

PythonTip 03 - Recursion

February 9, 2026

1 Recursion

A recursive algorithm is an algorithm that calls itself. You need a base case so you don't get stuck in an infinite loop.

Example: suppose we want to calculate the quantity $n! = n(n-1)(n-2)\cdots 3\cdot 2\cdot 1$.

$$5! = 5 * 4 * 3 * 2 * 1 = 120$$

We'll use the fact that $n! = n \cdot (n-1)!$.

```
[2]: # What's wrong with this function?  
def factorial(n):  
    return n * factorial(n-1)
```

```
[3]: factorial(3)
```

```
-----  
RecursionError                                Traceback (most recent call last)  
Cell In[3], line 1  
----> 1 factorial(3)  
  
Cell In[2], line 3, in factorial(n)  
      2 def factorial(n):  
----> 3     return n * factorial(n-1)  
  
Cell In[2], line 3, in factorial(n)  
      2 def factorial(n):  
----> 3     return n * factorial(n-1)  
  
[... skipping similar frames: factorial at line 3 (2975 times)]  
  
Cell In[2], line 3, in factorial(n)  
      2 def factorial(n):  
----> 3     return n * factorial(n-1)  
  
RecursionError: maximum recursion depth exceeded
```

What calls are happening?

```
[ ]: # What's wrong with this function?
def factorial(n):
    return n * factorial(n-1)
```

```
factorial(3)
-> 3 * factorial(2)
-> 3 * 2 * factorial(1)
-> 3 * 2 * 1 * factorial(0)
-> 3 * 2 * 1 * 0 * factorial(-1)
-> infinite recursion
```

```
[11]: # We need a base case!
def factorial(n):
    assert n >= 0 and isinstance(n, int), "invalid input into factorial"
    if n == 1:
        # factorial(1) = 1
        return 1
    return n * factorial(n-1)
```

```
[8]: factorial(5)
```

```
[8]: 120
```

```
[9]: factorial(3.5)
```

```
-----
AssertionError                                Traceback (most recent call last)
Cell In[9], line 1
----> 1 factorial(3.5)

Cell In[7], line 3, in factorial(n)
      2 def factorial(n):
----> 3     assert n >= 0 and isinstance(n, int), "invalid input into factorial"
      4     if n == 1:
      5         # factorial(1) = 1
      6         return 1

AssertionError: invalid input into factorial
```

```
[ ]: factorial(5)
```

```
factorial(5)
5 * factorial(4)
5 * (4 * factorial(3))
5 * (4 * (3 * factorial(2)))
```

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

```
[ ]: factorial(-1)
```

```
[ ]: factorial(0)
```

```
[ ]:
```