Interacting with the console

Interacting with the console

- Computers, and thus software, are good at accepting input, processing it, and producing new information.
- The applications we write in this program will certainly fit that pattern.
- With our first programs we'd like to interact with the user directly in our shell.

Example

Sending information to the console

- Interactions with the console use the `Console` class. In a different lesson we will cover classes. For now know that a class can be a collection of code.
- We will use a few different statements:
  - **`Console.WriteLine()`**
  - **`Console.Write()`**
  - **`Console.ReadLine()`**

Outputting to the console one line at a time

```
Console.WriteLine("Welcome to my program");
```

This will output one line to the console and move the cursor to the next line. Further output will be on its own line.

Outputting a blank line

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

Output text but do not move the cursor to the next line

To leave the cursor on the **same** line as the text use **`Console.Write()`**.

In this case the cursor remains in place after the last character of our output.
In this case we leave a blank space.

```
Console.Write("What is your name? ");
```

Reading information from the console

To read information we use **`Console.ReadLine()`**

This pauses our program for the user to type in text and press **`ENTER`**.

Then assigns the answer to a **`string`** variable we place on the left-hand-side.

```
var name = Console.ReadLine();
```

Outputting information containing a variable

First print out the fixed text

But don't move the cursor to the next line yet

```
Console.Write("It is a pleasure to meet you,");
Console.WriteLine(name);
```

Then print out the variable

And move the cursor to the next line

```
Console.Write("It is a pleasure to meet you,");
Console.WriteLine(name);
```

We can do better

Strings can be added together!

```
var greeting = "It is a pleasure to meet you, " + name;
Console.WriteLine(greeting);
```

Avoid writing the variable

Since we create the **`greeting`** variable just to use it once we can get rid of it.

```
Console.WriteLine("It is a pleasure to meet you, " + name);
```

Avoid the + string addition

This pattern of using a variable along with a string is so common C# gives us a way to do this right inside the string!

String interpolation

If we put a **$** before our first *double quote* the string gains **magic powers**.

String interpolation

```
$"It is a pleasure to meet you, {name}"
```

Inside a string with the **$** powers we can place { } inside the string. Anything inside the string is considered code.

Whatever that code *evaluates* to is placed at that spot in the string!

```
$"My favorite number is {6 * 7}";
```

Using string interpolation

```
Console.WriteLine($"It is a pleasure to meet you, {name}");
```

Full program

```
using System;

namespace OurDotNetApp
{
  class Program
  {
    static void Main(string[] args)
    {
      Console.WriteLine("Welcome to my program");
      Console.WriteLine("\n");

      Console.Write("What is your name? ");
      var name = Console.ReadLine();

      Console.WriteLine($"It is a pleasure to meet you, {name}");
    }
  }
}
```