**Más información sobre los bucles en Python**

Anteriormente, exploraste las sentencias iterativas. Una **sentencia iterativa** es un código que ejecuta repetidamente una serie de instrucciones. Dependiendo de los criterios, estas se ejecutan cero o más veces. Hemos iterado a través de código utilizando bucles **for** y bucles **while**. En esta lectura, repasarás la sintaxis de los bucles. Luego, aprenderás a utilizar las palabras clave **break** y **continue** para controlar la ejecución de los bucles.

**Bucles for**

Si necesitas iterar a través de una secuencia especificada, debes usar un bucle **for**.

El siguiente bucle **for** itera a través de una secuencia de nombres de usuario. Puedes ejecutarlo para observar la salida:

1

2

for i in ["elarson", "bmoreno", "tshah", "sgilmore"]:

    print(i)

EjecutarRestablecer

La primera línea de este código es el encabezado del bucle. Allí, la palabra clave **for** indica el inicio de un bucle **for**. Justo después de **for**, viene la variablede bucle. La **variable de bucle** se utiliza para controlar las iteraciones de un bucle. En los bucles **for**, la variable de bucle forma parte del encabezado. En este ejemplo, la variable de bucle es **i**.

El resto del encabezado del bucle indica la secuencia a iterar. El operador **in** aparece antes de la secuencia para indicar a Python que ejecute el bucle para cada elemento de la secuencia. En este ejemplo, la secuencia es la lista de nombres de usuario. El encabezado del bucle debe terminar con dos puntos (**:**).

La segunda línea de este ejemplo de bucle **for** es el cuerpo del bucle. El cuerpo del bucle **for** puede tener varias líneas de código. En el cuerpo, se indica lo que el bucle debe hacer en cada iteración. En este caso, es **print(i)**, o en otras palabras, mostrar el valor actual de la variable de bucle durante esa iteración del bucle. Para que Python ejecute el código correctamente, el cuerpo del bucle debe tener más sangría que el encabezado del bucle.

**Nota:** Cuando se usa en un bucle **for**, el operador **in** precede a la secuencia que iterará el bucle **for**. Cuando se utiliza en una sentencia condicional, el operador *in* se usa para evaluar si un objeto forma parte de una secuencia. El ejemplo **if "elarson" in ["tshah", "bmoreno", "elarson"]** se evalúa como **True** porque **"elarson"** forma parte de la secuencia que sigue a **in**.

**Recorrer en bucle una lista**

El uso de bucles **for** en Python permite iterar fácilmente a través de listas, como una lista de activos informáticos. En el siguiente bucle **for**, **asset** es la variable del bucle y otra variable, **computer\_assets**, es la secuencia. La variable **computer\_assets** almacena una lista. Esto significa que en la primera iteración el valor de **asset** será el primer elemento de esa lista, y en la segunda iteración, el valor de **asset** será el segundo elemento de esa lista. Puedes ejecutar el código para observar qué pasa:

1

2

3

computer\_assets = ["laptop1", "desktop20", "smartphone03"]

for asset in computer\_assets:

    print(asset)

EjecutarRestablecer

**Nota:** También es posible hacer un bucle a través de una cadena. Esto devolverá cada carácter uno por uno. Puedes observarlo ejecutando el siguiente bloque de código que itera a través de la cadena **"security"**:

1

2

3

string = "security"

for character in string:

    print(character)

EjecutarRestablecer

**Uso de range()**

Otra forma de iterar a través de un bucle **for** es basándose en una secuencia de números, y esto se puede realizar con **range()**. La función **range()** genera una secuencia de números. Acepta entradas para el punto de inicio, el punto de parada y el incremento entre paréntesis. Por ejemplo, el siguiente código indica que la secuencia de números comience en **0**, se detenga en **5** y se incremente cada vez en **1**:

**range(0, 5, 1)**

**Nota:** El punto de inicio es inclusivo, lo que significa que el **0** se incluirá en la secuencia de números, pero el punto de parada es exclusivo, lo que significa que el **5** se excluirá de la secuencia. Concluirá un entero antes del punto de parada.

Cuando ejecutes este código, podrás observar cómo el **5** queda excluido de la secuencia:

1

2

for i in range(0, 5, 1):

    print(i)

EjecutarRestablecer

Debes considerar que es necesario incluir el punto de parada, pero si el punto de inicio es el valor predeterminado de **0** y el incremento es el valor predeterminado de **1**, no es necesario especificarlos en el código. Si ejecutas este código, obtendrás los mismos resultados:

1

2

for i in range(5):

    print(i)

EjecutarRestablecer

**Nota:** Si el punto de inicio es distinto de **0** o el incremento es distinto de **1**, deben especificarse.

**Bucles while**

Si deseas que un bucle itere en función de una condición, debes usar un bucle **while**. Mientras la condición sea **True**, el bucle continúa, pero cuando se evalúa como **False**, se sale del bucle **while**. El siguiente bucle **while** continúa mientras la condición de que **i < 5** sea **True**:

1

2

3

4

i = 1

while i < 5:

    print(i)

    i = i + 1

EjecutarRestablecer

En este bucle **while**, el encabezado del bucle es la línea **while i < 5:**. A diferencia de los bucles **for**, el valor de una variable de bucle utilizada para controlar las iteraciones no se asigna dentro del encabezado del bucle en un bucle **while**. En su lugar, se asigna fuera del bucle. En este ejemplo, a **i** se le asigna un valor inicial de **1** en una línea que precede al bucle.

La palabra clave **while** indica el comienzo de un bucle **while**. Después de esto, el encabezado del bucle indica la condición que determina cuándo termina el bucle. Esta condición utiliza los mismos operadores de comparación que las sentencias condicionales. Al igual que en un bucle **for**, el encabezado de un bucle **while** debe terminar con dos puntos (**:**).

El cuerpo de un bucle **while** indica las acciones a realizar en cada iteración. En este ejemplo, es mostrar el valor de **i** e incrementar el valor de **i** en **1**. Para que el valor de **i** cambie con cada iteración, es necesario indicarlo en el cuerpo del bucle **while**. En este ejemplo, el bucle itera cuatro veces hasta que alcanza un valor de **5**.

**Números enteros en la condición del bucle**

Usualmente, como se acaba de demostrar, la condición del bucle se basa en valores enteros. Por ejemplo, puedes optar por permitir que un usuario inicie sesión siempre y cuando lo haya intentado menos de cinco veces. Entonces, tu variable de bucle, **login\_attempts**, se puede inicializar a **0**, incrementar en **1** en el bucle, y la condición del bucle puede especificar iterar solo cuando la variable sea menor que **5**. Puedes ejecutar el código a continuación y revisar el recuento de cada intento:

1

2

3

4

login\_attempts = 0

while login\_attempts < 5:

    print("Login attempts:", login\_attempts)

    login\_attempts = login\_attempts + 1

EjecutarRestablecer

El valor de **login\_attempts** pasó de **0** a **4** antes de que la condición del bucle se evaluara a **False**. Por lo tanto, los valores de **0** a **4** se imprimen, y el valor **5** no se imprime.

**Valores booleanos en la condición del bucle**

Las condiciones en los bucles **while** también pueden depender de otros tipos de datos, incluidas las comparaciones de datos booleanos. En estas, la condición del bucle puede comprobar si una variable del bucle es igual a un valor como **True** o **False**. El bucle itera un número indeterminado de veces hasta que la condición booleana deja de ser **True**.

En el siguiente ejemplo, se usa un valor booleano para salir de un bucle cuando un usuario ha realizado cinco intentos de inicio de sesión. Una variable llamada **count** mantiene un registro de cada intento de inicio de sesión y cambia la variable **login\_status** a **False** cuando **count** es igual a **4**. (Incrementar **count** de **0** a **4** representa cinco intentos de inicio de sesión). Dado que la condición **while** solo itera cuando **login\_status** es **True**, saldrá del bucle. Puedes ejecutar esto para explorar esta salida:

1

2

3

4

5

6

7

count = 0

login\_status = True

while login\_status == True:

    print("Try again.")

    count = count + 1

    if count == 4:

        login\_status = False

EjecutarRestablecer

El código imprime un mensaje para intentarlo de nuevo cuatro veces, pero sale del bucle una vez que **login\_status** se establece en **False**.

**Gestión de bucles**

Puedes usar las palabras clave **break** y **continue** para tener un control adicional sobre las iteraciones del bucle. Ambas se incorporan a una sentencia condicional dentro del cuerpo del bucle. Se pueden insertar para que se ejecuten cuando la condición en una sentencia **if** es **True**. La palabra clave **break** se utiliza para salir de un bucle. Asimismo, la palabra clave **continue** se emplea para omitir una iteración y continuar con la siguiente.

**break**

Cuando quieras salir de un bucle **for** o **while** basándote en que una condición particular de una sentencia **if** sea **True**, puedes escribir una sentencia condicional en el cuerpo del bucle e incluir la palabra clave **break** en el cuerpo de la condicional.

El siguiente ejemplo lo demuestra. La sentencia condicional con **break** indica a Python que salga del bucle **for** si el valor de la variable **asset** del bucle es igual a **"desktop20"**. En la segunda iteración, esta condición se evalúa como **True**. Puedes ejecutar este código para observar esto en la salida:

1

2

3

4

5

computer\_assets = ["laptop1", "desktop20", "smartphone03"]

for asset in computer\_assets:

    if asset == "desktop20":

        break

    print(asset)

EjecutarRestablecer

Como era de esperar, los valores de **"desktop20"** y **"smartphone03"** no se imprimen porque el bucle se rompe en la segunda iteración.

**continue**

Cuando desees omitir una iteración basándote en que cierta condición de una sentencia **if** sea **True**, puedes agregar la palabra clave **continue** en el cuerpo de una sentencia condicional dentro del bucle. En este ejemplo, **continue** se ejecutará cuando la variable del bucle **asset** sea igual a **"desktop20"**. Puedes ejecutar este código para observar cómo difiere esta salida del ejemplo anterior con **break**:

1

2

3

4

5

computer\_assets = ["laptop1", "desktop20", "smartphone03"]

for asset in computer\_assets:

    if asset == "desktop20":

        continue

    print(asset)

EjecutarRestablecer

El valor **"desktop20"** en la segunda iteración no se imprime. Sin embargo, en este caso, el bucle continúa a la siguiente iteración y se imprime **"smartphone03"**.

**Bucles infinitos**

Un bucle que no se detiene se denomina bucle infinito. Si lo creas, debes presionar **CTRL-C** O **CTRL-Z**en tu teclado para detenerlo. Es posible que necesites hacer esto al ejecutar un servicio que procesa datos constantemente, como un servidor web.

**Conclusiones clave**

Los analistas de seguridad deben familiarizarse con las sentencias iterativas. Pueden usar bucles **for** para realizar tareas que impliquen iterar a través de listas una cantidad predeterminada de veces. También pueden emplear bucles **while** para realizar tareas basadas en ciertas condiciones que se evalúan como **True**. Las palabras clave **break** y **continue** se utilizan en sentencias iterativas para controlar el flujo de los bucles basándose en condiciones adicionales.

Marcar como completo

Me gusta

No me gusta

Informar de un problema