Bucle for (1)

En esta lección se tratan los bucles for:

El bucle for

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 bucle for es 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 1

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

```
Comienzo
Hola Hola Hola
Final
```

Puede ver la ejecución paso a paso de este programa utilizando los iconos de avance y retroceso situados abajo a la derecha.

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")
```

```
Comienzo
Hola Hola Hola
Final
```

Puede ver la ejecución paso a paso de este programa utilizando los iconos de avance y retroceso situados abajo a la derecha.

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")
```

```
Comienzo

Final
```

Puede ver la ejecución paso a paso de este programa utilizando los iconos de avance y retroceso situados abajo a la derecha.

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(f"Hola. Ahora i vale {i} y su
cuadrado {i ** 2}")
print("Final")
```

```
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
```

Puede ver la ejecución paso a paso de este programa utilizando los iconos de avance y retroceso situados abajo a la derecha.

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]:
```

```
Comienzo
Hola. Ahora i vale Alba
```

```
    print(f"Hola. Ahora i vale {i}")
print("Final")
```

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

Puede ver la ejecución paso a paso de este programa utilizando los iconos de avance y retroceso situados abajo a la derecha.

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(f"{numero} * {numero} =
{numero ** 2}")
print("Final")
```

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

Puede ver la ejecución paso a paso de este programa utilizando los iconos de avance y retroceso situados abajo a la derecha.

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:

```
i = 10
print(f"El bucle no ha comenzado. Ahora
i vale {i}")

for i in [0, 1, 2, 3, 4]:
    print(f"{i} * {i} = {i ** 2}")

print(f"El bucle ha terminado. Ahora i
vale {i}")
```

```
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
```

Cuando se escriben dos o más bucles seguidos, la costumbre es utilizar el mismo nombre de variable puesto que cada bucle establece los valores de la variable sin importar los valores anteriores:

```
for i in [0, 1, 2]:
    print(f"{i} * {i} = {i ** 2}")

print()
```

```
0 * 0 = 0
1 * 1 = 1
2 * 2 = 4
```

```
for i in [0, 1, 2, 3]:
    print(f"{i} * {i} * {i} = {i ** 3}")
```

```
0 * 0 * 0 = 0
1 * 1 * 1 = 1
2 * 2 * 2 = 8
3 * 3 * 3 = 27
```

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:

```
for i in "AMIGO":
    print(f"Dame una {i}")
print("¡AMIGO!")
```

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

En los ejemplos anteriores se ha utilizado una lista para facilitar la comprensión del funcionamiento de los bucles pero, si es posible hacerlo, se recomienda utilizar tipos range(), entre otros motivos porque durante la ejecución del programa ocupan menos memoria en el ordenador.

El siguiente programa es equivalente al programa del ejemplo anterior:

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

```
Comienzo
Hola Hola Hola
Final
```

Otra de las ventajas de utilizar tipos range() es que el argumento del tipo range() controla 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")
```

```
Comienzo
Hola Hola Hola Hola Hola Hola Hola Hola
Hola Hola
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")
```

```
¿Cuántas veces quiere que le salude? 6
Hola Hola Hola Hola Hola Hola
Adios
```

Contadores y acumuladores

En muchos programas se necesitan variables que cuenten cuántas veces ha ocurrido algo (contadores) o que acumulen valores (acumuladores). Las situaciones pueden ser muy diversas, por lo que simplemente hay aquí un par de ejemplos para mostrar 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 (en este caso, la variable que hace de contador es la variable *cuenta*):

Ejemplo de contador

```
print("Comienzo")
cuenta = 0
for i in range(1, 6):
    if i % 2 == 0:
        cuenta = cuenta + 1
print(f"Desde 1 hasta 5 hay {cuenta}
múltiplos de 2")
```

```
Comienzo
Desde 1 hasta 5 hay 2 múltiplos de 2
```

Puede ver la ejecución paso a paso de este programa utilizando los iconos de avance y retroceso situados abajo a la derecha.

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 al contador (en este caso, 0)

Acumulador

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

Ejemplo de acumulador

```
print("Comienzo")
suma = 0
for i in [1, 2, 3, 4]:
    suma = suma + i
```

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

```
print(f"La suma de los números de 1 a 4
es {suma}")
```

Puede ver la ejecución paso a paso de este programa utilizando los iconos de avance y retroceso situados abajo a la derecha.

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)

Última modificación de esta página: 4 de abril de 2017