CS410P

LAB 10L

Purpose

The purpose of this assignment is to give you practice with writing functions, working with tuples and lists (and using them in a loop as well).

Problem

In an earlier lab you wrote a program to take a pair of (x, y) values and determine where it was on a Cartesian plane. Modify this program to create a function called where_is_xy() that takes x and y as arguments and returns one of the following strings based on where it is located: First, Second, Third, Fourth, Origin, X-Axis, Y-Axis. You must not print anything inside your where_is_xy() function but only the **caller** of this function should print the user input values (i.e. x and y) as well as the return value from the function.

Extend this program to make several calls to where_is_xy() until the user decides to input a non-numeric value to stop the program. Every (x, y) pair is stored in a list as a tuple. At the end of user input, print the locations of each point and compute the "distance" between all adjacent items of the list where distance D is given by:

$$D = \sqrt{dx^2 + dy^2}$$

where dx is the difference between the x-coordinates of the points and dy is the difference between the y-coordinates of the points

Program Flow

Program flow must proceed as follows:

- Your main function accepts a pair of numbers until a non-numeric value is entered. As each pair of numbers is accepted, the pair is stored as a "tuple" inside a list. So each of the list items is a tuple of numbers.
- For accepting inputs you must use a tuple to capture a pair of values at the same time, i.e. the user should be able to input x and y values on the same line (with one input statement). See the example mentioned in the lecture notes on "Tuples & Lists" if you are not sure how to do this. (Keep in mind that you will have to convert each value from string to numeric anyway, it's a two-step process)
- Once the input is complete, print the entire list to show all the pairs of numbers that were accepted (you can print the list object with the brackets).
- Next you need to navigate through the list using a for loop. When you use the for loop iterating through a list object, the loop counter variable is the "item" of the list. Note that each item of the list in this case will be a tuple (which is a pair of x and y). As you pick up each item of the list you must call your where_is_xy() function passing to it the x and y values as arguments to the function.
- Your where_is_xy() function determines where the point lies and returns a "string" variable to the caller (do not print anything in this function).
- The caller (in this case your main function) prints the x and y values as well as the returned value from your function call to where_is_xy().
- Finally you need to navigate through the list and get the distance between adjacent points. You can use any kind of loop you wish (while or for), keep in mind that the distance is computed between the first and second points, second and third points, etc. If there are less than 2 pairs of numbers input there is no distance to be computed.

Program Structure

Two files need to be used. You are being provided starter files just so that you are clear about what each file should contain. Submit both files location.py and main.py after you fill them with the appropriate source code.

Sample Output

Available in the public folder

Grade Key

A	Comments (including Name, brief description about program)	**5**
B	main function or identifying main entry point MUST be present	**5**
C	main function: accept pair of values separated by space (using a tuple)	**5**
D	main function: continuously accept pairs of values until a non-numeric value is entered	**10**
E	main function: store each pair of values as a tuple item in a list (list of tuples)	**10**
F	main function calls where_is_xy passing the pair of values and gets back a string value for result	**5**
G	Function where_is_xy() computes which quadrant/origin/axes point lies and returns a string (if printing is done within the function: -5)	**15**
H	Main function prints list location of each pair as well as prints the entire set of values (if printing is not done in main: -5)	**10**
I	Computing distance between adjacent points is accurately done within a loop (can use any loop, any form of iterator)	**25**
J	No global variables used, no code present outside of any function (unless you are using “constants”)	**10**