Estruturas de Repetição

Alex Sandro Costa e Everton Freitas Godinho

Definição:

As estruturas de repetição são utilizadas quando queremos que um bloco de código seja executado várias vezes.

Em Python existem duas formas de criar uma estrutura de repetição:

•O for é usado quando se quer iterar sobre um bloco de código um número determinado de vezes.

•O while é usado quando queremos que o bloco de código seja repetido até que uma condição seja satisfeita. Ou seja, é necessário que uma expressão booliana dada seja verdadeira. Assim que ela se tornar falsa, o while para.

Comando for

O **for** é utilizado para **percorrer** ou **iterar** sobre uma sequência de dados (seja esse uma lista, uma tupla, uma string), executando um **conjunto de instruções** em cada item.

Como você já sabe, o Python utiliza **indentação** para separar blocos de código: nos *loops* utilizando **for** não é diferente.

Sua sintaxe básica é: **for** <nome variável> **in** <iterável>. Vamos entender:

•<nome variável> é o nome da variável que vai receber os elemento de <iterável>.

•<iterável> é o container de dados sobre o qual vamos iterar, podendo ser: uma lista, uma tupla, uma string, um dicionário, entre outros.

Comando for

Vamos exemplificar a aplicação do comando para iterar sobre uma lista de números:

```
lista = [1, 2, 3, 4]  # Criamos uma lista de números.

for i in lista:  # Vamos pegar cada valor dentro da lista (i):
    print(i)  # Vamos imprimir na tela cada valor dentro da lista.
```

Vamos entender passo a passo:

•Na primeira iteração, i vai receber o valor do primeiro elemento da lista, que é 1. Portanto print(i) vai mostrar o valor 1.

•Na segunda iteração, i vai receber o valor do segundo elemento da lista, que é 2. Portanto print(i) vai mostrar o valor 2.

•E assim por diante até o último valor, que é 4.

Comando for

A resposta do programa será esse:

```
E:\pythonProject3\venv\Scripts\python.exe E:/pythonProject3/main.py
1
2
3
4

Process finished with exit code 0
```

Comando for

Adicionar o **else** ao final do **for** nos possibilita executar um bloco de código após o iterável ter sido **completamente** percorrido.

```
lista = [1, 2, 3, 4]  # Criamos uma lista de números.

for i in lista:  # Vamos pegar cada valor dentro da lista (i):
    print(i)  # Vamos imprimir na tela cada valor dentro da lista.
else:
    print('Todos os valores foram exibidos com sucesso')
```

```
E:\pythonProject3\venv\Scripts\python.exe E:/pythonProject3/main.py
1
2
3
4
Todos os valores foram exibidos com sucesso

Process finished with exit code 0
```

Comando for

Vamos exemplificar uma lista com valores não-numéricos:

```
lista = ['Alex', 'Everton', 'Simone']  # Criamos uma lista de nomes.

for i in lista:  # Vamos pegar cada valor dentro da lista (i):
    print(i)  # Vamos imprimir na tela cada valor dentro da lista.
else:
    print('Todos os valores foram exibidos com sucesso.')
```

```
E:\pythonProject3\venv\Scripts\python.exe E:/pythonProject3/main.py
Alex
Everton
Simone
Todos os valores foram exibidos com sucesso

Process finished with exit code 0
```

Comando for

Podemos iterar também sobre **dicionários**:

```
lista = {'Arroz': 'R$ 4,35',
         'Feijão': 'R$ 7,89',
         'Açucar': 'R$ 4,12',
         'Óleo': 'R$9,45'}  # Criamos uma dicionário de alimentos com seus respectivos preços.

for i,j in lista.items():  # Vamos pegar cada item dentro da dicionário (i) com seus respectivo valor(j):
    print(f"{i}:{j}")  # Vamos imprimir na tela cada valor dentro do dicionário.
else:
    print('Todos os valores foram exibidos com sucesso.')
```

```
E:\pythonProject3\venv\Scripts\python.exe E:/pythonProject3/main.py
Arroz:R$ 4,35
Feijão:R$ 7,89
Açucar:R$ 4,12
Óleo:R$9,45
Todos os valores foram exibidos com sucesso.

Process finished with exit code 0
```

Comando for

Além disso, podemos iterar também sobre as strings.

```
for i in 'Computacao': # vamos iterar sobre cada letra da palavra Computacao.
    print(i) # Exibe cada letra que é percorrida.
```

```
E:\pythonProject3\venv\Scripts\python.exe E:/pythonProject3/main.py
C
o
m
p
u
t
a
c
a
o
```

Comando While

É uma estrutura de repetição utilizada quando se quer que um determinado bloco de código seja executado **enquanto** determinada condição for satisfeita.
No caso, enquanto a condição não for satisfeita, a repetição se seguirá.

Sua sintaxe básica é:

while <condição>:
        # Código a ser executado.

Vamos exemplificar:

```
valor = 0           # Atribuímos um valor inicial, caso, valor =0
while valor <= 5:   # Atribuição do while diz: enquanto o valor for menor ou igual a 5, repete-se a comando.
    print(valor)    # O valor vai repetir enquanto o while não for satisfeito.
    valor += 1      # O valor atualiza com valor atual mais 1.
```

Comando While

```
E:\pythonProject3\venv\Scripts\python.exe E:/pythonProject3/main.py
0
1
2
3
4
5
```

O resultado do código anterior mostra que:

- O primeiro valor foi zero, ou seja, é menor ou igual a 5. Então o primeiro comando **print(valor)** vai imprimir o valor zero inicial. Em seguida, tem a expressão **valor+=1**, que significa que a variável valor será atualizada para **valor = 1,** pois **valor+=1** é o mesmo que **valor = valor +1**, isto é, **valor = 0+1**.

- Como o **valor** agora é **1**, foi atualizado pela varredura anterior, então **print(valor) = 1**, que será impresso na tela e, em seguida, **valor+=1 = 2**, pois pega o valor atualizado 1 e soma com mais 1.

- Como 2 é o valor atualizado e ainda menor que 5, novamente o comando é executado, onde é impresso na tela e, em seguida, **valor** é atualizado pra **3,** e assim por diante até chegar **valor = 5**, onde termina a repetição.

Comando while

Vamos exibir um exemplo agora acrescentando o comando **else**.

```
valor = 0  # Atribuímos um valor inicial, caso, valor =0
while valor <= 5:  # Atribuição do while diz: enquanto o valor for menor ou igual a 5, repete-se a comando.
    print(f'O valor agora é {valor}.')  # O valor vai repetir enquanto o while não for satisfeito.
    valor += 1  # O valor atualiza com valor atual mais 1.
else:
    print("Os valores foram exibidos com sucesso!")  # Após o término do comando acima, a mensagem será impressa.
```

```
E:\pythonProject3\venv\Scripts\python.exe E:/pythonProject3/main.py
O valor agora é 0.
O valor agora é 1.
O valor agora é 2.
O valor agora é 3.
O valor agora é 4.
O valor agora é 5.
Os valores foram exibidos com sucesso!
```

Auxiliares

Podemos utilizar **3 comandos auxiliares** quando queremos alterar o fluxo de uma estrutura de repetição:

break

continue

pass

break

É usado para finalizar a repetição(loop), ou seja, é usado para parar sua execução.

Vamos dar um exemplo em que percorremos de 1 a 10, mas agora vamos parar no 5.

```
for i in range(10): # Percorrendo de 0 a 9 com range.
    if i == 5: # Se o número for igual a 5, devemos parar o loop com o break abaixo
        break
    else:
        print(i) #Imprimindo os valores de i
```

```
E:\pythonProject3\venv\Scripts\python.exe E:/pythonProject3/main.py
0
1
2
3
4

Process finished with exit code 0
```

break

Vamos dar um exemplo em que temos valores de 0 a 9, onde no 5, o loop para.

```
valor = 0 # Variável valor começa com 0
while valor <= 5: # Enquanto o valor for menor ou igual a 5
    valor += 1 # Valor recebe mais 1 alterando seu valor
    if valor == 4: # Se valor recebe o valor 4, o loop acaba com o break abaixo
        break
    print(valor)
```

```
E:\pythonProject3\venv\Scripts\python.exe E:/pythonProject3/main.py
1
2
3

Process finished with exit code 0
```

continue

Bem semelhante ao **break**, porém ao invés de encerrar o loop, ele pula todo código que estiver abaixo dele(dentro do loop) indo para a próxima iteração.

```python
for i in range(6):  # Contando de 0 a 5
    if i == 3:  # Quando a contagem chegar a 3
        print('Encontrei o 3')  # Imprimir a frase 'Encontrei o 3'
        continue # Segue-se para o else
    else:
        print(i) # Imprimir os valores de i
    print('Estou abaixo do IF')
```

```
E:\pythonProject3\venv\Scripts\python.exe E:/pythonProject3/main.py
0
Estou abaixo do IF
1
Estou abaixo do IF
2
Estou abaixo do IF
Encontrei o 3
4
Estou abaixo do IF
5
Estou abaixo do IF

Process finished with exit code 0
```

pass

O pass nada mais é que uma forma de fazer um código que não realiza operação nenhuma.

Como os escopos de Classes, Funções, If/Else e *loops* for/while são definidos pela indentação do código (e não por chaves {} como geralmente se vê em outras linguagens de programação), usamos o pass para dizer ao Python que o bloco de código está vazio.

```
for i in range(100):
    pass

while False:
    pass


class Classe:
    pass


if True:
    pass

else:
    pass


def funcao():
    pass
```

```
E:\pythonProject3\venv\Scripts\python.exe E:/pythonProject3/main.py

Process finished with exit code 0
```

Observe que o **pass** não anulou os códigos, retornando erros, mas sim que o comando está vazio.

Agora é sua vez!

1. Crie uma lista com 4 nomes e, em seguida, acrescente um nome na lista onde o usuário o digita. Imprima toda a lista de nomes em seguida.
2. Crie uma lista de números de 2 a 20. Dessa lista, imprima os números pares.
3. Crie uma lista com 5 números que o usuário digitar e imprima a lista.
4. Crie uma lista com quantidade infinita de números que o usuário quiser digitar até que ele digite 0. Em seguido será impressa a lista com todos esses números digitados.