**Más sobre los condicionales en Python**

Anteriormente, exploraste las sentencias condicionales y su utilidad en la automatización de tareas en Python. Hasta ahora, te centraste en las palabras clave **if** y **else**. En esta lección, las repasarás y aprenderás otra palabra clave, **elif**. También, aprenderás cómo puedes aplicar los operadores **and**, **or**, y **not** a tus condiciones.

**Cómo funcionan las sentencias condicionales**

Una **sentencia condicional** es una sentencia que evalúa código para determinar si cumple con un conjunto específico de condiciones. Cuando una condición se cumple, se evalúa como un valor booleano **True** (verdadero) y realiza las acciones especificadas. Cuando la condición no se cumple, se evalúa como un valor booleano **False** (falso) y no realiza las acciones especificadas.

En las sentencias condicionales, la condición suele basarse en la comparación de dos valores. Esta tabla resume los operadores de comparación más comunes utilizados para comparar valores numéricos.

| **operador** | **us0** |
| --- | --- |
| **>** | mayor que |
| **<** | menor que |
| **>=** | mayor o igual que |
| **<=** | menor o igual que |
| **==** | igual que |
| **!=** | no igual que |

**Nota:** Los operadores igual a (**==**) y no igual que (**!=**) también se usan habitualmente para comparar datos de cadena.

**Sentencias if**

La palabra clave **if** inicia una sentencia condicional. Es un componente necesario de cualquier sentencia condicional. En el siguiente ejemplo, **if** comienza una sentencia que le indica a Python que imprima el mensaje **"OK"** cuando el código de estado de la respuesta HTTP sea igual a **200**:

**if status == 200:**

**print("OK")**

Este código consiste en un encabezado y un cuerpo.

**Encabezado de una sentencia *if***

The first line of this code is the header. In the header of an **if** statement, the keyword **if** is followed by the condition. Here, the condition is that the **status** variable is equal to a value of **200**. The condition can be placed in parentheses:

La primera línea de este código es el encabezado. En el encabezado de una sentencia **if**, la palabra clave **if** va seguida de la condición. Aquí, la condición es que la variable **status** sea igual a un valor de **200**. La condición se puede colocar entre paréntesis:

**if (status == 200):**

**print("OK")**

En casos como este, es opcional colocar paréntesis alrededor de las condiciones en Python. Puedes optar por incluirlos si te ayuda a mejorar la legibilidad del código. Sin embargo, esta condición será procesada de la misma manera si se escribe sin paréntesis.

En otras situaciones, debido a que Python evalúa primero las condiciones entre paréntesis, estos pueden afectar la manera en que Python procesa las condiciones. Leerás más sobre estos casos en la sección de esta lectura sobre **not**.

**Nota:** Siempre debes colocar dos puntos (**:**) al final del encabezado. Sin esta sintaxis, el código generará un error.

**Cuerpo de una sentencia if**

Después del encabezado de una sentencia **if***,* viene el cuerpo de la sentencia **if**. Esto le indica a Python qué acción o acciones realizar cuando la condición se evalúa como **True**. En este ejemplo, hay solo una acción, que consiste en imprimir **"OK"** en la pantalla. En otros casos, puede haber más líneas de código con acciones adicionales.

**Nota:** Para que el cuerpo de la sentencia **if** se ejecute según lo previsto, debe tener más sangría que el encabezado. Además, si hay varias líneas de código dentro del cuerpo, todas deben tener la misma sangría.

**Condicionales continuos con else y elif**

En el ejemplo anterior, si el código de respuesta de estado HTTP no fuera igual a **200**, la condición se evaluaría como **False** y Python continuaría con el resto del programa. Sin embargo, también es posible especificar acciones alternativas con **else** y **elif**.

**Sentencias else**

La palabra clave **else** precede a una sección de código que solo se evalúa cuando todas las condiciones que la preceden dentro de la sentencia condicional se evalúan como **False**.

En el siguiente ejemplo, cuando el código de estado de respuesta HTTP no es igual a **200**, imprime un mensaje alternativo de **"check other status"**:

**if status == 200:**

**print("OK")**

**else:**

**print("check other status")**

**Nota:** Al igual que con **if**, se requieren dos puntos (**:**) después de **else**, y el cuerpo que sigue al encabezado **else** está sangrado (o sea, tiene sangría).

**Sentencias elif**

En ocasiones, es posible que tengas varias acciones alternativas que dependen de nuevas condiciones. En ese caso, puedes usar **elif**. La palabra clave **elif** precede a una condición que solo se evalúa cuando las condiciones anteriores se evalúan como **False**. A diferencia de **else**, puede haber varias sentencias **elif** después de **if**.

Por ejemplo, puedes optar por imprimir un mensaje si el código de estado de respuesta HTTP es **200**, un mensaje si es **400** y otro si es **500**. El siguiente código demuestra cómo puedes utilizar **elif** para esto:

**if status == 200:**

**print("OK")**

**elif status == 400:**

**print("Bad Request")**

**elif status == 500:**

**print("Internal Server Error")**

Python verificará primero si el valor de **status** es **200**, y si se evalúa como **False**, pasará a la primera sentencia **elif**. Allí, verificará si el valor de **status** es **400**. Si es **True**, imprimirá **"Bad Request"**, pero si es **False**, pasará a la siguiente sentencia **elif**.

Si deseas que el código imprima otro mensaje cuando todas las condiciones se evalúen como **False**, puedes incorporar **else** después del último **elif**. En este ejemplo, si llega a la sentencia **else**, imprime un mensaje para verificar el estado:

**if status == 200:**

**print("OK")**

**elif status == 400:**

**print("Bad Request")**

**elif status == 500:**

**print("Internal Server Error")**

**else:**

**print("check other status")**

Al igual que con **if** y **else**, es importante colocar dos puntos (**:**) después del encabezado **elif** y sangrar el código que sigue a este encabezado.

**Nota:** Python procesa varias sentencias **elif** de manera diferente a varias sentencias **if**. Cuando llega a una sentencia **elif** que se evalúa como **True**, no verificará las siguientes sentencias **elif**. Por otro lado, Python ejecutará todas las sentencias **if**.

**Operadores lógicos para múltiples condiciones**

En algunos casos, puedes optar por que Python realice una acción basada en una condición más compleja. Podrías requerir que dos condiciones se evalúen como **True**. O bien, que solo una de las dos condiciones se evalúe como **True**. O  puedes optar por que Python ejecute una acción cuando una condición se evalúe como **False**. En estos casos se pueden utilizar los operadores **and**, **or**, y **not**.

**and**

El operador **and** requiere que las dos condiciones a ambos lados del operador se evalúen como **True**. Por ejemplo, todos los códigos de estado de respuesta HTTP entre **200** y **226** se refieren a respuestas correctas. Puedes usar **and** para unir una condición que sea mayor o igual a **200** con otra condición que sea menor o igual a **226**:

**if status >= 200 and status <= 226:**

**print("successful response")**

Si ambas condiciones son **True**, se imprimirá el mensaje **"successful response"**.

**or**

El operador *or* solo requiere que una de las condiciones a ambos lados del operador se evalúe como **True**. Por ejemplo, tanto un código de estado **100** como un código de estado **102** son respuestas informativas. Usando **or**, podrías pedirle a Python que imprima un mensaje de **"informational response"** cuando el código sea **100** o **102**:

**if status == 100 or status == 102:**

**print("informational response")**

Solo es necesario que se cumpla una de estas condiciones para que Python imprima el mensaje.

**not**

El operador **not** niega una condición dada de forma que se evalúe a **False** si la condición es **True** y a **True** si es **False**. Por ejemplo, si deseas indicar que Python debe verificar el código de estado cuando se trata de algo fuera del rango correcto, puedes usar **not**:

**if not(status >= 200 and status <= 226):**

**print("check status")**

Python primero verifica si el valor de estado es mayor o igual a **200** y menor o igual a **226**, y luego, debido al operador **not**, lo invierte. Esto significa que imprimirá el mensaje si **status** es menor que **200** o mayor que **226**.

**Nota:** En este caso, los paréntesis son necesarios para que el código aplique **not** a ambas condiciones. Python evaluará primero las condiciones dentro de los paréntesis. Esto significa que primero evaluará las condiciones a cada lado del operador **and** y luego aplicará **not** a ambas.

**Conclusiones clave**

Es importante que los analistas de seguridad se familiaricen con las sentencias condicionales. Estas requieren la palabra clave **if**. Además, puedes usar **else** y **elif** cuando trabajes con condicionales para especificar acciones adicionales. Los operadores lógicos **and**, **or**, y **not** también son útiles al escribir condicionales.

Marcar como completo

Me gusta

No me gusta

Informar de un problema