String-Permutations

May 18, 2020

0.0.1 Problem Statement

Given an input string, return all permutations of the string in an array.
Example 1: * `string = 'ab'` * `output = ['ab', 'ba']`
Example 2: * `string = 'abc'` * `output = ['abc', 'bac', 'bca', 'acb', 'cab', 'cba']` ---

Note - Strings are Immutable Strings in Python are immutable, whch means that we cannot overwrite the characters of the String objects. For example:

```
str1 = "Hello"
str1[0] = 'K'                          # Try changing the first character
```

will lead to

```
TypeError: 'str' object does not support item assignment
```

We can only re-assign the variable to a new value (string), as follows:

```
str1 = "Udacity"                       # re-assignment
str2 = "Welcome to the " + str1[3:]    # Returns 'Welcome to the city'
```

Therefore, we do not require a deep copy in this exercise, as it was the case in our last example of list permutation.

The Idea Starting with a blank list, add each character of original input string at all possible positions.
For example, take `"abc"` as the original string:

1. Start with a blank `list()` object. This is actually the last call of recursive function stack. Pick a character `'c'` of original string, making the output as `['c']`
2. Pick next character `b` of original input string, and place the current character at different indices of the each sub-string of previous output. **We can make use of the sub-string of previous output, to create a new sub-string.** Now, the output will become `['bc', 'cb']`.
3.

0.1 Pick next character a of original input string, and place the current character at different indices of the each sub-string of previous output. Now, the output will become ['abc', 'bac', 'bca', 'acb', 'cab', 'cba']..

0.1.1 Exercise - Write the function definition here

```
In [ ]: def permutations(string):
            """
            :param: input string
            Return - list of all permutations of the input string
            TODO: complete this function to return a list of all permutations of the string
            """
            pass
```

Show Solution

0.1.2 Test - Let's test your function

```
In [ ]: def test_function(test_case):
            string = test_case[0]
            solution = test_case[1]
            output = permutations(string)

            output.sort()
            solution.sort()

            if output == solution:
                print("Pass")
            else:
                print("Fail")
```

```
In [ ]: string = 'ab'
        solution = ['ab', 'ba']
        test_case = [string, solution]
        test_function(test_case)
```

```
In [ ]: string = 'abc'
        output = ['abc', 'bac', 'bca', 'acb', 'cab', 'cba']
        test_case = [string, output]
        test_function(test_case)
```

```
In [ ]: string = 'abcd'
        output = ['abcd', 'bacd', 'bcad', 'bcda', 'acbd', 'cabd', 'cbad', 'cbda', 'acdb', 'cadb'
        test_case = [string, output]
        test_function(test_case)
```