Lección 11: Condicionales

Índice

Anterior

Definición

Por defecto el código Python, un script, se ejecuta de forma ordenada, de principio a fin. Hay dos formas de alterar este comportamiento (obviando funciones, orientación a objetos, etc).

- *Condicionales*:Un determinado bloque de código únicamente se ejecutará cuando una condición sea cierta (*True*).
- *Bucles*: Un determinado bloque de código se ejecutará repetidamente mientras una condición sea cierta (*True*).

Condicional If

- *if* se utiliza para comprobar si una condición es cierta y ejecutar un bloque de código.

```
if condicion:
    do this
    and this
    and also this
```

```
# Condicional if

a = 10
if a < 5:
    print('a es menor que 5')


if a > 5:
    print('a es mayor que 5')
    print('Fin del condicional')
```

Condicional If Else

- Si la condición es cierta, se ejecuta el primer bloque de código (tras el *if*), si no es cierta se ejecuta el segundo bloque de código (tras el *else*)

```
# Condicional If Else
a = 10

if a < 5:
    print('a es menor que 5')
```

```
else:
    print('a es mayor que 5')
```

Condicional If Elif Else

- Se utiliza para comprobar múltiples condiciones.

```
if condicion1:
    haz esto
elif condicion2:
    haz esto otro
elif condicion3:
    haz esto
else:
    haz esto
```

```
# Condicional If Elif Else

print("Resultado del primer condicional:")
a = 100
if a < 0:
    print("Número negativo")
elif a > 0:
    print("Número positivo")
elif a > 5:
    print("Número mayor que 5") # Hay que se cuidadosos con las condiciones porque una vez
else:
    print("El número es 0")

print("Resultado del segundo condicional:")
a = 100
if a < 0:
    print("Número negativo")
elif a > 5:
    print("Número positivo mayor que 5")
elif a > 0:
    print("Número positivo") # Hay que se cuidadosos con las condiciones porque una vez que
else:
    print("El número es 0")
```

Condicionales simplificados en una línea

Hay una forma especial de escribir condicionales en una sola línea cuando se va a ejecutar una única intrucción por condición

```
do this if condicion else do this other thing
```

```
# Condicionales en una línea

a = 10
print('Positivo') if a >= 0 else print('Negativo')
```

Condicionales anidados

Los condicionales se pueden anidar como en cualquier otro lenguaje de programación. Sin embargo lo recomendable es no tener que recurrir a condicionales anidados.

```
if condicion:
    haz esto
    if condicion2:
        haz tambien esto
```

```
# Condicionales anidados

a = 0
if a > 0:
    if a > 5 == 0:
        print('A es positivo y mayor que 5')
    else:
        print('A es positivo')
elif a == 0:
    print('A es cero')
else:
    print('A es negativo')
```

Condicionales con operadores lógicos

- Se pueden utilizar los operadores lógicos *and or not*, junto con los operadores *is* e *in* para elaborar condiciones complejas.

```
## Condiciones con operadores

a=20

if type(a) == int or type(a) == float:
    if a > 0 and a < 5:
        print("Número positivo menor de 5")
    elif a >= 5:
        print("Número positivo mayor o igual a 5")
    elif a == 0:
        print("Número igual a cero")
    else:
        print("Número negativo")
```

```
else:
    print("No es un número")

nombres = ["pedro", "daniel", "josé", "teresa", "arturo"]

if "manuel" not in nombres:
    nombres.append("manuel")

print(nombres)
```

Lectura de datos por teclado con *input*

- *input("mensaje al usuario")*: Sirve para pedir al usuario que introduzca datos por teclado. Siempre devuelve una cadena de texto.

```
# Ejemplo de entrada de datos por teclado

entrada = input("Introduce un número entero: ")

if entrada.isnumeric():
    a = int(entrada)

    if a > 0 and a < 5:
        print("Número positivo menor de 5")
    elif a >= 5:
        print("Número positivo mayor o igual a 5")
    elif a == 0:
        print("Número igual a cero")
    else:
        print("Número negativo")
else:
    print("No es un número entero")
```

Un poco de gestión de excepciones (A LO BESTIA)

- Hay que distinguir *errores de sintaxis* de *excepciones*.
- Es recomendable (casi obligatorio) controlar y gestionar las *excepciones* que se puedan dar en tiempo de ejecución.

La gestión de excepciones en Python se puede controlar con *try-except*.

```
try
    intenta hacer esto
except tipoExcepcion:
    ejecuta esto otro si se produce una excepción de tipoExcepcion
```

```
# Ejemplo de gestión de excepciones
```

```python
entrada = input("Introduce un número: ")

try:
    a = int(entrada) # Esto dará error. Excepción de tipo ValueError si el texto introducid
    if a > 0 and a < 5:
        print("Número positivo menor de 5")
    elif a >= 5:
        print("Número positivo mayor o igual a 5")
    elif a == 0:
        print("Número igual a cero")
    else:
        print("Número negativo")

except:
    print("No es un número. Debes introducir un número.") # Esto se ejecuta si ocurre cualq

# Para controlar mejor, es necesario especificar el tipo de excepción.

entrada = input("Introduce un número: ")

try:
    a = int(entrada) # Esto dará error. Excepción de tipo ValueError si el texto introducid
    if a > 0 and a < 5:
        print("Número positivo menor de 5")
    elif a >= 5:
        print("Número positivo mayor o igual a 5")
    elif a == 0:
        print("Número igual a cero")
    else:
        print("Número negativo")

except ValueError:
    print("No es un número. Debes introducir un número.") # Esto se ejecuta si ocurre una ex
```

Siguiente