Recursive Functions - Lab

Introduction

Now that you've seen a little preview of recursive functions, it's time to give them a try!

Objectives

You will be able to:

- Understand and use the concept of a recursive function
- Understand scope in the context of recursive functions
- Understand and compare depth first versus breadth first searches

Fibonacci

The Fibonacci sequence starts off: 1,1,2,3,5,8,13,21,34,...

Each number is the sum of the two preceding. Write a recursive function that calculates the nth number of the Fibonacci sequence. For example, our sequence above would correspond to:

fib(1) = 1 #The 1st element in the sequence is 1
fib(2) = 1 #The 2nd element in the sequence is 1
fib(3) = 2 #The 3rd element in the sequence is 2
fib(4) = 3 #The 4th element in the sequence is 3
fib(5) = 5 #The 5th element in the sequence is 5
fib(6) = 8 #The 6th element in the sequence is 8
fib(7) = 13 #The 7th element in the sequence is 13
fib(8) = 21 #The 8th element in the sequence is 21
fib(9) = 34 #The 9th element in the sequence is 34

In [1]:

```
#Your code here
def fib(n):
    if n < 1:
        return "N must be an integer greater then 1"
    elif n in [1,2]:
            return 1
    else:
        return fib(n-1) + fib(n-2)
```

In [2]:

```
for i in range(1, 10):
    print(fib(i))
```

```
1
1
2
3
5
8
13
21
34
```

Flat List

Write a function that takes a nested list and flattens it to a list of ints, floats and strings. For example the nested list [1, [2,3[4,5,6]], 7, [8], [9,10]] would become [1,2,3,4,5,6,7,8,9,10] or [1,2[3,4,[5]]] would become [1,2,3,4,5].

In [3]:

```
def flat_list(L, result=[]):
    print('Current L:', L) #Optional, to display process
    for i in L:
        if type(i) == list:
            flat_list(i, result)
        else:
            result.append(i)
    return result
L = [1,[2,3,[4,5,6]], 7, [8], [9,10]]
flat_list(L)
```

```
Current L: [1, [2, 3, [4, 5, 6]], 7, [8], [9, 10]]
Current L: [1, [2, 3, [4, 5, 6]], 7, [8], [9, 10]]
Current L: [2, 3, [4, 5, 6]]
Current L: [4, 5, 6]
Current L: [8]
Current L: [9, 10]
```

Out[3]:
```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

Depth vs Breadth First Search

Did you use breadth or depth first recursive calls above? Explain.

Answers may vary. General explanation below.

Depth first search navigates down a nested data structure, while breadth first goes one layer at a time, successively deeper but processing an entire layer before moving on to deeper layers.

Summary

Well done! Recursive functions are an advanced topic in Python and you got some good practice tackling classic problems here.