M2C3- Python Preguntas Teóricas

1. ¿Cuáles son los tipos de datos en Python?

   Existen nueve tipos de datos en Python y son estos :

   a. Booleans => Representan valores True o False .

   b. Numbers => Se pueden representar , Enteros (int) , Decimales(float), Fracciones, Complejos ,Consideraciones.

   c. Strings => Secuencias de caracteres como nombres o documentos.

   d. Bytes y Byte Arrays => Sirve para datos binarios y manipulación avanzada. Son matrices de bytes que contienen valores entre 0 y 255 . La principal diferencia entre ambos es que los bytes son inmutables, mientras que los bytearray son mutables.

   e. None => Indica ausencia de valor, se utiliza para definir una variable cuando aún no se le quiere asignar ningún valor.

   f. Lista (list) => Estructuras mutables, dinámicas (se puede modificar) .

   g. Tuplas (tuple) => Estructuras inmutables (no se pueden modificar).

   h. Sets (set) => Colección de elementos únicos , sin orden específico.

   i. Diccionarios (dict) => Almacenan datos en pares clave-valor.

2. Que tipo de convención de nomenclatura deberíamos utilizar para las variables en Python?

   a. Las variables , funciones y métodos deben escribirse en **snake_case,** lo que significa que se usa _ entre palabras .
   b. **PascalCase** , se usa para clases .
   c. SCREAMING_SNAKE_CASE , se usa para constantes.

M2C3- Python Preguntas Teóricas

3. ¿Qué es un Heredoc en Python?

    a. En Python , un heredoc se representa con comillas triples (”””o ```). Lo usamos para manejar cadenas de texto multilínea de manera sencilla. Es una manera eficiente de mantener el formato de las líneas sin tener que agregar manualmente saltos de línea con(\n) .
    b. También se usa para documentar funciones , clases o módulos a través de los **Docstrings.**

    c. Ejemplos Heredoc

```
cadena_multilinea = """ Esta es una cadena
multilínea en el cual podemos escribir
texto de varias lineas """

print(cadena_multilinea)
```

        i. Ejemplo de Docstring

```
def mi_ function() :
  '''
  Esta función nos permite demostrar
  el uso de docstring , que son comentarios
  multilínea '''

  pass

  help(mi_function)
```

Estamos usando pass porque es una función vacía y por otro lado con help(my function()) nos estamos ayudando a que si lo imprimimos podemos ver nuestro **Docstring .**

M2C3- Python Preguntas Teóricas

4. ¿Qué es una interpolación de cadena ?

    a. La interpolación de cadenas es una forma de insertar valores dentro de un texto en Python. En lugar de escribir una cadena fija(Estática) , podemos combinar texto con valores dinámicos (variables , resultados de operaciones, etc.).
    b. Es útil porque hace que el código sea más fácil de leer y escribir .
    c. Nos permite insertar valores dentro de un texto
    d. Podemos usarlo de dos formas con el método f-strings(f"...") que es el método más moderno y con el método .format() que es mas antiguo pero todavía podemos encontrarnos en algunos proyectos antiguos .

5. ¿Cuándo deberíamos usar comentarios en Python ?

    Debemos usar comentarios para explicar que hace nuestro código , por que lo hace y cómo lo hace. Tanto para ayudarnos a nosotros como a otros programadores que revisen o modifiquen nuestro código.

    a. Debemos usar comentarios cuando :

        i. Al inicio de un bloque de código complejo.
        ii. Antes de líneas de código críticas.
        iii. Para explicar decisiones no obvias.
        iv. Al definir una función o clase.
        v. Para explicar correcciones de errores o parches temporales.
        vi. Para resaltar ciertas líneas o bloques enteros para mejorarlos.

M2C3- Python Preguntas Teóricas

6. ¿Cuáles son las diferencias entre aplicaciones monolíticas y de microservicios ?
    a. Las aplicaciones monolíticas son una sola unidad , mientras que las aplicaciones de microservicios son un conjunto de servicios independientes.
        i. Lo que las hace diferentes son estas cinco cosas: la Implementación , Escalabilidad , Flexibilidad , Desarrollo y las Actualizaciones.
            1. Monopólicas :
               Todo el código base está en un solo entorno , se escala todo como una unidad , tiene menos flexibilidad , es mucho más sencillo de desarrollar y mantener y se necesita actualizar todo el código base.
            2. Microservicios :
               Cada microservicio es un paquete de software independiente, se puede escalar de forma individual , es mucho más flexible y eficiente , también es más complejo de implementar y se pueden actualizar los microservicios de forma independiente.