FUNÇÕES

Funções pré-definidas, criando funções, escopo de funções, função como argumento

FUNÇÕES

- As linguagens de programação têm à sua disposição várias funções pré-definidas:
- EXEMPLO (Linguagem *Python*)
  - input()
  - print()
  - math.sqrt() (utilizando *import math*)
  - abs()

FUNÇÕES

- As linguagens de programação têm à sua disposição várias funções pré-definidas:
- EXEMPLO (Linguagem *Python*)
  - input()
  - print()
  - math.sqrt() (utilizando *import math*)

FUNÇÕES

- As linguagens de programação têm à sua disposição várias funções pré-definidas:
- EXEMPLO (Linguagem *Python*)
  - input()
  - print()
  - math.sqrt() (utilizando *import math*)
  - a s()

FUNÇÕES

- As linguagens de programação têm à sua disposição várias funções pré-definidas:
- EXEMPLO (Linguagem *Python*)
  - input()
  - print()
  - math.sqrt(16) (utilizando *import math*)
  - abs(-3)

FUNÇÕES

- As Funções Pré Definidas podem ser usadas diretamente em expressões:

- Exemplo:

```
h= math.sqrt (a + y**2 + 2 * abs(math.sin(y) ))
```

FUNÇÕES

- As Funções Pré Definidas podem ser usadas diretamente em expressões:
- Exemplo:

h= sqrt (a + pow (y,2) + 2 * sin (y))

Identificadores das FUNÇÕES

FUNÇÕES

Se houver necessidade o programador pode **definir** suas próprias **FUNÇÕES**

FUNÇÕES

- Permitem modularizar um programa.
- Facilitam o uso da abordagem dividir e conquistar no desenvolvimento de programas.
- Facilitam a reusabilidade de códigos.

FUNÇÕES

```
#declaração da função
def <nome>(<argumentos>):
    """
    Descreve input
    Descreve output
    """
    instrução
    instrução
    instrução
    instrução

    return <valorRetornado>

#chamada da função
variavel=<nome>(<arg1,arg2,...,argN)
```

FUNÇÕES

```
#declaração da função
def <nome>(<argumentos>):
    """
    Des
    Descr
    """
    instrução
    instrução
    instrução
    instrução

    return <valorRetornado>

#chamada da função
variavel=<nome>(<arg1,arg2,...,argN)
```

Palavra reservada

FUNÇÕES

```
#declaração da função
def <nome>(<argumentos>):
        """
    Descreve input
    Descreve output
    """
    instrução
    instrução
    instrução
    instrução

    return <valorRetornado>

#chamada da função
variavel=<nome>(<arg1,arg2,...,argN)
```

Parâmetros formais

FUNÇÕES

```
#declaração da função
def <nome>(<argumentos>):
    """
    Descreve input
    Descreve output
    """
    instrução
    instrução
    instrução
    instrução

    return <valorRetornado>

#chamada da função
variavel=<nome>(<arg1,arg2,...,argN)
```

docstring (opcional!!)

FUNÇÕES

```
#declaração da função
def <nome>(<argumentos>):
    """
    Descreve input
    Descreve output
    """
    instrução
    instrução
    instrução
    instrução

    return <valorRetornado>

#chamada da função
variavel=<nome>(<arg1,arg2,...,argN)
```

Corpo da função

FUNÇÕES

```
#declaração da função
def <nome>(<argumentos>):
    """
    Descreve input
    Descreve output
    """
    instrução
    instrução
    instrução
    instrução

    return <valorRetornado>

#chamada da função
variavel=<nome>(<arg1,arg2,...,argN)
```

Palavra reservada

FUNÇÕES

```
#declaração da função
def <nome>(<argumentos>):
    """
    Descreve input
    Descreve output
    """
    instrução
    instrução
    instrução
    instrução

    return <valorRetornado>

#chamada da função
variavel=<nome>(<arg1,arg2,...,argN)
```

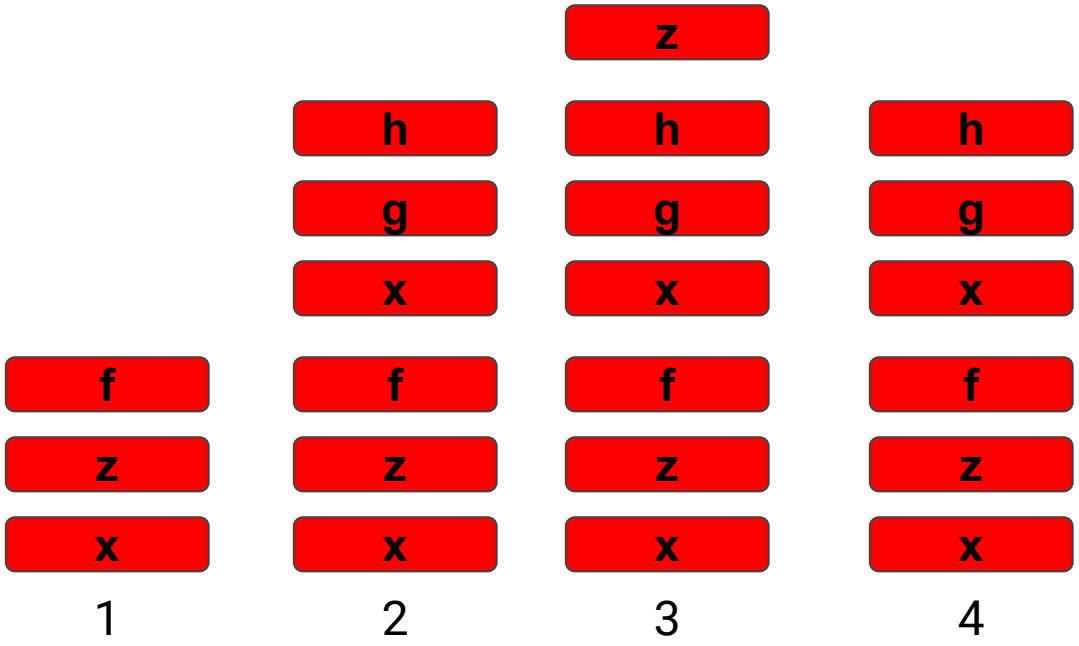
FUNÇÕES

```
#declaração da função
def <nome>(<argumentos>):
        """
    Descreve input
    Descreve output
    """
    instrução
    instrução
    instrução
    instrução

    return <valorRetornado>

#chamada da função
variavel=<nome>(<arg1,arg2,...,argN)
```

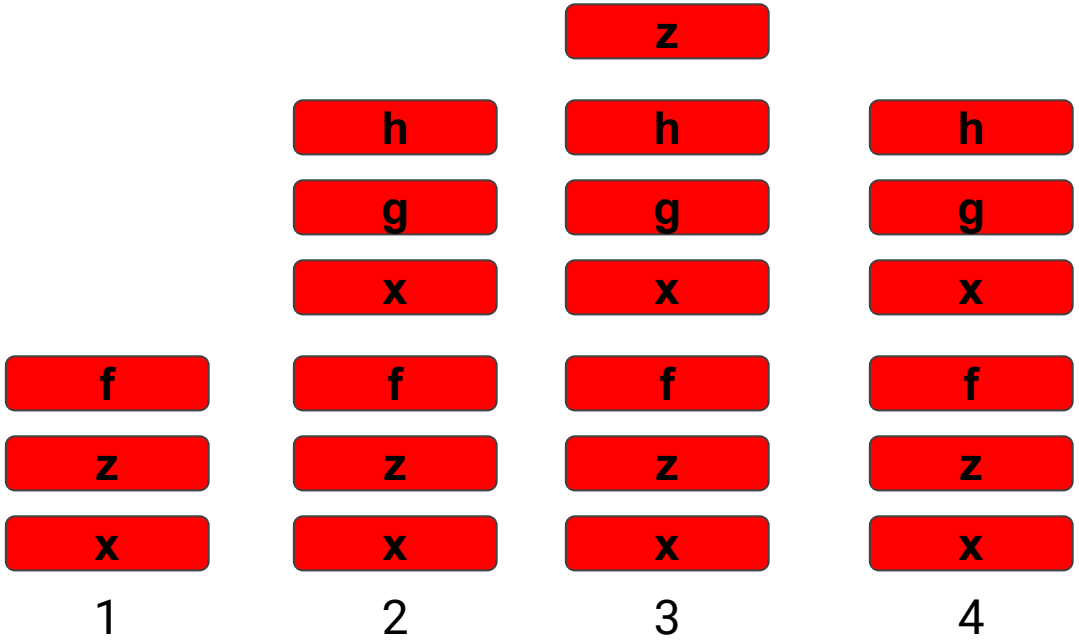
Parâmetros reais

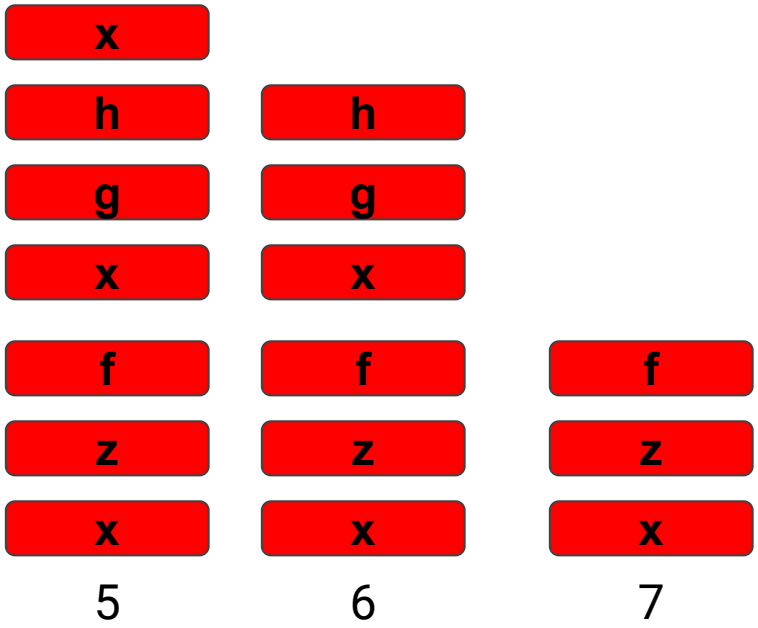
Exemplos

FUNÇÕES




```
#Exemplo: Escopo -
Empilhando chamadas
def f(x):
 def g():
   x = 'abc'
   print(f'{x =}')
 def h():
   z = x
   print(f'{z =}')
 x = x + 1
 print(f'{x =}')
 h()
 g()
 print(f'{x =}')
 return g
x = 3
z = f(x)
print(f'{x =}')
x
print(f'{z =}')
z()
```






```
#Exemplo: Escopo -
Empilhando chamadas
def f(x):
 def g():
  x = 'abc'
  print(f'{x =}')
 def h():
   z = x
   print(f'{z =}')
 x = x + 1
 print(f'{x =}')
 h()
 g()
 print(f'{x =}')
 return g
x = 3
z = f(x)
print(f'{x =}')
x
print(f'{z =}')
z()
```

FUNÇÕES

- Cada função define um escopo, ou seja, um novo espaço com códigos específicos.
- Todas as variáveis criadas no escopo definido na função são variáveis locais
  - Variáveis locais: existem apenas dentro da função onde são criadas.

- As variáveis na lista de argumentos da função também são variáveis locais.

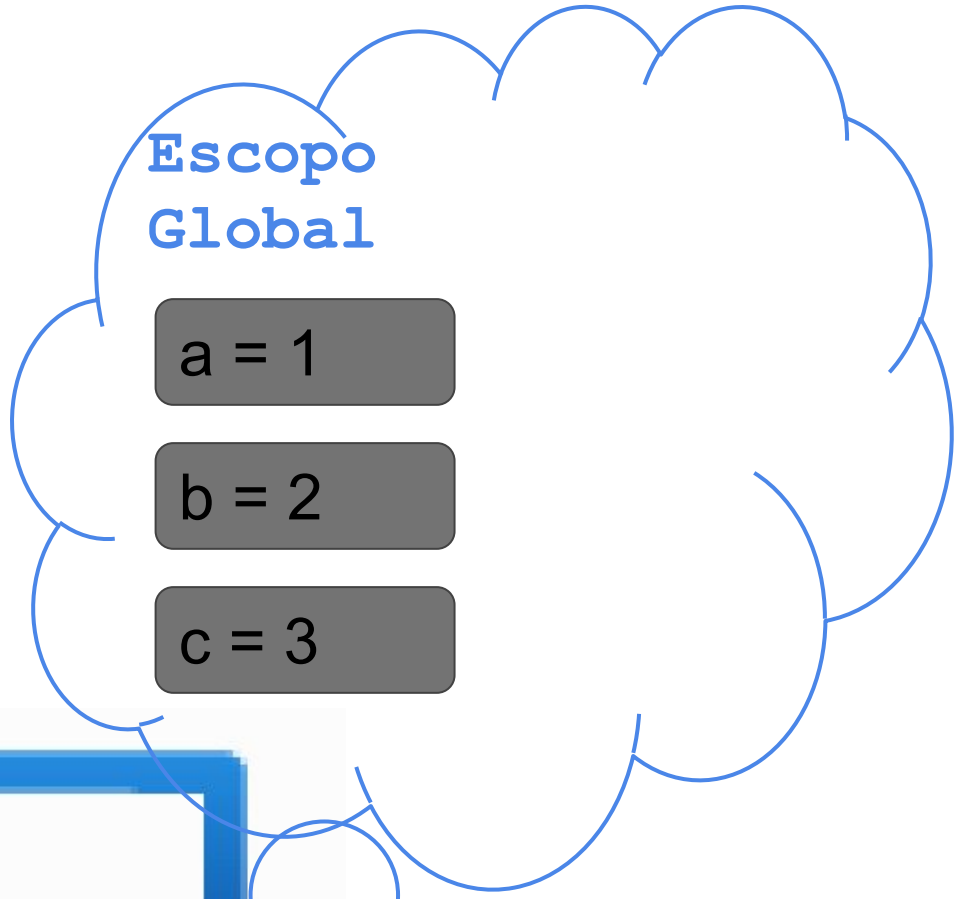
- As variáveis na lista de argumentos permitem a troca de informações entre funções.

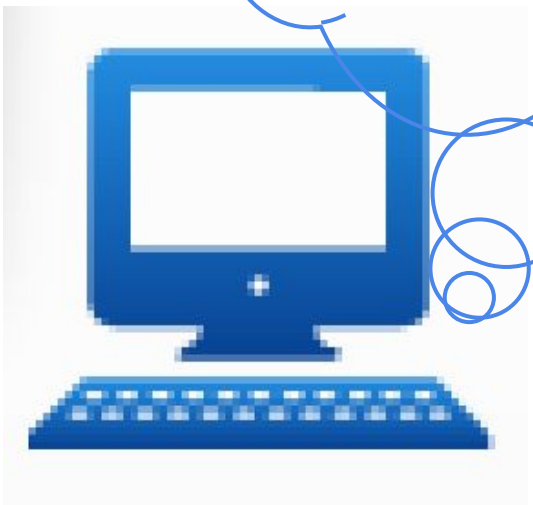
FUNÇÕES - Escopo das Variáveis

```
#declaração da função

def f(x):
   a=b=c=x
   print(a,b,c)
   return a+b+c

a,b,c=1,2,3
#chamada da função
d=f(4)
print(d)
print(a,b,c)
```

FUNÇÕES - Escopo das Variáveis

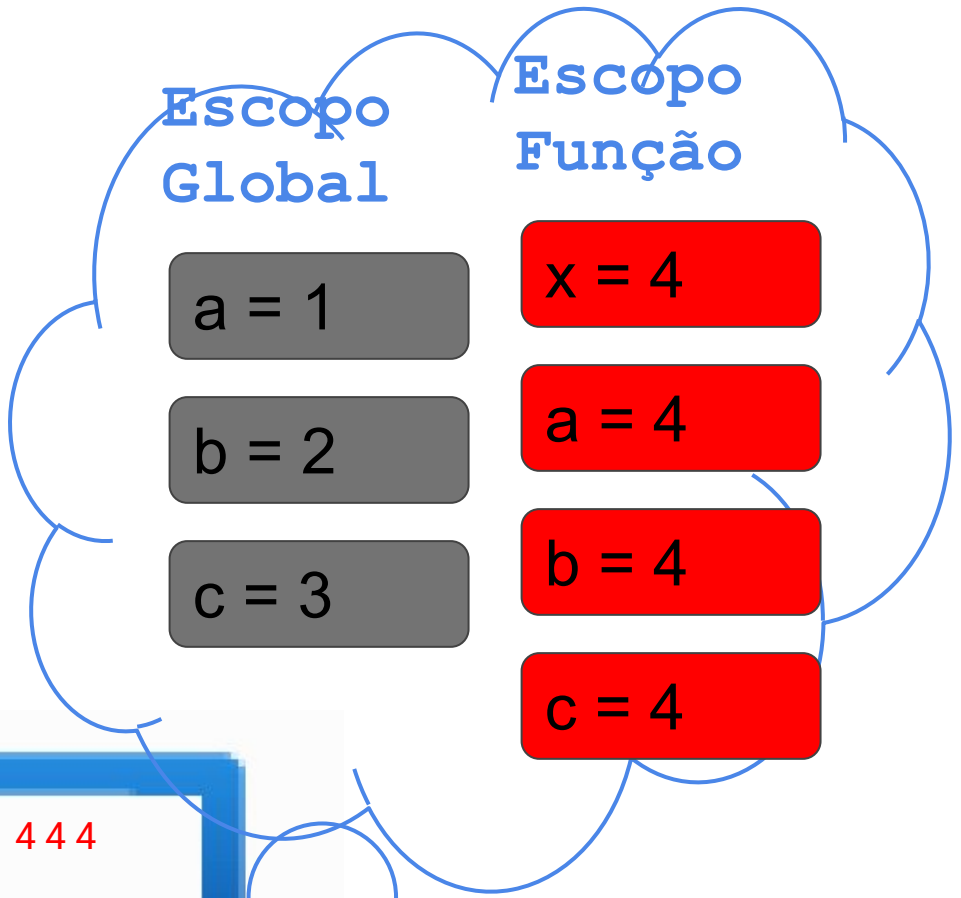
```
#declaração da função

def f(x):
    a=b=c=a=x
    print(a,b,c)
    return a+b+c

a,b,c=1,2,3
#chamada da função
d=f(4)
print(d)
print(a,b,c)
```

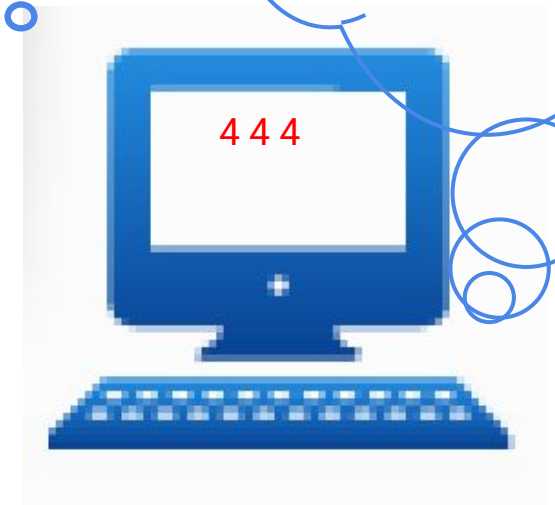




FUNÇÕES - Escopo das Variáveis

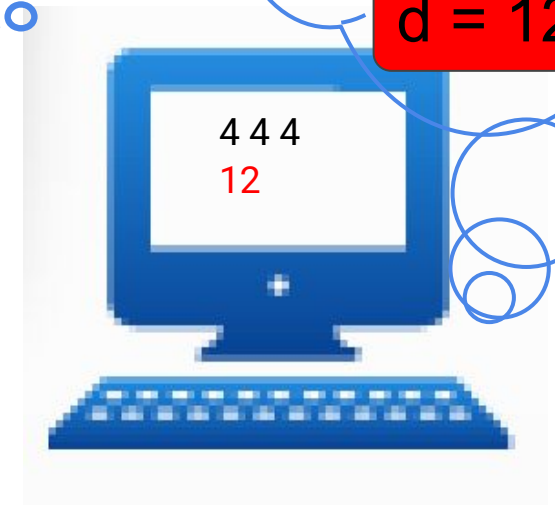
```
#declaração da função

def f(x):
    a=b=c=x
    print(a,b,c)
    return a+b+c

a,b,c=1,2,3
#chamada da função
d=f(4)
print(d)
print(a,b,c)
```

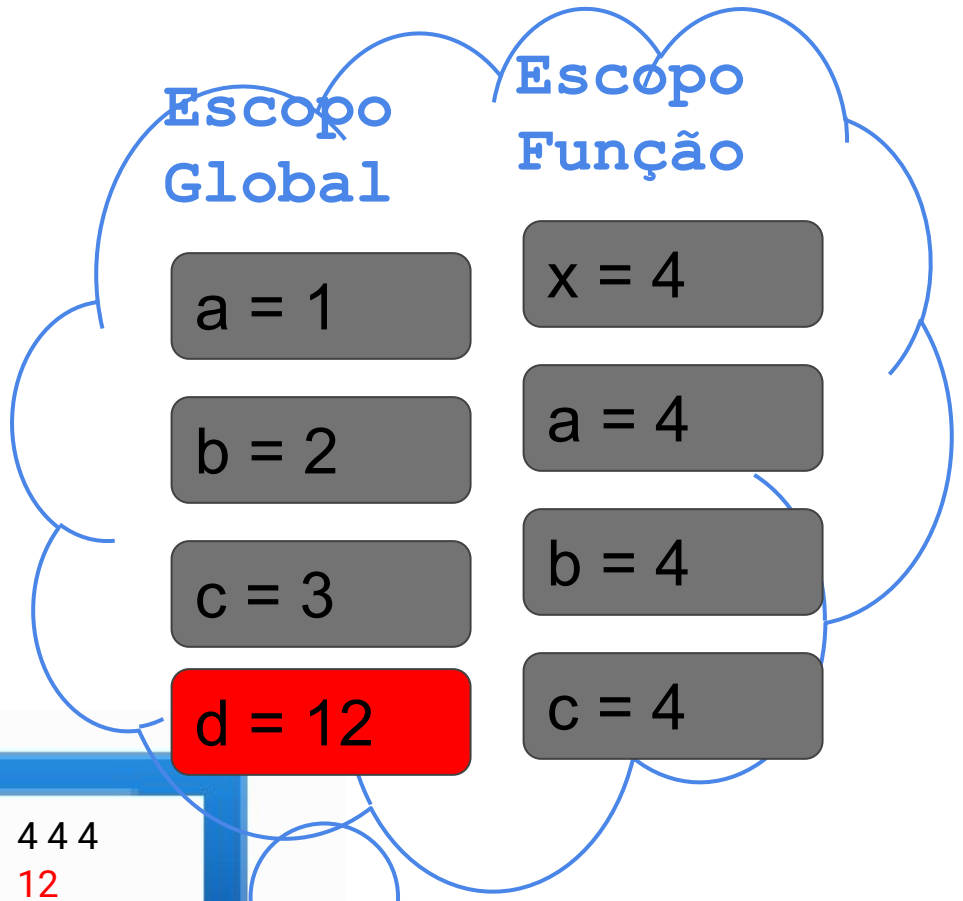




FUNÇÕES - Escopo das Variáveis

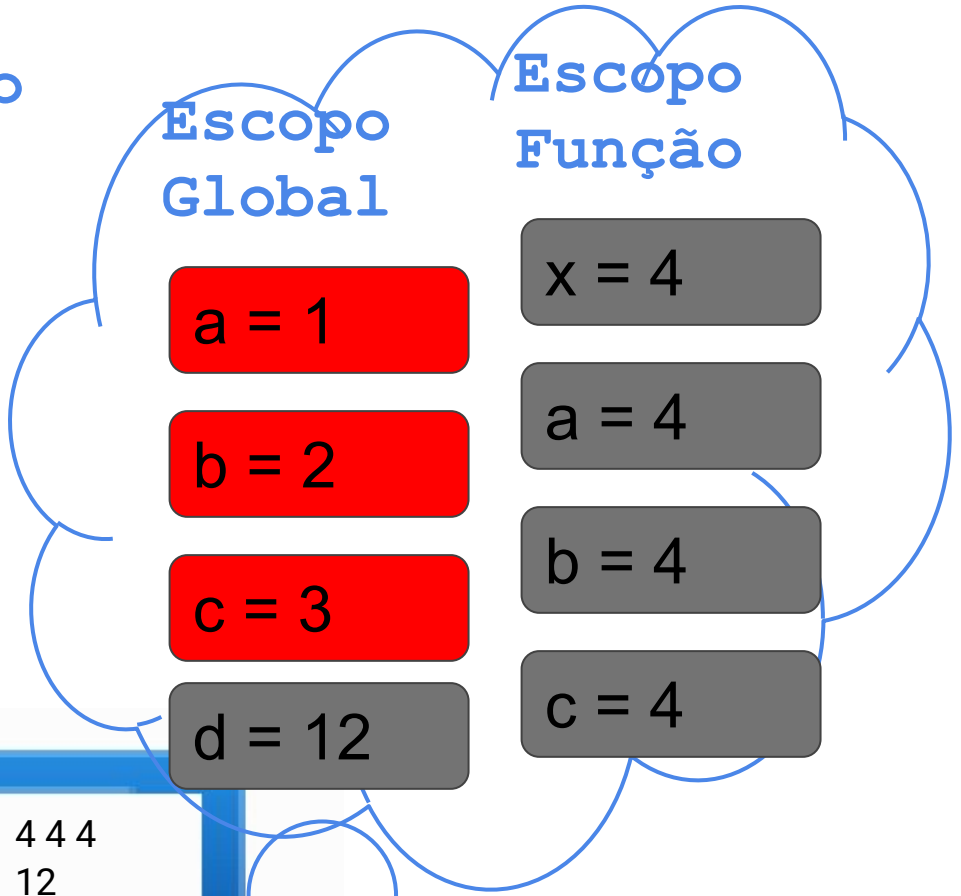
```
#declaração da função

def f(x):
    a=b=c=a=x
    print(a,b,c)
    return a+b+c

a,b,c=1,2,3
#chamada da função
d=f(4)
print(d)
print(a,b,c)
```

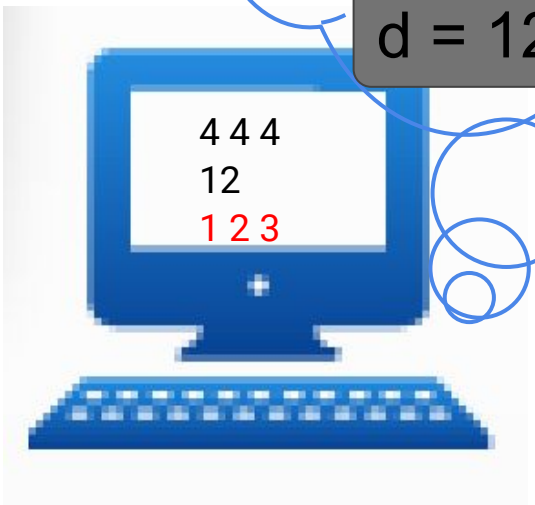




FUNÇÕES - Escopo das Variáveis

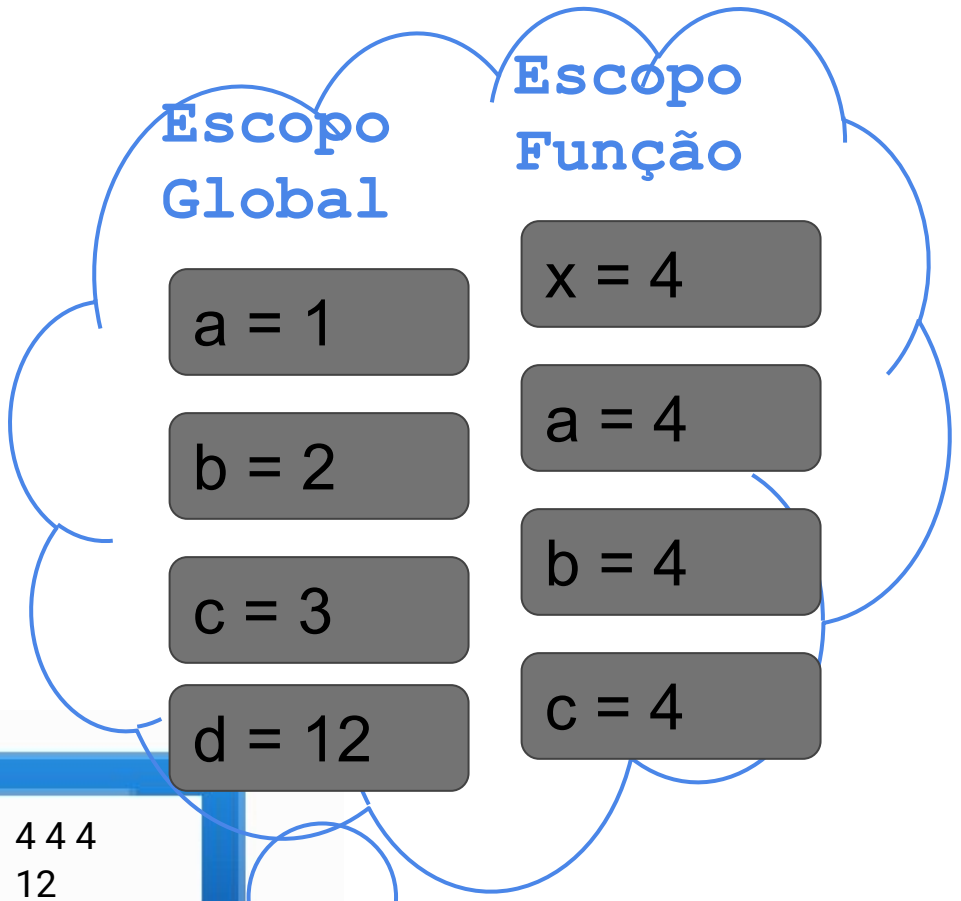
```
#declaração da função

def f(x):
    a=b=c=a=x
    print(a,b,c)
    return a+b+c

a,b,c=1,2,3
#chamada da função
d=f(4)
print(d)
print(a,b,c)
```

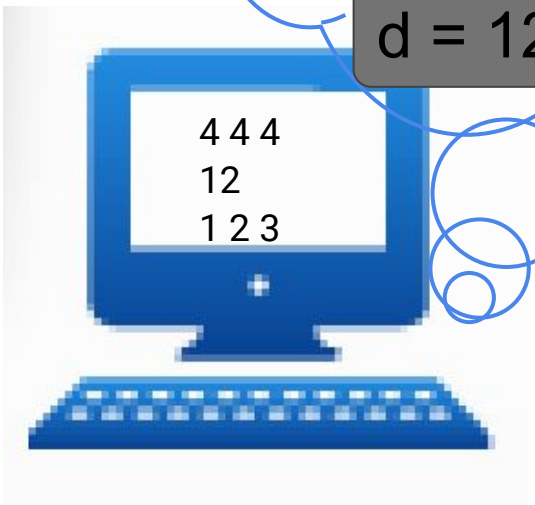




FUNÇÕES - Escopo das Variáveis

```
#declaração da função

def f(x):
    a=b=c=a=x
    print(a,b,c)
    return a+b+c

a,b,c=1,2,3
#chamada da função
d=f(4)
print(d)
print(a,b,c)
```





FUNÇÕES - Return

- O comando **return** é responsável por encerrar a execução da função e retornar o valor desejado.

- Uma função sem o comando return retorna None que indica a ausência de valor retornado.

Exemplos

Recursão

- Uma função é recursiva quando chama a si mesma.
- Exemplo:

```
def sum(n):
  if (n<=1):
     return n
  return (n+sum(n-1))
```

Entrada	Saída	
Sum(1)	1	
Sum(2)	2 + sum(1)	2+ 1
Sum(3)	3 + sum(2)	3 + 2 + 1
Sum(4)	4 + sum(3)	4 + 3 + 2 + 1

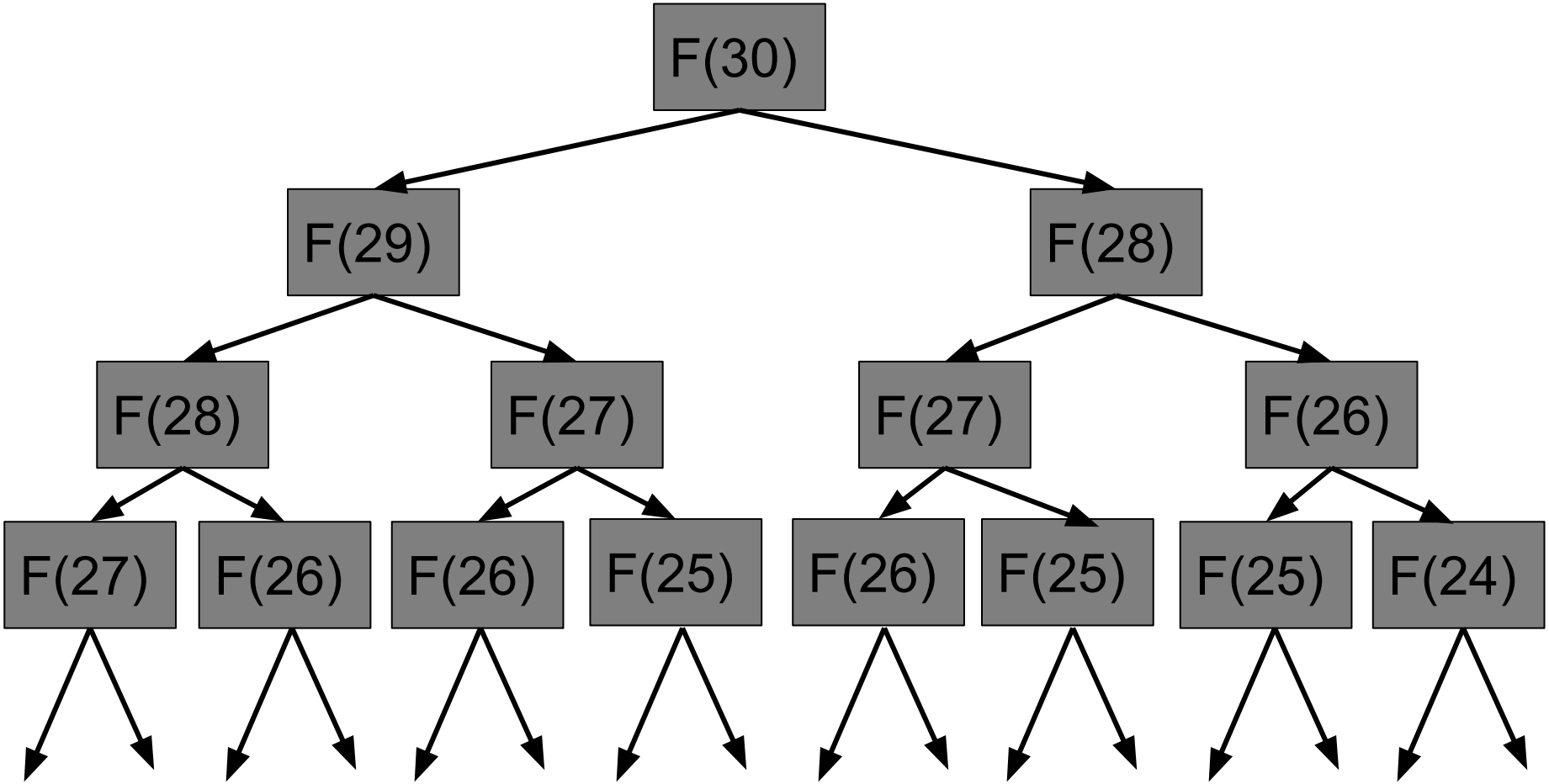
Recursão

- Exemplo: Sequência de Fibonacci.
  f(0)=0, f(1)=1, f(i+1)=f(i)+f(i-1), i=1,2,...

```
def fib(int n):
   if(n<=1):
      return 1
   else:
      return (fib(n-1) + fib(n-2))
```

n	F(n)	Número de chamadas da função
0	1	1
1	1	1
2	1	3
...	...	...
23	28657	92735
24	46368	150049

Recursão



Recursão

- Recursão vs Iteração
  - Iteração usa estrutura de repetição
  - Recursão usa estrutura de seleção (chamadas de funções repetitivas).
  - Iteração usa violação da condição como critério de parada.
  - Recursão usa passo base como critério de parada.
  - Ambos podem ser executados infinitamente, se o critério de parada não for satisfeito.
- Desvantagens da Recursão
  - Gera sobrecarga (overhead) com as chamadas de função, gerando gasto de tempo de processamento e espaço de memória.
  - Uma cópia da função (variáveis da função) é criada, consumindo memória.
  - Logo, a iteração tende a ser mais rápida por não fazer repetidas chamadas de funções.

Recursão

- Quando usar Recursão ou Iteração?

Recursão	Iteração
Estruturas condicionais	Estruturas de repetição
Repetição implícita	Repetição explícita
Caso base como critério de parada	Condição com critério de parada
Lento	Rápido
Solução simples	Solução complexa
Fácil manutenção	Difícil Manutenção

Recursão

Palíndromo

Exemplos