Unidad 2.3

Booleanos

1. Introducción

Los booleanos son un tipo de dato fundamental en Python y en la mayoría de los lenguajes de programación. Representan valores de verdadero o falso, esenciales para la toma de decisiones y el control de flujo en los programas. En Python, los valores booleanos están representados por las palabras reservadas True y False, que son equivalentes a los valores numéricos 1 y 0, respectivamente.

Los booleanos se utilizan principalmente en expresiones lógicas y condicionales, como en instrucciones if, bucles y comparaciones. Por ejemplo, podemos usar un booleano para determinar si una condición se cumple (True) o no (False), lo que permite que el programa tome decisiones en función de estas evaluaciones.

Con los diferentes operadores lógicos como and, or, y not para combinar o invertir valores booleanos, así como operadores de comparación (==, !=, <, >, etc.) podremos combinar expresiones para diferentes puntos en nuestros programas.

1.1. Creación

Hay diferentes formas de crear booleanos:

1. **Asignación directa:**

Puedes asignar directamente los valores booleanos True o False a una variable:

```
verdadero = True
falso = False

print(verdadero,falso)
```

```
True False
```

2. A partir de una comparación:

Puedes asignar el resultado de una expresión lógica o comparación:

```
es_igual = (5 == 5)
es_diferente = (3 == 7)
es_mayor = (10 > 5)
es_menor = (2 < 1)

print(es_igual)
print(es_diferente)
print(es_mayor)
print(es_menor)
```

```
True
False
True
False
```

3. Usando operadores lógicos

Puedes combinar valores booleanos con operadores lógicos como and, or y not:

```
condicion = True and False
otra_condicion = True or False
negacion = not True
print(condicion)
print(otra_condicion)
print(negacion)
```

```
False
True
False
```

4. Usando funciones que devuelven valores booleanos

Algunas funciones como las que hemos visto en los anteriores temas devuelven un valor booleano, como .isalpha() o .startswith().

5. Evaluando expresiones con bool()

Puedes usar la función bool() para convertir cualquier valor o expresión en un valor booleano. Son False 0 o cadenas y conjuntos vacíos, si una variable u objeto tiene valor (y a no ser que tenga un método bool() propio), Python lo interpretará como True. Esto se utiliza por ejemplo en estructuras if, para hacer algo solo si la variable/objeto tienen algún valor. Tenéis más información en la documentación de Python.

1.2. Operadores lógicos y expresiones

Como habéis podido ver en los apartados de creación, hay unos operadores (>,<,=,!) y unas expresiones (and, or, not) utilizados con valores booleanos y especialmente útiles a la hora de construir nuestro código.

Tabla Resumen

Tipo de Operador	Operador	Descripción	Ejemplo	Resultado
Comparación	==	Verifica si dos valores son iguales	5 == 5	True
	!=	Verifica si dos valores son diferentes	5 != 3	True
	>	Verifica si el valor de la izquierda es mayor que el de la derecha	10 > 5	True
	<	Verifica si el valor de la izquierda es menor que el de la derecha	2 < 5	True
	>=	Verifica si el valor de la izquierda es mayor o igual al de la derecha	7 >= 7	True
	<=	Verifica si el valor de la izquierda es menor o	4 <= 5	True

Tipo de Operador	Operador	Descripción	Ejemplo	Resultado
		igual al de la derecha		
Lógicos	and	Devuelve True si ambas condiciones son verdaderas	(5 > 2) and (3 < 4)	True
	or	Devuelve True si al menos una condición es verdadera	(5 > 10) or (3 < 4)	True
	not	Invierte el resultado de una condición	not (5 > 2)	False
Membership	in	Verifica si un elemento está presente en una secuencia	'a' in 'hola'	True
	not in	Verifica si un elemento no está presente en una secuencia	'z' not in 'hola'	True
Identidad	is	Verifica si dos referencias apuntan al mismo objeto en memoria	a = b = [1, 2]; a is b	True
	is not	Verifica si dos referencias no apuntan al mismo objeto	a = [1, 2]; b = [1, 2]; a is not b	True

Estas expresiones se pueden combinar usando and y or (como se puede ver en los ejemplos) para que nuestro código compruebe determinadas condiciones. Por ejemplo, si necesitamos ver si alguien puede conducir o no:

```
edad1 = 20
tieneCarnet1 = True

if (edad1>=18) and tieneCarnet1:
    print("Puede conducir")
else:
    print("No puede conducir")
```

```
Puede conducir
```

Es muy muy importante entender y aprender a construir correctamente las expresiones y sus combinaciones. Gran parte del funcionamiento de los programas recae muchas veces en la construcción de estas expresiones, por lo que no entenderlas puede provocar errores que signifiquen el error completo del programa.