Methods

Named Blocks of Code

Topics

- Methods
- Parameters
- Pass by Value
- Pass by Reference
- Pass by Reference using Out
- Optional Parameters
- Intermediate Level

Methods

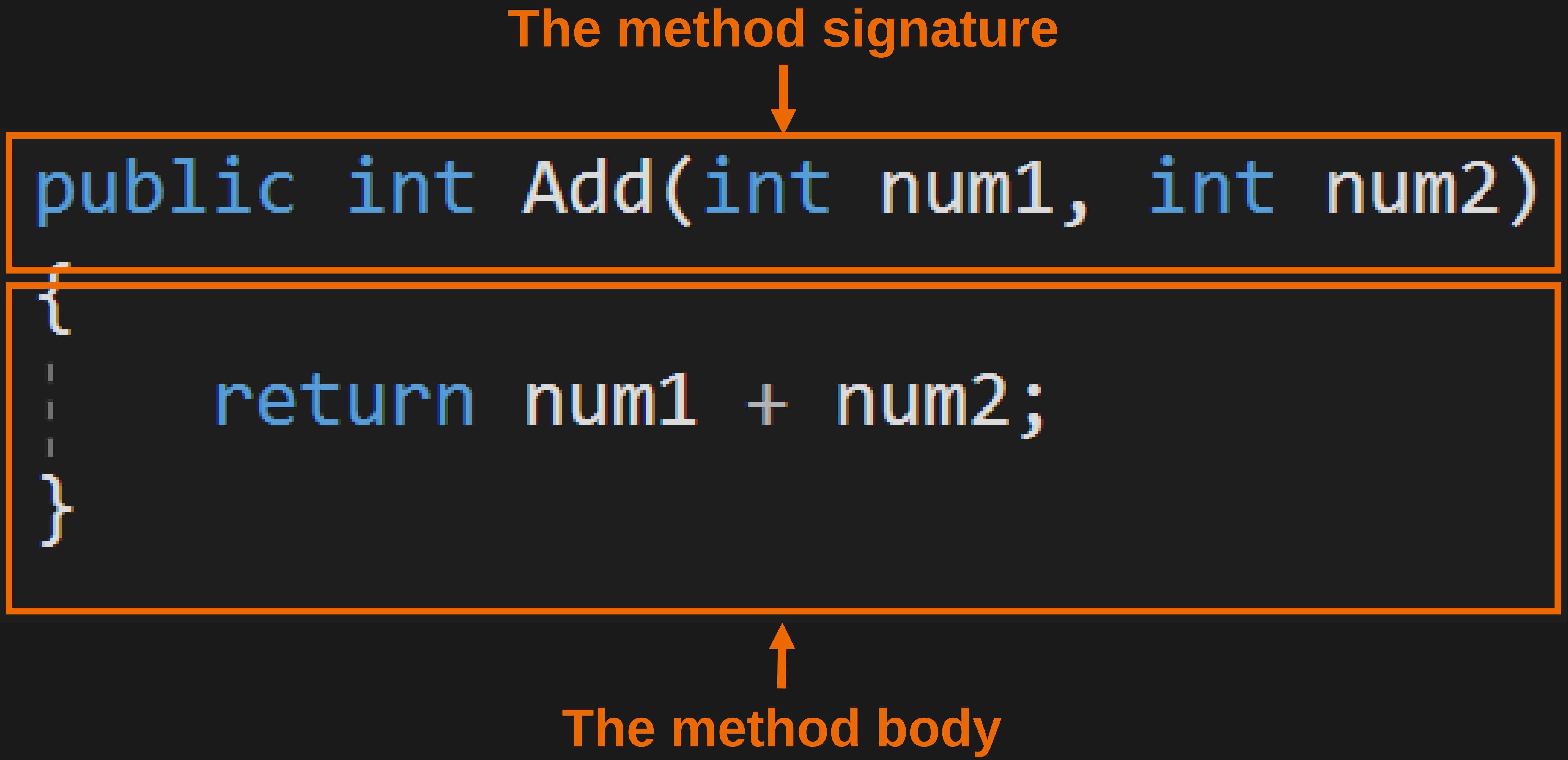
https://www.tutorialspoint.com/csharp/csharp_methods.htm

All Code

$$\underbrace{\text{variables} \qquad \begin{matrix}\text{if-else}\\ \\ \text{switch}\end{matrix} \qquad \begin{matrix}\text{for}\\ \text{while}\\ \text{do-while}\end{matrix}}_{\text{Code}}$$

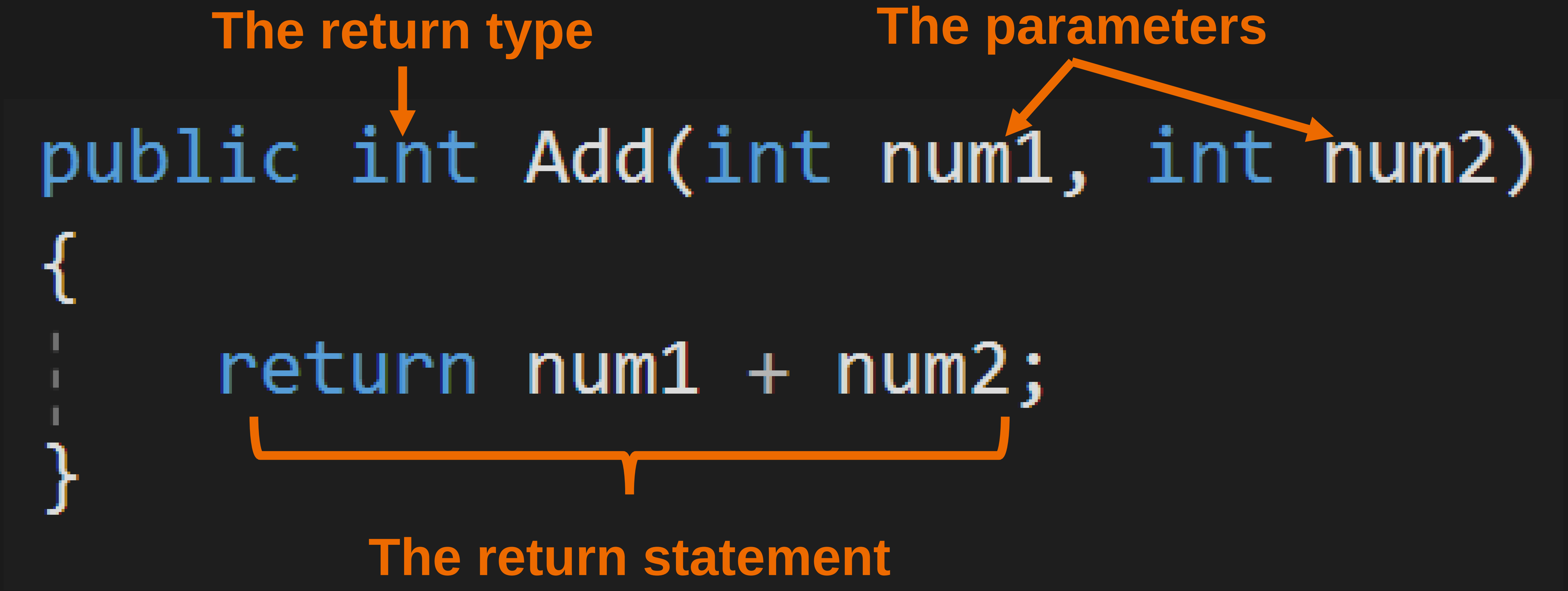
Methods are just named blocks of code.

The Basics: Methods



```
public int Add(int num1, int num2)
{
    return num1 + num2;
}
```

The Basics: Methods



```
public int Add(int num1, int num2)
{
    return num1 + num2;
}
```

#1 Method Challenge

- Create a method called **PrintMessage**.
  - In the method, print whatever message you want.
- Call **PrintMessage** from Main.

LINKS

[Method return type: void]

#2 Return Type Challenge

- Create a method called **GetMessage**.
  - In the method, ask the user to enter a message.
  - Return the message.
- Call **GetMessage** from Main.
- Store the message in a string variable.

LINKS

Method return type

Slides

Return Info

Method Parameters

Method Parameters

- There are 2 ways to pass parameters to methods:
  - Pass by Value
  - Pass by Reference

Pass by Value

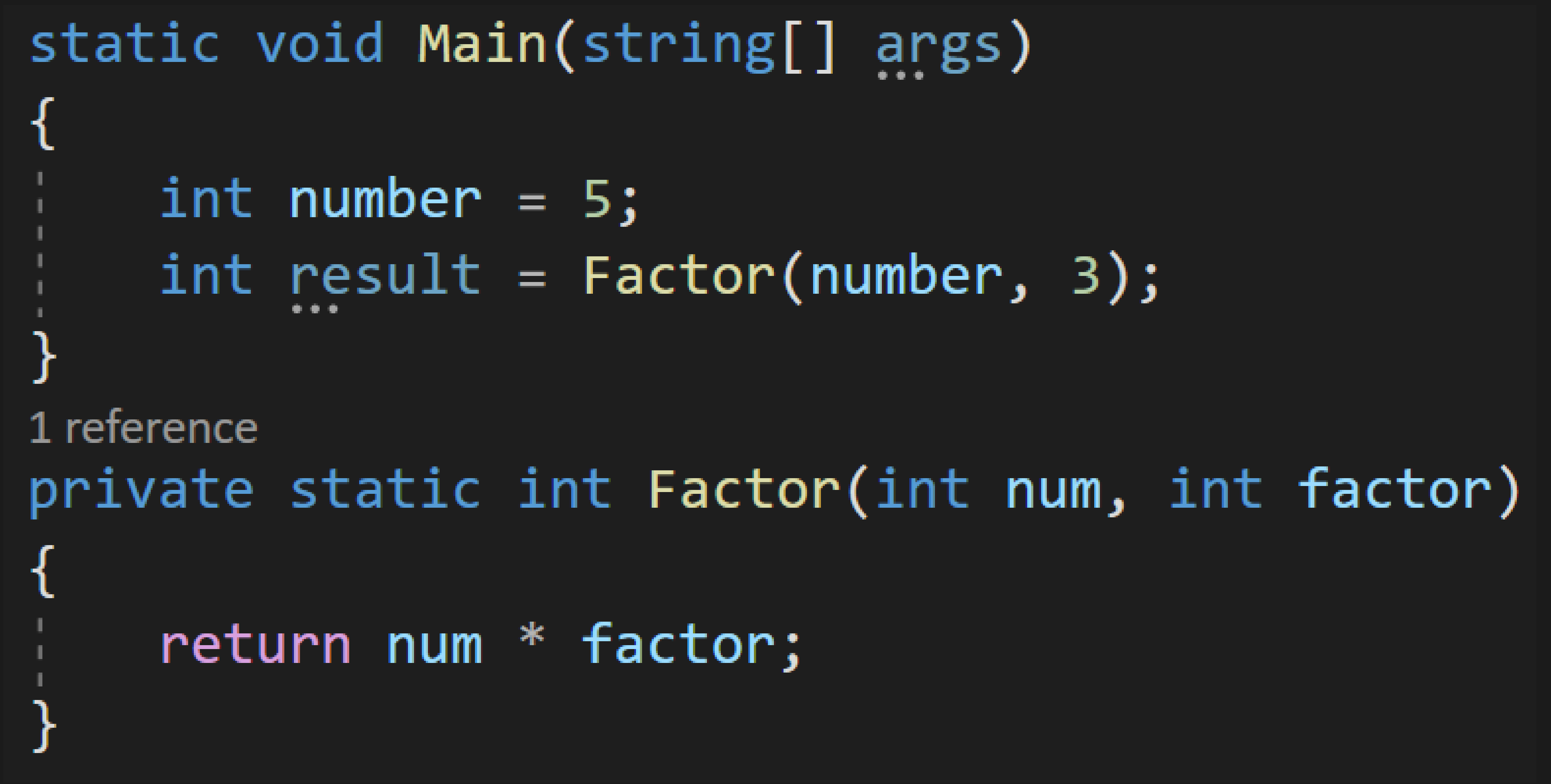
Methods: Pass by Value

- When you pass variables to a method using Pass by Value, you should think of one word: COPY

Pass by Value = COPY

- You are copying the value from the variable to a new variable (the parameter).
- Changes made to the parameter in the method DO NOT affect the variable used to call the method.

Methods: Pass by Value (COPY)



```
static void Main(string[] args)
{
    int number = 5;
    int result = Factor(number, 3);
}
1 reference
private static int Factor(int num, int factor)
{
    return num * factor;
}
```

- Factor has 2 local variables: num and factor.
- The *value* of number is copied into a new variable called num.

#3 Pass By Value Challenge

- Create another method called **PrintMessage**.
  - The method should have 1 string parameter
  - In the method, print the string parameter.
- Call **PrintMessage** from Main and pass the string variable from the prior challenge.

LINKS

Method parameters

Slides

Pass by Value How-To

Method Parameters

Method Parameters

- There are 2 ways to pass parameters to methods:
  - Pass by Value
  - Pass by Reference

Pass by Reference

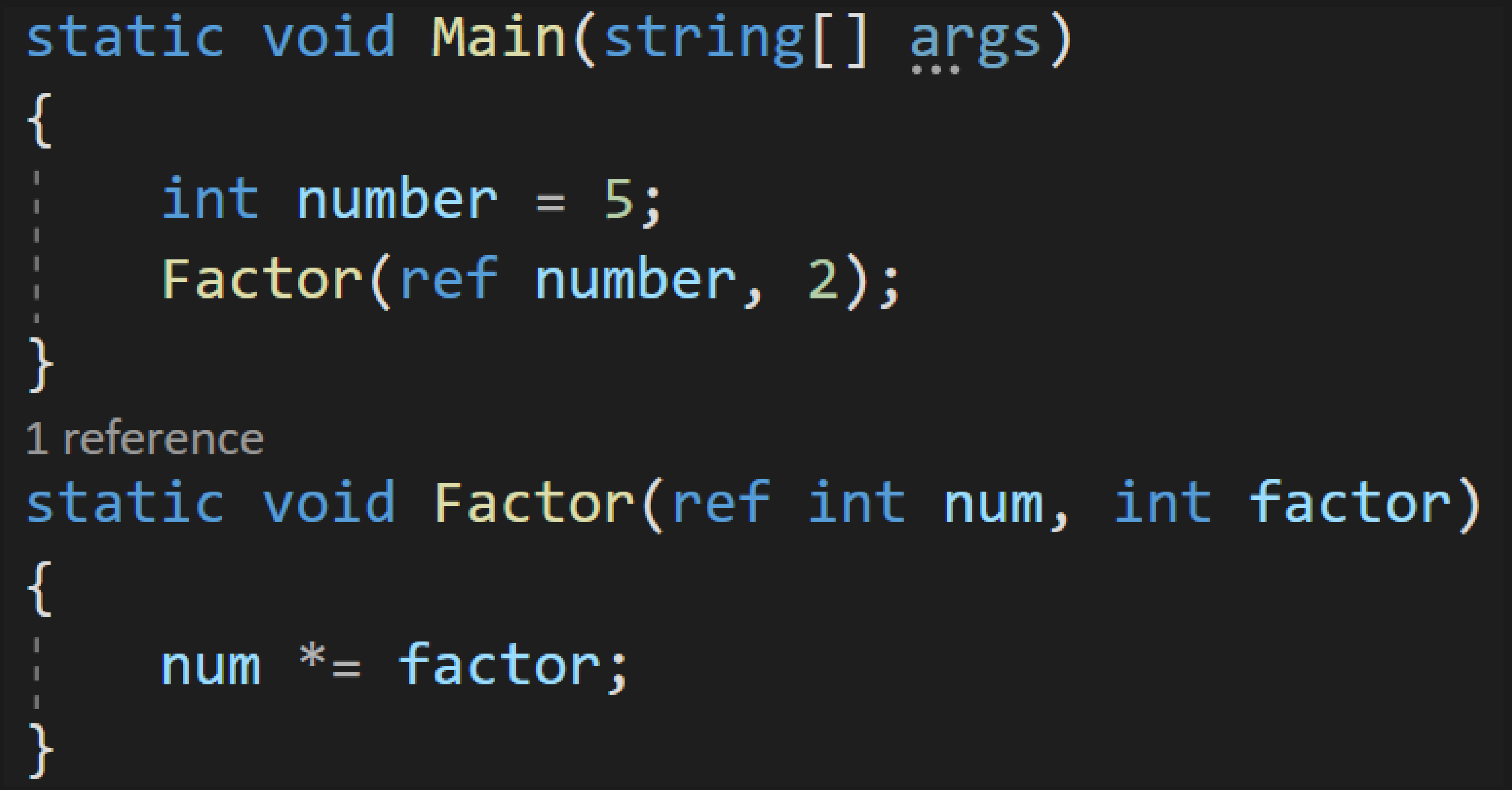
Methods: Pass by Reference

- When you pass variables to a method using Pass by Reference, you should think of one word: ALIAS

Pass by Reference = ALIAS

- You are creating a new name for the variable.
- Any changes to the parameter in the method affect the variable used when calling the method.

Methods: Pass by Reference (ALIAS)



```
static void Main(string[] args)
{
    int number = 5;
    Factor(ref number, 2);
}
1 reference
static void Factor(ref int num, int factor)
{
    num *= factor;
}
```

- number is given a new name (num) in Factor.
- num is the *same variable* as number.

#4 Pass by Ref Challenge

- Create another method called **TimeStamp**.
  - The method should have 1 string parameter that is **passed by reference**.
  - In the method, **prefix** the string with the current DateTime.
- Call **TimeStamp** from Main and pass the string variable from the prior challenge.
- Call PrintMessage and pass the newly updated string.

LINKS

Pass-by-ref parameters

DateTime.Now

Interpolated strings

Slides

Pass by Ref How-To

Out Parameters

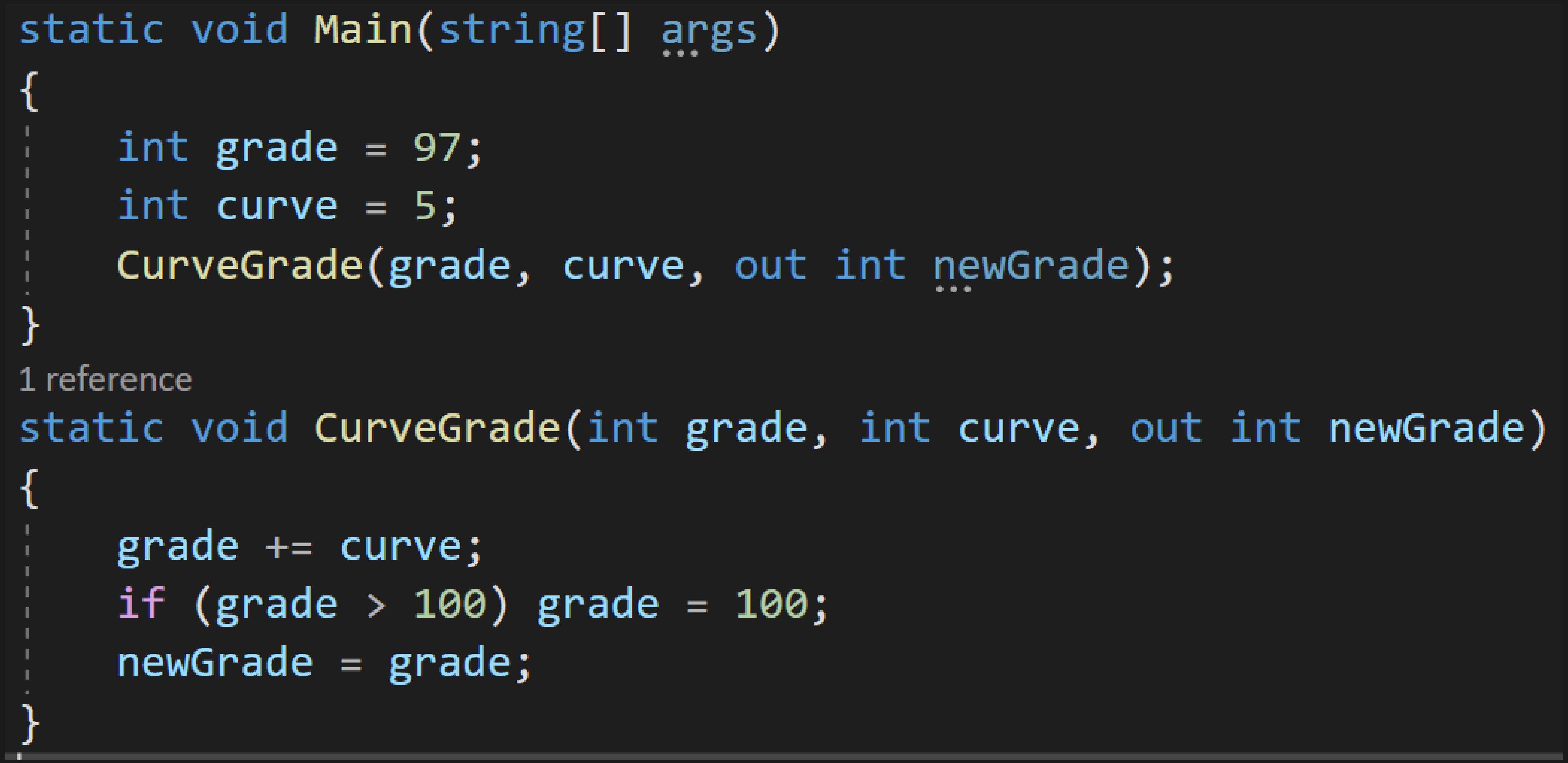
A special kind of Pass by Reference

Methods: Out vs Ref

- Pass by Ref Requirements:
  - the variable you pass must be initialized
    string name **= string.Empty**;
    GetName(ref name);

- **out** is a specialized passing by reference.
  - You do NOT need to initialize the variable before calling the method.
  - The method is *required* to set the variable before returning.
    bool result = int.TryParse(ageInput, **out int age**);

Methods: Pass by Reference using OUT



```
static void Main(string[] args)
{
    int grade = 97;
    int curve = 5;
    CurveGrade(grade, curve, out int newGrade);
}
1 reference
static void CurveGrade(int grade, int curve, out int newGrade)
{
    grade += curve;
    if (grade > 100) grade = 100;
    newGrade = grade;
}
```

- CurveGrade is *required* to set the newGrade variable.

#5 Pass by Ref with OUT Challenge

- Create a method called **MyFavoriteNumber**.
  - The method should have 1 int parameter that is **passed by reference using out**.
  - In the method, ask the user to enter their favorite number.
  - Convert the input to an integer and assign it to the parameter.
- Call **MyFavoriteNumber** from Main and pass an int variable.
- Print the int variable.
  - EX: “My favorite number is 5”

LINKS

Out parameters

Interpolated strings

int.TryParse

Slides

Pass by Ref with OUT How-To

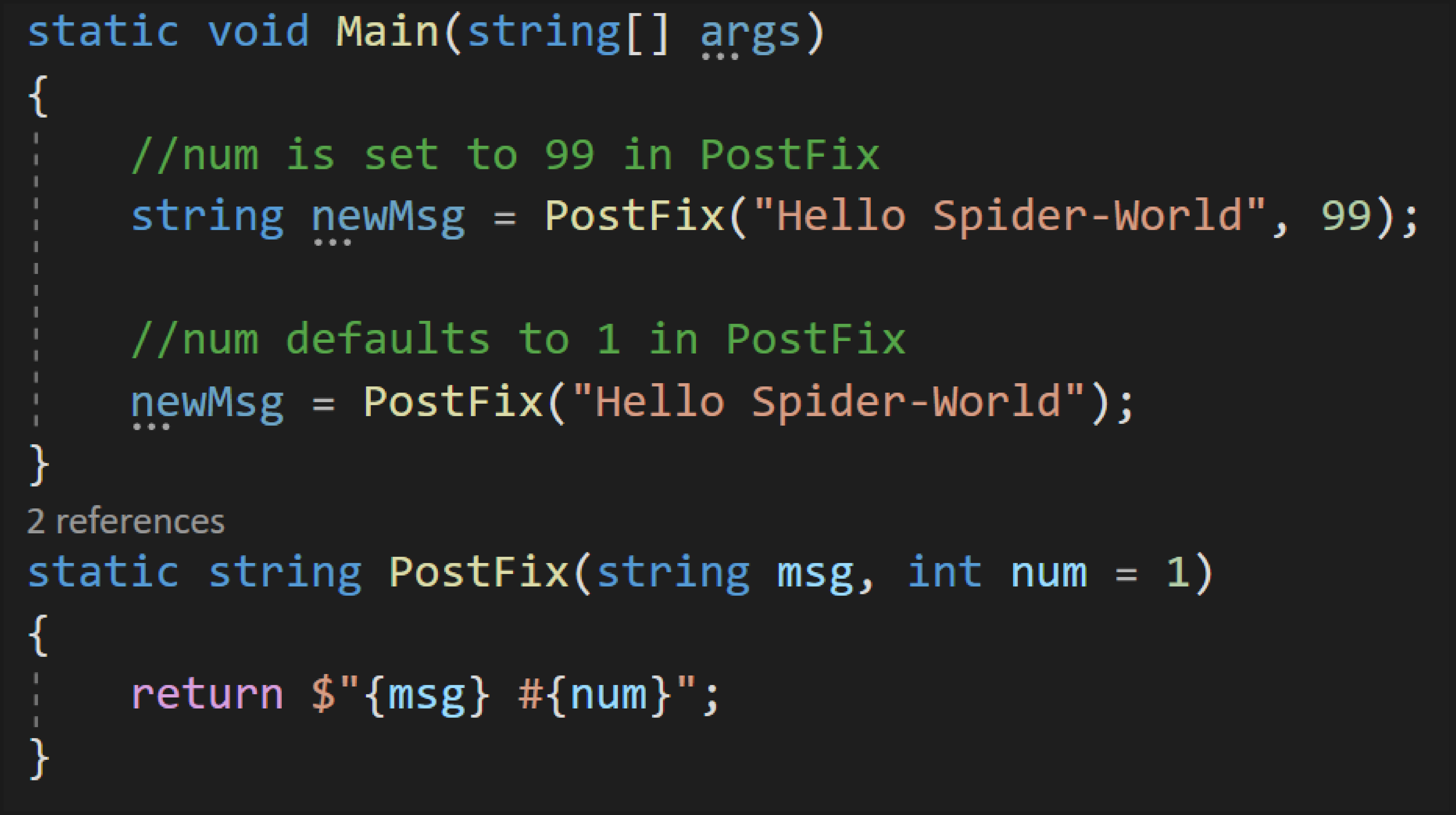
Optional Parameters

Methods: Optional Parameters

- You can make the parameters optional.
- Optional parameters means that the code that calls your method does not need to pass a value for the parameter.
- Optional parameters must appear at the end of the parameter list.

- EXAMPLE:
- public void MethodWithOptional(int nonoptional, bool isOk = false)

Methods: Optional Parameter



```
static void Main(string[] args)
{
    //num is set to 99 in PostFix
    string newMsg = PostFix("Hello Spider-World", 99);

    //num defaults to 1 in PostFix
    newMsg = PostFix("Hello Spider-World");
}
2 references
static string PostFix(string msg, int num = 1)
{
    return $"{msg} #{num}";
}
```

#3 Optional Parameter Challenge

- Modify **PrintMessage**.
    - Make the string parameter optional where the default is “Hello Gotham.”
- Remove the other PrintMessage that doesn’t have a parameter.
- Call **PrintMessage** from Main and pass the string variable from the prior challenge.
- Call **PrintMessage** from Main without a parameter.

LINKS

Method parameters

Slides

Pass by Value How-To

Intermediate Level

Return Multiple Values

#6 Tuple Return Type Challenge

- Create a method called **MyName**.
  - In the method, call Console.ReadLine 3 times to get the first, middle, and last name.
  - Return all 3 as a tuple return type.
- Call **MyName** from Main. Deconstruct the tuple into named variables first, middle, last.
- Print the name (last, first middle).
  - EX: “Poe, Edgar Allen”

LINKS

Tuple Return Types