LAB-006-Exemplar_Create loops

September 20, 2024

1 Modelo: Crea bucles

Introducción

Como analista de seguridad, algunas de las medidas que tomes para proteger un sistema implicarán la repetición. Por ejemplo, puede que tengas que investigar varias direcciones IP que hayan intentado conectarse a la red. En Python, las sentencias iterativas pueden ayudar a automatizar procesos repetitivos como estos para hacerlos más eficientes.

En este laboratorio, practicarás escribiendo sentencias iterativas en Python.

Consejos para completar este laboratorio

Mientras recorres este laboratorio, ten en cuenta los siguientes consejos:

— `### TU CÓDIGO AQUÍ ###` indica dónde debes escribir el código. Asegúrate de reemplazarlo con tu propio código antes de ejecutar la celda de código. — Siéntete libre de abrir las pistas para obtener información adicional a medida que trabajas en cada tarea. — Para ingresar tu respuesta a una pregunta, haz doble clic en la celda de markdown para editar. Asegúrate de reemplazar "[Haz doble clic para ingresar aquí tus respuestas.]" con tu propia respuesta. — Puedes guardar tu trabajo manualmente haciendo clic en Archivo y luego en Guardar en la barra de menú en la parte superior del cuaderno. — Puedes descargar tu trabajo localmente haciendo un clic en Archivo y luego en Descargar, luego puedes especificar el formato de archivo que prefieras en la barra de menú en la parte superior del cuaderno.

1.1 Situación hipotética

Trabajas como analista de seguridad y escribes programas en Python para automatizar la visualización de mensajes sobre intentos de conexión a la red, la detección de direcciones IP que intentan acceder a datos restringidos y la generación de números de ID de empleados para un departamento de ventas.

Tarea 1

En esta tarea, crearás un bucle relacionado con la conexión a una red.

Escribe una sentencia iterativa que muestre `No se pudo establecer la conexión` tres veces. Utiliza la palabra clave `for`, la función `range()` y una variable de bucle `i`. Asegúrate de reemplazar el fragmento `### TU CÓDIGO AQUÍ ###` con tu propio código antes de ejecutar la siguiente celda.

[1]:
```
# Sentencia iterativa usando `for`, `range()` y una variable de bucle `i`
# Muestra "No se pudo establecer la conexión" tres veces

for i in range(3):
    print("No se pudo establecer la conexión.")
```

```
No se pudo establecer la conexión.
No se pudo establecer la conexión.
No se pudo establecer la conexión.
```

Pista 1

Pasa el número apropiado a la función `range()` para que indique a Python que repita la acción especificada tres veces.

Pista 2

Utiliza `i` como variable de bucle y luego `in` como condición de bucle.

1.2 Tarea 2

La función `range()` también puede tomar una variable. Para repetir una acción especificada un determinado número de veces, primero puedes asignar un valor entero a una variable. A continuación, puedes pasar esa variable a la función `range()` dentro de un bucle `for`.

En tu código que muestra una conexión de mensajes de red, incorpora una variable llamada `connection_attempts`. Asigna el número entero positivo de tu elección como valor de esa variable y rellena la variable que falta en la sentencia iterativa. Asegúrate de reemplazar el fragmento `### TU CÓDIGO AQUÍ ###` con tu propio código antes de ejecutar la siguiente celda. Prueba el código con diferentes valores para `connection_attempts` y observa lo que ocurre.

[2]:
```
# Crea una variable llamada `connection_attempts` que almacene el número de␣
↪veces que el usuario ha intentado conectarse a la red

connection_attempts = 3

# Sentencia iterativa usando `for`, `range()`, una variable de bucle `i` y␣
↪`connection_attempts`
# Muestra "No se pudo establecer la conexión" tantas veces como se especifique␣
↪en `connection_attempts`

for i in range(connection_attempts):
    print("No se pudo establecer la conexión.")
```

```
No se pudo establecer la conexión
No se pudo establecer la conexión
No se pudo establecer la conexión
```

Pista 1

Asigna a la variable `connection_attempts` un número que represente cuántas veces intentará el usuario conectarse a la red.

Pista 2

Pasa la variable apropiada a la función `range()` para que indique a Python que repita la acción especificada el número de veces especificado.

Tarea 3

Esta tarea también se puede realizar con un bucle `while`. Completa el bucle `while` con el código correcto para indicarle que muestre `"No se pudo establecer la conexión"` tres veces.

En esta tarea, un bucle `for` y un bucle `while` producirán resultados similares, pero cada uno se basa en un enfoque diferente. (En otras palabras, la lógica subyacente es diferente en cada uno). Un bucle `for` termina después de que se haya completado un cierto número de iteraciones, mientras que un bucle `while` termina una vez que alcanza una cierta condición. En situaciones en las que no se sabe cuántas veces se debe repetir la acción especificada, los bucles `while` son los más apropiados.

Asegúrate de reemplazar el fragmento `### TU CÓDIGO AQUÍ ###` con tu propio código antes de ejecutar la siguiente celda.

[3]:
```
# Asigna a `connection_attempts` un valor inicial de 0, para llevar la cuenta␣
↪de las veces que el usuario ha intentado conectarse a la red

connection_attempts = 0

# Sentencia iterativa usando `while` y `connection_attempts`
# Muestra "No se pudo establecer la conexión" en cada iteración, hasta que␣
↪connection_attempts alcance un número especificado

while connection_attempts < 3:
    print("No se pudo establecer la conexión")

    # Actualiza `connection_attempts` (increméntalo en 1 al final de cada␣
↪iteración)
    connection_attempts = connection_attempts + 1
```

```
No se pudo establecer la conexión
No se pudo establecer la conexión
No se pudo establecer la conexión
```

Pista 1

En la condición de bucle, usa un operador de comparación para verificar si `connection_attempts` ha alcanzado un número específico. Este número representa el número de veces que se mostrará el mensaje.

Pista 2

En la condición de bucle, usa el operador de comparación `<' para verificar si`connection_attempts‘ es inferior a un número determinado. Este número representa el

número de veces que se mostrará el mensaje.

Pista 3

Utiliza la función `print()` para mostrar el mensaje apropiado al usuario.

Pregunta 1 ¿Qué diferencias observas entre el bucle `for` y el bucle `while` que escribiste?

Los mensajes emitidos por ambos bucles eran idénticos. La lógica es lo que difiere entre los dos bucles. En el bucle `for`, la variable de bucle `i` se definía automáticamente en la cabecera del bucle y se actualizaba automáticamente en cada iteración. En el bucle `while`, la variable de bucle `connection_attempts` tenía que definirse antes de la cabecera del bucle, y tenía que actualizarse explícitamente dentro del cuerpo del bucle.

1.3 Tarea 4

Ahora, pasarás a tu próxima tarea. Automatizarás la verificación de si las direcciones IP forman parte de una lista de permisos. Comenzarás con una lista de direcciones IP desde las que los usuarios han intentado iniciar sesión, almacenada en una variable llamada `ip_addresses`. Escribe un bucle `for` que muestre los elementos de esta lista de uno en uno. Utiliza `i` como variable de bucle en el bucle `for`.

Asegúrate de reemplazar el fragmento `### TU CÓDIGO AQUÍ ###` con tu propio código antes de ejecutar la siguiente celda.

[4]:
```
# Asigna `ip_addresses` a una lista de direcciones IP desde las que los␣
↪usuarios han intentado iniciar sesión

ip_addresses = ["192.168.142.245", "192.168.109.50", "192.168.86.232", "192.168.
↪131.147",
                "192.168.205.12", "192.168.200.48"]

# Bucle `for` que muestra los elementos de `ip_addresses` de uno en uno

for i in ip_addresses:
    print(i)
```

```
192.168.142.245
192.168.109.50
192.168.86.232
192.168.131.147
192.168.205.12
192.168.200.48
```

Pista 1

Utiliza `i` como variable de bucle y el operador `in` para indicar que la acción especificada debe repetirse para cada elemento de la lista `ip_addresses`.

Pista 2

Para mostrar la variable de bucle en cada iteración, utiliza la función `print()` dentro del bucle `for`.

Tarea 5 Ahora te proporcionan una lista de direcciones IP a las que se les permite iniciar sesión, almacenada en una variable llamada `allow_list`. Escribe una sentencia `if` dentro del bucle `for`. Para cada dirección IP de la lista de direcciones IP desde las que los usuarios han intentado iniciar sesión, muestra `"Se permite la dirección IP"` si está entre las direcciones permitidas y muestra `"No se permite la dirección IP"` en caso contrario.

Asegúrate de reemplazar el fragmento `### TU CÓDIGO AQUÍ ###` con tu propio código antes de ejecutar la siguiente celda.

[5]:
```
# Asigna `allow_list` a una lista de direcciones IP a las que se les permite␣
↪iniciar sesión

allow_list = ["192.168.243.140", "192.168.205.12", "192.168.151.162", "192.168.
↪178.71",
              "192.168.86.232", "192.168.3.24", "192.168.170.243", "192.168.119.
↪173"]

# Asigna `ip_addresses` a una lista de direcciones IP desde las que los␣
↪usuarios han intentado iniciar sesión

ip_addresses = ["192.168.142.245", "192.168.109.50", "192.168.86.232", "192.168.
↪131.147",
                "192.168.205.12", "192.168.200.48"]

# Para cada dirección IP de la lista de direcciones IP desde las que los␣
↪usuarios han intentado iniciar sesión,
# Si está entre las direcciones permitidas, muestra "Se permite la dirección IP"
# De lo contrario, muestra "No se permite la dirección IP"

for i in ip_addresses:
        if i in allow_list:
                print("Se permite la dirección IP")
        else:
                print("No se permite la dirección IP")
```

```
No se permite la dirección IP
No se permite la dirección IP
Se permite la dirección IP
No se permite la dirección IP
Se permite la dirección IP
No se permite la dirección IP
```

Pista 1

Utiliza `i` como variable de bucle y el operador `in` para indicar que la acción especificada debe

repetirse para cada elemento de la lista `ip_addresses`.

Pista 2

Asegúrate de que la sentencia `if` comprueba si la dirección IP del usuario está en la lista de direcciones IP permitidas.

Pista 3

Utiliza la función `print()` para mostrar los mensajes.

1.4 Tarea 6

Imaginemos ahora que la información a la que intentan acceder los usuarios está restringida, y si una dirección IP fuera de la lista de direcciones IP permitidas intenta acceder, el bucle debería terminar porque sería necesario investigar más a fondo para evaluar si esta actividad representa una amenaza. Para conseguirlo, usa la palabra clave `break` y amplía el mensaje que se muestra al usuario cuando su dirección IP no está en `allow_list` para proporcionar más detalles. En lugar de `"No se permite la dirección IP"`, muestra `"No se permite la dirección IP. Se requiere una investigación adicional de la actividad de inicio de sesión"`.

Asegúrate de reemplazar el fragmento `### TU CÓDIGO AQUÍ ###` con tu propio código antes de ejecutar la siguiente celda.

[6]:
```
# Asigna `allow_list` a una lista de direcciones IP a las que se les permite␣
↪iniciar sesión

allow_list = ["192.168.243.140", "192.168.205.12", "192.168.151.162", "192.168.
↪178.71",
              "192.168.86.232", "192.168.3.24", "192.168.170.243", "192.168.119.
↪173"]

# Asigna `ip_addresses` a una lista de direcciones IP desde las que los␣
↪usuarios han intentado iniciar sesión

ip_addresses = ["192.168.142.245", "192.168.109.50", "192.168.86.232", "192.168.
↪131.147",
                "192.168.205.12", "192.168.200.48"]

# Para cada dirección IP de la lista de direcciones IP desde las que los␣
↪usuarios han intentado iniciar sesión,
# Si está entre las direcciones permitidas, muestra "Se permite la dirección IP"
# De lo contrario, muestra "No se permite la dirección IP"

for i in ip_addresses:
        if i in allow_list:
                print("Se permite la dirección IP")
        else:
```

```
                print("No se permite la dirección IP". Se requiere una␣
↪investigación adicional de la actividad de inicio de sesión")
                break
```

```
No se permite la dirección IP. Se requiere una investigación adicional de la
actividad de inicio de sesión
```

Pista 1

Utiliza `i` como variable de bucle y el operador `in` para indicar que la acción especificada debe repetirse para cada elemento de la lista `ip_addresses`.

Asegúrate de que la sentencia `if` comprueba si la dirección IP del usuario está en la lista de direcciones IP permitidas.

Utiliza la palabra clave `break` para terminar el bucle en el momento apropiado.

Pista 2

Utiliza la palabra clave `break` dentro de la sentencia `else` después de que se muestre el mensaje apropiado.

Pista 3

Utiliza la función `print()` para mostrar los mensajes.

1.5 Tarea 7

Ahora completarás otra tarea. Se trata de automatizar la creación de nuevos ID de empleado.

Te han pedido que crees ID de empleados para un departamento de ventas, con el criterio de que todos estos ID deben ser números únicos, divisibles por 5 y comprendidos entre 5000 y 5150. Los ID de empleados pueden incluir tanto 5000 como 5150.

Escribe un bucle `while` que genere ID de empleados únicos para el departamento de ventas iterando a través de los números y muestre cada ID creado.

Asegúrate de reemplazar el fragmento `### TU CÓDIGO AQUÍ ###` con tu propio código antes de ejecutar la siguiente celda.

[7]:
```
# Asigna a la variable de bucle `i` un valor inicial de 5000

i = 5000

# Bucle while que genera ID de empleados únicos para el departamento de ventas␣
↪iterando a través de los números
# y muestra cada ID creado

while i <= 5150:
    print(i)
    i = i + 5
```

```
5000
5005
5010
5015
5020
5025
5030
5035
5040
5045
5050
5055
5060
5065
5070
5075
5080
5085
5090
5095
5100
5105
5110
5115
5120
5125
5130
5135
5140
5145
5150
```

Pista 1

Utiliza un operador de comparación para verificar si `i` ha alcanzado el límite superior (que es el número de ID de empleado más alto permitido). Recuerda que los ID de empleado deben estar comprendidos entre 5000 y 5150.

Asegúrate de actualizar el valor de la variable de bucle `i` al final del bucle.

Pista 2

Utiliza el operador de comparación `<=` para verificar si `i` ha alcanzado el límite superior, ya que los ID de empleado deben estar comprendidos entre 5000 y 5150.

Al final del bucle, incrementa la variable de bucle en 5. Esto se debe a que los ID de empleado deben ser divisibles por 5 y el primer ID de empleado se establece en 5000.

Pista 3

Utiliza el operador de comparación `<=` para verificar si `i` ha llegado a 5150, ya que los ID de empleado deben estar comprendidos entre 5000 y 5150.

Utiliza la función `print()` para mostrar la variable de bucle `i` en cada iteración.

Utiliza el operador de asignación `=` y el operador de suma `+` para incrementar el valor de la variable de bucle al final de cada iteración.

1.6 Tarea 8

Te gustaría incorporar un mensaje que muestre `Solo quedan 10 ID de empleado válidos` como una alerta útil una vez que la variable de bucle alcance `5100`.

Para hacerlo, incluye una sentencia `if` en tu código.

Asegúrate de reemplazar el fragmento `### TU CÓDIGO AQUÍ ###` con tu propio código antes de ejecutar la siguiente celda.

[8]:
```
# Asigna a la variable de bucle `i` un valor inicial de 5000

i = 5000

# Bucle while que genera ID de empleados únicos para el departamento de ventas␣
 ↪iterando a través de los números
# y muestra cada ID creado
# Este bucle muestra "Solo quedan 10 ID de empleado válidos" una vez que `i`␣
 ↪alcanza 5100

while i <= 5150:
    print(i)
    if i == 5100:
        print("Solo quedan 10 ID de empleado válidos")
    i = i + 5
```

```
5000
5005
5010
5015
5020
5025
5030
5035
5040
5045
5050
5055
5060
5065
5070
5075
5080
```

```
5085
5090
5095
5100
Solo quedan 10 ID de empleado válidos
5105
5110
5115
5120
5125
5130
5135
5140
5145
5150
```

Pista 1

Utiliza un operador de comparación para verificar si `i` ha alcanzado `5100`.

Pista 2

Utiliza el operador de comparación `==` para verificar si `i` ha alcanzado `5100`.

Pista 3

Utiliza la función `print()` para mostrar el mensaje.

Pregunta 2 ¿Por qué crees que la sentencia `print(i)` está escrita antes de la condicional en lugar de dentro de la condicional?

El objetivo es mostrar cada número de ID de empleado que se crea, y la variable de bucle `i` representa el número de ID creado en cada iteración del bucle. La sentencia `print(i)` se escribe antes de la condicional, para que el bucle se muestre en cada iteración. De lo contrario, si `print(i)` se escribiera dentro de la condicional, la variable de bucle solo se imprimiría cuando fuera igual a 5100. (Ya que la condición en la sentencia `if` es `i == 5100`.)

1.7 Conclusión

¿Qué conclusiones clave obtuviste de este laboratorio? * Las sentencias iterativas desempeñan un papel fundamental en la automatización de los procesos relacionados con la seguridad que deben repetirse.
* Los bucles `for` te permiten repetir un proceso un número determinado de veces. * Los bucles `while` te permiten repetir un proceso hasta que se cumpla una condición específica. En estas condiciones se suelen utilizar operadores de comparación. * El operador de comparación `<` te permite verificar si un valor es menor que otro. * El operador de comparación `<=` te permite verificar si un valor es menor o igual que otro. * El operador de comparación `==` te permite verificar si un valor es igual a otro.