Lab 3 – Writing Recursive Code

(a.k.a. Let's Get Familiar with Recursion)

Release: 14 Sep 2020 (Mon, Week 6)
1 week to attempt
Due: 20 Sep 2020, 11pm (Sun)

Some Words:

Recursion is an interesting concept that will be more easily understood if you try it out in code. You will get a chance to write two recursive functions here. The "Quality Score" and "Time Taken" are not significant for this lab.

Instructions:

- There are 2 questions in this exercise to be completed individually.
- For this exercise, your team ID is your name (i.e. you are the only member in your team).
- You need to submit code for this exercise at the Submission Server. No written submission is required.
- Edit **lab3a.py and lab3b.py** that are given to you. You can submit your solutions to the Submission Server as many times as you wish, but the final submission on the deadline will be taken as your final submission.

Lab 3a

The problem here is quite simple: given an integer (e.g. 74251), find the sum of all the individual digits in that integer (e.g. 7 + 4 + 2 + 5 + 1 = 19). This problem can be solved using iteration:

```
def sum_of_digits_iterative(i):
  input = str(i) # store number as a string for manipulation
  sum = 0

  for i in range(len(input)):  # use loop to extract each digit
    # convert character at position i to an integer, and add to sum
    sum += int(input[i])
  return sum
```

However, you will try to solve it using recursion for this exercise.

You are given the following file(s) for this exercise:

File name	Description	Comments
lab3a.py	Contains the **sum_of_digits** function that you will write.	You need to modify and submit this file. This is the only file that you may submit. Do not modify the file name or the function signature in the file.

A main file is not provided; you may test your **sum_of_digits()** function in IDLE, or write a single statement to call it with test values.

Requirements:

- Write a function called **sum_of_digits** that takes in 1 argument:
  sum_of_digits(i)
  where **i** is a non-negative integer, and returns the sum of all the digits in **i** as an integer.
  Edit **lab3a.py** provided to meet the requirements.
- **sum_of_digits** must be a recursive function (i.e. it calls itself).
- Your function should complete within 1 minute on the Submission Server.

Hints:

- You can use the `%` operator to "extract" the rightmost digit in an integer. For example: `456 % 10` gives you `6`, and `8529 % 10` gives you `9`.
- You can use the `//` operator to "remove" the rightmost digit in an integer. For example: `456 // 10` gives you `45`, and `8529 // 10` gives you `852`.

Note:

- All recursive functions must have a properly-written base case so that recursion does not happen indefinitely. If you see an error like this:

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<stdin>", line 2, in sum_of_digits
  File "<stdin>", line 2, in sum_of_digits
  File "<stdin>", line 2, in sum_of_digits
  [Previous line repeated 995 more times]
RecursionError: maximum recursion depth exceeded
```

  It probably means that your function keeps calling itself indefinitely (Python stops when the same function is recursively called 995 times). You should check if your base case condition will ever be met. For example, the following function will cause this type of error:

```
def sum_of_digits(i):
  return sum_of_digits(i)
```

To submit:

- **lab3a.py** (at submission server). Edit the comments at the top of your Python file to indicate your name and section.

Assessment:

- This exercise is not graded but submission of a working answer is mandatory.
- The Quality Score on the Submission Server for a correct answer is always 1.0. For this exercise, you can ignore both the Quality Score and the Time Taken on the score board.

Lab 3b

We are used to the base-10 (or decimal) number system which goes: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10… There are other number systems that are useful in certain scenarios. The base-2 (or binary), base-8 (or octal) and base-16 (hexadecimal) number systems are useful in certain domains. For this lab, you will write a recursive function to convert a base-10 number to a base-2 number. If you are unfamiliar with how to convert a base-10 number to base-2, browse through these useful sites:

http://www.purplemath.com/modules/numbbase.htm
http://mathbits.com/MathBits/CompSci/Introduction/frombase10.htm

Python has a simple function called **bin()** that converts a base-10 number to its base-2 equivalent. You can try this in IDLE to convert 6_{10} to 110_2:

```
>>> bin(6)
0b110
```

The "0b" in front of "110" indicates that you are dealing with binary (i.e. it's 110_2 instead of 110_{10}). If you only want the binary digits, just remove the first 2 characters ("0b") like this:

```
>>> bin(6)[2:]
'110'
>>> bin(10)[2:]
'1010'
```

This lab requires you to write a recursive function that duplicates this functionality.

You are given the following file(s) for this exercise:

File name	Description	Comments
lab3b.py	Contains the **to_binary** function that you will write.	You need to modify and submit this file. This is the only file that you may submit. Do not modify the file name or the function signature in the file.

Requirements:

- Write a function called **to_binary** that takes in 1 argument:
  to_binary (d)
  where **d** is a non-negative integer, and returns the binary (base-2) equivalent of the integer as a string. There should not be leading zeros in your returned string (i.e. **to_binary(4)** should return "100" and not "0100".)
  Edit **lab3b.py** provided to meet the requirements.
- **to_binary** must be a recursive function.
- You cannot use Python's built-in binary converter (i.e. **bin()**) in your code.
- Your function should complete within 1 minute on the Submission Server.

To submit:

- **lab3b.py** (at submission server).

Assessment:

- Same as 3(a) above.

~ End