Aula 4 - Laço de Repetição for

Desvendando Estruturas de Repetição for e Listas em Python

Estrutura de Repetição for e Introdução a Listas

1

Compreender o laço for

Explorar o funcionamento e a sintaxe básica para iterações.

2

Diferenciar for de while

Entender quando usar cada estrutura de repetição.

3

Iterar sobre sequências

Aplicar o for em strings e outras sequências.

4

Manipulação de listas

Introdução à criação e uso de listas para armazenar dados.

O que é o Laço for?

O laço `for` é uma ferramenta essencial em programação, utilizada para **percorrer elementos de uma sequência**, como listas, `strings` ou outros objetos iteráveis. Ele executa um bloco de código repetidamente, uma vez para cada item da sequência, tornando a iteração de dados muito mais simples e organizada.

Sintaxe Básica:

```
for variável in sequência:
    # bloco de código a ser executado
```

A `variável` assume o valor de cada elemento da `sequência` a cada iteração.

Dominando o range()

A função `range()` é frequentemente usada com o laço `for` para gerar sequências de números de forma eficiente. Ela é fundamental para controlar o número de repetições.

- `range(fim)`: Gera números de `0` até `fim-1`.
- `range(início, fim)`: Gera números de `início` até `fim-1`.
- `range(início, fim, passo)`: Gera números de `início` até `fim-1`, incrementando pelo valor de `passo`.

Exemplo:

```
for i in range(1, 6):
    print(i)
```

Saída: 1, 2, 3, 4, 5

Diferença entre while e for

✅ while

Usado quando **não sabemos o número exato** de repetições. A execução continua **enquanto uma condição for verdadeira**.

```
x = 0
while x < 5:
    print(x)
    x += 1
```

Saída: `0 1 2 3 4`

✅ for

Ideal quando **sabemos o número de iterações** ou queremos percorrer **todos os elementos de uma sequência**.

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

Saída: `0 1 2 3 4`

Ambos são laços de repetição, mas a escolha depende do cenário: controle por condição (`while`) ou por sequência (`for`).

Iterando sobre Strings e Listas

Uma das grandes vantagens do laço `for` é sua capacidade de iterar diretamente sobre diferentes tipos de sequências, simplificando o acesso a cada um de seus elementos. Isso é especialmente útil para manipular dados textuais e coleções de itens.

Iterando sobre Strings:

O `for` percorre cada caractere da `string`.

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

Saída:

```
P
y
t
h
o
n
```

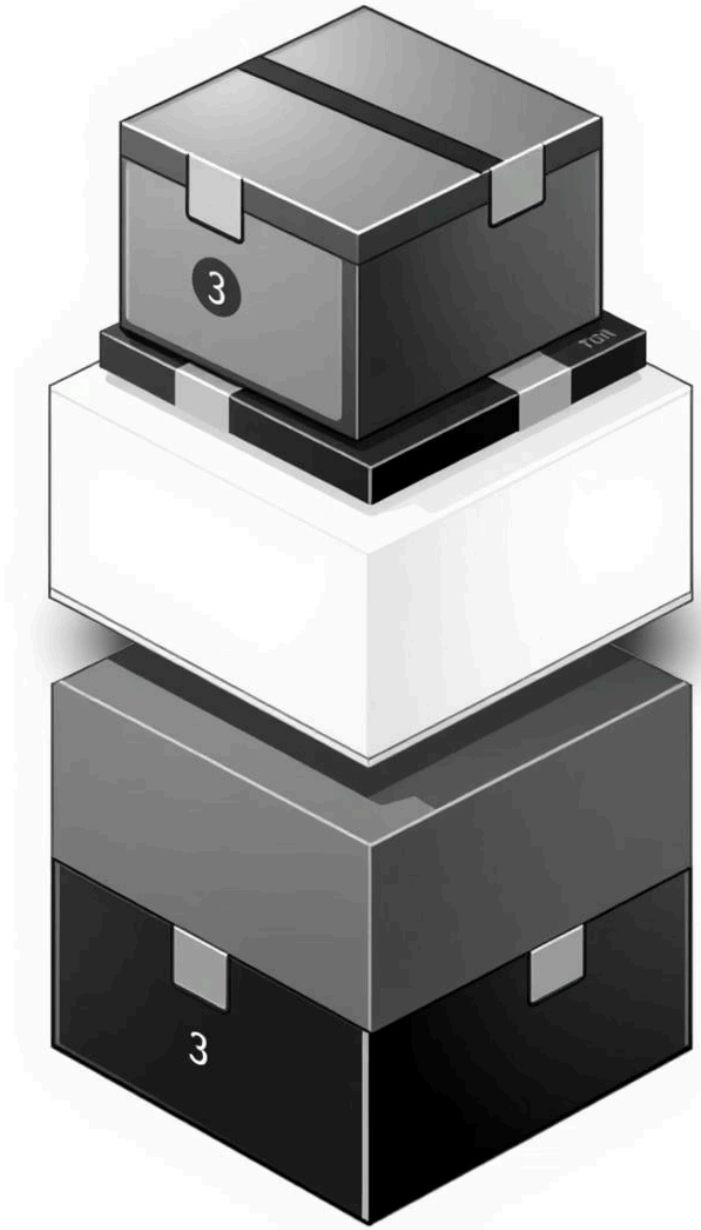
Iterando sobre Listas:

O `for` acessa cada item da lista.

```
frutas = ["maçã", "banana", "uva"]
for fruta in frutas:
    print(fruta)
```

Saída:

```
maçã
banana
uva
```



Introdução à Manipulação de Listas

Listas são coleções ordenadas e mutáveis de itens. Elas permitem armazenar múltiplos valores, de diferentes tipos, em uma única variável. São extremamente versáteis e fundamentais em Python para organizar dados.

Exemplo:

```
numeros = [10, 20, 30, 40, 50]
for n in numeros:
    print(f"O número é: {n}")
```

O laço `for` é o parceiro ideal das listas, pois facilita o **acesso e a manipulação individual de cada item**, tornando tarefas como exibir, processar ou filtrar dados muito mais eficientes.

Hora da Prática Guiada!

Para consolidar o que aprendemos, vamos colocar a mão na massa com alguns exercícios práticos usando o laço `for` e as listas. Lembrem-se: a prática leva à perfeição!

1 Contador Simples

Crie um programa que use um laço `for` para imprimir os números de `1` a `10`.

2 Contagem Regressiva

Desenvolva um código que utilize `for` para contar de `10` a `1`.

3 Contador Interativo

Peça ao usuário um número e faça um laço `for` que conte de `1` até esse número.

4 Tabuada Dinâmica

Solicite um número ao usuário e, usando `for`, exiba a tabuada completa desse número.