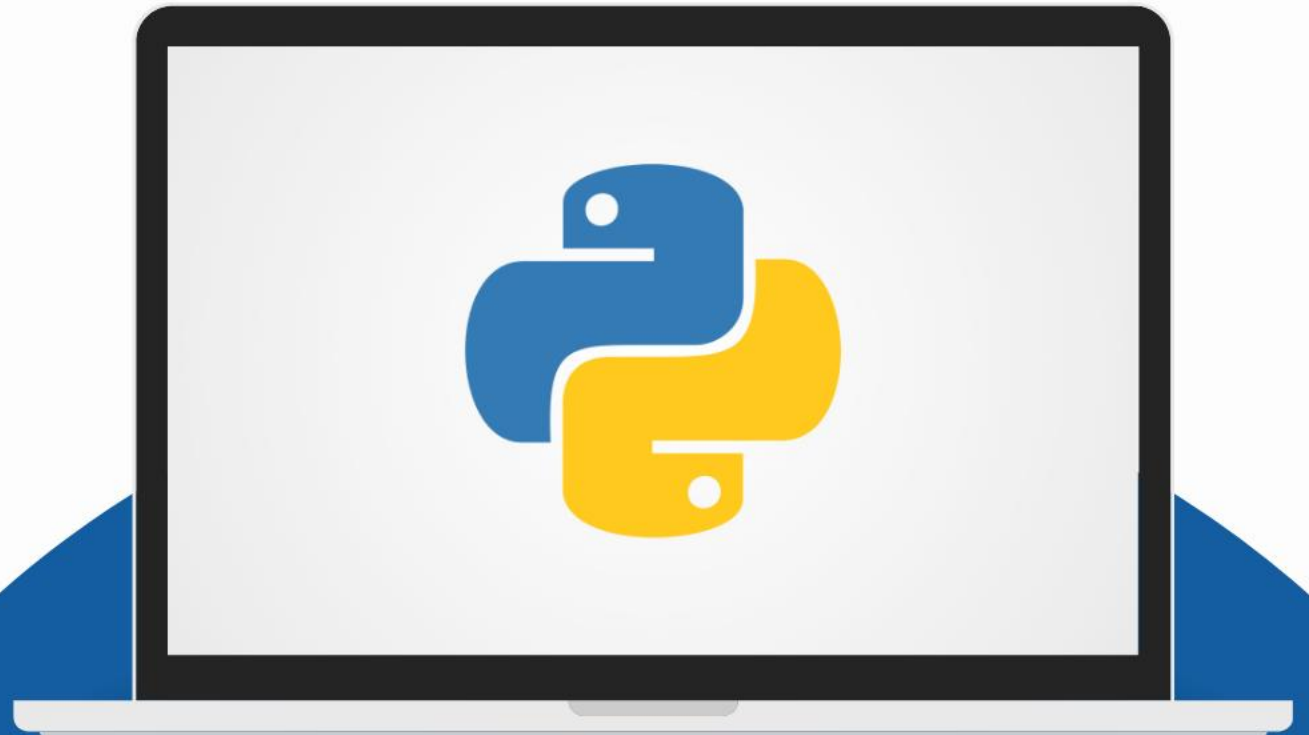


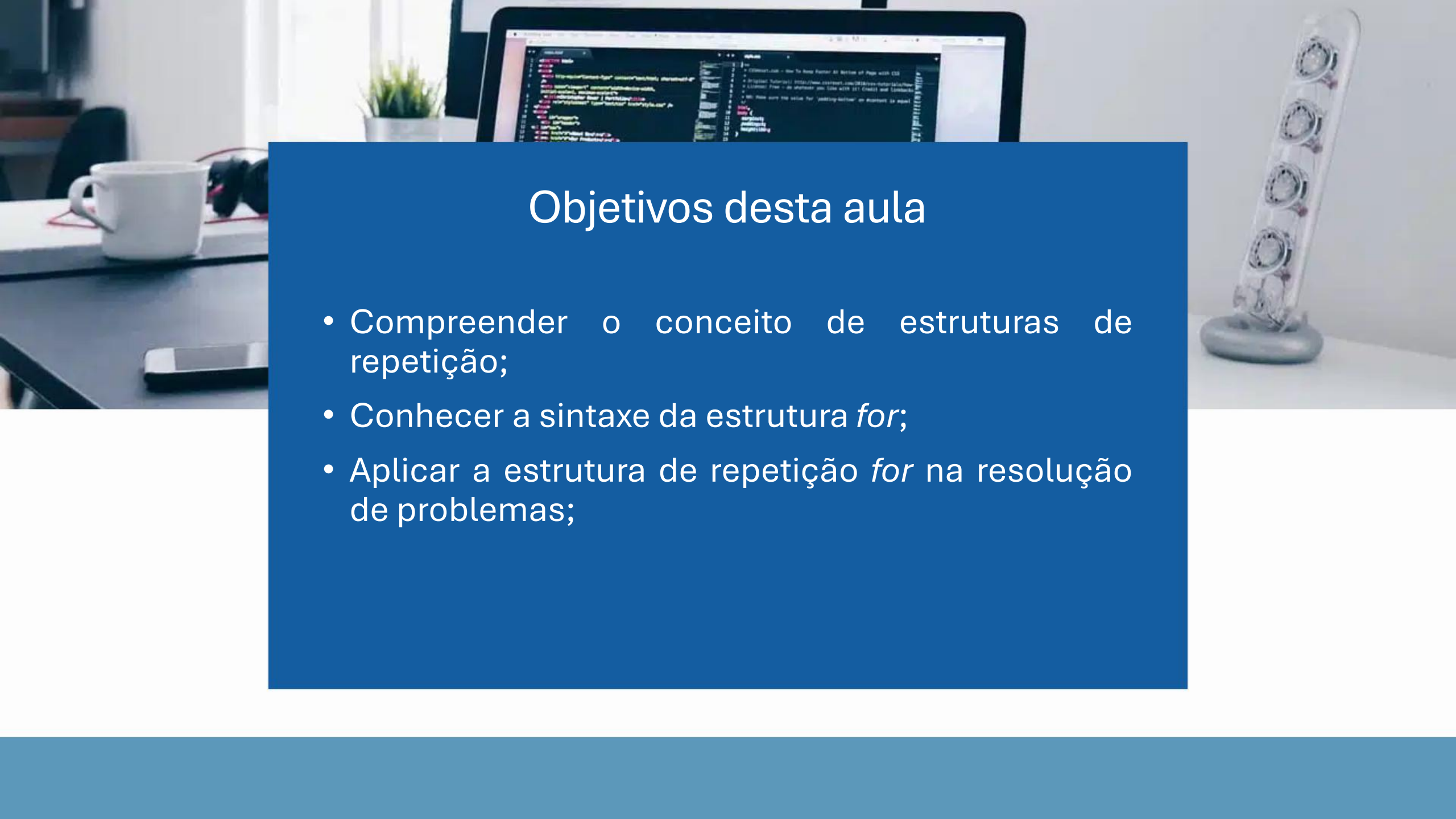


Aula #6:

Estruturas de repetição – o *for*

Prof. Dr. Luiz Álvaro de Oliveira Júnior





Objetivos desta aula

- Compreender o conceito de estruturas de repetição;
- Conhecer a sintaxe da estrutura *for*;
- Aplicar a estrutura de repetição *for* na resolução de problemas;

Sumário

▶ Estruturas de repetição	04
▶ A estrutura for	05
▶ A função range	08
▶ A função len	11
▶ Exemplos	13
▶ Exercícios	17



Estruturas de repetição

As **estruturas de repetição** (também chamadas de **laços** ou simplesmente **loops**) são estruturas que permitem executar um bloco de código várias vezes, de forma controlada e automática.

Elas são usadas para evitar que uma mesma instrução (ou conjunto de instruções) precise ser repetida manualmente no código, além de permitir lidar com situações em que o número de repetições é, a princípio, desconhecido.

No Python temos duas estruturas: **for** e **while**.

A estrutura **for**

A estrutura **for** é usada em situações nas quais o **número de repetições** que se deseja realizar é **conhecido** ou pode ser definido por uma **sequência**.

É muito útil para manipular estruturas de dados como listas, vetores, strings, tuplas e dicionários, permitindo percorrer seus diferentes elementos.

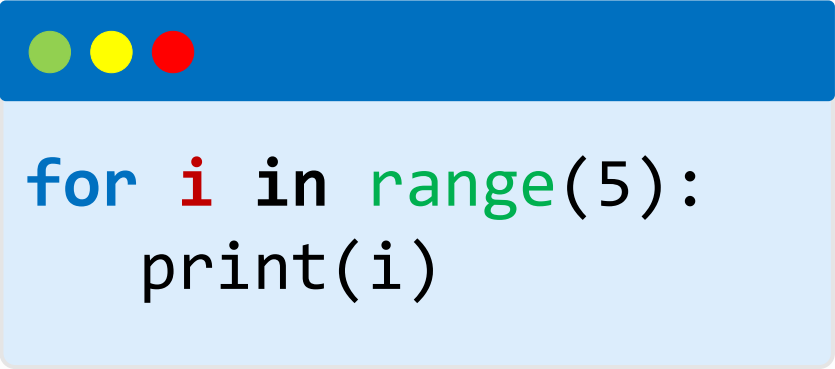
Também faz uso das indentações para explicitar o conjunto de instruções a serem repetidas.

A estrutura **for**

Como a estrutura **for** é normalmente usada em situações nas quais o número de repetições é conhecido, deve-se tomar cuidado para que a variável iteradora não extrapole o limite de iterações estabelecido no **for**, o que gera erro durante a execução do código.

A estrutura **for**

Vamos analisar o código abaixo para compreender a forma básica da estrutura for:



```
for i in range(5):  
    print(i)
```

- **for** é a estrutura de repetição;
- **i** é uma variável iteradora (ou contadora);
- **in** é uma palavra reservada usada para conectar o iterador à sequência ou estrutura de dados que será percorrida;
- **range** é uma função que cria uma sequência numérica, podendo ser substituída por uma estrutura iterável

A função **range**

A função **range** é uma função nativa do Python que gera uma sequência de números inteiros e que pode ser usada para percorrer as posições (ou elementos) de uma estrutura iterável, como listas, vetores, dicionários e até mesmo as próprias strings.

A função `range` tem três parâmetros e apresenta a seguinte sintaxe:

`range(inicio, fim, passo)`

A função **range**

O parâmetro **início** é um número inteiro que define **o primeiro elemento** da sequência, sendo, portanto, um limite includente. Seu uso é **opcional** e, não sendo informado, o padrão é zero.

O parâmetro **fim** é um número inteiro que define o **término da sequência**, porém, é um limite excludente, ou seja, **não ocorrerá na sequência**. Seu uso é obrigatório.

O parâmetro **passo** é um número inteiro que define o **intervalo entre dois elementos consecutivos** da sequência gerada. Seu uso é opcional e, por padrão, vale 1.

A função **range**

Vejam os alguns exemplos de retornos da função **range**:

Exemplo	Observação	Resultado
<code>range(0,5,1)</code>	Forma padrão	0, 1, 2, 3, 4
<code>range(0,5)</code>	Passo omitido	0, 1, 2, 3, 4
<code>range(5)</code>	Início e passo omitidos	0, 1, 2, 3, 4
<code>range(6,10,2)</code>	Forma padrão	6, 8
<code>range(6,11,2)</code>	Forma padrão	6, 8, 10
<code>range(10,0,-1)</code>	Contagem regressiva	10, 9, 8, 7, 6, 5, 4, 3, 2, 1

A função **len**

Outra função muito útil nas estruturas de repetição é a função **len** (do inglês *length*, que significa comprimento). Essa função é usada para retornar o número de elementos de um objeto que seja iterável ou que tenha comprimento definido, como as strings, listas, tuplas e dicionários.

Isso permite utilizar as estruturas de repetição **for** para acessar as posições desses elementos ou controlar a quantidade de repetições de forma genérica.

A função **len**

Vejam os alguns exemplos de retornos da função **len**:

Iterável	Uso	Resultado
<code>palavra = "Python"</code>	<code>len(palavra)</code>	6
<code>lista = [-11,2,4,92]</code>	<code>len(lista)</code>	4
<code>vetor = (-1.5,2.3,4.95)</code>	<code>len(vetor)</code>	3

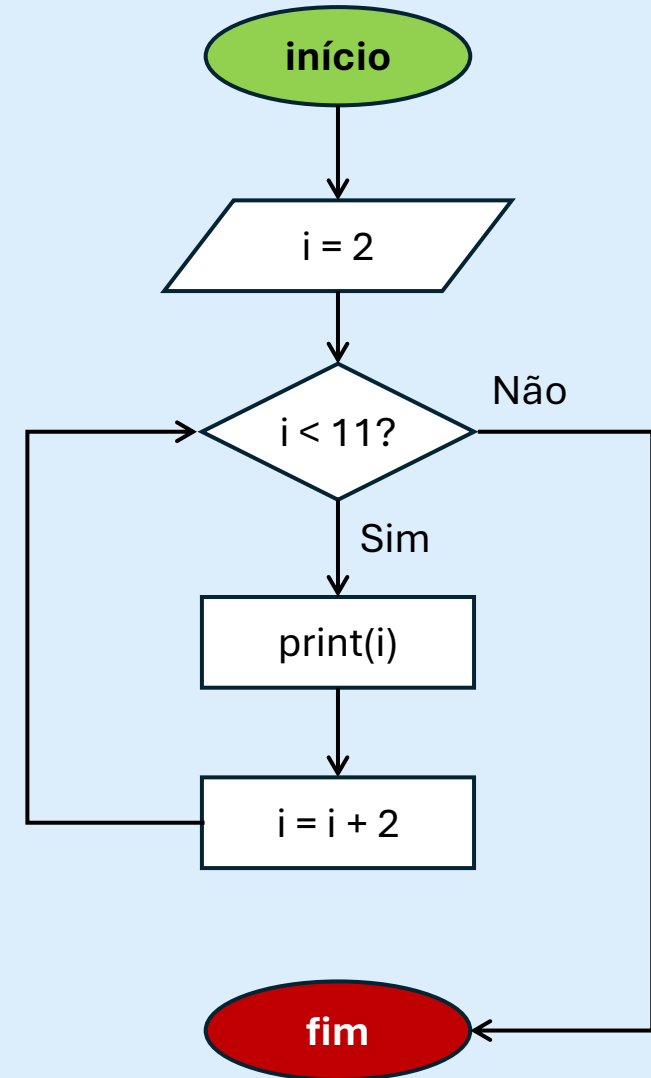
Exemplos

#1: Imprimir na tela os números inteiros pares entre 2 e 10 (incluso).



```
for i in range(2,11,2):  
    print(i)
```

#Acesso via sequência numérica



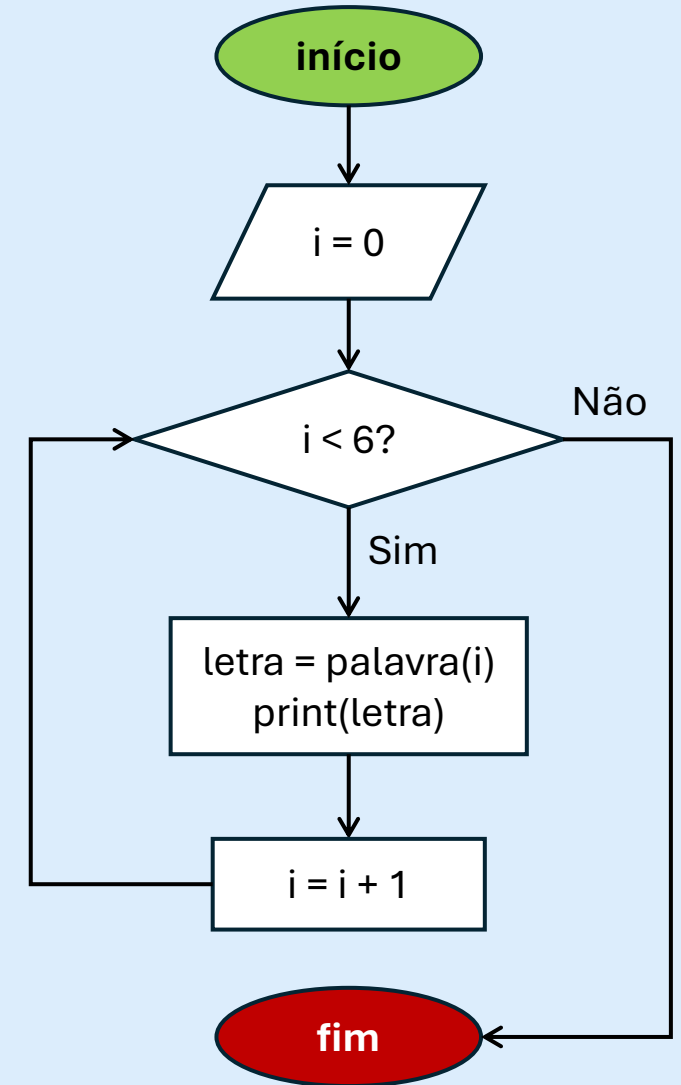
Exemplos

#2: Imprimir na tela os caracteres da palavra Python.



```
palavra = "Python"  
for letra in palavra:  
    print(letra)
```

#Acesso elemento a elemento



Exemplos

#3: Escreva um programa para calcular o fatorial de um número inteiro.

```
fat= 1
n = int(input("Deseja calcular o fatorial de qual número? "))
if n != 0:
    for i in range(1,n+1):
        fat *= i
    print(f"O fatorial de {n} é {fat}.")
else:
    print("Por definição, o fatorial de zero é 1.")
```

Exemplos

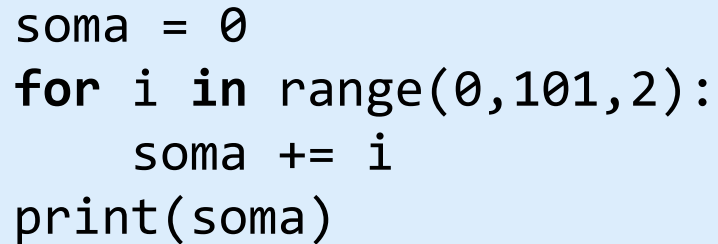
#4: Usando a estrutura for, escreva um programa que calcule a média das notas de um aluno. Interrompa digitando -1.

```
soma, n = 0, 0
for i in range(1000):
    nota = float(input("Digite a nota ou -1 para sair: "))
    if nota == -1: break
    soma += nota
    n += 1
if n > 0:
    media = soma / n
else:
    print("Nenhuma nota foi digitada.")
print(f"Média: {media:.1f}")
```

Exercícios

Exercício #1

Escreva um programa em Python que some todos os números pares entre 0 e 100. O programa deverá imprimir o resultado da soma na tela.

A code snippet is displayed within a light blue rectangular box that mimics a terminal window. The box has a dark blue header bar at the top containing three small circles in green, yellow, and red, representing window control buttons. The code is written in a monospaced font and consists of four lines: an initialization of a variable 'soma' to 0, a for loop that iterates over even numbers from 0 to 100, an accumulation of the current number into 'soma', and a final print statement.

```
soma = 0
for i in range(0,101,2):
    soma += i
print(soma)
```

Exercício #2

Escreva um programa em Python que some todos os números pares entre 0 e 100. O programa deverá imprimir o resultado da soma na tela. **Use uma estrutura condicional para verificar se o número é par antes de somar.**

```
soma = 0
for i in range(0,101,2):
    if i % 2 == 0:
        soma += i
    else:
        soma = soma
print(soma)
```

Exercício #3

Escreva um programa em Python que peça ao usuário para informar uma palavra e imprima as letras dessa palavra de trás para frente, uma por linha.

```
palavra = input("Digite uma palavra: ")
tamanho = len(palavra)
for pos in range(tamanho,0,-1):
    letra = palavra[pos-1]
    print(letra)
```

Exercício #4

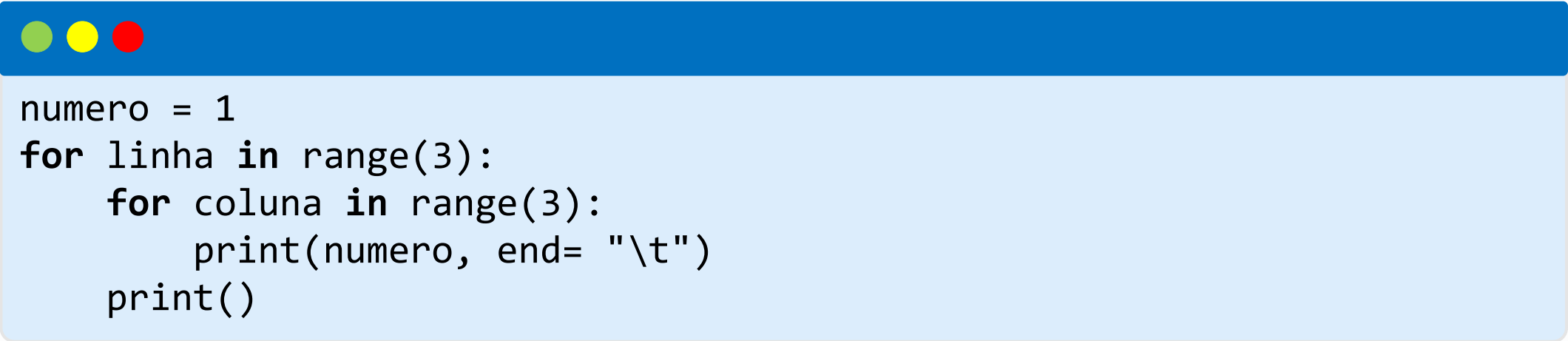
Escreva um programa em Python que peça ao usuário para informar uma palavra. Em seguida, imprima a palavra retirando a última letra até que não reste nenhuma. Veja o exemplo:

```
palavra = input("Digite uma palavra: ")
for i in range(len(palavra), 0, -1):
    aux = ""
    for j in range(i):
        aux += palavra[j]
    print(aux)
```

```
P Y T H O N
P Y T H O
P Y T H
P Y T
P Y
P
```

Exercício #5

Escreva um programa em Python que imprima os números de 1 a 9 na forma de uma matriz 3 por 3.

A screenshot of a Python code editor window with a blue title bar and three colored window control buttons (green, yellow, red) on the left. The code is written in a light blue area.

```
numero = 1
for linha in range(3):
    for coluna in range(3):
        print(numero, end= "\t")
    print()
```

