¿Cuáles son los tipos de Datos en Python?

Las variables pueden almacenar distintos tipos de datos y según esto podremos realizar unas tareas u otras con ellos. Los nueve tipos de Datos principales en Python son los siguientes:

- **Strings**: Son cadenas de texto. Van encuadradas en comillas simples o dobles cuando van en una sola línea: '_' o "_". Pero cuando queremos hacer una cadena de varias líneas tenemos que usar triples comillas, que también pueden ser triples o dobles. Cuando queremos asignar una cadena a una variable debemos declarar la variable y preceder la cadena de un signo de *igual*. Por ejemplo:
  my_string = "Hello world"

- **Numbers**: Pueden ser cualquier tipo de valor numérico, números enteros, decimales, floats, etc.

- **Booleans**: Almacenan valores de verdadero (True) o falso (False) y permiten crear condicionales. Por ejemplo: user_login = True.

- **Bytes y byte Arrays**: se usan para un tipo de trabajo muy complejo dentro de python como editar secuencias de bytes o editar imágenes a nivel de bytes

- **None**: Es un valor nulo que se usa cuando se quiere definir una variable pero no le queremos asignar un valor concreto aún. Esto te permite volver más tarde y cambiar el valor None para poner el tipo de dato y valor que necesites.

- Estructuras de datos: Los siguientes tipos corresponden a una categoría llamada estructuras de datos. Sirven para gestionar colecciones dentro de Python y cada uno de los tipos tiene una función específica:

Lists: Son listas de elementos. Los elementos van entre comillas y separados por comas y el conjunto entero va rodeado por corchetes. Por ejemplo: my_list = ['chocolate', 'leche', 'azúcar']

Tupples y **Sets**: Muy parecidos a las listas pero cada uno con distintas características

- **Dictionaries**: Son como las listas pero permiten usar palabras clave para encontrar y reemplazar palabras.

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

Para nombrar variables en Python debemos abstenernos de repetir nombres y sobre todo cuando ese nombre lo hemos usado primero para declarar otro tipo de dato pues esto puede dar lugar a numerosos bugs a lo largo del programa. Si la variable tiene varias

palabras estas deben ir separadas por un guión bajo y todas las palabras deben ir en minúsculas. Por ejemplo, si tenemos una variable llamada *Mi Frase,* la manera correcta de nombrarla sería: mi_frase.

¿Qué es un Heredoc en Python?
Un Heredoc es simplemente una cadena de texto multilínea en Python. Cuando escribimos una cadena hay veces que puede ir en una misma línea pero otras veces el texto excede una sola línea o queremos redactar varios párrafos, etc. Cuando nuestra cadena tiene más de una línea la llamamos Heredoc, y para que no causemos un error en el programa necesitamos encuadrar la cadena entera en triples comillas. También para que no se cuenten la primera y la última línea podemos aplicar la función strip(). Por ejemplo:

my_var = ‘ ‘ ‘
Hola

Buenos días

Adiós
‘ ‘ ‘.strip()

¿Qué es una interpolación de cadenas?
La interpolación de cadenas nos permite procesar código de Python dentro de cadenas de texto. Por ejemplo, si declaramos una variable
nombre = María y luego declaramos la variable saludo, pero antes de nuestro texto añadimos la letra f:
var_saludo = f'Hola {nombre}, cómo estás?'.
Este tipo de comportamiento es el que se implementa en los emails y mensajes personalizados y es por esto que la interpolación de cadenas es un aspecto tan práctico de Python.

¿Cuándo deberíamos usar comentarios en Python?
El problema de usar comentarios en Python es que muchas veces estos son estáticos mientras que el código tiende a cambiar y evolucionar. Normalmente los métodos van cambiando pero los comentarios iniciales que los definen no se modifican con ellos, por tanto corremos el riesgo de tener un comentario que ya no es útil e incluso puede llevar a cometer errores. La mejor práctica es redactar código ‘self-explanatory’ o que se explique sólo, es decir, que sea tan claro y comprensible que no sea necesario comentarlo en primer lugar.

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

Las aplicaciones monolíticas son aquellas en las que todas las funciones del sistema están basadas en la misma aplicación. Y al estar todo concentrado en el mismo lugar permite un desarrollo más veloz y escalar el proyecto inicial de manera sencilla a lo largo del tiempo. El aspecto negativo principal es que al ser monolíticas, un error o modificación en una parte del sistema puede afectar al sistema entero, con lo cual son mucho más delicadas.

Por otro lado, las aplicaciones microservicios permiten fragmentar los distintos procesos y desarrollar distintas funciones a lo largo de la aplicación que pueden ser creadas y testadas de forma independiente. Cada servicio que forma parte de la aplicación funcionará de forma independiente una vez que esta se inicie. La ventaja principal de este método es que cada parte de la aplicación se puede modificar, escalar y desarrollar de forma independiente sin que afecte al resto del conjunto. Un aspecto negativo de este sistema es que al necesitar muchas comunicaciones dentro de la misma aplicación esto puede dar lugar a retrasos si estas no funcionan bien.