Introducción a la programación con Python

Booleanos y estructuras condicionales

Booleanos

Una variable booleana es una variable que sólo puede tomar dos posibles valores:

True (verdadero) o False (falso).

En Python cualquier variable (en general, cualquier objeto) puede considerarse como una variable booleana. En general los elementos nulos o vacíos se consideran False y el resto se consideran True.

Para comprobar si un elemento se considera True o False, se puede convertir a su valor booleano mediante la función bool().

Operadores lógicos

Los operadores lógicos son unas operaciones que trabajan con valores booleanos.

Operador lógico “and”

"y" lógico. Este operador da como resultado True si y sólo si sus dos operandos son True:

True and True	->	True
True and False	->	False
False and True	->	False
False and False	->	False

Operador lógico “or"

"o" lógico. Este operador da como resultado True si algún operando es True:

True or True	->	True
True or False	->	True
False or True	->	True
False or False	->	False

Operador lógico “not”

Negación. Este operador da como resultado True si y sólo si su argumento es False:

not True -> False

not False -> True

Expresiones lógicas compuestas

Al componer expresiones más complejas hay que tener en cuenta que Python evalúa primero los not, luego los and y por último los or, como puede comprobarse en los ejemplos siguientes:

El operador not se evalúa antes que el operador and:

not True and False -> False

(not True) and False -> False

not (True and False) -> True

Expresiones lógicas compuestas

El operador not se evalúa antes que el operador or:

not False or True	->	True
(not False) or True	->	True
not (False or True)	->	False

El operador and se evalúa antes que el operador or:

False and True or True	->	True
(False and True) or True	->	True
False and (True or True)	->	False

Comparaciones

Las comparaciones también dan como resultado valores booleanos:

> Mayor que; < Menor que;

3 > 2 -> True

3 < 2 -> False

>= Mayor o igual que; <= Menor o igual que;

2 >= 1 + 1 -> True

4 - 2 <= 1 -> False

== Igual que; != Distinto de;

2 == 1 + 1 -> True

6 / 2 != 3 -> False

Sentencias condicionales

La estructura de control “if” permite que un programa ejecute unas instrucciones cuando se cumpla una condición.

En inglés "if" significa "si" (condición).

La sintaxis de la construcción if es la siguiente:

if condición:

    aquí van las órdenes que se ejecutan si la condición es cierta

    y que pueden ocupar varias líneas

Sentencias condicionales

La ejecución de esta construcción es la siguiente:

La condición se evalúa siempre.

- Si el resultado es True se ejecuta el bloque de sentencias
- Si el resultado es False no se ejecuta el bloque de sentencias.

La primera línea contiene la condición a evaluar y es una expresión lógica. Esta línea debe terminar siempre por dos puntos (:).

A continuación viene el bloque de órdenes que se ejecutan cuando la condición se cumple (es decir, cuando la condición es verdadera). Es importante señalar que este bloque debe ir sangrado, puesto que Python utiliza el sangrado para reconocer las líneas que forman un bloque de instrucciones. El sangrado que se suele utilizar en Python es de cuatro espacios, pero se pueden utilizar más o menos espacios. Para terminar un bloque, basta con volver al principio de la línea.

Ejemplo de sentencia if

Si el número introducido no es positivo (menor que 0) muestra un mensaje de aviso.

Si el número es positivo no muestra el mensaje.

En ambos casos se muestra el número introducido al finalizar.

```
numero = int(input("Escriba un número positivo: "))
if numero < 0:
    print("¡Le he dicho que escriba un número positivo!")
print(f"Ha escrito el número {numero}")
```

Bifurcaciones: if ... else ...

La estructura de control if ... else ... permite que un programa ejecute unas instrucciones cuando se cumple una condición y otras instrucciones cuando no se cumple esa condición. En inglés "if" significa "si" (condición) y "else" significa "si no".

Bifurcaciones: if ... else ...

La sintaxis de la construcción if ... else ... es la siguiente:

```
if condición:
    aquí van las órdenes que se ejecutan si la condición es cierta
    y que pueden ocupar varias líneas
else:
    y aquí van las órdenes que se ejecutan si la condición es
    falsa y que también pueden ocupar varias líneas
```

Bifurcaciones: if ... else ...

La ejecución de esta construcción es la siguiente:

La condición se evalúa siempre.

- Si el resultado es True se ejecuta solamente el bloque de sentencias 1
- Si el resultado es False se ejecuta solamente el bloque de sentencias 2.

Bifurcaciones: if … else …

Se introduce la edad por teclado.

Si es menor de 18, el programa mostrará el mensaje “Es usted menor de edad”

Si no, mostrará “Es usted mayor de edad”

En cualquiera de los casos, se despide.

```
edad = int(input("¿Cuántos años tiene? "))
if edad < 18:
    print("Es usted menor de edad.")
else:
    print("Es usted mayor de edad.")
print("¡Hasta la próxima!")
```

if ... elif ... else

La construcción if ... else ... se puede extender añadiendo la instrucción elif:

La estructura de control if ... elif ... else ... permite encadenar varias condiciones. elif es una contracción de else if. La orden en Python se escribe así:

```
if condición_1:
    bloque 1
elif condición_2:
    bloque 2
else:
    bloque 3
```

if ... elif ... else

Si se cumple la condición 1, se ejecuta el bloque 1

Si no se cumple la condición 1 pero sí que se cumple la condición 2, se ejecuta el bloque 2

Si no se cumplen ni la condición 1 ni la condición 2, se ejecuta el bloque 3.

Se pueden escribir tantos bloques elif como sean necesarios. El bloque else (que es opcional) se ejecuta si no se cumple ninguna de las condiciones anteriores.