Bucles. for (I)

El buclefor

En general, un bucle es una estructura de control que **repite** un bloque de instrucciones. Un bucle for es un bucle que repite el bloque de instrucciones un **número prederminado** de veces.

El bloque de instrucciones que se repite se suele llamar **cuerpo del bucle** y cada repetición se suele llamar **iteración**.

La sintaxis de un buclefores la siguiente:

```
for variable in elemento iterable (lista, cadena, range, etc.):
    cuerpo del bucle
```

No es necesario definir la variable de control antes del bucle, aunque se puede utilizar como variable de control una variable ya definida en el programa.

El cuerpo del bucle se ejecuta tantas veces como elementos tenga el elemento recorrible (elementos de una lista o de un range(), caracteres de una cadena, etc.). Por ejemplo:

Ejemplo de bucle

```
print("Comienzo")
for i in [0, 1, 2]:
    print("Hola ,end="")
print()
print("Final")
```

Resultado:

```
Comienzo
Hola Hola Hola
Final
```

Si ejecutamos el programa paso a paso:

Paso 1
Se ejecuta la primera instrucción del programa.En este caso, imprime el párrafo de comienzo.

```
Comienzo
```

Paso 2
A continuación se ejecuta el bucle. La variable de control toma el primer valor de la lista.En este caso, la **variable de control** es `i` y toma el valor `0`.

Paso 3
A continuación se ejecutan las **instrucciones del bloque**.En este caso el bloque consta de una sola instrucción que imprime el texto "Hola".

```
Comienzo
Hola
```

Paso 4

A continuación la variable de control toma el segundo valor de la lista, `i` pasa a valer `1`

Paso 5

A continuación se ejecutan por segunda vez las instrucciones del bloque. Como vimos, el bloque consta de una sola instrucción que imprime el texto "Hola".

```
Comienzo
Hola Hola
```

Paso 6
La variable de control toma el tercer y último valor de la lista.En este caso, la variable de control es `i` y toma el valor `2`.

Paso 7

Se ejecutan por última vez la instrucción del bloque:

```
Comienzo
Hola Hola Hola
```

Paso 8

Una vez terminado el bucle, se ejecuta la instrucción que sigue al bucle.En este caso, ejecuta `print()` que imprime un salto de línea

```
Comienzo
Hola Hola Hola
_
```

Paso 9

Se ejecuta la última instrucción del programa imprime el párrafo final.

```
Comienzo
Hola Hola Hola

Final
```

En el ejemplo anterior, los valores que toma la variable no son importantes, lo que importa es que la lista tiene **tres elementos** y por tanto el bucle se ejecuta **tres veces**.

El siguiente programa produciría el mismo resultado que el anterior:

Ejemplo de bucle 2

```
print("Comienzo")
for i in [1, 1, 1]:
    print("Hola ", end="")
print()
print("Final")
```

Si la lista está vacía, el bucle no se ejecuta ninguna vez. Por ejemplo:

Ejemplo de bucle 3

```
print("Comienzo")
for i in []:
    print("Hola ", end="")
print()
print("Final")
```

El resultado sería

```
Comienzo

Final
```

Al estar la lista vacia, la variable de contral no toma ningún valor y el bucle no se ejecuta.

En los ejemplos anteriores, la variable de control `i` no se utilizaba en el bloque de instrucciones, pero en muchos casos sí que se utiliza. Cuando se utiliza, hay que tener en cuenta que la variable de control va tomando los valores del elemento recorrible. Por ejemplo:

Ejemplo de bucle 4

```
print("Comienzo")
for i in [3, 4, 5]:
    print("Hola. Ahora i vale", i, "y su cuadrado", i ** 2)
print("Final")
```

Este programa al ejecutarse el resultado sería:

```
Comienzo
Hola. Ahora i vale 3 y su cuadrado 9
Hola. Ahora i vale 4 y su cuadrado 16
Hola. Ahora i vale 5 y su cuadrado 25
Final
```

La lista puede contener cualquier tipo de elementos, no sólo números. El bucle se repetirá siempre tantas veces como elementos tenga la lista y la variable irá tomando los valores de uno en uno. Por ejemplo

Ejemplo de bucle 5

```
print("Comienzo")
for i in ["Alba", "Benito", 27]:
    print("Hola. Ahora i vale", i)
print("Final")
```

El resultado será:

```
Comienzo
Hola. Ahora i vale Alba
Hola. Ahora i vale Benito
Hola. Ahora i vale 27
Final
```

La costumbre más extendida es utilizar la letra `i` como nombre de la variable de control, pero se puede utilizar cualquier otro nombre válido. Por ejemplo:

Ejemplo de bucle 6

```
print("Comienzo")
for numero in  [0, 1, 2, 3]:
    print(numero, "*", numero,"=", numero ** 2)
print("Final")
```

La salida de este programa sería:

```
Comienzo
0 * 0 = 0
1 * 1 = 1
2 * 2 = 4
3 * 3 = 9
Final
```

La variable de control puede ser una variable empleada antes del bucle. El valor que tuviera la variable no afecta a la ejecución del bucle, pero cuando termina el bucle, la variable de control conserva el último valor asignado

Ejemplo de bucle 7:

```
i = 10
print("El bucle no ha comenzado. Ahora i vale", i)
for i in [0, 1, 2, 3, 4]:
    print(i, "*", i,"=", i ** 2)
print("El bucle ha terminado. Ahora i vale", i)
```

La salida sería:

```
El bucle no ha comenzado.
Ahora i vale 10
0 * 0 = 0
1 * 1 = 1
2 * 2 = 4
3 * 3 = 9
4 * 4 = 16
El bucle ha terminado. Ahora i vale 4
```

En vez de una lista se puede escribir una cadena, en cuyo caso la variable de control va tomando como valor cada uno de los caracteres:

Ejemplo de bucle 8

```
for i in "AMIGO":
    print("Dame una ", i)
print("¡AMIGO!")
```

El resultado sería:

```
Dame una A
Dame una M
Dame una I
Dame una G
Dame una O
¡AMIGO!
```

El tipo range()

En los ejemplos anteriores se ha utilizado una lista para facilitar la comprensión del funcionamientode los bucles pero, si es posible hacerlo, se recomienda utilizar el **tipo** inmutable de datos `range()`, entre otros motivos porque durante la ejecución del programa ocupan menos memoria.

Para crear un tipo `range`, se pueden hacer de dos formas :

- `range(fin)`: Crea una secuencia numérica que va desde `0` hasta `fin - 1`.
- `range(inicio, fin, [paso])`: Crea una secuencia numérica que va desde `inicio` hasta `fin - 1`. Si además se indica el parámetro `paso`, la secuencia genera los números de `paso` en `paso`.

Veámos algunos ejemplos:

```
>>> list(range(10))

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

>>> list(range(5, 10))

[5, 6, 7, 8, 9]

>>> list(range(0, 10, 3))

[0, 3, 6, 9]

>>> list(range(0, -10, -1))

[0, -1, -2, -3, -4, -5, -6, -7, -8, -9]

>>> list(range(5, -5, -2))

[5, 3, 1, -1, -3]
```

El tipo `range` genera los valores en el momento de su ejecución. La función de conversión de tipos `list` convierte a lista los valores generados por `range`

Cómo se indicó, la principal ventaja de usar `range()` frente a variables de tipo `list` o `tupl` es que es un iterable que genera los elementos solo en el momento en que realmente los necesita. Esto implica que usa una cantidad de memoria mínima, por muy grande que sea el rango de números que represente.

Veamos una comparación de una lista que almacena los números del 0 al 100.000 y un rango del 0 al 100.000:

```
>>> import sys

>>> lista = list(range(0, 100000))

>>> rango = range(0, 100000)

>>> sys.getsizeof(lista)

900120

>>> sys.getsizeof(rango)

48
```

Como se puede ver, la lista ocupa casi *1 MB* en memoria frente a los *48 bytes* que ocupa el rango.

Continuando con el programa inicial, el siguiente programa es equivalente al del primer ejemplo:

```
print("Comienzo")
for i in range(3):
    print("Hola ", end="")
print()
print("Final")
```

Otra de las ventajas de utilizar tipos `range()` es que el argumento del tipo `range()` lo podemos usar para controlar el número de veces que se ejecuta el bucle.

En el ejemplo anterior basta cambiar el argumento para que el programa salude muchas más veces.

```
print("Comienzo")
for i in range(10):
    print("Hola ", end="")
print()
print("Final")
```

Esto permite que el número de iteraciones dependa del desarrollo del programa. En el ejemplo siguiente es el usuario quien decide cuántas veces se ejecuta el bucle:

```
veces = int(input("¿Cuántas veces quiere que le salude? ")
for i in range(veces):
    print("Hola ", end="")
print()
print("Adios")
```

Contadores y acumuladores

En muchos programas se necesitan variables que lleven la cuenta del número de veces que ha ocurrido algo (contadores) o que acumulen valores (acumuladores). Las situaciones pueden ser muy diversas. Veremos algunos ejemplos para entender la idea.

Contador

Se entiende por **contador** una variable que lleva la cuenta del número de veces que se ha cumplido una condición. El ejemplo siguiente es un ejemplo de programa con contador:

```
print("Comienzo")
n_pares = 0                          # Se inicializa el contador
for i in range(1, 6):
    if i % 2 == 0:
        n_pares = n_pares + 1        # Se incrementa cada vez que se cumple la
condición
print(f"Desde 1 hasta 5 hay {n_pares} números pares")
```

Después de ejecutar el programa obtenemos:

```
Comienzo
Desde 1 hasta 5 hay 2 números pares
```

Detalles importantes:

- En cada iteración, el programa comprueba si i es múltiplo de 2.
- El contador se modifica sólo si la variable de control i es múltiplo de 2.
- El contador va aumentando de uno en uno.
- Antes del bucle se debe dar un valor inicial a la variable **contador** (en este caso, 0)

Acumulador

Se entiende por **acumulador** una variable que acumula el resultado de una operación. En el ejemplo siguiente es un ejemplo de programa con acumulador (en este caso, la variable que hace de acumulador es la variable suma):

```
print("Comienzo")
suma = 0                      # Se inicializa el acumulador
for i in [1, 2, 3, 4]:
    suma = suma + i           # Se acumula valor
print(f"La suma de los números de 1 a 4 es {suma}")
```

Al ejecutar obtenemos:

```
Comienzo
La suma de los números de 1 a 4 es 10
```

Detalles importantes:

- El acumulador se modifica en cada iteración del bucle (en este caso, el valor de `i` se añade al acumulador `suma`).
- Antes del bucle se debe dar un **valor inicial** al acumulador (en este caso, 0)

Referencias

- J2LOGO - artículo sobre el tipo range()
- Apuntes generados a partir del curso Introducción a la programación con Python que se distribuye bajo una Licencia Creative Commons Reconocimiento-CompartirIgual 4.0 Internacional (CC BY-SA 4.0).

tags: `pro` `ut1` `python` `bucles` `for`