1.extension methods
extension methods are a way to add new functionality to an existing class without having to modify its source code or inherit from it. Extension methods allow you to "extend" a type's behavior by defining static methods that appear to be instance methods of the type.

2.generics
Generics are a language feature that allows you to define reusable code that can work with a variety of data types. The idea behind generics is to create a class, method, or interface that can work with any data type, rather than just a single type.
With generics, you can create classes and methods that work with collections of objects, such as lists, stacks, and dictionaries, and ensure type safety at compile-time, while still maintaining flexibility.

3.exception handling
You can use the try, catch, finally, and throw keywords to handle exceptions.

4.try-catch
The try block contains the code that may throw an exception, while the catch block is used to handle the exception and provide a fallback plan in case of an error.

5.Finally
The "finally" block is optional and contains code that will always run, whether or not an exception was thrown. It is commonly used for tasks such as closing open files or releasing resources.

6.throw
the throw keyword is used to explicitly throw an exception. An exception is a runtime error or unexpected behavior that occurs during the execution of a program.
When you use the throw keyword, you are essentially saying that an exceptional condition has occurred and you want to handle it in some way. You can throw exceptions of a built-in type such as ArgumentException or you can define your own custom exception type.
The throw keyword is typically used in combination with a try-catch block. The try block contains the code that might throw an exception, while the catch block contains the code that handles the exception. When an exception is thrown in the try block, the runtime looks for an appropriate catch block to handle it.

7.Delegates
Delegates in C# are objects that hold references to methods. They allow you to pass methods as parameters, store them in collections, and invoke them dynamically at runtime. A delegate is essentially a type-safe function pointer that can be used to encapsulate a method and its parameters.