¿Cuáles son los tipos de Datos en Python?

Enteros (int): Representan números enteros sin parte decimal. Se escriben sin comillas y pueden ser positivos o negativos. Ejemplo:

numero_entero = 10

Beneficios: Los enteros son útiles para representar recuentos, índices, y en general cualquier cantidad que no tenga parte fraccionaria.

Flotantes (float): Representan números con parte decimal. Se escriben con un punto decimal o en notación científica. Ejemplo:

numero_flotante = 3.14

Beneficios: Los flotantes son útiles cuando se necesitan representar valores que pueden tener decimales, como resultados de cálculos matemáticos.

Cadenas de caracteres (str): Representan secuencias de caracteres, es decir, texto. Se escriben entre comillas simples (' ') o dobles (" "). Ejemplo:

cadena = "Hola, mundo!"

Beneficios: Las cadenas de caracteres son esenciales para manejar texto, ya sea para mostrar información al usuario, manipular archivos de texto, entre otros.

Booleanos (bool): Representan valores de verdad, es decir, verdadero o falso. Se escriben como True o False. Ejemplo:

verdadero = True

Beneficios: Los booleanos son fundamentales para realizar operaciones de lógica y control de flujo, como condicionales y bucles.

Listas (list): Representan secuencias ordenadas de elementos que pueden ser de cualquier tipo. Se escriben entre corchetes [ ]. Ejemplo:

lista = [1, 2, 3, 4, 5]

Beneficios: Las listas son flexibles y versátiles, permitiendo almacenar múltiples valores en una sola estructura de datos, y se utilizan para almacenar y manipular colecciones de elementos.

Diccionarios (dict): Representan pares de clave-valor donde cada clave está asociada a un valor. Se escriben entre llaves { }, separando cada par clave-valor por comas y utilizando dos puntos para separar la clave del valor. Ejemplo:

diccionario = {"clave1": valor1, "clave2": valor2}

Beneficios: Los diccionarios son eficientes para buscar y acceder a datos a través de claves, y son útiles para representar estructuras de datos complejas.

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

Snake Case: En este estilo, todas las letras están en minúsculas y las palabras están separadas por guiones bajos ('_'). Se utiliza para nombres de variables, funciones y métodos.

Ejemplo:

nombre_variable = 10

nombre_funcion = mi_funcion()

Beneficios: Este estilo es fácil de leer y entender, lo que facilita la legibilidad del código, especialmente para nombres de variables más largos o compuestos.

¿Qué es un Heredoc en Python?

son cadenas de multiples lineas o de texto

Los beneficios de usar cadenas de texto de varias líneas son:

Legibilidad: Las cadenas de texto multilínea hacen que el código sea más legible, especialmente cuando necesitas incluir texto largo o estructurado.

Facilidad de escritura: No es necesario usar caracteres de escape para representar saltos de línea, lo que hace que la escritura de texto largo sea más fácil y menos propensa a errores.

Mantenimiento: Al utilizar cadenas de texto de varias líneas, el texto está claramente separado del código, lo que facilita el mantenimiento y la actualización del texto sin modificar el código en sí.

¿Qué es una interpolación de cadenas?

La interpolación de cadenas es un proceso mediante el cual se insertan valores de variables dentro de una cadena de texto de forma dinámica. Es una técnica muy útil para construir cadenas de texto complejas donde se necesitan combinar valores de variables con texto estático.

```
nombre = "Pedro"
edad = 35
mensaje = f"Hola, me llamo {nombre} y tengo {edad} años."
print(mensaje)
```

¿Cuándo deberíamos usar comentarios en Python?

Explicar el propósito de una sección de código: Los comentarios son útiles para explicar qué hace una sección de código, especialmente si es compleja o difícil de entender de inmediato.

Documentar funciones y métodos: Es una buena práctica incluir comentarios que describan qué hace una función o método, qué argumentos espera y qué devuelve.

Aclarar decisiones de diseño o implementación: Si hay decisiones específicas detrás de la implementación de una parte del código, los comentarios pueden ser útiles para explicarlas.

Indicar intenciones futuras o tareas pendientes: Si hay partes del código que necesitan ser revisadas o mejoradas en el futuro, los comentarios pueden servir como recordatorios para el desarrollador.

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

Aplicaciones Monolíticas: Toda la aplicación se desarrolla como una sola unidad, lo que facilita el despliegue y mantenimiento, pero puede ser menos escalable y más propensa a fallos.

Aplicaciones de Microservicios: La aplicación se divide en servicios pequeños e independientes, lo que permite una mayor escalabilidad, flexibilidad y resiliencia, pero puede ser más compleja de desplegar y mantener.