Assignment 1

B351 / Q351

Due: August 27, 2017 at 11:59 PM

This assignment is intended to bring the class to a baseline of proficiency with git, Python tools, syntax, and built in data structures.

We will be using Python 3 so be sure you aren't using older versions. Code that will compile in Python 2.7 may not compile in Python 3. See our installation guide for help removing old versions and installing Python 3.

Please submit your completed files to your private Github repository for this class. You may not make further revisions to your files beyond the above deadline without incurring a late penalty. No collaboration is allowed on this assignment.

1 Summary

- Familiarize self with the course tools and methodologies
- Refresh or learn basic Python syntax

2 Background

Before beginning to work on this assignment, please be sure that you have thoroughly read and understood the following files on the Canvas webpage:

- Git Installation and Configuration
- Python Installation and Configuration
- Assignment Procedures

3 Programming Component

3.1 Problem 1 (10%)

The Fibonacci sequence is a sequence of numbers, where each number is the sum of the previous two numbers. the mathematical definition is as follows:

$$F_0 = 0, F_1 = 1, F_n = F_{n-1} + F_{n-2}$$

Objective: Complete the `fib(n)` function to compute the nth fibonacci number using recursion.

3.2 Problem 2 (10%)

The sum of the numbers from 0 to n is as follows:

$$F_0 = 0, F_n = n + F_{n-1}$$

Objective: Complete the `sum(n)` function to use a loop to compute the sum of the natural numbers from 0 to n.

3.3 Problem 3 (20%)

A matrix looks like this:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

When you *transpose* a matrix, you make row n of the old matrix into column n of the new matrix. Here's the transpose of the previous matrix.

$$\begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$$

Objective: Complete the `transpose(matrix)` function so that it returns the transpose of the input matrix.

3.4 Problem 4 (20%)

Euclidean Distance measures the distance of a straight line between two points. It is defined as follows:

$$\text{euclidean}(p, q) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$

Objective: Complete the `euclidean(p1,p2)` function so that it returns the euclidean distance between two points (note that these points can be of any dimension ≥ 2).

3.5 Problem 5 (20%)

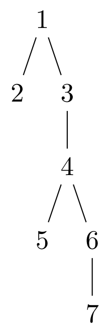
Objects in Python are useful for many things. One common reason you might write a Python class is to build a custom data structure.
For example, the following class defines a tree.

```
# A Node is an object
# - value : Number
# - children : List of Nodes
class Node:
    def __init__(self, value, children):
        self.value = value
        self.children = children
```

Here's an example of creating a tree object:

```
exTree = Node(1,[
Node(2,[]),Node(3,[
Node(4,[Node(5,[]),Node(6,[
Node(7,[])])])])])
```

This is how `exTree` looks when drawn.



Objective:

1. Complete the `sumNodes(root)` function so it returns the sum of the values at each node in the tree without using recursion. For example, `sumNodes(exTree)` will return 28.

2. Complete the `sumNodesRec(root)` function so it returns the sum of the values at each node in the tree using recursion.

3.6 Problem 6 (20%)

There is another way besides 'def' in Python to define a function. Here's an example:

```
f = lambda x: x**2
```

Now we can use f as follows:

f(2) = 4
f(10) = 100

We can do some interesting things with lambdas, such as having functions take functions as arguments, or return new functions. Here's a function which returns a function:

```
# makePowerFunction takes a number, and returns a
# function which will raise its argument to that power
makePowerFunction = lambda x: lambda y: y**x
```

This function can now be used to make new functions like this:

```
square = makePowerFunction(2)
cube = makePowerFunction(3)
square(3)
> 9
cube(3)
> 27
```

Objective: Complete the function `compose(fo, fi)`, which takes an outer function, an inner function, and returns a function which applies the outer function to the inner function to an argument.

3.7 Bonus Problem (10%)

Objective: Suppose you are given a tree. Write a function `treeToString(root)` which constructs a string with the values at each level of the tree on a new line. For example, using exTree from problem 5, we would see the following:

```
print(treeToString(exTree))
> 1
> 23
> 4
> 56
> 7
```

4 Grading

Items will are weighted according to the percentage assigned in the problem title. For this assignment, all points will be given on an all-or-nothing basis according to whether your solutions pass the given test cases.

Finally, remember that this is an individual assignment. The objective of this assignment is to ensure that you have enough programming knowledge to be successful in the later portions of this course. If you find this assignment particularly difficult, you may want to reach out to the instructors to discuss the best plan of action.