4 + 1 Kinds of Parameters

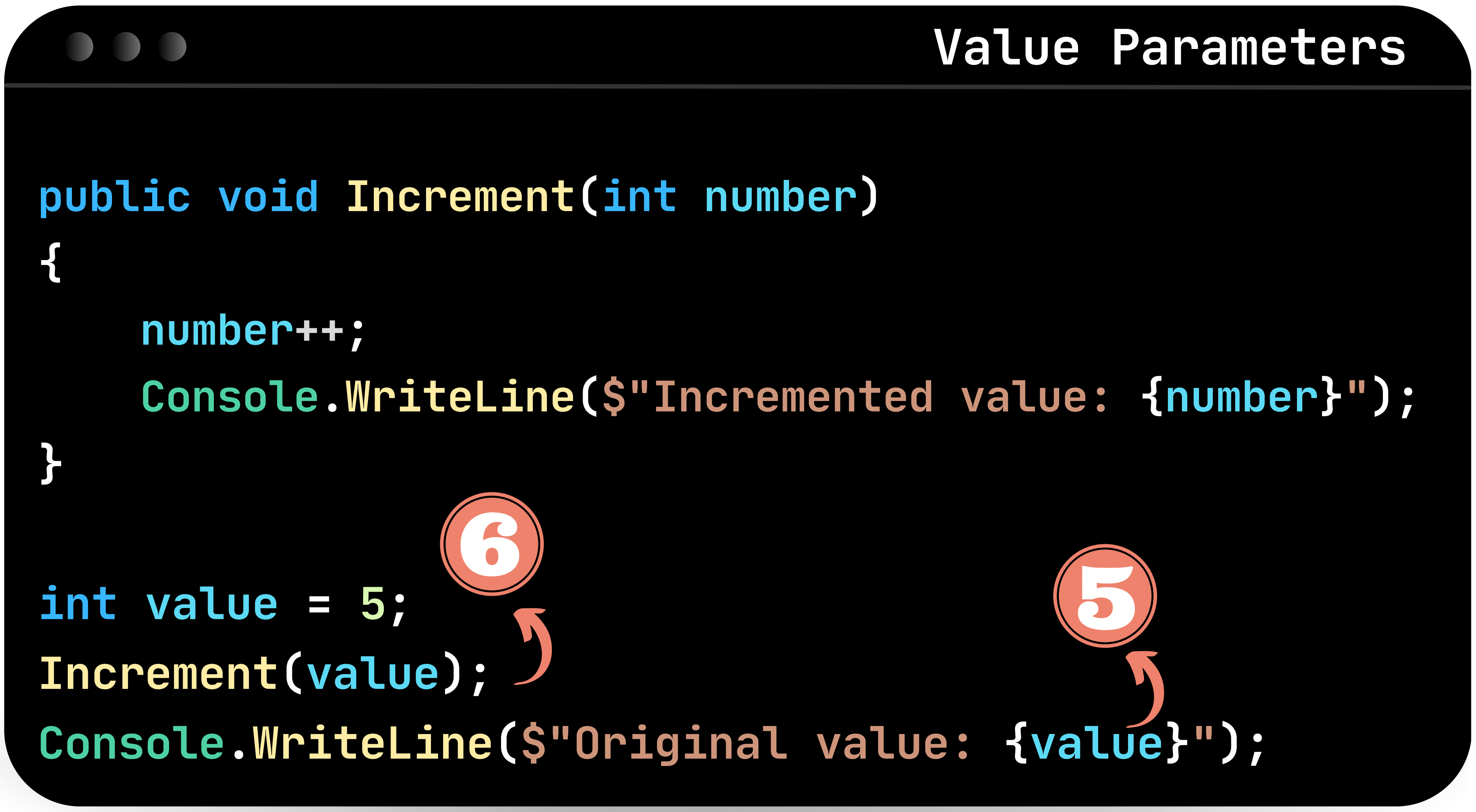
with examples

Georgios Petas

Value Parameters

- Pass values to a method by creating a **copy** of the **original value for** value types and **copy** of the **reference to the object** for reference types.
- **Changes** made to the parameter within the method **do not affect** the original value.
- They are declared **without** any special keywords.

Example :

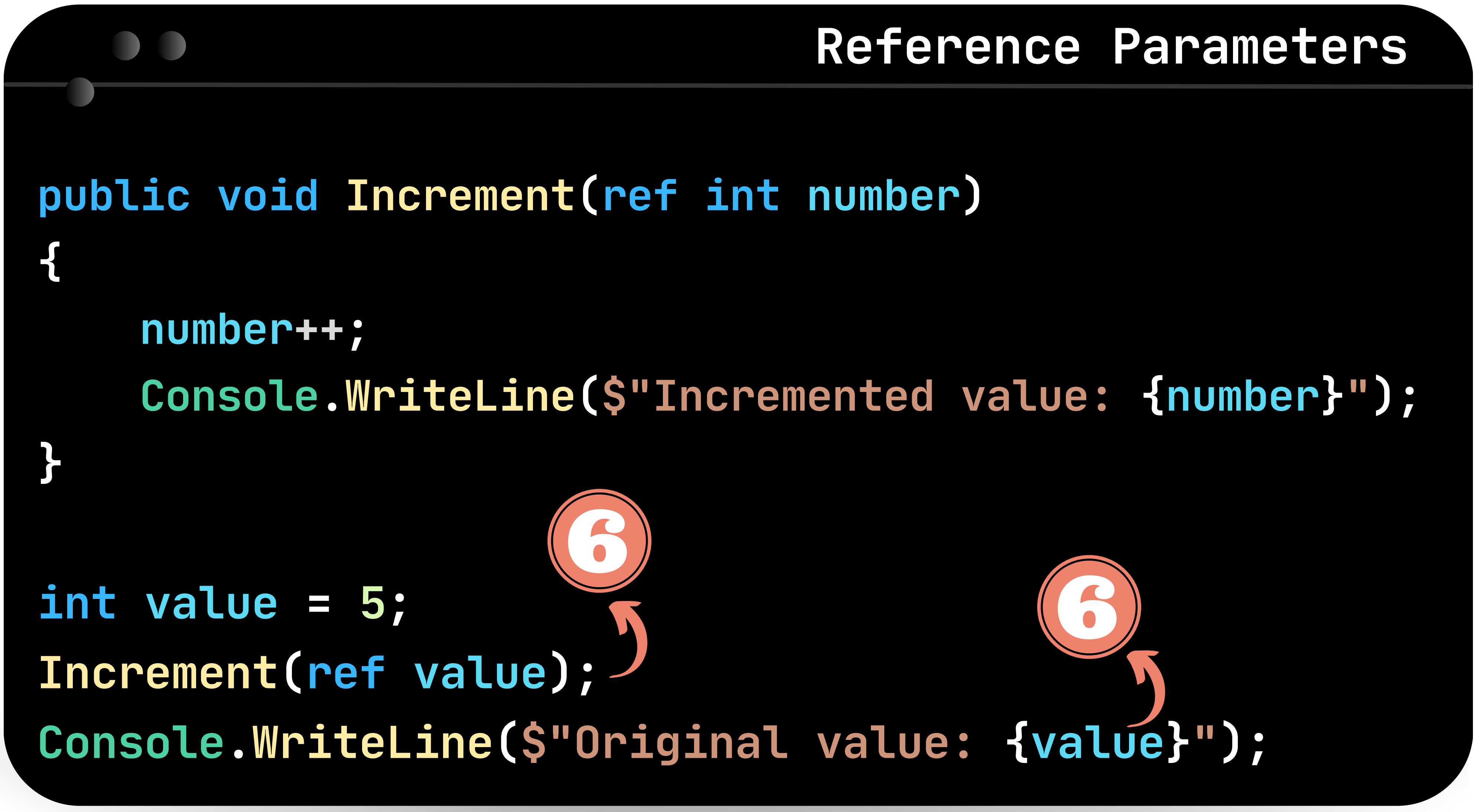


```
public void Increment(int number)
{
    number++;
    Console.WriteLine($"Incremented value: {number}");
}

int value = 5;
Increment(value);
Console.WriteLine($"Original value: {value}");
```

Reference Parameters

- Allow you to pass a **reference** to a variable.
- During execution of the method, it represents the **same** **storage location** as the argument variable.
- **Changes** made to the parameter **within the method** will **affect** the **original variable**.
- They are declared using the **ref** keyword **before** the **parameter type**.

Example :

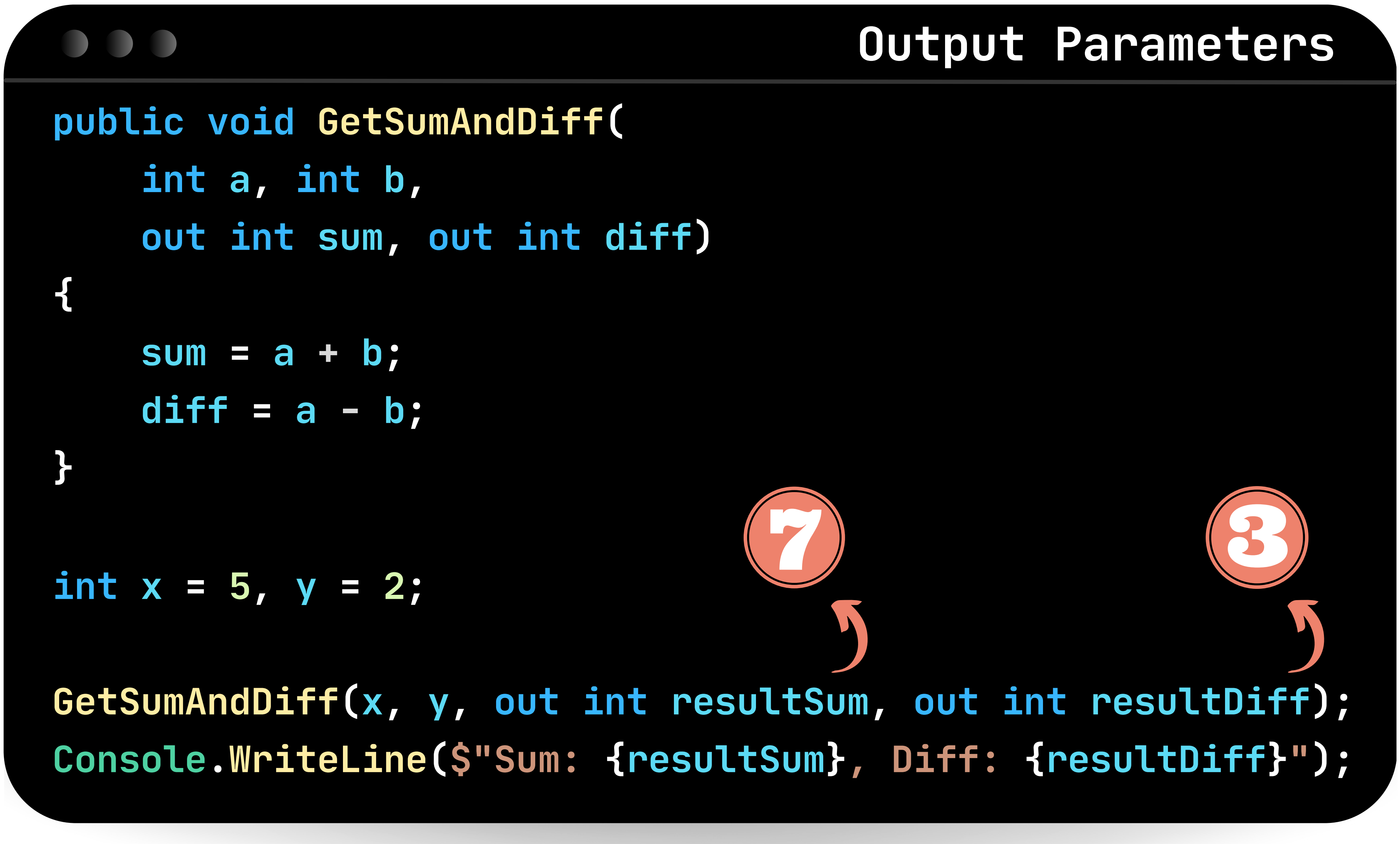


```
public void Increment(ref int number)
{
    number++;
    Console.WriteLine($"Incremented value: {number}");
}

int value = 5;
Increment(ref value);
Console.WriteLine($"Original value: {value}");
```

Output Parameters

- Similar to **reference** parameters but are used to **return values** from a method rather than pass values in.
- **Do not require** an initial value before passing them to the method.
  (*Unlike reference parameters*)
- They are declared using the out keyword **before** the **parameter** type.

Example :



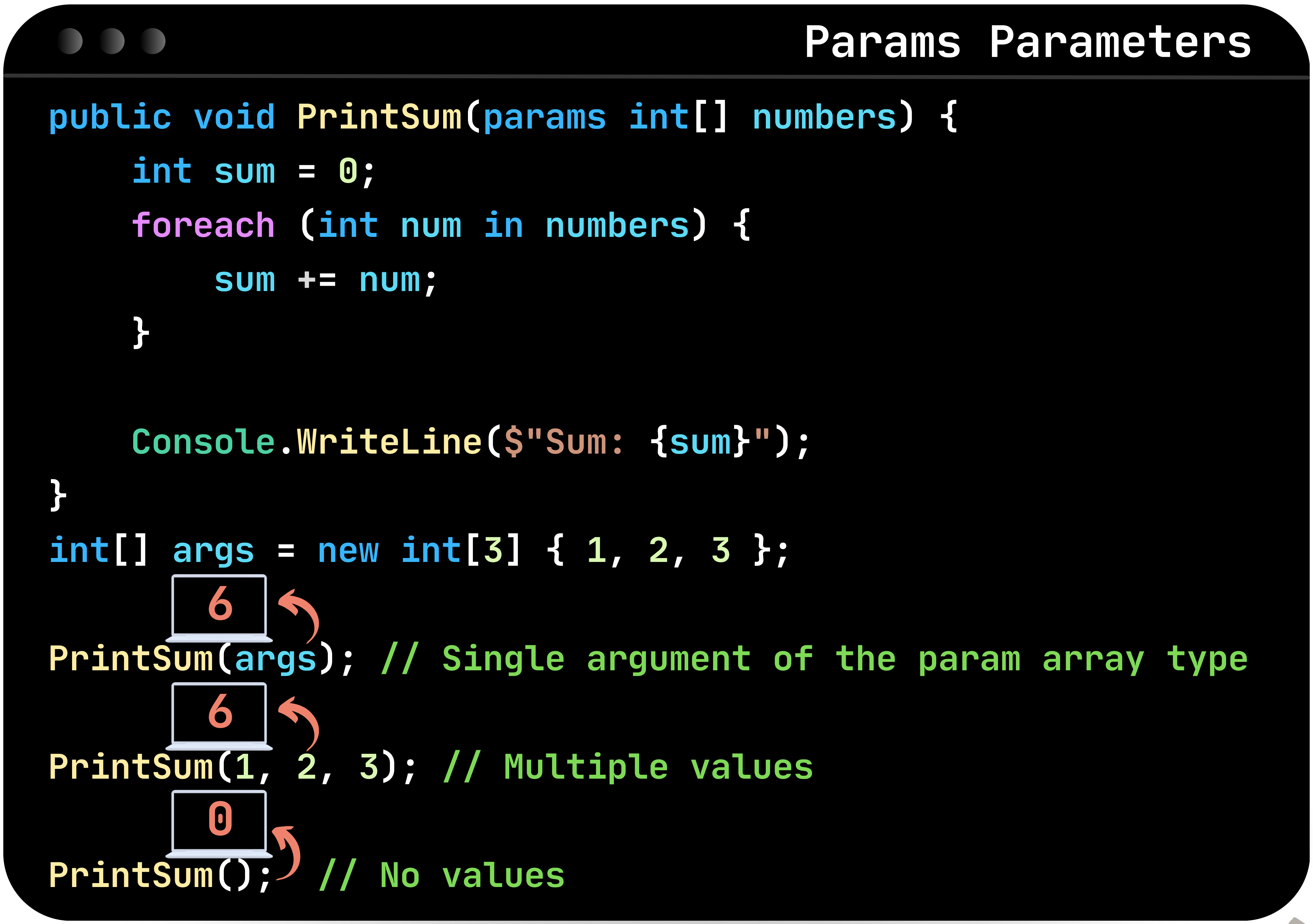
Output Parameters

```
public void GetSumAndDiff(
    int a, int b,
    out int sum, out int diff)
{
    sum = a + b;
    diff = a - b;
}

int x = 5, y = 2;

GetSumAndDiff(x, y, out int resultSum, out int resultDiff);
Console.WriteLine($"Sum: {resultSum}, Diff: {resultDiff}");
```

Params Parameters

- **Allow** you to pass a **variable number of arguments** to a method.
- They are **declared** using the **params** keyword **followed** by an **array type**.
- Params can receive **zero or more values** of the **specified type**.
- **Within the method**, the params parameter behaves **like an array** of the **specified type**.

Example :



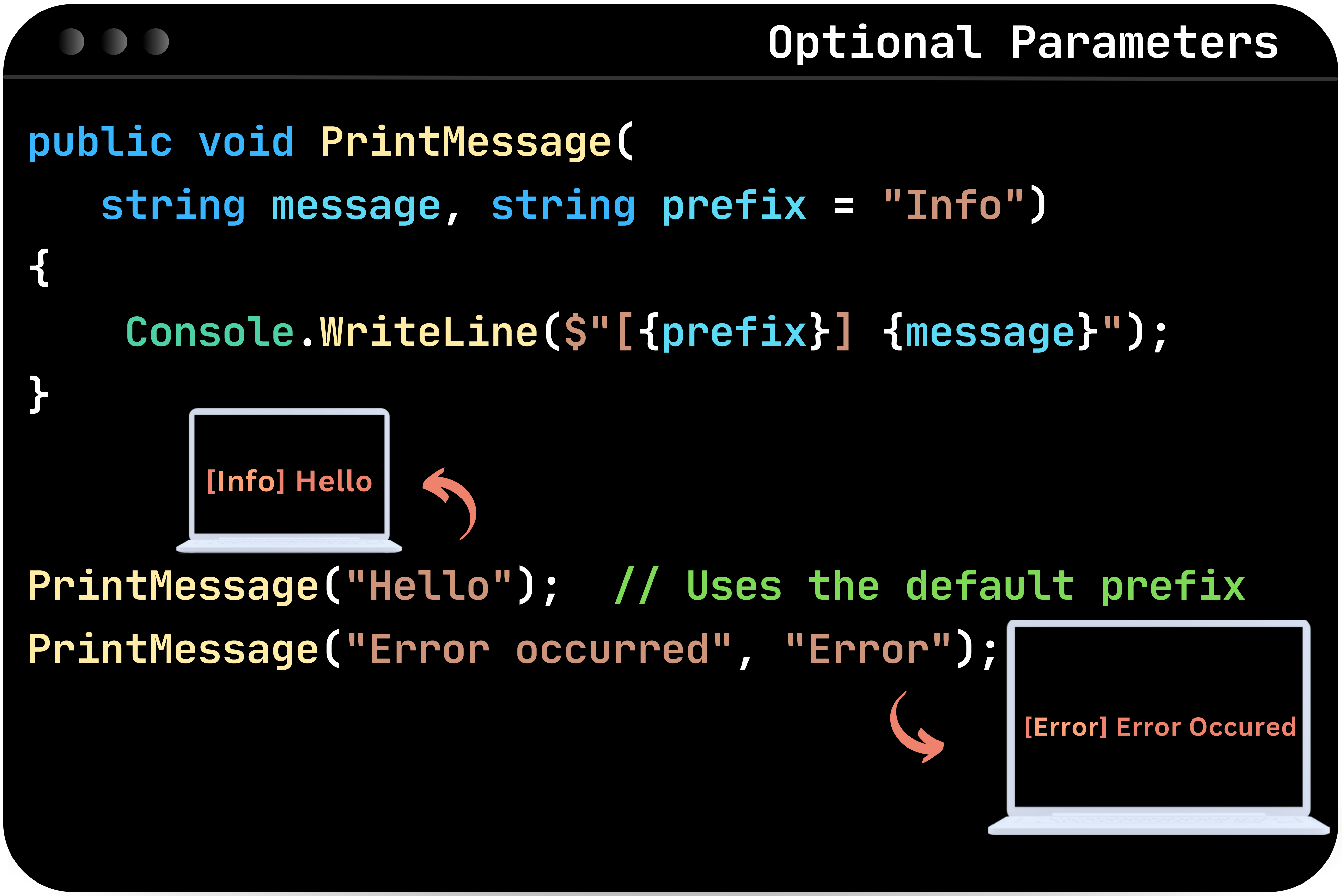
```
public void PrintSum(params int[] numbers) {
    int sum = 0;
    foreach (int num in numbers) {
        sum += num;
    }

    Console.WriteLine($"Sum: {sum}");
}
int[] args = new int[3] { 1, 2, 3 };

PrintSum(args); // Single argument of the param array type

PrintSum(1, 2, 3); // Multiple values

PrintSum(); // No values
```

Optional Parameters

*not considered a separate kind of parameter, but a feature.

- **Allow** you to specify **default values** for method parameters.
- They are **specified** by **assigning a default value** in the **method declaration**.
- Useful when you want to provide **flexibility** by allowing **certain parameters to be omitted**.

Example :



Optional Parameters

```
public void PrintMessage(
   string message, string prefix = "Info")
{
    Console.WriteLine($"[{prefix}] {message}");
}

PrintMessage("Hello");  // Uses the default prefix
PrintMessage("Error occurred", "Error");
```

```
[Info] Hello
```

```
[Error] Error Occured
```

That's it!!

If you find this valuable,

follow me for more.

And click the bell in my profile 🔔

Georgios Petas