CSE 101: Introduction to Computational and Algorithmic Thinking

Homework #1

Fall 2018

Assignment Due: September 28, 2018, by 11:59 pm

Assignment Objectives

By the end of this assignment you should be able to design, code, run and test original Python functions that solve programming problems involving if statements, strings, lists, and for loops.

Getting Started

The assignments in this course require you to write Python code to solve computational problems. To help you get started on each assignment, we will give you a "bare bones" file with a name like hw1.py. These files will contain one or more function stubs and a few tests you can try out to see if your code seems to be correct. Stubs are functions that have no bodies (or have very minimal bodies). You will fill in the bodies of these functions for the assignments. Do not, under any circumstance, change the names of the functions or their parameter lists. The automated grading system will be looking for exactly those functions provided in hw1.py. Please note that the test cases we provide you in the bare bones files will not necessarily be the same ones we use during grading!

Directions

This assignment is worth a total of 20 points. The automated grading system will execute your solution several times, using different input values each time. Each test that produces the correct/expected output will earn 4 points. You must earn at least 16 points in order to pass (receive credit for) this assignment.

- At the top of the hw1.py file, include the following information in comments, with each item on a separate line:
  - your full name AS IT APPEARS IN BLACKBOARD
  - your Stony Brook ID #
  - your Stony Brook NetID (your Blackboard username)
  - the course number (CSE 101)
  - the assignment name and number (Assignment #1)

⚠ Any functions that we tell you to complete (i.e., that we provide stubs for in the starter code) MUST use the names and parameter lists indicated in the starter code file. Submissions that have the wrong function names (or whose functions contain the wrong number of parameters) can't be graded by our automated grading system.

⚠ Be sure to submit your final work as directed by the indicated due date and time. Late work will not be accepted for grading. Work is late if it is submitted after the due date and time.

⚠ Programs that crash will likely receive a grade of zero, so make sure you thoroughly test your work before submitting it.

Lagged Fibonacci Digits

The Fibonacci sequence is a sequence of integer values, beginning with 0 and 1, where each term (after the first two) is equal to the sum of the two preceding terms. For example, the first few terms of the Fibonacci sequence are 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, etc.

This type of behavior can be used to develop a type of pseudorandom number generator called an Additive Lagged Fibonacci Generator (used in things like the Soviet VIC cipher from the 1950s). The process described below is often called "chain addition".

Each number or term in the sequence is a single digit. To generate the next term in the sequence, we add two adjacent digits from the sequence, modulo 10, and append the new digit to the end of the sequence. We always begin with the first two digits (elements) of the starting sequence.

Example:

1. Consider the starting list

   $$7, 2, 9, 1, 6, 7$$

   The first two digits are 7 and 2. 7 + 2 modulo 10 is 9, so we append 9 to the end of the list to get

   $$7, 2, 9, 1, 6, 7, 9$$

2. Next, we examine the second and third digits. 2 + 9 is 11. 11 modulo 10 is 1, so we append 1 to the end of the list to get

   $$7, 2, 9, 1, 6, 7, 9, 1$$

3. For the third step, we examine the third and fourth digits. 9 + 1 is 10; 10 modulo 10 is 0, so we append 0 to our list to get

   $$7, 2, 9, 1, 6, 7, 9, 1, 0$$

4. Subsequent terms (digits) in the sequence follow the same general pattern.

Complete the alfg() function, which takes two arguments: a list containing an initial integer sequence with at least 2 digits, and a positive integer representing the number of rounds of chained addition to perform. The function returns a new list: either the last five digits of the sequence after all of the requested rounds are complete, or the entire list if it has fewer than five terms (digits).

Examples:

Function Call	Return Value
alfg([5,0,2], 1)	[5,0,2,5]
alfg([1,2,3,4], 5)	[3,5,7,7,8]
alfg([2,6,5,3,9,1], 20)	[1,7,6,7,1]
alfg([6,7,3,4,5], 40)	[3,9,4,7,5]