Recursion Homework Problems - Solutions

This assignment is a variety of small problems to begin you getting used to the idea of recursion. They are not full-blown interview questions, but do serve as a great start for getting your mind "in the zone" for recursion problems.

Here are the solutions with some simple explanations to the problems.

Problem 1

Write a recursive function which takes an integer and computes the cumulative sum of 0 to that integer

For example, if n=4 , return 4+3+2+1+0, which is 10.

This problem is very similar to the factorial problem presented during the introduction to recursion. Remember, always think of what the base case will look like. In this case, we have a base case of n =0 (Note, you could have also designed the cut off to be 1).

In this case, we have: n + (n-1) + (n-2) + ... + 0

Check out a sample solution:

In [1]:
```
def rec_sum(n):

    # Base Case
    if n == 0:
        return 0

    # Recursion
    else:
        return n + rec_sum(n-1)
```

In [2]:
```
rec_sum(4)
```

Out[2]:
```
10
```

Problem 2

Given an integer, create a function which returns the sum of all the individual digits in that integer. For example: if n = 4321, return 4+3+2+1

In [3]:
```
def sum_func(n):
    # Base case
    if len(str(n)) == 1:
        return n

    # Recursion
```

```
    else:
        return n%10 + sum_func(n//10)
```

In [4]:
```
sum_func(4321)
```

Out[4]:
```
10
```

Hints:

In [1]:
```
# You'll neeed to use modulo
4321%10
```

Out[1]:
```
1
```

In [2]:
```
4321 // 10
```

Out[2]:
```
432
```

We'll need to think of this function recursively by knowing that: 4502 % 10 + sum_func(4502//10)

Hint2: python3 automatically converts the result of a division to float! Make sure to use integer division (a//b)

Problem 3

Note, this is a more advanced problem than the previous two! It aso has a lot of variation possibilities and we're ignoring strict requirements here.

Create a function called word_split() which takes in a string **phrase** and a set **list_of_words**. The function will then determine if it is possible to split the string in a way in which words can be made from the list of words. You can assume the phrase will only contain words found in the dictionary if it is completely splittable.

For example:

In [6]:
```
word_split('themanran',['the','ran','man'])
```

Out[6]:
```
['the', 'man', 'ran']
```

In [7]:
```
word_split('ilovedogsJohn',['i','am','a','dogs','lover','love','John'])
```

Out[7]:
```
['i', 'love', 'dogs', 'John']
```

In [8]:
```
word_split('themanran',['clown','ran','man'])
```

Out[8]:
```
[]
```

In [5]:
```
def word_split(phrase,list_of_words, output = None):
    '''
    Note: This is a very "python-y" solution.
    '''

    # Checks to see if any output has been initiated.
```

```
    # If you default output=[], it would be overwritten for every recursion!
    if output is None:
        output = []

    # For every word in list
    for word in list_of_words:

        # If the current phrase begins with the word, we have a split point!
        if phrase.startswith(word):

            # Add the word to the output
            output.append(word)

            # Recursively call the split function on the remaining portion of the phrase--- p
            # Remember to pass along the output and list of words
            return word_split(phrase[len(word):],list_of_words,output)

    # Finally return output if no phrase.startswith(word) returns True
    return output
```

Conclusion

Alright, so now that we've seen a few examples, let's dive in to the interview practice problems!