100 XP

Exercise - Write your first code

6 minutes

In this first hands-on exercise, you'll use C# to print a hallowed programmer's phrase to the standard output of a console.

Write your first line of code

There's a long-standing tradition among software developers to print the phrase "Hello World!" to the console output window. As you'll experience, you can learn a lot about programming and the C# programming language from this simple exercise.

Enter code into the .NET Editor

The .NET Editor and output console provide a great in-browser experience that's perfect for this tutorial approach. The .NET Editor is located on the right-hand side of this web page. The output console is below it.

1. Enter this code exactly as it appears into the .NET Editor on the right:

   C#

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

   You'll see an explanation of how and why it works soon. But first, you should experience it running, and ensure you entered it correctly. To do that, you'll run your code.

   Note

   You might be tempted to select `Copy` or `Run` and skip all the keystrokes. However, there are benefits to typing code yourself. Entering the code yourself reinforces memory and understanding that will help you gain insights that you wouldn't get otherwise.

Run your first code

1. Press the green Run button

   The green Run button performs two tasks:

   - It compiles your code into an executable format that a computer can understand.
   - It runs your compiled application and, when written correctly, will output `"Hello World!"`.

Observe your results

1. In the output console, observe the result of your code. You should get the following output:

   Output

   ```
   Hello World!
   ```

What to do if you get an error message

Writing C# code is an *exercise in precision*. If you type just one character incorrectly, you'll get an error message in the output area when you run the code.

For example, if you were to incorrectly enter a lower-case `c` in the word `console` like so:

C#

```csharp
console.WriteLine("Hello World!");
```

You'd get the following error message:

Output

```
(1,1): error CS0103: The name 'console' does not exist in the current context
```

The first part `(1,1)` indicates the line and column where the error occurred. But what does this error message mean?

C# is a case-sensitive language, meaning that the C# compiler considers the words `console` and `Console` as different as the words `cat` and `dog`. Sometimes, the error message can be a bit misleading. You'll need to understand the true reason why the error exists, and that comes through learning more about C#'s syntax.

Similarly, if you used single-quotation marks (`'`) to surround the literal string `Hello World!` like so:

C#

```
Console.WriteLine('Hello World!');
```

You would get the following error message:

Output

```
(1,19): error CS1012: Too many characters in character literal
```

Again, in line 1, character 19 points to the culprit. You can use the message as a clue as you investigate the problem. But what does the error message mean? What exactly is a "character literal?" Later, you'll learn more about literals of various data types (including character literals). For now, be careful when you're entering code.

Fortunately, errors are never permanent. You merely spot the error, fix it, and rerun your code.

If you got an error when you ran your code, take a moment to look at it closely. Examine each character and make sure you entered this line of code exactly.

Note

The code editor is constantly monitoring the code you write by performing pre-compilation to find potential errors. It will try to help you by adding red squiggly lines underlining the code that will produce an error.

Common mistakes new programmers make:

- Entering lower-case letters instead of capitalizing `C` in `Console`, or the letters `W` or `L` in `WriteLine`.
- Entering a comma instead of a period between `Console` and `WriteLine`.
- Forgetting to use double-quotation marks, or using single-quotation marks to surround the phrase `Hello World!`.
- Forgetting a semi-colon at the end of the command.

Each of these mistakes prevents your code from compiling successfully.

The code editor highlights pre-compilation errors to help you easily identify and correct mistakes as you develop your code. You can think of it like a spell-checker that helps you fix grammar or spelling errors in a document.

Assuming you were successful in the previous steps, let's continue.

Display a new message

In this task, you'll comment out the previous line of code, then add new lines of code in the .NET Editor to print a new message

1. Modify the code you wrote so that it's prefixed by a code comment using two forward slashes `//`:

   C#

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

   You can create a code comment by prefixing a line of code with two forward slashes `//`. This prefix instructs the compiler to ignore all the instructions on that line.

   Code comments are helpful when you're not ready to delete the code yet, but you want to ignore it for now. You can also use code comments to add messages to yourself or others who may later read the code, reminding you of what the code is doing.

2. Add new lines of code to match the following code snippet:

   C#

   ```
   Console.Write("Congratulations!");
   Console.Write(" ");
   Console.Write("You wrote your first lines of code.");
   ```

3. Press the green Run button again. This time, you should get the following output.

   Output

   ```
   Congratulations! You wrote your first lines of code.
   ```

The difference between Console.Write and Console.WriteLine

The three new lines of code you added demonstrated the difference between the Console.WriteLine() and Console.Write methods.

To print an entire message to the output console, you used the first technique, `Console.WriteLine()`. At the end of the line, it added a line feed similar to how to create a new line of text by pressing Enter or Return.

To print to the output console, but without adding a line feed at the end, you used the second technique, `Console.Write()`. So, the next call to `Console.Write()` prints another message to the same line.

Congratulations on writing your first lines of code!

Module complete:

Unlock achievement

.NET Editor

Press CTRL + M, TAB to exit the editor

Clear Run

Output