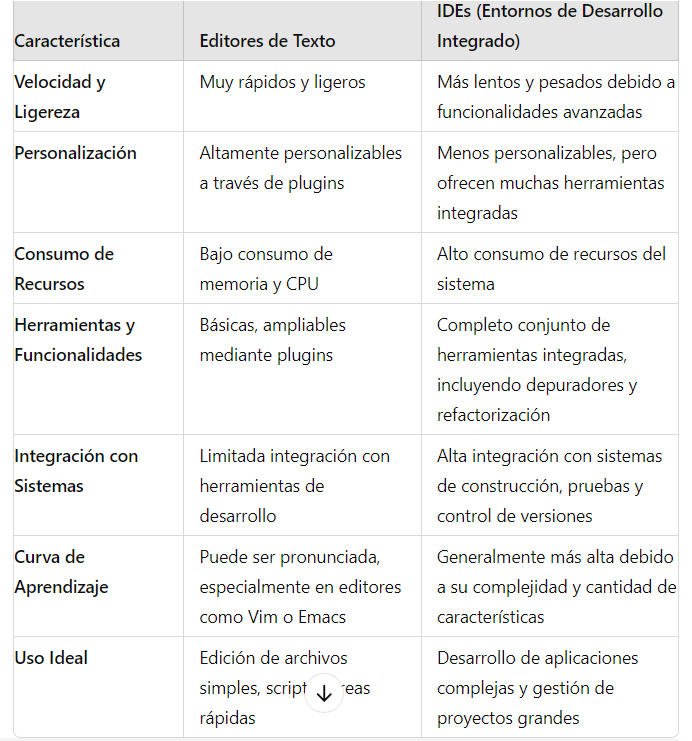
* **Texto**
* **Definición**: Un editor de texto es un programa de software utilizado principalmente para crear, editar y visualizar archivos de texto sin formato. Está diseñado para manipular contenido en formato texto sin aplicar ningún formato adicional, como negrita, cursiva o cambios de color.
* **Entorno de Desarrollo Integrado (IDE)**
* **Definición**: Un Entorno de Desarrollo Integrado (IDE) es un software que proporciona un conjunto de herramientas completas para el desarrollo de software. Incluye un editor de código, un compilador o intérprete, un depurador, y herramientas de construcción, todo dentro de una sola interfaz.

****

* **Editores de Texto**: Visual Studio Code, Sublime Text, Notepad++, Vim, Emacs
* **IDEs**: Visual Studio, IntelliJ IDEA, PyCharm, Eclipse, Android Studio

2 **Lenguajes con Tipado Estático:**

* **Definición:** En estos lenguajes, el tipo de las variables debe ser conocido en tiempo de compilación. El programador especifica explícitamente el tipo de cada variable, y el compilador verifica los tipos en tiempo de compilación.

**2. Lenguajes con Tipado Dinámico:**

* **Definición:** En estos lenguajes, los tipos de las variables se determinan en tiempo de ejecución. No es necesario declarar el tipo de una variable antes de usarla.
* **Ventajas:**
  + **Mayor flexibilidad:** Permite una codificación más rápida y simple, sin necesidad de declarar tipos.
  + **Más amigable para prototipos y scripts:** Ideal para desarrollo rápido y exploratorio.
* **Desventajas:**
  + **Mayor riesgo de errores:** Los errores de tipo no se detectan hasta que el código se ejecuta.
  + **Rendimiento menor:** Puede ser menos eficiente, ya que el sistema debe comprobar los tipos en tiempo de ejecución.
* **Ejemplos:** Python, JavaScript, Ruby, PHP, Perl.

**Lenguajes No Tipados**

* **Definición:** En realidad, casi todos los lenguajes tienen algún tipo de sistema de tipos. Sin embargo, algunos lenguajes (como lenguajes de programación más antiguos o muy simples) pueden manejar datos sin imponer un sistema de tipos estricto o incluso sin imponer tipos en absoluto.
* **Ejemplos:** Ensamblador es un ejemplo extremo donde las instrucciones operan directamente sobre bytes de memoria sin verificar tipos. Otro ejemplo podría ser el lenguaje BCPL (un antecesor de C), que tenía una noción mínima de tipos.

**Diferencia entre Lenguajes Fuertemente Tipados y Débilmente Tipados**

* **Lenguajes Fuertemente Tipados:**
  + **Definición:** En estos lenguajes, las reglas de tipo son estrictas, y el lenguaje impide o limita la conversión implícita entre tipos.
  + **Ejemplos:** Python, Java, Haskell.
* **Lenguajes Débilmente Tipados:**
  + **Definición:** En estos lenguajes, las reglas de tipo son más flexibles, permitiendo conversiones implícitas entre tipos de datos.
  + **Ejemplos:** JavaScript, PHP, C.

**Lenguajes Compilados**

**Definición:**

* En los lenguajes compilados, el código fuente se traduce directamente a código máquina por medio de un programa llamado **compilador**. Este proceso genera un archivo ejecutable que puede ser ejecutado directamente por el sistema operativo.

**Ejemplos:**

* **C, C++, Rust, Go, Fortran, Pascal**

**Lenguajes Interpretados**

**Definición:**

* En los lenguajes interpretados, el código fuente se ejecuta directamente mediante un programa llamado **intérprete**. Este programa lee el código fuente línea por línea y lo ejecuta sin generar un archivo binario intermedio.

**Características:**

* **Ejecución en tiempo real:** El intérprete traduce y ejecuta el código sobre la marcha.
* **Flexibilidad:** Es más fácil hacer cambios y probarlos rápidamente.
* **Disponibilidad del código fuente:** El código fuente debe estar disponible durante la ejecución.

**Ventajas:**

* **Desarrollo rápido:** Los desarrolladores pueden escribir y ejecutar código rápidamente, lo que es útil para prototipos y scripts.
* **Portabilidad:** El mismo código fuente puede ejecutarse en diferentes plataformas con un intérprete adecuado.
* **Facilidad de depuración:** Es más fácil encontrar y corregir errores porque el código se ejecuta línea por línea.

**Desventajas:**

* **Rendimiento inferior:** La necesidad de interpretar cada línea de código en tiempo de ejecución puede hacer que los programas sean más lentos.
* **Seguridad:** El código fuente está expuesto, lo que puede ser un riesgo de seguridad.
* **Dependencia del intérprete:** Se requiere un intérprete específico para ejecutar el código en cada plataforma.

**Ejemplos:**

* **Python, JavaScript, Ruby, PHP, Perl**

**Lenguajes Híbridos (Compilación e Interpretación)**

**Definición:**

* Algunos lenguajes combinan elementos de compilación e interpretación para aprovechar las ventajas de ambos métodos. El código fuente se compila primero en un formato intermedio (como bytecode), que luego es ejecutado por una máquina virtual o intérprete.

**Ventajas:**

* **Equilibrio entre rendimiento y flexibilidad:** Ofrece un mejor rendimiento que los lenguajes puramente interpretados y más flexibilidad que los lenguajes puramente compilados.
* **Portabilidad:** El bytecode es independiente de la plataforma, lo que permite que el mismo código se ejecute en diferentes sistemas operativos.

**Desventajas:**

* **Complejidad:** El proceso de compilación e interpretación puede añadir complejidad al sistema.
* **Rendimiento intermedio:** Aunque mejor que la interpretación pura, el rendimiento puede ser inferior a la compilación directa a código máquina.

**Ejemplos:**

* **Java (compila a bytecode y ejecuta en la JVM), C# (compila a MSIL y ejecuta en .NET CLR), Python (a menudo se compila a bytecode antes de ser interpretado)**

3 Los niveles de lenguajes de programación se refieren a la abstracción que estos proporcionan en relación con la máquina física. Aquí están los principales niveles, desde el más bajo hasta el más alto:

**1. Lenguaje de Máquina**

* **Descripción**: Consiste en instrucciones en código binario (0s y 1s) que son ejecutadas directamente por la CPU.
* **Uso**: Muy específico de la arquitectura de hardware. Raramente utilizado por los programadores debido a su complejidad y propensión a errores.

**2. Lenguaje Ensamblador**

* **Descripción**: Una capa encima del lenguaje de máquina. Usa mnemonics para representar operaciones básicas (e.g., MOV, ADD).
* **Uso**: Proporciona más control y eficiencia que los lenguajes de alto nivel, pero es difícil de escribir y mantener.

**3. Lenguajes de Bajo Nivel**

* **Descripción**: A menudo, incluyen lenguajes como C y C++. Ofrecen control sobre la gestión de memoria y el hardware.
* **Uso**: Ideal para sistemas operativos, controladores y aplicaciones donde el rendimiento y el control son críticos.

**4. Lenguajes de Alto Nivel**

* **Descripción**: Abstracciones más altas que ocultan muchos detalles del hardware. Ejemplos incluyen Python, Java, y C#.
* **Uso**: Desarrollo de aplicaciones de escritorio, web y móviles. Más fácil de aprender y usar debido a su sintaxis simple y bibliotecas extensas.

**5. Lenguajes de Muy Alto Nivel**

* **Descripción**: Incluyen lenguajes de dominio específico (DSLs) y lenguajes de scripting. Enfocados en tareas específicas.
* **Uso**: Ejemplos incluyen SQL para bases de datos, HTML/CSS para diseño web, y MATLAB para cálculos matemáticos.

**6. Lenguajes de Cuarta Generación (4GL)**

* **Descripción**: Diseñados para ser más naturales y accesibles para usuarios no técnicos. Se centran en simplificar la programación mediante la reducción de líneas de código.
* **Uso**: Aplicaciones de bases de datos, informes y generación de interfaces de usuario. Ejemplos: SQL (para algunas operaciones), Oracle Reports, etc.

**7. Lenguajes de Quinta Generación (5GL)**

* **Descripción**: Se centran en resolver problemas mediante la utilización de restricciones y lógica, sin necesidad de especificar cómo resolverlo.
* **Uso**: Inteligencia artificial y aprendizaje automático. Ejemplos: Prolog, algunos entornos de desarrollo de IA.