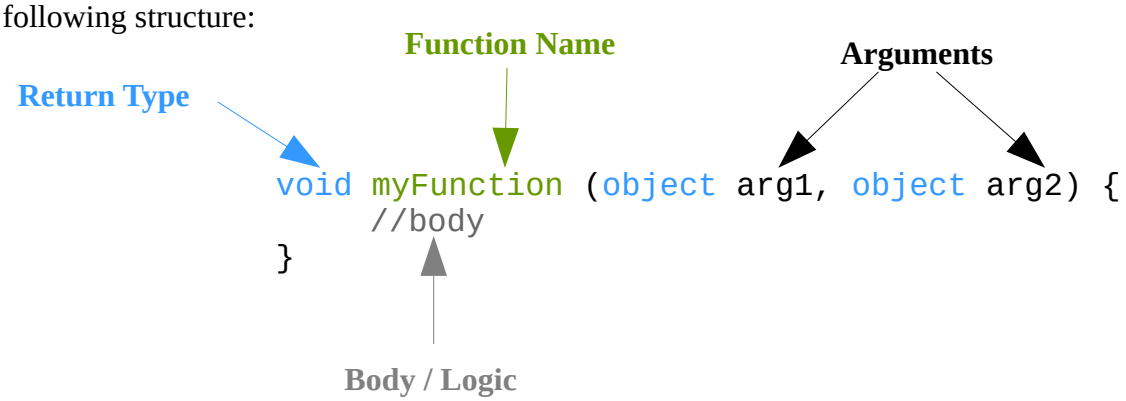
Programming For Arduino

Part 1 - Functions

We will start this tutorial on Arduino programming by looking at the most trivial Arduino program. From the Arduino IDE, just select **file-->new** to see a bare minimum sketch.

```
void setup ( ){
     // put your setup code here, to run once:
}
```

There are two main parts to any Arduino sketch: setup and loop. Each of these blocks of code is a programming construct known as a **function**. A function in a language like the Arduino's has the following structure:



Return Type – The type of object that the function outputs. This could be something like a number or a string of text. It's very common for the return type to be 'void' which means this function doesn't actually output anything, and that's perfectly valid.

Arguments – These are the inputs to the function. There can be any number of them, including none at all. Each argument has both a type and a name; more on that in the part 2 of this document. In the event there are no inputs, you will just see the open and close parens ( ) with nothing inside, and this is a very common case.

Function Name – This is a word that describes what the function does. It can be pretty much anything you choose, but its best to try and pick a name that succintly describes what the function actually *does*, since you will refer to this name when you want to execute the function's code.

Body/Logic - This is the code that actually implements logic to *do* something. You'll notice that the body of the function is wrapped inside curly braces { } and this is important to mark the start and end of the function.

Lets look at the setup function from the blank sketch we loaded. First, we see that the return type is void, meaning this function doesn't return any data. Next, we see that the function is named 'setup'. Next, we see that the function does not take any input arguments. Finally we see that the fuction's body is empty. It does contain a comment (a bit of text with the // symbol in front of it) but this is ignored by the compiler. Comments are a way for programmers to leave notes for each other (and for themselves) in the code files.

The function 'setup' will be executed just once when the Arduino first starts up. It has expects no arguments and returns no argument. The second function 'loop', which also expects and returns no arguments and has an empty body, gets run continuously over and over while the Arduino is powered.

Part 2 - Variables

Now we're going to do something new: declare a variable. A variable is a contruct where we can save some value to memory and recall it later. Before we can use a variable, we must declare it. The declaration of a variable has the following format:

```
type variableName = value;
```

Type – This can be a difficult concept for first time programmers. When we assign a varaible to memory, we do so with context of what *type* of data we are storing. For instance, we may be saving an integer, a list of characters, a number with floating point precision, etc. Each of these examples would be represented by a different data type.

VariableName – This is the name we will call this variable. It can be just about any word, but it can't include spaces or some other special characters. This is why we often use a format called 'camel case', where we capitalize the first letters of new words instead of using spaces. Choosing descriptive variable names can make your program easier to understand.

Value – This is the actual value we are assigning to that variable. Some times we may just manually type in a value, and other times we might assign this value to be the output of some other process or function. Note that you can create a variable without assigning a value to it, a value can be assigned later in the program.

Here is an example where we create a new variable (of type integer), that represents the meaning of life, the universe, and everything. We give an an appropriate name and assign it a value of 42.

```
int lifeTheUniverseAndEverything = 42;
```

Here is another example where we create a new variable (of type float), name it runningTotal but do not assign it a value (yet).

```
float runningTotal;
```

Part 3 – Programming Challenge

- Now open the included sketch '**ArduinoProgramming.ino**'

This program is designed to generate two random numbers, add them together, and then print the results out to the serial monitor for us to view. Its supposed to do this by using a function called AddInts that has the following structure:

```
int AddInts (int num1, int num2) {
     //body
}
```

However, this program isn't working! Instead of returning the sum of num1 and num2, it ALWAYS returns the number 10, regardness of what the input values were. Can you modify this program so that it returns the sum of num1 and num2 instead? Note the use of the special word 'return' in the body, this is how we specify what value to return out of the function.