[Practice] **Timed Lab 1 – Python Fundamentals**

This is a Practice Timed Lab; this Timed Lab is worth 21 **Attendance & Participation** points.

For this Timed Lab, you *may* use	However, you *may not*
• Course notes • Homeworks • Recitation assignments • Other course material • Any material you may find on the Internet that don't involve communicating "live" with other people.	• Communicate with other people/students in real-time via any means. This means no Facebook, email, Piazza, IM, IRC, cell phones, Google Talk, smoke signals, etc. • Share code with other students. • Look at other students work.

The TAs will be available to answer clarifying questions about the problem, but they are not permitted to give assistance with debugging code, program logic, etc. You will have an entire recitation period to work on this assignment; this time begins *exactly* when your recitation begins and ends *exactly* when your recitation ends: No extra time will be given if you arrive late, except in highly extenuating circumstances that must be approved by Dr. Summet.

T-Square will not permit any late submissions; ensure that you submit your code to T-Square several times to prevent earning a zero due to you being unable to submit. Your TAs will give a verbal warning 10 and 5 minutes before the end of the recitation period; you should submit at these times. If you are taking this timed lab out of section (with approval from Dr. Summet or the Head TA), please e-mail your code to your grading TA by the end of the recitation you are in. **Modifying your code after you leave the recitation room will result in a grade of zero.**

In your collaboration statement, if you use code from somewhere that is *not* a class resource (i.e. not listed on the course calendar), please list where this code came from. Ensure that you fill out the header at the top of the file.

Problem Description:

This timed lab will consist of some Python functions that you must write. The first function you will write involves modifying elements in a list based upon their index in the list and the data contained in the element.

Function 1 - numFilter:

Write a function called numFilter that takes in one parameter, a list. This list may contain any number of items, which may be of any Python type. You should process each item in the list, and perform the following actions on each item:

1. If the item is a positive number (either an integer or a float), replace that item in the list with the number times the position of the item in the list (i.e. its index). However, if the index of the number is 0, you should multiply the number by 1 instead of 0.
2. If the item is a negative number (either an integer or a float), you should replace the number with the string "-".
3. If the item is not in either of the two categories above (Note: 0 is neither positive or negative), replace the item with "*"

Once processed, you should return the list.

Test cases:

```
numFilter([0,-30,9,("hello",3,6),-3,"here"]) --> [‘*’, '-', 18, '*', '-', '*']
numFilter([10,3,-5,0,'Cat']) --> [10, 3, '-', ‘*’, '*']
```

Function 2 – stddev:

This function, named stddev, should take in one parameter, a list of numbers which may be either integers or floats. This function will compute the population standard deviation, which is computed using the following formula.

1. Compute the average of the list.
2. Subtract the average of the list from each data point, then square this difference.
3. Average all of the differences you found in step 2.
4. Take the square root of the average found in step 3. This number is the standard deviation.

You should return the resulting standard deviation.

Test Cases:
stddev([2,4,4,4,5,5,7,9]) → 2.0
stddev([2,4,4,0,5,5,7,9.7,21.5,7.2]) → 5.616618199593062

Function 3 – isAnagram:

This function named isAnagram, which takes in two parameters, both strings, which will return True if the given string is an anagram, and False if it is not. The two string are anagrams if both strings contain the same number of each letter in them. For example, “mary” and “army” are anagrams. For this program, you will only be dealing with single-word anagrams that also lack punctuation. You may assume that all strings will be lowercase characters.

Test Cases:
isAnagram(“aardvark”, “varkarad”) → True
isAnagram(“aardvark”, “anteater”) → False
isAnagram(“army”, “mary”) → True

Grading:

numFilter:

+1: Correct function header
+2: Handles all positive numbers correctly
+2: Handles all negative numbers correctly
+1: Handles all other data correctly.
+1: Returns the list after manipulations have been completed.

stddev:

+1: Correct function header
+2: Correctly finds list average.
+2: Correctly finds difference from mean for each number, squared.
+1: Correctly averages and takes sqrt of differences squared.
+1: Returned the population standard deviation as a float.

isAnagram:

+1: Correct function header
+2: Correctly finds that strings of different length are not anagrams
+3: Correctly finds that strings that have the same count of each letter are anagrams (Method of doing this is unimportant)
+1: Returned Boolean value