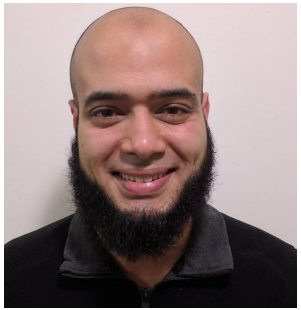
Python Programming

Recursive Functions 2

Mostafa S. Ibrahim
Teaching, Training and Coaching since more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)



Factorial: A recursive function

- A recursive function: Function that **calls itself** with smaller input (sub-problem) till calls reach a base case

```
2   def factorial(n):
3       print("Function Call: factorial: n=", n)
4
5       if n == 1:      # base case
6           return 1
7       return factorial(n-1) * n
8
9
10  print(factorial(6))
11
```

```
Function Call: factorial: n=6
Function Call: factorial: n=5
Function Call: factorial: n=4
Function Call: factorial: n=3
Function Call: factorial: n=2
Function Call: factorial: n=1
720
```

Let’s trace it

```
def factorial(n):
    print("Function Call: factorial: n=", n)

    if n == 1:      # base case
        return 1
    return factorial(n-1) * n


print(factorial(6))
```

- Call **Factorial**(6)
  - If 6 == 1? False
  - Call **Factorial** (5) and multiply results with 6
    - If 5 == 1? False
    - Call **Factorial** (4) and multiply results with 5
      - If 4 == 1? False
      - Call **Factorial** (3) and multiply results with 4
        - If 3 == 1? False
        - Call **Factorial** (2) and multiply results with 3
          - If 2 == 1? False
          - Call **Factorial** (1) and multiply results with 2
            - If 1 == 1? True
              - Return 1

Let’s trace it

Main: factorial(6)

Let’s trace it

factorial(6)
    Return factorial(5) * 6

Main: factorial(6)

Let’s trace it

factorial(5)
    Return factorial(4) * 5

factorial(6)
    Return factorial(5) * 6

Main: factorial(6)

Let’s trace it

factorial(4)
    Return factorial(3) * 4

factorial(5)
    Return factorial(4) * 5

factorial(6)
    Return factorial(5) * 6

Main: factorial(6)

Let’s trace it

factorial(3)
    Return factorial(2) * 3

factorial(4)
    Return factorial(3) * 4

factorial(5)
    Return factorial(4) * 5

factorial(6)
    Return factorial(5) * 6

Main: factorial(6)

Let’s trace it

factorial(3)
Return factorial(2) * 3

factorial(4)
Return factorial(3) * 4

factorial(5)
Return factorial(4) * 5

factorial(6)
Return factorial(5) * 6

Main: factorial(6)

factorial(2)
Return factorial(1) * 2

Let’s trace it

factorial(3)
Return factorial(2) * 3

factorial(4)
Return factorial(3) * 4

factorial(5)
Return factorial(4) * 5

factorial(6)
Return factorial(5) * 6

Main: factorial(6)

factorial(1)
Return 1

factorial(2)
Return factorial(1) * 2

Let’s trace it

factorial(3)
Return factorial(2) * 3

factorial(4)
Return factorial(3) * 4

factorial(5)
Return factorial(4) * 5

factorial(6)
Return factorial(5) * 6

Main: factorial(6)

factorial(2)
Return 1 * 2 ⇒ 2

Let’s trace it

factorial(3)
    Return 2 * 3 ⇒ 6

factorial(4)
    Return factorial(3) * 4

factorial(5)
    Return factorial(4) * 5

factorial(6)
    Return factorial(5) * 6

Main: factorial(6)

Let’s trace it

factorial(4)
Return 6 * 4 ⇒ 24

factorial(5)
Return factorial(4) * 5

factorial(6)
Return factorial(5) * 6

Main: factorial(6)

Let’s trace it

factorial(5)
Return 24 * 5 ⇒ 120

factorial(6)
Return factorial(5) * 6

Main: factorial(6)

Let’s trace it

factorial(6)
Return 120 * 6 ⇒ 720

Main: factorial(6)

Let’s trace it

Main: factorial(6) ⇒ 720

What did program print?

```
2  def factorial(n):
3      print("Function Call: factorial: n=", n)
4
5      return factorial(n-1) * n
6
7
8  print(factorial(6))
```

```
Function Call: factorial: n= 6
Function Call: factorial: n= 5
Function Call: factorial: n= 4
Function Call: factorial: n= 3
Function Call: factorial: n= 2
Function Call: factorial: n= 1
Function Call: factorial: n= 0
Function Call: factorial: n= -1
Function Call: factorial: n= -2
Function Call: factorial: n= -3
Function Call: factorial: n= -4
Function Call: factorial: n= -5
Function Call: factorial: n= -6
Function Call: factorial: n= -7
Function Call: factorial: n= -8
Function Call: factorial: n= -9
Function Call: factorial: n= -10
```

What did program print?

```
Function Call: factorial: n= -984
Function Call: factorial: n= -985
Function Call: factorial: n= -986
Function Call: factorial: n= -987
Function Call: factorial: n= -988
Function Call: factorial: n= -989
Traceback (most recent call last):
  File "/home/moustafa/00Udemy/CPP/private_gitlab_code/python_skills/26_recursion/
    print(factorial(6))
  File "/home/moustafa/00Udemy/CPP/private_gitlab_code/python_skills/26_recursion/
    return factorial(n-1) * n
  File "/home/moustafa/00Udemy/CPP/private_gitlab_code/python_skills/26_recursion/
    return factorial(n-1) * n
  File "/home/moustafa/00Udemy/CPP/private_gitlab_code/python_skills/26_recursion/
    return factorial(n-1) * n
  [Previous line repeated 993 more times]
  File "/home/moustafa/00Udemy/CPP/private_gitlab_code/python_skills/26_recursion/
    print("Function Call: factorial: n=", n)
RecursionError: maximum recursion depth exceeded while calling a Python object
```

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."