

ALGORITMOS E LÓGICA DE PROGRAMAÇÃO

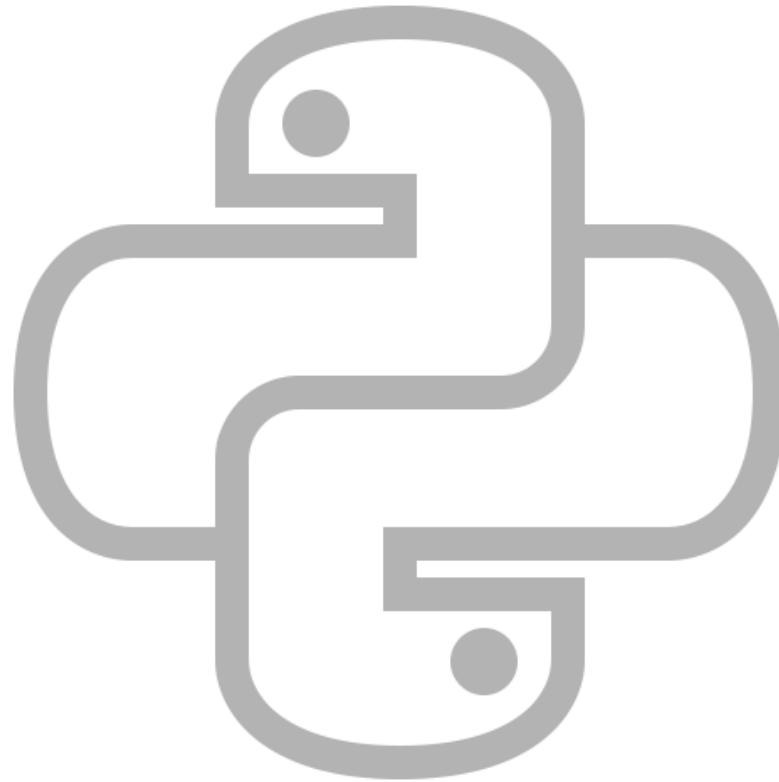
AULA 6: ESTRUTURA DE REPETIÇÃO FOR

DO QUE VAMOS FALAR

- **1.** Estrutura de repetição for
- **2.** Variações do range
- **3.** Exercícios extras



1. ESTRUTURA DE REPETIÇÃO FOR

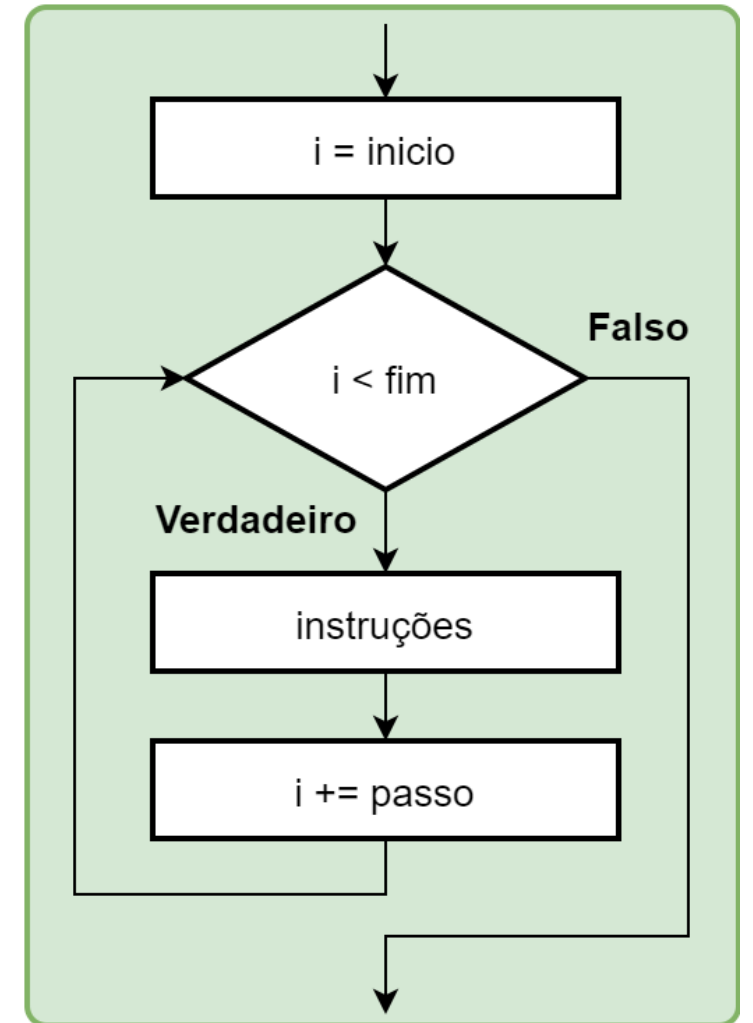


ESTRUTURA DE REPETIÇÃO FOR

A estrutura de repetição **for** executa um bloco de instruções internas repetidamente um número pré-determinado de vezes. Portanto, a quantidade de repetições **deve ser conhecida** antecipadamente.

```
...  
i = início  
while i < fim:  
    instruções  
    i += passo  
...
```

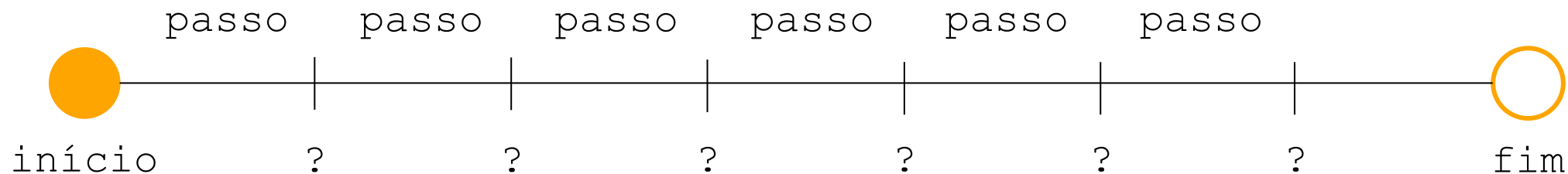
```
...  
for i in range(início, fim, passo):  
    instruções  
...
```



ESTRUTURA DE REPETIÇÃO FOR

```
...  
for i in range(início, fim, passo):  
    instruções  
...
```

- Inicialmente, usaremos o laço **for** juntamente com a função `range`.
- A princípio, podemos imaginar que `range` gera uma sequência de números inteiros no intervalo `[início..fim[` e pulando de `passo` em `passo`.
- Os três argumentos de `range` devem ser números inteiros e `passo` $\neq 0$.



ESTRUTURA DE REPETIÇÃO FOR

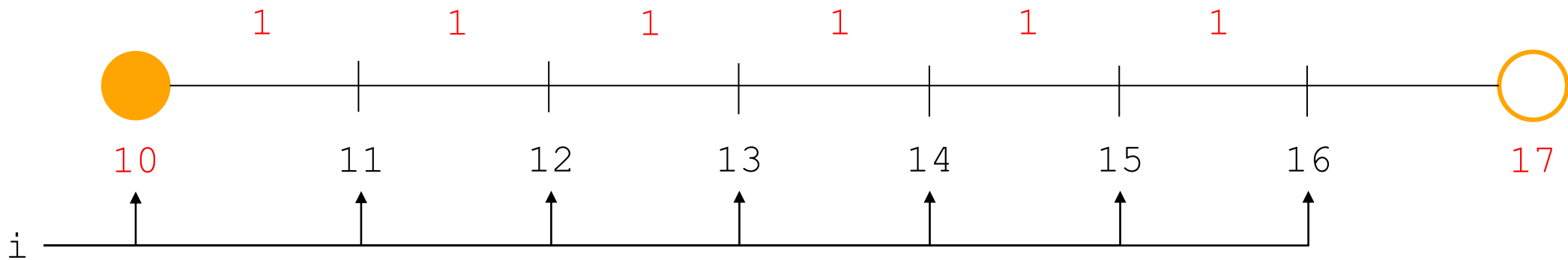
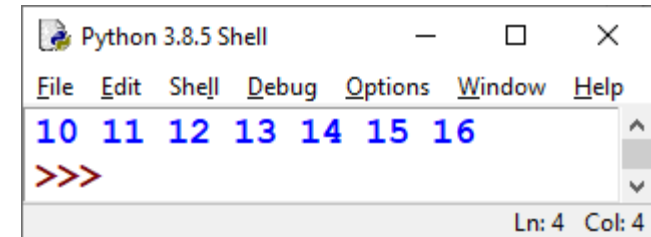
```
...  
for i in range(início, fim, passo):  
    instruções  
...
```

- A variável de controle do laço **for** pode ter qualquer identificador válido em Python, em nosso exemplo a chamamos de `i`.
- Caso `i` não exista previamente, será criada na primeira execução do laço.
- Na primeira execução do laço, `i` receberá o valor de `início`.
- Após cada execução de `instruções`, `i` receberá o próximo valor da sequência gerada por `range`.

ESTRUTURA DE REPETIÇÃO FOR

[EXEMPLO 1] Crie um programa que exiba todos os números naturais entre 10 e 16, inclusive os extremos.

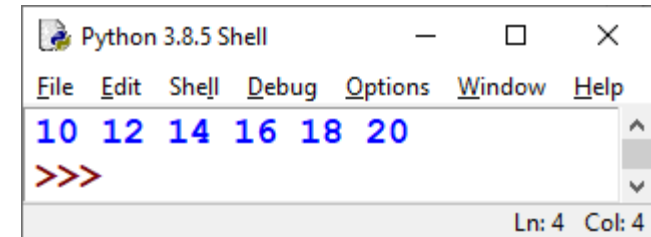
```
for i in range(10, 17, 1):  
    print(i, end=' ')
```



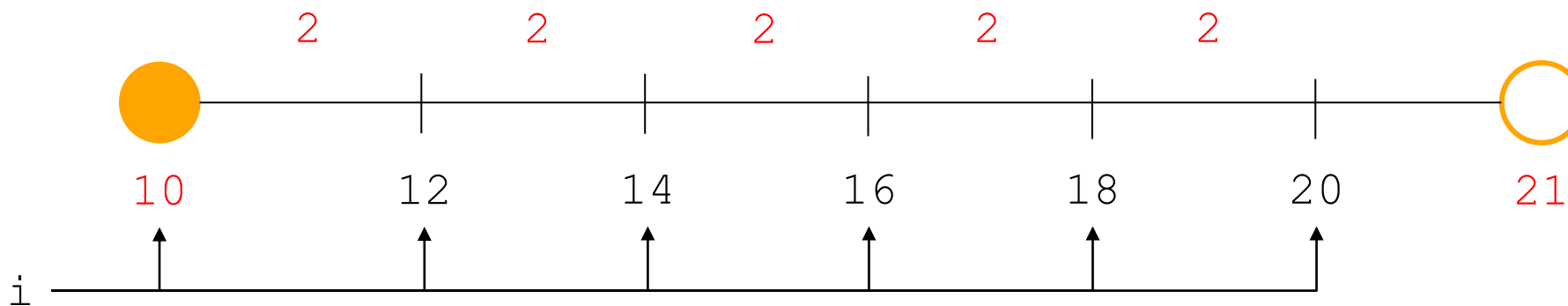
ESTRUTURA DE REPETIÇÃO FOR

[EXEMPLO 2] Crie um programa que exiba todos os números naturais pares entre 10 e 20, inclusive os extremos.

```
for i in range(10, 21, 2):  
    print(i, end=' ')
```



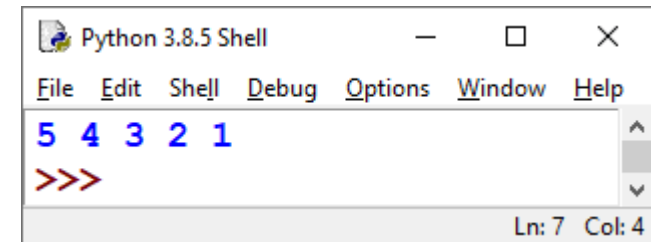
```
Python 3.8.5 Shell  
File Edit Shell Debug Options Window Help  
10 12 14 16 18 20  
>>>  
Ln: 4 Col: 4
```



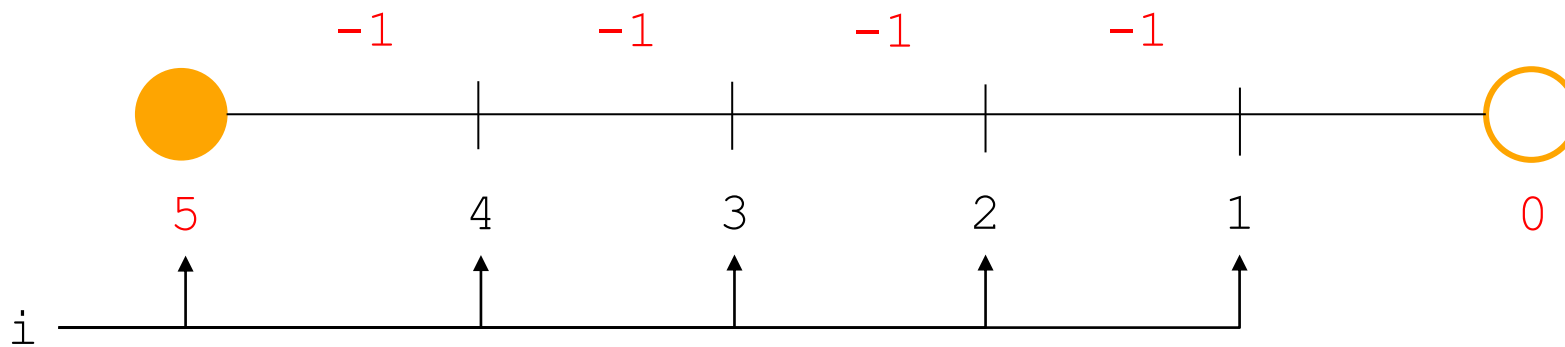
ESTRUTURA DE REPETIÇÃO FOR

[EXEMPLO 3] Crie um programa que exiba uma contagem regressiva de 5 à 1.

```
for i in range(5, 0, -1):  
    print(i, end=' ')
```



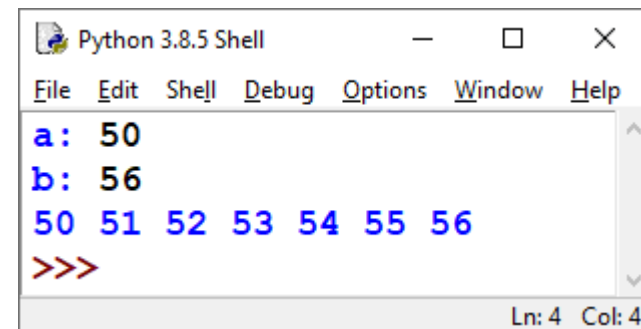
```
Python 3.8.5 Shell  
File Edit Shell Debug Options Window Help  
5 4 3 2 1  
>>>  
Ln: 7 Col: 4
```



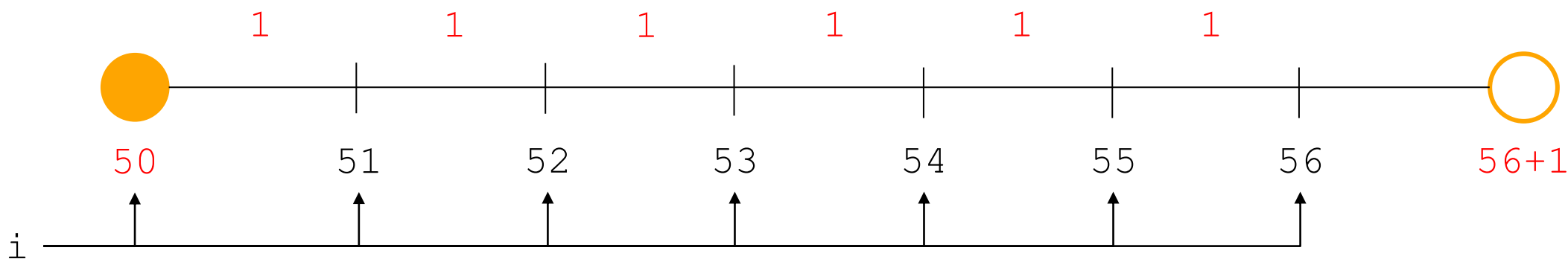
ESTRUTURA DE REPETIÇÃO FOR

[EXEMPLO 4] Crie um programa que tenha como entrada dois números inteiros a e b e exiba uma contagem progressiva de a até b .

```
a = int(input('a: '))  
b = int(input('b: '))  
for i in range(a, b+1, 1):  
    print(i, end=' ')
```



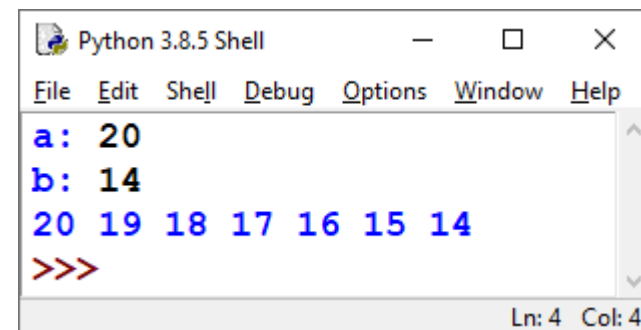
```
Python 3.8.5 Shell  
File Edit Shell Debug Options Window Help  
a: 50  
b: 56  
50 51 52 53 54 55 56  
>>>  
Ln: 4 Col: 4
```



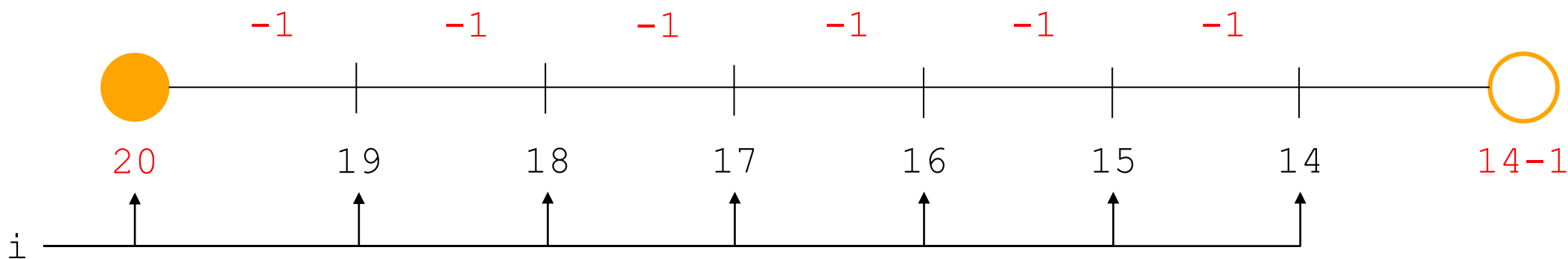
ESTRUTURA DE REPETIÇÃO FOR

[EXEMPLO 5] Crie um programa que tenha como entrada dois números inteiros a e b e exiba uma contagem regressiva de a até b.

```
a = int(input('a: '))
b = int(input('b: '))
for i in range(a, b-1, -1):
    print(i, end=' ')
```



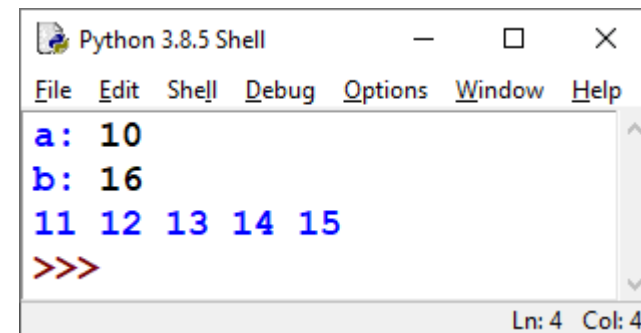
```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
a: 20
b: 14
20 19 18 17 16 15 14
>>>
Ln: 4 Col: 4
```



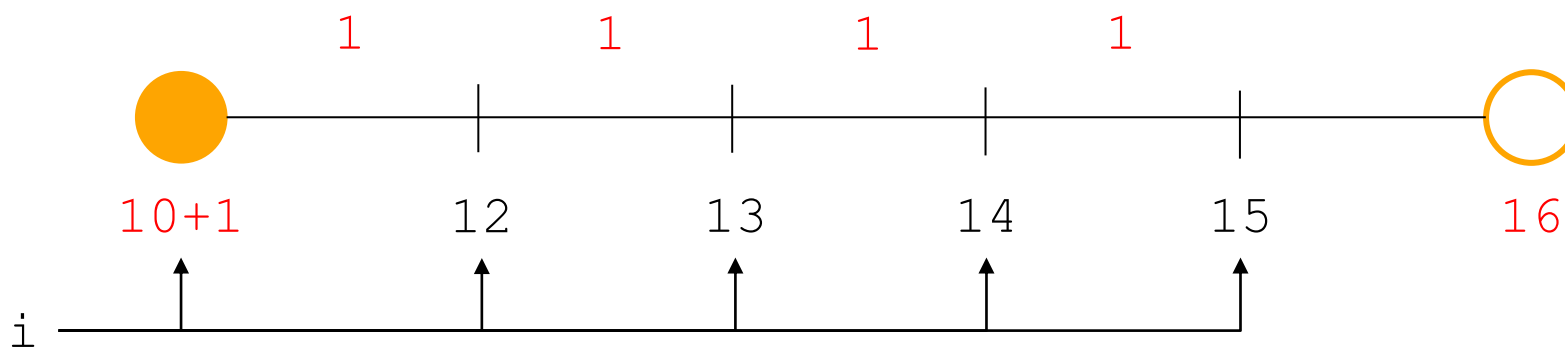
ESTRUTURA DE REPETIÇÃO FOR

[EXEMPLO 6] Crie um programa que tenha como entrada dois números inteiros a e b e exiba os inteiros do intervalo aberto crescente $]a..b[$.

```
a = int(input('a: '))
b = int(input('b: '))
for i in range(a+1, b, 1):
    print(i, end=' ')
```



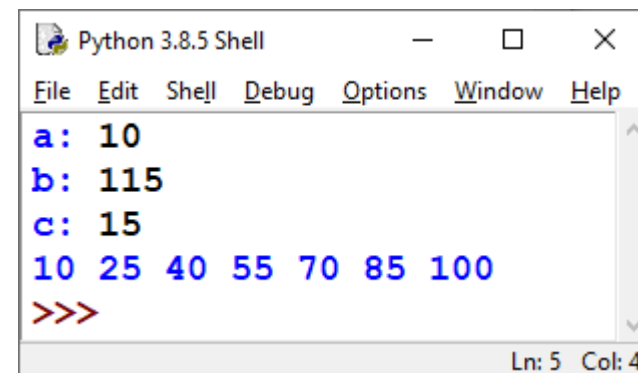
```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
a: 10
b: 16
11 12 13 14 15
>>>
```



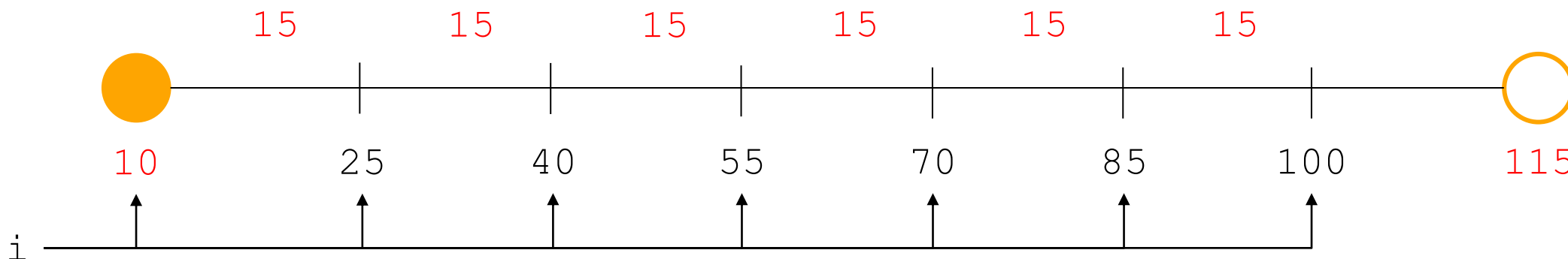
ESTRUTURA DE REPETIÇÃO FOR

[EXEMPLO 7] Crie um programa que tenha como entrada três números inteiros a , b e c e exiba os inteiros de $[a..b[$ pulando de c em c .

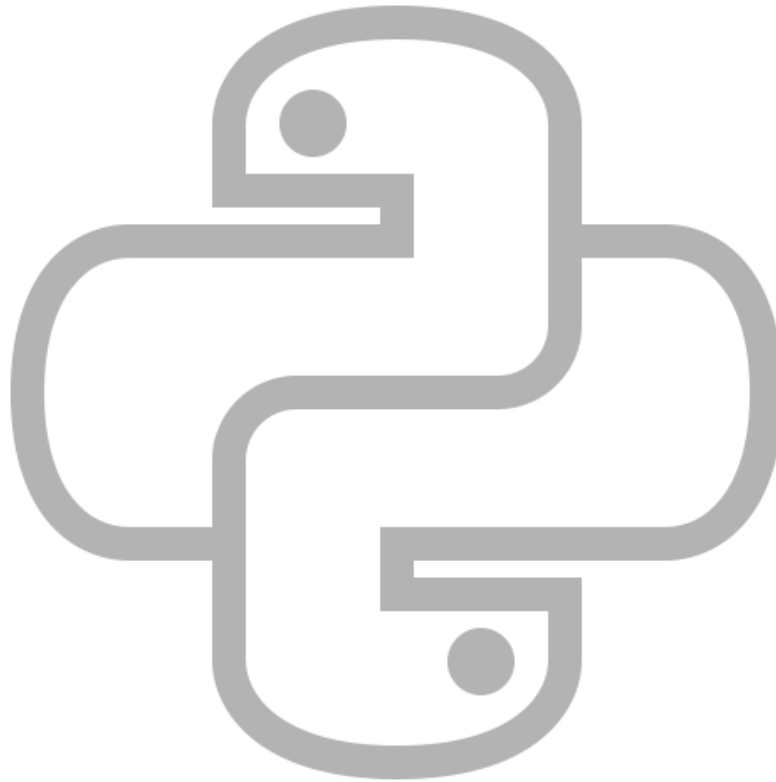
```
a = int(input('a: '))
b = int(input('b: '))
c = int(input('c: '))
for i in range(a, b, c):
    print(i, end=' ')
```



```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
a: 10
b: 115
c: 15
10 25 40 55 70 85 100
>>>
```



2. VARIAÇÕES DO RANGE



VARIAÇÕES DO RANGE

Até agora utilizamos o laço **for** e a função `range` com três argumentos, porém há três variações para `range`:

```
# início 0 e passo 1 por padrão.  
range(fim)
```

```
# passo 1 por padrão.  
range(início, fim)
```

```
# todos parâmetros personalizados.  
range(início, fim, passo)
```

OBSERVAÇÃO 1

Opte pela versão mais simples que puder, ou seja, caso o passo desejado seja 1, utilize a versão com apenas dois argumentos; caso o passo desejado seja 1 e o início zero, opte pela versão com apenas um argumento.

OBSERVAÇÃO 2

A versão com só um argumento também é recomendada quando queremos apenas definir a quantidade de vezes que o bloco de instruções do **for** será repetido, não importando o valor da variável de controle.

VARIAÇÕES DO RANGE

[EXEMPLO 8] Variações do range.

```
for i in range(5):  
    print(i, end=' ')
```

SAÍDA: 0 1 2 3 4

```
for i in range(2, 5):  
    print(i, end=' ')
```

SAÍDA: 2 3 4

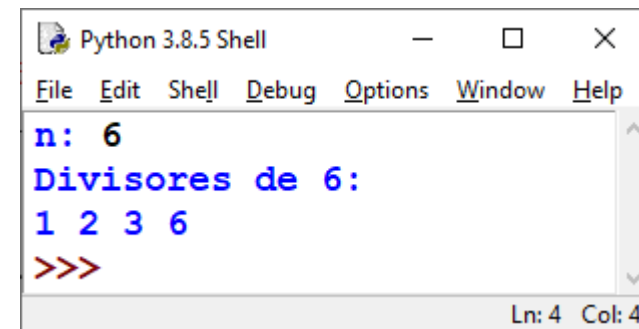
```
for i in range(5, 2, -1):  
    print(i, end=' ')
```

SAÍDA: 5 4 3

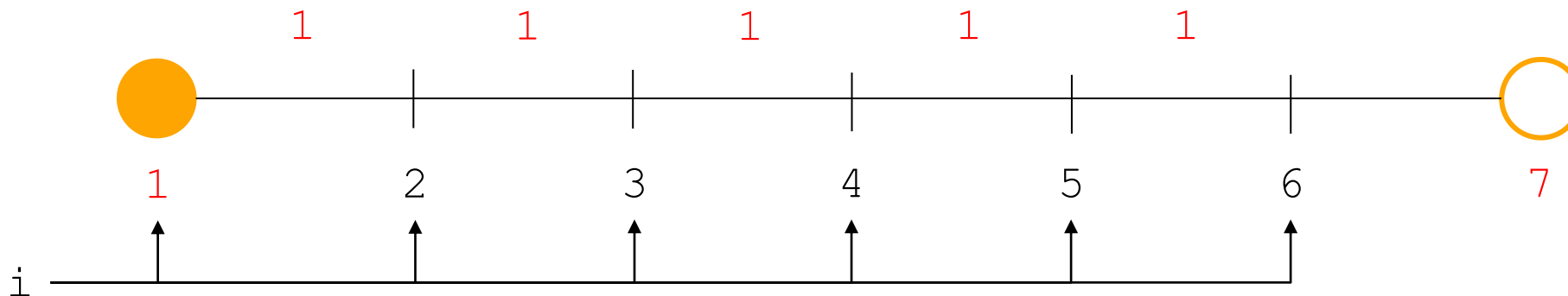
VARIAÇÕES DO RANGE

[EXEMPLO 9] Crie um programa que tenha como entrada um natural n e exiba todos os divisores naturais de n .

```
n = int(input('n: '))
print('Divisores de %d: ' % n)
for i in range(1, n+1):
    if n%i==0: print(i, end=' ')
```



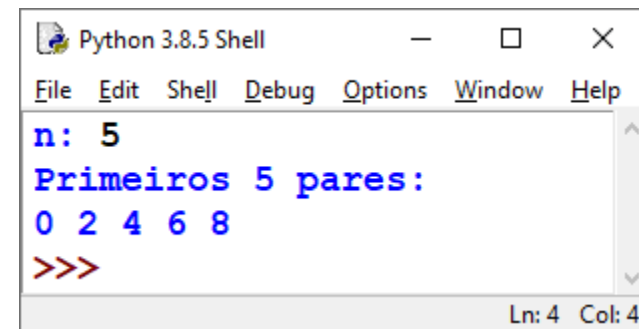
```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
n: 6
Divisores de 6:
1 2 3 6
>>>
Ln: 4 Col: 4
```



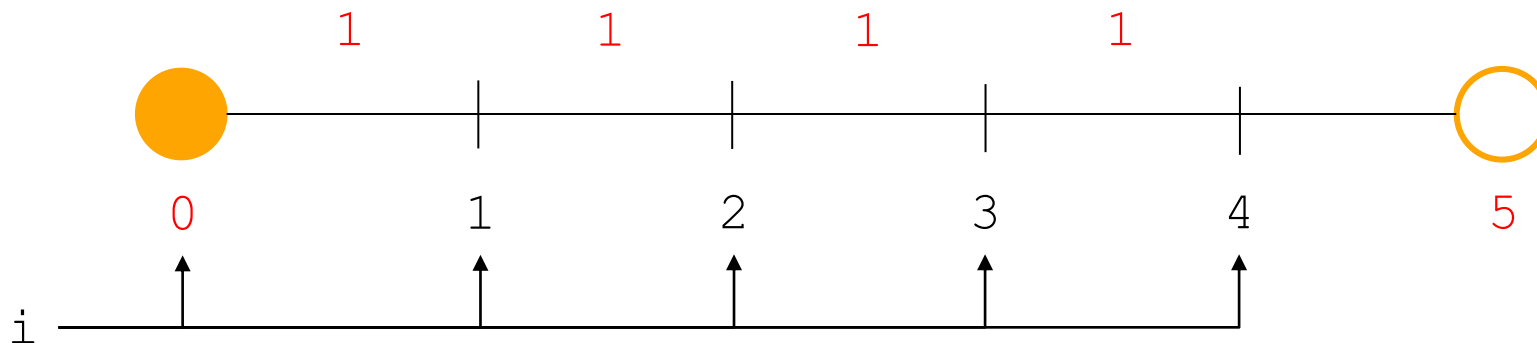
VARIAÇÕES DO RANGE

[EXEMPLO 10] Crie um programa que tenha como entrada um natural n e exiba os n primeiros naturais pares.

```
n = int(input('n: '))
print('Primeiros %d pares:' % n)
for i in range(n):
    print(2*i, end=' ')
```



```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
n: 5
Primeiros 5 pares:
0 2 4 6 8
>>>
Ln: 4 Col: 4
```



3. EXERCÍCIOS EXTRAS

