for y while

Para iterar contamos con dos sentencias que nos ayudarán a crear bucles, nos referimos a *for* y a *while*. La primera de ellas aplica una serie de sentencias sobre cada uno de los elementos que contiene el objeto sobre el que aplicamos la sentencia *for*. Python incorpora una función llamada *range()* que podemos utilizar para iterar sobre una serie de valores. Por ejemplo, echemos un vistazo al siguiente ejemplo:

```
>>> for x in range(1, 3):
...         print(x)
...
1
2
3
```

Asimismo, tal y como hemos visto en el capítulo anterior, es muy común iterar a través de *for* sobre los elementos de una tupla o de una lista:

```
>>> lista = ["uno", "dos", "tres"]
>>> cad = ""
>>> for ele in lista:
...         cad += ele
...
>>> cad
"unodostres"
```

Opcionalmente, *for* admite la sentencia *else*. Si esta aparece, todas las sentencias posteriores serán ejecutadas si no se encuentra otra sentencia que provoque la salida del bucle. Por ejemplo, en la ejecución de un bucle *for* que no contiene ningún *break*, siempre serán ejecutadas las sentencias que pertenecen al *else* al finalizar el bucle. A continuación, veamos un ejemplo para ilustrar este caso:

```
>>> for item in (1, 2, 3):
...         print(item)
...     else:
...         print("fin")
...
1
2
3
fin
```

Otra sentencia utilizada para iterar es *while*, la cual ejecuta una serie de sentencias siempre y cuando se cumpla una determinada condición o condiciones.

Para salir del bucle podemos utilizar diferentes técnicas. La más sencilla es cambiar la condición o condiciones iniciales para así dejar que se cumplan y detener la iteración. Otra técnica es llamar directamente a *break* que provocará la salida inmediata del bucle. Esta última sentencia también funciona con *for*. A continuación, veamos un ejemplo de cómo utilizar *while*:

```
>>> x = 0
>>> y = 3
>>> while x < y:
...         print(x)
...         x += 1
0
1
2
```

Al igual que *for*, *while* también admite opcionalmente *else*. Observemos el siguiente código y el resultado de su ejecución:

```
>>> x = 0
>>> y = 3
>>> while x < y:
...         print(x)
...         x+=1
...         if x == 2:
...             break
...         else:
...             print("x es igual a 2")
0
1
```

Si en el ejemplo anterior eliminamos la sentencia *break,* comprobaremos cómo la última sentencia *print* es ejecutada.

Además de *break*, otra sentencia asociada a *for* y *while* es *continue*, la cual se emplea para provocar un salto inmediato a la siguiente iteración del bucle. Esto puede ser útil, por ejemplo, cuando no deseamos ejecutar una determinada sentencia para una iteración concreta. Supongamos que estamos iterando sobre una secuencia y solo queremos imprimir los números pares:

```
>>> for i in range(1, 10):
...         if i % 2 != 0:
...             continue
...         print(i)
2
4
```