## Debugging code

### Objective:

Demonstrate how to use basic debugging techniques and tools, such as setting breakpoints, inspecting variable values, and stepping through code to identify and fix errors.

### Description:

This activity will guide participants through debugging C# code, including identifying errors, understanding their causes, and applying fixes. It will start with two fully debugged examples and then provide two additional debugging problems for learners to solve. Common error types, such as syntax, runtime, and logical errors, will be addressed.

### Set Up Your Environment

- Use the Visual Studio Code console application you created at the start of the course. Remove any existing code in the Program.cs file of your console application.
- Copy and edit the code in this activity into the Program.cs file to complete the steps.

### Problem 2: Finding the Maximum Number in an Array

### Problem Description:

The following code tries to find the maximum number in an array. It has a logical error that causes it to produce incorrect results when all the numbers are negative.

```
public class Program
{
    public static int FindMax(int[] numbers)
    {
        int max = 0;
        for (int i = 0; i < numbers.Length; i++)
        {
            if (numbers[i] > max)
            {
                max = numbers[i];
            }
        }
        return max;
    }
    public static void Main()
    {
        int[] myNumbers = { -5, -10, -3, -8, -2 };
        int maxNumber = FindMax(myNumbers);
        Console.WriteLine("The maximum number is: " + maxNumber);
    }
}
```

**Code:**

```
public class Program
{
    // Method to find the maximum number in an array
    public static int FindMax(int[] numbers)
    {
        // Error check 1: array is null
        if (numbers == null)
        {
            throw new ArgumentNullException(nameof(numbers), "Input 
array cannot be null.");
        }

        // Error check 2: array is empty
        if (numbers.Length == 0)
        {
            throw new ArgumentException("The array cannot be empty.");
        }

        // Initialize max to the first element
        int max = numbers[0];

        // Loop through the array starting from the second element
        for (int i = 1; i < numbers.Length; i++)
        {
            // Error check 3: check for int.MinValue overflow
            if (numbers[i] < int.MinValue || numbers[i] > int.MaxValue)
            {
                throw new OverflowException("Array contains an out-of-
range integer.");
            }

            if (numbers[i] > max)
            {
                max = numbers[i];
            }
        }

        return max;
    }

    public static void Main()
    {
        try
        {
            // Test array
            int[] myNumbers = { -5, -10, -3, -8, -2 };

            // Uncomment the following lines one at a time to test 
different errors:
            // int[] myNumbers = null;               // Triggers 
ArgumentNullException
            // int[] myNumbers = { };                // Triggers 
ArgumentException
            // int[] myNumbers = new int[1000000]; // Simulated size 
check (if needed)

            int maxNumber = FindMax(myNumbers);
            Console.WriteLine("The maximum number is: " + maxNumber);
        }
        catch (ArgumentNullException ex)
        {
            Console.WriteLine("Null Error: " + ex.Message);
        }
        catch (ArgumentException ex)
        {
            Console.WriteLine("Argument Error: " + ex.Message);
        }
        catch (OverflowException ex)
        {
            Console.WriteLine("Overflow Error: " + ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine("Unexpected Error: " + ex.Message);
        }
    }
}
```