Estructuras de control en Python

DigitalHouse >
Coding School

Certified Tech Developer
The Ultimate Degree

Índice

1 Indentación (sangría)

Indentación (sangría) en python.

En Python, las líneas de código que están dentro de un mismo deben estar agrupadas, teniendo el mismo número de espacios a la izquierda de cada línea, al igual que sucedería en la vida real.

A modo de ejemplo:

```
-  Carrefour
      -Carnicería
          -Cerdo
          - Pollo
      - Pescadería
```

Indentación (sangría) en python.

Este siguiente caso no sería correcto, y en Python generaría un error (o el funcionamiento no sería el esperado):

```
    - Frutería
       - Peras
    - Manzanas
```

Lógicamente, `Manzanas` **no puede estar al mismo nivel** que `Frutería`.

En Python, se recomienda usar siempre bloques de cuatro espacios, aunque si se usan otro número de espacios, también funcionará. También se pueden usar tabuladores, aunque se recomienda no usarlos.

2 | if

if en Python

En todo programa llega el momento, que en función de una determinada condición, hay que realizar una serie de cosas u otra.

Esto se hace con el comando if (condición principal), con los opcionales elif (condiciones adicionales, se pueden poner tantas como se quiera) y else (si no se ha cumplido ninguna de las anteriores, solo se puede poner una vez y al final).

A modo de ejemplo:

```
>>> Alonso_Position=1
>>> if (Alonso_Position==1):
>>>     print("Espectacular Alonso, se ha hecho justicia a pesar del coche")
>>>     print("Ya queda menos para ganar el mundal")
>>> elif (Alonso_Position>1):
>>>     print("Gran carrera de Alonso, lástima que el coche no esté a la altura")
>>> else:
>>>     print("No ha podido terminar la carrera por una avería mecánica")
```

3 | Condiciones

Condiciones en Python

Las condiciones que se suelen usar con más frecuencia son:

a == b ➡ Indica si a es igual a b

a < b

a > b

not ➡ NO: Niega la condición que le sigue.

and ➡ Y: Junta dos condiciones que tienen que cumplirse las dos

or ➡ O: Junta dos condiciones y tienen que cumplirse alguna de las dos.

4 | Bucles while y for

while

En ocasiones, tenemos que repetir varias veces una determinada tarea hasta conseguir nuestro objetivo.

En Python esto se realiza con el comando while. Con los while, hay que tener la precaución de no realizar un «bucle infinito», que consiste en un bucle que nunca termina por un error en la programación. En el caso siguiente, esto ocurriría si no hubiéramos puesto la línea vuelta=vuelta+1.

while

A modo de ejemplo while en Python se usa así:

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

for

En ocasiones, tenemos que repetir varias veces una determinada tarea hasta conseguir nuestro objetivo.

En Python esto se realiza con el comando for. En el caso del for, no es posible realizar un bucle infinito.

Como se puede ver en el siguiente ejemplo, range genera una secuencia de números desde 1 hasta 10. for se puede utilizar con cualquier objeto con el que se pueda iterar (ir saltando de elemento en elemento).

for

A modo de ejemplo for en Python se usa así:

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

for

for se puede utilizar con cualquier objeto con el que se pueda iterar (ir saltando de elemento en elemento), como vemos en este ejemplo con una lista:

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

Ferrari
Tesla
BMW
Audi
```

for

Si lo combinamos con la función enumerate, además irá dándole un número a cada elemento:

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

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


DigitalHouse>
Coding School