¿Cuáles son los tipos de Datos en Python?

- String

Las cadenas pueden ser de cualquier tamaño. Puede ser un símbolo, una palabra, una frase o incluso un texto de varias líneas (Heredoc). Las cadenas siempre deben estar entre comillas simples o dobles.

```
"""
cadena1 = 'Hola!'
cadena2 = "Hola!"
"""
```

- Number

Puede ser un número entero (2025), un flotante (3.5) o una fracción (1/3).

- Boolean

Puede tener un valor booleano: True or False

- Bytes and byte arrays
- None

valor vacío, no contiene nada

- Lists

similar a las matrices en otros lenguajes de programación

- Tuples
- Sets
- Dictionaries

contener pares clave-valor

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

- Existen varias convenciones que se utilizan para nombrar variables en Python. Varias palabras del título están separadas por un guión bajo y escrito en letras minúsculas. Se llama **snake_case.**
- El nombre de la variable no puede comenzar con un número.
- Es mejor evitar el uso de variables de una sola letra, como **x, y, a, b** y otros.
- El nombre de la variable debe ser claro y descriptivo, para que quede claro lo que se tiene en su interior (**user_name, product_price**).
- Las letras minúsculas y mayúsculas tienen significado. **user_name** y **User_name** son diferentes variables.

En detalle, todos los matices de la denominación de variables se pueden encontrar en las “Naming Conventions”, que están prescritas en PEP 8 - el estándar de sintaxis del código Python https://peps.python.org/pep-0008/#naming-conventions

¿Qué es un Heredoc en Python?

Heredoc es el nombre de una cadena multilínea en Python. Para crear una cadena multilínea, debe estar entre comillas triples simples o dobles.
Si escribe una cadena multilínea como una sola línea normal ('text'), se producirá un SyntaxError.
Sintaxis correcta:

```
"""
multi = '''
String
Number
Boolean
```

```
Bytes and byte arrays
None
Lists
Tuples
Sets
Dictionaries
'''
"""
```

o

```
"""
multi = """
String
Number
Boolean
Bytes and byte arrays
None
Lists
Tuples
Sets
Dictionaries
"""
"""
```

Dentro de un heredoc siempre se tienen en cuenta los saltos de línea.
Para eliminar la primera y la última línea vacía, puede cambiar la entrada del código evitando saltos de línea entre texto y comillas:

```
"""
multi = """String
Number
Boolean
Bytes and byte arrays
None
Lists
Tuples
Sets
Dictionaries"""
"""
```

o agregue la función strip al final, también eliminará la primera y la última línea vacía

```
"""
multi = """
String
Number
Boolean
Bytes and byte arrays
None
```

```
Lists
Tuples
Sets
Dictionaries
""".strip()
"""
```

¿Qué es una interpolación de cadenas?

La interpolación de cadenas es el proceso de insertar valores de variables o expresiones dentro de una cadena de texto de manera dinámica. Esto le permite crear cadenas no codificadas fijo, sino con datos que cambian dinámicamente. La letra f se usa para interpolar cadenas y {} se usa para insertar un valor dinámico.
Por ejemplo:

```
"""
user_name = 'Mary'
text = f'Hello, {user_name}!'
"""
```

el resultado será - Hello, Mary!

Cuando configuramos la f, Python busca llaves en la cadena y lee todo lo que hay dentro de ellas como código Python, ¡y no como contenido normal! escapar de los paréntesis con una barra invertida no funcionará \{. Para mostrar corchetes, debe escribirlos dos veces {{text}}, entonces esto se considera llaves de contenido único.

¿Cuándo deberíamos usar comentarios en Python?

Un comentario nos permite agregar contexto y documentación adicional al código.
No es necesario comentar todo, solo los componentes importantes. Es mejor utilizar nombres de variables significativos siempre que sea posible.
Los comentarios bien escritos hacen que el código se autodocumente.
Hay varias formas de escribir un comentario en Python.
Un comentario de una línea se escribe después de #

```
# este es un comentario
```

Un comentario de varias líneas cabe entre dos líneas con comillas dobles en la parte superior e inferior. Cada línea contiene tres comillas.

```
"""
este es un comentario
publicado en varias líneas
"""
```

También puedes escribir un comentario en línea en la línea de código al final de esta línea.

```
print('text') # comentario
```

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

La arquitectura monolítica es la configuración utilizada para los sistemas de servidores tradicionales. La arquitectura monolítica es más rápida de construir y puede tener más capacidad de respuesta. Este es un conjunto de archivos, un conjunto de funciones que se encuentra en un servidor y todos los componentes interactúan entre sí dentro de la misma arquitectura de una aplicación.

En una arquitectura de microservicio, las funciones pueden ser aplicaciones separadas (autenticación separada, publicaciones separadas, etc.). Estas son entidades separadas, donde cada una está ubicada en su propio servidor y se comunican entre sí como API separadas.

Mantener aplicaciones monolíticas puede resultar todo un desafío. Hacer un cambio puede tener un efecto dominó y causar una serie de problemas en diferentes partes del código.

En un microservicio los elementos no dependen tanto unos de otros. Se desarrollan dentro de la misma arquitectura, pero cada uno por separado.
La mayor ventaja de los microservicios es la capacidad de escalar componentes individuales en lugar de la aplicación completa. Si algún componente falla, no falla todo el sistema.

Para aplicaciones pequeñas los microservicios no tienen sentido (lleva más tiempo y es más caro), pero para servicios grandes puede ser una muy buena solución. Mucho depende del tiempo, el presupuesto y la experiencia de los desarrolladores.