

Fatorial

```
fatorial(5)
 5 * fatorial(5 - 1)
   4 * fatorial(4 - 1)
    3 * fatorial(3 - 1)
     2 * fatorial(2 - 1)
      1 * fatorial(1 - 1)
       1
```

Fonte:

<https://olamundo0.wordpress.com/2010/04/20/recursividade/>

Fatorial recursivo

$$n! = \begin{cases} 1, & \text{quando } n = 0, \\ n \times (n - 1)!, & \text{quando } n > 0. \end{cases}$$

```
def fatorial(n)
    '''(int) -> int
    Recebe um inteiro n e retorna n!
    '''
    if n == 0:
        return 1
    return n * fatorial(n-1)
```

fatorial(10)

```
fatorial(10)
  fatorial(9)
    fatorial(8)
      fatorial(7)
        fatorial(6)
          fatorial(5)
            fatorial(4)
              fatorial(3)
                fatorial(2)
                  fatorial(1)
                    fatorial(0)
```

fatorial de 10 e' 3628800.

Diagramas de execução

fatorial(3)

n

3

fatorial(2)

n

2

fatorial(1)

n

1

fatorial(1)

n

0

return 1

return n * fatorial(0) = 1 * 1

return n * fatorial(1) = 2 * 1 = 2

return n * fatorial(2) = 3 * 2 = 6

```
hanoi(2,'A','B','C')
```

```
hanoi(1,'A','C','B')
```

```
hanoi(0,'A','B','C')
```

1: mova o disco 1 do pino A para o pino B.

```
hanoi(0,'B','A','B')
```

2: mova o disco 2 do pino A para o pino C.

```
hanoi(1,'B','A','C')
```

```
hanoi(0,'B','C','A')
```

3: mova o disco 1 do pino B para o pino C.

```
hanoi(0,'A','B','C')
```

Fatorial iterativo

```
def factorial(n):
    '''(int) -> int
    Recebe um inteiro n e retorna n!
    '''
    ifat = 1
    for i in range(2,n+1): # /*1*/
        ifat *= i

    return ifat
```

Em /*1*/ vale que **ifat** == (**i**-1) !