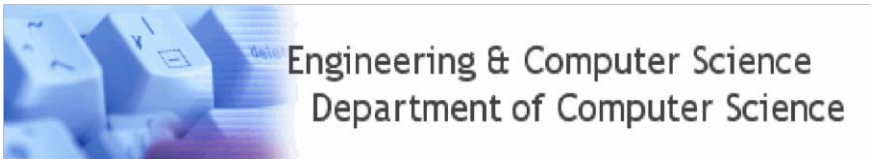

CS161 Programming and Problem Solving
Winter 2015
Due Jan. 21

Part 1: Jogging activity recorder

Exercise 2-1 is intended to introduce you to developing simple algorithms. This will include looping and decision making. You are required to use at least one 'while' loop and one 'if' construct in this program.

For your New Year's Resolution, you have started getting exercise by running. To keep track of your progress, you would like to write a Python program to keep track of how far you've gone and your average time per mile. You would like to keep a running total, so your program will start off by asking how far and how much time you've previously run.

Your program will behave as follows:

(1) The program will prompt the user for the previous distance run, which will then be entered as a decimal value with 0, 1 or 2 decimal places (miles, tenths and/or hundredths of miles).
(2) The program will then prompt the user for the previous total time in minutes spent running. The user will respond by entering the number of minutes with 0, 1 or 2 decimal places. You only enter minutes, rather than hours and minutes. For example, 2 hours and 5 minutes would be entered as 125 minutes.
(3) The program will then assume that new jogging sessions have been completed, and will prompt the user to enter the new distance and time for a session.
(4) The most recent distance will then be added to the previous total distance, and the most recent time will be added to the previous total time.
(5) The program will then display the average pace of the most recent running activity.
(6) The program will then return to Steps 3 and 4, continuing to prompt and accept distance and time information for other running sessions from the user, and displaying the average pace. This will continue until the user enters "ALL DONE" (two words separated by one space, all upper case) in response to being prompted for distance.
(7) Once the user has entered "ALL DONE" the program will display the new total running time and the new average pace for all running activity.

Sample Output (User input in Bold):

```
Welcome to the Running Activity Tracker
Please tell me the previous total distance in miles: 12.4
Please tell me the previous total time in minutes: 97.4
Please tell me the most recent running distance in miles: 2.3
Please tell me the most recent running time in minutes: 18.5
Average pace for this session: 8.04 minutes per mile.
Please tell me the most recent running distance in miles: 2.1
Please tell me the most recent running time in minutes: 17.3
Average pace for this session: 8.24 minutes per mile.
Please tell me the most recent running distance in miles: ALL DONE

New total distance: 16.8 miles.
New total time: 133.20 minutes.
Average Total Pace: 7.93 minutes per mile.
```

Design Considerations

- To accumulate a running total, use the total on both sides of the equal sign. For example:
  - `totalDistance = totalDistance + newDistance`
- To show 2 (or any number of) decimal places, you can use the round function:
  - `print ("Average Pace: ", round(avgPace))`
- Remember that input always returns a string type. To stop your loop, you can use an "if" statement to check the new distance value. If the new distance value is not equal to "ALL DONE," then you can convert it to a float, and then ask the user for the new time value. If the new distance value *is* equal to "ALL DONE," then you can stop the loop either with a break statement or as part of the while loop condition.

Submission Instructions
Save your Python file under the name hw2-1_LastName-FirstName.py and submit it using D2L. On the D2L class page, go to Activities and then Dropbox, and then click on homework 2. Click the "Add a File" button, upload your Python program, and then click Submit.

Part 2: Dice Rolling.
Exercise 2-2 is intended to exercise your skill in looping and decision-making in Python. You are required to use at least one 'while' loop in this program.
Write a program in Python to generate a dice rolling game. Many games use dice that have more than 6 sides. For example a 4-sided dice is shaped like a pyramid, and a 12-sided dice has faces that are shaped like pentagons. So your program should first ask how many sides the dice has, and then ask how many times they would like to roll the dice. Your program will then pick a random number to simulate the dice roll, using 1 as the minimum, and the number of sides of the dice as the maximum value. For example, your program would generate a random number between 1 and 6 for a 6-sided dice. You will then "show" the dice by displaying asterisks to simulate dots, and then it will specify the number of dots. For example, if your program generates a 3, then it will display "***", and then say "3 dots." Note that "1 dot" is singular, so your program should not say "1 dots." As you generate dice rolls, keep a running total. For example, 3 dice rolls of 1, 3, and 5 would be a total of 9. See below in the Design Considerations section to see how to generate random numbers.

Sample Output 1:
```
Welcome to the dice roll simulator.
How many sides does the dice have? 4
How many times would you like to roll the dice? 3
** , 2 dots.
**** , 4 dots.
* , 1 dot.
Total value of all rolls:  7
```

Sample Output 2:
```
How many sides does the dice have? 10
How many times would you like to roll the dice? 4
** ,2 dots.
**** ,4 dots.
******* ,7 dots.
****** ,6 dots.
Total value of all rolls:  19
```

Design Considerations

- To simulate a dice roll, use the random module. To use the random module in your Python program, put this line at the top of your program file:
    - `import random`
    - Now you will be able to generate random numbers:
    - `diceRoll = random.randint(1, diceSides)`
        - Notice that diceSides is a variable that stores the user input for how many sides the dice has.
- To print some number of asterisks, remember that you can use multiplication with strings to repeat them. To print 3 asterisks, you might use something like this:
    - `print ("*" * 3)`

Be sure to use the proper format for naming your file (hw2-2_LastName-FirstName.py) and submit it using D2L.