Recursion in Python

A function that calls itself is a recursive function. This method is used when a certain problem is defined in terms of itself. Although this involves iteration, using an iterative approach to solve such a problem can be tedious. The recursive approach provides a very concise solution to a seemingly complex problem. It looks glamorous but can be difficult to comprehend!

The most popular example of recursion is the calculation of the factorial. Mathematically the factorial is defined as: n! = n * (n-1)!

We use the factorial itself to define the factorial. Hence, this is a suitable case to write a recursive function. Let us expand the above definition for the calculation of the factorial value of 5.

```
5! = 5 X 4!
     5 X4 X 3!
     5 X4 X 3 X 2!
     5 X4 X 3 X  2 X 1!
     5 X4 X 3 X  2 X 1
   = 120
```

While we can perform this calculation using a loop, its recursive function involves successively calling it by decrementing the number until it reaches 1. The following is a recursive function to calculate the factorial.

Example: Recursive Function

```
def factorial(n):
    if n == 1:
        print(n)
        return 1
    else:
        print (n,'*', end=' ')
        return n * factorial(n-1)
```

The above recursive function can be called as below.

```
>>> factorial(5)
5 * 4 * 3 * 2 * 1
120
```

When the factorial function is called with 5 as argument, successive calls to the same function are placed, while reducing the value of 5. Functions start returning to their earlier call after the argument reaches 1. The return value of the first call is a cumulative product of the return values of all calls.