# Exercise - Literal values

5 minutes

In this exercise, you'll print out messages containing other types of data and learn why data types are so important in C#.

## What is a literal value?

A literal value is a hard-coded value that never changes. Previously, we displayed a literal string to the Output pane. In other words, we literally wanted that string of alphanumeric characters `H`, `e`, `l`, `l`, `o`, and so on, displayed in the Output window.

The string data type is used whenever you have alphanumeric words, phrases, or data for presentation, not calculation. More about that in a moment.

What other kinds of literal data can we print to the Output pane?

## Exercise - Print out different literal data types

There are many data types in C#. But as you're getting started you only need to know about five or six data types since they cover most scenarios. Let's display a literal instance of data type to the output pane.

> **Note**
>
> You may notice as you begin to work in the code window that it colors certain syntax in different colors to indicate keywords, operators, data types and more. Begin to take notice of the colors. It can help you spot syntax errors as you enter characters, and can help you understand the code more effectively.

### Step 1: Write a char literal to the console

If we only wanted a single alphanumeric character printed to screen, we could create a **char literal** by surrounding one alphanumeric character in single-quotes.

Add the following line of code in the code editor:

C# Copy Run

```
Console.WriteLine('b');
```

If you enter the following code:

C# Copy Run

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

You would get the following error:

Output Copy

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

The C# compiler was expecting a single character (since you used the character literal syntax) but you supplied 12 characters instead!

Just like the string data type, you use char whenever you have a single alphanumeric character for presentation (not calculation).

The term *char* is short for *character*. In C#, they're officially referred to as "char", but frequently referred to as a "character".

## Step 2: Write an int literal to the console

If you want to print a numeric whole number (no fractions) value to Output, you can use an **int literal**. An `int` literal requires no additional operators like the `string` or `char`.

Add the following line of code in the code editor:

C# Copy Run

```
Console.WriteLine(123);
```

If you run the code, you'll get the following output.

Output Copy

```
123
```

The term *int* is short for integer, which you may recognize from studying math. In C#, they're officially referred to as "int", but frequently known by their alter ego "integer".

## Step 3: Write a decimal literal to the console

If we wanted to print a number that includes values after the decimal point, we could use a **decimal literal**.

To create a decimal literal, append the letter `m` after the number. In this context, the `m` is called a *literal suffix*. The literal suffix tells the compiler you wish to work with a value of type decimal.

Add the following line of code in the code editor:

C# Copy Run

```
Console.WriteLine(12.3m);
```

If you run the code, you'll get the following output.

Output Copy

```
12.3
```

> **Note**
>
> You can use either a lower-case `m` or upper-case `M` as the literal suffix for a decimal.

## Step 4: Write a bool literal to the console

If we wanted to print a value representing either `true` or `false`, we could use a **bool literal**.

Add the following lines of code in the code editor:

C# Copy Run

```
Console.WriteLine(true);
Console.WriteLine(false);
```

This will produce the following output:

Output Copy

```
True
False
```

The term *bool* is short for *boolean*, which you may also recognize from studying math. In C#, they're officially referred to as "bool", but often developers use the term "boolean".

The bool literals represent the idea of truth and falsehood. We'll use bool values extensively when we start to add decision logic to our applications. We'll evaluate expressions to see whether the expression is true or false.

## Why emphasize data types?

Data types play a central role in C#. In fact, the emphasis on data types is one of the key distinguishing features of C# compared to other languages like Python and JavaScript. The designers of C# believed they can help developers avoid common software bugs by *enforcing* data types. You'll see this concept unfold as you learn more about C#.

## Presentation versus calculation and evaluation

Earlier we said that `string`s and `char`s are used for "presentation, not calculation". If you need to perform a mathematical operation on numeric values, you should use an `int` or `decimal`. If you have data that is used for presentation or reference purposes only, you should use a `string` or `char` data type.

Suppose you needed to collect data from a user like a phone number or postal code. Depending on the country where you live, that data may consist of numeric characters. However, since you rarely perform mathematical calculations on phone numbers and postal codes, you should prefer to use a `string` data type when working with them.

The same can be said of `bool`. If you need to work with the words `"true"` and `"false"` in your application, you would use a `string`. However, if you need to work with the concept of `true` or `false` when performing an evaluation, you use a `bool`. This should become clearer as we perform evaluations in other modules.

It's important to know that these values may look like their string literal equivalents. In other words, you may think these statements are the same:

C# Copy Run

```
Console.WriteLine("123");
Console.WriteLine(123);
```

```
Console.WriteLine("true");
Console.WriteLine(true);
```

However, that's merely how they're printed to screen. The fact is that the kinds of things you can do with the underlying `int` or `bool` will be different than their `string` equivalent.

## Recap

The main takeaway is that there are many data types, but we'll focus on just a few for now:

- string for words, phrases, or any alphanumeric data for presentation, not calculation
- char for a single alphanumeric character
- int for a whole number
- decimal for a number with a decimal
- bool for a true/false value

## Next unit: Declare variables

Continue >

How are we doing? ☆ ☆ ☆ ☆ ☆

English (United States) Theme

.NET Editor

Press CTRL+M, TAB to exit the editor

Clear Run

```
1
```

Output