Estruturas de controle em Python

DigitalHouse >
Coding School

Certified Tech Developer
The Ultimate Degree

Índice

1 | Indentação (sangria)

Indentação (indentation) em python.

Em Python, as linhas de código que estão dentro de um mesmo código devem ser agrupadas, tendo o mesmo número de espaços à esquerda de cada linha, assim como seriam na vida real.

Como um exemplo:

```
-  Carrefour
      - Açougue
          - Carne
          - Frango
      - Peixaria
```

Indentação (indentation) em python.

Este próximo caso não estaria correto e em Python geraria um erro (ou a operação não seria como o esperado):

```
    - Fruteira
       - Peras
    - Maçãs
```

Lógicamente, as Maçãs **não podem estar no mesmo nível** que Fruteira.

Em Python, recomenda-se sempre usar blocos de quatro espaços, embora qualquer outro número de espaços também funcione. As abas também podem ser usadas, embora seja recomendável não usá-las.

2 | if

if em Python

Em todo programa, chega uma hora que, dependendo de uma determinada condição, você tem que fazer uma série de coisas ou outra.

Isso é feito com o comando if (condição principal), com o opcional elif (condições adicionais, você pode colocar quantos quiser) e else (se nenhum dos itens acima foi atendido, você pode colocá-lo apenas uma vez e no fim).

Exemplo:

```
>>> Posicao_de_Alonso=1
>>> if (Posicao_de_Alonso==1):
>>>     print("Espetacular Alonso, a justiça foi feita apesar do carro")
>>>     print("Falta menos para vencer o campeonato")
>>> elif (Posicao_de_Alonso>1):
>>>     print("Grande corrida para Alonso, pena que o carro não está à altura da tarefa")
>>> else:
>>>     print("Ele não conseguiu terminar a corrida devido a uma falha mecânica")
```

3 | Condições

Condições em Python

As condições mais utilizadas são:

a == b ➡ Indica se a é igual a b

a < b

a > b

not ➡ NÃO: Negue a condição a seguir.

and ➡ E: Ele tem duas condições que ambos devem ser atendidos.

or ➡ OU: Existem duas condições e uma das duas deve ser cumprida.

4 | Laços, while and for

while

Às vezes, temos que repetir uma determinada tarefa várias vezes até atingirmos nosso objetivo.

Em Python, isso é feito com o comando while. Com o while, é preciso ter cuidado para não realizar um "loop infinito", que consiste em um loop que nunca termina devido a um erro de programação. No caso a seguir, isso aconteceria se não tivéssemos colocado a linha de volta=voltar+1.

while

Como exemplo, enquanto em Python é usado assim:

```
>>> while volta<10:
>>>     print("volta "+str(volta))
>>>     volta=volta+1
volta 1
volta 2
volta 3
volta 4
volta 5
volta 6
volta 7
volta 8
volta 9
```

for

Às vezes, temos que repetir uma determinada tarefa várias vezes até atingirmos nosso objetivo.

Em Python isso é feito com o comando for. No caso de for, não é possível realizar um loop infinito.

Como você pode ver no exemplo a seguir, range gera uma sequência de números de 1 a 10, e for pode ser usado com qualquer objeto que possa iterar (pular de elemento para elemento).

for

Como exemplo para em Python é usado assim:

```
>>> for volta in range(1,10):
>>>     print("volta "+str(volta))
volta 1
volta 2
volta 3
volta 4
volta 5
volta 6
volta 7
volta 8
volta 9
```

for

O **for** pode ser usado com qualquer objeto com que se possa iterar (ir saltando de elemento em elemento), como vemos neste exemplo com uma lista:

```
>>> carros = (‘Ferrari’, ‘Tesla’, ‘BMW’, ‘Audi’)
>>> for coche in carros:
>>>     print(carros)

Ferrari
Tesla
BMW
Audi
```

for

Se combinarmos com a função numérica, ela também dará um número para cada elemento:

```
>>> carros= ('Ferrari', 'Tesla', 'BMW', 'Audi')
>>> for i, carros in enumerate(carros):
>>>    print(str(i) + " - "+ carros)

0 - Ferrari
1 - Tesla
2 - BMW
3 - Audi
```


DigitalHouse>
Coding School