Introdução à programação para ciência e engenharia em *Python*

Iuri Soter Viana Segtovich

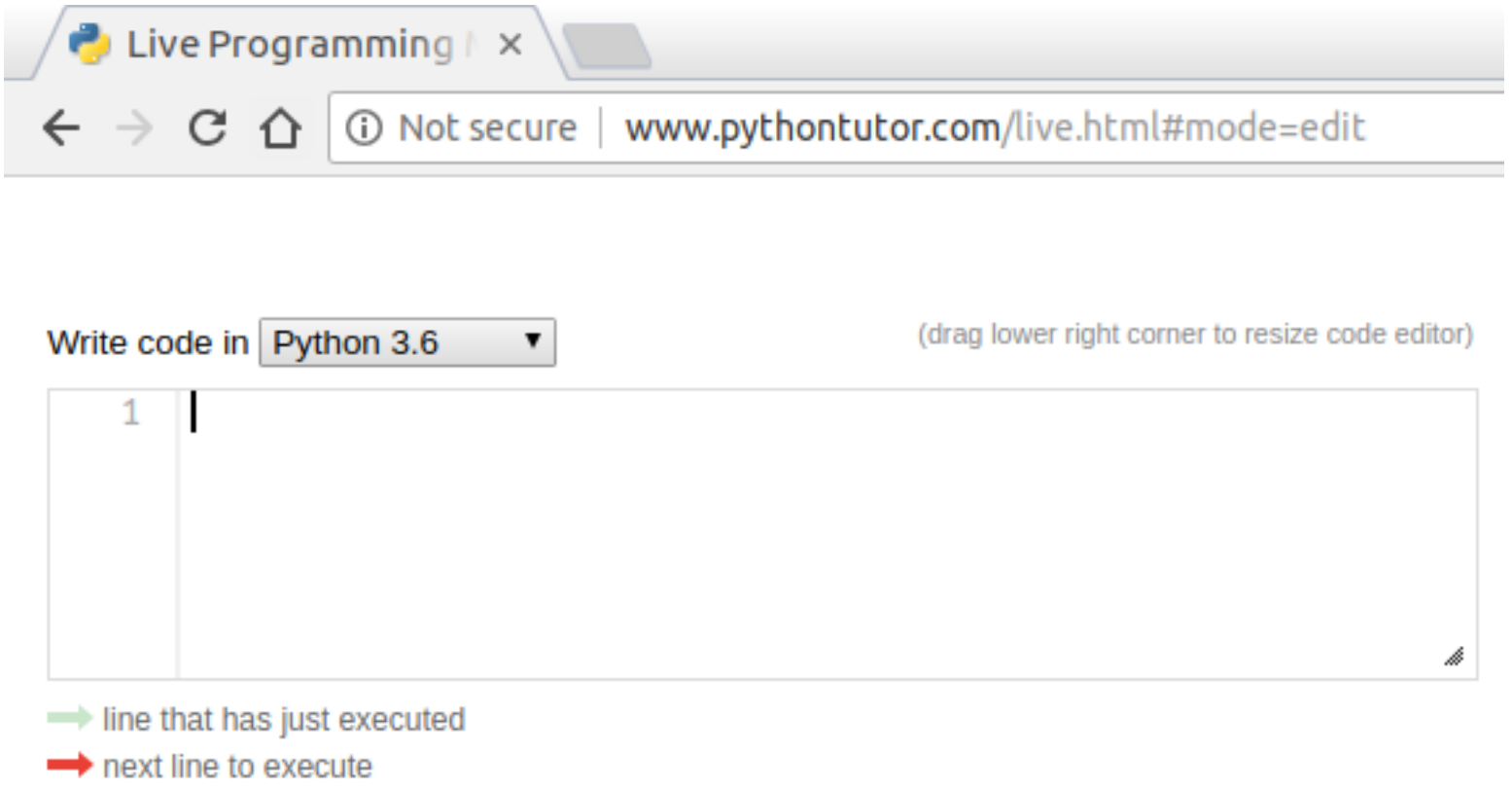
Parte 2: Lógica e Sintaxe

Funções (`def`, `return`, `*args`, `*kargs`, `lambda`)

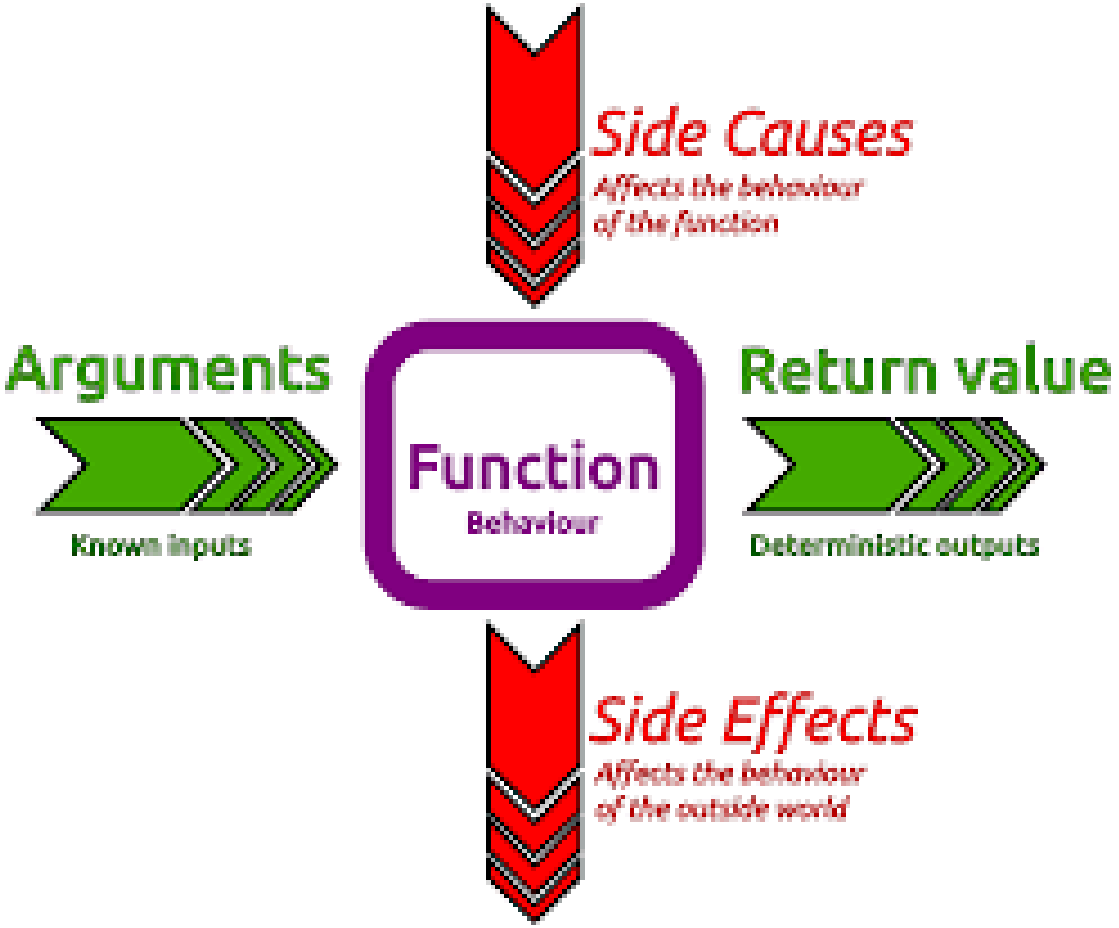
python tutor

www.pythontutor.com/

live.html#mode=edit



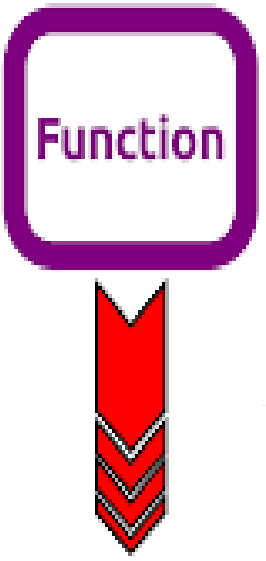
O caso geral das funções



Def funções

```
1  def diga_oi():
2      print("oi")
3
4  diga_oi()
```

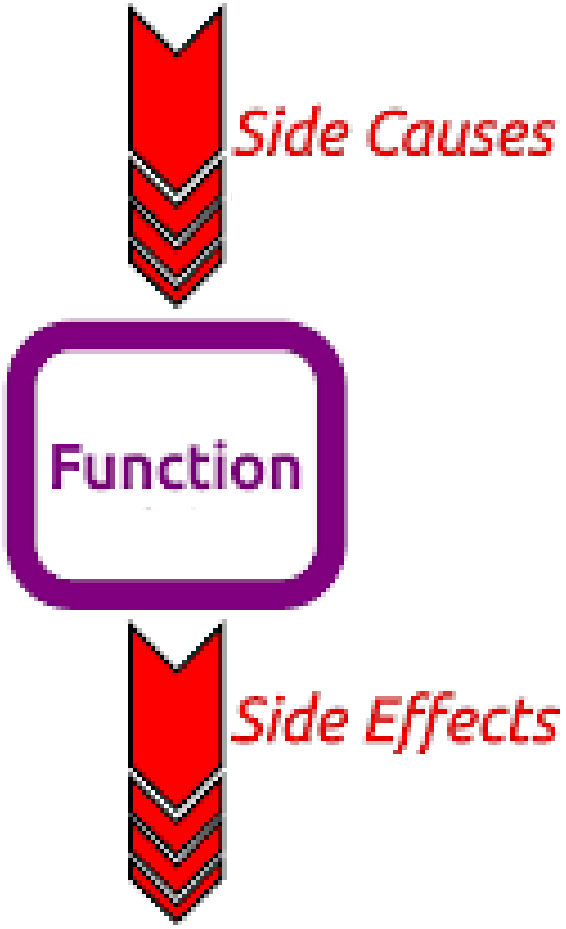
```
oi
```



- Def, Parênteses vazios - definição
- Dois pontos
- Identação
- Parênteses vazios - chamada

Side causes

```
1  x=1
2
3  def leia_x():
4      print("x é", x)
5
6  leia_x()
```

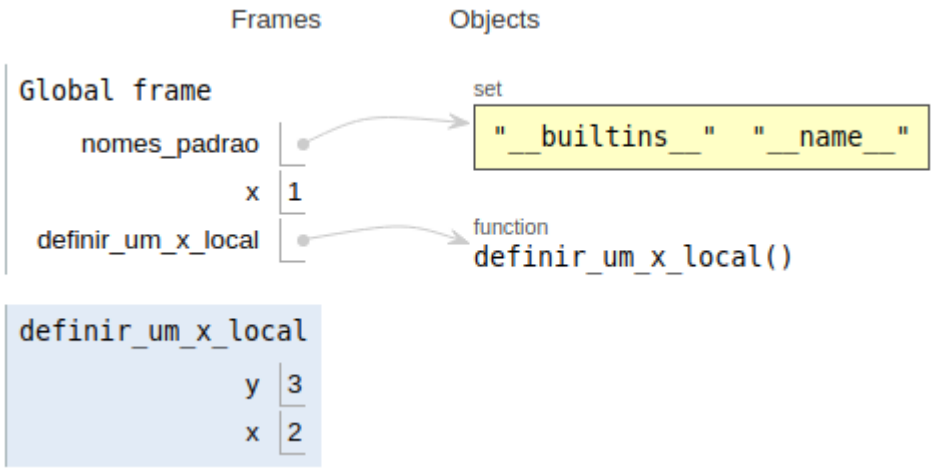


- Acesso de leitura às globais

Namespaces - função dir()

```
1  nomes_padrao=set(dir())
2  x=1
3  print("nomes no namepace externo", set(dir())-nomes_padrao)
4  def definir_um_x_local():
5      y=3 #definindo um nome y, localmente
6      x=2 #definindo um nome x, localmente
7      print("x do namespace interno é", x)
8      print("nomes no namepace interno",set(dir())-nomes_padrao)
9
10 print("x do namespace externo é", x)
11 definir_um_x_local()
12 print("x do namespace externo ainda é", x)
13 print("nomes no namepace externo",set(dir())-nomes_padrao)
```

```
nomes no namepace externo {'nomes_padrao', 'x'}
x do namespace externo é 1
x do namespace interno é 2
nomes no namepace interno {'x', 'y'}
x do namespace externo ainda é 1
nomes no namepace externo {'nomes_padrao', 'x', 'definir_um_x_local'}
```



global

```
x do namespace externo é 1
x do namespace externo é 1
x é 1
[]
x do namespace externo é 1
```

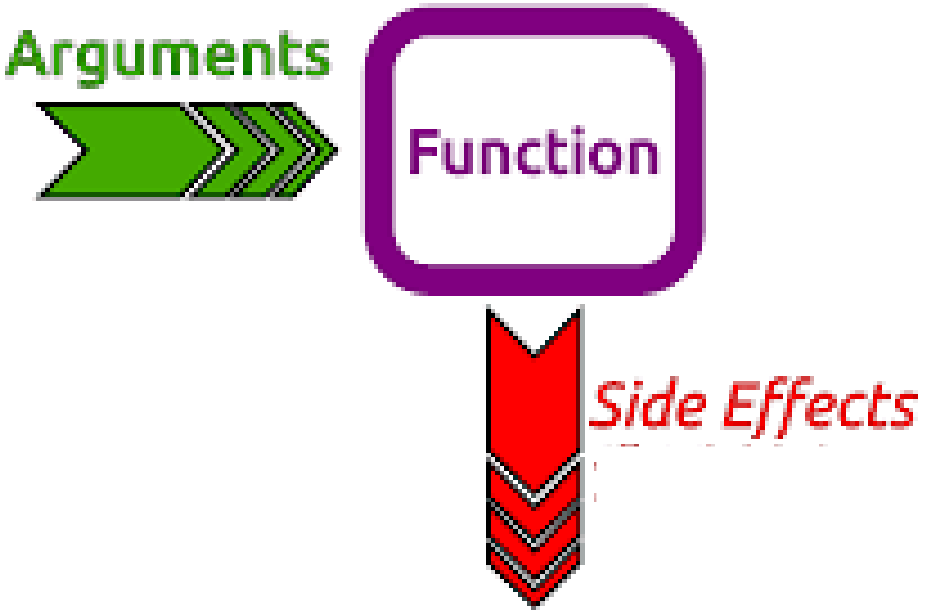
```
1  x=1
2
3  def mude_x():
4      global x
5      print("x do namespace externo é", x)
6      print("x é", x)
7      print(dir())
8
9  print("x do namespace externo é", x)
10 mude_x()
11 print("x do namespace externo é", x)
```

- Acesso de modificação
- (Rebinding) das globais

argumentos

```
1  def converte_m_para_cm(x_in):
2      x_out = 100.*x_in
3      print(x_out)
4
5
6  converte_m_para_cm(5.)
```

```
500.0
```



- argumentos posicionais

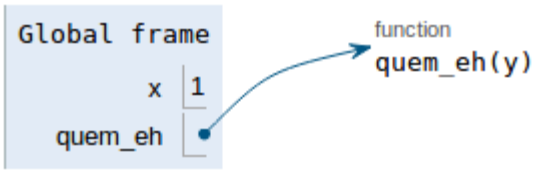
nome *dummy* x nome *actual*

```
1  x_actual=1
2
3  def f(x_dummy):
4      print(x_dummy)
5
6  f(x_actual)
7
8  #o nome x_dummy passa a referenciar
9  #aquele objeto ao qual o nome x_actual dava nome
10 #enquanto estivermos dentro da função
11
12 f(x_dummy=x_actual) #chamada com palavra_chave
```

```
1
1
```

Passa referência para um objeto único, não passa o nome

```
1  x=1
2  print(id(x))
3
4  def quem_eh(y):
5      print(id(y)) #outro nome, mesmo objeto
6
7  quem_eh(x)
8
```




```
1  x=1
2  print(id(x))
3
4  def quem_eh(y):
5      print(id(x))
6      print(id(y)) #outro nome, mesmo objeto
7      y+=1
8      print(id(x)) # x continua dando nome ao objeto 1
9      print(id(y)) # y dá nome a um novo objeto
10
11 quem_eh(x)
12
```

```
10538048
10538048
10538048
10538048
10538080
```

Se o argumento é objeto mutavel

```
1  x=[1]
2  print(id(x))
3
4  def quem_eh(y):
5      print(id(x))
6      print(id(y)) #outro nome, mesmo objeto
7      y[0]+=1
8      print(id(x)) # x continua dando nome ao objeto 1
9      print(id(y)) # y dá nome ao mesmo objeto, o qual foi modificado
10
11 quem_eh(x)
12
```

```
139666956102472
139666956102472
139666956102472
139666956102472
139666956102472
```

- É possivel mudar o conteúdo sem desvincular o nome

mais args menos global

```
 1  a=5
 2  b=99
 3  l=[1,2,3]
 4
 5  def f():
 6      print(gx**2)
 7
 8  gx=a
 9  f()
10
11  gx=a+b
12  f()
13
14  for i in l:
15      gx=i
16      f()
```

```
 1  a=5
 2  b=99
 3  l=[1,2,3]
 4
 5  def f(x):
 6      print(x**2)
 7
 8  f(a)
 9  f(a+b)
10  for i in l:
11      f(i)
```

return

```
1  def converte_m_para_cm(x_in):
2      x_out = 100.*x_in
3      return x_out
4
5  medida_m=5.
6  medida_cm = converte_m_para_cm(medida_m)
7  print(medida_cm)
```

```
500.0
```

- Return objetos ou objetos

Exemplo: função de conversão de unidades de temperatura de F para C e vice-versa.

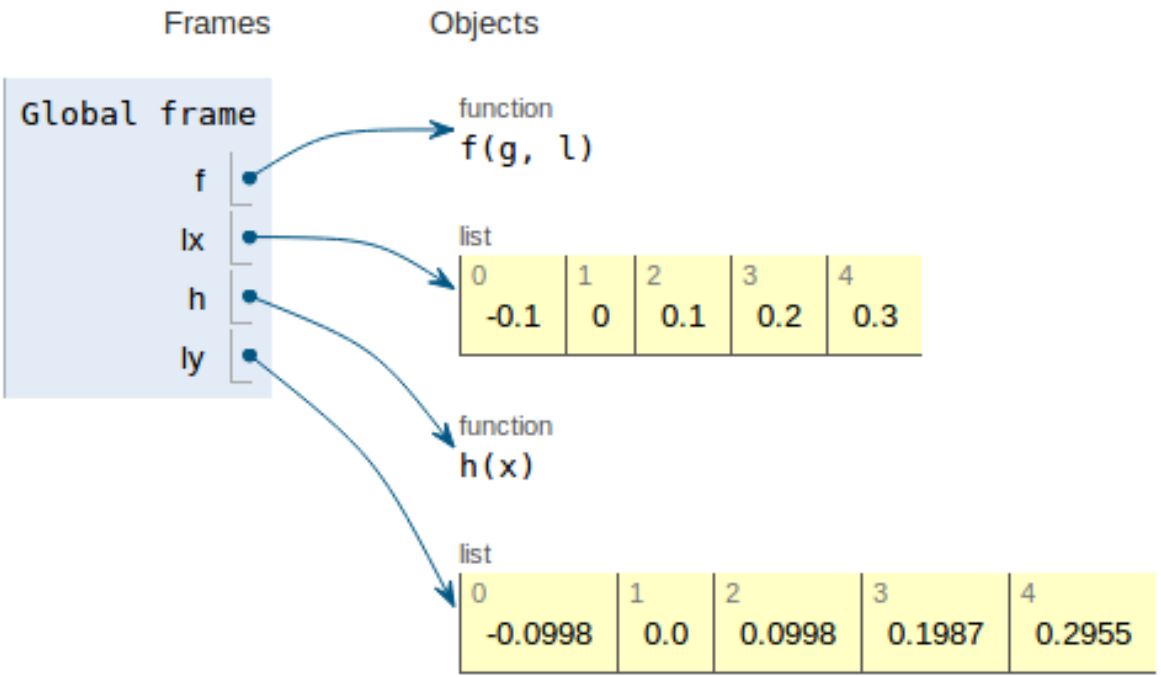
$$F = 1,8C + 32$$

Função f usando função g

```
1  def converte_m_para_cm(x_in):
2      x_out = 100.*x_in
3      return x_out
4
5  medida_m=5.
6  medida_cm = converte_m_para_cm(medida_m)
7  print(medida_cm)
8
9  def area_quadrado(l_m):
10     l_cm=converte_m_para_cm(l_m)
11     area_cm2=l_cm*l_cm
12     return area_cm2
13
14 lado_m=2.
15 area_cm2=area_quadrado(lado_m)
```

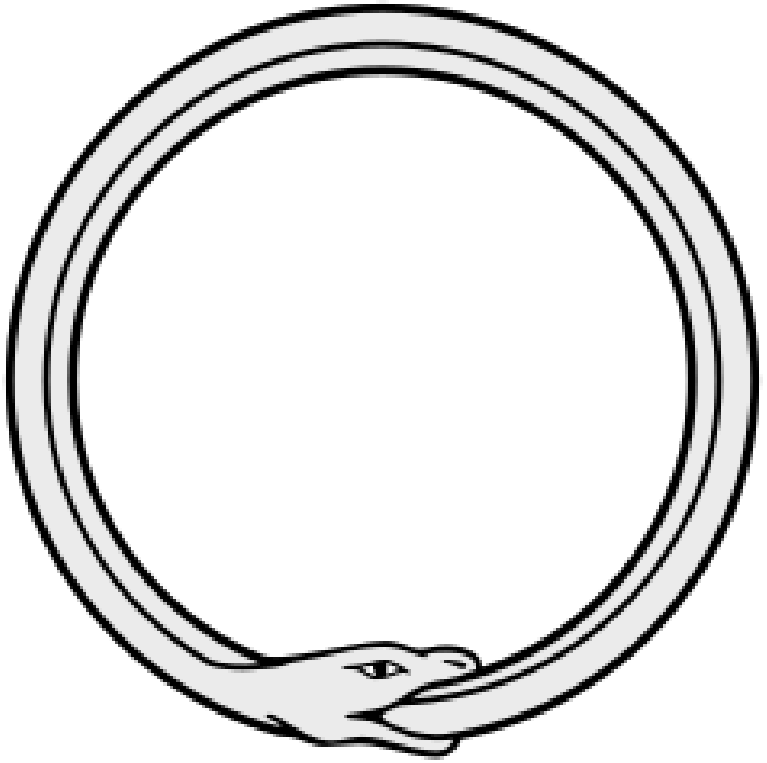
Função f recebendo uma função g por argumento

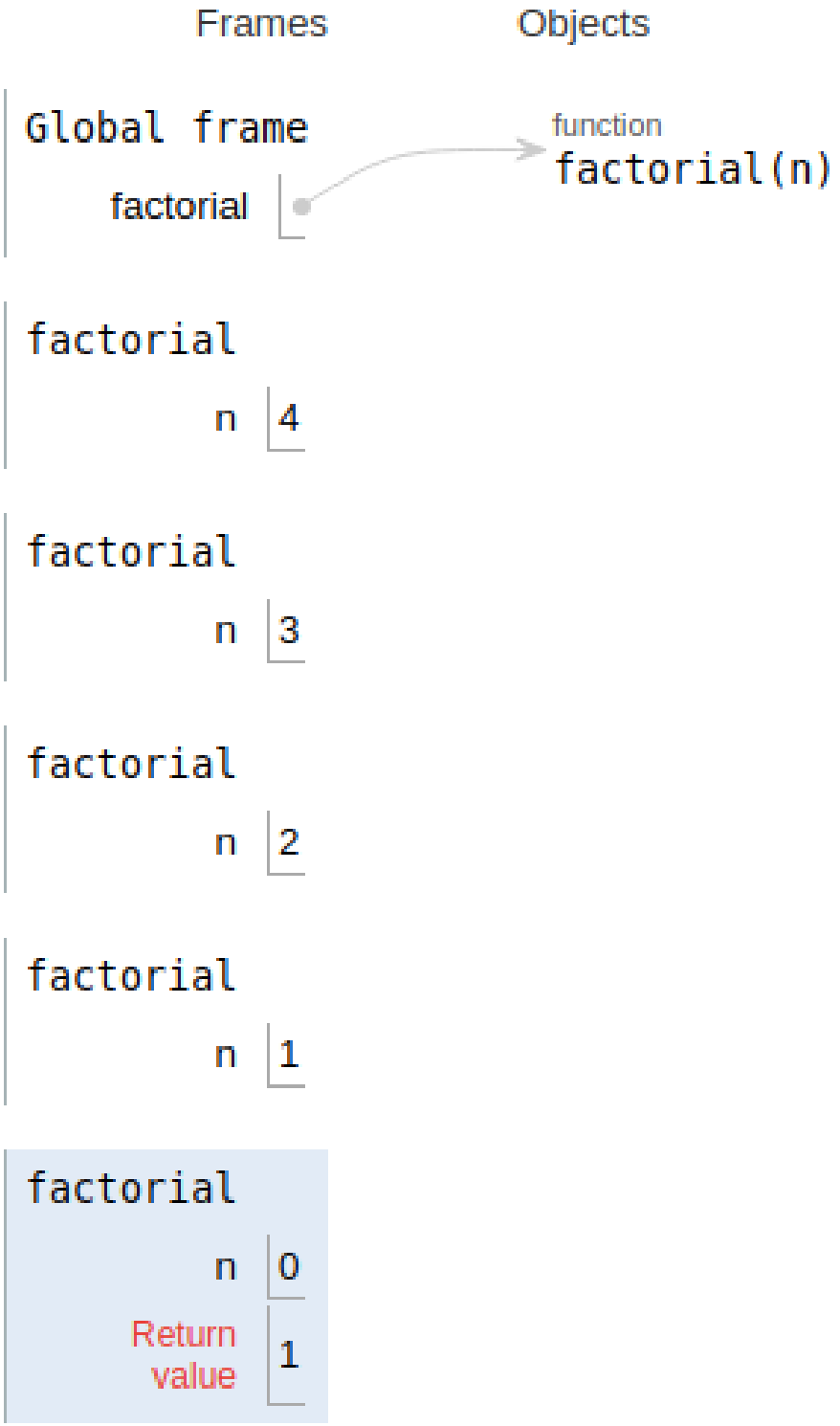
```
def f(g,l):
    fl=[]
    for li in l:
        fl.append(g(li))
    return fl

lx=[-.1,0,.1,.2,.3]

def h(x):
    from math import sin
    return sin(x)

ly=f(h,lx)
```



Função recursiva

```
def factorial( n ):
    if n <1:    # base case
        return 1
    else:                # recursive call
        return n * factorial( n - 1 )
```



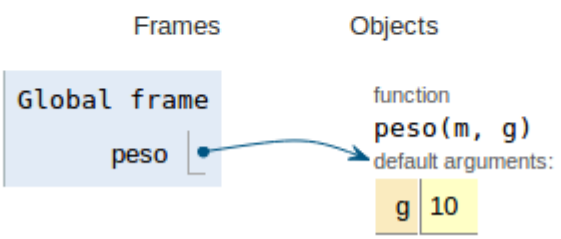


Opcionais e kargs

```
1  def peso(m,g=10):
2      p=m*g
3      return p
4
5  print(peso(100))
6  print(peso(100,g=9.8))
7  print(peso(100,g=3.711)) # em marte
8
```

Print output (drag lower right corner to resize)

```
1000
980.0000000000001
371.09999999999997
```



- kargs, argumentos palavra chave

closure

```
5  R=8.314
6
7  def v(n,T,P):
8      return n*R*T/P
9
10 Tamb=298.15
11 Patm=1e5
12 def v_amb_atm(n):
13     return v(n,T=Tamb,P=Patm)
14
15 print( v_amb_atm(3.) )
16
17
```

```
0.07436457299999999
```

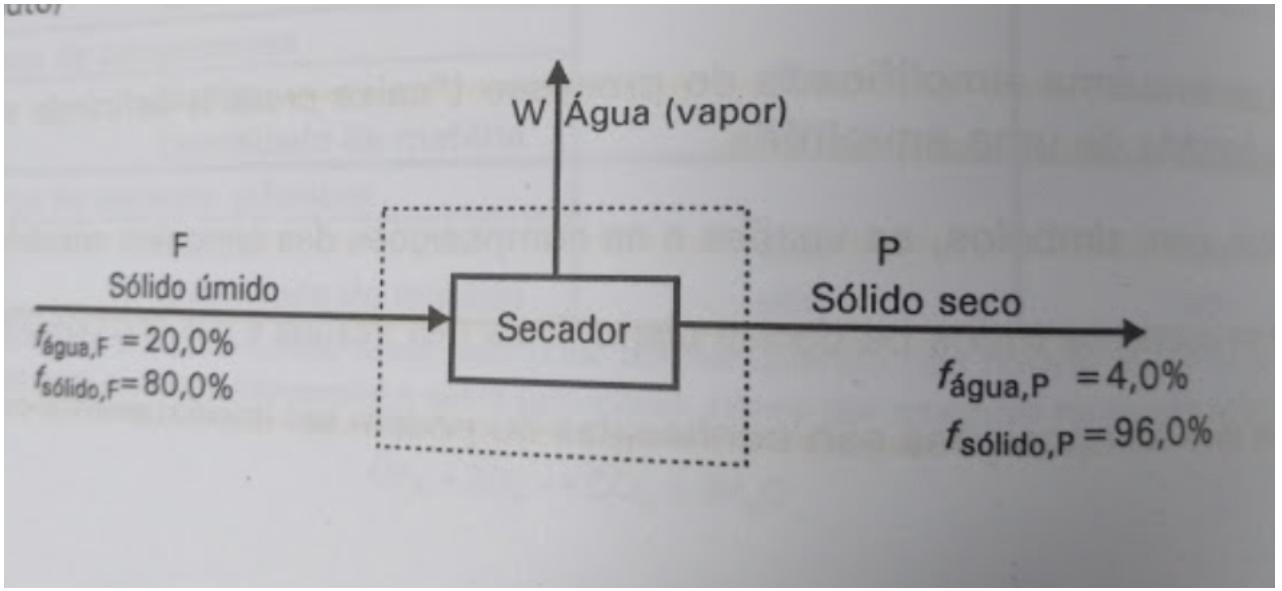
- Pegar uma função de 3 argumentos e gerar uma função de 1 argumento, fixando os demais.

lambda

```
10  v_amb_atm = lambda n: v(n,Tamb,Patm)
11
12  print(v_amb_atm(3.))
13
```

- lambda
  - nome_da_dummy
  - Dois pontos
  - Chamada de função usando
    - o nome da dummy
    - nomes definidos no namespace

Exercício



$$P = F\frac{x_s^f}{x_s^p}$$

$$W = Fx_w^f - Px_w^p$$

$$R\% = 100\frac{W}{Fx_w^f}$$

```python
def CalcRecSecador(xfs,xfw,xps,xpw):
    #base de cálculo
    F=100
    #bm sólido
    P=F*xfs/xps
    W=F*xfw-P*xpw
    Rec=W/(F*xfw)*100
    return P,W,Rec

P, W, Rec = CalcRecSecador(.8,.2,.96,.04)

print("P, W, Rec %")
print(P, W, Rec)
```

```
P, W, Rec %
83.33333333333334 16.666666666666664 83.33333333333333
```

Referências principais

https://www.tutorialspoint.com/

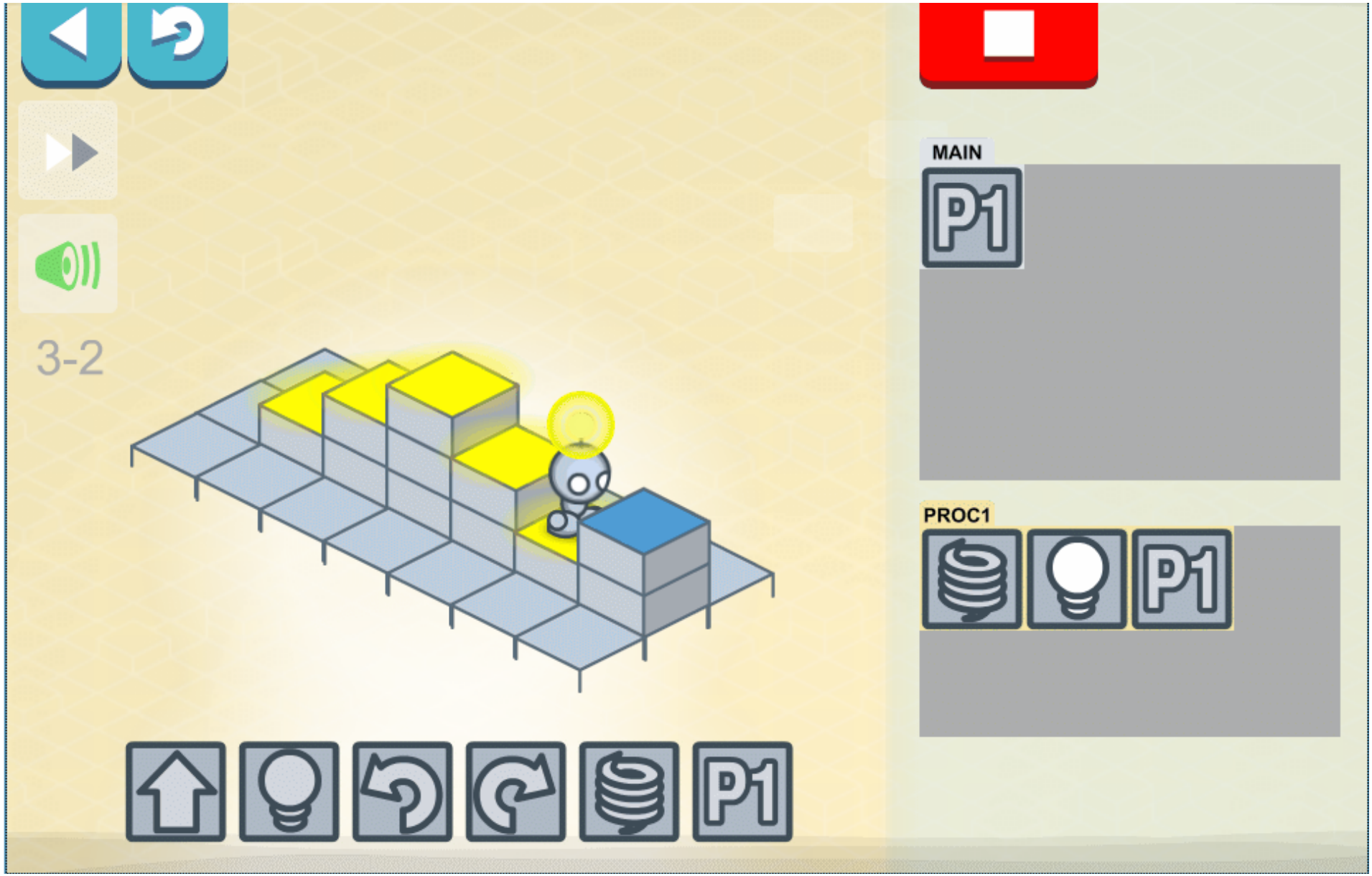
python3/

python_basic_syntax.htm

https://stackoverflow.com/

search

EXERCÍCIO DE lógica DE PROGRAMAÇÃO



perguntas