CIS 122 Fall 2015 Project 7
Due Monday, November 30, 11:59 PM

Briefly

Submit several Python 3 programs
Your programs are worth a total of 40 points

"**http://www.cs.uoregon.edu/Classes/15F/cis122/data/**xxx.txt"

where you must change xxx.txt to the correct name of the text file

Notice that the url is **case-sensitive** after .edu
Classes, not classes,
15F not 15f
cis122 not CIS122
data not Data not DATA

Test your programs -- did they work right? -- before uploading to Canvas.

P7_TurtleWalk.py 10 points

Learning Objectives:
Simulate a random walk graphically
Provide an opportunity for simple user interaction
(choice from a menu)

Note: be sure to import both random and turtle such as this way:

```
import random
import turtle as t
```

This is a "random walk" – the turtle will take "steps" of 5 to 10 pixels, but in a random direction.

At each step, use random.randint(0, 9) * 36 choose how far to turn left.

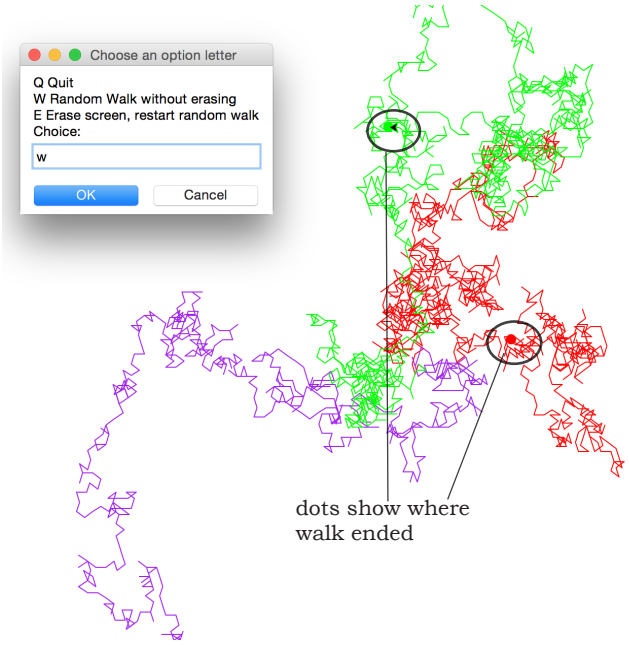
Set the turtle in motion taking 1000 turns and steps.

Mark the turtle's end postion by

```
t.dot(10)  # Draw a dot
```

In a living cell, molecules bump each other billions of times a second, a chaotic process that produces something like a random walk. Your program will simulate this process.
Rubric 5 points

1 point import both turtle and random
1 point create a loop that runs 1000 times
2 point the turtle turns a random number of degrees
such as 0 36 72 108 144 180 216 252 288 324
Others (0 45 90 etc could also work)
Units of 360/n where n is some number 4 through 12 could all work
1 point Each time through the for loop, the turtle moves forward the same number of steps (8 steps works nicely)
1 point the turtle draws a large dot to mark its final position
1 point You define and use a **jumpto(x, y)** function to jump to x,y without making a mark
2 point Use a while loop to give users 3 options
Q Quit
W Walk randomly for 1000 steps
E Erase screen and walk randomly
1 point Each random walk gets a different color for the steps such as red for 1st random walk, blue for 2nd, purple for the 3rd and so on. If your user requests more walks than you have colors, just start re-using colors.
Erase screen option will also start with the first color again.



The turtle provides a
`option = t.textinput(title, menu).lower()` function that may be useful. But you can also try doing something like
`option = input(menu).lower()`

P7_Scrabble_Linear.py 10 points

Learning Objective:
Craft a linear search (looking one by one through a list of some 260,000 words) working on a sorted list, providing quick exit . (Looking for 'aag' should very very quickly report not found, where looking for 'zzzz' not found will take far longer.)

Your basic logic in a linear search of a list of word (this is the idea, not Python statements)

Looking for item
index = 0
for each word in the list
if word matches the item looked for
return True, index
if word is beyond item (such as "cat" vs item "capp"'\
then we cannot find the item in the list
so take an early exit from loop
return False, -1
Finally if you reach end of list
return False, -1

Start by using the small **sowpods_short.txt** file.
Read the file's contents into a list.

Search loop
Get a word to look for from user
Then search using a linear search.

Example

apple
filbert
loganberry
pumpkin

You are searching for danish

Is it apple? No, keep searching
Is it filbert? No, but since filbert comes after danish, and the list is in alphabetic order, danish cannot be in the rest of the list, so break out of the loop.

When you have it working well on the small file, try a larger file. Test your search on **sowpods.txt** It is in Canvas and on the web as well.

1 point Read data from disk or web and store it in a word_list. Note that no sublist is needed; the file only contains one word per line.

4 points **linear_search** function returns True and the index of the item in the list when found, or False, -1 when not found.

2 points Linear search fails whenever the word in the list is past (>) the item searched for. In other words, your linear search will seldom search every item in its word_list.

3 points **while loop** repeatedly asks for an item to search, or gives some way to quit. When working with an item, does a linear search.

P7_Scrabble_Binary.py 10 points

Learning Objective:
Craft a binary search – it cuts a list in half, checks the mid position to see if the word at that position is > item to look for. If so, it set right to mid - 1 to enable a search of the right side. If not, look in the right side by setting left to mid + 1. Repeat the process.

Start by using the small **sowpods_short.txt** file.
Read the file's contents into a word_list.

The binary search function returns 2 items - a True value and an index number telling where in the list the match was found (or a False, -1 if not found).

```
def binary_search(item, the_list):
    """ str, list -> bool
        Binary search for item in the_list
        if in list, return True,
            index of 1st matching item
        else return False, -1
    """
    left = 0
    right = len(the_list) - 1
    while left <= right:
        mid = (left + right) // 2  # integer
        if the_list[mid] < item:
            # Search the right (later) area
            left = mid + 1
        else: # the_list[mid] >= item
            # Search the left (earlier) area
            right = mid - 1
        #end if
    #end while
    if left >=0 and left < len(the_list) and
        the_list[left] == item
        return True, left
    else:
        return False, -1
    #end if
#end def
```

When you have it working well on the small file, try a larger file. Test your search on **sowpods.txt** It is in Canvas and on the web.

Rubric

1 point Read data from disk or web and store it in a word_list. Note that no sublist is needed; the file only contains one word per line.

4 points **binary_search** function returns True and the index of the item in the list when found, or False, -1 when not found.

2 points Binary search returns False, -1 when you search for item 'a' and item 'zzzz'

3 points **while loop** repeatedly asks for an item to search, or gives some way to quit. When working with an item, does a binary search.

P7_Dictionary.py 10 points

Learning Objective:
Build a dictionary from the contents of a file. Retrieve values from the dictionary for keys you use.

Read the file **birds.txt (or the larger birds2.txt)** either from a text file (you can download it from Canvas Files > Example Programs > Week8) **or** on the web at **http://www.cs.uoregon.edu/Classes/15F/cis122/data/birds.txt** or **birds2.txt**

Build a **bird_to_counts** dictionary where the bird name such as '**duck**' or '**blue jay**' is the key, and the value is a count of how many times the bird has appeared so far in the file.

You will need to check whether your bird is already in the dictionary; if so, add 1 to the bird count. If not, you need to create a new dictionary entry by assigning that dictionary-key combination a count of 1. Page 215 of your text has some sample code.

After creating the dictionay, use a **for key in dictionary** loop to get each key, and use the key to get its count from the dictionary.

Format the data neatly for presentation, in order by bird species such as this

```
  4 Blue Jay
 16 Duck
  9 Goose
 11 Hummingbird
```

and so on.

Rubric 10 points total

1 point Start with an empty dictionary

4 points Read the birds.txt file and put its contents into the dictionary

You can read and update the dictionary directly or you may choose to read the file into a birds_list, then process the birds_list into dictionary entries.

3 points Use a for key in dictionary or for key in sorted(dictionary) loop to access the contents of the dictionary.

2 points Create a neatly formatted printout of the dictionary contents.

Title
count bird species

Bonus

3 points Figure out how to get the total of all bird counts, then show the percent each species' count is of the total count

Also display the total of the bird counts.