**Traducción**:

* **Compiladores**: **Traducción completa** antes de la ejecución a código máquina.
  + **Ejemplo**: Un programa en C es compilado usando gcc, generando un archivo ejecutable (a.out).
* **Intérpretes**: **Traducción línea por línea** durante la ejecución.
  + **Ejemplo**: Un script en Python es ejecutado usando python script.py, traduciendo y ejecutando el código en tiempo real.

**Tiempo de Ejecución**:

* **Compiladores**: **Más rápido** en tiempo de ejecución (ejecuta código maquina).
  + **Ejemplo**: Un juego compilado en C++ ejecuta rápidamente porque ya está en código máquina.
* **Intérpretes**: **Más lento** en tiempo de ejecución (interpreta código a máquina en tiempo real).
  + **Ejemplo**: Un programa en Ruby puede ser más lento porque se traduce y ejecuta línea por línea.

**Errores**:

* **Compiladores**: Detecta **errores en compilación** (antes de ejecutar).
  + **Ejemplo**: El compilador gcc detecta errores antes de ejecutarse al momento de intentar compilar.
* **Intérpretes**: Detecta **errores en ejecución** (mientras se ejecuta).
  + **Ejemplo**: En PHP, un error de sintaxis en una línea se detecta solo cuando esa línea es ejecutada.

**Memoria**:

* **Compiladores**: **Uso eficiente** de memoria.
  + **Ejemplo**: Un programa en C optimizado usa menos memoria durante la ejecución porque el código ya está en un formato optimizado.
* **Intérpretes**: **Mayor uso** de memoria (código y entorno en memoria).
  + **Ejemplo**: Un script en JavaScript puede usar más memoria porque el motor V8 de Node.js mantiene el código y el entorno de ejecución en memoria.

**Compatibilidad**:

* **Compiladores**: **Específico por plataforma**, necesita compilación para cada plataforma.
  + **Ejemplo**: Un programa en C++ compilado en Windows necesita recompilarse para ejecutarse en Linux.
* **Intérpretes**: **Altamente portátil**, ejecuta en cualquier sistema con el intérprete adecuado.
  + **Ejemplo**: Un script en Python puede ejecutarse en Windows, Mac y Linux sin cambios si el intérprete Python está instalado.

**Portabilidad**:

* **Compiladores**: Solo necesita ser compilado 1 vez y su archivo puede .exe o .out puede ejecutarse sin necesidad de otros archivos o programas.
  + **Ejemplo**: Un programa en C++ compilado en Windows puede ejecutarse en cualquier otro dispositivo Windows sin necesidad de tener gcc.
* **Intérpretes**: Necesitan siempre de su programa de interprete para ejecutar, lo que no permite utilizar el código fuente sin el intérprete del lenguaje correspondiente.
  + **Ejemplo**: Un script en Python necesita el intérprete de Python para ser ejecutado, por lo que el código fuente por sí mismo no es suficiente para ejecutar o utilizar el programa.

**Almacenamiento**:

* **Compiladores**: Los lenguajes compilados necesitan crear los archivos en código maquina para ejecutarse lo cual genera más consumo de almacenamiento durante la creación del código.
  + **Ejemplo**: Un programa en C++ compilado va a crear archivos .exe o .out cada vez que compila y que el programador necesite ejecutar el código fuente, lo cual generará un archivo extra.
* **Intérpretes**: Los intérpretes no crean un archivo extra para poder ejecutar el código.
  + **Ejemplo**: Un script en Python al ejecutarse no generará un .exe y solo necesitará que el interprete esté instalado.

**Depuración**:

* **Compiladores**: Los errores de tipo compilación son más difíciles de entender al momento de depurar
  + **Ejemplo**: Un programa en C++ compila el código por completo y puede llegar a dar un error de compilación no entendible.
* **Intérpretes**: Los intérpretes no compilan el código y solo se necesita depurar los errores de lógica.
  + **Ejemplo**: Un script en Python al no tener un correcto funcionamiento se puede depurar y ver el manejo de variables y resultados en tiempo de ejecución, no tendrá errores de compilación.