**Más información sobre los tipos de datos**

Anteriormente, exploraste los tipos de datos en Python. Un **tipo de datos** es una categoría para una clase particular de elemento de datos. Te enfocaste en los datos de cadena, de lista, float, entero y booleano, que son los tipos de datos con los que trabajarás a lo largo de este curso. En esta lectura, profundizarás tus conocimientos sobre estos tipos de datos y se te presentarán, además, tres tipos adicionales.

**Cadena**

En Python, las **cadenas** son datos que constan de una secuencia ordenada de caracteres. Los caracteres de una cadena pueden incluir letras, números, símbolos y espacios. Estos deben colocarse entre comillas. A continuación, verás una serie de cadenas válidas:

* **"updates needed"**
* **"20%"**
* **"5.0"**
* **"35"**
* **"\*\*/\*\*/\*\*"**
* **""**

**Nota:** El último elemento (**""**), que no contiene nada entre comillas, se denomina cadena vacía.

Puedes usar la función **print()** para mostrar una cadena. Puedes explorar esto ejecutando este código

1

print("updates needed")

EjecutarRestablecer

updates needed

El código imprime **"updates needed"**.

Las cadenas pueden ir entre comillas dobles (**""**) o simples (**''**). El siguiente código demuestra que el mismo mensaje se imprime cuando la cadena está entre comillas simples:

1

print('updates needed')

EjecutarRestablecer

updates needed

**Nota:** Elegir un tipo de comillas y utilizarlo de forma consistente facilita la lectura del código. En este curso, se utilizan comillas dobles.

**Lista**

En Python, **una lista** es una estructura que consta de una serie de datos en forma secuencial. Los elementos de las listas pueden ser de cualquier tipo de datos, como cadenas, enteros, booleanos o, incluso, otras listas. Los elementos de una lista se colocan entre corchetes y cada elemento se separa con una coma. La siguiente lista contiene elementos de varios tipos de datos:

* **[12, 36, 54, 1, 7]**
* **["eraab", "arusso", "drosas"]**
* **[True, False, True, True]**
* **[15, "approved", True, 45.5, False]**
* **[]**

**Nota:** El último elemento **[]**, que no contiene nada entre corchetes, se denomina lista vacía.

También puedes usar la función **print()** para mostrar una lista:

1

print([12, 36, 54, 1, 7])

EjecutarRestablecer

[12, 36, 54, 1, 7]

Esto muestra una lista que contiene los números entero **12**, **36**, **54**, **1**, y **7**.

**Enteros**

En Python, **los datos enteros** son datos formados por un número sin punto decimal. A continuación, verás ejemplos de datos enteros:

* **-100**
* **-12**
* **-1**
* **0**
* **1**
* **20**
* **500**

Los enteros no se colocan entre comillas. Puedes usar la función **print()** para mostrar un número entero. Cuando ejecutas este código, muestra el número **5**:

1

print(5)

EjecutarRestablecer

5

También puedes utilizar la función **print()** para realizar operaciones matemáticas con números enteros. Por ejemplo, este código suma dos enteros:

1

print(5 + 2)

EjecutarRestablecer

7

El resultado es **7**. También, puedes restar, multiplicar o dividir dos números enteros.

**Float**

Los **datos float (de punto flotante)** son datos formados por un número con un punto decimal. A continuación, verás ejemplos de datos flotantes:

* **-2.2**
* **-1.34**
* **0.0**
* **0.34**

Al igual que los datos enteros, los float no se colocan entre comillas. Además, también puedes utilizar la función **print()** para mostrar float o realizar cálculos matemáticos con ellos. Puedes ejecutar el siguiente código para revisar el resultado de este cálculo:

1

print(1.2 + 2.8)

EjecutarRestablecer

4.0

El resultado es **4.0**.

**Nota:** La división de dos valores enteros o dos valores float da como resultado una salida float cuando se utiliza el símbolo **/**:

1

2

print(1/4)

print(1.0/4.0)

EjecutarRestablecer

0.25

0.25

La salida de ambos cálculos es el valor float de **.25**.

Si deseas devolver un valor entero de un cálculo que divide dos números enteros, debes usar el símbolo **//**en su lugar:

1

print(1//4)

EjecutarRestablecer

0

En este caso, la salida es el valor entero **0**, ya que el uso de este símbolo redondea el cálculo de **.25** al entero más cercano.

**Booleanos**

Los datos booleanos son datos que solo pueden tener uno de dos valores: **True**(verdadero) o **False**(falso).

No debes colocar los valores booleanos entre comillas. Al ejecutar el siguiente código, se muestra el valor booleano **True**:

1

print(True)

EjecutarRestablecer

True

También puedes devolver un valor booleano comparando números. Como **9** no es mayor que **10**, este código se evalúa como **False**:

1

print(9 > 10)

EjecutarRestablecer

False

**Tipos de datos adicionales**

En este curso, trabajarás con los tipos de datos cadena, lista, entero, float y booleano, aunque existen otros tipos de datos adicionales. Estos incluyen datos de tupla, de diccionario y de conjunto.

**Tupla**

Una **tupla** consta de un conjunto de datos que no se pueden modificar. Al igual que las listas, las tuplas pueden contener elementos de distintos tipos de datos.

Una diferencia entre los datos de tupla y los de lista es que es posible modificar los elementos en una lista, pero no es posible hacerlo en una tupla. Esto podría ser útil en un contexto de ciberseguridad. Por ejemplo, si se almacenan identificadores de software en una tupla para garantizar que no serán alterados, esto puede brindar la certeza de que una lista de control de acceso solo bloqueará el software previsto.

La sintaxis de una tupla también es diferente de la de una lista. Una tupla se coloca entre paréntesis en lugar de entre corchetes. A continuación, verás ejemplos de tuplas:

* **("wjaffrey", "arutley", "dkot")**
* **(46, 2, 13, 2, 8, 0, 0)**
* **(True, False, True, True)**
* **("wjaffrey", 13, True)**

**Consejo profesional:** Las tuplas son más eficientes en cuanto a memoria que las listas, por lo cual resultan útiles cuando se trabaja con una gran cantidad de datos.

**Diccionario**

Un **diccionario** consiste en una o más parejas clave-valor. Cada clave se asigna a un valor. Entre la clave y el valor se intercalan dos puntos (**:**). Las comas separan las parejas clave-valor de otras parejas clave-valor, y el diccionario se coloca entre llaves (**{}**).

Los diccionarios son útiles para almacenar y recuperar datos de forma predecible. Por ejemplo, el siguiente diccionario asigna un nombre de un edificio a un número. El nombre del edificio es el valor y el número es la clave. Después de la clave se colocan dos puntos.

**{ 1: "East",**

**2: "West",**

**3: "North",**

**4: "South" }**

**Conjunto**

En Python, un **conjunto** es una colección desordenada de valores únicos. Esto significa que no puede haber dos valores iguales en un conjunto.

Los elementos de un conjunto siempre se colocan entre llaves y se separan con una coma. Estos elementos pueden ser de cualquier tipo de datos. Este ejemplo de conjunto contiene cadenas de nombres de usuario:

**{"jlanksy", "drosas", "nmason"}**

**Conclusiones clave**

Es importante que los analistas de seguridad que programan en Python estén familiarizados con los distintos tipos de datos. Los tipos de datos con los que trabajarás en este curso son cadena, lista, entero, float y booleano. Otros tipos de datos adicionales son tupla, diccionario y conjunto. Cada uno tiene su propio propósito y sintaxis.