

Lecture 34 | Recursion

Example: Computing factorial of a number n .

$$n! = n \cdot (n-1) \cdot (n-2) \cdot (n-3) \cdot \dots \cdot 1$$

n must be an $\text{int} \geq 1$.

Assume $0! = 1$

$$4! = 4 \cdot 3 \cdot 2 \cdot 1 = 24$$

$$3! = 3 \cdot 2 \cdot 1 = 6$$

$$\text{factorial}(4) = 4 \cdot \text{factorial}(3)$$

$$4! = 4 \cdot 3 \cdot 2 \cdot 1 = 4 \cdot (3 \cdot 2 \cdot 1)$$

$$\text{factorial}(3) = 3 \cdot \text{factorial}(2)$$
$$3! = 3 \cdot 2 \cdot 1 = 3 \cdot (2 \cdot 1)$$

$$\text{factorial}(2) = 2 \cdot \text{factorial}(1)$$
$$2! = 2 \cdot 1 = 2 \cdot (1)$$

$$\text{factorial}(1) = 1 \cdot \text{factorial}(0)$$
$$1! = 1 \cdot 1 = 1 \cdot (1)$$

defined as 1

$0! = \text{factorial}(0)$
= base case where
problem boils down to

one known number.

[1]

Factorial symbol table

def factorial(n):

if n == 0:

return 1

else:

return n * factorial(n-1)

print(factorial(3))

main

factorial fun

factorial X

Name	Value
n	3

factorial

Name	Value
n	2

factorial

Name	Value
n	1

factorial

Name	Value
n	0

return 6

3 * 2 = 6
3 * factorial(2)

return 2

2 * 1 = 2
2 * factorial(1)

return 1

1 * 1 = 1
1 * factorial(0)

return 1

then symbol table goes away

After we return a value, remember that that symbol table goes away.

base case where recursion stops

main

factorial fun

```
def factorial(n):
```

```
    if n == 0:
```

```
        ans = 1
```

```
    else:
```

```
        ans = n * factorial(n-1)
```

```
    return ans
```

factorial

<u>Name</u>	<u>Value</u>
n	3

factorial

<u>Name</u>	<u>Value</u>
n	2
ans	2

factorial

<u>Name</u>	<u>Value</u>
n	1
ans	1

factorial

<u>Name</u>	<u>Value</u>
n	0
ans	1

return ans

ans = 2 * 1 = 2

2 * factorial(1)

return 1

ans = 1 * factorial(0)

return ans