Python Tutorial

Related Examples

Python Recursion

Recursion is the process of defining something in terms of itself. A physical world example would be to place two parallel mirrors facing each other. Any object in between them would be reflected recursively.

Python Recursive Function

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.

Following is an example of recursive function to find the factorial of an integer. Factorial of a number is the product of all the integers from 1 to that number. For example, the factorial of 6 (denoted as 6!) is 1*2*3*4*5*6 = 720.

Example of recursive function

```
# An example of a recursive function to
# find the factorial of a number

def recur_fact(x):
    """This is a recursive function
    to find the factorial of an integer"""

    if x == 1:
         return 1
    else:
         return (x * recur_fact(x-1))


num = int(input("Enter a number: "))
if num >= 1:
    print("The factorial of", num, "is", recur_fact(num))
```

Output

```
Enter a number: 4
The factorial of 4 is 24
```

Explanation

In the above example, recur_fact() is a recursive functions as it calls itself. When we call this function with a positive integer, it will recursively call itself by decreasing the number. Each function call multiples the number with the factorial of number-1 until the number is equal to one. This recursive call can be explained in the following steps.

```
recur_fact(4)                 # 1st call with 4
4 * recur_fact(3)             # 2nd call with 3
4 * 3 * recur_fact(2)         # 3rd call with 2
4 * 3 * 2 * recur_fact(1)     # 4th call with 1
4 * 3 * 2 * 1                 # retrun from 4th call as number=1
4 * 3 * 2                     # return from 3rd call
```

```
4 * 6                                  # return from 2nd call
24                                     # return from 1st call
```

Our recursion ends when the number reduces to 1. This is called the base condition. Every recursive function must have a base condition that stops the recursion or else the function calls itself infinitely. We must avoid infinite recursion.

Advantages of recursion

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.

Disadvantages of recursion

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.

◄ Previous Page Next Page ►