

# Breaking Down with Methods in C#

By

Narasimha Rao T

*Microsoft .Net FSD Trainer*

Professional Development Trainer

[tnrao.trainer@gmail.com](mailto:tnrao.trainer@gmail.com)

## 1. Introduction to Methods

- **Definition:** A method is a block of code that performs a specific task, grouped together under a name.
- **Purpose:** Promote *code reusability, modularity, and organization*.

```
public int Add(int a, int b) {  
    return a + b;  
}
```

## 2. Why Use Methods?

- Break code into logical units.
- Avoid duplication (DRY principle).
- Improve readability and maintainability.
- Enable code reuse and testing.

### 3. Benefits of Methods

Benefit	Description
Reusability	Use the same logic in multiple places.
Readability	Shorter, cleaner main program.
Debugging Ease	Isolate and test small code segments.
Abstraction	Hide implementation details.
Maintainability	Easier to update or modify functionality.

## 4. Declaring and Calling Methods

### Declaration Syntax:

```
[access_modifier] [return_type] MethodName([parameters]) {  
    // method body  
}
```

### Calling a Method:

```
int result = Add(5, 3);
```

## 5. Method Parameters

### ref Parameter

- Passes reference to the original variable.
- Requires initialization before call.

```
void Add(ref int a) {  
    a += 10;  
}
```

## **out** Parameter

- Method *must* assign a value.
- Doesn't require initialization before call.

```
void GetValues(out int x, out int y) {  
    x = 5;  
    y = 10;  
}
```

## Optional Parameters

- Provide default values for parameters.

```
void Greet(string name = "Guest") {  
    Console.WriteLine($"Hello, {name}");  
}
```

## 6. Method Overloading

- Define multiple methods with the same name but different signatures.

```
int Add(int a, int b) {  
    return a + b;  
}  
  
double Add(double a, double b) {  
    return a + b;  
}
```

## 7. Variable Scope

Scope	Description
Local	Declared inside a method, only accessible there.
Global	C# doesn't support true global variables; use static class-level members.
Static	Belongs to the class, shared across instances.

## Example

```
static int counter = 0;

void DoSomething() {
    int localVar = 10; // Local
    counter++;        // Static/global-level
}
```

# Naming Conventions and Organization

## Naming Guidelines:

- **PascalCase** for method names: `CalculateSum` , `PrintResult`
- Use **verbs** to indicate action: `GetData()` , `SetName()`
- Keep method names **descriptive**, not too short.

## Organization Tips:

- Group related methods in classes.
- Place public methods at the top of the class.
- Keep methods short — single responsibility principle (SRP).

## Q & A

---

Narasimha Rao T

[tnrao.trainer@gmail.com](mailto:tnrao.trainer@gmail.com)