what's the difference between compiled and interpreted languages and in this way what about Csharp?

Difference Between Compiled and Interpreted Languages

Feature	Compiled Language	Interpreted Language
Execution	Entire code is converted to machine code **before** running	Code is translated **line-by-line during** execution
Speed	Generally faster	Generally slower (due to line-by-line execution)
Output	Produces an executable file (e.g., `.exe`)	Doesn't produce a standalone binary
Error Detection	Errors found **at compile-time**	Errors found **during runtime**
Examples	C, C++, Go	Python, JavaScript, PHP

What About C#?

C# is a **hybrid**:
Compiled *and* **Interpreted (Just-In-Time compiled)**

How It Works:

1. C# code is **compiled** into **Intermediate Language (IL)** → not machine code yet.
2. At runtime, the **.NET CLR** uses a **Just-In-Time (JIT) compiler** to convert IL into machine code.

Compare between implicit, explicit, Convert and parse casting

Comparison: Implicit, Explicit, `Convert`, and `Parse` Casting in C#

Feature	Implicit Casting	Explicit Casting	Convert Class	Parse Method
Definition	Automatic type conversion	Manual type conversion	Converts between types using methods	Converts string to a value type
Risk of Data Loss	No	Yes (possible)	May throw exception	May throw exception
Syntax	`int x = 10; double d = x;`	`double d = 9.7; int x = (int)d;`	`int x = Convert.ToInt32("123");`	`int x = int.Parse("123");`
Use Case	Safe widening conversions	Narrowing conversions	Converts various types to others	Specifically used for strings
Handles null?	Not applicable	Not applicable	Yes (`Convert.ToInt32(null)` → 0)	No (`Parse(null)` → Exception)
Performance	Fast	Fast	⚠ Slightly slower (more overhead)	Fast for string to type
Null-safe?	Not applicable	Not applicable	Yes	No

- implicit/`explicit` for **numeric types**
- Convert for **safe general conversions**
- Parse for **converting strings to value types** (like `int.Parse("42")`)