Laços de Repetição em Python

Neste capítulo, exploraremos os conceitos de laços de repetição em Python, que são fundamentais para executar uma mesma sequência de código várias vezes. Abordaremos dois tipos principais de laços: `for` e `while`. Aqui estão os tópicos que serão discutidos:

1. **Laço `for` para Iteração:**
   - Aprenda como usar o laço `for` para iterar sobre sequências como listas, tuplas e strings.

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

2. **Estrutura `range()`:**
   - Explore a função `range()` para gerar sequências numéricas utilizadas frequentemente com laços `for`.

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

3. **Laço `while` para Execução Condicional:**
   - Entenda como o laço `while` executa repetições com base em uma condição.

```
contador = 0
while contador < 5:
    print("Contagem:", contador)
    contador += 1
```

4. **Instruções `break` e `continue`:**
   - Introdução às instruções `break` e `continue` para controle do fluxo dentro de laços.

```
for i in range(10):
    if i == 5:
        break
    print(i)
```

5. **Laços com Estruturas de Dados:**
   - Utilize laços `for` e `while` para iterar sobre listas, tuplas, dicionários e outros tipos de dados.

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

Exemplos adicionais

Abaixo estão alguns exemplos adicionais de laços de repetição em Python, incluindo o uso do laço `for` sem a necessidade de uma variável de iteração:

1. **Laço `for` com Strings:**
    - Iteração sobre os caracteres de uma string.

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

2. **Laço `for` com Enumerate:**
    - Utilização da função `enumerate()` para obter tanto o índice quanto o valor durante a iteração.

```
frutas = ['maçã', 'banana', 'uva']
for indice, fruta in enumerate(frutas):
    print(f"Índice {indice}: {fruta}")
```

3. **Laço `for` sem Variável de Iteração:**
    - Execução do laço `for` sem utilizar uma variável de iteração. Isso é útil quando não precisamos do valor da variável em cada iteração.

```
for _ in range(5):
    print("Executando esta ação 5 vezes.")
```

Quando utilizar o _

O uso do _ como variável de descarte é uma convenção comum em Python. A variável _ é frequentemente utilizada quando o valor de uma variável não será utilizado ou é ignorado em uma iteração específica. Isso torna o código mais legível e indica explicitamente aos leitores que o valor não é relevante naquele contexto.

A variável _ é usada para indicar que não estamos interessados no valor de cada iteração, apenas queremos executar a ação um número específico de vezes.

É importante observar que o uso do _ é uma convenção e não uma regra rígida. Algumas IDEs ou linters podem sugerir que você não está usando a variável, e é uma prática aceitável em muitos casos, especialmente

quando o valor não é relevante para a lógica do código.

Além de utilizar ele nos laços de repetição, você pode usar o `_` para indicar que não está interessado em algum trecho de dados que considere descartável. Aqui está um exemplo de como você poderia fazer isso:

```
dados_pessoa = ["João", "Silva", "Viúvo", "456789123", "789123456"]

nome, sobrenome, _, *documentos = dados_pessoa

print("Nome:", nome)
print("Sobrenome:", sobrenome)
print("Documentos:", documentos)
```

Neste exemplo, `nome` e `sobrenome` recebem os dois primeiros elementos da lista, enquanto `_` é usado para indicar que não estamos interessados no terceiro elemento. A notação `*documentos` é usada para coletar todos os itens restantes da lista em uma variável chamada `documentos`.

A saída seria:

```
Nome: João
Sobrenome: Silva
Documentos: ['456789123', '789123456']
```

Assim, você consegue extrair os elementos desejados da lista e ignorar aqueles que não são necessários.

Conclusão

Esses exemplos adicionais ilustram diferentes formas de aplicar laços de repetição em Python, demonstrando a versatilidade e praticidade dessas estruturas de controle de fluxo.

Ao final deste capítulo, você estará apto a utilizar laços de repetição para automatizar tarefas repetitivas e processar dados de maneira eficaz em seus programas Python. Prepare-se para aprofundar seus conhecimentos em estruturas de controle de fluxo!