Improvements

- I've recovered past Learn C# lessons and have included them in the class Github page as PDF's.
- New lessons will be stored on the Github page as PDF's as well.
- Lessons will now be simplified to avoid wordy blocks of text. In order to prevent you from losing interest

Introduction to Methods/Functions

Thus far we have learned a lot of powerful techniques that have already made you a competent programmer. Some of you have maybe even gone off on your own and made several projects of your own. This week we will learn about Methods or as they are known in C#, Methods. A lot of languages switch between the two names and it's standard to call them Methods in C#. **Methods are reusable blocks of code that a programmer can rely on.**

Keep in Mind

If you create a variable in our main code it exists nearly every where and we can use it nearly anywhere. If we create a variable in an **if** or **loop** the variable only exists inside that block of code. **Methods will have the same scope problem and will force you to think on how to move data into and out of them.**

Creating Methods

This entire time we have been programming in main, Methods must be placed outside of the main code block. Usually we begin programming inside our **Main** shown below.

```
static void Main(string[] args)
{

}
```

Methods will be created before that. Not only that we will need a way to keep coming back to the Methods above our **Main** block of code. In order to do that in our **Main** we will use a technique called a **Call**. A **Call** is tells our program to stop reading from our **Main** block of code and go to the Method we **Call,** once that is finished we go back to **Main**.

Example

To read this code start at Main read the code in it, stop at the call Add();, then go to the Method Add() finish reading all the code inside then return back to main to the line after the Call.

```
static void Add(){


}

static void Main(string[] args)
{

        Add();

}
```

Tips & Tricks

To see more clearly what is happening in the code try stepping through that code. In Visual Studio's toolbar click on the debug window, click step into, after that you will see a yellow arrow on the left side of your code. That tells you the current line of code being executed. Hit F11 on your keyboard to see the code execute step by step. You'll notice that when it reaches the **Call** it will go to the Method instead of continuing normally. Step Into is a good way to see where issues in your code exist.

Methods and Variables

Okay so we now know the structure of Methods so lets pass data to them. Think of a Methods as a factory that will spit out the thing you desire. You typically don't know what goes on in the factory you just know you get what you want. For Methods we have to implement what happens in that factory.

```
static int Add(int valA, int valB){
       int result = a + b;
       return result;

}

static void Main(string[] args)
{

       Convert.ToInt32(Console.WriteLine(Add(5,8)));

}
```

In the above example we a Method named Add with a **parameter** of int valA and int valB. The parameter is those variables next to the Methods name. Those variables will store any data we send to them. The data we sent is the **argument** (5,8). We wrapped the Call with a Convert.ToInt32(Console.WriteLine()) because we want to display whatever it is the Method returns and because it's a number to we need to convert the values into integers.

Structure of a Method

Method

```
static int Add(int valA, int valB)
```

- `int` is the return type, if you enter void it should return nothing
- Add is the Method name
- (int valA, int valB) is the parameter
- `return` We need to return an integer

Method Call

- Add(5,8) is the Method call
- (5,8) is the **argument**

Creativity

From here we can continue to reuse the Method and add several different values. From here we can continue using the Method. Below the call we can add another call such as

```
Convert.ToInt32(Console.WriteLine(Add(10,10)));
```

The result will display 20. You could probably see the added benefit of never having to write several variables for a program to do the same thing over and over. Just call a Method, pass some values, and it will return whatever you want.

We can even store the value returned from the Method to another variable. In our Main we can write the following and it will store the result from our Method in a variable in Main.

```
int newResult = Add(10, 10);
```

Assignment

Objective

Now that you know about 30% of what every programmer has under their belt you can make a game of your own. Well at least the beginnings a text based adventure game. You've learned about variables and their types, how to display them on the screen, how to check values with if statements along with loops, you have even learned how to keep data in a list known as an array. Time to put it all together.

High Level Description

Put what you've learned to the test. Create a Start Screen, ask the user for a name and age, ask for an items the user wants, display it all on the screen. You've done it before the main difference now is that I want you to do it all **by putting each separate lesson in a Method.**

```
Knights


Enter any key to continue.
Enter You Name
Cris
Please Enter Your Age
89
Your name is Cris and your are 89 years old.
...
Pick your companion Charmander, Bulbasaur, or Squirtle
Charmander
You selected Charmanderis this correct?
Yes/No?
No
Pick your companion Charmander, Bulbasaur, or Squirtle
Squirtle
You selected Squirtle is this correct?
Yes/No?
Yes
Your first pokemon is Squirtle!
...
Enter a number value
10
Enter another number value
15
25
```

- Alternative: If you don't want to do the same thing over again but with Methods do something different. Try making a Ticket booth for a movie. Have the user select the name of a movie that's in your Array.

Help/Resources

- Some of you may say this is insane, no it isn't, you've done it before. Just put it in a Method.
- I don't know where to start? An example Method file is up on Github showing you multiple ways to do the homework in fact it's the actual solution.
- Methods should only do one thing, don't put your entire program inside of a Method. Just take a look at the example I gave.
- If you're looking for help on Methods online you'll more than likely get more help by typing Methods.