1. **Introducción al lenguaje y a su entorno de desarrollo:**
    - ¿Qué es un lenguaje de programación y cuál es su importancia en el desarrollo de software?

      Un lenguaje de programación es un conjunto de reglas y símbolos que permite a los programadores escribir instrucciones que una computadora puede entender y ejecutar. La importancia de un lenguaje de programación en el desarrollo de software radica en su capacidad para facilitar la creación de aplicaciones, automatizar tareas, procesar datos y resolver problemas complejos. Por ejemplo, lenguajes como Python y JavaScript permiten el desarrollo de aplicaciones web, mientras que C++ se utiliza en sistemas embebidos y software de alto rendimiento.
    - Menciona y compara tres entornos de desarrollo integrados (IDEs) utilizados en la actualidad.

Visual Studio

Lenguajes soportados: C#, C++, VB.NET.

Ventajas: Potentes herramientas de depuración, integración de bases de datos y un diseño muy amigable.

Desventajas: Consume muchos recursos y puede ser excesivo para proyectos pequeños.

Eclipse

Lenguajes soportados: Principalmente Java, pero soporta otros mediante plugins.

Ventajas: Gran capacidad de extensión y soporte para desarrollo en Java.

Desventajas: Puede ser lento al cargarse y tiene una curva de aprendizaje elevada.

PyCharm

Lenguajes soportados: Python.

Ventajas: Excelente autocompletado de código y herramientas de análisis.

Desventajas: La versión gratuita tiene limitaciones en funcionalidades avanzadas.

2. **Estructura de un programa:**

    - Explica la estructura general de un programa en el lenguaje que estás estudiando.

      Tomando como ejemplo Python, una estructura básica de un programa podría ser:

      ```
      def main():
          # Código principal aquí
          print("Hola, Mundo!")

      if __name__ == "__main__":
          main()
      ```

      La estructura general de un programa incluye:

- **Funciones:** Bloques de código reutilizables.
- **Sentencias condicionales:** Para ejecutar código basado en condiciones.
- **Bucles:** Para repetir bloques de código

    - ¿Por qué es importante seguir una estructura clara en la programación?

      Seguir una estructura clara es crucial porque:

- Facilita la lectura y el mantenimiento del código.
- Reduce la posibilidad de errores y confusiones.
- Permite la colaboración efectiva entre varios desarrolladores.
- Mejora la capacidad para escalar y extender el software en el futuro.

3. **Identificadores (Variables y constantes):**

    - Explica la diferencia entre una variable y una constante.

Variable: Es un identificador que puede cambiar su valor durante la ejecución del programa.

Constante: Es un identificador cuyo valor no puede cambiar una vez asignado

- Proporciona ejemplos de cómo se declaran y utilizan en un lenguaje de programación.

.

Ejemplo en Python:

```
python

# Variable
edad = 30
edad = 31  # Valor cambiado

# Constante (convenio de nombres)
PI = 3.14  # No se espera que este valor cambie
```

4. **Tipos de datos:**
   - ¿Qué son los tipos de datos y por qué son importantes en la programación?

     Los tipos de datos son categorías que determinan el tipo de valor que una variable puede almacenar, así como las operaciones que se pueden realizar sobre esos valores. Son importantes porque afectan la forma en que manejamos los datos en la aplicación y optimizan el uso de memoria.
   - Clasifica los tipos de datos más comunes y da un ejemplo de cada uno.

   Numéricos: Representan números enteros (int) y decimales (float).
   - Ejemplo: `edad = 25` (int), `altura = 1.75` (float).

   Cadenas de texto: Representan texto.
   - Ejemplo: `nombre = "Juan"`.

   Booleanos: Representan valores de verdad (True o False).
   - Ejemplo: `activo = True`.

5. **Operadores aritméticos, lógicos y relacionales:**

- Explica la diferencia entre operadores aritméticos, lógicos y relacionales.
- Proporciona ejemplos prácticos de su uso.

Operadores aritméticos: Realizan operaciones matemáticas.

- Ejemplo: `suma = a + b`.

Operadores lógicos: Manejan operaciones booleanas.

- Ejemplo: `if (a > 10 and b < 5):`.

Operadores relacionales: Comparan valores y devuelven un booleano.

- Ejemplo: `if (a == b):`.

6. **Programación estructurada:**
   - ¿En qué consiste la programación estructurada y cuáles son sus principios fundamentales?

     La programación estructurada es un paradigma que promueve la creación de programas de software de manera más comprensible y organizada. Sus principios fundamentales incluyen:

- **Secuencia:** Ejecución lineal de instrucciones.
- **Selección:** Toma de decisiones basada en condiciones.
- **Iteración:** Repetición de bloques de código.

   - Explica cómo se relaciona con el concepto de funciones o procedimientos en la programación.

     Las funciones son bloques de código que encapsulan tareas específicas y se pueden llamar en diferentes partes del programa, lo que refuerza la modularidad y la reutilización del código, facilitando la implementación de los principios de la programación estructurada.

7. **Programación orientada a objetos:**
   - ¿Qué es la programación orientada a objetos (POO) y cómo se diferencia de la programación estructurada?

La programación orientada a objetos es un paradigma que organiza la programación alrededor de "objetos", que son instancias de clases. Se diferencia de la programación estructurada en que POO permite encapsular datos y comportamientos, promoviendo la reutilización del código y la jerarquía.

- Explica los conceptos de clase y objeto con un ejemplo.

Clase: Es un blueprint que define el comportamiento y las propiedades de los objetos.

Objeto: Es una instancia de una clase.

Ejemplo en Python:

```
python

class Persona:
    def __init__(self, nombre, edad):
        self.nombre = nombre
        self.edad = edad

    def saludar(self):
        print(f"Hola, mi nombre es {self.nombre} y tengo {self.edad} años.")

# Creación de un objeto
persona1 = Persona("Ana", 30)
persona1.saludar()  # Salida: Hola, mi nombre es Ana y tengo 30 años.
```

Estos conceptos son fundamentales en el desarrollo de software y ayudan a construir aplicaciones robustas y mantenibles.