### ****Basics of C# and .NET****

C# (C-Sharp) is a modern, object-oriented programming language developed by Microsoft, primarily used with the .NET framework for building web applications, desktop apps, APIs, and more.

### ****1. Basic Syntax****

#### ****Hello World Example****

using System;

class Program {

static void Main() {

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

}

}

* using System; → Imports the System namespace (for input/output operations).
* Main() → The entry point of a C# program.
* Console.WriteLine() → Prints output to the console.

### ****2. Variables & Data Types****

int age = 25; // Integer

double price = 99.99; // Decimal numbers

string name = "John"; // String

bool isOnline = true; // Boolean

char grade = 'A'; // Character

### ****3. Conditional Statements****

if (age > 18) {

Console.WriteLine("Adult");

} else {

Console.WriteLine("Minor");

}

### ****4. Loops****

#### ****For Loop****

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

Console.WriteLine(i);

}

#### ****While Loop****

int x = 0;

while (x < 5) {

Console.WriteLine(x);

x++;

}

### ****5. Functions****

static int Add(int a, int b) {

return a + b;

}

Console.WriteLine(Add(5, 3)); // Output: 8

### ****6. Classes & Objects****

class Car {

public string brand; // Field

public Car(string brand) { // Constructor

this.brand = brand;

}

public void ShowBrand() {

Console.WriteLine("Car brand: " + brand);

}

}

Car myCar = new Car("Tesla");

myCar.ShowBrand(); // Output: Car brand: Tesla

### ****7. Generics (****<>****)****

Generics in C# allow you to create **type-safe** classes, methods, and collections without specifying a concrete data type.

#### ****Generic Class Example****

class Box<T> { // 'T' can be any data type

public T Value;

public Box(T value) {

Value = value;

}

public void Show() {

Console.WriteLine("Value: " + Value);

}

}

Box<int> intBox = new Box<int>(10);

intBox.Show(); // Output: Value: 10

Box<string> strBox = new Box<string>("Hello");

strBox.Show(); // Output: Value: Hello

#### ****Generic Method Example****

static void Print<T>(T item) {

Console.WriteLine(item);

}

Print(100); // Output: 100

Print("Hello"); // Output: Hello

### ****8. .NET Framework vs .NET Core****

* **.NET Framework** → Windows-based applications.
* **.NET Core / .NET (latest versions)** → Cross-platform (Windows, Linux, macOS).

Would you like a specific focus, such as **Web API with .NET** or **Entity Framework (ORM for databases)**? 🚀

### ****Understanding .NET Framework & ASP.NET****

.NET is a software development framework by Microsoft used to build desktop, web, and cloud applications.

### ****1. .NET Framework vs .NET Core vs .NET****

* **.NET Framework** → Windows-only, used for legacy enterprise applications.
* **.NET Core** → Cross-platform (Windows, Linux, macOS), open-source.
* **.NET (Latest)** → Unified version (combines .NET Framework & .NET Core).

### ****2. What is ASP.NET?****

ASP.NET is a **web framework** used to build web applications, APIs, and services.  
It runs on the .NET framework and supports **MVC (Model-View-Controller), Web APIs, and Razor Pages**.

### ****3. ASP.NET Key Technologies****

#### ****a) ASP.NET Web Forms (Old Approach)****

* Event-driven model (like Windows Forms).
* Drag-and-drop UI with .aspx pages.
* Not recommended for modern apps.

#### ****b) ASP.NET MVC (Model-View-Controller)****

* Separates application logic into **Model (data), View (UI), and Controller (handles requests).**
* More control over HTML, CSS, and JavaScript.
* Example:

public class HomeController : Controller {

public IActionResult Index() {

return View();

}

}

#### ****c) ASP.NET Web API****

* Used to build RESTful APIs.
* Works with JSON/XML responses.
* Example:

[ApiController]

[Route("api/products")]

public class ProductsController : ControllerBase {

[HttpGet]

public IEnumerable<string> Get() {

return new string[] { "Product1", "Product2" };

}

}

#### ****d) ASP.NET Core (Modern Approach)****

* Cross-platform, high-performance framework.
* Works on Windows, Linux, and macOS.
* Uses Middleware instead of Web Forms.
* Example:

var builder = WebApplication.CreateBuilder(args);

var app = builder.Build();

app.MapGet("/", () => "Hello, ASP.NET Core!");

app.Run();

### ****4. ASP.NET Deployment Options****

* **IIS (Internet Information Services)** → For Windows Server hosting.
* **Azure App Service** → Cloud-based hosting by Microsoft.
* **Docker Containers** → Deploy as microservices.
* **Self-hosted** → Use Kestrel server for direct hosting.

Would you like a **step-by-step guide** to setting up an **ASP.NET Core project** on Fedora? 🚀