between each blank
#Returns nothing.

```

```

#Uses a loop to draw in the turtle window a blank for each blank in the list, with
#a space between each blank
(5 pts)def drawblanks(blank,size,space):

#Function 17:
#takes as input 3 parameters:
#    a list: the list of letters that have been guessed
#    an integer: the x coordinate of where the guessed letters will be printed
#    an integer: the y coordinate of where the guessed letters will be printed
#it returns nothing.
#writes the guessed letter (the last letter in the list of letters) at the top left
of the turtle screen (Note that you don't want to write over previously guessed
letters, so you will need to calculate the appropriate x coordinate to go to.
(3 pts) def writeguessed(letters):

# Function 18:
# 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 loops to modify 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','_']
# placeletter('t','byte',['b','_','_','_'],30,10) => ['b','_','t','_']
(10 pts) def placeletter(letter,word,blank,size,space):
#####
These functions are written for you and are in the hangman.py file

#Function 19:
#takes 2 input parameters:
#    a list: a list of words.
#    a string: holds either "yes" or "no"
#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 playground function.
# returns nothing.
def game(listofwords):
    x = "yes"
    while (x.lower() == "yes"):
        letters = []
        word = getword(listofwords)
        playground(word,letters,0)
        x = turtle.textinput("Title Window","Play again? (Yes or no)")

#Function 20:
#This function initializes values used in the a_round function, 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.
#
# The function calls the drawstruct function to draw the basic hangman structure.
# It then calls the makeblank function to make a list of blanks, and sets the
# variable blank to be that returned list of blanks.
#
# It sets the following variables:
# size (for the width of the blanks drawn on the page
# space (for the width of the spaces between the blanks
# xcoord (for the xcoordinate for blanks to be drawn)
# ycoord (for the y coordinate for blanks being drawn)
# gxcoord (for the x coordinate of the guessed letters)
# gycoord (for the y coordinate of the guessed letters)
#
# It draws the blanks on the turtle window by calling the drawblanks function.
# It then calls the a_round function.
def playround(word,letters,wrong):
    turtle.setup(900,1000,100,100)
    drawstruct()
    playvar = True
    blank = makeblank(word)
    size = 30
    space = 10
    drawblanks(blank,size,space)
    xcoord = -400
    ycoord = 400
    line = 0

    while playvar:
        print(letters)
        writeguessed(letters)
        letter = turtle.textinput("Title Window","Guess a letter")
        #letter = input("Guess a letter: ")
        line = line + 1
        if letter in letters:
            print("sorry, that letter has already been guessed")
        elif letter in word:
            placeletter(letter,word,blank,size,space)
            letters.append(letter)
            if checkwin(word,blank):
                Wongame(word,letters)
                playvar = False;
        else:
            wrong = drawbody(wrong)
            letters.append(letter)
            if wrong == 10:
                Lostgame(word)
                playvar = False;
        print(blank)
    return

```

#Function 21

#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,opened):  
    if not opened:  
        f = open('wordlist.txt','r')  
        return(readfile(listofwords,f,True))  
    else:  
        line = f.readline()  
        if not line:  
            f.close()  
            return(listofwords)  
        else:  
            listofwords.append(line.strip())  
            return(readfile(listofwords,f,True))
```

#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():  
    #Note: for testing purposes, you may want to uncomment out the first line and comment  
    #out the second line. When you have everything working, then you can uncomment out  
    #the second line and comment out the first line.  
    #    ls = ['aardvark','computer','dog','sesquipedalian','binary','gigabyte']  
    ls = readfile([],0,False)  
    print(ls)  
    game(ls,"yes")
```

```
main()
```

Extra Credit (15 pts):

Modify the game so that it is a two-player game. As soon as one player guesses a letter that is not in the word, the players switch. The player who guesses the last letter wins.