Pluralsight: C# Fundamentals with Scott Allen

* If needed install enterprise (can use with python, c++, .NET etc.)

.NET  
- .NET **framework** (2001-currently) only on windows  
- .NET **core** works on multiple platforms! Newer version

# What is inside .NET

* CLR (Common language **runtime**) provides a space for us to run our program in our desired processors
* FCL (Framework class **library**) provides a library of previously written accessible code for us to use in our programs

## Using the .NET

* Using .NET we can create project files specific to our processor(console, web, app)
* We can then use our editor to change the code and make the commands more complex

## NuGet

* Contains packages that we can use from 3rd party developers including Microsoft
* “dotnet restore” takes the external dependencies and adds them the file path
* “dotnet build” produces .dll extension file (code in binary format)
* “.obj” contains extra files (can be deleted)

# C# Introductory:

### Mathematical operators

* +,-,”\*”(multiplication),”/”(Division)

### Var

“var” is a blank type keyword that lets the program recognize what variable type it is using:

Ex: var x = 12.3;   
 double y = 12.3;

In the example the C# complier will recognize that “x” is a double because of the initializing statement

### Typing Shortcuts:

Can write abbreviations to what you want to write, the complier will recommend a statement which you can select using the **tab** key

### Indexing arrays:

When indexing an array, beside the variable use “[#]” where the number relates to the index position of the variable inside the array.

C# complier will not allow you to use a variable that is unasigned, because it usually leads to errors. EX:

double[] numbers ;

vs.

double[] numbers = new double[3];

vs.

var numbers = new[] {12.3, 15.7, 4.6};

hl

“foreach()” operation lets you iterate through each component in an array

foreach(double number in number)

{

//instead of using double you can write var!

}

## Encapsulation:

Breaking lines of code into methods and classes to understand more easily. Ex: imagine you write 10 lines of code for some unnamed purpose and you come back after 5 months and are asked what it does… you wont know, however if you were to put that code into a method and named it by what it does you could easily understand what your code does instantly without having to try sort through and re-understand it!

## Abstractions:

making sure others can use and understand your code???

# Unit Testing:

Prove or verify our code is behaving correctly, under regular and edge conditions(0, null, out of bounds tricky cases).

* A **unit** refers to a specific piece of code we want to look at
* Testing is automated using test runners (will use xUnit to test in C#)