

Exercise - "Hello World!"

5 minutes

In this first hands-on exercise, you'll use C# to print a hallowed programmer's phrase to the Output window.

Write your first line of code



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

Step 1: Enter the code into the .NET Editor

The .NET Editor and output pane 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 pane is below it.

Later, you'll graduate to use either Visual Studio Code, or the full Visual Studio IDE to build larger and more interesting code examples--even entire applications.

Enter this code into the .NET Editor on the right:

C#	 Copy	 Run
<pre>Console.WriteLine("Hello World!");</pre>		

We'll explain how and why it works soon. But first, you should see it running, and ensure you didn't enter it incorrectly. To do that, you'll run your code.

ⓘ Note

You might be tempted to select Copy or Run and skip all the keystrokes. However, we encourage you to enter this line of code yourself. Entering the code yourself builds muscle memory, and helps you gain insights that you wouldn't get otherwise.


Step 2: 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 outputs the desired command.

Step 3: In the output window, observe the result



You should see the following output.

Output	 Copy
Hello World!	


What to do if you see an error message

Writing C# code is an *exercise in precision*. If you enter just one character incorrectly, you'll see 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#	 Copy	 Run
<pre>console.WriteLine("Hello World!");</pre>		

You would see the following error message:

Output	 Copy
<pre>(1,1): error CS0103: The name 'console' does not exist in the current context</pre>	

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#

 Copy Run

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

You would see the following error message:

Output

 Copy

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

Again, line 1, character 19 points us 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"? While we'll describe literals of various data types (including character literals) in another module, our advice for now is to be careful when you're entering code.

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

If you got an error when you ran your code, take a moment, look closely, and 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 beneath 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 double-quotation marks, or using single-quotation 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.


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

Step 4: Comment out the previous line of code, then add new lines of code in the .NET Editor to print a new message

Modify the code you wrote so that it's prefixed by a code comment using two forward slashes `//` , and add new lines of code to match the following code snippet:



C#	 Copy	 Run
<pre>// Console.WriteLine("Hello World!"); Console.Write("Congratulations!"); Console.Write(" "); Console.Write("You wrote your first lines of code!");</pre>		

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

Output	 Copy
<pre>Congratulations! You wrote your first lines of code!</pre>	

You create a code comment by prefixing a line of code with a two forward slashes `//` . This 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 that reminds you what the code is doing. We'll talk about code comments in an upcoming module.

The three new lines of code you added demonstrated the difference between the `Console.WriteLine()` and `Console.Write()` methods. Admittedly, the new lines of code you added could easily be shortened to a single line of code like you wrote in step 1. However, you wanted to learn a second technique to print a message to the output console.

To print an entire message to the output console, the first technique used `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 at the end, not adding a line feed, the second technique used `Console.Write()` . So, the next call to `Console.Write()` prints an additional message to the same line.

Beyond two techniques for printing messages to the output console, there's more to learn from this exercise. What do all of these words and symbols mean?

Next unit: How it works

[Continue >](#)

How are we doing? ☆ ☆ ☆ ☆ ☆

🌐 English (United States)

☀ Theme

[Previous Version Docs](#)

[Blog](#)

[Contribute](#)

[Privacy & Cookies](#)

[Terms of Use](#)

[Trademarks](#)

© Microsoft 2022

.NET Editor

Press **CTRL** + **M**, **TAB** to exit the editor



```
1 //Console.WriteLine("Hellow World!");  
2  
3 Console.Write("Congratulations!");  
4 Console.Write(" ");  
5 Console.Write("You wrote your fist line")
```

Output



Congratulations! You wrote your fist lines

of code!