C# Type Conversion Comparison

Question 1: What's the difference between compiled and interpreted languages? How does C# fit in?

- Compiled languages: Translated to machine code before running (e.g., C, C++). Fast, errors caught at compile time.
- Interpreted languages: Executed line by line by an interpreter (e.g., Python, JavaScript). Slower, errors show at runtime.
- C#: Hybrid. Compiled to Intermediate Language (IL) first, then Just-In-Time (JIT) compiled to machine code at runtime.

Question 2: Compare between implicit, explicit, Convert and Parse casting

1 Implicit Casting (Type Conversion)

- Automatic conversion by C# when no data will be lost.
- Happens without using any keyword.
- Works with small → large types (int → long, float → double)
- Example:

```
int num = 100;
double bigNum = num; // implicit
```

- Key point: Safe conversion, compiler handles it.

---

2 Explicit Casting

- Manual conversion using a cast operator `(type)`.
- Needed when there might be data loss.
- Example:

```
double bigNum = 9.78;
int num = (int)bigNum; // explicit
Console.WriteLine(num); // 9 (fraction lost)
```

- Key point: Programmer takes responsibility; might lose data.

---

3 Convert Class

- `Convert` is a .NET helper class with methods like `Convert.ToInt32()`, `Convert.ToDouble()`, `Convert.ToString()`.
- Can convert between different types, sometimes safely handling nulls.
- Example:

```
string s = "123";
int num = Convert.ToInt32(s);
```

- Key points:
    - Throws an exception if conversion fails (e.g., string “abc”).
    - More flexible than explicit casting because it works with strings, bool, etc.

---

4 Parse Method

- `Parse` is a method of a specific type (like `int.Parse()`, `double.Parse()`) to convert from string → numeric type.
- Example:

```
string s = "456";
int num = int.Parse(s);
```

- Key points:
    - Only works when converting strings to numbers or booleans.
    - Throws an exception if string format is wrong.
    - You can also use `TryParse` to avoid exceptions.

✔ Quick Comparison Table

Type	How It Works	Need Cast?	Works With	Risk of Error
Implicit	Automatic, no data loss	No	Small → Large types	None
Explicit	Manual cast `(type)`	Yes	Any compatible types	Possible data loss
Convert Class	Methods like `Convert.ToInt32()`	No cast	Many types, strings	Throws exception if invalid
Parse Method	`int.Parse(string)`	No cast	Strings → numeric/boolean	Throws exception if format wrong