

Programación con



Tema 3: Control de flujo

Alma Mallo Casdelo - alma.mallo@udc.es

Operadores lógicas

Símbolo	Operación	Resultado
==	Igual	$A == B \rightarrow$ True si A y B son iguales, False en caso contrario
!=	Distinto	$A != B \rightarrow$ True si A y B son distintos, False en caso contrario
>	Mayor	$A > B \rightarrow$ True si A es mayor que B, False en caso contrario
>=	Mayor o igual	$A >= B \rightarrow$ True si A es mayor o igual que B, False en caso contrario
<	Menor	$A < B \rightarrow$ True si A es menor que B, False en caso contrario
<=	Menor o igual	$A <= B \rightarrow$ True si A es menor o igual que B, False en caso contrario
and	And lógico	$\text{ExpA and ExpB} \rightarrow$ True si ExpA es True y ExpB es True, False en caso contrario
or	Or lógico	$\text{ExpA or ExpB} \rightarrow$ True si ExpA es True o ExpB es True, False si ambas son False
not	Not lógico	$\text{not Exp} \rightarrow$ True si Exp es False, False en caso contrario

Control de flujo: **if**

- Permite ejecutar una sentencia o grupo de sentencias en función de si se cumple o no una condición. Sintaxis:

```
if expresion_logica1:  
    # Sentencias que se ejecutan si expresion_logica1 es verdadera  
elif expresion_logica2:  
    # Sentencias que se ejecutan si expresion_logica1 es falsa y expresion_logica2 es  
verdadera  
else:  
    # Sentencias que se ejecutan si no se cumple ninguna de las condiciones anteriores
```

- **elif** y **else** son opcionales y puede haber varios **elif**.
- Todas las sentencias “dentro” de **if**, **elif** y **else** tienen que estar indentadas hacia la derecha.

Ejemplo

El siguiente programa pide al usuario por teclado su edad y muestra un mensaje indicando si es mayor de edad o no:

```
edad = int(input("¿Qué edad tiene? "))

if edad < 0 or edad > 120:
    print("No es una edad válida")
elif edad < 18:
    print("Es usted menor de edad")
else:
    print("Es usted mayor de edad")

print("¡Fin del programa!")
```

Control de flujo: **match .. case**

- Se introdujo en Python 3.10.
- Es una herramienta de pattern matching, no una sentencia *switch - case* como las existentes en otros lenguajes. Aunque puede utilizarse también de la misma forma:

```
option = int(input("Selecciona una opción [1..3]:"))

match option:
    case 1:
        print('Opción 1 seleccionada')
    case 2:
        print('Opción 2 seleccionada')
    case 3:
        print('Opción 3 seleccionada')
    case other:
        print('Opción no disponible')
```

Control de flujo: **while**

- Repite la ejecución de una sentencia o grupo de sentencias mientras se cumpla una condición:

```
while expresion_logica:  
    # Sentencias que se repiten mientras  
    # expresion_logica sea verdadera
```

- Todas las sentencias “dentro” del **while** tienen que estar indentadas hacia la derecha.

Ejemplo

Imprime la tabla de multiplicar de un número que se pide al usuario.

```
n = int(input("Introduce el número: "))
while n<1 or n>10:
    n = int(input("Error en el rango. Introduce el número: "))

indice=1
while indice<=10:
    mult=indice*n
    print(f"{n}x{indice} = {mult}")
    indice=indice+1
```


Control de flujo: **for**

- Repite la ejecución de una sentencia o grupo de sentencias para cada uno de los elementos proporcionados por un iterador:

```
for elemento in expresion:  
    # Sentencias que se ejecutan para cada elemento
```

- ¿Cómo funciona **for** internamente?:
 1. Antes de entrar en el bucle se evalúa **expresion** y se obtiene como resultado un objeto iterable.
 2. Se llama al método `__iter__()` de dicho objeto, obteniendo así un iterador.
 3. Se llama al método `__next__()` del iterador obteniendo así un elemento.
 4. Se ejecutan las sentencias de dentro del **for**.
 5. Se vuelve a 3. mientras el iterador no lance la excepción `StopIteration`, momento el que acaba el **for**.
- Todas las sentencias “dentro” del **for** tienen que estar indentadas hacia la derecha.

Control de flujo: **for**

- Con el bucle for se puede usar la función **range(start, stop, step)**
- Tiene tres parámetros con valores por defecto, con lo que podemos usarla con uno, dos o tres argumentos. Con tres argumentos:
 - **start** es el primer número entero por el que comienza el rango.
 - **stop** es el número que para el rango. No se llega a ese valor, para en el anterior.
 - **step** es el valor que se va sumando a start para conseguir los números consecutivos. Puede ser positivo o negativo.

Ejemplo

Imprime la tabla de multiplicar de un número que se pide al usuario.

```
n = int(input("Introduce el número: "))
while n<0 or n>10:
    n = int(input("Error en el rango. Introduce el número: "))

for indice in range(1,11):
    mult=indice*n
    print(f"{n}x{indice} = {mult}")
```

Control de flujo: **break**

- Se puede usar **break** para salir de un **while** o un **for** en cualquier momento.
- Ejemplo: buscar el primer múltiplo de 5 y 7 entre 1 y 50.

```
for num in range(1, 50+1):  
    if num % 5 == 0 and num % 7 == 0:  
        print(f"{num} es el primer múltiplo de 5 y 7")  
        break
```

Control de flujo: **continue**

- Se puede usar **continue** para abortar la ejecución de las sentencias de una iteración, continuando en el siguiente elemento proporcionado por el iterador.
- Ejemplo: programa que imprime todos los números hasta max que no son múltiplos de 5.

```
max = 20
for num in range(1, max+1):
    if num % 5 == 0:
        continue
    print(num, end = " ")
```

Control de flujo: **else** en bucles


- En Python los bucles while y for permiten un bloque **else**.
- Es una característica que se usa poco.
- De forma general, se ejecuta cuando el bucle termina sus iteraciones de forma *normal* (si el bucle termina con break, no se ejecuta).

Ejemplo

- El siguiente programa busca el primer múltiplo de 5 y 7 entre dos números:

```
min = 1
max = 50
for num in range(min, max+1):
    if num % 5 == 0 and num % 7 == 0:
        print(f"{num} es el primer múltiplo de 5 y 7")
        break
else:
    print(f"No hay múltiplos de 5 y 7")
```

Si lo ejecutamos con
min=40 y max=50,
el else se ejecuta



Referencias

- <https://docs.python.org/3/tutorial/controlflow.html>
- https://docs.python.org/3/reference/compound_stmts.html