Recursividade

SSC0301

Prof. Márcio Delamaro

ICMC USP SÃO CARLOS SSC

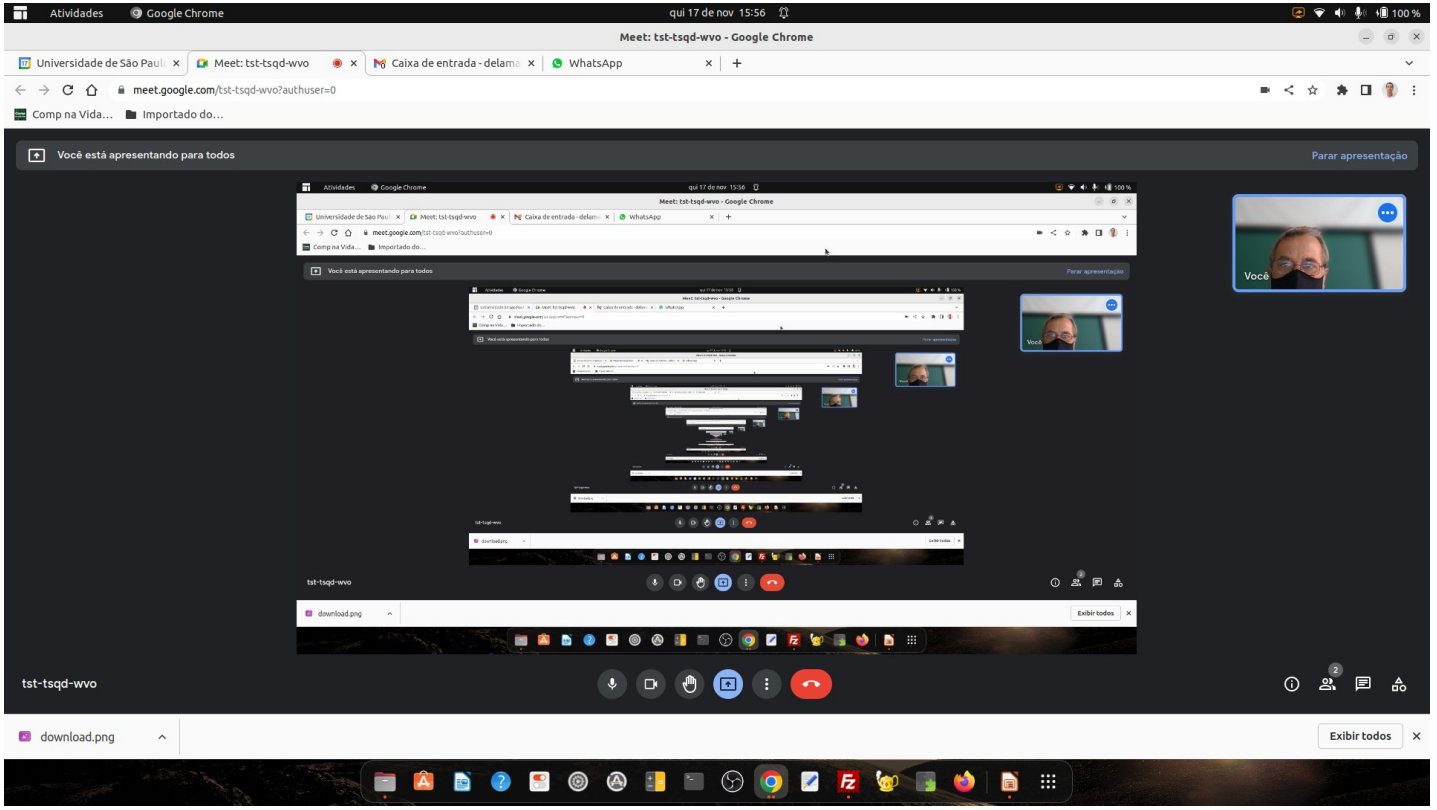
Recursividade

- Vimos que uma função pode chamar outra
- Uma função pode chamar ela mesma?
- Faz sentido?

Definição

- 'Recursividade' é um termo usado de maneira mais geral para descrever o processo de repetição de um objeto de um jeito similar ao que já fora mostrado. Um bom exemplo disso são as imagens repetidas que aparecem quando dois espelhos são apontados um para o outro.

Uma imagem...



ICMC USP SÃO CARLOS SSC

Definição

- 'Recursividade' é um termo usado de maneira mais geral para descrever o processo de repetição de um objeto de um jeito similar ao que já fora mostrado. Um bom exemplo disso são as imagens repetidas que aparecem quando dois espelhos são apontados um para o outro.
- Na matemática e na ciência da computação, a recursão especifica (ou constrói) uma classe de objetos ou métodos (ou um objeto de uma certa classe) definindo alguns poucos casos base ou métodos muito simples (frequentemente apenas um), e então definindo regras para formular casos complexos em termos de casos mais simples.

O que é?

- Ancestralidade de uma pessoa:
  - Os pais de uma pessoa são seus ancestrais (caso base);
  - Os pais de qualquer antepassado são também ancestrais da pessoa em consideração (passo recursivo).

- A definição formal dos números naturais diz que 0 (zero) é um número natural, e todo número natural tem um sucessor, que é também um número natural.

Fatorial

- Qual o valor de n! ?
  - a multiplicação de todos os inteiros de 1 até n
-
```
def fatorial(n):
    f = 1
    for k in range(1,n+1):
        f *= k
    return f
```

Ou...

Fatorial

- Qual o valor de n! ?
  - a multiplicação de todos os inteiros de 1 até n
  - n * (n-1)! . Além disso, 0! = 1
- 4! = 4 * 3! = 4 * 3 * 2! = 4 * 3 * 2 * 1! = 4 * 3 * 2 * 1 * 0! = 4 * 3 * 2 * 1 * 1 = 24
- Podemos usar essa definição para escrever uma função que computa o fatorial?

ICMC USP SÃO CARLOS SSC

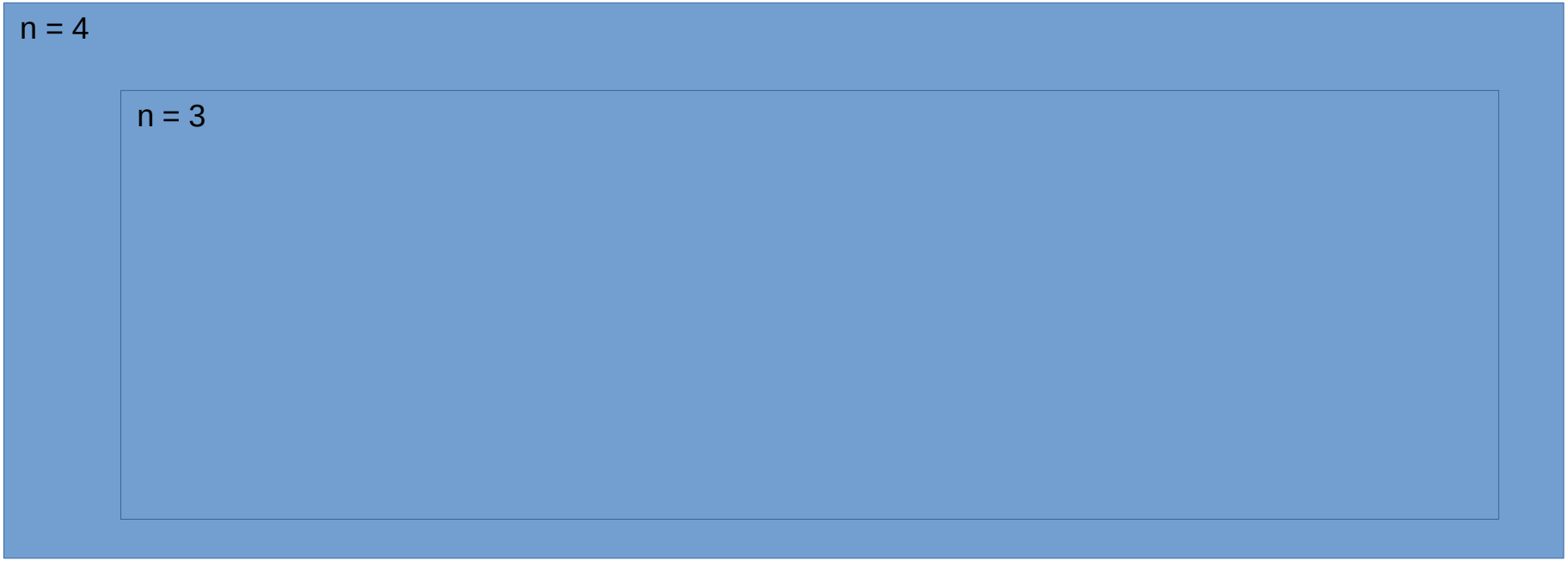
Fatorial

- Qual o valor de n! ?
  - a multiplicação de todos os inteiros de 1 até n
  - n * (n-1)! . Além disso, 0! = 1
- 4! = 4 * 3! = 4 * 3 * 2! = 4 * 3 * 2 * 1! = 4 * 3 * 2 * 1 * 0! = 4 * 3 * 2 * 1 * 1 = 24
-
```
def fatorial(n):
    if n == 0:
        return 1
    return n * fatorial(n-1)
```

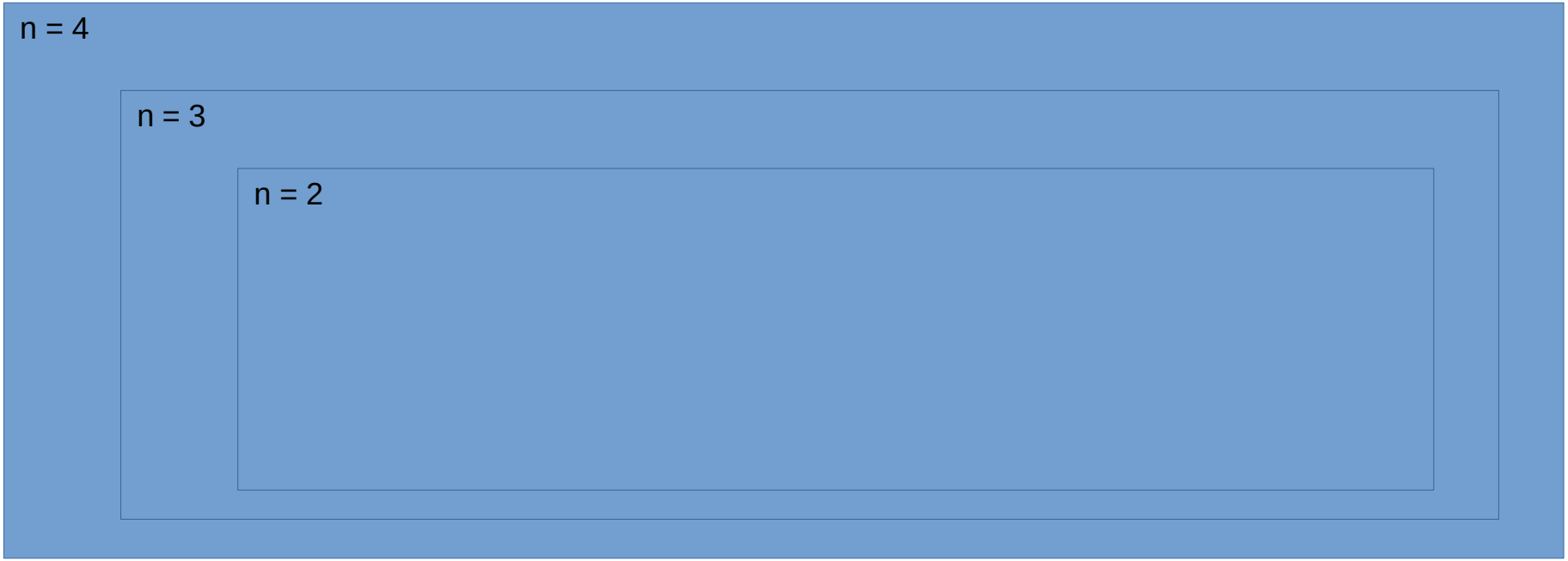
O que acontece?

fatorial(4)

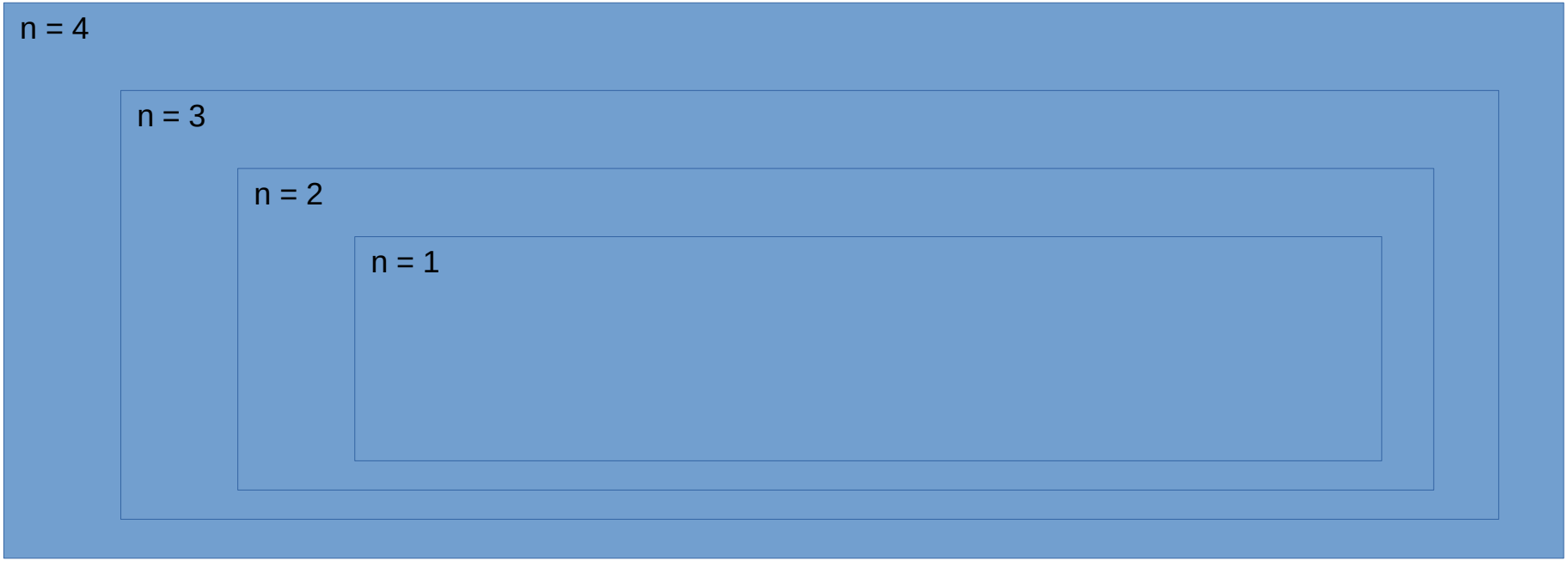
O que acontece?



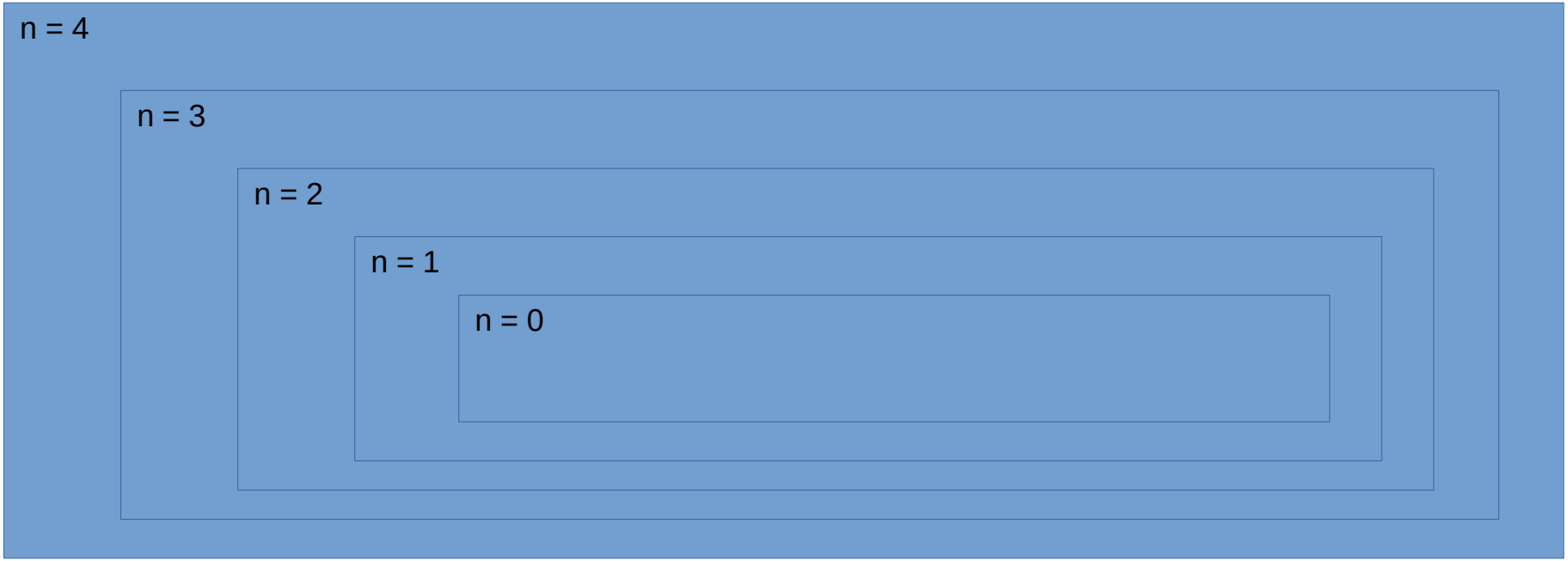
O que acontece?



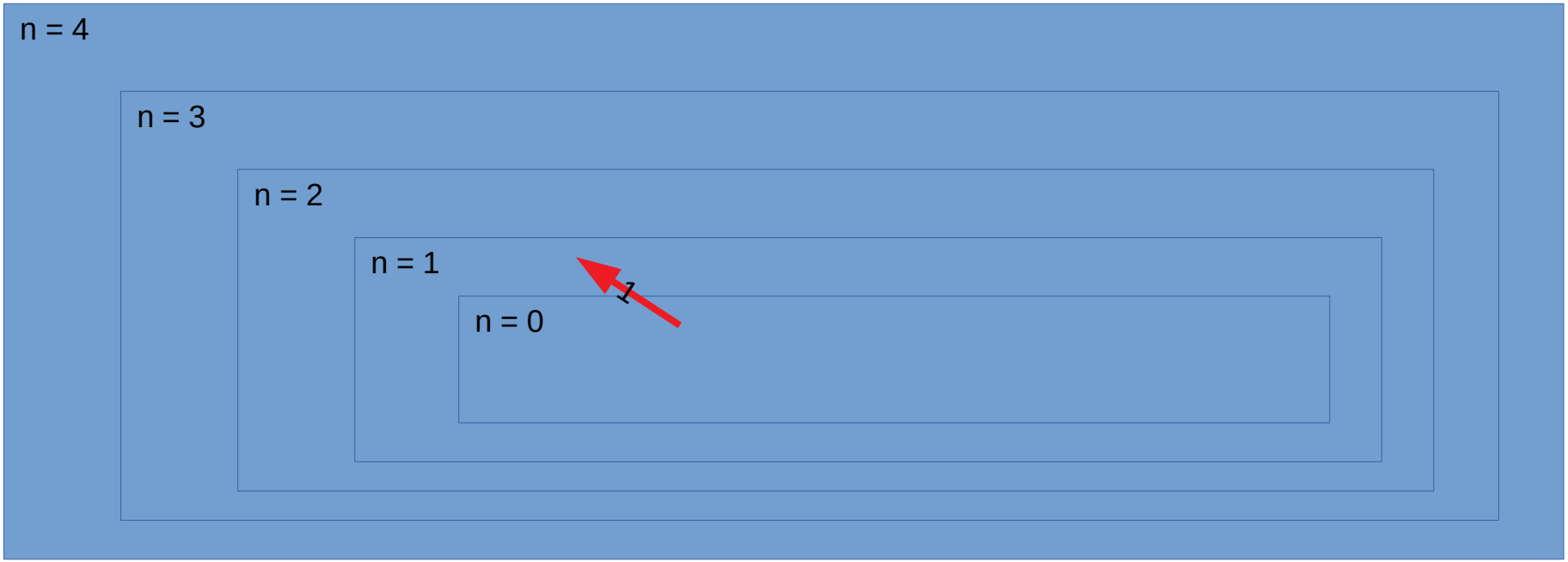
O que acontece?



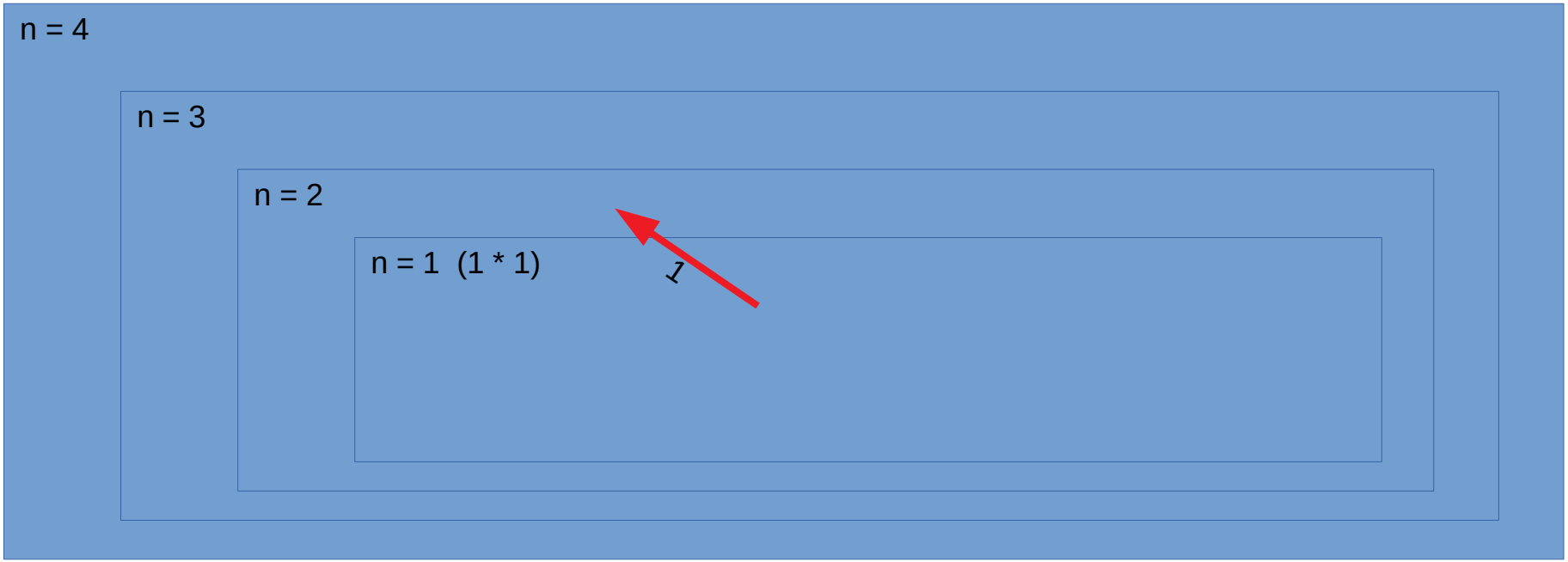
O que acontece?



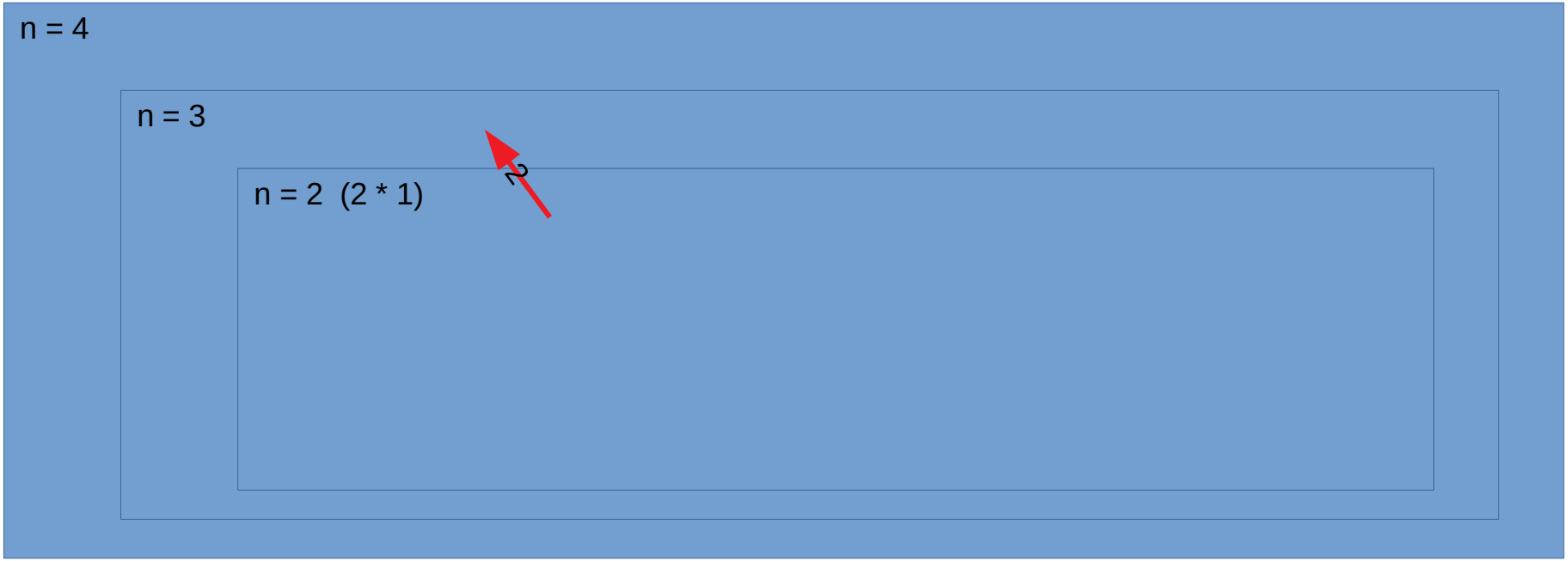
O que acontece?



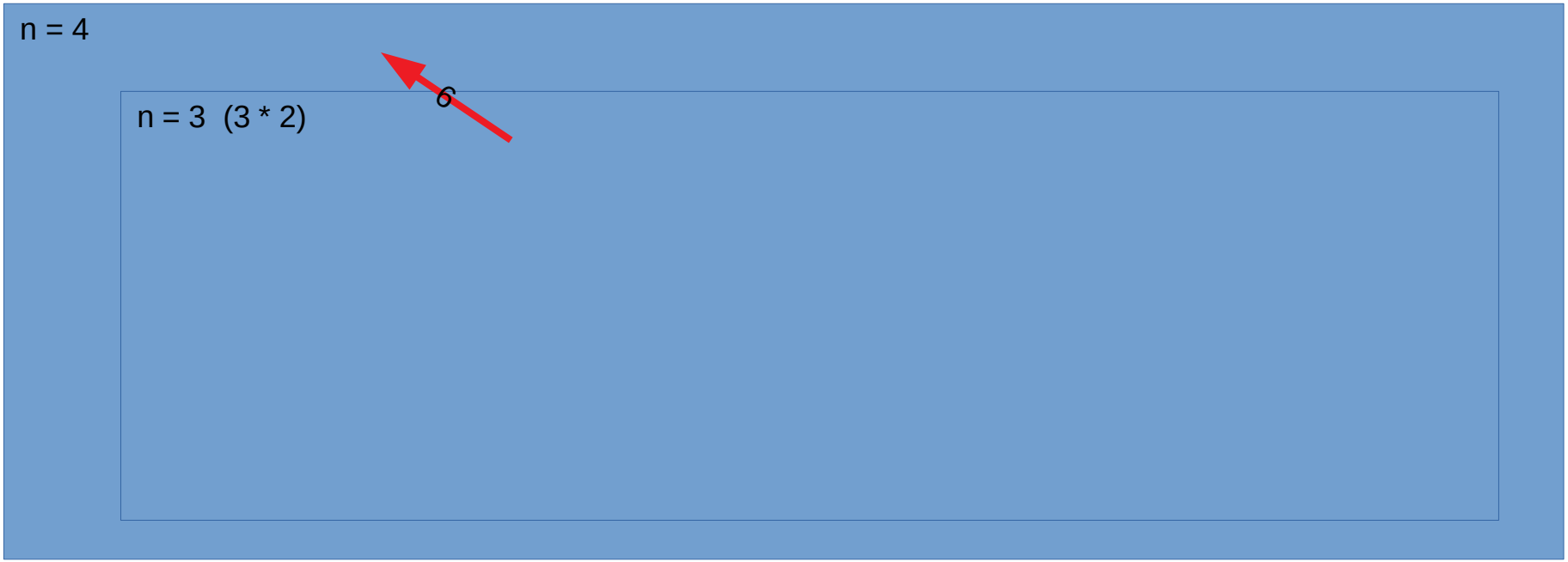
O que acontece?



O que acontece?

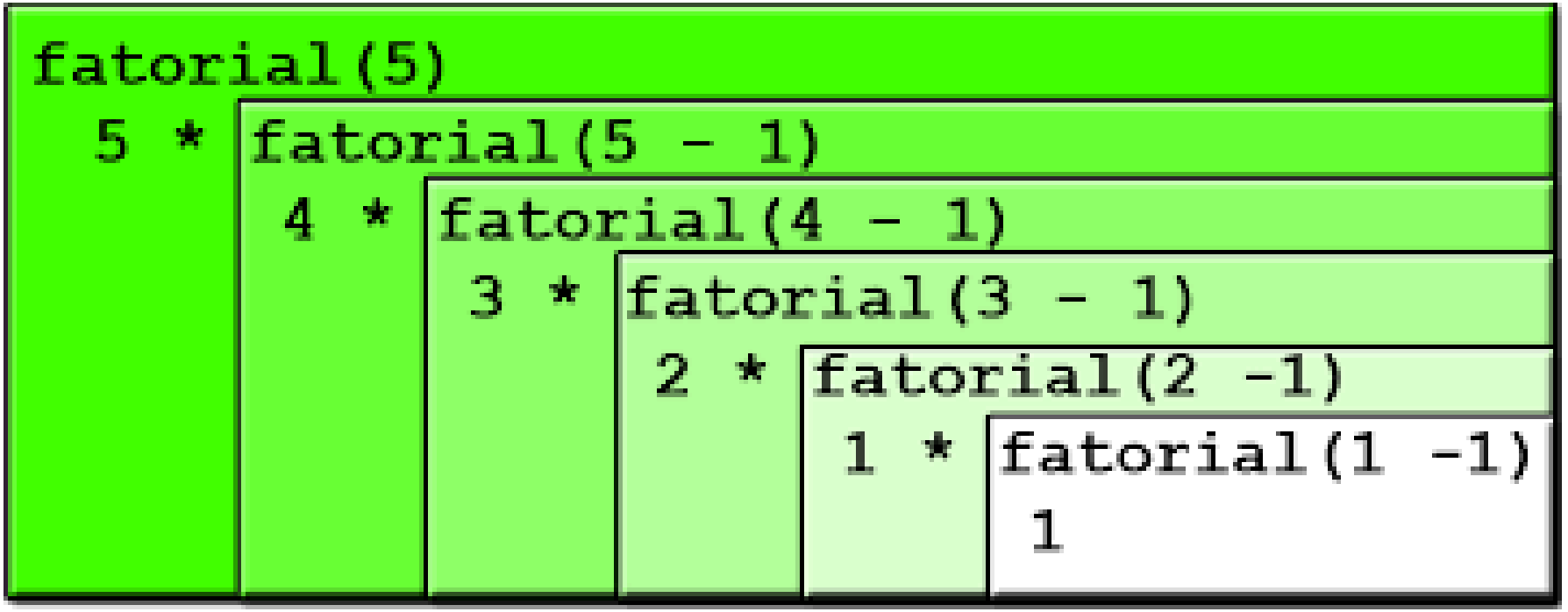


O que acontece?



O que acontece?

Resumo


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

Outro exemplo

- O máximo divisor comum (abreviadamente, MDC) entre dois ou mais números é o maior número que é fator de tais números. Por exemplo, os divisores comuns de 12 e 18 são 1,2,3 e 6, logo mdc(12 , 18) = 6.

mdc(1038, 366)

Outro exemplo

- O máximo divisor comum (abreviadamente, MDC) entre dois ou mais números é o maior número que é fator de tais números. Por exemplo, os divisores comuns de 12 e 18 são 1,2,3 e 6, logo mdc(12 , 18) = 6.

```
def mdc_obvio(a, b):
    if a < b:
        divisor = a
    else:
        divisor = b

    while a % divisor != 0 or b % divisor !=0:
        divisor = divisor - 1
    return divisor
```

MDC – Euclides

- Para calcular o mdc entre dois números **m** e **n**
  - mdc(m,0) → m;
  - mdc(m,n) → mdc(n, m % n), para n > 0.

MDC – Euclides

- Para calcular o mdc entre dois números **m** e **n**
  - mdc(m,0) → m;
  - mdc(m,n) → mdc(n, m % n), para n > 0.

```
def mdc_euclides(x,y):
    # print(x, y)
     if y == 0:
         return x

     return mdc_euclides(y,x % y)
```

Outra forma

- MDC(x, x) = x;
- MDC(x, y) = MDC(x − y, y), se x > y;
- MDC(x, y) = MDC(y, x).
- Como implementar?

Outro exemplo

- MDC(x, x) = x;
- MDC(x, y) = MDC(x − y, y), se x > y;
- MDC(x, y) = MDC(y, x).
- Como implementar?

MDC(76,76) = 76

MDC(76,20) = MDC(56,20)

MDC(20,76) = MDC(76,20)

MDC

```
def mdc(x,y):
    if x == y:
        return x
    if x > y:
        return mdc(x-y, y)
    return mdc(y,x)
```

Advertência

- Cada chamada de função, consome memória
- O interpretador vai limitar o número de chamadas

Advertência

```
> python mdc.py
Entre com o 1o. valor: 1234567
Entre com o 2o. valor: 890
Traceback (most recent call last):
   File "mdc.py", line 13, in <module>
   print('O valor do MDC é: '.format(mdc(x,y)))
   File "mdc.py", line 6, in mdc
   return mdc(x-y, y)
   [Previous line repeated 994 more times]
   File "mdc.py", line 3, in mdc
   if x == y:
RecursionError: maximum recursion depth exceeded in comparison
```

Praticando

- 1. Crie uma função recursiva que receba um número inteiro N e calcule a soma dos números de 1 até N.

- 2. Escreva uma função recursiva para somar os elementos de uma lista de números. Ou seja, a função recebe uma lista como parâmetro e retorna um número, que é a soma dos elementos da lista.