¿Cuáles son los tipos de Datos en Python?

Debido a que Python se escribe dinámicamente, es importante conocer los tipos de datos que contiene, para que podamos dar el valor adecuado a cada dato con el que trabajemos.

A continuación, menciono los datos que hemos trabajado hasta el Checkpoint 3 del curso:

Booleanos

Los booleanos son datos que indican un dato que solamente permite dos valores: verdadero o falso.

Por ejemplo:

```
my_boolean: True
print(my_boolean)
```

Números

Los números son algunos de los datos más complejos dentro de Python. Pueden ser desde números enteros hasta números con decimales o fracciones, por ejemplo.

He aquí un ejemplo básico:

```
my_number: 50
print(my_number)
```

Strings o cadenas

Las cadenas son secuencias de cualquier tipo de dato. Pueden ser números, palabras, etc. Generalmente, suelen ir dentro de unas comillas y separadas por comas.

Por ejemplo:

```
my_string: 'Oihane'
print(my_string)
```

Listas

Nos sirven para coleccionar elementos, como números, palabras u otro tipo de datos.

Por ejemplo:

```
my_list: [1,2,3,4]
print(my_list)
```

¿Qué tipo de convención de nomenclatura deberíamos utilizar para las variables en Python?

A veces es sencillo nombrar variables, ya que cuentan con una sola palabra (por ejemplo, “name”). Sin embargo, en muchas ocasiones necesitamos utilizar varias palabras para describir una variable. En ese caso, se recomienda utilizar una barra baja entre las palabras (por ejemplo, “name_list”).

Asimismo, es importante que utilicemos nomenclaturas breves, claras y descriptivas, que dejen claro el contenido que viene a continuación.

A pesar de que no es obligatorio hacerlo así (ya que no afecta en los resultados a la hora de imprimir datos), es recomendable seguir dichas pautas. Esto nos permite estandarizar la nomenclatura de las variables para que cualquier persona desarrolladora de Python pueda entender nuestro código.

¿Qué es un Heredoc en Python?

Así como las *strings* o cadenas constan de palabras o frases de una sola línea, el Heredoc consiste en cadenas de varias líneas. A la hora de escribir nuestros párrafos, debemos añadir tres comillas (“) antes y después del párrafo.

Por ejemplo:

```
my_heredoc = """

Me llaman Don Drama. Ruedo por la cama envuelto en llamas. Un pájaro de fuego entre mis sábanas. Todo lo que hago es para hacerte reír. Si te veo llorar otra vez me voy a tener que ir o me va a estallar el pecho aquí mismo, y voy a mancharlo todo de dolor, que es un color feísimo.

"""

print(my_heredoc)
```

¿Qué es una interpolación de cadenas?

Una interpolación de cadenas consiste en sustituir una variable por su valor dentro de una cadena. Para ello, creamos un texto que mencione las cadenas en las que después insertaremos los valores que queramos.

Es necesario que añadamos una “F” antes de incluir el texto, para que Python interprete que debe sustituir el contenido de los corchetes. Si no ponemos esa “F”, se imprimirá la frase con los corchetes, sin la sustitución de los datos.

```
nombre = 'Oihane'

color = 'negro'

saludo_automatico = f’Hola, soy {nombre} y mi color favorito es el {color}.’

print(saludo_automatico)
```

¿Cuándo deberíamos usar comentarios en Python?

Los comentarios, que en Python se redactan comenzando con una almohadilla (#), son recomendables siempre y cuando sirvan para agilizar y facilitar el trabajo. En ocasiones, puede ocurrir que tengamos un exceso de comentarios innecesario, o que se nos olvide actualizarlos o borrarlos, y terminen generando más confusión por no estar en sintonía con los datos.

Sin embargo, los comentarios pueden ser muy útiles para organizar los datos. Podemos utilizar los comentarios para dejar notas sobre el comienzo de una sección, como si fuesen títulos de lo que viene a continuación. Además de ser útiles para quienes hemos creado esos datos, permitirán que cualquier persona que acceda a los datos pueda entender mejor la organización de estos.

Adicionalmente, al utilizar comentarios para organizar datos, es menos probable que esos comentarios se queden obsoletos o requieran de actualizaciones constantes, ya que son meramente descriptivos de lo que aparece a continuación y, a pesar de que cambiemos algunos datos, seguirán sirviendo como referencia.

¿Cuáles son las diferencias entre aplicaciones monolíticas y de microservicios?

Por una parte, las arquitecturas monolíticas consisten en configuraciones con sistemas que se basan en una sola aplicación. Esto significa que pone todas las funciones en un solo proceso.

Son más fáciles de desarrollar, porque no tienen que comunicarse con diversos servicios. Sin embargo, el mantenimiento se puede complicar si las aplicaciones no están bien diseñadas y crear un efecto dominó con cualquier cambio que se haga.

Por otra parte, las arquitecturas de microservicios, en vez de poner todas las funciones en un solo proceso, rompen esos procesos y crean varios servicios dentro de la aplicación. Cada servicio se puede desarrollar de forma individual y se ejecutarán por separado dentro de la aplicación.

Esto significa que tienen una mayor capacidad de ampliación respecto a las arquitecturas monolíticas, porque se pueden aislar los errores o problemas y se pueden encontrar las soluciones de una forma mucho más rápida. Es decir, si algo falla, no impacta en todo el proceso.