



86ª EDIÇÃO

**SEQ
UFRJ**

26 a 30 de agosto



Introdução à programação para ciência e engenharia em *Python*

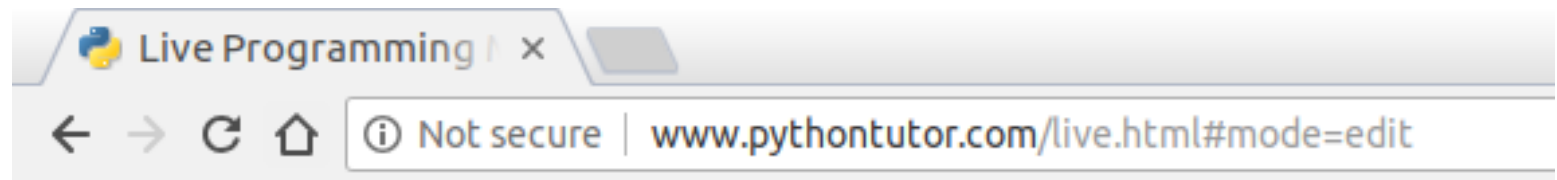
Iuri Segtovich

Parte 2: Lógica e Sintaxe

Tipos coleção e iterações (tuple, list, set, dict, for, while)

python tutor

[www.pythontutor.com/
live.html#mode=edit](http://www.pythontutor.com/live.html#mode=edit)



Write code in Python 3.6

(drag lower right corner to resize code editor)

```
1 |
```

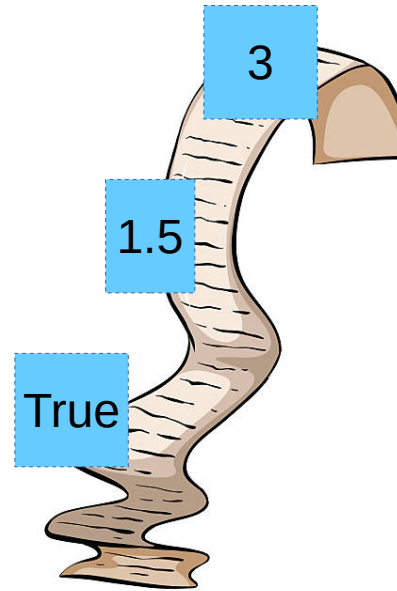
→ line that has just executed

→ next line to execute

mais tipos

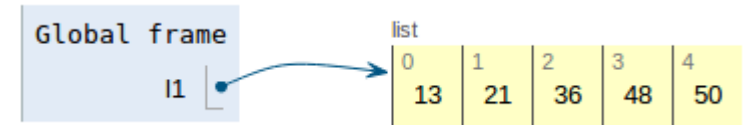
Containers / Collections

- list



list

```
→ 1 | l1 = [13, 21, 36, 48, 50]
```

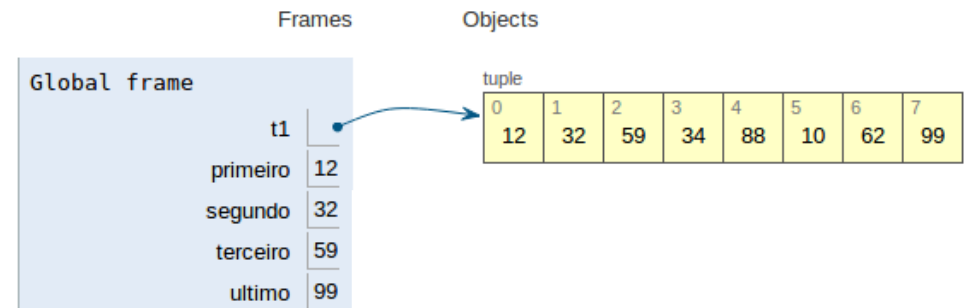


- Expressão literal – vírgulas e colchetes

Indexing

acessar elementos por índice

```
1 t1 = (12, 32, 59, 34, 88, 10, 62, 99)
2 primeiro = t1[0]
3 segundo = t1[1]
4 terceiro = t1[2]
5
6 ultimo = t1[-1]
7
```



- Começa no zero
- -1 significa último

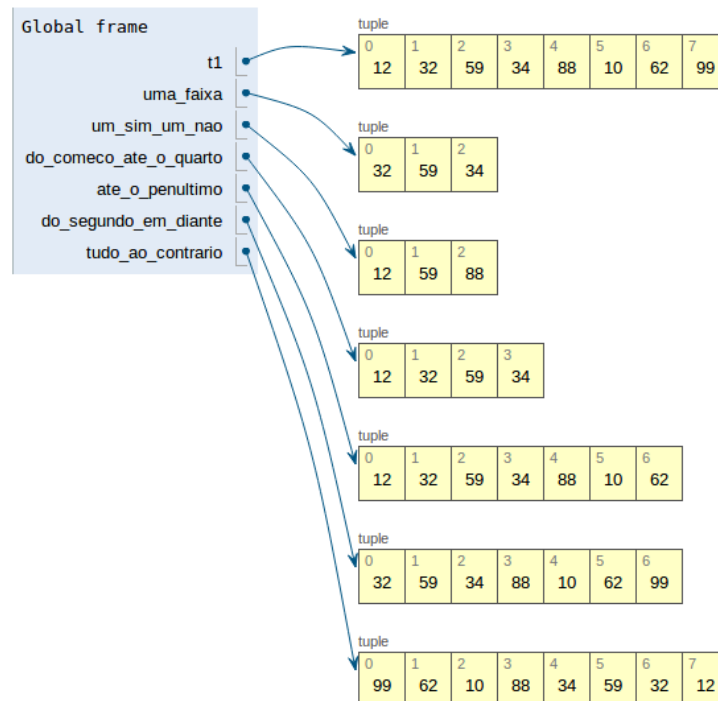
Index from rear:	-6	-5	-4	-3	-2	-1
Index from front:	0	1	2	3	4	5
	+---+	+---+	+---+	+---+	+---+	+---+
	a	b	c	d	e	f
	+---+	+---+	+---+	+---+	+---+	+---+

Slicing

fatiar, cria cópias de seções



```
1 t1 = (12, 32, 59, 34, 88, 10, 62, 99)
2
3 uma_faixa = t1[1:4]
4 #começa no 1 e pára antes do 4
5
6 um_sim_um_nao = t1[0:5:2]
7 #começa no zero e pára antes do 5, com passo de 2
8
9 do_comeco_ate_o_quarto = t1[:4]
10 #primeiro argumento vazio, começa do começo
11
12 ate_o_penultimo = t1[:-1]
13 #pára antes do último
14
15 do_segundo_em_diante = t1[1:]
16 #segundo argumento vazio, vai até o final
17
18 tudo_ao_contrario = t1[::-1]
19 #do final ao começo com passo de -1
```

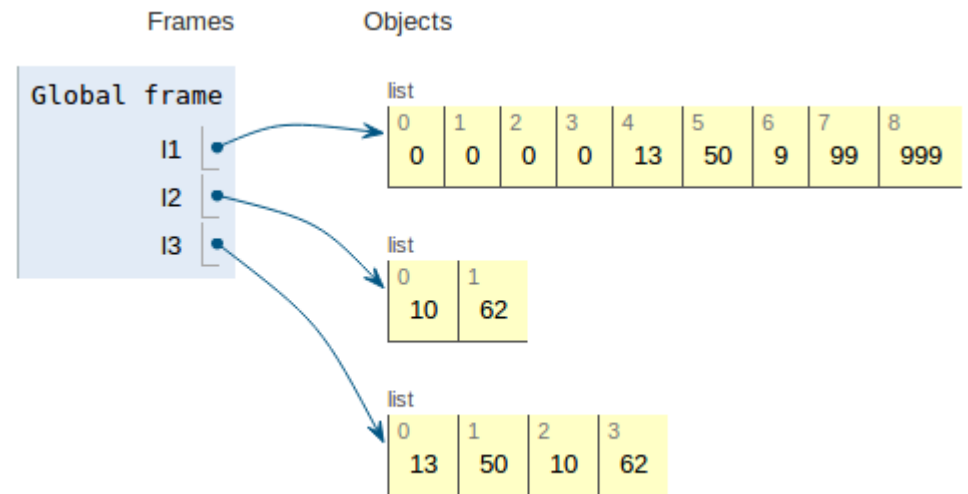


- Argumento vazio segue um comportamento padrão

Index from rear:	-6	-5	-4	-3	-2	-1										
Index from front:	0	1	2	3	4	5										
	+	-	-	+	-	-	+	-	-	+	-	-	+	-	-	+
		a		b		c		d		e		f				
	+	-	-	+	-	-	+	-	-	+	-	-	+	-	-	+
Slice from front:	:		1		2		3		4		5		:			
Slice from rear:	:		-5		-4		-3		-2		-1		:			

list

```
1 l1 = [13,50]
2 l2 = [10,62]
3
4 l3 = l1+l2 #concatenation
5
6 l1+=[9] #extend
7 l1.extend([99]) #extend
8 l1*=2 #extend
9
10 l1.append(999) #append
11
12 l1[0]=0 #rebind item
13 l1[1:4]=3*[0] #rebind items
```



- Append (apensar)
- Extend (extender)
- Rebind item (re vincular elemento)

Iterações



for

```
1 n=8
2
→ 3 for i in range(n):
4     print("I WILL NOT MOCK MRS. DUMBFACE")
```

```
I WILL NOT MOCK MRS. DUMBFACE
I WILL NOT MOCK MRS. DUMBFACE
I WILL NOT MOCK MRS. DUMBFACE
I WILL NOT MOCK MRS. DUMBFACE
I WILL NOT MOCK MRS. DUMBFACE
I WILL NOT MOCK MRS. DUMBFACE
I WILL NOT MOCK MRS. DUMBFACE
I WILL NOT MOCK MRS. DUMBFACE
I WILL NOT MOCK MRS. DUMBFACE
```

- Dois pontos
- Identação
- Função range

Operações com o contador

```
1 lista=[] #vazia
2 n=8
3
4 for i in range(n):
5     print("i: ", i)
6     lista.append(i/7)
7
```


```
i: 0
i: 1
i: 2
i: 3
i: 4
i: 5
i: 6
i: 7
```

Global frame		list							
lista		0	1	2	3	4	5	6	7
n	8	0.0	0.1429	0.2857	0.4286	0.5714	0.7143	0.8571	1.0
i	7								

somatório

FAZER O SOMATÓRIO DA LISTA
[2,4,6,8,10,12]

Nested for



```
1 k=1  
2 for i in range(3):  
3     for j in range(4):  
4         print(i,j,k)  
5     k+=1
```

Iteração ao longo dos elementos de uma lista

Global frame		list							
lista		0	1	2	3	4	5	6	7
n	8	0.0	0.1429	0.2857	0.4286	0.5714	0.7143	0.8571	1.0
i	7								

```
8 for i in range(n):  
9     print("item: ", lista[i])
```

```
10  
→ 11 for item in lista:  
12     print("item: ", item)  
13
```

```
item: 0.0  
item: 0.14285714285714285  
item: 0.2857142857142857  
item: 0.42857142857142855  
item: 0.5714285714285714  
item: 0.7142857142857143  
item: 0.8571428571428571  
item: 1.0
```

```
item: 0.0  
item: 0.14285714285714285  
item: 0.2857142857142857  
item: 0.42857142857142855  
item: 0.5714285714285714  
item: 0.7142857142857143  
item: 0.8571428571428571  
item: 1.0
```

- In - iteração sobre Cada elemento de uma lista

while

```
1  n=8
2  for i in range(n):
3      print("for:",i)
4
5  n=8
6  i=0
→ 7  while i<n:
8      print("while:",i)
9      i+=1
```

```
for: 0
for: 1
for: 2
for: 3
for: 4
for: 5
for: 6
for: 7
while: 0
while: 1
while: 2
while: 3
while: 4
while: 5
while: 6
while: 7|
```

- While (predicate)
- Loop infinito

Exemplo: achar maximo

Achar o maximo em [1,346432,68,1223,5,47,678]

Exemplo: Fatorial

$$0! = 1$$

$$n! = n * (n-1)! \text{ para } n > 0$$

Exemplo: exponencial

$$\begin{aligned} e^x &= f(0)\frac{x^0}{0!} + f'(0)\frac{x^1}{1!} + f''(0)\frac{x^2}{2!} + f'''(0)\frac{x^3}{3!} + f^{(4)}(0)\frac{x^4}{4!} + f^{(5)}(0)\frac{x^5}{5!} + \dots \\ &= \frac{x^0}{0!} + \frac{x^1}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \frac{x^5}{5!} + \dots \\ &= \sum_{n=0}^{\infty} \frac{x^n}{n!} \end{aligned}$$

Exemplo: Ordenar (sort)

Colocar a lista [1,346432,68,1223,5,47,678] em ordem crescente

$(1 < 5 < 47 < 68 < 678 < 1223 < 346432)$

Mutability

- IMMUTABLE

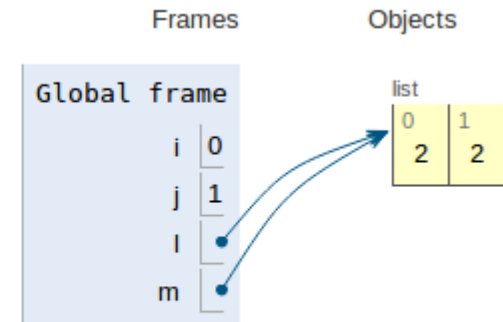
- Numbers
- String
- Bool

- MUTABLE

- List

Mutability

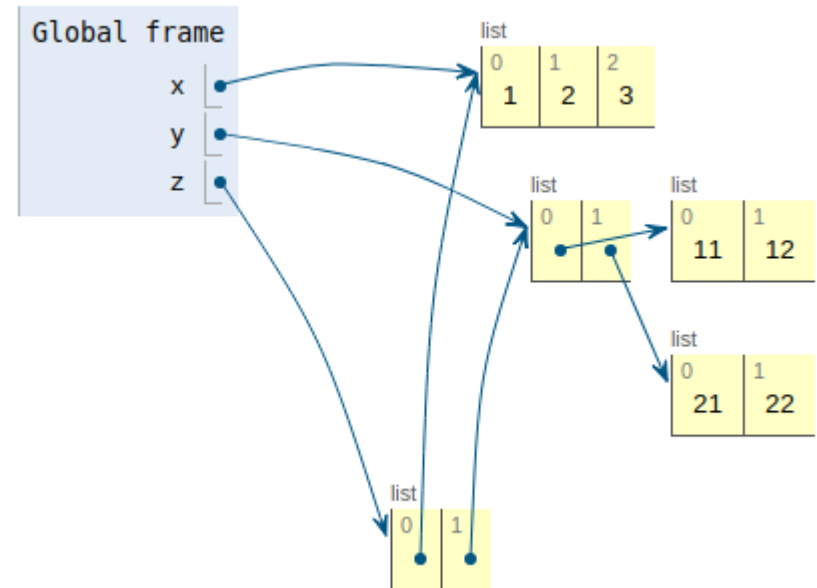
```
1 i=0
2 j=i
3 j+=1 #j agora dá nome a um novo objeto
4
5 l=[1,2]
6 m=l
→ 7 m[0]+=1 #modificamos o objeto lista ao qual l e m dão nome
```



listas de listas

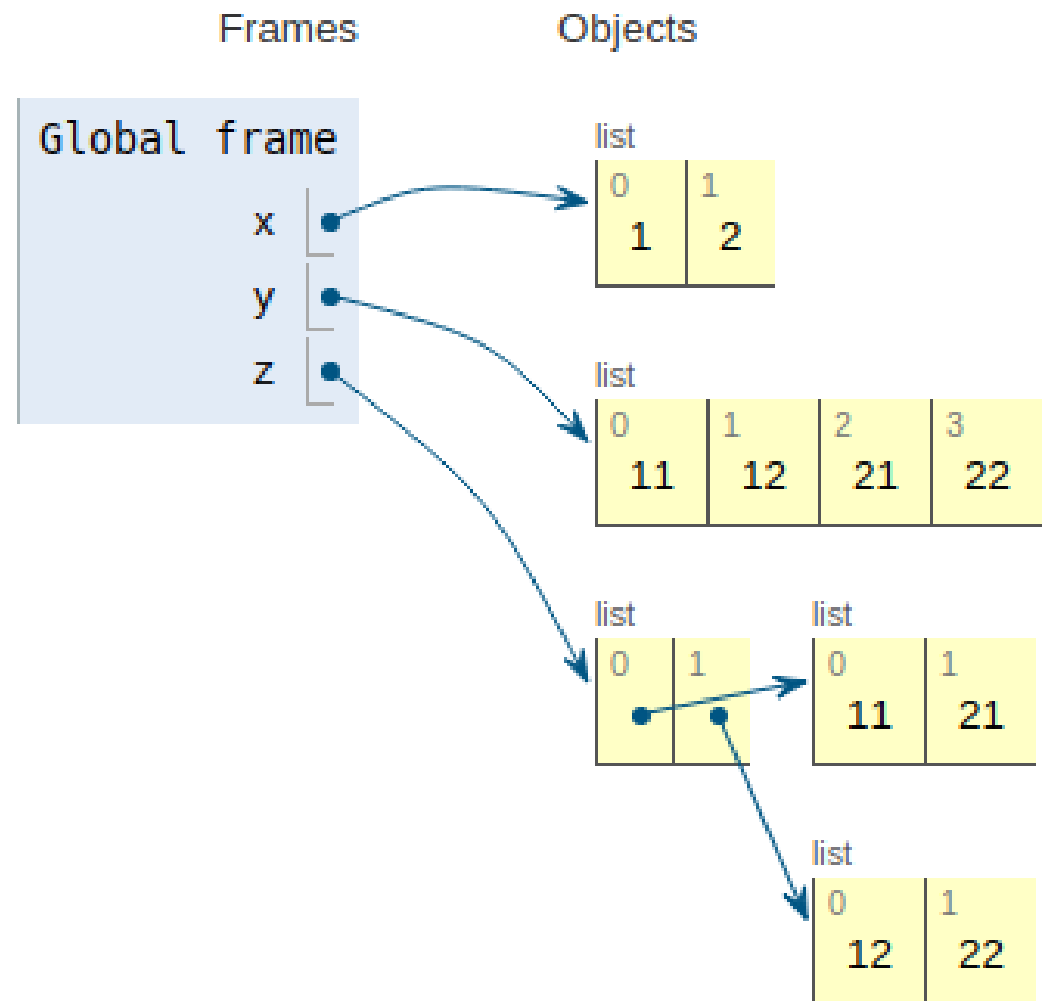
```
1 x=[1,2,3]
2
3 y=[[11,12],[21,22]]
4
→ 5 z=[x,y]
```

```
7 print('x0',x[0])
8 print('y0',y[0])
9 print('y00',y[0][0])
10 print('z0',z[0])
11 print('z00',z[0][0])
12 print('z1',z[1])
13 print('z10',z[1][0])
→ 14 print('z100',z[1][0][0])
```



```
x0 1
y0 [11, 12]
y00 11
z0 [1, 2, 3]
z00 1
z1 [[11, 12], [21, 22]]
z10 [11, 12]
z100 11
```

Comprehension



```
x=[i for i in range(1,3)]
```

```
y=[10*i+j for i in range(1,3) for j in range(1,3)]
```

```
z=[[10*i+j for i in range(1,3)] for j in range(1,3)]
```

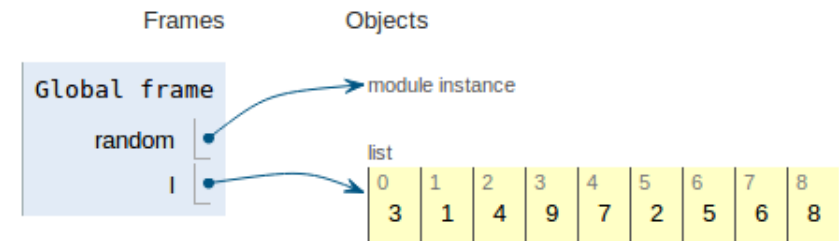
module random



```
1 import random
2 l=[1,2,3,4,5,6,7,8,9]
3 print(random.choice(l))
4 print(random.choice(l))
5 print(random.choice(l))
6 print(random.choice(l))
7
8 print(random.random())
9 print(random.random())
10 print(random.random())
11 print(random.random())
12
13 random.shuffle(l)
```

Print output (drag lower right corner to resize)

```
5
0.9654648863619172
0.48592769656281265
0.9182343317851318
0.8298529036589914
```



exercício

Jogo de par ou impar

Jogo de pedra papel e tesoura

Jogo da FORCA

Cálculo de movimento Browniano

Referências principais

[https://www.tutorialspoint.com/
python3/
python_basic_syntax.htm](https://www.tutorialspoint.com/python3/python_basic_syntax.htm)

[https://stackoverflow.com/
search](https://stackoverflow.com/search)

perguntas

