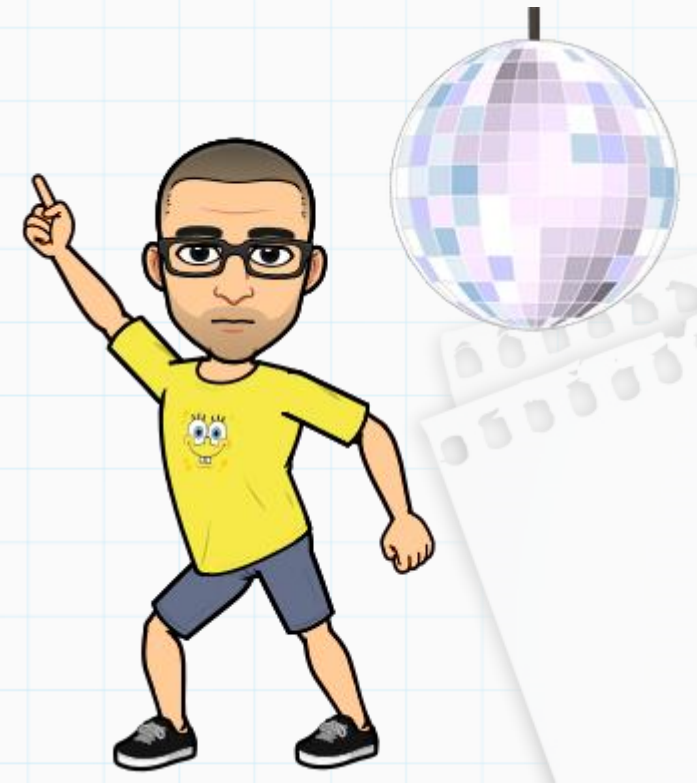


Programação De Computadores

Professor : Yuri Frota

www.ic.uff.br/~yuri/prog.html

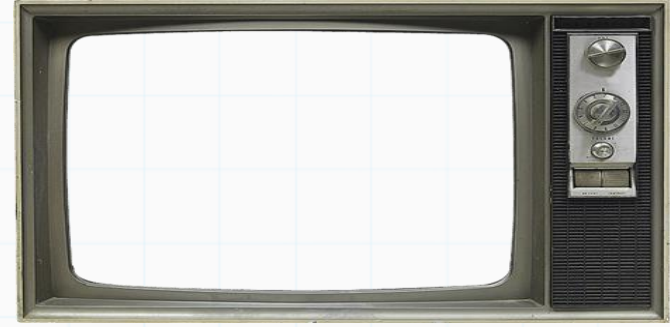
yuri@ic.uff.br



Subrotinas e Funções

Seja o seguinte código:

```
1  x = 5
2
3  a = 4
4  f = 1
5  for i in range (1,a+1):
6      f = f*i
7  x = x + f
8
9  a = 7
10 f = 1
11 for i in range (1,a+1):
12     f = f*i
13 x = x + f
14
15 a = 5
16 f = 1
17 for i in range (1,a+1):
18     f = f*i
19 x = x + f
```



Subrotinas e Funções

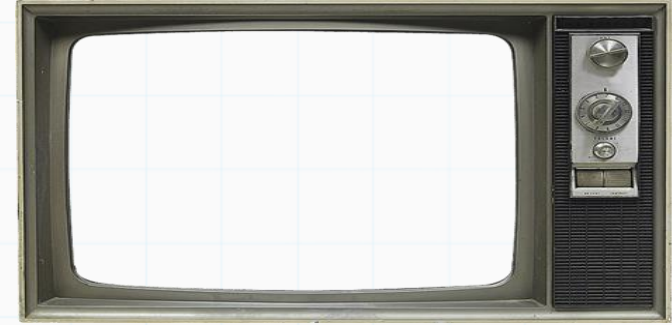
Seja o seguinte código:

```
1  x = 5
2
3  a = 4
4  f = 1
5  for i in range (1,a+1):
6      f = f*i
7  x = x + f
8
9  a = 7
10 f = 1
11 for i in range (1,a+1):
12     f = f*i
13 x = x + f
14
15 a = 5
16 f = 1
17 for i in range (1,a+1):
18     f = f*i
19 x = x + f
```

- Vemos que temos trechos de códigos repetidos (fatorial)

- Problema:

- Repetição do trecho de código
- Erros ficam difíceis de corrigir (e se eu esquecer de corrigir o erro em uma das N repetições daquele trecho de código?)



Subrotinas e Funções

Seja o seguinte código:

```
1  x = 5
2
3  a = 4
4  f = 1
5  for i in range (1,a+1):
6      f = f*i
7  x = x + f
8
9  a = 7
10 f = 1
11 for i in range (1,a+1):
12     f = f*i
13 x = x + f
14
15 a = 5
16 f = 1
17 for i in range (1,a+1):
18     f = f*i
19 x = x + f
```

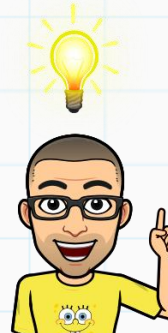
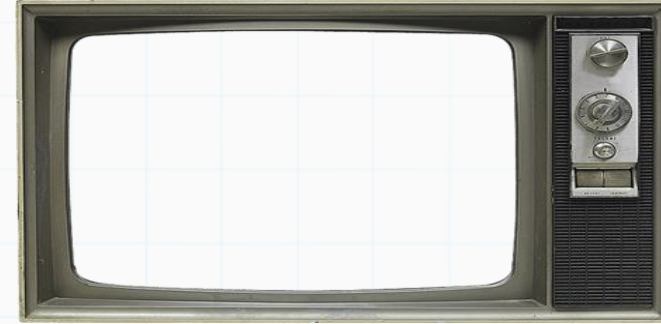
- Vemos que temos trechos de códigos repetidos (fatorial)

- Problema:

- Repetição do trecho de código
- Erros ficam difíceis de corrigir (e se eu esquecer de corrigir o erro em uma das N repetições daquele trecho de código?)

- Solução:

- encapsular o código em uma subrotina (função) para poder usar quantas vezes quiser.
- Código fica mais organizado

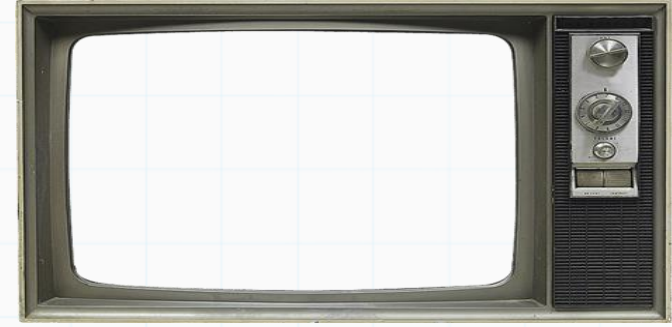


Subrotinas e Funções

Seja o seguinte código:

```
1  x = 5
2
3  a = 4
4  f = 1
5  for i in range (1,a+1):
6      f = f*i
7  x = x + f
8
9  a = 7
10 f = 1
11 for i in range (1,a+1):
12     f = f*i
13 x = x + f
14
15 a = 5
16 f = 1
17 for i in range (1,a+1):
18     f = f*i
19 x = x + f
```

Função Fatorial:



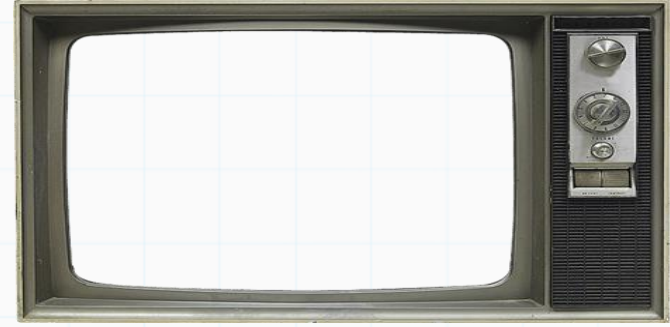
Subrotinas e Funções

Fluxo de Execução:

programa principal



```
...  
...  
a()  
...  
...  
...  
b()  
...  
...
```



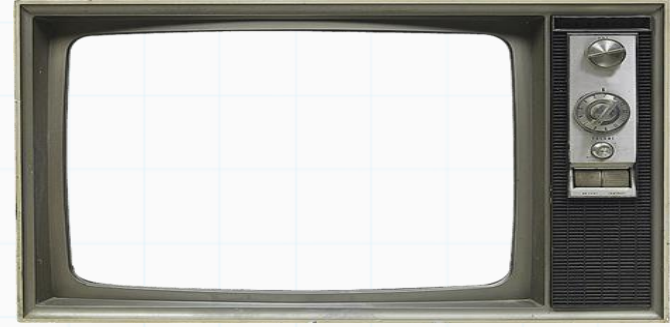
Subrotinas e Funções

Fluxo de Execução:

programa principal

...
...
a()
...
...
...
b()
...
...

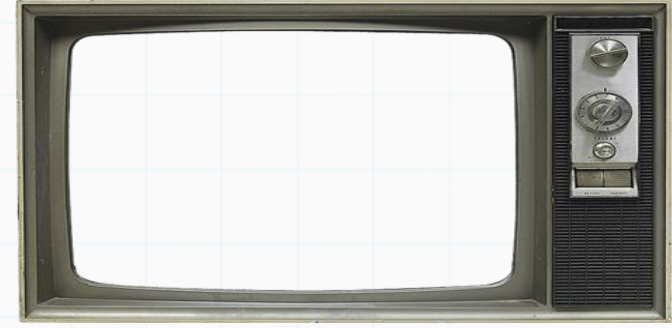
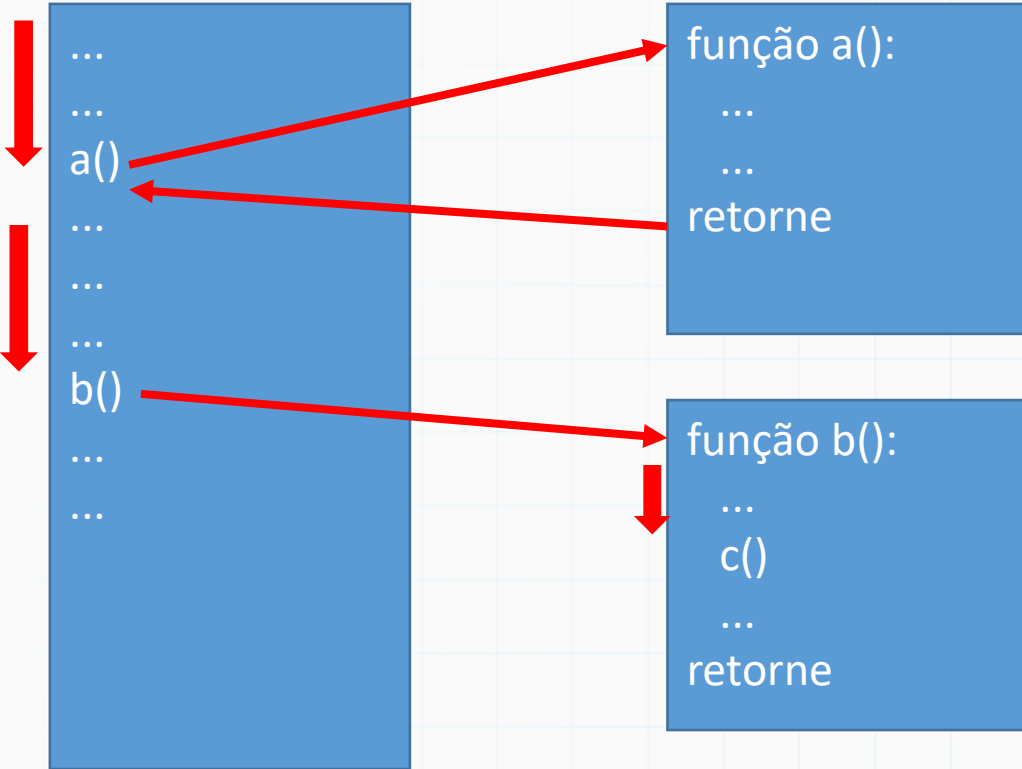
função a():
...
...
retorne



Subrotinas e Funções

Fluxo de Execução:

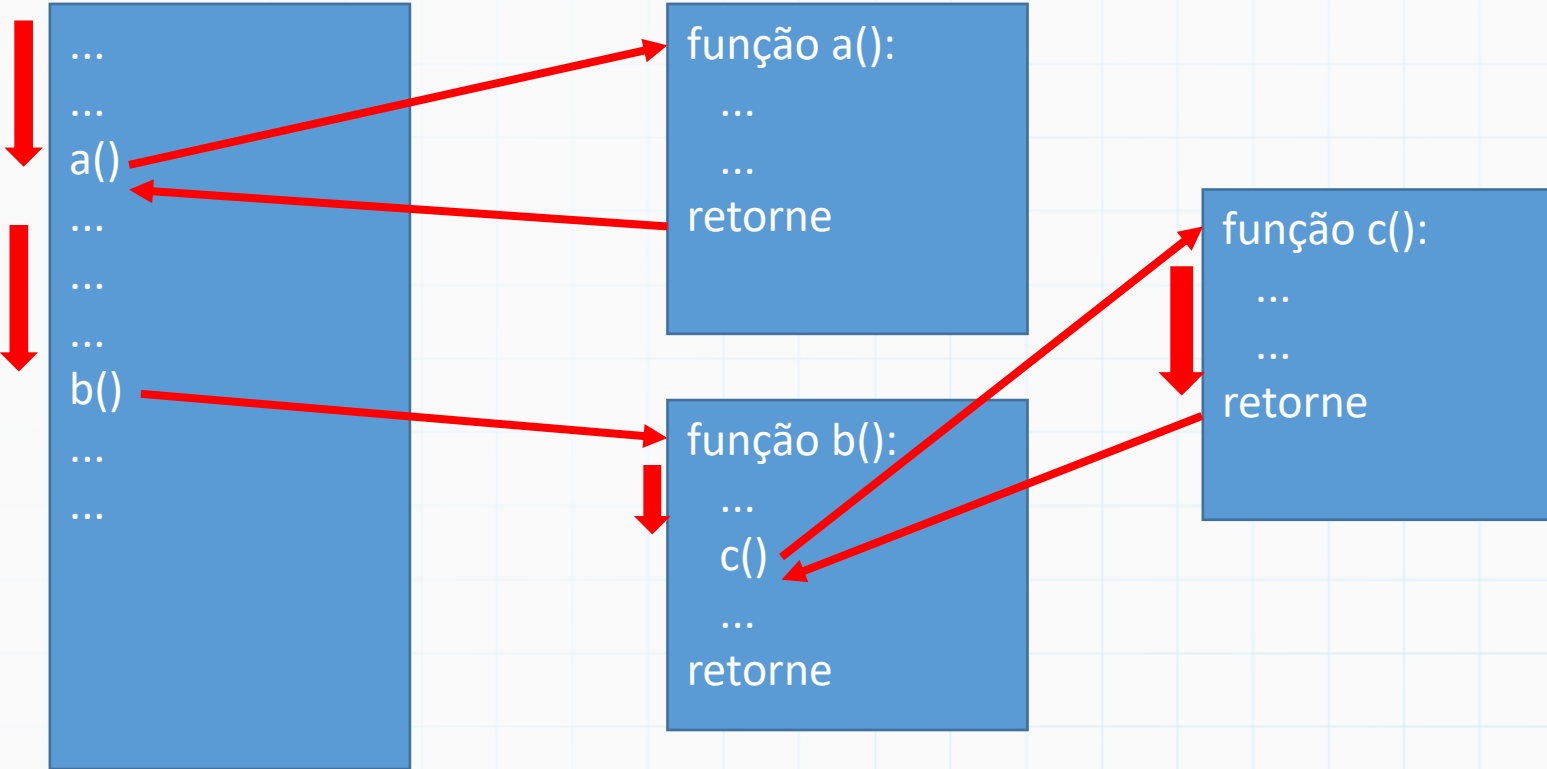
programa principal



Subrotinas e Funções

Fluxo de Execução:

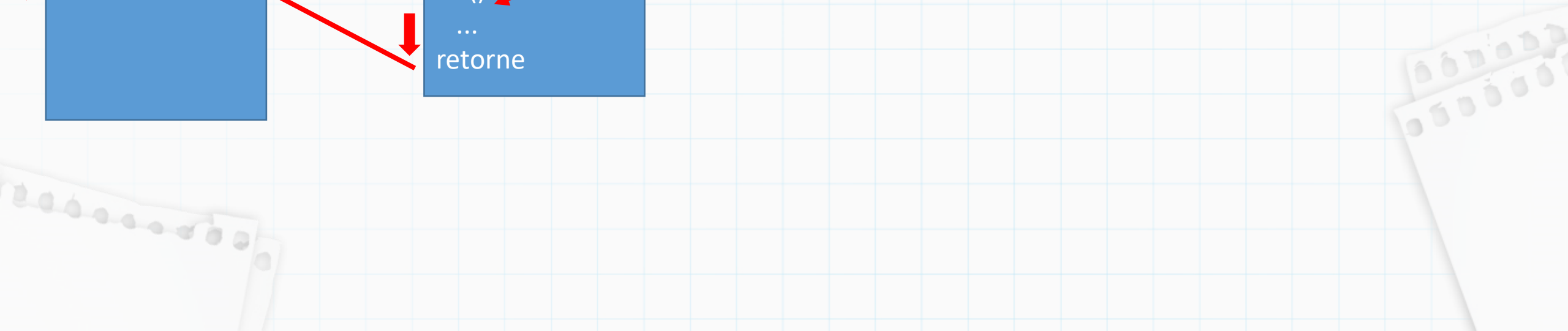
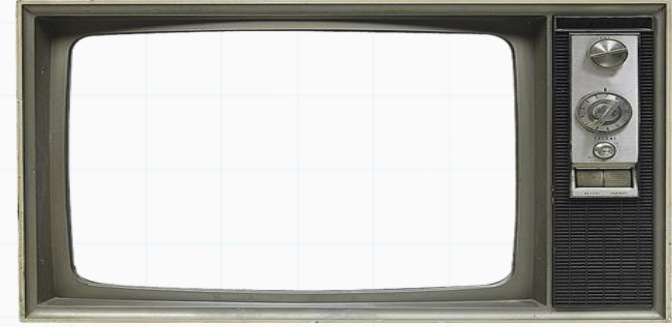
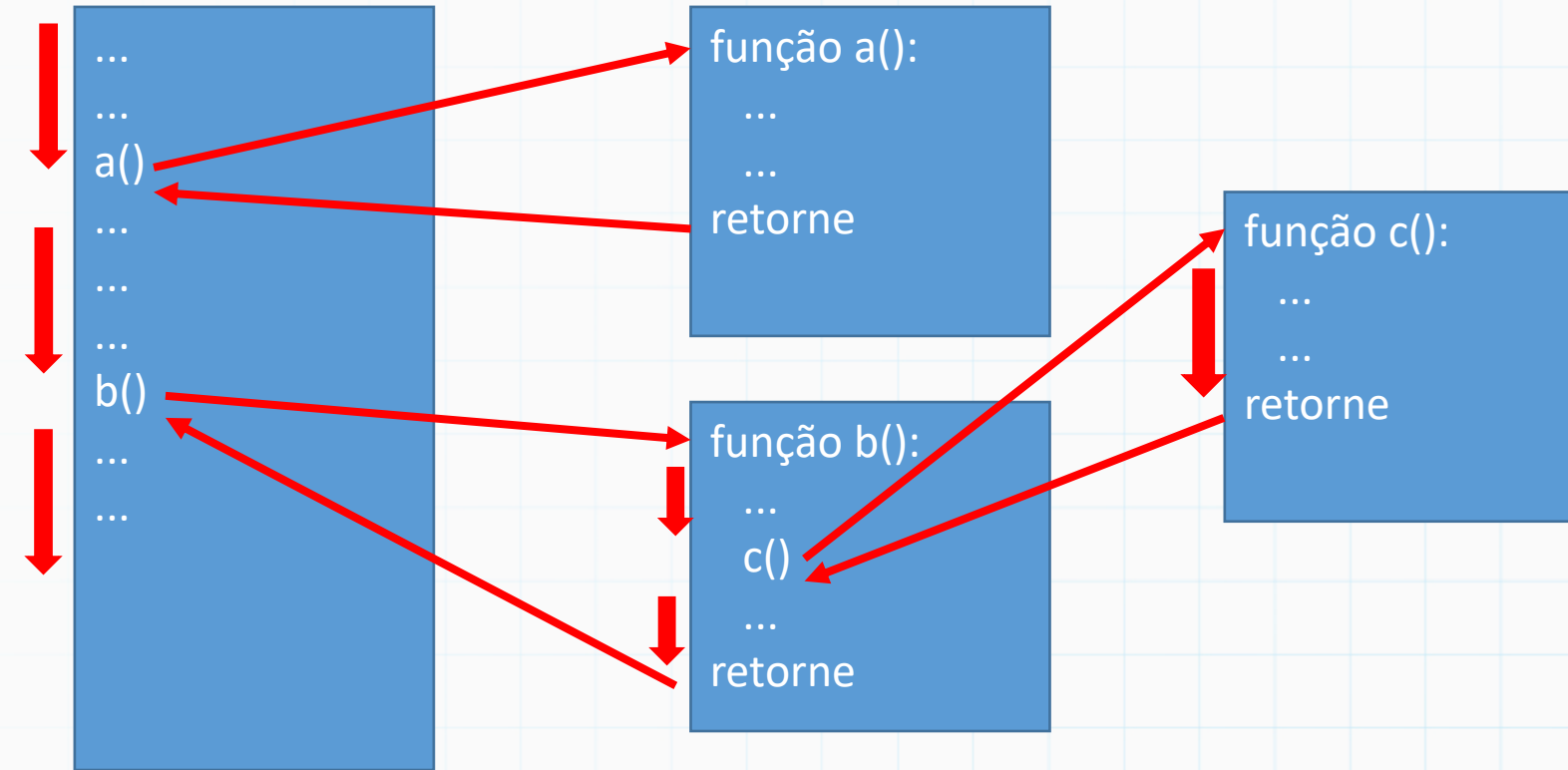
programa principal



Subrotinas e Funções

Fluxo de Execução:

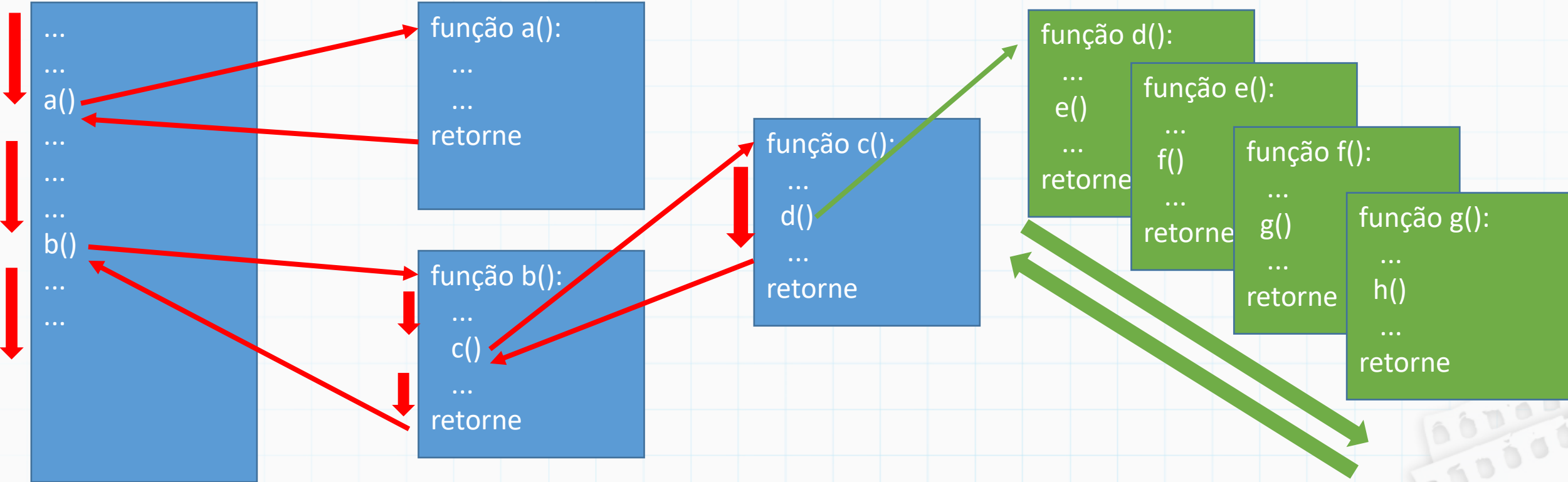
programa principal



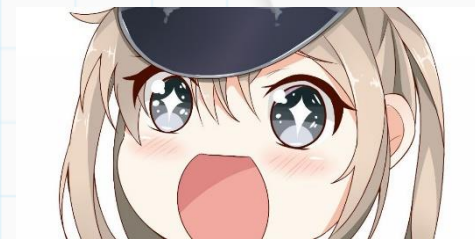
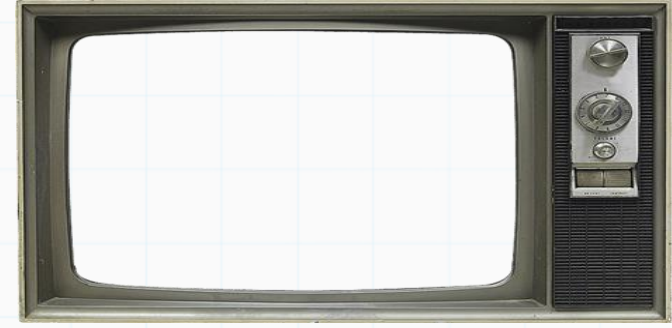
Subrotinas e Funções

Fluxo de Execução:

programa principal



e se...



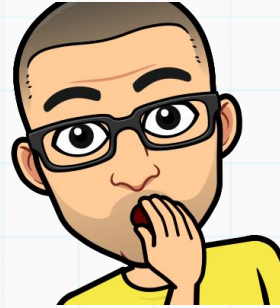
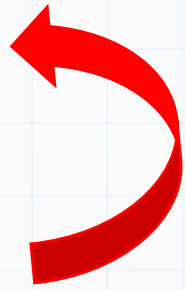
Subrotinas e Funções

Fluxo de Execução: Existem até funções que chamam a si próprias (recursão)

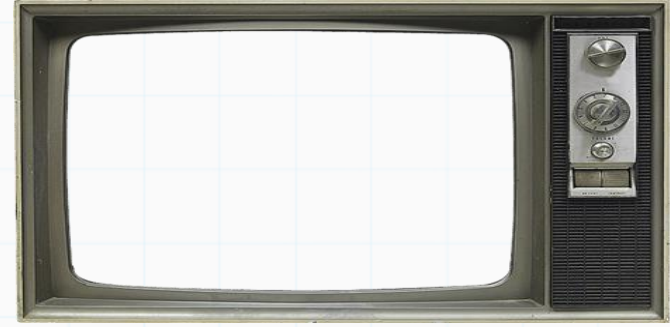
programa principal

...
...
a()
...
...
b()
...
...

função a():
...
a()
...
retorne



cuidado!




Subrotinas e Funções

Funções são geralmente declaradas no início do programa, logo a execução do programa principal começa no primeiro comando fora de uma função

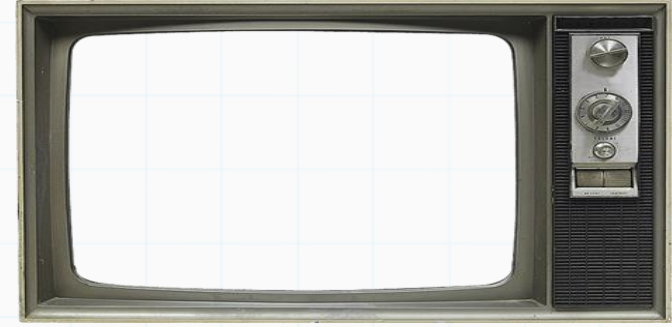
```
função a():  
    retorne
```

```
função b():  
    retorne
```

```
função c():  
    retorne
```



```
1 x = 5  
2  
3 a = 4  
4 f = 1  
5 for i in range (1,a+1):  
6     f = f*i  
7 x = x + f  
8
```



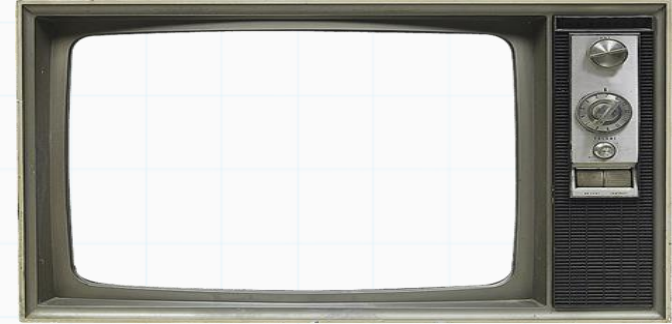
Subrotinas e Funções

Então, como declarar uma função ?

```
def nome_funcao (parâmetro, parâmetro, ..., parâmetro):  
    <comandos>  
    [return <variável ou valor>]
```

Ex:

```
1 def fatorial(a):  
2     f = 1  
3     for i in range (1,a+1):  
4         f = f*i  
5     return f
```



Subrotinas e Funções

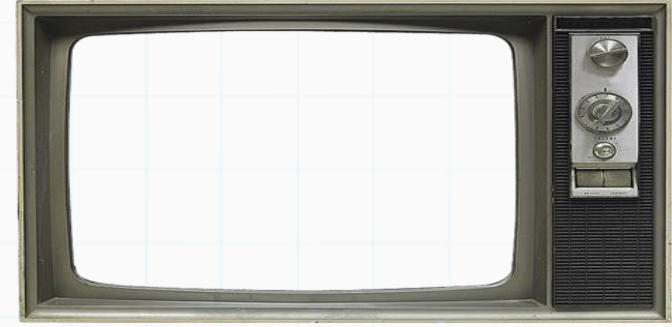
Então, como declarar uma função ?

```
def nome_funcao (parâmetro, parâmetro, ..., parâmetro):  
    <comandos>  
    [return <variável ou valor>]
```

Ex:

```
1 def fatorial(a):  
2     f = 1  
3     for i in range (1,a+1):  
4         f = f*i  
5     return f  
6  
7 x = 5  
8 a = 4  
9 x = x + fatorial(a)  
10 a = 7  
11 x = x + fatorial(a)  
12 a = 5  
13 x = x + fatorial(a)
```

[código](#)



A vintage television set with a dark wood-grain cabinet and a large, rounded screen. The right side of the cabinet features a vertical control panel with a silver-toned dial, a smaller dial, and a speaker grille.

```
def nome_funcao (parâmetro, parâmetro, ..., parâmetro):  
    <comandos>  
    [return <variável ou valor>]
```

```
1 def fatorial(a):
2     f = 1
3     for i in range(1, a+1):
4         f = f*i
5     return f
6
7 x = 5
8 a = 4
9 x = x + fatorial(a)
10 a = 7
11 x = x + fatorial(a)
12 a = 5
13 x = x + fatorial(a)
```

- Uma função pode retornar um valor ou não

```
1 def asterisco(n):
2     for i in range (n):
3         print('*',end=' ')
4
5 asterisco(10)
```

Shell ×

```
""" %KUTL  ZZ - ex2.py
```

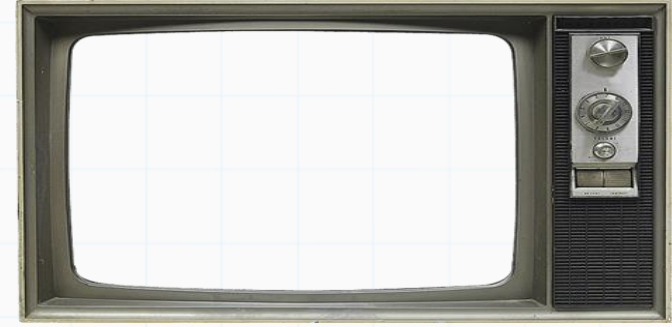
★ ★ ★ ★ ★ ★ ★ ★ ★ ★

código

Subrotinas e Funções

Então, como declarar uma função ?

```
def nome_funcao (parâmetro, parâmetro, ..., parâmetro):  
    <comandos>  
    [return <variável ou valor>]
```



Ex:

```
1 def fatorial(a):  
2     f = 1  
3     for i in range (1,a+1):  
4         f = f*i  
5     return f  
6  
7 x = 5  
8 a = 4  
9 x = x + fatorial(a)  
10 a = 7  
11 x = x + fatorial(a)  
12 a = 5  
13 x = x + fatorial(a)
```

[código](#)

- Uma função pode retornar mais de um valor

```
1 def esfera(raio):  
2     diam = 2 * raio  
3     area = 3.14 * (raio**2)  
4  
5     return diam,area  
6  
7 r = 5  
8 a,b = esfera(r)  
9 print(a,b)
```

Shell ×

```
>>> %Run teste.py  
10 78.5
```

[código](#)

Subrotinas e Funções

Então, como declarar uma função ?

```
def nome_funcao (parâmetro, parâmetro, ..., parâmetro):  
    <comandos>  
    [return <variável ou valor>]
```

Ex:

```
1 def fatorial(a):  
2     f = 1  
3     for i in range (1,a+1):  
4         f = f*i  
5     return f  
6  
7 x = 5  
8 a = 4  
9 x = x + fatorial(a)  
10 a = 7  
11 x = x + fatorial(a)  
12 a = 5  
13 x = x + fatorial(a)
```

[código](#)

- Uma função pode não ter nenhum parâmetro

```
1 def menu():  
2     print('--Menu--')  
3     print('1) Somar')  
4     print('2) Subtrair')  
5     print('3) Sair')  
6  
7 menu()
```

Shell ×

Python 3.7.7 (bundled)

>>> %Run teste.py

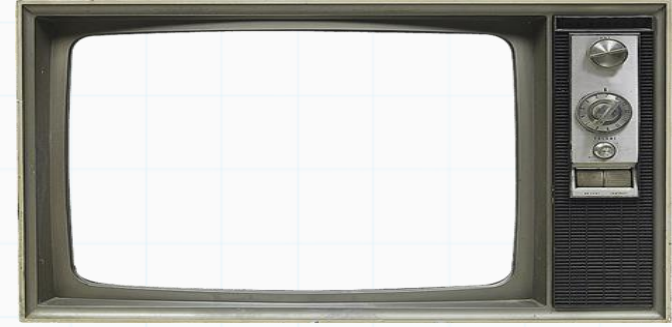
--Menu--

1) Somar

2) Subtrair

3) Sair

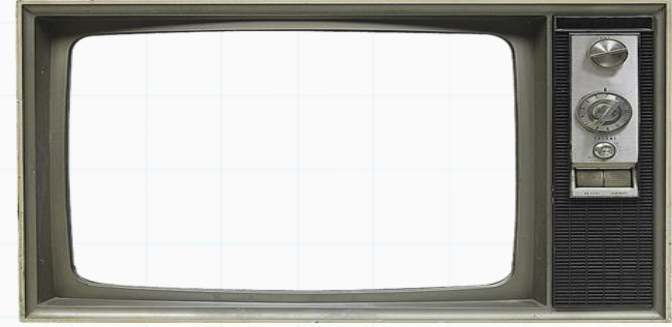
[código](#)



Subrotinas e Funções

Então, como declarar uma função ?

```
def nome_funcao (parâmetro, parâmetro, ..., parâmetro):  
    <comandos>  
    [return <variável ou valor>]
```



Ex:

```
1 def fatorial(a):  
2     f = 1  
3     for i in range (1,a+1):  
4         f = f*i  
5     return f  
6  
7 x = 5  
8 a = 4  
9 x = x + fatorial(a)  
10 a = 7  
11 x = x + fatorial(a)  
12 a = 5  
13 x = x + fatorial(a)
```

- ou mais de um parâmetro

```
1 def calcula_tempo(velocidade, distancia):  
2     tempo = distancia/velocidade  
3     return tempo  
4  
5 t = calcula_tempo(10, 5)  
6 print(t)  
7
```

Shell ×

Python 3.7.7 (bundled)

>>> %Run teste.py

0.5

[código](#)

[código](#)

Subrotinas e Funções

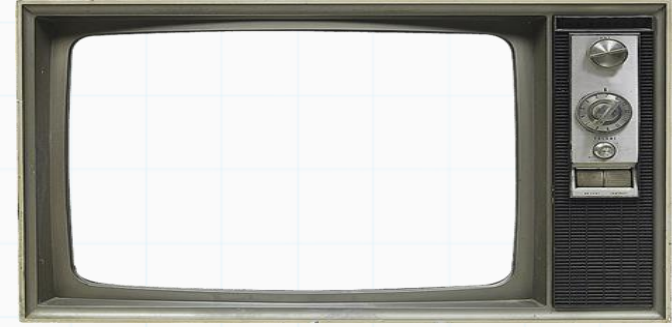
Escopo das Variáveis:

Variáveis locais (locais a função)

Declaradas dentro de uma função

São visíveis somente dentro da função onde foram declaradas

São destruídas ao término da execução da função



```
1 def calcula_tempo(velocidade, distancia):  
2     tempo = distancia/velocidade  
3     return tempo  
4  
5 v = 10  
6 d = 5  
7 t = calcula_tempo(v, d)  
8 print(t)
```

parâmetros são
variáveis locais

Subrotinas e Funções

Escopo das Variáveis:

Variáveis locais (locais a função)

Declaradas dentro de uma função

São visíveis somente dentro da função onde foram declaradas

São destruídas ao término da execução da função

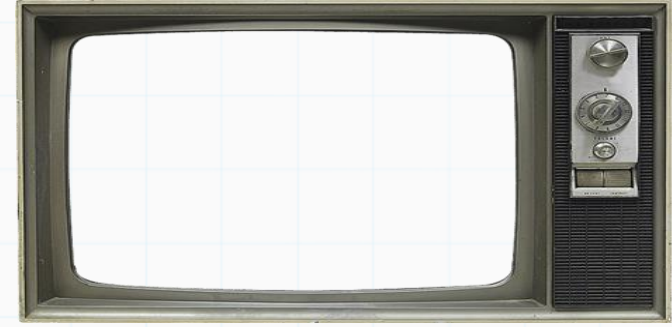
Variáveis globais

Declaradas fora de todas as funções

São visíveis por TODAS as funções do programa

```
1 def calcula_tempo(velocidade, distancia):
2     tempo = distancia/velocidade
3     return tempo
4
5 v = 10
6 d = 5
7 t = calcula_tempo(v, d)
8 print(t)
```

parâmetros são
variáveis locais



Subrotinas e Funções

Escopo das Variáveis:

Variáveis locais (locais a função)

Declaradas dentro de uma função

São visíveis somente dentro da função onde foram declaradas

São destruídas ao término da execução da função

Variáveis globais

Declaradas fora de todas as funções

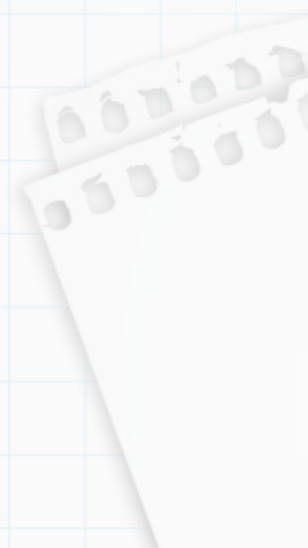
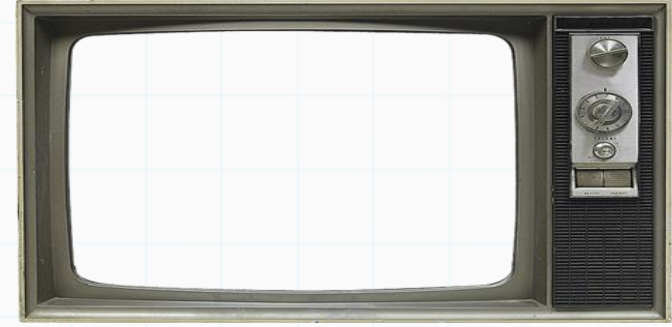
São visíveis por TODAS as funções do programa

```
1 def calcula_tempo(velocidade, distancia):
2     tempo = distancia/velocidade
3     print(v)
4     return tempo
5
6 v = 10
7 d = 5
8 t = calcula_tempo(v, d)
9 print(t)
```

Variáveis Globais podem ser acessadas dentro da função

10
0.5

[código](#)



Subrotinas e Funções

Escopo das Variáveis:

Variáveis locais (locais a função)

Declaradas dentro de uma função

São visíveis somente dentro da função onde foram declaradas

São destruídas ao término da execução da função

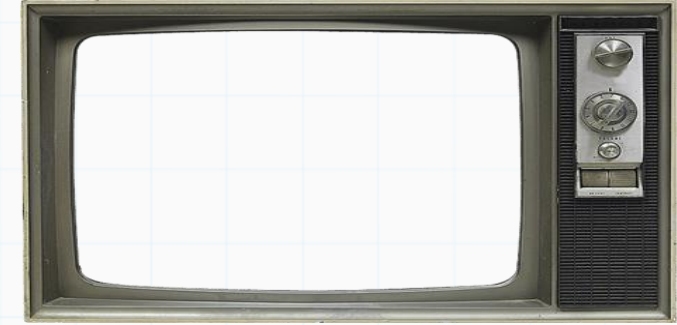
Variáveis globais

Declaradas fora de todas as funções

São visíveis por TODAS as funções do programa

```
1 def calcula_tempo(velocidade, distancia):
2     tempo = distancia/velocidade
3     print(v)
4     return tempo
5
6 v = 10
7 d = 5
8 t = calcula_tempo(v, d)
9 print(t)
```

[código](#)



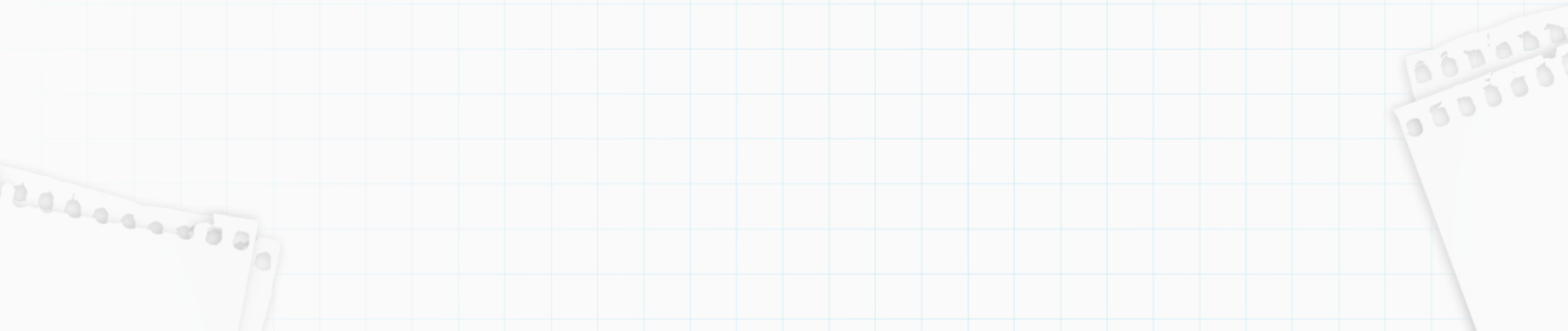
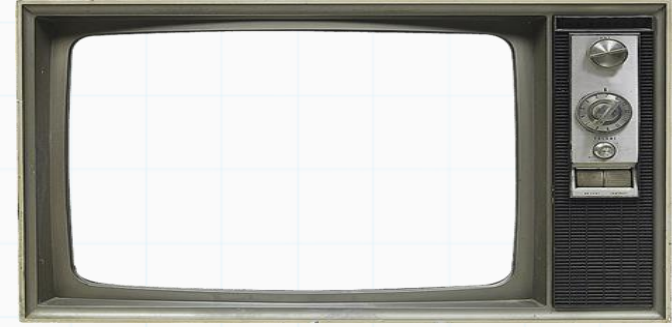
CUIDADO:

Se a variável pode ser usada por qualquer função do programa, encontrar um erro envolvendo o valor desta variável pode ser muito complexo

Sempre que possível, usar variáveis LOCAIS e passar os valores necessários para a função como parâmetro

Subrotinas e Funções

Passagem de Parâmetros: Quando uma função é chamada, é necessário fornecer um valor para cada um de seus parâmetros

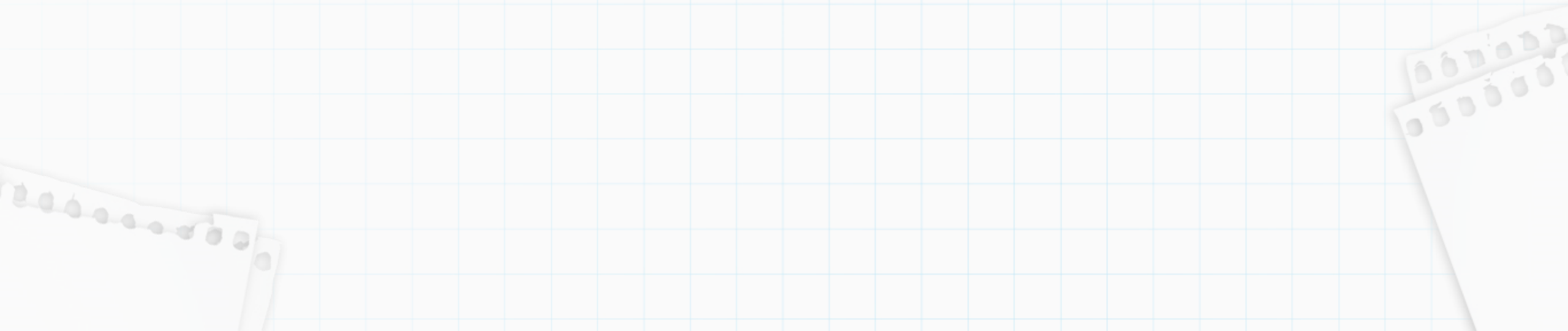
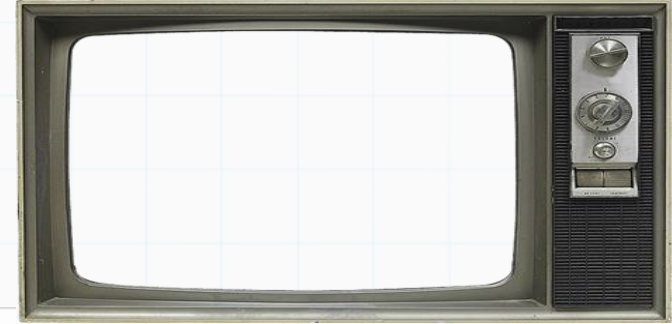


Subrotinas e Funções

Passagem de Parâmetros: Quando uma função é chamada, é necessário fornecer um valor para cada um de seus parâmetros

- Pode ser diretamente

```
1 def calcula_tempo(velocidade, distancia):  
2     tempo = distancia/velocidade  
3     return tempo  
4  
5 t = calcula_tempo(10, 5)  
6 print(t)  
7
```



Subrotinas e Funções

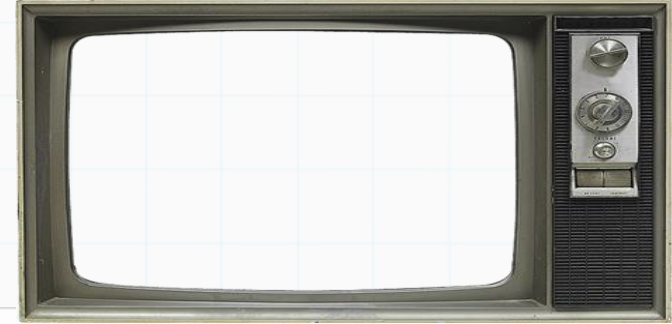
Passagem de Parâmetros: Quando uma função é chamada, é necessário fornecer um valor para cada um de seus parâmetros

- Pode ser diretamente

```
1 def calcula_tempo(velocidade, distancia):
2     tempo = distancia/velocidade
3     return tempo
4
5 t = calcula_tempo(10, 5)
6 print(t)
```

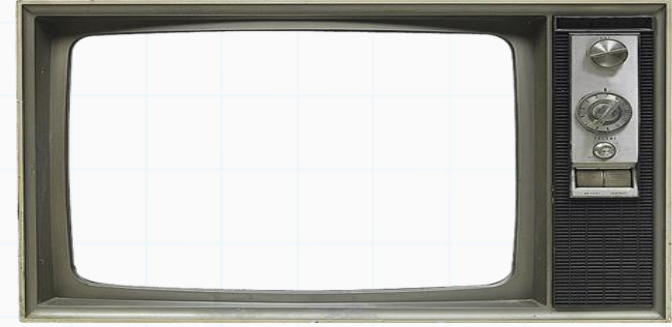
- Ou usando valor de variável

```
1 def calcula_tempo(velocidade, distancia):
2     tempo = distancia/velocidade
3     print(v)
4     return tempo
5
6 v = 10
7 d = 5
8 t = calcula_tempo(v, d)
9 print(t)
```



Subrotinas e Funções

Passagem de Parâmetros: Os valores são copiados (de forma ordenada) para a variável que representa o parâmetro na função



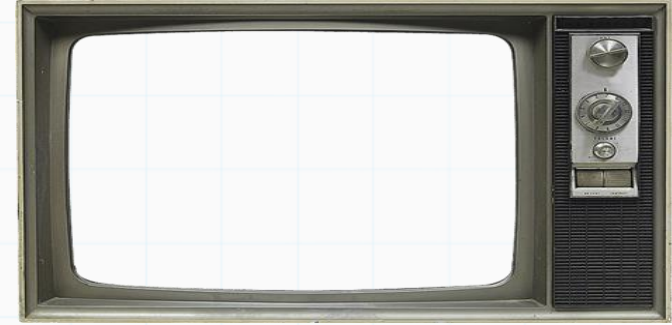
```
1 def calcula_tempo(velocidade, distancia):  
2     tempo = distancia/velocidade  
3     print(v)  
4     return tempo  
5  
6 v = 10  
7 d = 5  
8 t = calcula_tempo(v, d)  
9 print(t)
```



v	10
d	5
t	

Subrotinas e Funções

Passagem de Parâmetros: Os valores são copiados (de forma ordenada) para a variável que representa o parâmetro na função



→

```
1 def calcula_tempo(velocidade, distancia):  
2     tempo = distancia/velocidade  
3     print(v)  
4     return tempo  
5  
6 v = 10  
7 d = 5  
8 t = calcula_tempo(v, d)  
9 print(t)
```

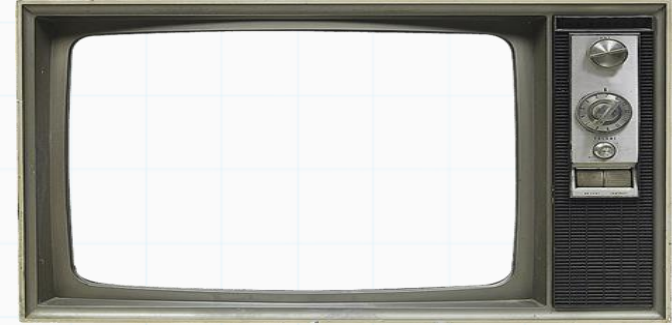
Cópia

velocidade	10
distância	5
tempo	

v	10
d	5
t	

Subrotinas e Funções

Passagem de Parâmetros: Os valores são copiados (de forma ordenada) para a variável que representa o parâmetro na função



```
1 def calcula_tempo(velocidade, distancia):  
2     tempo = distancia/velocidade  
3     print(v)  
4     return tempo  
5  
6 v = 10  
7 d = 5  
8 t = calcula_tempo(v, d)  
9 print(t)
```

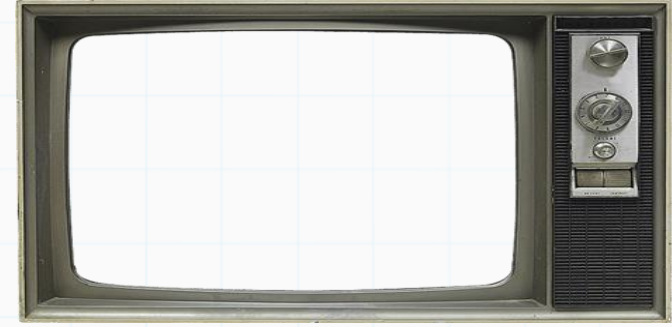
velocidade	10
distância	5
tempo	0.5

v	10
d	5
t	



Subrotinas e Funções

Passagem de Parâmetros: Os valores são copiados (de forma ordenada) para a variável que representa o parâmetro na função



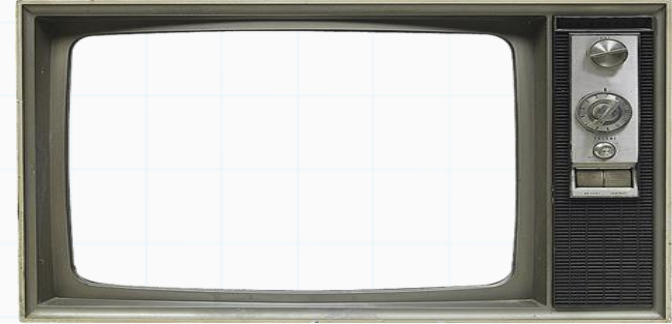
```
1 def calcula_tempo(velocidade, distancia):  
2     tempo = distancia/velocidade  
3     print(v)  
4     return tempo  
5  
6 v = 10  
7 d = 5  
8 t = calcula_tempo(v, d)  
9 print(t)
```

Cópia

v	10
d	5
t	0.5

Subrotinas e Funções

Passagem de Parâmetros: Alterações no valor parâmetro não são refletidas na variável correspondente àquele parâmetro no programa principal



Passagem de Parâmetros por valor:
Os valores são copiados

```
1 def calcula_tempo(velocidade, distancia):
2     tempo = distancia/velocidade
3     distancia = 0
4     velocidade = 0
5     return tempo
6
7 v = 10
8 d = 5
9 t = calcula_tempo(v, d)
10 print(v,d,t)
```

Shell ×

Python 3.7.7 (bundled)

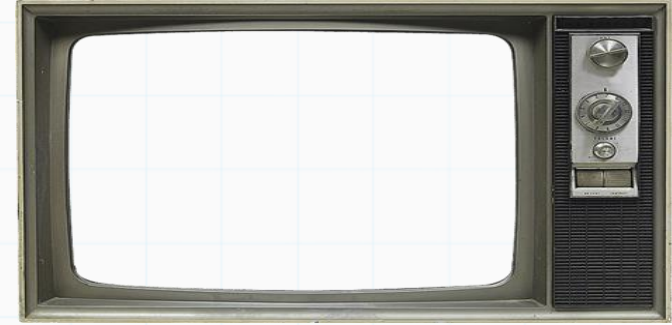
>>> %Run teste.py

10 5 0.5

[código](#)

Subrotinas e Funções

Passagem de Parâmetros: A exceção são os vetores e matrizes (lista, dicionários, etc.)



```
1 def soma_um(vet):  
2     for i in range(len(vet)):  
3         vet[i] = vet[i]+1  
4  
5 v = [0,5,10,15]  
6 soma_um(v)  
7 print(v)
```

Shell x

```
>>> %run teste.py
```

```
[1, 6, 11, 16]
```

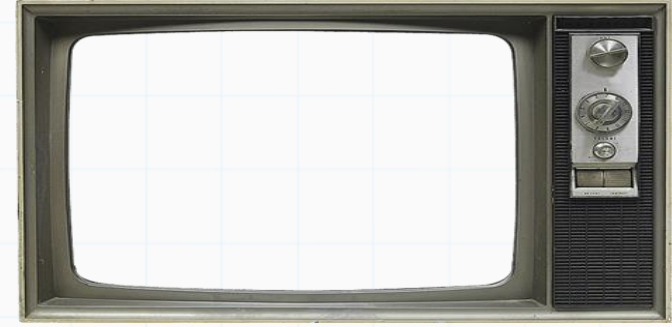
[código](#)

Passagem de Parâmetros por referência:
São passados o endereço de memória, por isso altera

Subrotinas e Funções

Uso de variáveis globais:

- Variáveis globais podem ser acessadas dentro de uma função, mas normalmente não podem ser alteradas dentro das funções.



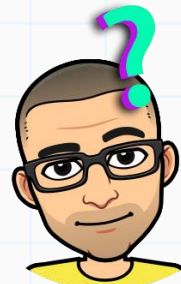
```
1 def maior():
2     if a > b:
3         m = a
4     else:
5         m = b
6
7 m = 0
8 a = 1
9 b = 2
10 maior()
11 print(m)
```

acessar variáveis globais dentro da função é uma prática ruim

Shell x

Python 3.7.7 (bundle
>>> %Run teste.py

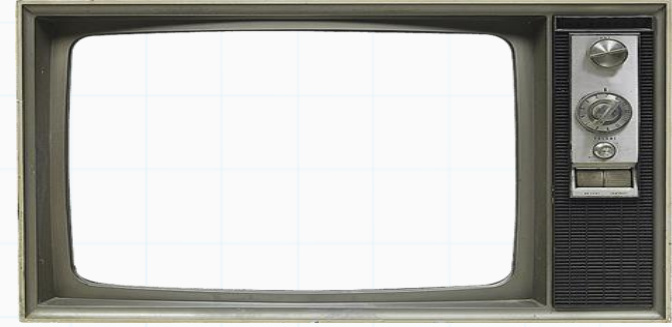
0



Subrotinas e Funções

Uso de variáveis globais:

- Variáveis globais podem ser acessadas dentro de uma função, mas normalmente não podem ser alteradas dentro das funções.



```
1 def maior():
2     if a > b:
3         m = a
4     else:
5         m = b
6
7 m = 0
8 a = 1
9 b = 2
10 maior()
11 print(m)
```

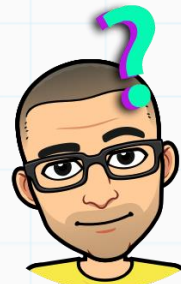
acessar variáveis globais dentro da função é uma prática ruim

Shell ×

Python 3.7.7 (bundle

>>> %Run teste.py

0



Como não se pode mudar o valor de uma variável global dentro da função, o PYTHON cria uma variável local com o mesmo nome

Subrotinas e Funções

Uso de variáveis globais:

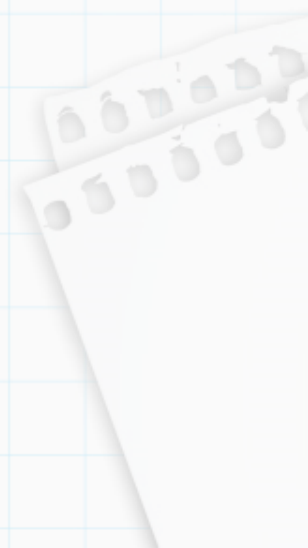
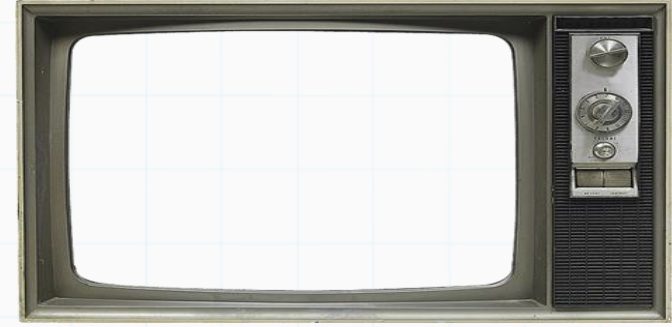
- Para alterar a variável global temos que usar o comando global

```
1 def maior():  
2     global m  
3     if a > b:  
4         m = a  
5     else:  
6         m = b  
7  
8 m = 0  
9 a = 1  
10 b = 2  
11 maior()  
12 print(m)
```

Shell x

2

[código](#)



Subrotinas e Funções

Uso de variáveis globais:

- Para alterar a variável global temos que usar o comando global

```
1 def maior():  
2     global m  
3     if a > b:  
4         m = a  
5     else:  
6         m = b  
7  
8 m = 0  
9 a = 1  
10 b = 2  
11 maior()  
12 print(m)
```

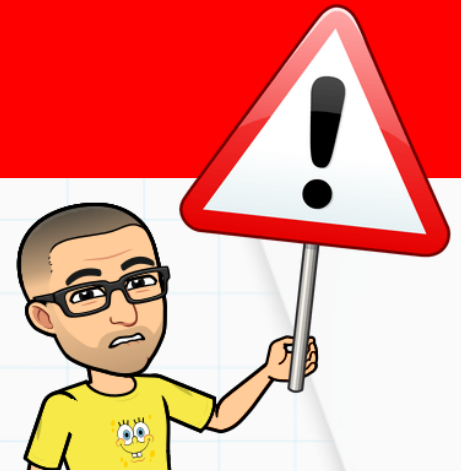
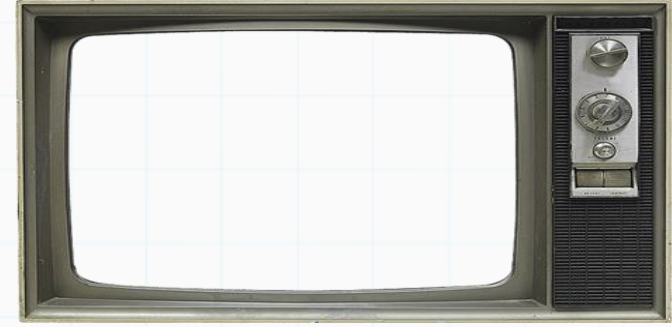
Shell x

2

[código](#)

acessar variáveis globais dentro da função é uma prática ruim

ALTERAR variáveis globais dentro da função é uma prática MUITO ruim



Subrotinas e Funções

Uso de variáveis globais:

- A forma mais segura seria não usando variáveis globais e sim parâmetros.

```
1 def maior():
2     global m
3     if a > b:
4         m = a
5     else:
6         m = b
7
8 m = 0
9 a = 1
10 b = 2
11 maior()
12 print(m)
```

Shell ×

~/run teste.py

2

[código](#)

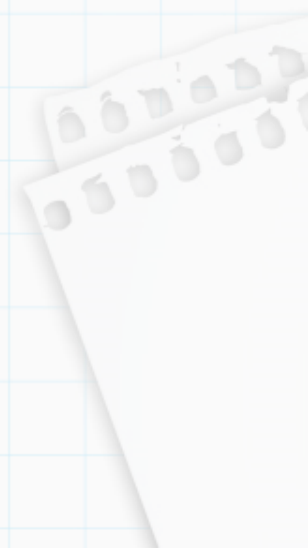
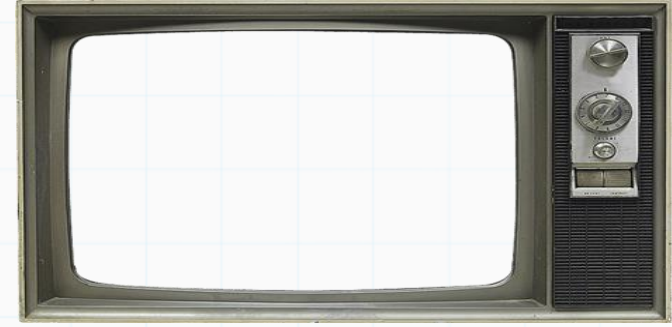
```
1 def maior(x,y):
2     if x > y:
3         return x
4     else:
5         return y
6
7 m = 0
8 a = 1
9 b = 2
10 m = maior(a,b)
11 print(m)
```

Shell ×

~/run 22 - ex10.py

2

[código](#)

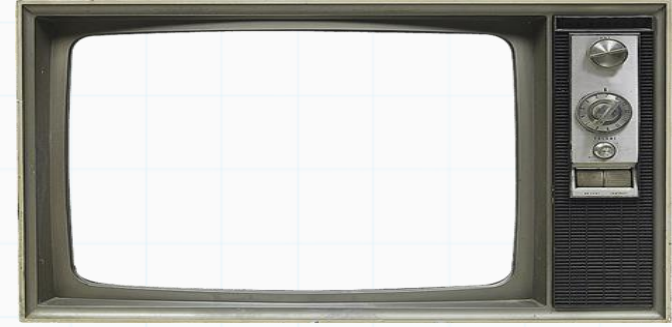


Subrotinas e Funções

Exemplo de uso de funções: O que seria impresso ?

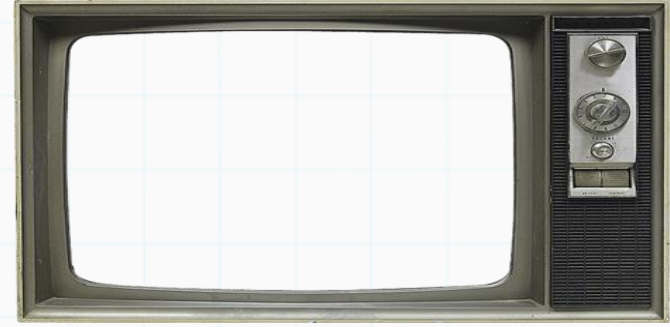
```
1 def func():  
2     x = 0  
3     y = 0  
4  
5     x = 3  
6     y = 4  
7     func()  
8     print(x,y)
```

[código](#)



Subrotinas e Funções

Exemplo de uso de funções: O que seria impresso ?



```
1 def func():  
2     x = 0  
3     y = 0  
4  
5     x = 3  
6     y = 4  
7     func()  
8     print(x,y)
```

[código](#)

variáveis locais com mesmo
nome das globais

```
>>> %Run teste.py
```

```
3 4
```

```
>>>
```

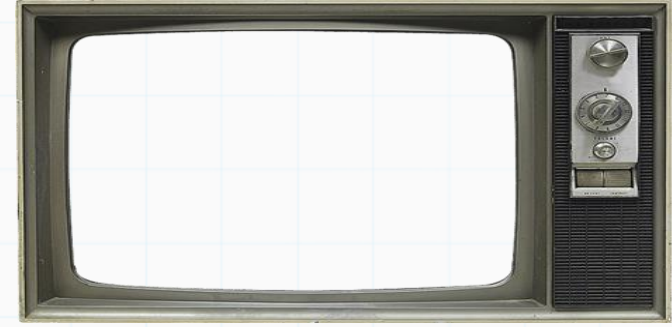
```
.
```

Subrotinas e Funções

Exemplo de uso de funções: O que seria impresso ?

```
1 def func(a):  
2     a = 10  
3  
4 a = 1  
5 func(a)  
6 print(a)  
7
```

[código](#)



Subrotinas e Funções

Exemplo de uso de funções: O que seria impresso ?

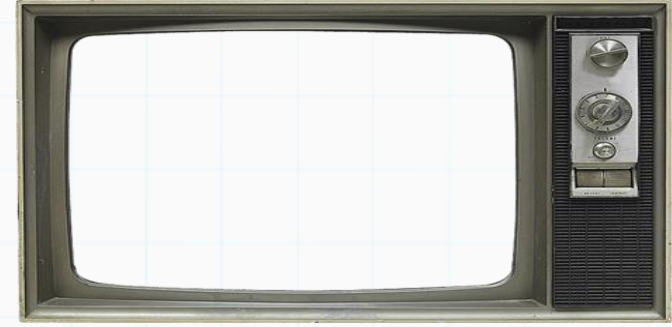
```
1 def func(a):  
2     a = 10  
3  
4 a = 1  
5 func(a)  
6 print(a)  
7
```

[código](#)

Shell ×

1

parâmetro é variável local
com mesmo nome da variável
global

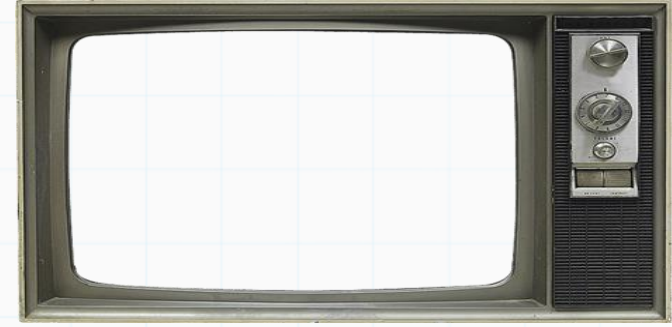


Subrotinas e Funções

Exemplo de uso de funções: O que seria impresso ?

```
1 def func(a,b):  
2     a     = 0  
3     b[0]  = 20  
4  
5 x = 1  
6 y = [2,3]  
7 func(x,y)  
8 print(x,y)
```

[código](#)



Subrotinas e Funções

Exemplo de uso de funções: O que seria impresso ?

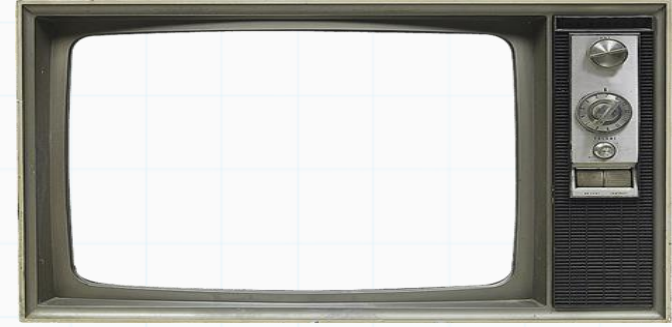
```
1 def func(a,b):  
2     a     = 0  
3     b[0]  = 20  
4  
5 x = 1  
6 y = [2,3]  
7 func(x,y)  
8 print(x,y)
```

[código](#)

Shell ×

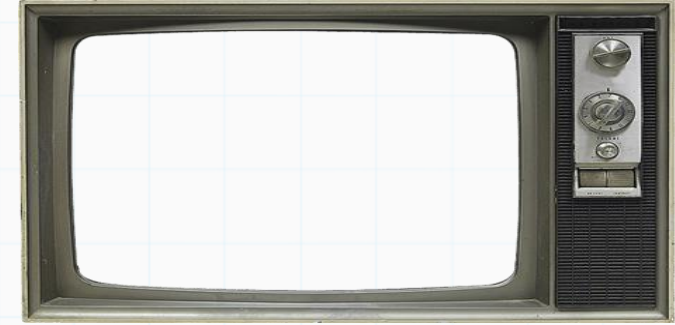
```
1 [20, 3]
```

vetores são alterados nas
funções



Subrotinas e Funções

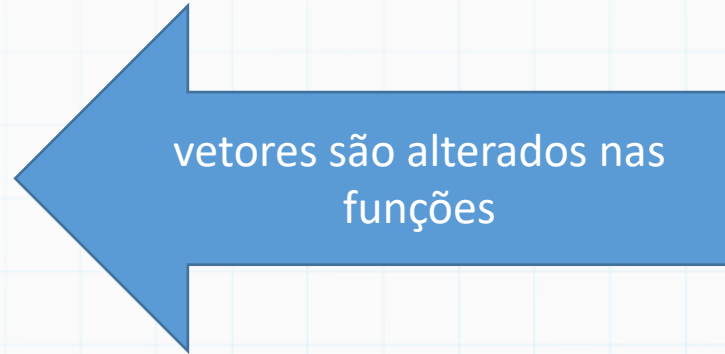
Exemplo de uso de funções: O que seria impresso ?



```
1 def func(a,b):  
2     a     = 0  
3     b[0]  = 20  
4  
5 x = 1  
6 y = [2,3]  
7 func(x,y)  
8 print(x,y)
```

[código](#)

```
Shell ×  
1 [20, 3]
```



vetores são alterados nas
funções



```
1 def func():  
2     y[0]=10  
3  
4 x = 1  
5 y = [2,3]  
6 func()  
7 print(x,y)
```

```
Shell ×  
1 [10, 3]
```

mesmo se não forem
passados por parâmetros.

ALTERAR variáveis
globais dentro da
função é uma prática
MUITO ruim



Subrotinas e Funções

Fura Olho: O que será escrito ?

```
1 def change(a,b):
2     c=b
3     b=a+b
4     a=c
5     return a,b
6
7 a,b,c=1,2,3
8 print(a,b,c)
9 a,b=change(a,b)
10 print(a,b,c)
```



```
1 def t1(jj):
2     jj=1
3     t2(jj)
4     return jj
```

```
6 def t2(jj):
7     jj=2
8     t3(jj)
9     jj=4
10    return jj
```

```
12 def t3(jj):
13     jj=3
14     return jj
```

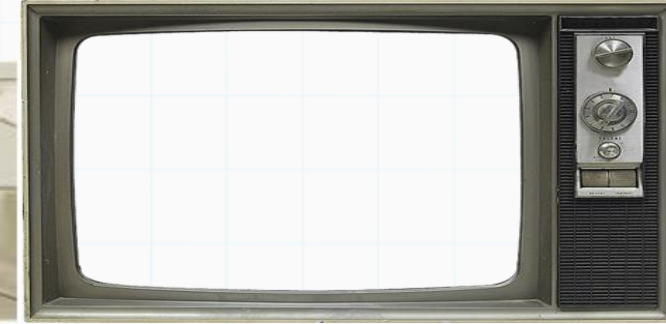
```
16 jj=1
17 t1(jj)
18 print(jj)
```

```
1 # Entrada 5, 3
2 def k(h,y):
3     h=int(input())
4     y=int(input())
5     h=h//(y+h)
6     y=h+3
7     x=y+2//3
8     print(h,y,x)
9     return h,y
```

```
11 h=2
12 y=0
13 x=0
14 x,y=k(x,y)
15 print(h,y,x)
```

Subrotinas e Funções

Fura Olho: O que será escrito ?



```
1 def change(a,b):  
2     c=b  
3     b=a+b  
4     a=c  
5     return a,b  
6  
7 a,b,c=1,2,3  
8 print(a,b,c)  
9 a,b=change(a,b)  
10 print(a,b,c)
```

```
1 def t1(jj):  
2     jj=1  
3     t2(jj)  
4     return jj  
5  
6 def t2(jj):  
7     jj=2  
8     t3(jj)  
9     jj=4  
10    return jj  
11  
12 def t3(jj):  
13     jj=3  
14     return jj  
15  
16 jj=1  
17 t1(jj)  
18 print(jj)
```

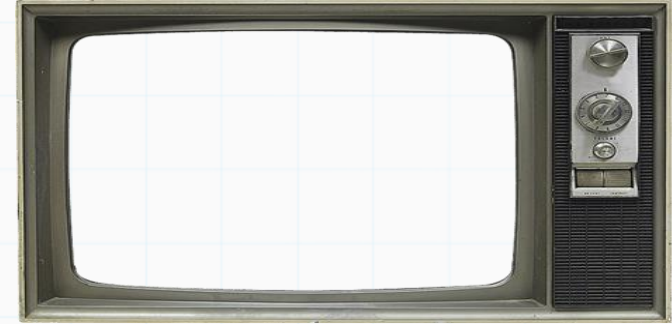
```
1 # Entrada 5, 3  
2 def k(h,y):  
3     h=int(input())  
4     y=int(input())  
5     h=h//(y+h)  
6     y=h+3  
7     x=y+2//3  
8     print(h,y,x)  
9     return h,y  
10  
11 h=2  
12 y=0  
13 x=0  
14 x,y=k(x,y)  
15 print(h,y,x)
```

Subrotinas e Funções

Fura Olho: O que será escrito ?

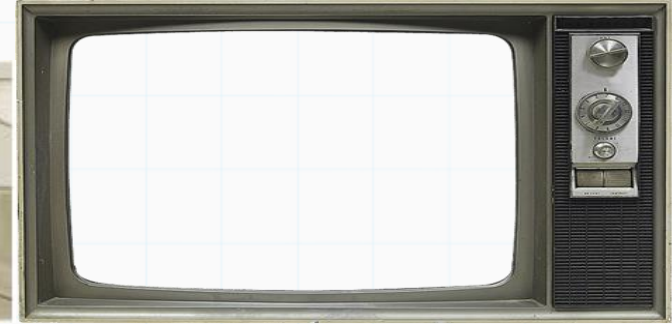
```
1 def p(a):
2     a=0
3     return a
4
5 def g(w):
6     a=(w+1)**2
7     p(w)
8     return w
9
10 def f(a):
11     b=a/10
12     a=a+1
13     a=g(a)
14     return a,b
```

```
16 def s(x,y):
17     i=1
18     while (i <= x+y):
19         i,ii=f(i)
20         print(ii)
21         i=i+1
22     print(i)
23     x=x-y
24     return x,y
25
26 x=2
27 y=3
28 print(x)
29 y,x=s(y,x)
30 print(y)
```



Subrotinas e Funções

Fura Olho: O que será escrito ?

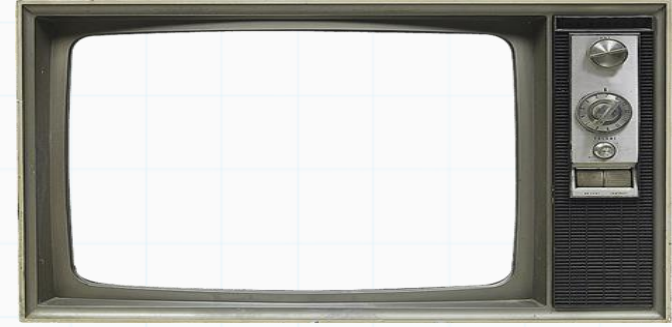


```
1 def p(a):  
2     a=0  
3     return a  
4  
5 def g(w):  
6     a=(w+1)**2  
7     p(w)  
8     return w  
9  
10 def f(a):  
11     b=a/10  
12     a=a+1  
13     a=g(a)  
14     return a,b
```

```
16 def s(x,y):  
17     i=1  
18     while (i <= x+y):  
19         i,ii=f(i)  
20         print(ii)  
21         i=i+1  
22     print(i)  
23     x=x-y  
24     return x,y  
25  
26 x=2  
27 y=3  
28 print(x)  
29 y,x=s(y,x)  
30 print(y)
```

2
0.1
0.3
0.5
7
1

Até a próxima



Slides baseados no curso de Vanessa Braganholo

