**OOPS - OBJECT ORIENTED PROGRAMMING SYSTEMS**

**TANGIBLE AND INTANGIBLE**

**c# - BASED ON OOPS**

**MAJOR-PILLARS**

**INHERITANCE  
POLYMORPHISM**

**ASBTRACTION**

**ENCAPSULATION**

**MINOR-PILLARS**

**Delegates**

**ENUM**

**1. Microsoft .NET Fundamentals:**

**What is Microsoft .NET?**

* .NET is a free, open-source, and cross-platform development platform from Microsoft that simplifies building various applications (web, mobile, desktop, cloud, games, IoT).
* It provides a comprehensive set of tools, libraries, and languages for developers.

**.NET Framework**

* A mature, Windows-specific development framework focused on Windows desktop applications and web services.
* Offers a large class library for common functionalities.
* **Example:** (.NET Framework console application)

C#

using System;

namespace Hello  
{

class HelloWorld

{

static void Main(string[] args)

{

Console.WriteLine("Hello, World!");

}

}

}

Use code [with caution.](/faq#coding)

**.NET Core**

* A modern, open-source, and cross-platform framework built on .NET, supporting Windows, macOS, Linux, and more.
* Ideal for creating cloud-native applications, microservices, and modern web APIs.
* Leaner and faster than .NET Framework.
* **Example:** (.NET Core console application)

C#

using System;

public class HelloWorld

{

public static void Main(string[] args)

{

Console.WriteLine("Hello, World!");

}

}

Use code [with caution.](/faq#coding)

**Key Differences:**

|  |  |  |
| --- | --- | --- |
| **Feature** | **.NET Framework** | **.NET Core** |
| Platform | Windows-specific | Cross-platform (Windows, macOS, Linux) |
| Focus | Windows desktop and web services | Cloud-native applications, microservices, modern web APIs |
| Size | Larger footprint | Lighter weight |
| Open Source | No | Yes |
| Recommended | Existing projects, maintaining compatibility | New projects for flexibility and performance |

Export to Sheets

**Common Language Runtime (CLR)**

* The heart of .NET, providing a managed execution environment for applications.
* Handles memory management, thread management, security, and other essential services.

**Metadata**

* Information about types, methods, and other constructs embedded in compiled code.
* Used by the CLR to understand and execute the application.

**Common Type System (CTS)**

* Defines a set of rules for data types and their interactions.
* Enables different .NET languages to work seamlessly with each other.

**Framework Class Library (FCL)**

* A vast collection of pre-written, reusable code for common tasks.
* Saves developers time and effort by providing ready-made functionalities.

**Language Interoperability**

* .NET allows developers to use various languages (C#, VB.NET, F#, etc.) within a single project.

**2. C# First Program**

**First .NET Application using Visual Studio 2022**

1. Open Visual Studio.
2. Create a new Console App (.NET Framework) or Console App (.NET) project.
3. Write the code in the Main method (as shown in the previous examples).
4. Run the application (Ctrl+F5).

**Command Line Arguments**

* Arguments passed to the program when executed from the command line.
* Accessible in the string[] args parameter of the Main method.
* **Example:**

C#

using System;

class ArgsDemo

{

static void Main(string[] args)

{

if (args.Length > 0)

{

Console.WriteLine("Hello, " + args[0] + "!");

}

else

{

Console.WriteLine("Hello, World!");

}

}

}

Use code [with caution.](/faq#coding)

**Return Value of Main**

* The Main method can return an integer value (usually 0 for success).
* Used to communicate program exit status to the operating system (used in specific scenarios).

**Using Command Line Compiler (csc.exe)**

1. Open command prompt.
2. Navigate to the directory containing the C# source file (.cs).
3. Compile using csc.exe (e.g., csc.exe HelloWorld.cs). This creates an executable file.
4. Run the executable file (e

**Creating .NET Core Applications using VSCode**

**Prerequisites:**

* **.NET SDK:** Ensure you have the .NET SDK installed. Download it from <https://dotnet.microsoft.com/download>
* **Visual Studio Code:** Install Visual Studio Code from <https://code.visualstudio.com/>

**Steps:**

1. **Open a Terminal:** In VS Code, open a new terminal (Ctrl+Shift+`).
2. **Create a new project:** Use the dotnet new command to create a new project. For example, to create a console application:

Bash

dotnet new console

Use code [with caution.](/faq#coding)

1. **Open the project in VS Code:** Use the code . command in the terminal to open the project in VS Code.
2. **Run the application:** Use the dotnet run command in the terminal to run the application.

**Example:**

C#

// Program.cs

using System;

namespace MyFirstApp

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Hello, World!");

}

}

}

Use code [with caution.](/faq#coding)

**C# Language Syntax**

**C# Introduction and Evolution**

* C# is a modern, object-oriented programming language developed by Microsoft.
* It evolved from C++ and Java, incorporating their strengths while addressing their shortcomings.
* It is strongly typed, type-safe, and supports garbage collection.

**Classes and Structures**

* **Classes:** Blueprints for creating objects. They define properties, methods, and events.
* **Structures:** Value types that hold data. They are similar to classes but have limitations.

C#

// Class example

public class Person

{

public string Name { get; set; }

public int Age { get; set; }

}

// Structure example

public struct Point

{

public int X;

public int Y;

}

Use code [with caution.](/faq#coding)

**Data Types**

* C# supports various data types:
  + **Value types:** int, double, bool, char, etc.
  + **Reference types:** string, arrays, classes, interfaces, etc.

C#

int age = 30;

double price = 9.99;

bool isTrue = true;

char letter = 'A';

string name = "John Doe";

Use code [with caution.](/faq#coding)

**Value Types and Reference Types**

* **Value types:** Store data directly in their variables.
* **Reference types:** Store references to objects in memory.

C#

int num1 = 10;

int num2 = num1; // Creates a copy of num1

Person person1 = new Person { Name = "Alice" };

Person person2 = person1; // Both variables refer to the same object

Use code [with caution.](/faq#coding)

**Implicit and Explicit Casting**

* **Implicit casting:** Converting a smaller data type to a larger one automatically.
* **Explicit casting:** Converting a larger data type to a smaller one manually using the cast operator.

C#

int number = 10;

double result = number; // Implicit casting

double decimalValue = 3.14;

int intValue = (int)decimalValue; // Explicit casting

Use code [with caution.](/faq#coding)

**Programming Constructs**

* C# supports various control flow statements:
  + **Conditional statements:** if, else, switch
  + **Loops:** for, while, do-while
  + **Jump statements:** break, continue, return

C#

if (age >= 18)

{

Console.WriteLine("You are an adult.");

}

else

{

Console.WriteLine("You are a minor.");

}

for (int i = 0; i < 5; i++)

{

Console.WriteLine(i);

}

Use code [with caution.](/faq#coding)

**Additional Topics:**

* **Arrays:** Collections of elements of the same data type.
* **Methods:** Blocks of code that perform specific tasks.
* **Object-Oriented Programming (OOP) concepts:** Inheritance, polymorphism, encapsulation.
* **Error handling:** try-catch blocks.
* **Generics:** Create reusable code components.

**Understanding the Basics**

In C#, data types are classified into two primary categories:

**1. Value Types**

* Directly hold their data within their memory allocation.
* When assigned or passed as arguments, a copy of the data is created.
* Changes to the copy do not affect the original value.

**Examples:**

* int, float, double, decimal, bool, char, struct, enum

**2. Reference Types**

* Refer to objects stored in the managed heap.
* When assigned or passed as arguments, a copy of the reference is created, pointing to the same object.
* Changes made through one reference affect the object accessible by all references.

**Examples:**

* string, class, interface, array, delegate

**The Hierarchical Structure**

While there's no strict hierarchical inheritance between reference and value types in the traditional sense, we can visualize their relationship in terms of the System.Object class.

* **All reference types implicitly inherit from System.Object**. This provides a common base class for various reference types, offering shared methods like ToString(), Equals(), and GetHashCode().
* **Value types do not directly inherit from System.Object**. However, they can be implicitly converted (boxed) to System.Object. This boxing operation creates a reference to a newly allocated object on the heap that holds the value type's data.

**Visual Representation**

System.Object

|

+-- string

+-- class types

+-- interface types

+-- array types

+-- delegate types

|

+-- (boxed value types)

|

+-- int

+-- float

+-- ...

**Key Differences and Implications**

|  |  |  |
| --- | --- | --- |
| **Feature** | **Value Type** | **Reference Type** |
| Storage | Stack | Heap |
| Assignment | Copy of data | Copy of reference |
| Null | Not allowed | Allowed |
| Inheritance | No | Inherits from System.Object |
| Boxing/Unboxing | Can be boxed to object | No need for boxing |

Export to Sheets

**Additional Considerations**

* **Structs:** While value types, they can define properties and methods, offering some object-like features. However, they are still fundamentally value types.
* **Nullable Value Types:** These can hold either a valu

**Understanding Command Line Arguments**

Command-line arguments are values passed to an executable program when it's run from the command line. In C#, these arguments are accessible through the string[] args parameter in the Main method.

**Accessing Command Line Arguments**

Here's a basic example of how to access and print command-line arguments:

C#

using System;

namespace CommandLineArgsExample

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Number of arguments: " + args.Length);

if (args.Length > 0)

{

Console.WriteLine("Arguments:");

foreach (string arg in args)

{

Console.WriteLine(arg);

}

}

}

}

}

Use code [with caution.](/faq#coding)

**Passing Arguments from the Command Line**

To pass arguments when running the program, you would typically use the following syntax:

Bash

MyProgram.exe argument1 argument2 argument3

Use code [with caution.](/faq#coding)

**Important Points to Remember:**

* The args array is a string array, so you might need to convert arguments to other data types if necessary (e.g., using int.Parse, double.Parse, etc.).
* The first argument (index 0) is often the program name itself, but this behavior might vary across different operating systems.
* For more complex argument parsing, consider using dedicated command-line argument parsing libraries like CommandLineParser or FluentCommandLineParser.

**Example with Argument Parsing and Usage**

C#

using System;

namespace CommandLineArgsExample

{

class Program

{

static void Main(string[] args)

{

if (args.Length != 2)

{

Console.WriteLine("Usage: MyProgram.exe <number1> <number2>");

return;

}

int num1, num2;

if (!int.TryParse(args[0], out num1) || !int.TryParse(args[1], out num2))

{

Console.WriteLine("Invalid input. Please enter two numbers.");

return;

}

int sum = num1 + num2;

Console.WriteLine("Sum: " + sum);

}

}

}

Use code [with caution.](/faq#coding)

**Additional Considerations**

* **Error Handling:** Always validate user input to prevent unexpected behavior.
* **Argument Parsing Libraries:** For complex scenarios, explore dedicated libraries that offer features like argument validation, default values, and help generation.
* **Security:** Be cautious when handling user-provided input to avoid security vulnerabilities like injection attacks.

By understanding these concepts, you can effectively use command-line arguments to enhance the flexibility and usability of your C# applications

Mod>C:\Windows\Microsoft.NET\Framework\v4.0.30319\csc demo.cs

**Using VS Code for Building .NET Core Applications**

**Prerequisites**

Before we dive into creating .NET Core applications using VS Code, ensure you have the following installed:

* **.NET Core SDK:** Download and install the appropriate version from <https://dotnet.microsoft.com/download>
* **Visual Studio Code:** Download and install from <https://code.visualstudio.com/>
* **C# extension:** Install the C# extension for VS Code from the Extensions marketplace.

**Creating a .NET Core Application**

**1. Open VS Code:** Launch VS Code on your system.

**2. Create a New Project Folder:** Create a new folder for your project. Open this folder in VS Code by going to **File** -> **Open Folder**.

**3. Initialize the Project:** Open a terminal in VS Code (Ctrl+Shift+`) and navigate to your project directory. Use the following command to initialize a new console application:

Bash

dotnet new console

Use code [with caution.](/faq#coding)

This command creates a basic project structure with necessary files.

**4. Open the Project in VS Code:** Once the project is created, open it in VS Code.

**Running the Application**

To run the application, use the following command in the terminal:

Bash

dotnet run

Use code [with caution.](/faq#coding)

This will compile and execute your application.

**Exploring the Project Structure**

A typical .NET Core project structure looks like this:

* **obj:** Contains intermediate build files.
* **bin:** Contains the compiled output.
* **Program.cs:** The main entry point of the application.
* **project.json:** (Older projects) Contains project metadata.
* **.csproj:** (Newer projects) Contains project metadata in XML format.

**Building the Application**

To build the application without running it, use the following command:

Bash

dotnet build

Use code [with caution.](/faq#coding)

This creates the output assembly in the bin folder.

**Debugging**

VS Code provides debugging capabilities for .NET Core applications.

* Set breakpoints in your code.
* Start the debugging session (F5).
* Step through the code, inspect variables, and debug your application.

**Example: A Simple Console Application**

C#

using System;

namespace MyFirstApp

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Hello, World!");

}

}

}

Use code [with caution.](/faq#coding)

**Additional Tips**

* **IntelliSense:** VS Code provides IntelliSense for C# code, making development easier.
* **Code Formatting:** Use the built-in code formatter to maintain consistent code style.
* **Extensions:** Explore additional extensions for .NET development, such as debugging tools, code analysis, and productivity enhancements.
* **NuGet Packages:** Manage external dependencies using the NuGet package manager.

**Creating Different Types of .NET Core Applications**

VS Code supports creating various types of .NET Core applications using different templates. For example:

* **Web API:** dotnet new webapi
* **ASP.NET Core MVC:** dotnet new mvc
* **Class Library:** dotnet new classlib
* **Blazor App:** dotnet new blazor

By following these steps and leveraging the featu

**.NET Framework Features**

The .NET Framework is a robust platform for building applications on Windows.

It offers a comprehensive set of features that streamline development, enhance performance, and improve security. Here's a breakdown of its key features:

[1. What is .NET Framework? A software development framework. - Dot Net](https://dotnet.microsoft.com/en-us/learn/dotnet/what-is-dotnet-framework" \t "_blank)

[[Source icon](https://dotnet.microsoft.com/en-us/learn/dotnet/what-is-dotnet-framework)](https://dotnet.microsoft.com/en-us/learn/dotnet/what-is-dotnet-framework" \t "_blank)

[dotnet.microsoft.com](https://dotnet.microsoft.com/en-us/learn/dotnet/what-is-dotnet-framework" \t "_blank)

[2. Overview of .NET Framework - Learn Microsoft](https://learn.microsoft.com/en-us/dotnet/framework/get-started/overview" \t "_blank)

[[Source icon](https://learn.microsoft.com/en-us/dotnet/framework/get-started/overview)](https://learn.microsoft.com/en-us/dotnet/framework/get-started/overview" \t "_blank)

[learn.microsoft.com](https://learn.microsoft.com/en-us/dotnet/framework/get-started/overview" \t "_blank)

**Core Components**

* **Common Language Runtime (CLR):** The execution engine that manages memory, thread management, exception handling, and security.

[1. Common Language Runtime (CLR) overview - .NET - Learn Microsoft](https://learn.microsoft.com/en-us/dotnet/standard/clr" \l ":~:text=Managed%20code%20benefits%20from%20features,security%2C%20versioning%20and%20deployment%20support%2C" \t "_blank)

[[Source icon](https://learn.microsoft.com/en-us/dotnet/standard/clr#:~:text=Managed%20code%20benefits%20from%20features,security%2C%20versioning%20and%20deployment%20support%2C)](https://learn.microsoft.com/en-us/dotnet/standard/clr" \l ":~:text=Managed%20code%20benefits%20from%20features,security%2C%20versioning%20and%20deployment%20support%2C" \t "_blank)

[learn.microsoft.com](https://learn.microsoft.com/en-us/dotnet/standard/clr" \l ":~:text=Managed%20code%20benefits%20from%20features,security%2C%20versioning%20and%20deployment%20support%2C" \t "_blank)

* **.NET Framework Class Library:** A vast collection of reusable classes and interfaces for common programming tasks.

[1. .NET class library overview - .NET | Microsoft Learn](https://learn.microsoft.com/en-us/dotnet/standard/class-library-overview" \t "_blank)

[[Source icon](https://learn.microsoft.com/en-us/dotnet/standard/class-library-overview)](https://learn.microsoft.com/en-us/dotnet/standard/class-library-overview" \t "_blank)

[learn.microsoft.com](https://learn.microsoft.com/en-us/dotnet/standard/class-library-overview" \t "_blank)

**Key Features**

* **Language Interoperability:** Supports multiple languages (C#, VB.NET, F#, etc.) to work seamlessly together.

[1. Introduction to .NET Framework - GeeksforGeeks](https://www.geeksforgeeks.org/introduction-to-net-framework/" \l ":~:text=Multi%2Dlanguage%20support%3A%20The%20.,which%20allows%20developers%20to%20choose" \t "_blank)

[[Source icon](https://www.geeksforgeeks.org/introduction-to-net-framework/#:~:text=Multi%2Dlanguage%20support%3A%20The%20.,which%20allows%20developers%20to%20choose)](https://www.geeksforgeeks.org/introduction-to-net-framework/" \l ":~:text=Multi%2Dlanguage%20support%3A%20The%20.,which%20allows%20developers%20to%20choose" \t "_blank)

[www.geeksforgeeks.org](https://www.geeksforgeeks.org/introduction-to-net-framework/" \l ":~:text=Multi%2Dlanguage%20support%3A%20The%20.,which%20allows%20developers%20to%20choose" \t "_blank)

* **Common Type System (CTS):** Defines a common language specification for data types, ensuring type safety and compatibility.

[1. Common Type System - .NET | Microsoft Learn](https://learn.microsoft.com/en-us/dotnet/standard/base-types/common-type-system" \l ":~:text=The%20common%20type%20system%20defines,part%20of%20the%20runtime's%20support" \t "_blank)

[[Source icon](https://learn.microsoft.com/en-us/dotnet/standard/base-types/common-type-system#:~:text=The%20common%20type%20system%20defines,part%20of%20the%20runtime's%20support)](https://learn.microsoft.com/en-us/dotnet/standard/base-types/common-type-system" \l ":~:text=The%20common%20type%20system%20defines,part%20of%20the%20runtime's%20support" \t "_blank)

[learn.microsoft.com](https://learn.microsoft.com/en-us/dotnet/standard/base-types/common-type-system" \l ":~:text=The%20common%20type%20system%20defines,part%20of%20the%20runtime's%20support" \t "_blank)

* **Automatic Garbage Collection:** Manages memory allocation and deallocation, reducing memory leaks.

[1. Fundamentals of garbage collection - .NET | Microsoft Learn](https://learn.microsoft.com/en-us/dotnet/standard/garbage-collection/fundamentals" \l ":~:text=Automatic%20memory%20management%20can%20eliminate,attempting%20to%20access%20freed%20memory" \t "_blank)

[[Source icon](https://learn.microsoft.com/en-us/dotnet/standard/garbage-collection/fundamentals#:~:text=Automatic%20memory%20management%20can%20eliminate,attempting%20to%20access%20freed%20memory)](https://learn.microsoft.com/en-us/dotnet/standard/garbage-collection/fundamentals" \l ":~:text=Automatic%20memory%20management%20can%20eliminate,attempting%20to%20access%20freed%20memory" \t "_blank)

[learn.microsoft.com](https://learn.microsoft.com/en-us/dotnet/standard/garbage-collection/fundamentals" \l ":~:text=Automatic%20memory%20management%20can%20eliminate,attempting%20to%20access%20freed%20memory" \t "_blank)

* **Exception Handling:** Provides a structured way to handle errors and exceptions.

[1. Structured Exception Handling - Win32 apps | Microsoft Learn](https://learn.microsoft.com/en-us/windows/win32/debug/structured-exception-handling" \l ":~:text=Structured%20exception%20handling%20enables%20you,and%20is%20usable%20across%20all" \t "_blank)

[[Source icon](https://learn.microsoft.com/en-us/windows/win32/debug/structured-exception-handling#:~:text=Structured%20exception%20handling%20enables%20you,and%20is%20usable%20across%20all)](https://learn.microsoft.com/en-us/windows/win32/debug/structured-exception-handling" \l ":~:text=Structured%20exception%20handling%20enables%20you,and%20is%20usable%20across%20all" \t "_blank)

[learn.microsoft.com](https://learn.microsoft.com/en-us/windows/win32/debug/structured-exception-handling" \l ":~:text=Structured%20exception%20handling%20enables%20you,and%20is%20usable%20across%20all" \t "_blank)

* **Security:** Offers robust security features like code access security, role-based security, and cryptography.

[1. Code Access Security - ADO.NET - Learn Microsoft](https://learn.microsoft.com/en-us/dotnet/framework/data/adonet/code-access-security" \l ":~:text=NET%20Framework%20offers%20role%2Dbased,common%20infrastructure%20supplied%20by%20the" \t "_blank)

[[Source icon](https://learn.microsoft.com/en-us/dotnet/framework/data/adonet/code-access-security#:~:text=NET%20Framework%20offers%20role%2Dbased,common%20infrastructure%20supplied%20by%20the)](https://learn.microsoft.com/en-us/dotnet/framework/data/adonet/code-access-security" \l ":~:text=NET%20Framework%20offers%20role%2Dbased,common%20infrastructure%20supplied%20by%20the" \t "_blank)

[learn.microsoft.com](https://learn.microsoft.com/en-us/dotnet/framework/data/adonet/code-access-security" \l ":~:text=NET%20Framework%20offers%20role%2Dbased,common%20infrastructure%20supplied%20by%20the" \t "_blank)

* **Deployment:** Simplifies deployment with features like side-by-side versioning and ClickOnce deployment.

[1. Deploying the .NET Framework and Applications - Learn Microsoft](https://learn.microsoft.com/en-us/dotnet/framework/deployment/" \t "_blank)

[[Source icon](https://learn.microsoft.com/en-us/dotnet/framework/deployment/)](https://learn.microsoft.com/en-us/dotnet/framework/deployment/" \t "_blank)

[learn.microsoft.com](https://learn.microsoft.com/en-us/dotnet/framework/deployment/" \t "_blank)

* **Windows Forms:** A rich set of controls for building Windows desktop applications.

[1. Desktop Guide (Windows Forms .NET) - Learn Microsoft](https://learn.microsoft.com/en-us/dotnet/desktop/winforms/overview/?view=netdesktop-8.0" \l ":~:text=The%20Windows%20Forms%20development%20platform,data%20binding%2C%20and%20user%20input." \t "_blank)

[[Source icon](https://learn.microsoft.com/en-us/dotnet/desktop/winforms/overview/?view=netdesktop-8.0#:~:text=The%20Windows%20Forms%20development%20platform,data%20binding%2C%20and%20user%20input.)](https://learn.microsoft.com/en-us/dotnet/desktop/winforms/overview/?view=netdesktop-8.0" \l ":~:text=The%20Windows%20Forms%20development%20platform,data%20binding%2C%20and%20user%20input." \t "_blank)

[learn.microsoft.com](https://learn.microsoft.com/en-us/dotnet/desktop/winforms/overview/?view=netdesktop-8.0" \l ":~:text=The%20Windows%20Forms%20development%20platform,data%20binding%2C%20and%20user%20input." \t "_blank)

* **ASP.NET:** A framework for building web applications and services.

[1. What is ASP.NET?](https://dotnet.microsoft.com/en-us/learn/aspnet/what-is-aspnet" \l ":~:text=ASP.NET%20is%20an%20open,NET." \t "_blank)

[[Source icon](https://dotnet.microsoft.com/en-us/learn/aspnet/what-is-aspnet#:~:text=ASP.NET%20is%20an%20open,NET.)](https://dotnet.microsoft.com/en-us/learn/aspnet/what-is-aspnet" \l ":~:text=ASP.NET%20is%20an%20open,NET." \t "_blank)

[dotnet.microsoft.com](https://dotnet.microsoft.com/en-us/learn/aspnet/what-is-aspnet" \l ":~:text=ASP.NET%20is%20an%20open,NET." \t "_blank)

* **ADO.NET:** Provides data access services for relational and non-relational data sources.

[1. ADO.NET | Microsoft Learn](https://learn.microsoft.com/en-us/dotnet/framework/data/adonet/" \t "_blank)

[[Source icon](https://learn.microsoft.com/en-us/dotnet/framework/data/adonet/)](https://learn.microsoft.com/en-us/dotnet/framework/data/adonet/" \t "_blank)

[learn.microsoft.com](https://learn.microsoft.com/en-us/dotnet/framework/data/adonet/" \t "_blank)

* **WCF:** For building service-oriented applications.

[1. What is .NET Framework? .Net Programming Explained - AltexSoft](https://www.altexsoft.com/blog/the-good-and-the-bad-of-net-framework-programming/" \t "_blank)

[[Source icon](https://www.altexsoft.com/blog/the-good-and-the-bad-of-net-framework-programming/)](https://www.altexsoft.com/blog/the-good-and-the-bad-of-net-framework-programming/" \t "_blank)

[www.altexsoft.com](https://www.altexsoft.com/blog/the-good-and-the-bad-of-net-framework-programming/" \t "_blank)

* **WPF:** A framework for building rich user interfaces.

[1. Hello World app with WPF in C# - Visual Studio (Windows) | Microsoft Learn](https://learn.microsoft.com/en-us/visualstudio/get-started/csharp/tutorial-wpf?view=vs-2022" \l ":~:text=What%20is%20WPF%3F,that%20creates%20desktop%20client%20applications." \t "_blank)

[[Source icon](https://learn.microsoft.com/en-us/visualstudio/get-started/csharp/tutorial-wpf?view=vs-2022#:~:text=What%20is%20WPF%3F,that%20creates%20desktop%20client%20applications.)](https://learn.microsoft.com/en-us/visualstudio/get-started/csharp/tutorial-wpf?view=vs-2022" \l ":~:text=What%20is%20WPF%3F,that%20creates%20desktop%20client%20applications." \t "_blank)

[learn.microsoft.com](https://learn.microsoft.com/en-us/visualstudio/get-started/csharp/tutorial-wpf?view=vs-2022" \l ":~:text=What%20is%20WPF%3F,that%20creates%20desktop%20client%20applications." \t "_blank)

* **LINQ:** Language Integrated Query for querying data sources in a declarative way.

[1. Language Integrated Query (LINQ) - C# | Microsoft Learn](https://learn.microsoft.com/en-us/dotnet/csharp/linq/" \l ":~:text=Language%2DIntegrated%20Query%20(LINQ),directly%20into%20the%20C%23%20language." \t "_blank)

[[Source icon](https://learn.microsoft.com/en-us/dotnet/csharp/linq/#:~:text=Language%2DIntegrated%20Query%20(LINQ),directly%20into%20the%20C%23%20language.)](https://learn.microsoft.com/en-us/dotnet/csharp/linq/" \l ":~:text=Language%2DIntegrated%20Query%20(LINQ),directly%20into%20the%20C%23%20language." \t "_blank)

[learn.microsoft.com](https://learn.microsoft.com/en-us/dotnet/csharp/linq/" \l ":~:text=Language%2DIntegrated%20Query%20(LINQ),directly%20into%20the%20C%23%20language." \t "_blank)

**Additional Benefits**

* **Improved Performance:** Features like Just-In-Time (JIT) compilation and native code optimization enhance performance.

[1. Understanding .NET Just-In-Time Compilation - Telerik.com](https://www.telerik.com/blogs/understanding-net-just-in-time-compilation" \l ":~:text=JIT%20can%20profile%20the%20application,(the%20most%20used%20functions)." \t "_blank)

[[Source icon](https://www.telerik.com/blogs/understanding-net-just-in-time-compilation#:~:text=JIT%20can%20profile%20the%20application,(the%20most%20used%20functions).)](https://www.telerik.com/blogs/understanding-net-just-in-time-compilation" \l ":~:text=JIT%20can%20profile%20the%20application,(the%20most%20used%20functions)." \t "_blank)

[www.telerik.com](https://www.telerik.com/blogs/understanding-net-just-in-time-compilation" \l ":~:text=JIT%20can%20profile%20the%20application,(the%20most%20used%20functions)." \t "_blank)

* **Simplified Development:** Rapid application development (RAD) capabilities and extensive class libraries accelerate development.

[1. Guide: Benefits and Features of .NET Framework - Prakash Software Solutions](https://prakashinfotech.com/guide-benefits-and-features-of-net-framework" \t "_blank)

[[Source icon](https://prakashinfotech.com/guide-benefits-and-features-of-net-framework)](https://prakashinfotech.com/guide-benefits-and-features-of-net-framework" \t "_blank)

[prakashinfotech.com](https://prakashinfotech.com/guide-benefits-and-features-of-net-framework" \t "_blank)

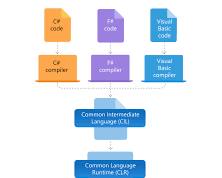
* **Enhanced Scalability:** Can handle large-scale applications and high transaction volumes.
* **Strong Community Support:** A large and active community provides resources, libraries, and support.

[1. Introduction to .NET Framework - GeeksforGeeks](https://www.geeksforgeeks.org/introduction-to-net-framework/" \l ":~:text=The%20.NET%20Framework%20also%20provides,seamlessly%20with%20other%20Microsoft%20products." \t "_blank)

[[Source icon](https://www.geeksforgeeks.org/introduction-to-net-framework/#:~:text=The%20.NET%20Framework%20also%20provides,seamlessly%20with%20other%20Microsoft%20products.)](https://www.geeksforgeeks.org/introduction-to-net-framework/" \l ":~:text=The%20.NET%20Framework%20also%20provides,seamlessly%20with%20other%20Microsoft%20products." \t "_blank)

[www.geeksforgeeks.org](https://www.geeksforgeeks.org/introduction-to-net-framework/" \l ":~:text=The%20.NET%20Framework%20also%20provides,seamlessly%20with%20other%20Microsoft%20products." \t "_blank)

**Visual Representation**

[**Opens in a new window [](https://dotnet.microsoft.com/en-us/learn/dotnet/what-is-dotnet-framework)dotnet.microsoft.com**](https://dotnet.microsoft.com/en-us/learn/dotnet/what-is-dotnet-framework)

**.NET Framework architecture with key components**

**Would you like to delve deeper into a specific feature or explore how these features contribute to building different types of applications?**

**.NET Core Framework Features**

.NET Core is a cross-platform, open-source framework developed by Microsoft for building modern cloud-native applications.

It offers a wide range of features that make it a popular choice for developers.

[1. NET | Build. Test. Deploy - Microsoft](https://dotnet.microsoft.com/en-us/" \l ":~:text=Build.-,Test.,apps%20and%20powerful%20cloud%20services." \t "_blank)

[[Source icon](https://dotnet.microsoft.com/en-us/#:~:text=Build.-,Test.,apps%20and%20powerful%20cloud%20services.)](https://dotnet.microsoft.com/en-us/" \l ":~:text=Build.-,Test.,apps%20and%20powerful%20cloud%20services." \t "_blank)

[dotnet.microsoft.com](https://dotnet.microsoft.com/en-us/" \l ":~:text=Build.-,Test.,apps%20and%20powerful%20cloud%20services." \t "_blank)

[2. ASP.NET Core | Open-source web framework for .NET](https://dotnet.microsoft.com/en-us/apps/aspnet" \t "_blank)

[[Source icon](https://dotnet.microsoft.com/en-us/apps/aspnet)](https://dotnet.microsoft.com/en-us/apps/aspnet" \t "_blank)

[dotnet.microsoft.com](https://dotnet.microsoft.com/en-us/apps/aspnet" \t "_blank)

**Core Features**

* **Cross-Platform Compatibility:** Run applications on Windows, macOS, and Linux.

[1. Choose between .NET and .NET Framework for server apps - Learn Microsoft](https://learn.microsoft.com/en-us/dotnet/standard/choosing-core-framework-server" \l ":~:text=Works%20cross%2Dplatform.,Windows%2C%20Linux%2C%20and%20macOS." \t "_blank)

[[Source icon](https://learn.microsoft.com/en-us/dotnet/standard/choosing-core-framework-server#:~:text=Works%20cross%2Dplatform.,Windows%2C%20Linux%2C%20and%20macOS.)](https://learn.microsoft.com/en-us/dotnet/standard/choosing-core-framework-server" \l ":~:text=Works%20cross%2Dplatform.,Windows%2C%20Linux%2C%20and%20macOS." \t "_blank)

[learn.microsoft.com](https://learn.microsoft.com/en-us/dotnet/standard/choosing-core-framework-server" \l ":~:text=Works%20cross%2Dplatform.,Windows%2C%20Linux%2C%20and%20macOS." \t "_blank)

* **Open-Source:** Active community contributions and flexibility.

[1. List of Top .NET Core Features You Need to Discover Right Now - Integrative Systems](https://www.integrativesystems.com/features-of-dot-net-core/" \t "_blank)

[[Source icon](https://www.integrativesystems.com/features-of-dot-net-core/)](https://www.integrativesystems.com/features-of-dot-net-core/" \t "_blank)

[www.integrativesystems.com](https://www.integrativesystems.com/features-of-dot-net-core/" \t "_blank)

* **High Performance:** Optimized for cloud-native applications with improved performance.

[1. Improving Cloud-Native Applications with .NET 8 - Influential Software](https://www.influentialsoftware.com/cloud-native-applications-net-8/" \t "_blank)

[[Source icon](https://www.influentialsoftware.com/cloud-native-applications-net-8/)](https://www.influentialsoftware.com/cloud-native-applications-net-8/" \t "_blank)

[www.influentialsoftware.com](https://www.influentialsoftware.com/cloud-native-applications-net-8/" \t "_blank)

* **Cloud-Ready:** Built-in support for cloud services and containerization.

[1. ASP.NET Core for Modern Web & Cloud Applications - Features To Know](https://www.stridelysolutions.com/insights/blog/asp-net-core-for-modern-web-cloud-applications-features-to-know/" \l ":~:text=Docker%20as%20a%20PaaS%2C%20Azure,NET%20Core%20integrates%20all." \t "_blank)

[[Source icon](https://www.stridelysolutions.com/insights/blog/asp-net-core-for-modern-web-cloud-applications-features-to-know/#:~:text=Docker%20as%20a%20PaaS%2C%20Azure,NET%20Core%20integrates%20all.)](https://www.stridelysolutions.com/insights/blog/asp-net-core-for-modern-web-cloud-applications-features-to-know/" \l ":~:text=Docker%20as%20a%20PaaS%2C%20Azure,NET%20Core%20integrates%20all." \t "_blank)

[www.stridelysolutions.com](https://www.stridelysolutions.com/insights/blog/asp-net-core-for-modern-web-cloud-applications-features-to-know/" \l ":~:text=Docker%20as%20a%20PaaS%2C%20Azure,NET%20Core%20integrates%20all." \t "_blank)

* **Modular Design:** Choose only the components you need, reducing application size.
* **ASP.NET Core:** A unified framework for building web UI and web APIs.

[1. Overview of ASP.NET Core | Microsoft Learn](https://learn.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-8.0" \t "_blank)

[[Source icon](https://learn.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-8.0)](https://learn.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-8.0" \t "_blank)

[learn.microsoft.com](https://learn.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-8.0" \t "_blank)

* **Entity Framework Core:** Object-relational mapper for data access.
* **Command-Line Interface (CLI):** Efficient development and deployment using the dotnet CLI.

**Advanced Features**

* **Blazor:** Build interactive web UIs using C# instead of JavaScript.

[1. Blazor | Build client web apps with C# | .NET](https://dotnet.microsoft.com/en-us/apps/aspnet/web-apps/blazor" \l ":~:text=Build%20beautiful%20web%20apps%20with,writing%20a%20line%20of%20JavaScript." \t "_blank)

[[Source icon](https://dotnet.microsoft.com/en-us/apps/aspnet/web-apps/blazor#:~:text=Build%20beautiful%20web%20apps%20with,writing%20a%20line%20of%20JavaScript.)](https://dotnet.microsoft.com/en-us/apps/aspnet/web-apps/blazor" \l ":~:text=Build%20beautiful%20web%20apps%20with,writing%20a%20line%20of%20JavaScript." \t "_blank)

[dotnet.microsoft.com](https://dotnet.microsoft.com/en-us/apps/aspnet/web-apps/blazor" \l ":~:text=Build%20beautiful%20web%20apps%20with,writing%20a%20line%20of%20JavaScript." \t "_blank)

* **gRPC:** High-performance RPC framework for building microservices.

[1. Efficient Microservices Communication: Role Of RPC Framework In A Cloud-Native Ecosystem | by Sudip Sengupta](https://sudip-says-hi.medium.com/efficient-microservices-communication-role-of-rpc-framework-in-a-cloud-native-ecosystem-9122cf665c2e" \l ":~:text=gRPC%20is%20a%20modern%20high,Protocol%20Buffers%20and%20HTTP%2F2." \t "_blank)

[[Source icon](https://sudip-says-hi.medium.com/efficient-microservices-communication-role-of-rpc-framework-in-a-cloud-native-ecosystem-9122cf665c2e#:~:text=gRPC%20is%20a%20modern%20high,Protocol%20Buffers%20and%20HTTP%2F2.)](https://sudip-says-hi.medium.com/efficient-microservices-communication-role-of-rpc-framework-in-a-cloud-native-ecosystem-9122cf665c2e" \l ":~:text=gRPC%20is%20a%20modern%20high,Protocol%20Buffers%20and%20HTTP%2F2." \t "_blank)

[sudip-says-hi.medium.com](https://sudip-says-hi.medium.com/efficient-microservices-communication-role-of-rpc-framework-in-a-cloud-native-ecosystem-9122cf665c2e" \l ":~:text=gRPC%20is%20a%20modern%20high,Protocol%20Buffers%20and%20HTTP%2F2." \t "_blank)

* **SignalR:** Real-time communication between server and clients.

[1. Real-time ASP.NET with SignalR](https://dotnet.microsoft.com/en-us/apps/aspnet/signalr" \l ":~:text=Bring%20your%20ASP.NET%20apps%20to%20life%20with%20SignalR&It's%20the%20ability%20to%20have,happens%2C%20in%20real%2Dtime." \t "_blank)

[[Source icon](https://dotnet.microsoft.com/en-us/apps/aspnet/signalr#:~:text=Bring%20your%20ASP.NET%20apps%20to%20life%20with%20SignalR&It's%20the%20ability%20to%20have,happens%2C%20in%20real%2Dtime.)](https://dotnet.microsoft.com/en-us/apps/aspnet/signalr" \l ":~:text=Bring%20your%20ASP.NET%20apps%20to%20life%20with%20SignalR&It's%20the%20ability%20to%20have,happens%2C%20in%20real%2Dtime." \t "_blank)

[dotnet.microsoft.com](https://dotnet.microsoft.com/en-us/apps/aspnet/signalr" \l ":~:text=Bring%20your%20ASP.NET%20apps%20to%20life%20with%20SignalR&It's%20the%20ability%20to%20have,happens%2C%20in%20real%2Dtime." \t "_blank)

* **Dependency Injection:** Manage dependencies effectively.

[1. Dependency injection - .NET | Microsoft Learn](https://learn.microsoft.com/en-us/dotnet/core/extensions/dependency-injection" \l ":~:text=NET%20supports%20the%20dependency%20injection,IoC)%20between%20classes%20and%20their" \t "_blank)

[[Source icon](https://learn.microsoft.com/en-us/dotnet/core/extensions/dependency-injection#:~:text=NET%20supports%20the%20dependency%20injection,IoC)%20between%20classes%20and%20their)](https://learn.microsoft.com/en-us/dotnet/core/extensions/dependency-injection" \l ":~:text=NET%20supports%20the%20dependency%20injection,IoC)%20between%20classes%20and%20their" \t "_blank)

[learn.microsoft.com](https://learn.microsoft.com/en-us/dotnet/core/extensions/dependency-injection" \l ":~:text=NET%20supports%20the%20dependency%20injection,IoC)%20between%20classes%20and%20their" \t "_blank)

* **Configuration:** Flexible configuration options for different environments.

[1. Using the ASP.NET Core Environment Feature to manage Development vs. Production for any config file type - Scott Hanselman](https://www.hanselman.com/blog/using-the-aspnet-core-environment-feature-to-manage-development-vs-production-for-any-config-file-type" \l ":~:text=Since%20the%20configuration%20features%20of,to%20our%20own%20config%20files." \t "_blank)

[[Source icon](https://www.hanselman.com/blog/using-the-aspnet-core-environment-feature-to-manage-development-vs-production-for-any-config-file-type#:~:text=Since%20the%20configuration%20features%20of,to%20our%20own%20config%20files.)](https://www.hanselman.com/blog/using-the-aspnet-core-environment-feature-to-manage-development-vs-production-for-any-config-file-type" \l ":~:text=Since%20the%20configuration%20features%20of,to%20our%20own%20config%20files." \t "_blank)

[www.hanselman.com](https://www.hanselman.com/blog/using-the-aspnet-core-environment-feature-to-manage-development-vs-production-for-any-config-file-type" \l ":~:text=Since%20the%20configuration%20features%20of,to%20our%20own%20config%20files." \t "_blank)

* **Logging:** Comprehensive logging framework for application insights.
* **Testing:** Built-in support for unit testing and integration testing.

[1. Integration tests in ASP.NET Core | Microsoft Learn](https://learn.microsoft.com/en-us/aspnet/core/test/integration-tests?view=aspnetcore-8.0" \l ":~:text=In%20this%20article&ASP.NET%20Core%20supports%20integration,an%20in%2Dmemory%20test%20server." \t "_blank)

[[Source icon](https://learn.microsoft.com/en-us/aspnet/core/test/integration-tests?view=aspnetcore-8.0#:~:text=In%20this%20article&ASP.NET%20Core%20supports%20integration,an%20in%2Dmemory%20test%20server.)](https://learn.microsoft.com/en-us/aspnet/core/test/integration-tests?view=aspnetcore-8.0" \l ":~:text=In%20this%20article&ASP.NET%20Core%20supports%20integration,an%20in%2Dmemory%20test%20server." \t "_blank)

[learn.microsoft.com](https://learn.microsoft.com/en-us/aspnet/core/test/integration-tests?view=aspnetcore-8.0" \l ":~:text=In%20this%20article&ASP.NET%20Core%20supports%20integration,an%20in%2Dmemory%20test%20server." \t "_blank)

* **Security:** Robust security features like authentication, authorization, and data protection.

[1. List of Top .NET Core Features You Need to Discover Right Now - Integrative Systems](https://www.integrativesystems.com/features-of-dot-net-core/" \t "_blank)

[[Source icon](https://www.integrativesystems.com/features-of-dot-net-core/)](https://www.integrativesystems.com/features-of-dot-net-core/" \t "_blank)

[www.integrativesystems.com](https://www.integrativesystems.com/features-of-dot-net-core/" \t "_blank)

**Benefits of Using .NET Core**

* **Faster development:** Increased productivity with modern tools and features.

[1. .NET 8: A Complete Overview and New Features - eLuminous Technologies](https://eluminoustechnologies.com/blog/net-8-features/" \t "_blank)

[[Source icon](https://eluminoustechnologies.com/blog/net-8-features/)](https://eluminoustechnologies.com/blog/net-8-features/" \t "_blank)

[eluminoustechnologies.com](https://eluminoustechnologies.com/blog/net-8-features/" \t "_blank)

* **Lower costs:** Reduced licensing costs and cloud optimization.
* **Improved performance:** High-performance applications for better user experience.

[1. Why .NET Core is the Best Choice for Modern Application Development – Its Key Features & Benefits - KANINI](https://kanini.com/blog/net-core-application-development/" \l ":~:text=It%20furnishes%20a%20high%2Dperformance,and%20web%20apps%20that%20precisely" \t "_blank)

[[Source icon](https://kanini.com/blog/net-core-application-development/#:~:text=It%20furnishes%20a%20high%2Dperformance,and%20web%20apps%20that%20precisely)](https://kanini.com/blog/net-core-application-development/" \l ":~:text=It%20furnishes%20a%20high%2Dperformance,and%20web%20apps%20that%20precisely" \t "_blank)

[kanini.com](https://kanini.com/blog/net-core-application-development/" \l ":~:text=It%20furnishes%20a%20high%2Dperformance,and%20web%20apps%20that%20precisely" \t "_blank)

* **Scalability:** Easily scale applications to meet growing demands.

[1. Why Choose .NET Frameworks for Scalable Applications Development - YourStory.com](https://yourstory.com/mystory/why-choose-net-frameworks-for-scalable-application-6vj3rq5tlz" \t "_blank)

[[Source icon](https://yourstory.com/mystory/why-choose-net-frameworks-for-scalable-application-6vj3rq5tlz)](https://yourstory.com/mystory/why-choose-net-frameworks-for-scalable-application-6vj3rq5tlz" \t "_blank)

[yourstory.com](https://yourstory.com/mystory/why-choose-net-frameworks-for-scalable-application-6vj3rq5tlz" \t "_blank)

* **Flexibility:** Choose the best tools and technologies for your project.

**Example Use Cases**

* **Web applications:** Build modern, scalable web applications using ASP.NET Core.

[1. ASP.NET Core | Open-source web framework for .NET](https://dotnet.microsoft.com/en-us/apps/aspnet" \t "_blank)

[[Source icon](https://dotnet.microsoft.com/en-us/apps/aspnet)](https://dotnet.microsoft.com/en-us/apps/aspnet" \t "_blank)

[dotnet.microsoft.com](https://dotnet.microsoft.com/en-us/apps/aspnet" \t "_blank)

* **Cloud services:** Create cloud-native microservices and serverless functions.
* **IoT applications:** Develop IoT solutions with real-time data processing.
* **Mobile backends:** Build backend services for mobile apps.
* **Desktop applications:** Create cross-platform desktop applications with .NET MAUI.

[1. NET Multi-platform App UI (.NET MAUI) - Dot Net - Microsoft](https://dotnet.microsoft.com/en-us/apps/maui" \t "_blank)

[[Source icon](https://dotnet.microsoft.com/en-us/apps/maui)](https://dotnet.microsoft.com/en-us/apps/maui" \t "_blank)

[dotnet.microsoft.com](https://dotnet.microsoft.com/en-us/apps/maui" \t "_blank)

**Would you like to delve deeper into a specific feature or use case?**

I can also provide code examples or comparisons with other frameworks if you'd like.

Sources and related content

Just-In-Time compiler is an advanced dynamic compilation mechanism that is widely used for .NET-based applications. In the .NET environment, it’s a main part of the common language runtime. Due to this, the software speed and performance get refined, and customers experience smooth navigation.

Mainly there are three types of JIT compilers: pre-JIT, normal JIT, and Econo JIT compilers. Among all three, Econo JIT is now not available. It’s read to understand its working and differentiate it from the other two JIT compilers. In addition, the pre-JIT compiler uses the MS intermediate language and compiles it in a single process. Also, it outputs a file that can only be run on one compatible platform.

On the other hand, normal JIT compiles only the called method and stores it in the cache memory. Using the type of compiler depends on the requirements, for which one should consult a .[NET development company](https://positiwise.com/technologies/dot-net-development).

**Programming Constructs in C#**

Programming constructs are the fundamental building blocks of any programming language. C# offers a rich set of constructs to control the flow of execution, manipulate data, and create complex algorithms.

**Control Flow Statements**

These constructs determine the order in which statements are executed.

* **Conditional Statements:**
  + if...else: Executes code based on a condition.
  + switch: Provides a multi-way branch based on the value of an expression.
* **Iterative Statements:**
  + for: Executes a block of code a specified number of times.
  + while: Executes a block of code repeatedly as long as a condition is true.
  + do...while: Similar to while, but the condition is checked after the first iteration.
  + foreach: Iterates over elements in a collection.
* **Jump Statements:**
  + break: Terminates a loop or switch statement.
  + continue: Skips the current iteration of a loop.
  + goto: Transfers control to a labeled statement (generally discouraged due to potential for code readability issues).

**Data Types and Variables**

C# supports a variety of data types to store different kinds of values.

* **Value Types:**
  + int, long, short, byte: Integer types.
  + float, double, decimal: Floating-point numbers.
  + char: Single character.
  + bool: Boolean value (true or false).
* **Reference Types:**
  + string: Sequence of characters.
  + object: Base class for all objects.
  + array: Collection of elements of the same type.
  + class: User-defined data type.
  + struct: Value type that encapsulates data members.
  + enum: Enumeration of named constants.

**Operators**

Operators are symbols that perform operations on operands.

* **Arithmetic operators:** +, -, \*, /, %.
* **Comparison operators:** ==, !=, <, >, <=, >=.
* **Logical operators:** &&, ||, !.
* **Assignment operators:** =, +=, -=, \*=, /=, %=.
* **Bitwise operators:** &, |, ^, ~, <<, >>.

**Other Constructs**

* **Methods:** Reusable blocks of code.
* **Classes and Objects:** Encapsulation of data and behavior.
* **Inheritance:** Creating new classes based on existing ones.
* **Polymorphism:** Ability of objects to take on multiple forms.
* **Interfaces:** Contracts defining methods that classes must implement.
* **Exceptions:** Handling runtime errors gracefully.
* **Generics:** Creating type-safe code reusable for different data types.
* **Lambda expressions:** Anonymous functions.
* **LINQ:** Language Integrated Query for querying data sources.