1. **¿Cuáles son los tipos de Datos en Python?**

Booleans: Representan valores de verdad (True) o falso (False)

es_mayor_de_edad = True

es_adulto = False

Numbers: Representan números. Pueden representar números enteros, números decimales o números complejos.

numero_entero = 5

numero_decimal = 3.14

Strings: Representan cadenas de texto. Se delimitan por comillas simples o dobles.

cadena_de_texto = "Hola Mundo"

cadena_de_texto ='Hola Mundo'

Bytes and byte arrays: Representas secuencias inmutables de bytes (bytes), o secuencias modificables de bytes (bytearray)

None: Representa la ausencia de valor o valor nulo.

variable_vacia=None

Lists: Representa una estructura de datos ordenada que se puede modificar. Puede contener distintos tipos de datos.

lista =[1,2,3,4]

lista =[1,2,3, 'Hola']

Tuples (Igual que la lista pero, a diferencia de ella, no permiten modificar los elementos)

tupla =(1,2,3, 'Hola')

Sets (Representa un conjunto de elementos. No permiten elementos duplicados)

conjunto ={1,2,3,4}

Dictionaries (Estructura que almacena los datos con clave:valor. Cada clave es única)

diccionario = {"nombre":"Juan", "edad":25}

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

Se usa la convención snake_case para nombrar variables. Significa que los nombres deben estar en minúsculas y separar las palabras con guion bajo.

Otras reglas que hay que seguir son:

- Usar nombres descriptivos claros.
  nombre_usuario:
  edad_usuario:

- No empezar el nombre con números ni caracteres especiales.

- No usar palabras reservadas de Python (if, for, ...).

- Para valores que no cambian, recomiendan mayúsculas:
  PI = 3.1416

3. **¿Qué es un Heredoc en Python?**

Un Heredoc es una manera de crear cadenas de texto multilínea. Es útil cuando necesitamos almacenar o trabajar con textos largos que ocupan varias líneas, como párrafos o descripciones. Para conseguirlo, se utilizan comillas triples (dobles " o no ') que permiten que el string se extienda todas las líneas que sea necesario.

ejemplo_heredoc= """The world is changed. I feel it in the water. I feel it in the Earth. I smell it in the air. Much that once was is lost. For none now live who remember it.

It began with the forging of the great rings. Three were given to the Elves, immortal, wisest and fairest of all beings. Seven to the Dwarf lords, great miners and craftsmen of the mountain halls. And nine, nine rings were gifted to the race of Men, who above all else, desire power. """.strip()

Lo que hace .strip() al final del texto es eliminar saltos de línea adicionales que crean las comillas.

4. **¿Qué es una interpolación de cadenas?**

La interpolación de cadenas en Python permite incluir valores de variables y expresiones directamente dentro de una cadena de texto sin necesidad de concatenar manualmente. Es útil porque permite que los programas creados sean más dinámicos. En Python, la mejor forma de hacerlo es usando *f-strings*

1. **Solo texto**

   nombre = "June"

   edad = 39

   texto = f"Hola, me llamo {nombre} y tengo {edad} años."

   print(texto)

   Saca por pantalla = Hola, me llamo June y tengo 39 años.

2. **Otras expresiones de Python, como operaciones matemáticas:**

   a= 10

   b = 5

   resultado = f"La suma de {a } y {b } es {a + b}."

   print(resultado)

   Saca por pantalla = La suma de 10 y 5 es 15.

Otra forma de hacerlo es usando .format():

1. **Solo texto**

   nombre = "June"

   edad = 39

   texto = "Hola, me llamo {} y tengo {} años.".format(nombre, edad)

   print(texto)

   Saca por pantalla = Hola, me llamo June y tengo 39 años.

2. **Otras expresiones de Python, como operaciones matemáticas:**

   a = 10

   b = 5

   resultado = "La suma de {} y {} es {}.".format(a, b, a + b)

   print(resultado)

   Saca por pantalla = La suma de 10 y 5 es 15.

5. **¿Cuándo deberíamos usar comentarios en Python?**

Los comentarios en Python sirven para documentar el código y hacer que sea más comprensible. Existen tres formas principales de añadir comentarios en un programa:

1. **Comentarios de una sola línea:**
   Se utilizan con el símbolo # y sirven para añadir notas rápidas en el código.

   ```
   print("Hola Mundo")  # Esto es un comentario al final de la línea
   ```

2. **Comentarios en línea (inline comments):**
   Se colocan en la misma línea de una instrucción de código para explicar su función.

   ```
   nombre = "June"  # Variable que almacena el nombre de la persona
   ```

3. **Comentarios de varias líneas:**
   Se pueden escribir con triples comillas, como un string de múltiples líneas ya que Python los trata como tal.

   ```
   """
   Esto es un comentario
   de múltiples líneas.
   """
   ```

Los comentarios con # son los más comunes y recomendados. Se recomienda también solo comentar partes esenciales del código y crear nombres de variables lo suficientemente descriptivas para evitar los comentarios en la medida de lo posible.

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

Arquitectura Monolítica:

- Toda la aplicación funciona como un solo sistema.
- Es más rápida de desarrollar y ejecutar.
- Puede volverse difícil de mantener debido al fuerte acoplamiento entre sus componentes.

Arquitectura de Microservicios:

- Divide la aplicación en pequeños servicios independientes.
- Facilita la escalabilidad y la detección de errores.
- Su desarrollo es más complejo y puede generar problemas de compatibilidad entre servicios.