

ATLETAS DA PROGRAMAÇÃO

AULA 8



CONTINUAÇÃO DE LOOPS

- Introduzindo a função *for*.
- A função *range*.
- O operador *in*.
- O que é iteração.
- Acumuladores.



INTRODUZINDO A FUNÇÃO FOR

- A função `for` é utilizada para percorrer (isto é, iterar) uma sequência de dados (seja esse uma lista, uma tupla, uma string), executando um conjunto de instruções em cada item.
- Como você já sabe, Python utiliza indentação para separar blocos de código. Nos *loops* utilizando `for` não é diferente.



INTRODUZINDO A FUNÇÃO FOR

Sua sintaxe básica é: `for <nome variável> in <iterável>:`

Vamos entender:

- `<nome variável>` é o nome da variável que vai receber os elementos de `<iterável>`.
- `<iterável>` é o container de dados sobre o qual vamos iterar, podendo ser uma lista, uma tupla, uma string, um dicionário, entre outros.

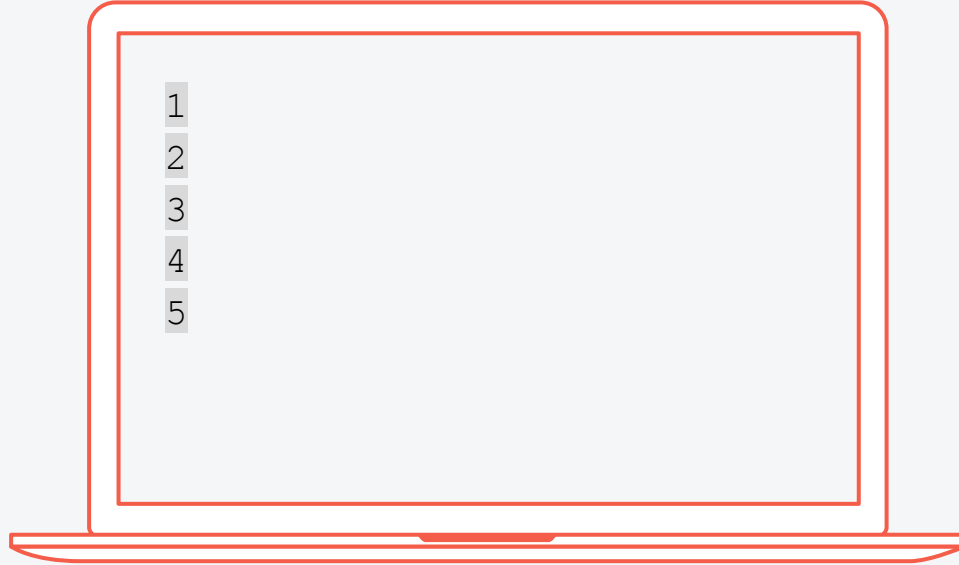


```
lista = [1, 2, 3, 4, 5]
```

```
for item in lista:  
    print(item)
```

EXEMPLO:

Vamos ver um exemplo para facilitar nossa vida.

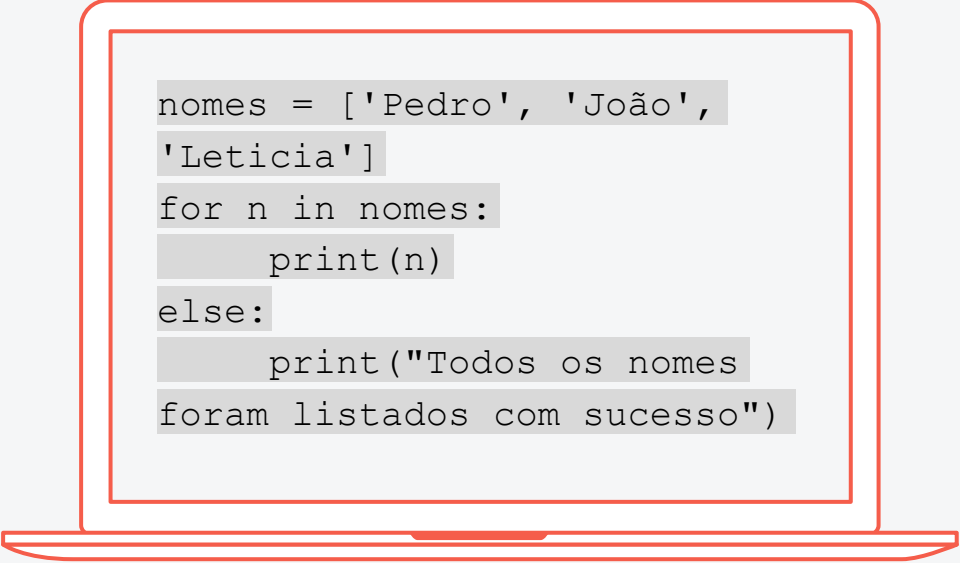


OUTPUT:

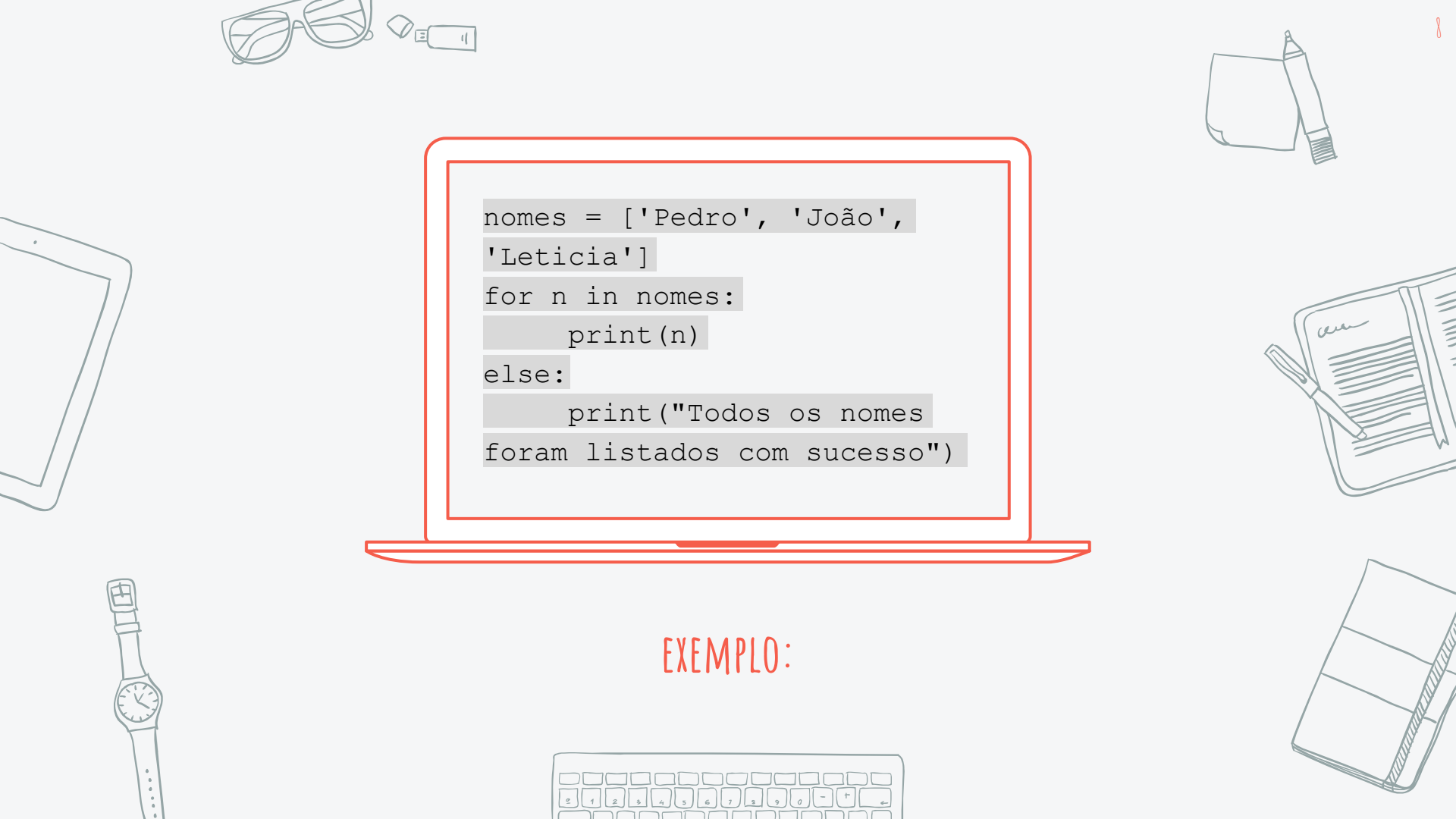
Vamos ver um exemplo para facilitar nossa vida.

FOR/ELSE

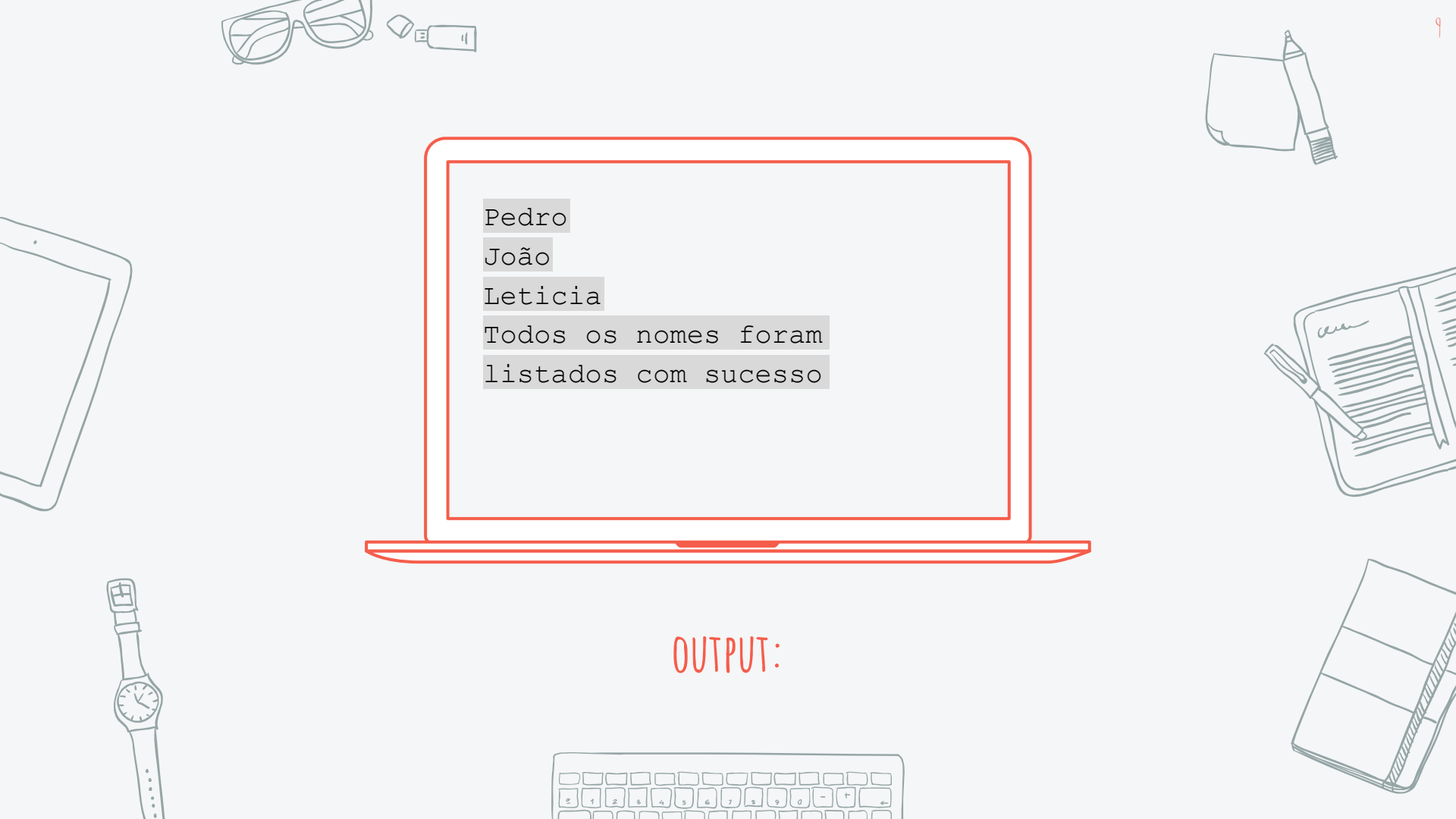
- É possível adicionar a instrução *else* ao final do *for*. Isso faz com que um bloco de código seja executado ao final da iteração, como mostra o exemplo do próximo slide.



```
nomes = ['Pedro', 'João',  
         'Leticia']  
for n in nomes:  
    print(n)  
else:  
    print("Todos os nomes  
foram listados com sucesso")
```



EXEMPLO:



Pedro

João

Leticia

Todos os nomes foram
listados com sucesso

OUTPUT:

O OPERADOR IN

- O operador **in** faz com que o operando à sua esquerda (a variável), receba o respectivo valor do iterável à direita (a estrutura de dados).
- No geral, uma estrutura for pode ser traduzida da seguinte forma:
- “Para (**for**) cada valor (**variável**) na (**in**) estrutura de dados (**iterável**) faça o seguinte (**bloco de código**).”
- Perceba que, se pegarmos as palavras em negrito, teremos:

```
for variável in iterável:  
    bloco de código
```


- Que é exatamente a sintaxe de um loop *for*!

A FUNÇÃO RANGE

A função range é bastante utilizada ao lado do for, pois ela gera uma lista de inteiros que, é claro, podemos iterar!

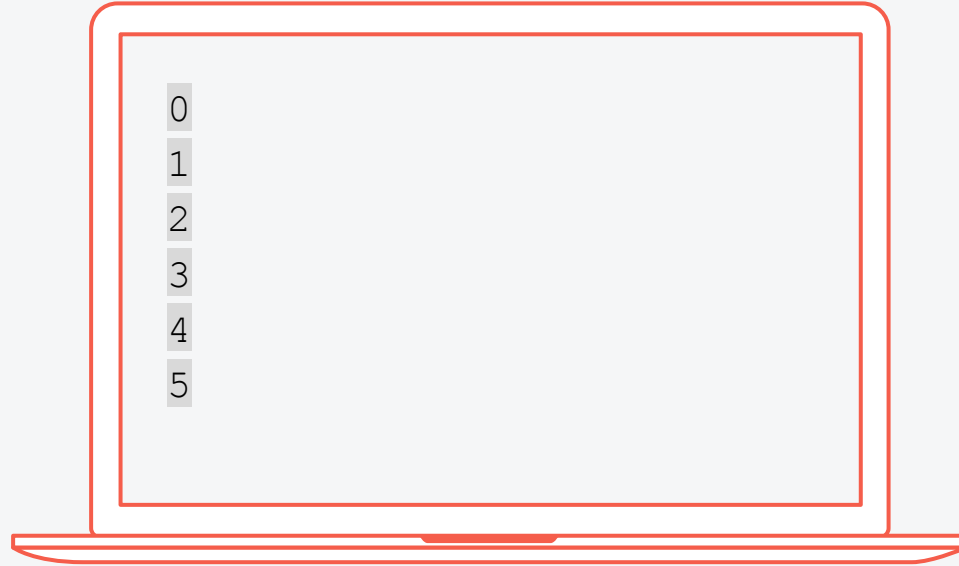
Para percorrer um bloco de código uma determinada quantidade de vezes, podemos usar a função range().

Por padrão, a função retorna uma lista de números, começando em 0 e incrementando em 1 até alcançar o número especificado.

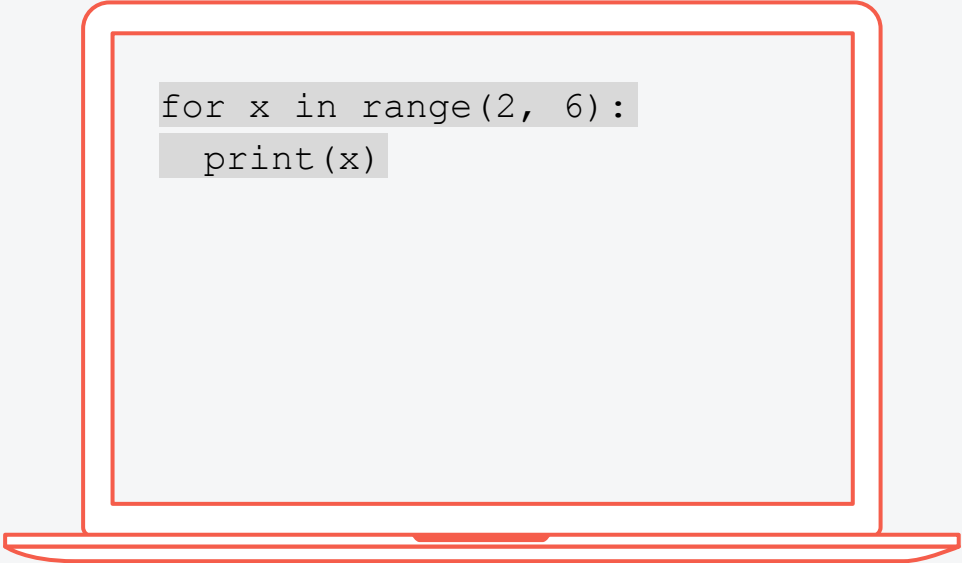


```
for x in range(6):  
    print(x)
```

EXEMPLO:



OUTPUT:



```
for x in range(2, 6):  
    print(x)
```

The image shows a central laptop screen with a red border. On the screen, there is a Python code snippet. The code is displayed in a light gray background with a darker gray highlight for the first line. The code is: `for x in range(2, 6):` followed by `print(x)` on the next line. The entire scene is decorated with faint line drawings of various objects: a pair of glasses and a USB drive at the top left, a notepad and a pen at the top right, a watch at the bottom left, and a keyboard at the bottom center.

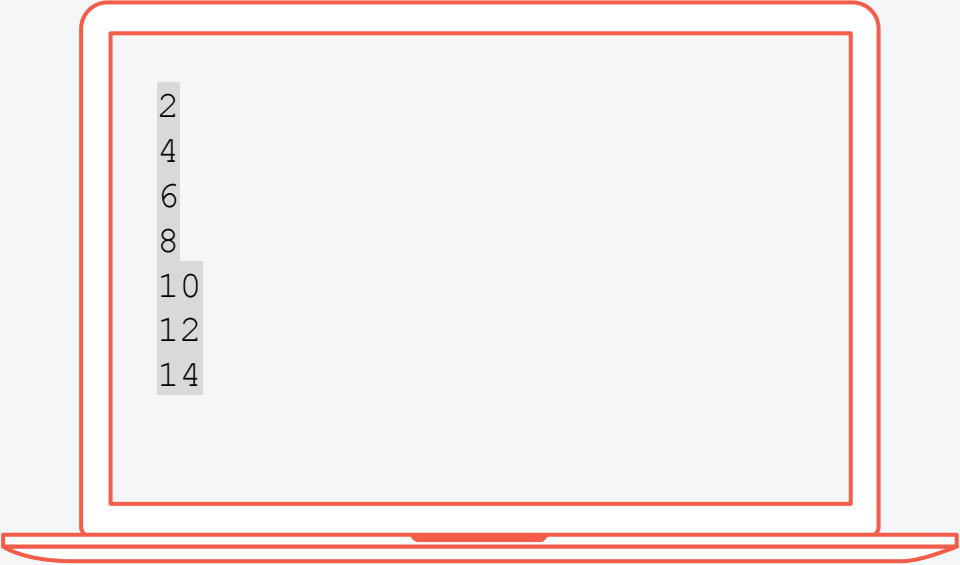
EXEMPLO 2: A FUNÇÃO `RANGE()` TEM COMO PADRÃO 0 COMO VALOR INICIAL, PORÉM É POSSÍVEL ESPECIFICAR O VALOR INICIAL ADICIONANDO UM PARÂMETRO: `RANGE(2, 6)`, QUE SIGNIFICA VALORES DE 2 A 6 (MAS NÃO INCLUINDO 6).

```
for x in range(2, 6):  
    print(x)
```

OUTPUT DO EXEMPLO 2

```
for x in range(2, 16, 2):  
    print(x)
```

EXEMPLO 3: TAMBÉM PODEMOS ALTERAR O PASSO COM O QUAL PERCORREMOS OS VALORES GERADOS PELA FUNÇÃO `RANGE()`:



2
4
6
8
10
12
14

OUTPUT DO EXEMPLO 3

A FUNÇÃO RANGE

Então, de maneira geral, a função range tem os seguintes parâmetros:

`range(<início>, <fim>, <passo>)`

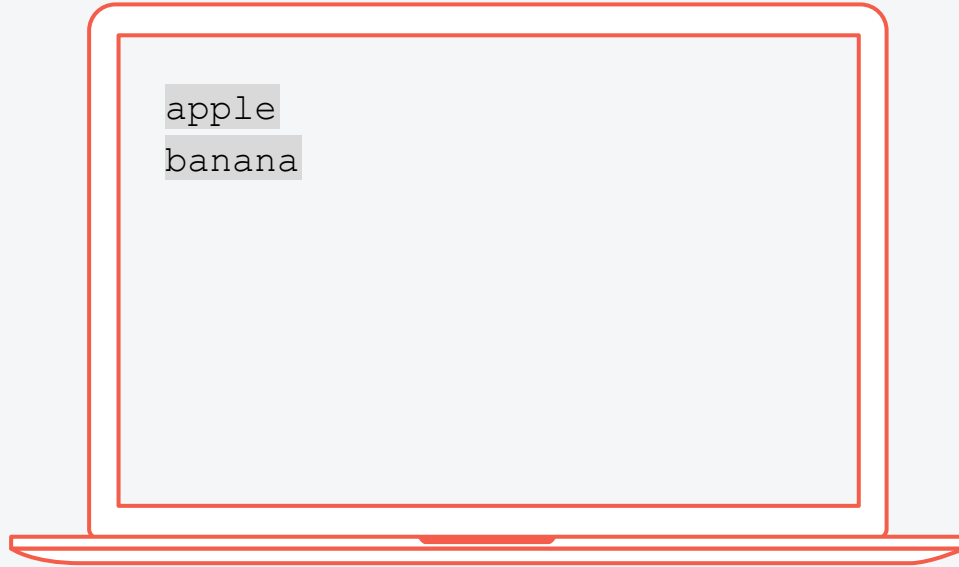
mas lembre que o interpretador sempre conta até, no máximo, **fim-1**, como nos exemplos anteriores, e nunca até **fim**!

A FUNÇÃO BREAK

- Com a instrução break podemos parar o loop antes que ele tenha percorrido todos os itens, desde que uma condição específica seja atendida!

```
fruits = ["apple", "banana",  
"cherry"]  
for x in fruits:  
    print(x)  
    if x == "banana":  
        break
```

EXEMPLO:



OUTPUT:

O QUE É ITERAÇÃO

- Iterar é a ação de repetir algo.
- Na programação, iteração significa a repetição de um conjunto de instruções por uma quantidade finita de vezes ou então, enquanto uma condição seja aceita.
- Na aula de hoje, **iteramos estruturas de dados** utilizando a função `for`, a qual repetirá um bloco de código uma quantidade de vezes conhecida previamente.

ACUMULADORES

- É uma variável que atua acumulando os valores a cada vez que o código é executado.

```
total = 0
for compra in range(10):
    valor = float(input("Valor
do produto: "))
    total += valor
```

EXEMPLO: NESSE CASO, A VARIÁVEL *TOTAL*, QUE GUARDA O VALOR COMPLETO DAS 10 COMPRAS, É UM ACUMULADOR.



DÚVIDAS?

Até a próxima aula!

