Laços de Repetição

Instrutor: Mateus De Oliveira Silveira

1 Introdução

Nesta aula, vamos aprender sobre laços de repetição em Python. Eles são como aquelas tarefas que precisamos repetir várias vezes, como quando estamos contando quantas vezes uma bola quica no chão. Os dois principais laços que vamos ver são o `for` e o `while`. Também vamos conhecer um pouco sobre algo chamado iteradores, que você pode imaginar como uma maneira de passar por uma lista de coisas, uma de cada vez.

2 Laço `for`

Imagine que você tem uma lista de coisas para fazer, como uma lista de compras. O laço `for` é como você passar por cada item dessa lista, um por um, até acabar.

2.1 Sintaxe Básica

A sintaxe básica do laço `for` em Python é:

```
for variavel in sequencia:
    # bloco de código
```

Aqui, `variavel` é como uma pequena sacola onde você coloca cada item da sua lista, um por vez.

2.2 Exemplo 1: Iterando sobre uma lista

Vamos ver um exemplo onde passamos por uma lista de números e imprimimos cada um deles:

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

for numero in numeros:
```

```
    print(numero)
```

É como se você tivesse uma lista de números e estivesse lendo cada um em voz alta.

2.3 Exemplo 2: Usando `range()`

E se você quisesse contar até 5? A função `range()` cria uma lista de números para você:

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

Aqui, é como se você tivesse uma lista de números de 1 a 5, e estivesse passando por cada um deles.

2.4 Exemplo 3: Iterando sobre uma string

Você pode até passar por cada letra de uma palavra, como se estivesse soletrando:

```
palavra = "Python"

for letra in palavra:
    print(letra)
```

Imagine que você está lendo cada letra de uma palavra, uma de cada vez.

2.5 Exemplo 4: Iterando sobre um dicionário

E se você tivesse uma lista de alunos e suas notas? Você pode passar por cada aluno e imprimir o nome e a nota:

```
alunos = {"Ana": 85, "Bruno": 92, "Carlos": 78}

for nome, nota in alunos.items():
    print(f"{nome}: {nota}")
```

É como se você estivesse lendo uma lista de nomes e notas e falando para cada aluno a sua nota.

3 Laço `while`

O laço `while` é como ficar jogando uma bola para cima até que ela caia no chão. Continuamos fazendo algo até que uma condição mude.

3.1 Sintaxe Básica

A sintaxe básica do laço `while` é:

```
while condicao:
    # bloco de código
```

Aqui, `condicao` é como uma pergunta que fazemos o tempo todo. Enquanto a resposta for ”sim”, continuamos.

3.2 Exemplo 5: Contagem regressiva

Vamos fazer uma contagem regressiva até o lançamento de um foguete:

```
contador = 5

while contador > 0:
    print(contador)
    contador -= 1
```

É como se estivéssemos contando de 5 até 1, e a cada número, nos aproximamos do lançamento.

3.3 Exemplo 6: Laço `while` com condição de saída

E se você estivesse tentando adivinhar um número até acertar? O laço `while` continua até você adivinhar corretamente:

```
numero_secreto = 7
palpite = 0

while palpite != numero_secreto:
    palpite = int(input("Adivinhe o número: "))

print("Parabéns! Você acertou.")
```

Aqui, é como se você estivesse jogando um jogo de adivinhação. O computador continua perguntando até você acertar.

3.4 Exemplo 7: Evitando loops infinitos

Cuidado para não ficar preso em um loop infinito, onde o laço nunca para. Para evitar isso, podemos usar a palavra `break` para sair do laço antes:

```python
contador = 10

while contador > 0:
    print(contador)
    contador -= 1
    if contador == 2:
        break  # Saindo do loop antecipadamente
```

É como se você estivesse descendo uma escada e, de repente, decide parar no segundo degrau.

4 Iteradores e Iteráveis

Pense nos iteradores como um livro, onde você pode ler uma página de cada vez. Um iterável é como o livro inteiro, e o iterador é a forma de passar por cada página, uma por uma.

4.1 Exemplo 8: Criando um iterador

Vamos criar um iterador manualmente para passar por uma lista de frutas:

```python
frutas = ['maçã', 'banana', 'cereja']
iterador = iter(frutas)

print(next(iterador))  # Saída: maçã
print(next(iterador))  # Saída: banana
print(next(iterador))  # Saída: cereja
```

É como se você estivesse folheando um álbum de fotos, vendo cada fruta uma por uma.

4.2 Exemplo 9: Usando iteradores com `for`

O laço `for` pode fazer todo esse trabalho de folhear para você:

```python
frutas = ['maçã', 'banana', 'cereja']

for fruta in frutas:
    print(fruta)
```

Aqui, o `for` é como alguém virando as páginas do álbum para você.

4.3 Exemplo 10: Iteradores infinitos com `itertools`

E se você tivesse um contador que nunca para? Podemos criar um com o módulo `itertools`:

```
import itertools

contador = itertools.count(start=1, step=2)

for i in range(5):
    print(next(contador))  # Saída: 1, 3, 5, 7, 9
```

Isso é como um relógio que continua contando sem parar, mas aqui estamos olhando apenas para os primeiros 5 números.

5 Conclusão

Hoje, aprendemos sobre laços de repetição em Python, como o `for` e o `while`, que nos ajudam a repetir tarefas. Também vimos como os iteradores são como livros que podemos folhear uma página de cada vez.

6 Exercícios Práticos

Para praticar o que aprendemos, tente resolver esses exercícios:

1. Escreva um código que utilize um laço `for` para imprimir todos os números pares de 1 a 20.
2. Crie um programa que use um laço `while` para solicitar ao usuário que adivinhe um número secreto até que ele acerte.
3. Utilize um `for` para iterar sobre um dicionário e imprimir suas chaves e valores.
4. Escreva um código que converta uma lista de strings em uma lista de inteiros, utilizando um laço `for`.
5. Implemente um exemplo que crie manualmente um iterador e use um laço `while` para percorrê-lo até que todos os elementos sejam consumidos.
6. Usando o módulo `itertools`, crie um iterador infinito que gere números pares e imprima os primeiros 10 números gerados.