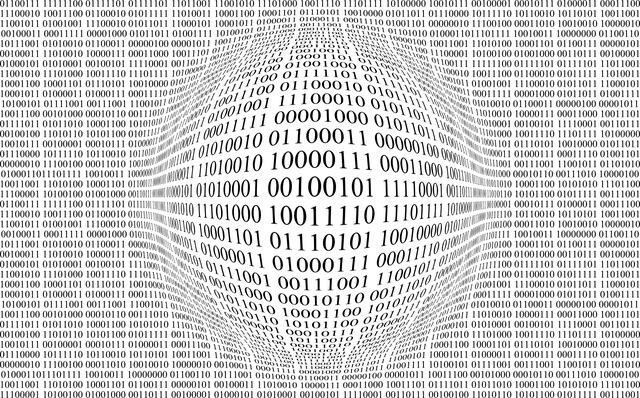
Sharp Q’s: Learning C# Through Conway’s Game of Life Part 1



Contents

[Lesson 0 – The Basics 4](#_Toc513458727)

[Programming Languages 4](#_Toc513458728)

[Markup Languages 4](#_Toc513458729)

[How to Succeed In This Course 4](#_Toc513458730)

[Resources to Use in This Course 4](#_Toc513458731)

[Course Breakdown 5](#_Toc513458732)

[Lesson 1 – Code Repositories 5](#_Toc513458733)

[What is a code repository? 5](#_Toc513458734)

[Further Reading 5](#_Toc513458735)

[Practice 5](#_Toc513458736)

[Lesson 2 – Forking the Conway’s Game of Life Project – CAPSTONE 7](#_Toc513458737)

[What Is a Fork? 7](#_Toc513458738)

[Practice 7](#_Toc513458739)

[Further Reading 8](#_Toc513458740)

[Lesson 3 – Cloning a Repo to Your Local Machine, and Navigating Visual Studio 8](#_Toc513458741)

[What You Need To Begin 8](#_Toc513458742)

[Cloning the Repo to Your Local Machine 8](#_Toc513458743)

[Navigating Visual Studio 12](#_Toc513458744)

[Lesson 4 – Data Types 12](#_Toc513458745)

[What Are Data Types 12](#_Toc513458746)

[Common C# Data Types 13](#_Toc513458747)

[Byte 13](#_Toc513458748)

[Int 13](#_Toc513458749)

[Double 13](#_Toc513458750)

[Char 13](#_Toc513458751)

[Bool 13](#_Toc513458752)

[String 13](#_Toc513458753)

[Further Learning 13](#_Toc513458754)

[Lesson 5 – Variables 13](#_Toc513458755)

[What Is a Variable 13](#_Toc513458756)

[Further Learning 14](#_Toc513458757)

[Lesson 6 – Working with the String Data Type 14](#_Toc513458758)

[What Is a String 14](#_Toc513458759)

[Further Learning 14](#_Toc513458760)

[Lesson 7 – Writing Our First Real Program in C# 14](#_Toc513458761)

[What You Need To Begin 14](#_Toc513458762)

[Practice - Hello World! 15](#_Toc513458763)

[Lesson 8 – User Input to the Console 15](#_Toc513458764)

[What You Need To Begin 15](#_Toc513458765)

[System.Console Class 15](#_Toc513458766)

[Console.WriteLine 16](#_Toc513458767)

[Console.ReadLine 16](#_Toc513458768)

[Creating Strings with Variables 16](#_Toc513458769)

[String Concatenation 16](#_Toc513458770)

[String.Format 17](#_Toc513458771)

[String Interpolation 17](#_Toc513458772)

[Practice 18](#_Toc513458773)

[Further Learning 18](#_Toc513458774)

[Lesson 9 – Creating a New Solution – CAPSTONE 18](#_Toc513458775)

[What You Need To Begin 18](#_Toc513458776)

[Creating a New Solution 18](#_Toc513458777)

[Further Learning 20](#_Toc513458778)

[Lesson 10 – Working with Integers 20](#_Toc513458779)

[What Is an Integer? 20](#_Toc513458780)

[Practice 20](#_Toc513458781)

[Further Learning 21](#_Toc513458782)

[Lesson 11 – The Start of Conway's Game of Life – CAPSTONE 21](#_Toc513458783)

[What You Need To Begin 21](#_Toc513458784)

[First Things First 21](#_Toc513458785)

# Lesson 0 – The Basics

## Programming Languages

First off, what is a programming language? Britannica defines a computer programming language as “any of various languages for expressing a set of detailed instructions for a digital computer.” To make that simple, a programming language makes the computer do what we want. The top programming languages according to the TIOBE Index as of April 2018[[1]](https://www.tiobe.com/tiobe-index/) are Java, C, C++, Python, and C#.

Al says:

This is not necessarily a “true” rank of popularity, only search engine queries pertaining to a language.

## Markup Languages

You will notice that languages such as HTML do not appear on this list. That is because HTML and other such languages are *markup* languages. They do not perform any calculation of data, only specify how to display the data. HTML and LaTeX are two popular markup languages.

## How to Succeed In This Course

This course is at your own pace. There are no “tests”, no “homework”, no one who is going to tell you to do this course. Your success is solely dependent on your ability to read the material, study the concepts presented, and use the resources available to you to further learn the concepts. I cannot stress enough that this is at your own pace. Some may take 2 weeks to finish this course, some may take months, each person learns differently. I encourage you to ask your peers questions, but look online for answers first. There is perhaps no skill more valuable to a programmer than the ability to use the internet to find answers when they are stuck.

## Resources to Use in This Course

<https://www.youtube.com/>

<http://www.dreamincode.net/>

<https://www.stackoverflow.com/>

<https://www.bing.com/>

When performing a Bing or Google search, try to format your query with the language in front. For example, if you are searching for “Writing to the console”, format your search “C# writing to the console”.

## Course Breakdown

Each lesson in this course will cover a few topics. There will be some light reading, followed by some links you can use to further discover the topic. Finally, there will be a practice porject for some lessons, which will allow you to apply your knowledge. Some lessons will be appended with “— CAPSTONE”. These lessons are extra important as they will be implementing a new piece of our Game of Life, which is the capstone project for this course.

# Lesson 1 – Code Repositories

## What is a code repository?

A code repository (or repo) is one of the most important pieces of software development. Code repositories allow you to store your code away from your local machine, generally in the cloud or on a private server, to protect your work against things like drive failure. Some common code repositories are Git, TFS, SVN, and external drives, such as USB drives. Most professional repo’s also have some implementation of version control, which allows you to revert your code to an earlier version. For this course, we will be using Git, by way of Github.com[[2]](https://github.com/). Take a moment to familiarize yourself with Github using the references provided, and any you find on your own.

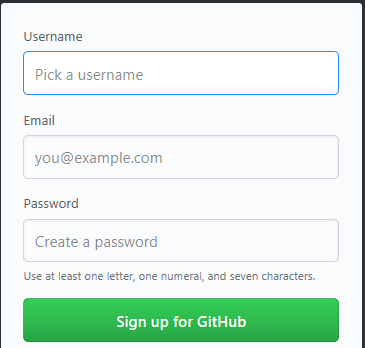
## Further Reading

<https://www.bing.com/search?q=what%20is%20github>

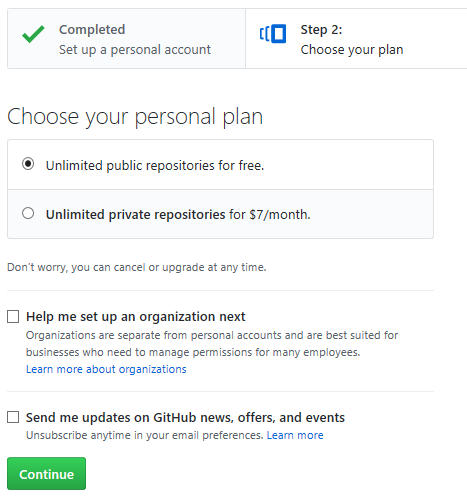
<https://www.bing.com/videos/search?q=navigating+github&view=detail&mid=B08BD7CA979C4858CED4B08BD7CA979C4858CED4&FORM=VIRE>

## Practice

Let’s create a Github account. Head over to Github.com, and follow the instructions on the homepage.

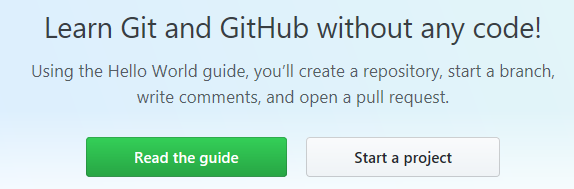


Next, we are going to leave the settings as shown below



You may change the plan if you wish, but for this course it is not necessary. Click continue when ready.

Finally, you can choose to input information in this step, or just choose to skip it. When you are done, you will be presented with the following screen:



Let’s go ahead and click on Read the guide. Follow the guide, which will give you the information needed for Lesson 2!

# Lesson 2 – Forking the Conway’s Game of Life Project – CAPSTONE

## What Is a Fork?

GitHub Says:

“A fork is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project.

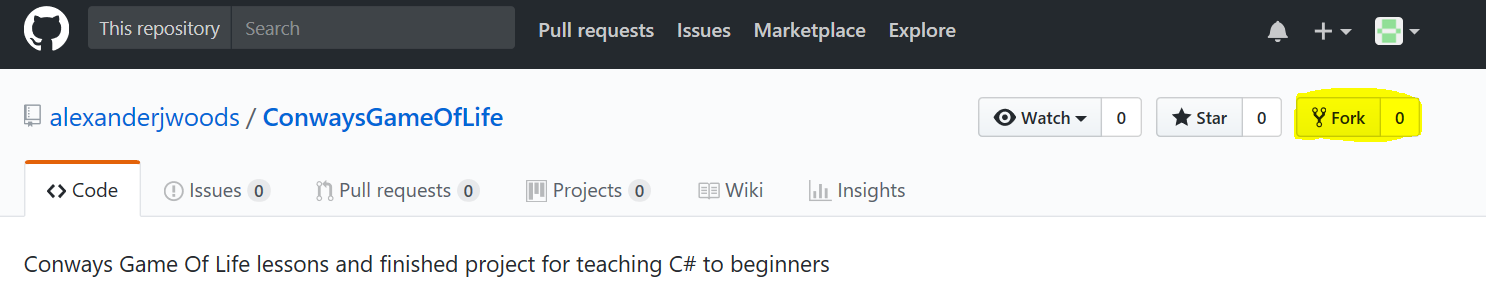
Most commonly, forks are used to either propose changes to someone else's project or to use someone else's project as a starting point for your own idea.”[[4]](https://help.github.com/articles/fork-a-repo/)

## Practice

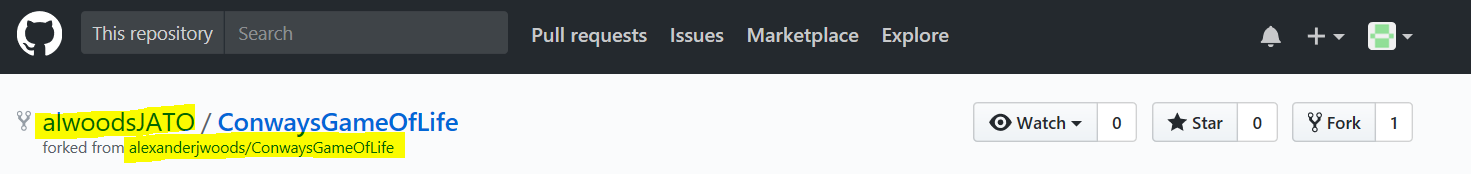
Log into the GitHub account you created in Lesson 1.

Navigate to <https://github.com/alexanderjwoods/ConwaysGameOfLife>

Click the Fork button (highlighted) to fork the repo



After the forking process completes, you will see that the repo has now been copied to your account, and says forked from alexanderjwoods/ConwaysGameOfLife



## Further Reading

<https://www.youtube.com/watch?v=f5grYMXbAV0>

<https://help.github.com/articles/fork-a-repo/>

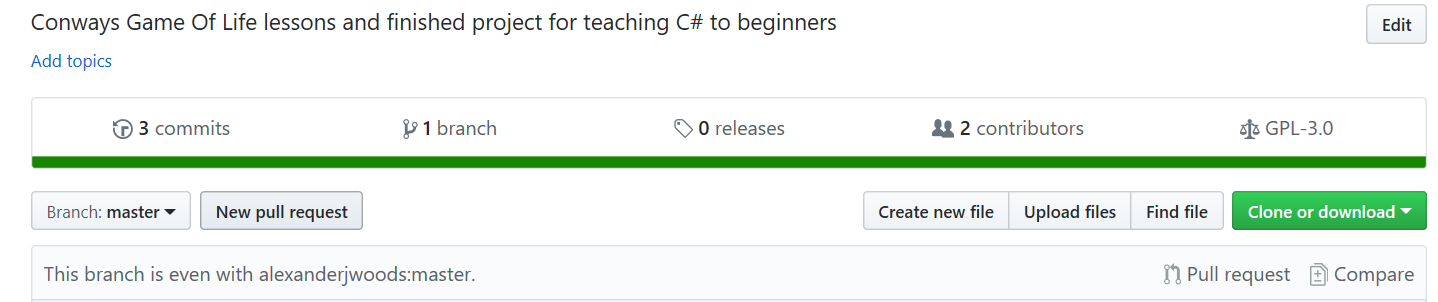
# Lesson 3 – Cloning a Repo to Your Local Machine, and Navigating Visual Studio

## What You Need To Begin

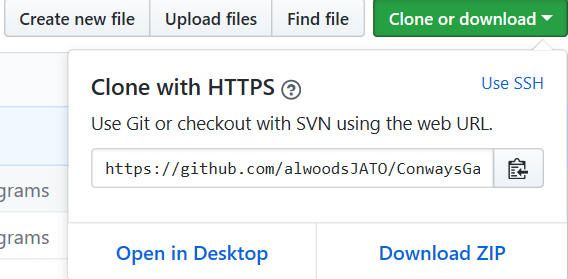
You should have a copy of Visual Studio installed, as well as forked the ConwaysGameOfLife project from my GitHub (alexanderjwoods) to your own GitHub.

## Cloning the Repo to Your Local Machine

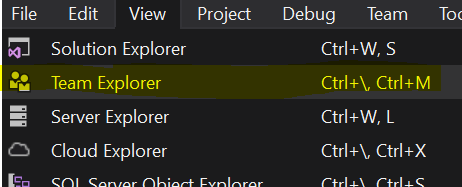
Navigate to the repo we forked in Lesson 2, and look for the green button that says “Clone or download”



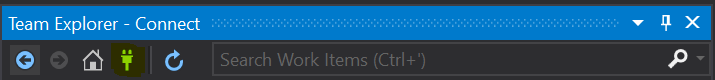
Clicking this will bring up the following options:



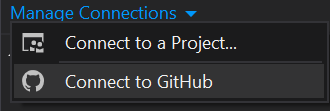
Copy the Url shown to your clipboard, and open Visual Studio. When it is open, go to View > Team Explorer.



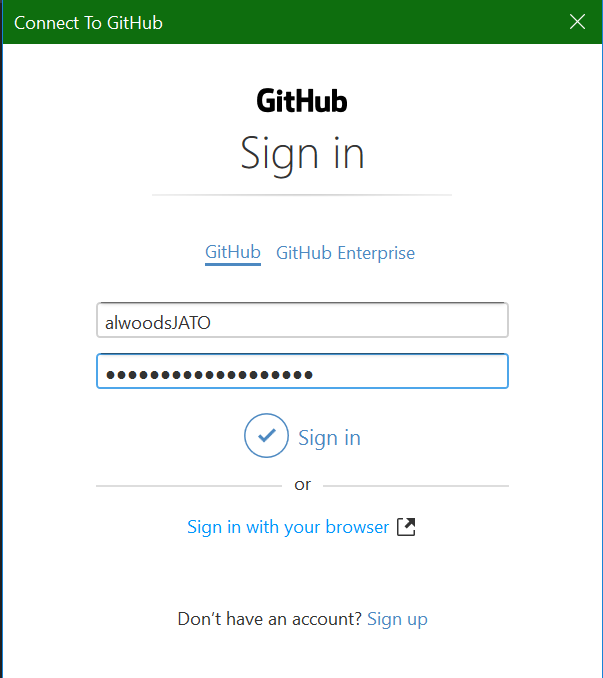
In the team explorer window, click the green button that looks like a plug, to the right of the icon that looks like a house:



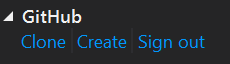
Choose Manage Connections, and Connect to GitHub



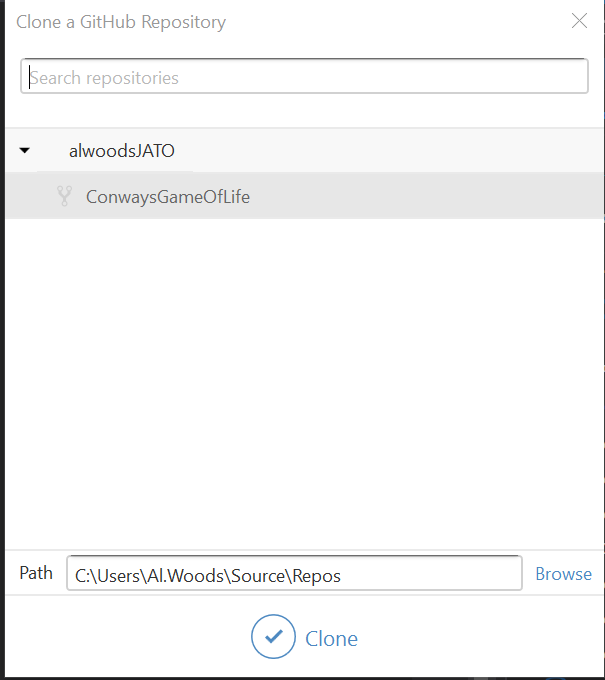
Enter your GitHub credentials to connect to GitHub



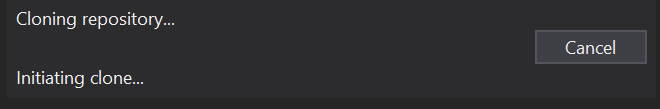
Now, Choose Clone under GitHub in the Team Explorer



Select ConwaysGameOfLife, enter a Path where the files will be stored, and press Clone.

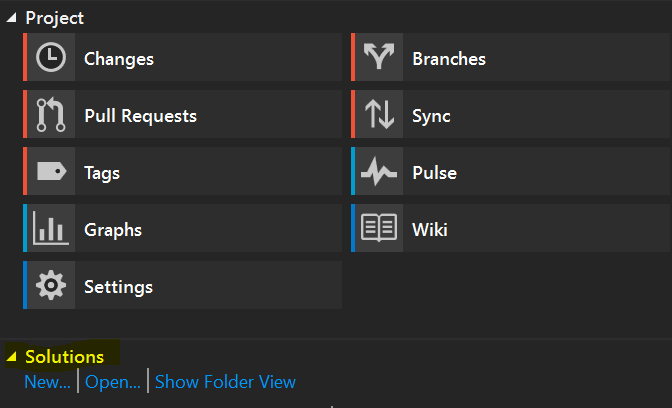


You will see this appear in the team explorer. Wait for this to finish and double click the cloned repository when it is finished.





This will take you to the Home tab of the Team Explorer, where you will find the most commonly used tools.



Some lesson will ask you to open projects, and you will find these in the projects inside your solution explorer. If you do not see your solution explorer, you can find it in the View menu. Remember, most programs are opened by double clicking the program.cs file.

## Navigating Visual Studio

For help with navigating Visual Studio, utilize the tools at our disposal. Youtube, Search Engines, StackOverflow, and your own curiosity! To get you started, watch this video: <https://www.youtube.com/watch?v=sBUXVM3kEa8>

# Lesson 4 – Data Types

## What Are Data Types

A data type is a classification of data which tells the compiler or interpreter how the programmer intends to use the data. For now, don’t worry with that a compiler or interpreter is, we will go over that shortly. The most common data types that you will find in programming languages are integers, Booleans, characters, floating point numbers, and alphanumeric strings.

## Common C# Data Types

### Byte

A byte is 8 bits of data, and can hold values from 0 to 255.

### Int

Int is short for integer, and holds 32 bits of data. This means the range for an int is -2,147,483,648 to 2,147,483,647. This data type is used to hold whole numbers, as it cannot handle numbers with decimals (that is what double, float, and decimal are for).

### Double

Double is a double-precision floating point integer. This datatype is similar to int, but can hold decimal values. It is 64 bits ranging from -1.79769313486232e308 to 1.79769313486232e308.

### Char

The char data type holds a single Unicode character.

### Bool

The bool data type stands for Boolean. It holds just two possible values, 1 and 0, which evaluate to either true or false. Generally, we only every deal with Bool’s as true or false, not as 1 or 0.

### String

A string is a 128 bit collection of alphanumeric characters.

## Further Learning

To learn more about data types:

<http://www.tutorialsteacher.com/csharp/csharp-data-types>

<https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/types>

<http://zetcode.com/lang/csharp/datatypes/>

<https://www.bing.com/search?q=c%23+data+types+list>

# Lesson 5 – Variables

## What Is a Variable

A variable is an object we use to refer to something else. Variables in c# are mutable, which means the can be changed at will without creating a copy. However, in c#, variables cannot change their data type. For example, I can change the value of a string variable from one string to another, but I cannot change the value of a string variable to an integer.

## Further Learning

<https://www.youtube.com/watch?v=5qQQ3yzbKp8>

<https://www.youtube.com/watch?v=udoMi4mGYYw>

# Lesson 6 – Working with the String Data Type

## What Is a String

A string is a data type that represents a series of alphanumeric Unicode characters. Strings can be contained in variables, or can be string literals. A string literal is *always* wrapped in “ “. For example:



Here, we define the type of the variable, then name the variable, then set it equal to a string literal.

## Further Learning

<https://www.youtube.com/watch?v=CLsmRBmteas&t=166s>

<https://www.youtube.com/watch?v=afY8t4Iv-nc>

<https://www.youtube.com/watch?v=UFDxqX57eH0>

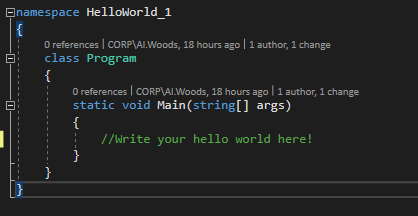
# Lesson 7 – Writing Our First Real Program in C#

## What You Need To Begin

Before beginning lesson 7, you should feel comfortable with navigating GitHub and Visual Studio. You should be comfortable working with strings as well.

## Practice - Hello World!

Hello World [[3]](https://en.wikipedia.org/wiki/%22Hello,_World!%22_program) is the first program that everyone creates when they learn a new language. It is a rite of passage in a sense! Open up Visual Studio, and open the “HelloWorld\_1” project. Opening the program.cs file will present you with the following screen:



Replace the comment “Write your hello world here!” with code that will display Hello World when run.

# Lesson 8 – User Input to the Console

## What You Need To Begin

You should be comfortable with using strings and understanding how variables work.

## System.Console Class

We are going to look at the System.Console class a bit more closely.

MSDN Says:

”Represents the standard input, output, and error streams for console applications. This class cannot be inherited.”[[5]](https://msdn.microsoft.com/en-us/library/system.console(v=vs.110).aspx)

### Console.WriteLine

In the last lesson, you should have written something like:

Console.WriteLine(“Hello World!”);

In this statement, we are using the WriteLine method, from the Console class, and passing in the string argument “Hello World”. Looking at the MSDN [[6]](https://msdn.microsoft.com/en-us/library/xf2k8ftb(v=vs.110).aspx) for WriteLine, we see it defined as:

*public static void WriteLine(string value)*

This tells us that the method WriteLine, takes a parameter of type string, and has no return type. The other two keywords mean that it is public, and can be accessed by anyone, and that it is static, which means we do not have to instantiate a class to use it. But don’t worry too much about public and static, we will go over access modifiers in a future chapter. For now, pay attention to the return type (void), the name (WriteLine), and the parameters (string value).

## Console.ReadLine

ReadLine is defined on MSDN [[7]](https://msdn.microsoft.com/en-us/library/system.console.readline(v=vs.110).aspx) as:

public static string ReadLine()

We can see here that ReadLine takes no parameters, but does return a string. It is also a public static method, so we can use it in a similar fashion to WriteLine. Reading the documentation on MSDN will also tell us that the string returned is whatever the user entered into the Console window once they press Enter.

## Creating Strings with Variables

### String Concatenation

String concatenation is a simple way to merge two strings together. It can be done with string variables, string literals, or a combination of both. Simply done using a ‘+’ sign.

string myString = “Hello ”;  
string concatenated = myString + “Al”;

The value of concatenated is now “Hello Al”.

### String.Format

Format is a static method of the String class, much like WriteLine and ReadLine.

MS Docs Says [[8]](https://docs.microsoft.com/en-us/dotnet/api/system.string.format?view=netframework-4.7.2):

“Converts the value of objects to strings based on the formats specified and inserts them into another string.”

For an in depth overview of how this works, check the [Get Started with the String.Format method](https://docs.microsoft.com/en-us/dotnet/api/system.string.format?view=netframework-4.7.2#Starting) section of MS Docs.

Example:

string concatenated = String.Format(“Hello {0}!”, “Al”);

### String Interpolation

String interpolation is similar to String.Format, except instead of using {0} to specify the variable to insert, you can directly call the variable.

MS Docs Says [[9]](https://docs.microsoft.com/en-us/dotnet/csharp/tutorials/string-interpolation):

“String Interpolation is the way that placeholders in a string are replaced by the value of a string variable”

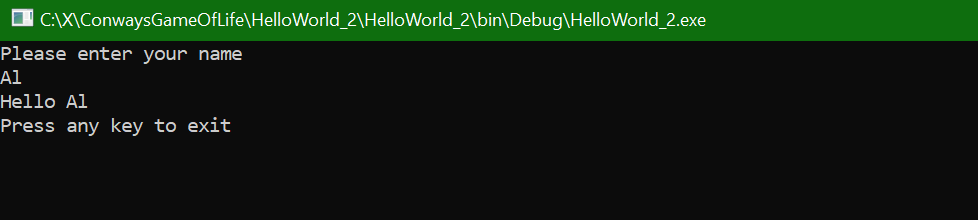
Example:

string name = “Al”;

string concatenated = $”Hello {name}”;

## Practice

Open the HelloWorld\_2 project, and modify it to ask the user for input, and displaying their name after Hello. For an example of the finished output see below:



## Further Learning

<https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/inside-a-program/hello-world-your-first-program>

<https://www.dotnetperls.com/console-readline>

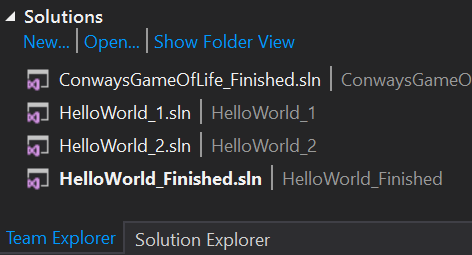
# Lesson 9 – Creating a New Solution – CAPSTONE

## What You Need To Begin

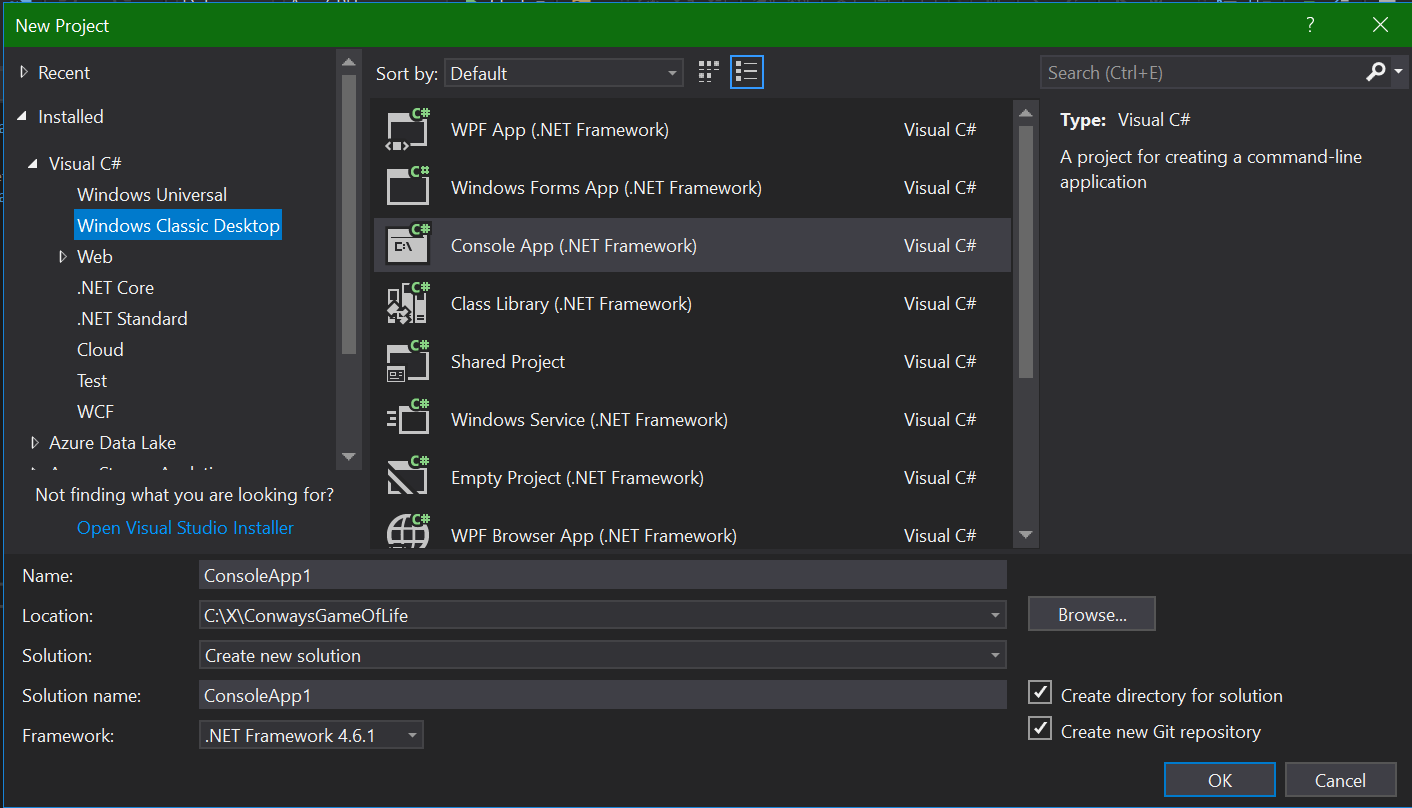
To begin, you should have a good understanding of strings, console input/output, and navigating Visual Studio.

## Creating a New Solution

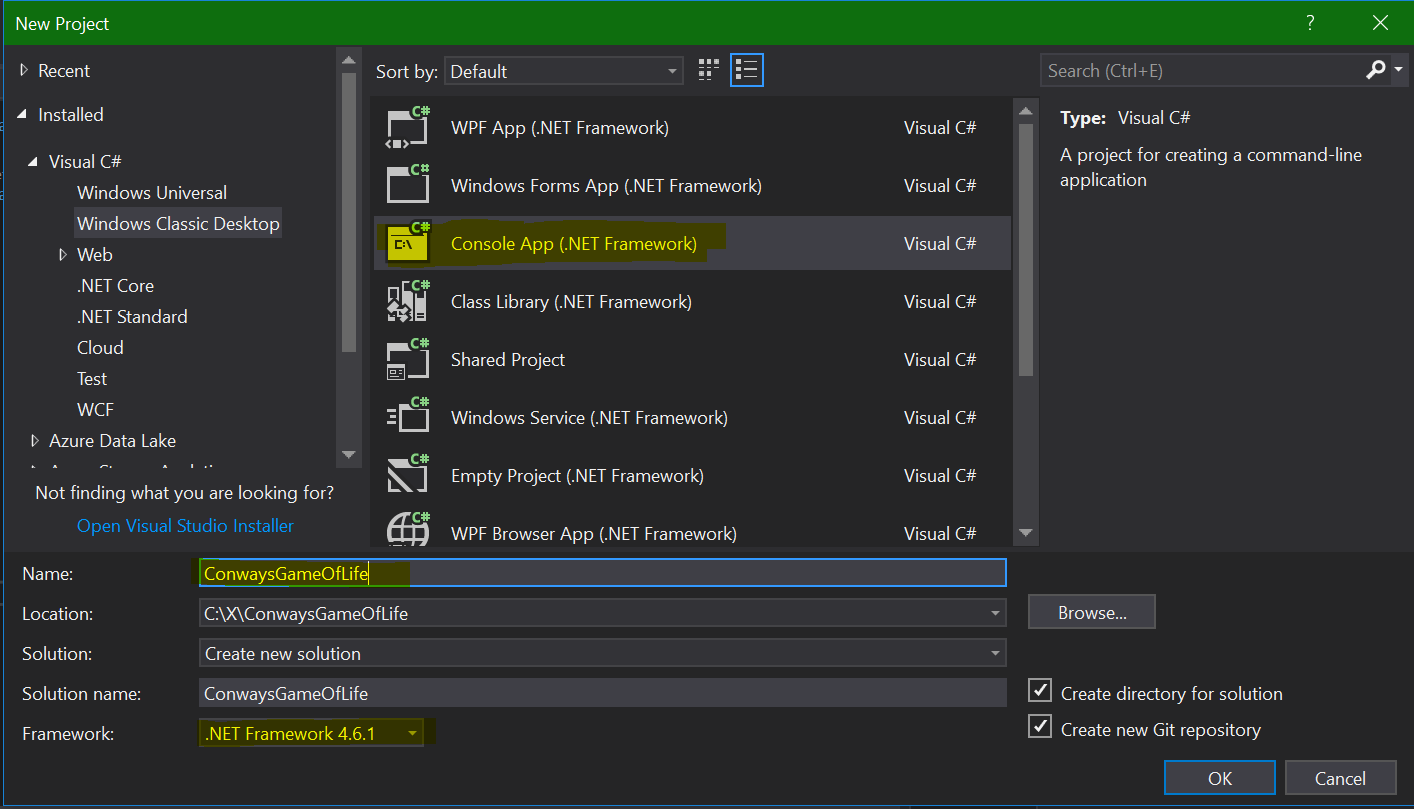
Creating a new solution in our project is simple. Navigate to your team explorer, and the solutions area.



\*NOTE: The solutions you see above may not reflect the solutions you have!

Click the New… button, to be presented with the following dialogue box:  


Change the Name to “ConwaysGameOfLife” (no spaces), and makes sure the solution type is Console App (.NET Framework). Finally, make sure the framework is 4.6.1!



Press OK, and the new Solution is created, opened, and added to the solutions list in the team explorer.

## Further Learning

<https://docs.microsoft.com/en-us/visualstudio/ide/creating-solutions-and-projects>

# Lesson 10 – Working with Integers

## What Is an Integer?

An Integer is simply a whole number, i.e., 1, 2, or 3. Integers cannot have decimal places.

## Practice

Open the project “HelloWorld\_3”, and add a second prompt for the user to enter their age, converting the string input into an integer (Hint: Look up Int32.Parse method) and displaying it to the user.

## Further Learning

<https://docs.microsoft.com/en-us/dotnet/csharp/quick-starts/numbers-in-csharp-local>

<https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/types/how-to-convert-a-string-to-a-number>

# Lesson 11 – The Start of Conway's Game of Life – CAPSTONE

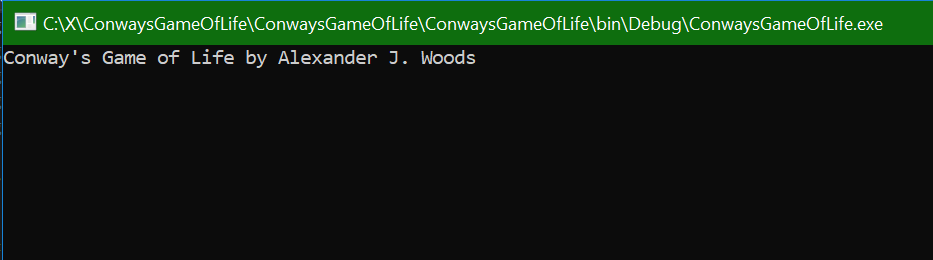
## What You Need To Begin

You should have a basic understanding of what Conway’s Game of Life is and how it works. If you don’t, open your favorite search engine and do some searching! You should also understand string input and output in c#.

## First Things First

Create a new Solution somewhere, called “ConwaysGameOfLife”.

The first thing every good program should have is an introduction. Let’s use our knowledge of strings to output to the Console “Conway’s Game of Life by [YOUR NAME HERE]”. This tells our user what program they are running (obviously)! That is all for now, but don’t worry, we will come back to it!



# Part 1 Knowledge Check

You may wish to take a quick, 8 question knowledge check on some of the concepts that we touched on in part 1. I recommend you do not advance to part 2 until you a confident you can score 8 correct on the testmoz test I have created for you found at <https://testmoz.com/1694612>!

References:

1. <https://www.tiobe.com/tiobe-index/>
2. <https://github.com/>
3. <https://en.wikipedia.org/wiki/%22Hello,_World!%22_program>
4. <https://help.github.com/articles/fork-a-repo/>
5. <https://msdn.microsoft.com/en-us/library/system.console(v=vs.110).aspx>
6. <https://msdn.microsoft.com/en-us/library/xf2k8ftb(v=vs.110).aspx>
7. <https://msdn.microsoft.com/en-us/library/system.console.readline(v=vs.110).aspx>
8. <https://docs.microsoft.com/en-us/dotnet/api/system.string.format?view=netframework-4.7.2>