

Programação em Python – UC1

Aula 06 – Parte 1

Estruturas de Repetição

ESTRUTURAS DE REPETIÇÃO



As estruturas de repetição (ou **loops**) permitem que se execute um conjunto de instruções **várias vezes** sem precisar escrever o mesmo código repetidamente.

Existem três principais estruturas de repetição em Python: **for**, **while** e **nested loops** (laços aninhados).

ESTRUTURAS DE REPETIÇÃO



Laço de restrição “for”

O for é usado para iterar sobre **uma sequência** (como uma **lista**, **string** ou **range**) e executar um **bloco** de código para cada item.

 Quando usar?

- ◆ Quando se **sabe quantas vezes** quer repetir algo.
- ◆ Para **percorrer** listas, strings ou outros **iteráveis**.

ESTRUTURAS DE REPETIÇÃO



```
for elemento in sequencia:  
    # faça algo com o elemento
```

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

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

ESTRUTURAS DE REPETIÇÃO



Laço de restrição “while”

A estrutura de repetição while é usada quando **não sabemos o número exato de repetições**, mas sabemos a condição que deve ser verdadeira para continuar executando o código.

 Ou seja, enquanto uma condição for verdadeira, o loop continua.

ESTRUTURAS DE REPETIÇÃO



```
while condição:  
    # faça algo
```

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

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

ESTRUTURAS DE REPETIÇÃO

Diferenças principais:

- ◆ **for**: utilizado quando sabemos o número de repetições ou iterando sobre uma sequência.
- ◆ **while**: utilizado quando o número de repetições depende de uma condição.

ESTRUTURAS DE REPETIÇÃO



Laços Aninhados (Nested Loops)

Laços aninhados são **laços dentro de outros laços**.

Úteis para trabalhar com estruturas de dados mais complexas, como listas de listas (matrizes).

ESTRUTURAS DE REPETIÇÃO



```
matriz = [  
    [1, 2, 3],  
    [4, 5, 6],  
    [7, 8, 9]  
]  
  
for linha in matriz:  
    for elemento in linha:  
        print(elemento, end=" ")  
    print()
```

ESTRUTURAS DE REPETIÇÃO



Break e Continue

break: Interrompe o loop completamente.

continue: Pula para a próxima iteração do loop

ESTRUTURAS DE REPETIÇÃO



```
for i in range(10):  
    if i == 5:  
        break      # Para o loop quando i for 5  
    if i % 2 == 0:  
        continue   # Pula números pares  
    print(i)
```

ESTRUTURAS DE REPETIÇÃO



else em loops

Sim, loops podem ter um **else**

Ele é executado **após o loop terminar**, mas só se o loop **não for interrompido** por um **break**

ESTRUTURAS DE REPETIÇÃO




```
for i in range(3):  
    print(i)  
else:  
    print("Loop concluído!")
```

Desafio 6:



DESAFIO 6



O recrutador entregou uma lista de atividades para você executar, relacionadas às estruturas de repetições 

Reflexão

REFLEXÃO



No primeiro desafio você pesquisou e conheceu quais são as exigências de mercado para vaga de programador Full Stack, listou quais os conhecimentos, as habilidades e os pré-requisitos que o mercado está exigindo.

Fazendo uma análise da sua preparação para essa vaga, como você avalia as suas chances, considerando tudo que já desenvolveu até aqui?

O quanto você considera que já evoluiu em conhecimentos, habilidades, atitudes e valores?

REFLEXÃO



O que já trabalhados até o momento:

 os conhecimentos

 as habilidades

 as atitudes

 os valores

Quais você já conseguiu desenvolver ?

Avaliação

AVALIAÇÃO



De acordo com a resolução das atividades dessa aula, o instrutor deverá verificar os Indicadores que os estudantes estão desenvolvendo:

1. Seleciona conceitos da lógica de programação conforme o cenário proposto para a solução.
2. Utiliza estruturas de controle pertinentes conforme a construção de algoritmos.
3. Testa os algoritmos desenvolvidos de acordo com o cenário proposto.