ESTRUTURAS DE REPETIÇÃO

for

while

FOR

INTRODUÇÃO

- A estrutura de repetição **for** permite executar um bloco de códigos repetidas vezes até que uma condição seja verdadeira. Na linguagem Python, ela é utilizada para percorrer elementos em sequência, como uma string, uma lista, uma tupla ou objetos iteráveis.
- **Tupla é um tipo de estrutura de dados** utilizada em Python que funciona de modo semelhante a uma lista, entretanto, **com a característica principal de ser imutável**. Podemos utilizar uma tupla de dois elementos, por exemplo, para indicar a sigla do estado em uma posição e o nome dele em outra.

INTRODUÇÃO

- Quando o laço **for** é inicializado, ele executa a instrução ou o bloco de códigos uma vez e utiliza uma referência, que funciona como um índice, para indicar o próximo elemento da sequência.
- Internamente, ele usa as funções **__iter__()** e **__next__()** para que a estrutura de repetição funcione.
- O método **__iter__()**, obtém um objeto iterador;
- O método **__next__()** faz a passagem para o próximo elemento.

SINTAXE

for <item> **in** <conjunto_de_itens>**:**
    <bloco_de_codigo>

- Onde:
  - **item**: corresponde a cada elemento presente na variável que permite a iteração;
  - **conjunto_de_itens**: pode ser uma lista, uma string, uma tupla, um dicionário ou um objeto que permita iterações.

EXEMPLO 1

```
frutas = ['Abacaxi', 'Morango', 'Uva']
for a in frutas:
    print(a)
```

```
Abacaxi
Morango
Uva
```

EXEMPLO 2 – ELEMENTO ITERÁVEL

```python
vogais = "aeiou"
contador = 0
frase = input("Digite uma frase: ")
for caractere in frase:
    if caractere in vogais:
        contador += 1

print(f"A frase '{frase}' contém {contador} vogais.")
```

```
Digite uma frase: cec 1023 programacao estruturada
A frase 'cec 1023 programacao estruturada' contém 11 vogais.
```

OUTRO EXEMPLO

```
lista = "Abacaxi;Morango;Uva"
x = lista.split(";")
for palavra in x:
    print(palavra)
    if (palavra == "Abacaxi"):
        print("acido")
print("Fim")
```

```
Abacaxi
acido
Morango
Uva
Fim
```

INSTRUÇÃO BREAK

- A estrutura de repetição só termina depois de ler o último elemento da variável iterável. Entretanto, é possível modificar essa condição e interromper o loop no meio do caminho. Para isso, utilizamos a instrução **break**, que encerra a execução do loop ao encontrar uma condição específica.

```
for <item> in <conjunto_de_itens>:
    <bloco_de_codigo>
    if <condicao_verdadeira>:
        <outras_instrucoes>
        break
```

TIPO DE DADOS - DICIONÁRIO

- O tipo de dados dicionário é usado para armazenar valores em pares, sendo que o primeiro elemento corresponde à chave e o segundo ao valor.

EXEMPLO 3 - DICIONÁRIO

```
pessoas = [
({'nome': 'Joao', 'cidade': 'Aparecida de Goiania'}),
({'nome': 'Maria', 'cidade': 'Goiania'}),
({'nome': 'Pedro', 'cidade': 'Catalao'})
]
contador = 0
for pessoa in pessoas:
    contador += 1
    print(contador, " - ", pessoa)
    if pessoa['nome'] == "Maria":
        print(pessoa['nome'], "mora em", pessoa['cidade'])
        break
print("Fim")
```

```
1  -  {'nome': 'Joao', 'cidade': 'Aparecida de Goiania'}
2  -  {'nome': 'Maria', 'cidade': 'Goiania'}
Maria mora em Goiania
Fim
```

INSTRUÇÃO CONTINUE

- Instrução **continue** em conjunto com uma validação, que pode ser feita com uma estrutura condicional ou outro laço de repetição, permite pular para o próximo item.

EXEMPLO 4 - CONTINUE

```
pessoas = [({'nome': 'João', 'cidade': 'Aparecida de Goiânia'}),
           ({'nome': 'Maria', 'cidade': 'Goiânia'}),
           ({'nome': 'Pedro', 'cidade': 'Catalão'})]
contador = 0
for pessoa in pessoas:
        contador += 1
        if pessoa['nome'] == 'Maria’:
                continue
        print(contador)
        print(pessoa['nome'], "mora em", pessoa['cidade'])
```

FUNÇÃO RANGE

- A função **range()** é utilizada na estrutura de repetição **for** para permite a execução de um determinado conjunto de instruções pela quantidade de vezes indicadas na função.

- Ela retorna uma série de números consecutivos. Por padrão, ela inicia no número 0 e é incrementada adicionando 1, por exemplo: range(4), retornará o seguinte valor : “0, 1, 2, 3”, pois ao chegar ao número 4, o loop será concluído.

FUNÇÃO RANGE

- Sintaxe:

range(início, parada, incremento)

- onde:
  - **início**: é um valor opcional e corresponde a partir de qual número o range será iniciado;
  - **parada**: é um valor obrigatório e indica o número de parada do range;
  - **incremento**: é opcional e indica o valor que queremos adicionar entre um item e outro.

EXEMPLO 5 – FUNÇÃO RANGE

```
for numero in range(10):
    if numero % 2 == 0:
        print("O número", numero, "é par")
```

EXEMPLO 6 – FUNÇÃO RANGE - INTERVALO

```
for numero in range(10, 21):
    if numero % 2 == 0:
        print("O número", numero, "é par")
```

ELSE

- A estrutura de repetição **for** também pode ser utilizada com a cláusula **else** (opcional).

- Sintaxe:

```
for <item> in <conjunto_de_itens>:
    <bloco_de_codigo>
else:
    <novo_bloco_de_codigo>
```

EXEMPLO 7 - ELSE

```
frutas = ['Abacaxi', 'Morango', 'Uva']
for a in frutas:
        print(a)
else:
        print("Laço de repetição finalizado.")
```

LAÇOS ANINHADOS

- Há situações em que é necessário percorrer outra variável iterável dentro de uma estrutura de repetição. Para isso, utiliza-se um loop dentro do outro.

EXEMPLO 8 - LAÇOS ANINHADOS

```
for coluna1 in range(2, 5):
    print("Tabuada do ", coluna1)
    for coluna2 in range(11):
        print(coluna1, "x", coluna2, " = ", coluna1 * coluna2)
```

WHILE

WHILE

- O comando **while** faz com que um conjunto de instruções seja executado enquanto uma condição é atendida. Quando o resultado dessa condição passa a ser falso, a execução do loop é interrompida.

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

WHILE-ELSE

- Ao final do while podemos utilizar a instrução else. O propósito disso é executar alguma instrução ou bloco de código ao final do loop;
- No loop while, a expressão é testada enquanto for verdadeira. A partir do momento que ela se torna falsa, o código da cláusula else será executado, se estiver presente.

```
contador = 0
while (contador < 5):
        print(contador)
        contador = contador + 1
else:
        print("Fim while!")
```

EXERCÍCIOS

1. Faça um programa que leia 5 números e informe o maior número.
2. Faça um programa que, dado um conjunto de N números, determine o menor valor, o maior valor e a soma dos valores.
3. Desenvolva um programa que faça a tabuada de um número qualquer inteiro que será digitado pelo usuário, mas a tabuada não deve necessariamente iniciar em 1 e terminar em 10, o valor inicial e final devem ser informados também pelo usuário, conforme exemplo abaixo:

   Montar a tabuada de: 5

   Começar por: 4

   Terminar em: 7

   Tabuada de 5 começando em 4 e terminando em 7:

   5 X 4 = 20

   5 X 5 = 25

   5 X 6 = 30

   5 X 7 = 35

Obs: Você deve verificar se o usuário não digitou o final menor que o inicial.

EXERCÍCIOS

4 – Faça um programa que calcule o fatorial de um número inteiro fornecido - pelo usuário. Ex.: 5!=5.4.3.2.1=120. A saída deve ser conforme o exemplo abaixo:

5! = 5 . 4 . 3 . 2 . 1 = 120

5 - A série de Fibonacci é formada pela sequência 1,1,2,3,5,8,13,21,34,55,... Faça um programa capaz de gerar a série até o n−ésimo termo.

6 - Uma academia deseja fazer um censo entre seus clientes para descobrir o mais alto, o mais baixo, o mais pesado e o mais leve, para isto você deve fazer um programa que pergunte a cada um dos clientes da academia seu código, sua altura e seu peso. O final da digitação de dados deve ser dada quando o usuário digitar 0 (zero) no campo código. Ao encerrar o programa também deve ser informados os códigos e valores do cliente mais alto, do mais baixo, do mais pesado e do mais leve, além da média das alturas e dos pesos dos clientes.

REFERÊNCIAS

- HORN, Michele. **Python for: usando loop com essa estrutura de repetição!**. Disponível em: https://blog.betrybe.com/python/python-for/#2. Acesso em: 20 set. 2021.
- PYTHON BRASIL. **Estrutura de Repetição**. Disponível em: https://wiki.python.org.br/EstruturaDeRepeticao. Acesso em: 20 set. 2021.
- DEVMEDIA. **Python: Estrutura de repetição while.** Disponível em: https://www.devmedia.com.br/python-estrutura-de-repeticao-while/38546. Acesso em: 20 set. 2021.