Condicionales y ciclos en Python

Condicionales *if else*

Nos encontramos ante el tipo de **condicional más fundamental y básico**. Se trata de ejecutar un fragmento de código si se cumple una condición dada y, si no se cumple dicha condición, no se ejecuta el fragmento. Para ello utilizamos la sentencia if con el formato del siguiente ejemplo:

```
valor = 10
if valor > 5:
     print('Se cumple la condición')
```

Fíjate en **dos cosas muy importantes**. La primera es que tras la condición debemos poner dos puntos (:). La segunda es que el bloque de código que queremos que se ejecute debe estar correctamente *indentado* o *sangrado*, es decir las líneas de código deben empezar más a la derecha que la línea de if. En Python se define un tamaño de tabulación de cuatro espacios.

Al ejecutarse el código anterior, como la condición es cierta, se nos mostrará por pantalla el mensaje «*Se cumple la condición*«.

Si queremos que se ejecute un código alternativo en caso de que la condición no se cumpla, es decir, de que sea False, podemos crear una nueva rama alternativa mediante la sentencia else de la siguiente manera:

```
valor = 10
if valor > 15:
     print('Se cumple la condición')
else:
     print('No se cumple la condición')
```

Fíjate que la palabra else se escribe al mismo nivel (en vertical) que el if y el código de la nueva rama también debe escribirse *indentado*. La ejecución del código anterior nos mostrará por pantalla el mensaje «*No se cumple la condición*» únicamente.

Condicionales *if elif else*

Si necesitamos hacer más de dos ramas alternativas podemos escribirlas encadenando sentencias elif, que es una contracción de else e if, y proporcionando **distintas condiciones** de la siguiente manera:

```
valor = 9
if valor < 5:
     print('Es menor que 5')
elif valor < 10:
     print('Es menor que 10')else:    print('Es mayor o igual que 10')
```

Este código imprimirá por pantalla, como ya te imaginas, el mensaje «*Es menor que 10*» y nada más.

Esta estructura nos permite ejecutar una única rama de todas las definidas, es decir, que son **excluyentes entre sí**.

Bucle *while*

El bucle while (*mientras*) de Python es el **bucle típico de casi cualquier lenguaje de programación**. Permite repetir un fragmento de código una y otra vez **mientras** se cumpla una condición dada. El fragmento de código que se repite en un bucle es lo que se conoce como **cuerpo del bucle**.

La sintaxis es muy sencilla como podemos ver en el siguiente ejemplo y sigue unas reglas parecidas a las explicadas previamente para el if:

```
valor = 0
while (valor < 5):
     print(f'valor {valor}')
     valor += 1 #nos aseguramos dentro del bucle de que en algún momento la condición se haga falsa
```

Fíjate que **basta con poner una condición que sea cierta y que en algún momento se haga falsa** para detener el bucle. Recuerda que un buen truco para entender este bucle es pensar *el código se repite **mientras** la condición sea cierta*.

Bucle *for*

El bucle for en Python es muy diferente de los bucles for que nos encontramos habitualmente en otros lenguajes de programación. No es el objetivo de este artículo ahondar en esto, pero sí es interesante mencionarlo porque **la condición de este bucle es implícita**, no hace falta indicarla.

Normalmente **se utiliza para iterar por todos y cada uno de los elementos de una secuencia**, como puede ser una lista, una *tupla*, un diccionario, range, cadenas de texto, etc.

De esta manera, la condición implícita de un bucle for nos permite ejecutar un código **tantas veces como elementos tenga la secuencia**. Además, nos proporciona cada uno de esos elementos en una variable. Lo vemos en un ejemplo:

```
lista = [1, 2, 3]
for elemento in lista:
     print(elemento)
```

Fíjate que el cuerpo del bucle se ejecuta tantas veces como elementos tiene la lista. Además, en cada vuelta del bucle tenemos en la variable elemento el valor de cada uno de los elementos de la lista.

La ventaja de este tipo de bucles es que nos podemos olvidar de utilizar índices y, por tanto, de escribir condiciones que siempre tienden a ser iguales al hacer este tipo de recorridos.

Bucles con rama *else*

Una cosa ciertamente extraña, y que no conoce mucha gente, de los bucles en Python es que permiten añadirles una rama else. **En dicha rama podemos escribir un fragmento de código que se ejecuta cuando la condición del bucle (ya sea implícita o explícita) se hace falsa**, eso sí, siempre y cuando el bucle no haya terminado a causa de una sentencia break.

Un break es una instrucción de detiene de manera inmediata la ejecución del bucle, aunque la condición del mismo sea True, y permite al resto del código posterior al bucle continuar con su ejecución.

La sentencia else se puede utilizar tanto en bucles for como en bucles while. Veamos un ejemplo:

```
lista = [1, 2, 3]
for elemento in lista:
     print(elemento)else:
     print('Hemos llegado al final del bucle')
```

Tienes que darte cuenta de que el mensaje final, que se imprime en la rama else del bucle se muestra por pantalla porque el bucle ha terminado sin un break. Veamos a continuación este caso:

```
lista = [1, 2, 3]
for elemento in lista:
     print(elemento)
     if elemento == 3:
          break
else:
     print('Hemos llegado al final del bucle')
```

Una cosa ciertamente extraña, y que no conoce mucha gente, de los bucles en Python es que permiten añadirles una rama else. **En dicha rama podemos escribir un fragmento de código que se ejecuta cuando la condición del bucle (ya sea implícita o explícita) se hace falsa**, eso sí, siempre y cuando el bucle no haya terminado a causa de una sentencia break.

Un break es una instrucción de detiene de manera inmediata la ejecución del bucle, aunque la condición del mismo sea True, y permite al resto del código posterior al bucle continuar con su ejecución.

La sentencia else se puede utilizar tanto en bucles for como en bucles while. Veamos un ejemplo:

```
lista = [1, 2, 3]
for elemento in lista:
     print(elemento)
else:
     print('Hemos llegado al final del bucle')
```

Tienes que darte cuenta de que el mensaje final, que se imprime en la rama else del bucle se muestra por pantalla porque el bucle ha terminado sin un break. Veamos a continuación este caso:

```
lista = [1, 2, 3]
for elemento in lista:
     print(elemento)
     if elemento == 3:
          break
else:
     print('Hemos llegado al final del bucle')
```

En este caso, el bucle termina a causa del break, por lo que el print con el mensaje final **no se ejecuta**, pues no hemos dado tiempo a que se reevalúe la condición para comprobar que es falsa. Por tanto, el resultado es el siguiente:

De la misma manera funciona para un bucle while. Vemos un pequeño ejemplo:

```
n = 10
while n < 20:
     print(n)
     n += 1
     if n % 7 == 0:
          break
else:
     print('Hemos llegado al final del bucle')
```

Bibliografía:

- Extraído de <<Condicionales en python: mucho más que if else>>. Consultado el 28 de noviembre del 2022