



Python 3

4. Control de flujo

Carolina Mañoso, Ángel P. de Madrid y Miguel Romero

◆ Estructuras de control de flujo:

- Sentencias condicionales o de selección:
 - ◆ if
 - ◆ if...else
 - ◆ if...elif...elif...else
- Bucles o sentencias iterativas:
 - ◆ while
 - ◆ for...in

Sentencias condicionales (1/3)

- ◆ Permiten ejecutar un fragmento de código u otro dependiendo de que se cumpla una condición.
- ◆ La forma más simple se construye mediante `if` seguido de la condición a evaluar (tipo de dato `bool`), dos puntos (`:`), y en la siguiente línea y *sangrado*, el código a ejecutar en el caso de que se cumpla la condición.

```
tiempo = input("soleado o nuboso: ")
if tiempo == "soleado":
    print("Iremos a la playa")
print("Qué bien!!")
```

- ◆ El sangrado sustituye a las llaves `{ }`, de otros lenguajes.

```
tiempo = input("soleado o nuboso: ")
if tiempo == "soleado":
    print("Iremos a la playa")
    print("Qué bien!!")
```

Sentencias condicionales (2/3)

- ◆ Si queremos que en caso de que no se cumpla la condición se ejecute otro bloque de código usaremos la construcción : `if...else`.

```
if tiempo == "soleado":  
    print("Iremos a la playa")  
else:  
    print("Nos quedaremos en casa")
```

- ◆ Si queremos anidar condiciones usaremos la construcción: `if...elif...elif...else`.

```
if tiempo == "soleado":  
    print("Iremos a la playa")  
elif tiempo == "nublado":  
    print("Daremos un paseo")  
else:  
    print("Nos quedaremos en casa")
```

Sentencias condicionales (2/3)

- ◆ Observe que si se cumple una condición, se ejecuta ese bloque y se sale de la estructura. Si ninguna es cierta se ejecuta `else`.

```
num = int(input("Introduce un numero: "))
''' input devuelve una cadena de caracteres,
    de ahí la conversión de tipo
'''
if num % 2 == 0:
    print("Es divisible por 2")
elif num % 3 == 0:
    print("Es divisible por 3")
elif num % 5 == 0:
    print("Es divisible por 5")
else:
    print("No es divisible por 2, 3 ó 5")
```

Nota: Ejecutarlo con `num = 10`.

Bucles (1/8)

- ◆ Los bucles o sentencias iterativas nos permiten ejecutar un fragmento de código un cierto número de veces, mientras se cumpla una determinada condición.
- ◆ Un tipo de estructura de bucle es: `while` seguido de la condición a evaluar, dos puntos (:), y en la siguiente línea y *sangrado*, el código a ejecutar en el caso de que se cumpla la condición.

```
# Serie de Fibonacci
a, b = 0, 1      # Asignación múltiple
while a < 1000:
    print(a)
    a, b = b, a+b
```



Bucles (2/8)

- ◆ Nos debemos asegurar la salida del bucle (que la condición de convierta en `False`).

```
tiempo = "soleado"
while tiempo == "soleado":
    print("Iremos a la playa")
    tiempo = input("¿Qué tiempo hace?") ➡
```

- ◆ Si debemos salir de un bucle infinito: `Ctrl + C`.

- ◆ Podemos escribir bucles infinitos con `True` y salir con `break`.

```
tiempo = "soleado"
while True:
    print("Iremos a la playa")
    tiempo = input("¿Qué tiempo hace?")
    if tiempo != "soleado":
        break
```

Bucles (3/8)

- ◆ Otra forma de construir un bucle es con la estructura `for...in`. Se utiliza para recorrer los elementos de un lista.
 - La construcción utiliza una variable que en cada ciclo toma el valor de un nuevo elemento de la lista de la cláusula `in`.
 - Esta variable que usamos en la construcción se declara en *tiempo de ejecución* (dinámicamente).
 - El bucle terminará cuando se alcance el último elemento de la lista o cuando se fuerce su fin con `break`.

```
# Escribe los elementos de la lista
listaColores = ['rojo', 'verde', 'azul']
for color in listaColores:
    print(color)
```


Bucles (4/8)

◆ Si se necesita iterar sobre una secuencia de números, es apropiado utilizar la función `range()` :

- `range(x)` : recorre el valor de 0 a $x-1$.
- `range(x, y)` : recorre desde x hasta $y-1$.
- `range(x, y, s)` : recorre desde x hasta $y-1$, de s en s .

```
for i in range(5):  
    print(i)
```

```
for i in range(0,5):  
    print(i)
```

```
for i in range(0,5,1):  
    print(i)
```

Bucles (5/8)

- ◆ La construcción `for`, además de sobre una lista, también permite *iterar* sobre tupla, conjunto, cadena y diccionario.

```
for i in [0,1,2,3,4]:
```

```
    print(i)
```

```
for i in (0,1,2,3,4):
```

```
    print(i)
```

```
for i in {0,1,2,3,4}:
```

```
    print(i)
```

```
# El conjunto no puede especificar el orden
```

```
for i in {4,3,2,1,0}:
```

```
    print(i)
```

```
# Mal uso del tipo de dato
```

```
for i in "01234":
```

```
    print(i)
```

Bucles (6/8)

◆ Existen *métodos* que nos ayudan en las iteraciones:

- Cuando iteramos sobre diccionarios se puede obtener la clave y el valor con `items()`.

```
dic = {'piso1' : 'Juan', 'piso2' : 'Pepe'}  
for k,v in dic.items():  
    print(k,v)
```

- Cuando iteramos sobre secuencias se puede obtener el índice de posición junto con el valor con `enumerate()`.

```
for i,v in enumerate(['a', 'b', 'c']):  
    print(i,v)
```

Bucles (7/8)

- Cuando iteramos sobre dos valores pueden emparejarse con `zip()`.

```
preguntas = ['nombre', 'apellido', 'dirección']
datos = ['Pepe', 'López', 'Madrid']
for i,v in zip(preguntas, datos):
    print(i,v)
```

- Para iterar sobre una secuencia en orden inverso: `reversed()`.

```
for i in reversed([0,1,2,3,4]):
    print(i)
```

- Para iterar sobre una secuencia ordenada `sorted()`.

```
for i in sorted([0,3,2,4,1]):
    print(i)
```

Bucles (8/8)

◆ La construcción `for` también permite anidamiento.

```
# Calcula los números primos entre 1:50 y los divisores
for i in range(1,50):
    primo = True
    for k in range(2,i):
        if (i%k) == 0:
            print(i, "es divisible por", k)
            primo = False
    if primo:
        print(i, "es primo")
```

Aviso



Python 3 by C. Mañoso, A. P. de Madrid, M. Romero is licensed under a [Creative Commons Reconocimiento-NoComercial-CompartirIgual 4.0 Internacional License](https://creativecommons.org/licenses/by-nc-sa/4.0/).

Esta colección de transparencias se distribuye con fines meramente docentes.

Todas las marcas comerciales y nombres propios de sistemas operativos, programas, hardware, etc. que aparecen en el texto son marcas registradas propiedad de sus respectivas compañías u organizaciones.