centro universitário facvest
unifacvest

Fábrica de Software

Aula 07

Python

Estruturas de Repetição

Estruturas de repetição são artifícios das linguagens de programação para executar um determinado bloco de código por uma certa quantidade de vezes, seja ela definida (utilizando o for) ou a partir de uma condição (utilizando o while).

WHILE

O `while` é uma estrutura de repetição que permite executar um determinado bloco de código enquanto uma condição for verdadeira. É muito similar ao `if`, com a diferença que o bloco será executado **enquanto** a condição for verdadeira, e não **se** a condição for verdadeira. Para isso, após o comando `while` precisamos definir uma condição que será testada a cada execução do seu loop. O `while` só será finalizado quando essa condição **não** for mais atendida.

```
while <condição>:
        <bloco de comandos1>
else:
        <bloco de comandos2>
```

Nota: Em Python, para indicar o bloco de código pertencente ao while, devemos apenas indentar o código, conforme demonstrado no exemplo.

Vamos ver alguns exemplos…

Ex1: Faça um programa em Python que imprima os 10 primeiros números naturais.

```
lista = 0
while lista <= 10:
    print(lista)
    lista = lista +1
```

```
0
1
2
3
4
5
6
7
8
9
10
```

Note que primeiro atribuímos o valor 0 à variável lista. Então, enquanto lista for menor ou igual a 10 nós vamos mostrr seu valor. Note que após o print, somamos 1 ao valor da lista. Então, na primeira vez que passamos fica assim:

lista = lista + 1

lista = 0 + 1 (note que o valor da lista era zero e agora vai ser 1)

Já na segunda vez, acontece:

lista = lista + 1

lista = 1+1 (o valor de lista era 1 e agora é 2)

Assim será feito até lista ser 10.

Olha o mesmo exemplo mas até 5:

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

```
0
1
2
3
4
5
```

Ex3: Imagine que estamos desenvolvendo um controle de gastos e que, enquanto os gastos não somarem R$ 1000,00, nós poderemos adicionar novas contas. Este é um ótimo exemplo do uso do while, já que o bloco de código que será responsável por incrementar a quantidade dos gastos será executado enquanto a soma de todos os valores não for menor que 1000:

```
gastos = 0
valor_gasto = 0
while gastos < 1000:
    valor_gasto = int(input("Digite o valor do novo gasto: "))
    gastos = gastos + valor_gasto

print(gastos)
```

```
Digite o valor do novo gasto: 250
Digite o valor do novo gasto: 800
1050
```

Ex4: Crie um programa em Linguagem Python que solicite a senha de um usuário e depois, peça pra digitar novamente até que as duas senhas sejam correspondentes.

```
print("Confirmação de senha")
senha1 = input("Digite a senha: ")
senha2 = input("Confirme a senha: ")
while senha1 != senha2:
    print("Senha errada, digite novamente.")
    senha1 = input("Digite a senha: ")
    senha2 = input("Confirme a senha: ")

print("senha confirmada, parabéns!")
```

```
Confirmação de senha
Digite a senha: 125
Confirme a senha: 356
Senha errada, digite novamente.
Digite a senha: 125
Confirme a senha: 125
senha confirmada, parabéns!
```

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.

```
contador = 0
while (contador < 5):
      print(contador)
      contador = contador + 1
else:
      print("O loop while foi encerrado com sucesso!")
```

```
0
1
2
3
4
O loop while foi encerrado com sucesso!
```

Assim como acontece com for/else, declarando o else ao final do while é possível executar um código ao final do loop. Neste caso será impressa a mensagem: “O loop while foi encerrado com sucesso!”.

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.

```
x = 0
while x < 10:
    print(x)
    x += 1
else:
    print("fim while")
```

```
0
1
2
3
4
5
6
7
8
9
fim while
```

Se dentro da repetição for executado um break, o loop será encerrado sem executar o conjunto da cláusula else.

```
x = 0
while x < 10:
    print(x)
    x += 1
    if x == 6:
        print("x é igual a 6")
        break
else:
    print("fim while")
```

```
0
1
2
3
4
5
x é igual a 6
```

FOR

O laço **for** nos permite percorrer os itens de uma coleção e, para cada um deles, executar o bloco de código declarado no loop. Sua sintaxe é a seguinte:

for var in lista:
    <bloco de comandos1>
eles:
    <bloco de comandos2>

Diferente do while, o `for` executará um determinado bloco de código por um número definido de vezes. Esta estrutura é muito útil quando já sabemos a quantidade de vezes que precisamos executar determinado bloco de código. Diferente da maioria das linguagens, para criar um intervalo de vezes que o `for` será executado, precisamos utilizar a função `range` e definir o intervalo, como podemos ver abaixo:

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

```
1
2
3
4
```

Ex2:

```
nomes = ['Pedro', 'João', 'Leticia']
for n in nomes:
    print(n)
```

```
Pedro
João
Leticia
```

Ex3:

```
# Para iterar sobre uma lista:
lista = [1, 2, 3, 4, 10]

for numero in lista:

    print(numero ** 2)
```

```
1
4
9
16
100
```

Ex4:

```
# Para cada letra na palavra, imprimir a letra

palavra = "casa"

for letra in palavra:

    print(letra)
```

```
c
a
s
a
```

Ex5:

```
gatinhos = {"Português": "gato", "Inglês": "cat", "Francês": "chat", "Finlandês": "Kissa"}

for chave, valor in gatinhos.items():

    print(chave, "->", valor)
```

```
Português -> gato
Inglês -> cat
Francês -> chat
Finlandês -> Kissa
```

Observe: A variável definida na linha 1 é uma lista inicializada com uma sequência de valores do tipo string. A instrução for percorre todos esses elementos, um por vez e, em cada caso, atribui o valor do item à variável n, que é impressa em seguida. O resultado, então, é a impressão de todos os nomes contidos na lista

FOR/ELSE

É possível adicionar a instrução else ao final do for. Isso faz com que um bloco de código seja executado ao final da iteração, como mostra o exemplo a seguir:

```
nomes = ['Pedro', 'João', 'Leticia']
for n in nomes:
    print(n)
else:
    print("Todos os nomes foram listados com sucesso")
```

```
Pedro
João
Leticia
Todos os nomes foram listados com sucesso
```

Na primeira linha definimos uma variável que armazenará uma lista de nomes. Após isso, a instrução for percorre todos esses elementos e atribui um a um à variável n, que será impressa, como pode ser visto na linha 3. Após o loop se encerrar, o bloco de código contido na instrução else é acionado, imprimindo a mensagem na tela.

EXERCÍCIOS

1) Faça um programa que peça um número qualquer e diga se ele é um número par ou ímpar.

2) Peça ao usuário 5 valores e diga qual é o maior e qual é o menor deles.

3) Peça ao usuário um número e indique se o número é par ou ímpar, se é divisível por 5 ou por 10.

4) Leia um número fornecido pelo usuário. Se esse número for positivo, calcule a raiz quadrada do número. Se o número for negativo, mostre uma mensagem dizendo que o número é inválido.

5) Faça um programa que leia 2 notas de um aluno, verifique se as notas são válidas e exiba na tela a média destas notas. Uma nota válida deve ser, obrigatoriamente, um valor entre 0.0 e 10.0, onde caso a nota não possua um valor válido, este fato deve ser informado ao usuário e o programa termina.

6) Leia o salário de um trabalhador e o valor da prestação de um empréstimo. Se a prestação for maior que 20% do salário imprima: Empréstimo nâo concedido, caso contrário imprima: Empréstimo concedido