Tutorial – Variables:
Using Variables

In this tutorial we're going to discuss variables. What they are, how they work and how we can use them to build games.

What Are Variables:

A variable is a container that stores information within our code. Variables can store different types of information such as numbers, text or a true or false state, but they can only store one type of information at a time. A variable could store either a number or text, but not both at once.

What Do We Use Them For?

We use variables to keep track of different things in our game. For example, if our game has a score, we will use a variable to store that number.

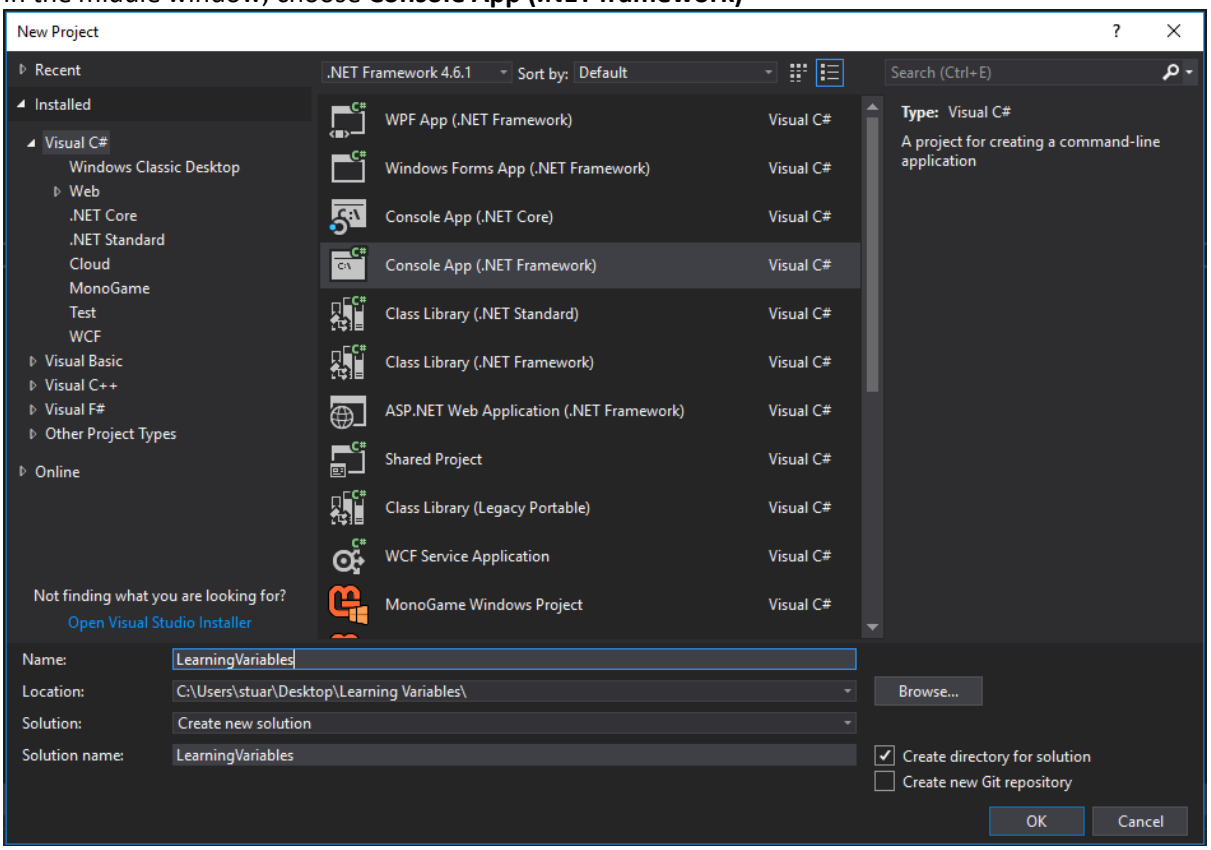
Much of programming is about cleverly manipulating variables to create interesting and desired results. For example, in a typical RPG, players gain experience points to grow levels. In that case, we would have at least two variables: a variable to store our players current experience points, and a variable to store his current level.

Then, during the game, we continue to increase the experience points of the player and check if it's high enough for the player to grow another level. When it is, we increase the players level variable. This is one example of manipulating variables to create a game experience.

Create A New Project:

Open up Visual Studio and create a new project. Just to refresh you:

1. Open Visual Studio
2. Click **File > New > Project**
3. On the left window, select **Visual C#**
4. In the middle window, choose **Console App (.NET framework)**



5. Call the projects name **LearningVariables**
6. Set the **Location** to where you want to store it on your PC
7. Click **OK**

This will create you a new blank project for your next program.

How We Use Them:

Here is a list of different types of variables:

Variable Type	Description
int	Integer, or whole number. Cannot have a decimal value.
float	Floating point number. A number that can have decimal values.
double	Similar to a float: it stores numbers with decimal values, but can have a higher number of decimal values
boolean	True or false
string	A variable that stores text
char	A variable that stores a single character of text, ie **'A'** or **'a'** or **'1'**

We call creating a variable **declaring** it. To declare a variable, you need to say what type of variable it is you're creating, then give the variable a name so you can refer to it in your code later. Like this:

```
int score;
```

Scores in video games are numbers, generally without decimals, so we declare a variable of the type **int.** Look at the table above: **int's** are whole numbers with no decimal.

Creating Your First Variable

Let's add some code and make a new variable. First, let's remove some code:

```
    class Program
    {
        ~~static void Main(string[] args)~~
        static void Main()
        {
        }
    }
```

Remember that green is code to add, and a ~~line through~~ means code to remove.

We don't need to keep the **string[] args** in the **Main** function so we'll just remove it. Now, let's create a variable!

```
class Program
{
    static void Main()
    {
        string greeting;
        int playerScore;
    }
}
```

We declare a **string** variable called **greeting.** String's are text, often used to display text or store data in a text format, like storing people's names.

We also declare a variable of the type **int** called **playerScore,** to store a number without decimals**.**

Notice how the name **playerScore** starts with a lower case **p,** and then the second word starts with a capital **S**. We always begin variable names with a lower case letter, and if there are multiple words as part of the name, every other word starts with a capital. This is called **Camel Case,** and it makes code easier to read.

Let's Make Our Variables Do Something

When you create a variable, it doesn't have to have a value – it can be empty, which is what our variables currently are. But you can also create them with a set value. Let's do that now:

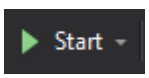
```
static void Main()
{
    string greeting = "Hello, user!";
    int playerScore = 0;
}
```

You can see that our our greeting has stored actual text, **Hello, user!** and our **playerScore** is 0. We can now view them in the console.

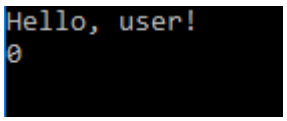
```
static void Main()
{
    string greeting = "Hello, user!";
    int playerScore = 0;

    Console.WriteLine(greeting);
    Console.WriteLine(playerScore);
    Console.ReadKey();
}
```

Now test your program by clicking the **Start** button.



You should see your variables printed to the Console.



Console.WriteLine () will print out whatever we put between the brackets, which are called **parentheses,** then start a new line. If we name our variables, it will print the value stored inside the **variables**.

Console.ReadKey pauses our application until you press the **Enter** key.

Try adding this:

```
        static void Main()
        {
            string greeting = "Hello, user!";
            int playerScore = 0;
            string displayScore = "Your score currently is " + playerScore + ".";

            Console.WriteLine(greeting);
            Console.WriteLine(displayScore);
            Console.ReadKey();

        }
```

Test your program and see what happens.

We create a **string** called **displayScore**. We put some text in the quotation marks, but then we add the **int** variable **playerScore** so that we can display the number as text. When you print an **int** as text, you can't do maths on it, but you can see it's value. We then add a full stop. You can see the result in the console.

Changing Variables After You Make Them

We've looked at how you create variables, and some of the ways you can use them. The last lesson to teach you right now is how you can change the value of a variable after you've created it. Remove all of the code inside the **Main** function you wrote previously and try this:

```
        static void Main()
        {
            string greeting = "Hello, user! The game's start condition is: ";
            bool gameStarted = false;
            string displayGameState = greeting + gameStarted;
            Console.WriteLine(displayGameState);

            gameStarted = true;
            displayGameState = greeting + gameStarted;
            Console.WriteLine(displayGameState);
            Console.ReadKey();
        }
```

Test your program.

We add a **bool** type variable, which can either be **true** or **false**.
We create a new string which is a combination of the **greeting** and **gameStarted**.
Then we print out the new string**.** It tells us the **bool** is false.
Then, on the next line we name the **bool** again, **gameStarted**, and change it's value to **true.**
We also then update the **displayGameState** string. This is important, which you'll see next page.

When you place a variable on the **left side** of an **=** sign, it will take the value of whatever is on the **right side** of an = sign.

Look what happens when we do this:

```
        static void Main()
         {
             string greeting = "Hello, user! The game's start condition is: ";
             bool gameStarted = false;
             string displayGameState = greeting + gameStarted;
             Console.WriteLine(displayGameState);

             gameStarted = true;
             displayGameState = greeting + gameStarted;
             Console.WriteLine(displayGameState);
             Console.ReadKey();
         }
```

Take out the line of code with a line through it. Now test your program.

It prints the **bool** as **false** twice. Why? We updated the **bool** variable to be true, didn't we?

It's important for you to understand that all code is read from top to bottom, left to right. When you create the string **displayGameState**, we tell it to display the value of the variable **gameStarted** at that current point in time, which happens to be **false.**

If the **bool gameStarted** ever changes, you have to update **displayGameState** again otherwise the string won't show the correct values. It's like a snapshot in time. This is because of the way that code is read from top to bottom. It's a very important lesson for you to learn and remember for the future when you are trying to solve programming problems.

Challenge:

Experiment creating some different variables, changing their values and displaying them. This will help you become comfortable with variables.

Later we will show you more things you can do with them.