Variables booleanas y operadores de decisión

Booleanos

Dato booleano. Es un tipo de dato que solamente puede tomar 2 valores: `True` (verdadero) o `False` (falso).

Variable lógica. Variable que almacena datos booleanos.

```
is_adult = True
type(is_adult)
```

```
bool
```

Observación. Observad que tanto `True` como `False` únicamente tienen la primera letra mayúscula. Además, a diferencia de otros lenguajes de programación, como por ejemplo `R`, `Python` solamente admite los booleanos escritos de esta forma: `True` o `False`.

Tablas de verdad

Dadas dos variables lógicas, A y B, , podemos definir los operadores básicos mediante tablas de verdad, donde el valor verdadero se representa con la letra V o bien con un 1, mientras que el valor falso se representa mediante la letra F o bien con un 0.

La tabla de verdad para la variable A sería

A
V
F

La tabla de verdad para la variable B sería

B
V
F

Negación

El operador negación aplicado a una variable se representa con $\neg$ y devuelve el valor contrario.

A	$\neg A$
V	F

V	F
F	V

Conjunción

La conjunción entre dos variables se representa con $\wedge$ y devuelve verdadero únicamente cuando ambas variables valen verdadero.

A	B	$A \wedge B$
V	V	V
V	F	F
F	V	F
F	F	F

Disyunción

La disyunción entre dos variables se representa con $\vee$ y devuelve verdadero cuando almenos una de las variables lógicas vale verdadero.

A	B	$A \vee B$
V	V	V
V	F	V
F	V	V
F	F	F

Operadores lógicos en `Python`

Para hacer la negación, utilizamos el operador `not`.

```
A = True
not A
```

```
False
```

```
B = False
not B
```

```
True
```

Para hacer la conjunción entre dos variables lógicas, utilizamos el operador `and`.

```
A, B = True, True
A and B
```

```
True
```

```
A and (not B)
```

```
False
```

Para hacer la disyunción entre dos variables lógicas, utilizamos el operador `or`.

```
A, B = False, False
A or B
```

```
False
```

```
(not A) or B
```

```
True
```

Operadores de comparación

En `Python` podemos comparar datos y obtener un resultado booleano. Los operadores de comparación disponibles son

Operador	Significado
$>$	Estrictamente mayor
$\geq$	Mayor o igual
$<$	Estrictamente menor
$\leq$	Menor o igual
$==$	Igual
$!=$	Diferente

```
7 == 7.0
```

```
True
```

```
3.14 > 9
```

```
False
```

```
7 != "7"
```

```
True
```

```
0.01 <= 1
```

```
True
```

Observación. Observad que cuando hemos comparado el número 7 en formato integer y en formato float, hemos obtenido que eran iguales, mientras que al haber comparado el número 7 en formato integer con el mismo número, pero en formato string, nos ha devuelto que son diferentes. Esto lo que nos viene a decir es que numéricamente, `Python` considera iguales los números enteros tanto si están en formato integer como en formato float. No obstante, nunca considerará iguales dos datos donde uno esté en formato numérico y el otro, en formato string.

Múltiples comparaciones simultáneas

Podemos realizar múltiples comparaciones a la vez.

Supongamos que tenemos que tener 16 años o más, pero como mucho 40 para poder concertar una entrevista y aspirar a ser miembros de la tripulación del pirata Pyratilla.

Queremos saber si nos concederá una entrevista si tenemos 17 años.

```
edad = 17
(edad >= 16) and (edad <= 40)
```

```
True
```

Hemos obtenido `True`, por tanto podemos concertar una entrevista. Que nos admita o no en su tripulación tras la entrevista, eso ya es otra cosa.

Comparaciones de strings

No solamente podemos comparar datos numéricos, sino que también podemos comparar strings en relación al orden alfabético.

```
"Mallorca" < "Dubai"
```

```
False
```

El resultado que obtenemos no se debe a que Mallorca sea mejor que Dubai, sino a que la primera letra de la primera palabra, M, no se encuentra antes en el abecedario que la primera letra de la segunda palabra, D.

Observación. En caso de que la primera letra de cada una de las palabras comparadas coincidan, se comparan los caracteres que se encuentran en la siguiente posición. En caso de empate, seguiríamos comparando los caracteres de la tercera posición y así sucesivamente.

```
"Mallorca" >= "Madrid"
```

```
True
```

Más métodos de string

El método `.startswith()` nos devuelve verdadero si el string empieza con el caracter o la cadena de caracteres indicado.

```
s = "Mallorca es una isla preciosa"
s.startswith("m")
```

```
False
```

```
s.startswith("Mallorca")
```

```
True
```

Observación. Como podéis observar, `Python` diferencia entre letras en mayúscula y letras en minúscula. Por eso es que obtenemos falso al preguntar si el string empieza por m, en vez de M.

El método `.endswith()` nos devuelve verdadero si el string acaba con el caracter o la cadena de caracteres indicado.

```
s = "Mallorca es una isla preciosa"
s.endswith("a")
```

```
True
```

```
s.endswith("bonita")
```

```
False
```

El método `.isalnum()` nos devuelve verdadero si todos los caracteres del string son alfanuméricos.

```
s = "Python365"
s.isalnum()
```

```
True
```

```
s = "Han creado un blog llamado Python365"
s.isalnum()
```

```
False
```

¡Cuidado! No se consideran caracteres alfanuméricos los siguientes: espacio en blanco, !, %, ?, & y un largo etcétera.

El método `.isalpha()` nos devuelve verdadero si todos los caracteres del string son del alfabeto.

```
s = "Cachalote"
s.isalpha()
```

```
True
```

```
s = "Mi perro se llama Guindilla"
s.isalpha()
```

```
False
```

El método `.isdigit()` nos devuelve verdadero si todos los caracteres del string son dígitos.

```
s = "365"
s.isdigit()
```

```
True
```

```
s = "Pyo365"
s.isdigit()
```

```
False
```

El método `.isspace()` nos devuelve verdadero si todos los caracteres del string son espacios en blanco.

```
s = "                  "
s.isspace()
```

```
True
```

El método `.islower()` nos devuelve verdadero si todos los caracteres del string están en minúscula.

```
s = "Mi gato se llama Bigotes"
s.islower()
```

```
False
```

```
s = "me gusta hacer puzzles"
s.islower()
```

```
s.islower()
```

```
True
```

El método `.isupper()` nos devuelve verdadero si todos los caracteres del string están en mayúscula.

```
s = "Mi gato se llama Bigotes"
s.isupper()
```

```
False
```

```
s = "ME GUSTA HACER PUZZLES"
s.isupper()
```

```
True
```

El método `.istitle()` nos devuelve verdadero si todas las palabras del string empiezan en mayúscula y el resto de las letras de la palabra están en minúscula.

```
s = "Platero Y Yo"
s.istitle()
```

```
True
```

```
s = "PLATERO Y YO"
s.istitle()
```

```
False
```

Operadores de decisión

`if`

Cuando queremos comprobar si se cumple alguna condición, utilizamos el operador de decisión `if`. La sintaxis que debemos seguir es la siguiente:

```
if condicion:
    consecuencia
```

¡Cuidado! La sintaxis de los dos puntos después de la condición y la indentación (equivalente a una tabulación, un total de 4 espacios en blanco) que precede a la consecuencia es muy importante. De hecho, si se omite alguna de las dos cosas o bien nos pasamos de indentación, nos saltará error.

- Si queréis hacer una tabulación, debéis pulsar el tabulador una vez.
- Podéis hacer tabulaciones en bloque seleccionando las líneas de código que queráis indentar y, a continuación, pulsando el tabulador.
- Podéis deshacer tabulaciones pulsando Shift + Tab
- Podéis deshacer tabulaciones en bloque seleccionando las líneas de código que queráis desindentar y, a continuación, pulsando el Shift + Tab.

Siguiendo con nuestro ejemplo, si el usuario tiene más de 16 años, pero menos de 40, entonces puede formar parte de la tripulación de Pyratilla.

```
age = 23
if (age >= 16 and age <= 40):
    print("Puedes formar parte de la tripulación de Pyratilla")
```

```
Puedes formar parte de la tripulación de Pyratilla
```

`else`

Ahora, nos podríamos preguntar qué le podríamos decir al usuario en el caso en que no satisfaga la condición. Ahí es donde entra en juego el operador de decisión `else`. Esta vez, la sintaxis a seguir es la siguiente:

```
if condición:
    consecuencia_si_es_verdad
else:
    consecuencia_si_es_falsa
```

Siguiendo el ejemplo anterior, si el usuario tiene 16 años o más, pero menos de 40, entonces puede formar parte de la tripulación de Pyratilla. Si no, le diremos que no satisface una necesidad básica para ser miembro.

```
age = 13
if (age >= 16 and age <= 40):
    print("Puedes formar parte de la tripulación de Pyratilla")
else:
    print("No satisfaces una necesidad básica para pertenecer a la tripulación del gra
```

```
No satisfaces una necesidad básica para pertenecer a la tripulación del gran Pyrat
```

`elif`

Ahora, en vez de comprobar si se cumple o no una condición, nos podríamos preguntar cómo haríamos para comprobar más de una condición. Podríamos hacerlo a lo bruto anidando operadores `if`, esto es, metiendo un `if` dentro de otro; o bien, podríamos hacerlo

anidando operadores `if`, esto es, metiendo un `if` dentro de otro, o bien, podríamos hacerlo utilizando el operador de decisión `elif`.

El operador `elif` funciona del siguiente modo: se empieza con un operador `if`; si la condición de este no se cumple, pasamos a la siguiente condición posible precedida de un `elif`; si esta tampoco se cumple, pasamos al siguiente `elif`; seguimos así hasta que o bien se satisface alguna condición y realizamos su consecuencia, o hasta llegar al `else`, que implica que no se ha satisfecho ninguna de las condiciones anteriores.

La sintaxis del operador de decisión `elif` es la siguiente:

```
if condición_1:
  consecuencia
elif condición_2:
  consecuencia
elif condición_3:
  consecuencia
.
.
.
else:
  consecuencia
```

Recuperemos el ejemplo de que tenemos que tener 16 años o más, pero menos de 40 para ser miembros de la tripulación del pirata Pyratilla. Vamos a mejorar lo que habíamos conseguido con el `if` y el `else`, añadiendo el operador `elif`.

Además, veremos que nos dice con la edad de 20 años.

```
age = 20

if age > 40:
  print("No puedes pertenecer a la tripulación, te pasas de la edad límite que ha pues
elif age >= 16:
  print("Podrás optar a pertenecer la tripulación. Aún te queda superar la entrevista
else:
  print("Eres muy pequeño todavía para la vida pirata!")
```

```
Podrás optar a pertenecer la tripulación. Aún te queda superar la entrevista con e
```

El funcionamiento del código anterior es el siguiente:

- El `if` comprueba si la edad introducida es mayor a 40.
- El `elif` comprueba si la edad se encuentra en el intervalo [16, 40]
- El `else` implica que la edad introducida es menor a 16

Operador ternario

Si queremos hacer un simple `if` / `else` en una sola línea de código, podemos utilizar el operador ternario, que tiene la siguiente estructura:

```
consecuencia_cierto if condición else consecuencia_falso
```

Por ejemplo, hagamos el caso de mayor de edad. Si la edad es mayor o igual a 18, entonces es mayor de edad en España. Si no, entonces es menor de edad en España.

```
age = 20
texto_mayor = "Eres mayor de edad en España"
texto_menor = "Eres menor de edad en España"

print(texto_mayor) if age >= 18 else print(texto_menor)
```

```
Eres mayor de edad en España
```

Operadores Anidados

```
age = 20
name = "Martin"

if age >= 18:
  if name.startswith("M") or name.startswith("m"):
    print("Eres mayor de edad pues tienes {} años y tu nombre, que es {}, empieza por M"
  else:
    print("Eres mayor de edad pues tienes {} años".format(age))
else:
  print("Eres muy joven")
```

```
Eres muy joven
```