CSE 015: Discrete Mathematics
Spring 2024
Lab Assignment #3

You have THREE LAB SESSIONS to complete this lab, the week of Apr. 8, the week of Apr. 15, and the week of Apr. 22.

YOUR SOLUTION MUST BE DEMONSTRATED TO YOUR TA DURING LAB NO LATER THAN THE LAB SESSIONS IN THE WEEK OF APR. 22:

02L M 8:30am-10:20am (TA Dhawal Modi): IN LAB MONDAY APR. 22

03L M 10:30am-12:20pm: (TA Dhawal Modi): IN LAB MONDAY APR. 22

04L F 4:30pm-6:20pm: (TA Dhawal Modi): IN LAB FRIDAY APR. 26

05L Tu 10:30am-1:00pm: (TA Shangye Chen): IN LAB TUESDAY APR. 23

06L Th 10:30am-1:00pm: (TA Shangye Chen): IN LAB THURSDAY APR. 25

07L W 7:30pm-8:30pm: (TA Shangye Chen): IN LAB WEDNESDAY APR. 24

08L F 8:30am-10:20am: (TA Xingjian Ding): IN LAB FRIDAY APR. 26

If you demonstrate your solution to your TA in the first or second lab session, you don't need to attend the subsequent lab sessions for this assignment.

You must demonstrate your solution to YOUR TA IN YOUR LAB SESSION. You cannot demonstrate to another TA or in a lab session other than your own.

Introduction

This lab is all about using recursion as a problem solving tool. All solutions you provide *must* be recursive. That is, your solutions cannot use loops (`for`, `while`, etc.) but, instead, your functions must call themselves with a simpler version of the problem to be solved.

First, download the file `recursion.py` from the folder associated with this lab on CatCourses. This file contains empty definitions of three functions: `factorial(n)`, `fib(n)`, and `addup(list)`. There are also test cases for each function. Your task is to complete each of the three functions buy adding code under the comment `# Provide your code here`.

Problem 1: Computing the factorial function recursively (10 pts.)

Complete the function `factorial(n)` that recursively computes the factorial of n: $n! = n \times \ldots \times 3 \times 2 \times 1$. Think about how this can be computed recursively by noting that $n! = n \times (n-1)!$. That is, $n!$ is equal to n times $(n-1)!$. Think about what your base case is, that is, when your function should return a value instead of calling itself.

To get you started on this lab, here is the solution to this first problem:

```
if n == 0:
    return 1
else:
    return n * factorial(n-1)
```

Problem 2: Computing the n^{th} Fibonacci number recursively (10 pts.)

The Fibonacci *sequence* is the series of numbers: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ... That is, the next number is the sum of the two numbers preceding it and the first two numbers are 0 and 1.

The n^{th} Fibonacci number F_n can be computed as the sum of F_{n-1} and F_{n-2}: $F_n = F_{n-1} + F_{n-2}$.

Complete the function `fib(n)` that recursively computes the n^{th} Fibonacci number where $n \geq 1$.

Think about how you can compute the n^{th} Fibonacci number recursively. Think about what the base cases are (note that there are actually two).

Assume that $n \geq 1$, that is, the first Fibonacci number is 0, the second is 1, the third is 1, the fourth is 2, etc. Specifically:

- `fib(1)` should return 0.
- `fib(2)` should return 1.
- `fib(3)` should return 1.
- `fib(4)` should return 2.
- `fib(5)` should return 3.
- `fib(6)` should return 5.
- `fib(7)` should return 8.
- `fib(8)` should return 13.

Problem 3: Adding up the items in a list recursively (10 pts.)

Complete the function `addup(list)` that recursively computes the sum of the items in `list`. You can assume the items are numbers.

Note that the sum of a list can be computed recursively as the first item plus the sum of the remaining items: sum = first_item + sum of remaining items.

To access the first item of a list in Python use: `list[0]`.

To access the remainder of the list use: `list[1:]`.

Think about what your base case should be (what should you return if your list is empty?). To determine the length of a list use: `len(list)`.