4_recursive_functions

March 6, 2019

1 Recurison

We know that in Python, a function can call other functions. It is even possible for the function to call itself. These type of construct are termed as recursive functions.

2 Example:

```
In [4]: #python program to print factorial of a number using recurion

        def factorial(num):
            """
            This is a recursive function to find the factorial of a given number
            """
            return 1 if num == 1 else (num * factorial(num-1))

        num = 5
        print ("Factorial of {0} is {1} ".format(num, factorial(num)))
```

```
Factorial of 5 is 120
```

3 Advantages

1. Recursive functions make the code look clean and elegant.
2. A complex task can be broken down into simpler sub-problems using recursion.
3. Sequence generation is easier with recursion than using some nested iteration.

4 Disadvantages

1. Sometimes the logic behind recursion is hard to follow through.
2. Recursive calls are expensive (inefficient) as they take up a lot of memory and time.
3. Recursive functions are hard to debug.

5 Python program to display the fibonacci sequence up to n-th term using recursive function

```
In [0]: def fibonacci(num):
            """
            Recursive function to print fibonacci sequence
            """
            return num if num <= 1 else fibonacci(num-1) + fibonacci(num-2)

        nterms = 10
        print("Fibonacci sequence")
        for num in range(nterms):
            print(fibonacci(num))
```

```
Fibonacci sequence
0
1
1
2
3
5
8
13
21
34
```

```
In [0]:
```