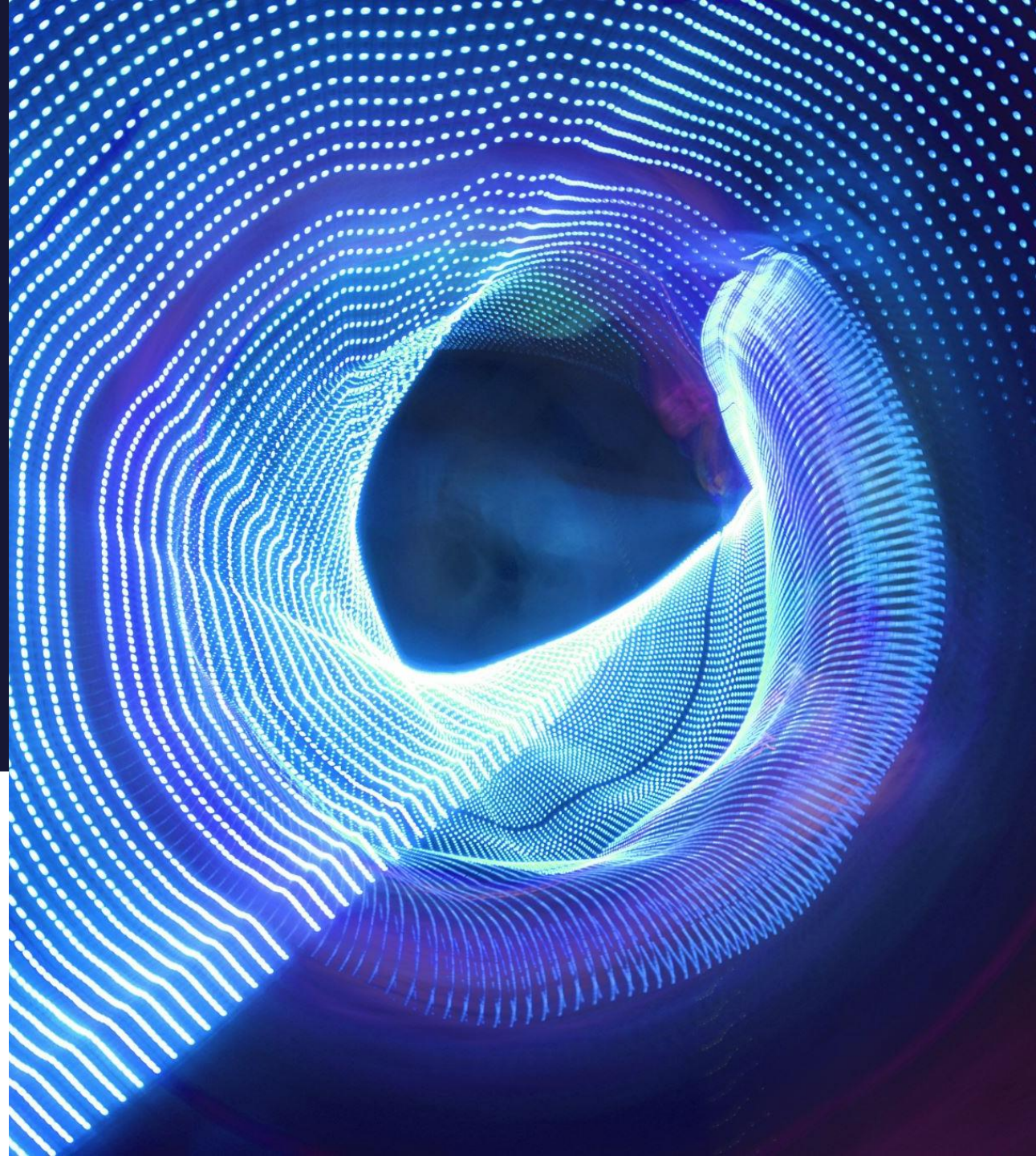
Bucles en tu código con `for`



Otro tipo de bucle disponible en Python proviene de la observación de que a veces es más importante contar los "giros o vueltas" del bucle que verificar las condiciones.

Imagina que el cuerpo de un bucle debe ejecutarse exactamente cien veces. Si deseas utilizar el bucle while para hacer

```
i = 0
while i < 100:
    # do_something()
    i += 1
```

Sería bueno si alguien pudiera hacer esta cuenta aburrida por ti. ¿Es eso posible?

for

Por supuesto que lo es, hay un bucle especial para este tipo de tareas, y se llama for.

En realidad, el bucle for está diseñado para realizar tareas más complicadas, puede "explorar" grandes colecciones de datos elemento por elemento. Te mostraremos como hacerlo pronto, pero ahora presentaremos una variante más sencilla de su aplicación.

Echa un vistazo al fragmento:

```
for i in range(100):
    # do_something()
    pass
```

Existen algunos elementos nuevos. Déjanos contarte sobre ellos:

La palabra reservada for abre el bucle for; nota - No hay condición después de eso; no tienes que pensar en las condiciones, ya que se verifican internamente, sin ninguna intervención.

Cualquier variable después de la palabra reservada for es la variable de control del bucle; cuenta los giros del bucle y lo hace automáticamente.

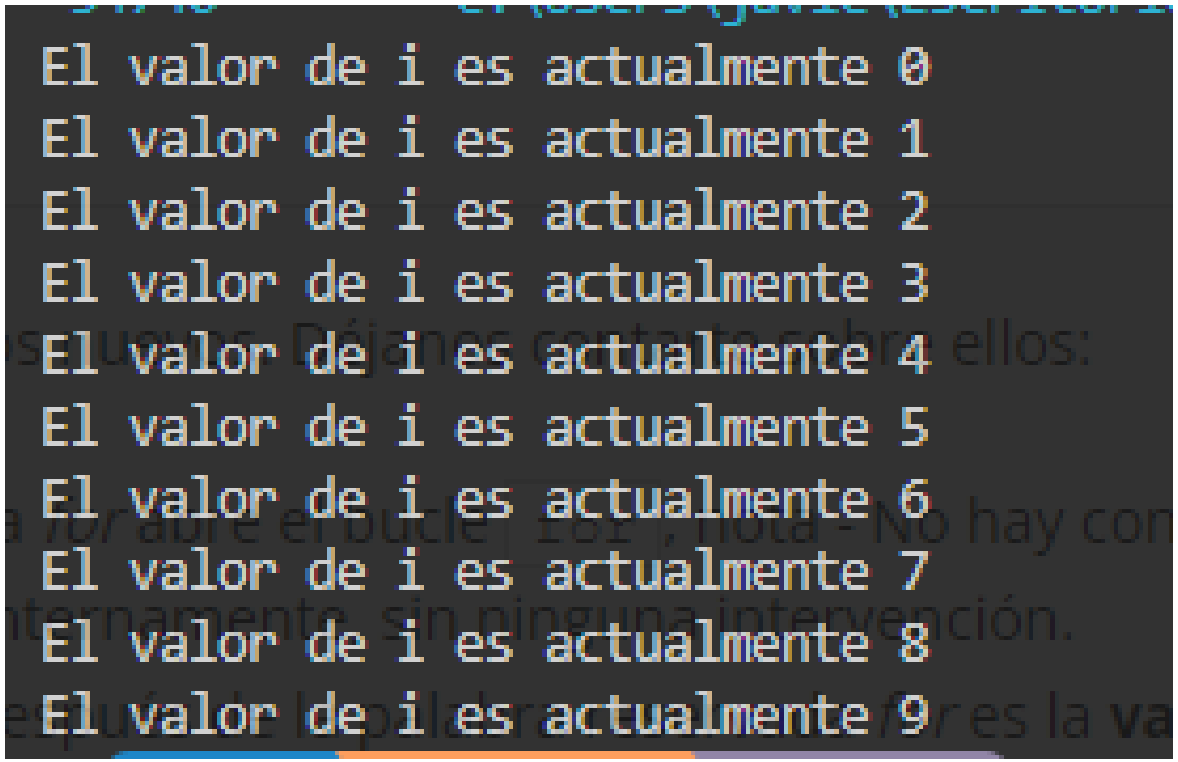
La palabra reservada in introduce un elemento de sintaxis que describe el rango de valores posibles que se asignan a la variable de control.

- La función range() (esta es una función muy especial) es responsable de generar todos los valores deseados de la variable de control; en nuestro ejemplo, la función creará (incluso podemos decir que alimentará el bucle con) valores subsiguientes del siguiente conjunto: 0, 1, 2 .. 97, 98, 99; nota: en este caso, la función range() comienza su trabajo desde 0 y lo finaliza un paso (un número entero) antes del valor de su argumento.
- Nota la palabra clave pass dentro del cuerpo del bucle - no hace nada en absoluto; es una instrucción vacía : la colocamos aquí porque la sintaxis del bucle for exige al menos una instrucción dentro del cuerpo (por cierto, if, elif, else y while expresan lo mismo).

Echa un vistazo al fragmento de abajo. ¿Puedes predecir su salida?

```
for i in range(10):
    print("El valor de i es actualmente", i)
```

¿Es tu salida eperada?
Salida:



El bucle se ha ejecutado diez veces (es el argumento de la función range()).

El valor de la última variable de control es 9 no 10, ya que comienza desde 0 , no desde).

La invocación de la función range() puede estar equipada con dos argumentos, no solo uno:

```
for i in range(2, 8):
    print("El valor de i es actualmente", i)
```

En este caso, el primer argumento determina el valor inicial (primero) de la variable de control.

El último argumento muestra el primer valor que no se asignará a la variable de control.

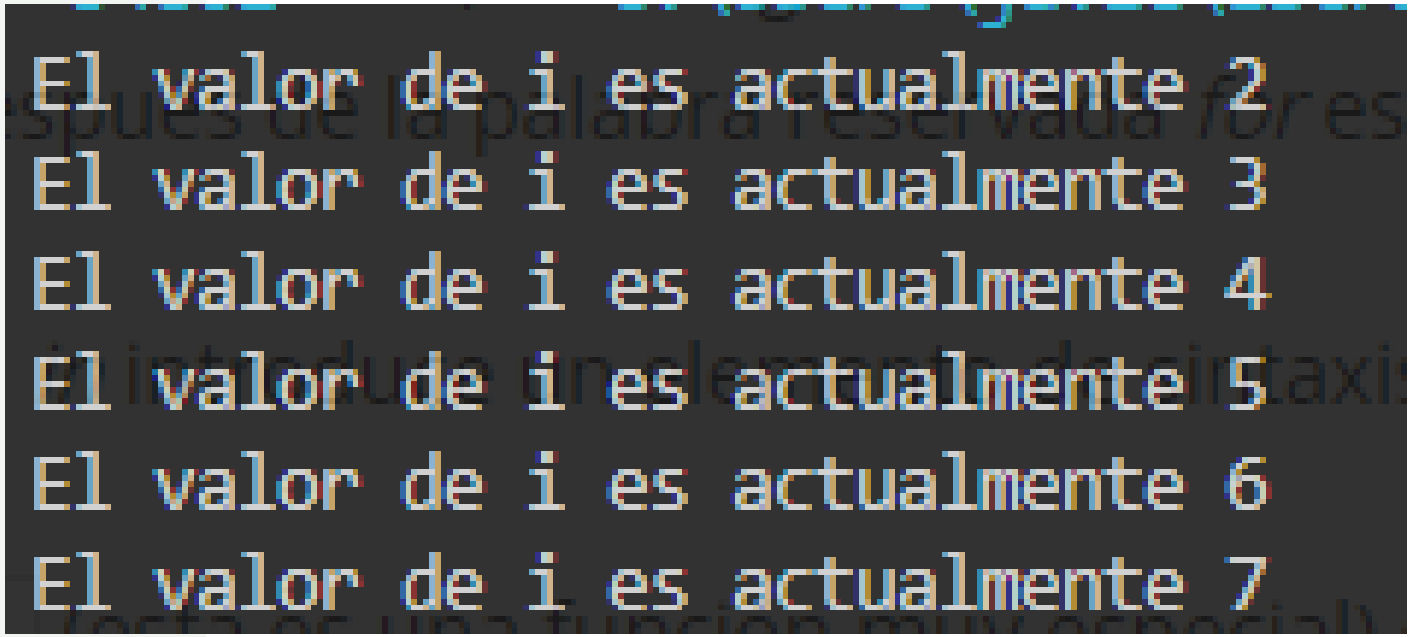
Nota: la función range() solo acepta enteros como argumentos y genera secuencias de enteros.

¿Puedes adivinar la salida del programa? Ejecútalo para comprobar si ahora también estabas en lo cierto.

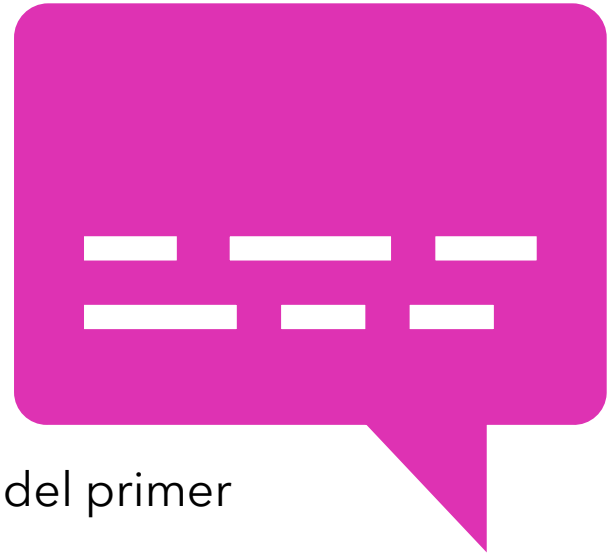
Probemos el código

```
for i in range(2, 8):
    print("El valor de i es actualmente", i)
```

Salida:


El valor de i es actualmente 2
El valor de i es actualmente 3
El valor de i es actualmente 4
El valor de i es actualmente 5
El valor de i es actualmente 6
El valor de i es actualmente 7

Es tu respuesta correcta?



El primer valor mostrado es 2 (tomado del primer argumento de range()).
El último es 7 (aunque el segundo argumento de range() es 8).