Mr. Bellavia

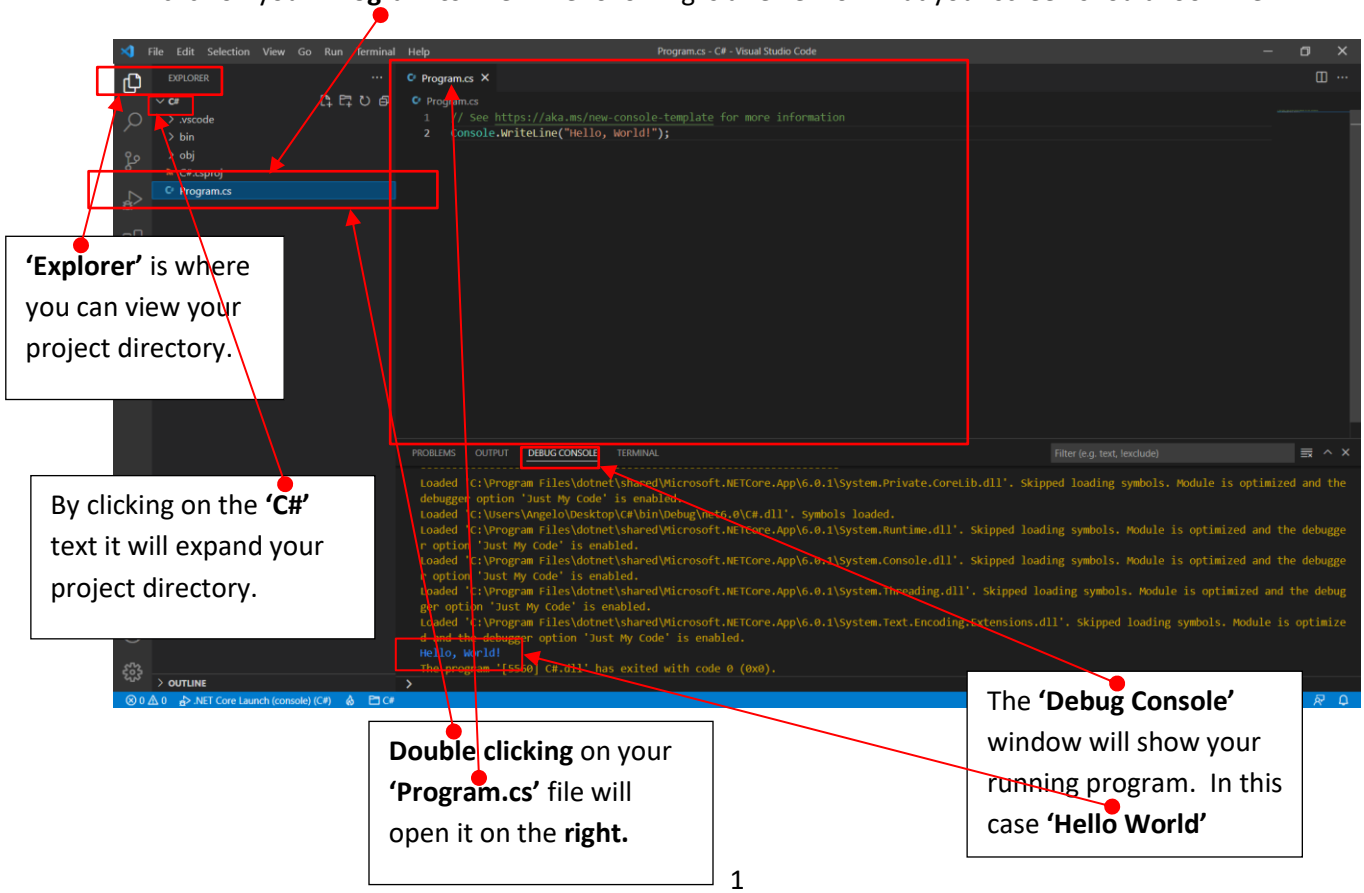
LESSON 01 - INTRODUCTION TO C# (HELLO WORLD & OUTPUT)

In this lesson we will explore the 'Hello World!' program that we did at the end of the last unit, as well as examine the 'Console.WriteLine()' statement used for output. Please know that coding takes time to learn so we will go slow. Before we can do anything fantastic, like create a game with Unity, we need to learn the basics of C#.

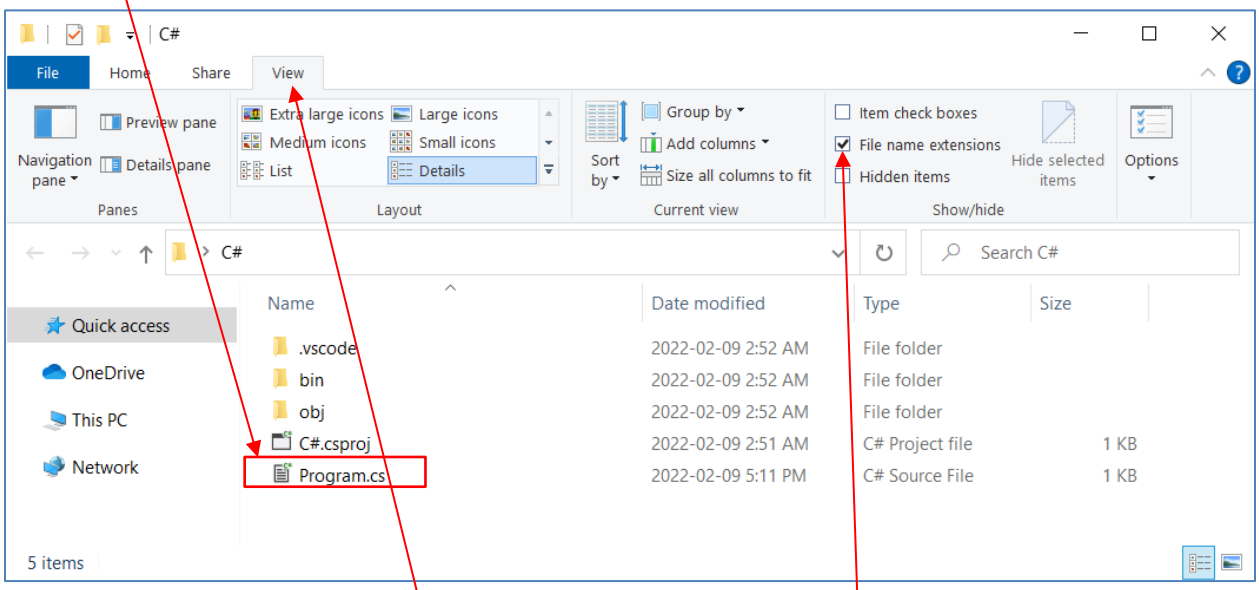
Sections:

I. 'HELLO WORLD' PROGRAM:

Let us review our 'Hello World' program we did at the end of the last unit. When you open Visual Studio Code it should open to the last project you were working on. If it does not, then open your working C# folder within Visual Studio that we created at the end of the last unit, then double click on your **'Program.cs'** file. The following is a review of what your screen should look like:



One thing to notice is the name of your file **'Program.cs'**. Just like Word files have a file extension of .docx, or images may have a file extension of .jpg, all C# files have a file extension of **'.cs'** (short for C#). In fact, we can double click our C# folder on your desktop, and you will see the **'Program.cs'** file directly:

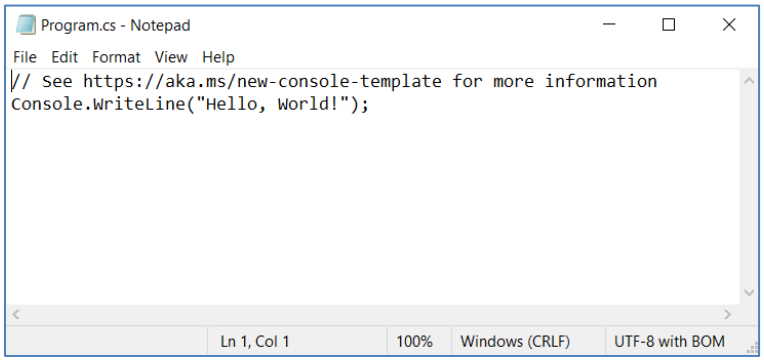


Note: If you can not see the file extensions like what is shown above there is an option to show them. In **Windows** click the **'View'** tab followed by the **'File name extensions'** checkbox.

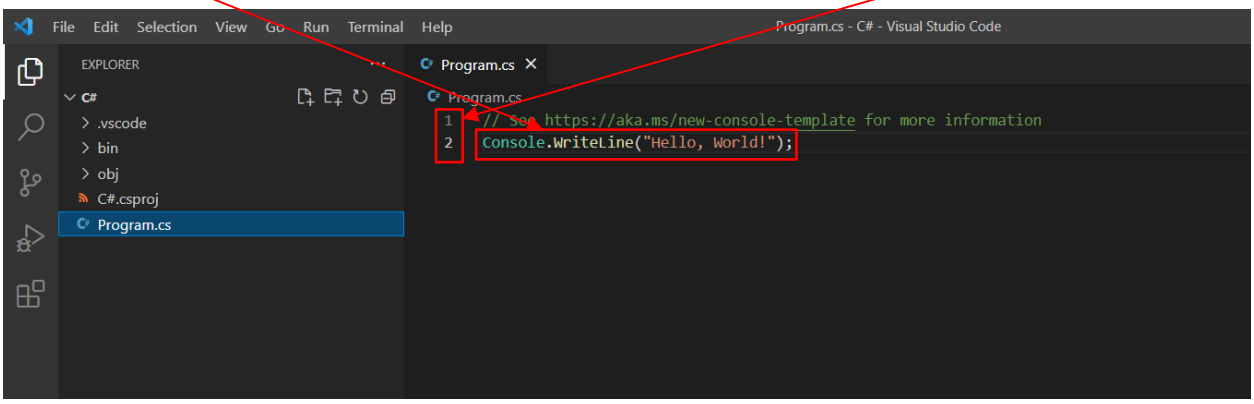
The only file you are ever going to use is this **'Program.cs'** file, so **do not worry about all these other files!**

Let us focus on the **Program.cs** file. This file is just a simple text file, but because it has a **.cs** file extension the icon changes to a .cs icon in Windows:

You can edit this file with a text editor, and you will see your code. Here is an example of **'Program.cs'** opened in Windows Notepad:



In fact, you can run this C# code without using Visual Studio. Why should we use Visual Studio? Because we need the .Net Framework installed and because Visual Studio gives us a lot more tools that are helpful when programming. For example, in Visual Studio the code has **line numbers** and is **colour coded**, the compiler is built-in, and much more:



Let us dive into the code in our **'Program.cs'** file. First and foremost, **code generally executes sequentially,** that is, from top to bottom starting from **line 1** (which in this case is just a comment). Recall that comments and whitespace get ignored. Feel free to remove this comment from your code.

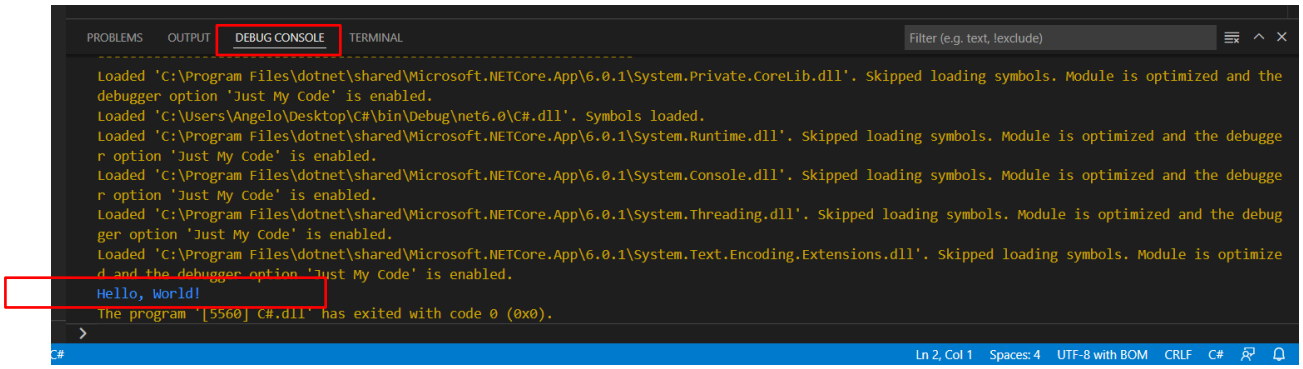
```
1. // See https://aka.ms/new-console-template for more information
2. Console.WriteLine("Hello, World!");
```

Our code above, for all intents and purpose, contains only 1 line of code which appears on **line 2** (**Note:** A line of code is usually referred to as a **statement**):

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

This statement is what displays the message **"Hello World!"** to the **Debug Console** when you run your program. (**Remember**: To run your program click **'Run'** from the top menu bar then click **'Start Debugging'** or hit **'F5'.**)

The output will appear in the **Debug Console:**



II. OUTPUT USING 'Console.WriteLine()':

The **Console.WriteLine()** statement is considered a **built-in function.** You'll see many built-in functions pop up as you become a more experience coder. A **function** in C# has a **name**, in this case 'Console.WriteLine' and a set of brackets **( )**. The value we put inside these brackets is called the **function parameter.** In the case of Console.WriteLine(), this function takes a text message saying, "Hello World!" as a parameter (text in programming are called 'strings', but we will discuss that in a later lesson). The text inside must always be surrounded by quotation marks **" "**. Also remember that all statements in C# must end in a **semi-colon.**

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

The **Console.WriteLine()** statement can obviously contain any message you wish. You can also output special characters and escape characters. An escape character is a special character that allows you to output things like tabs and newlines. For example, the '\n' escape character will output a newline:

```
Console.WriteLine("Hello, World!\nHow are you?");
```

Output will be:

```
Hello World!
How are you?
```

Of course, you could achieve the same output above using **two** Console.WriteLine() statements:

```
Console.WriteLine("Hello, World");
Console.WriteLine("How are you?");
```

Note: You can look up all the available escape characters in C# on google (for example, '\t' which is the TAB character).