|  |  |
| --- | --- |
| **Reading Time** | 20 mins |
| **Difficulty** | Beginner |

|  |
| --- |
| **Sections** |
| Five Ingredients of Coding |
| What does code look like? |

Time Taken: 10 mins

What you’ll need: 1 brain

Difficulty: Easy

We’ll be covering:

1. What is Coding
2. What are variables?
3. What are functions?
4. What are structures?

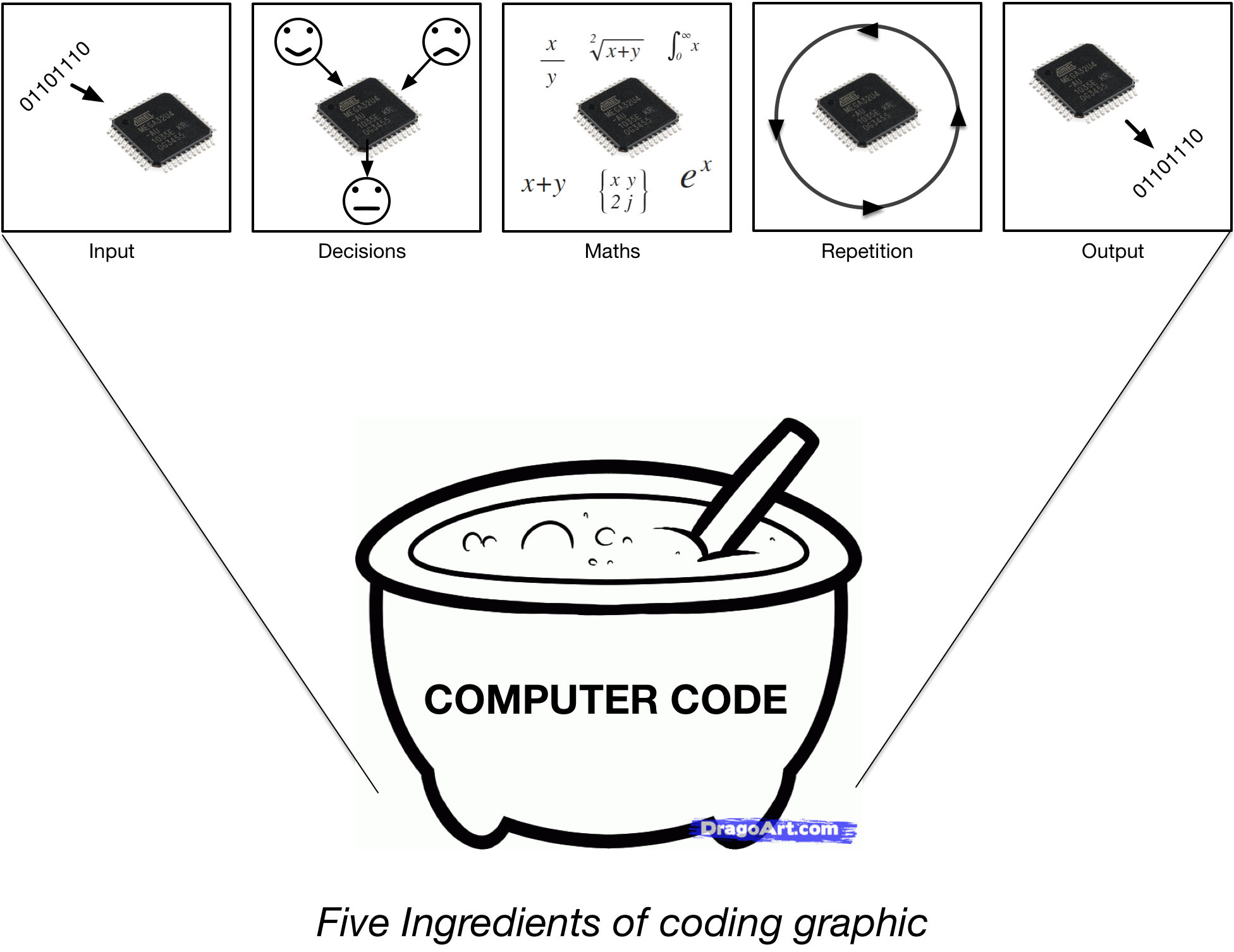
**What is Coding**

**Coding means telling a computer what to do. It’s how we make computers do what we want. Coding is like writing a set of instructions for someone to follow, except that the “someone” has no intelligence and needs to be told exactly what to do.**

**What’s the essence of what a computer can do?**

**The Five Ingredients of Coding**

1. **Input: Reading information from outside.**
2. **Decisions: Making decisions based on information.**
3. **Output: Sending information out.**
4. **Math: Performing calculation**
5. **Repetition: Repeating operations many times.**



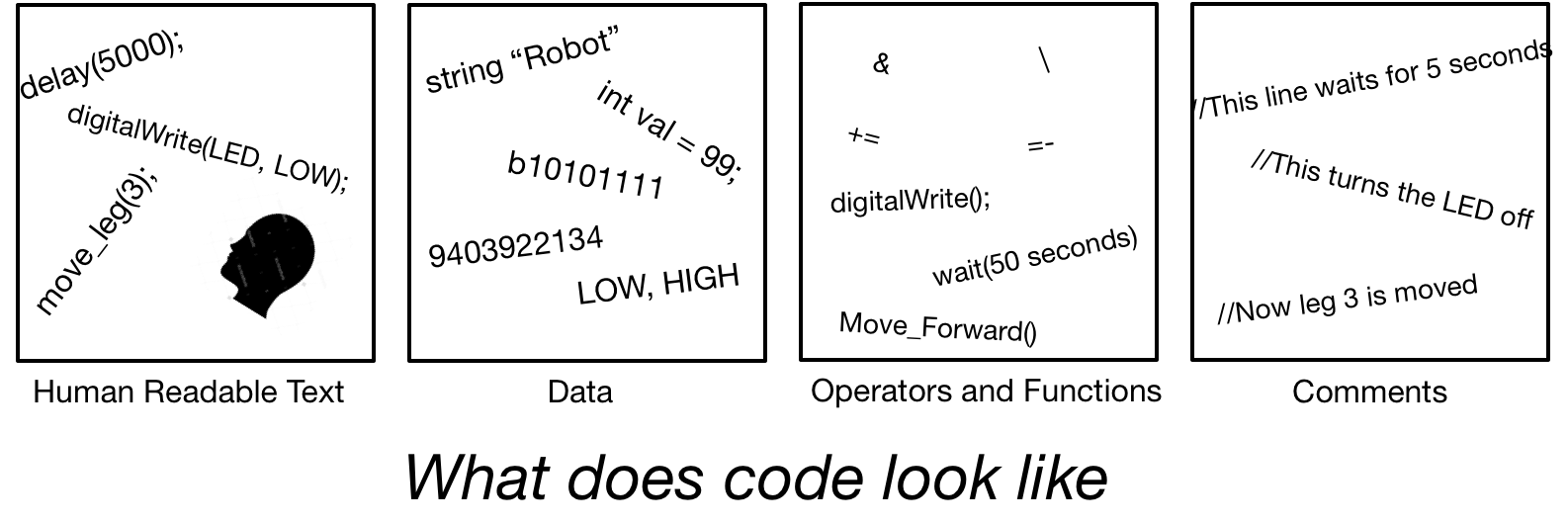
**When we code, we mix the five ingredients together to create a program. Just how you might mix flour, milk, sugar and eggs to make a cake (if you like baking).**

**With the five basic ingredients, we have all the complexity of modern computers, smartphones, robots and so on. That’s it.**

**What does code look like?**

**There are many different types of code, each called a Programming Language. They are all different but share some characteristics.**

1. **Human Readable Syntax: Keywords and structure that a human can understand.**
2. **Data: Some way of handling data or information.**
3. **Operators and functions: Symbols or words that perform operations on data.**
4. **Comments: Text to document the code, that doesn’t affect the operation.**



**Here’s an example of some code…**

while(1){ //Repeat forever

digitalWrite(13, HIGH); //Turn on LED on pin 13

delay(1500); //Program waits for 1.5 seconds

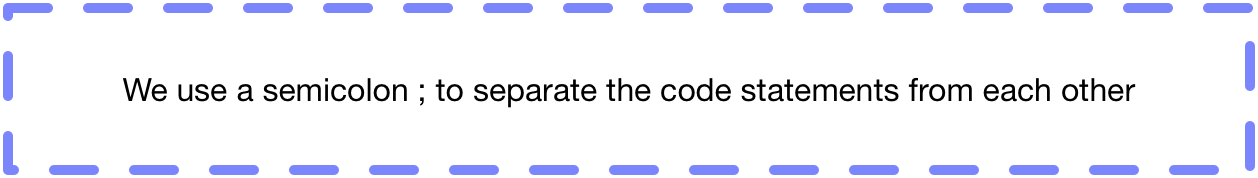
digitalWrite(13, LOW); //Turn off LED on pin 13

delay(1500); //Program waits for 1.5 seconds

}

**This code uses some two different functions, some data, and a structure to flash an LED on an output pin of a computer. There’s also comments to help a user understand to program.**

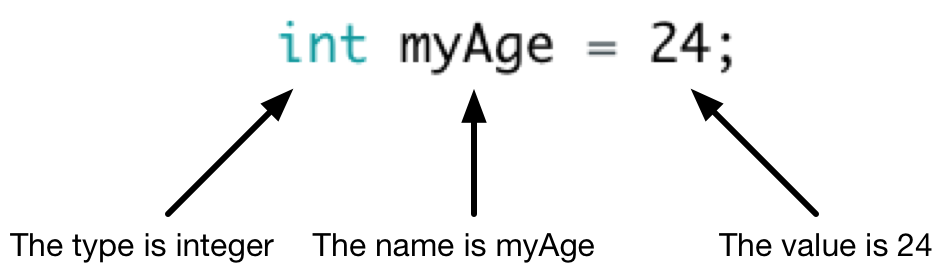
**Let’s dive into the details and look at the Arduino C++ language. This language is used to program Arduino hardware and QuadBot.**



**What are variables?**

**We use variables to store information in a program. A variable has a name, a type and a value.**

* **The type specifies what kind of information it is. For example a whole number, a character a decimal number and so on.**
* **The value is the actual data we are storing.**
* **The name is what we use in a program to access the data.**



* **This variable is an integer (whole number).**
* **The variable is called myAge.**
* **The value of the variable is 24.**

**Data Types**

**There’s loads of variable data types we can use depending on what we need. Here are some common ones we’ll use in the Arduino language.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Type** | **What does it store** | **Range** | **Example** |
| **byte** | **An integer** | **0 to 255** | **byte height = 181;** |
| **int** | **An integer** | **-2,147,483,648 to +2,147,483,647** | **int time = 1700;** |
| **float** | **A floating point number (a number with a decimal point)** | **-3.4028235 x 1038 to +3.4028235 x 1038** | **float weight = 76.2;** |
| **boolean** | **A single boolean number (1 or 0)** | **1 or 0** | **boolean hungry = 1;** |
| **char** | **A character, like a letter, symbol or number** | **All** [**ASCII**](https://www.arduino.cc/en/Reference/ASCIIchart) **characters** | **char grade = ‘A’;** |
| **string** | **A string of characters** | **Any combination of characters** | **string name = “Josh”;** |

**These are the data types you’ll see and use with QuadBot but there are many more** [**data types**](https://www.arduino.cc/en/Reference/HomePage)**.**

**How do we use variables?**

**Declaration**

**First we must declare our variable to the computer. The computer can then allocate some memory to store that variable.**



**We must always tell the computer the name of the variable and the data type. We don’t have to tell the computer the value if we don’t know it yet.**

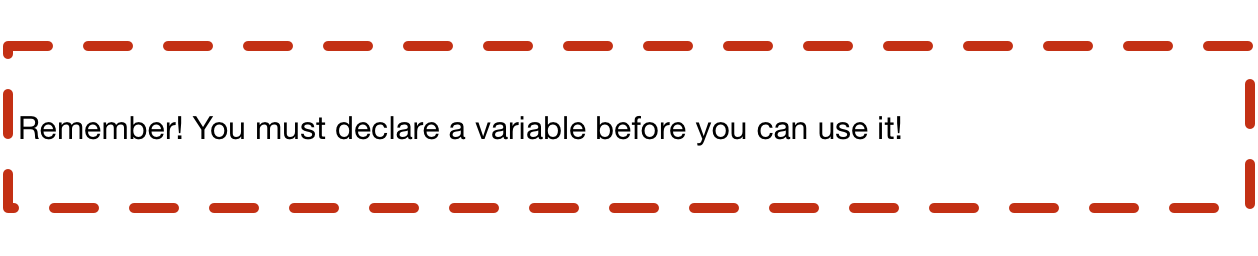
int life;

**Declares a variable called “life” of type int. We can now use the variable “life” in our program.**

**If we know the value, we can declare the value as well, like this…**

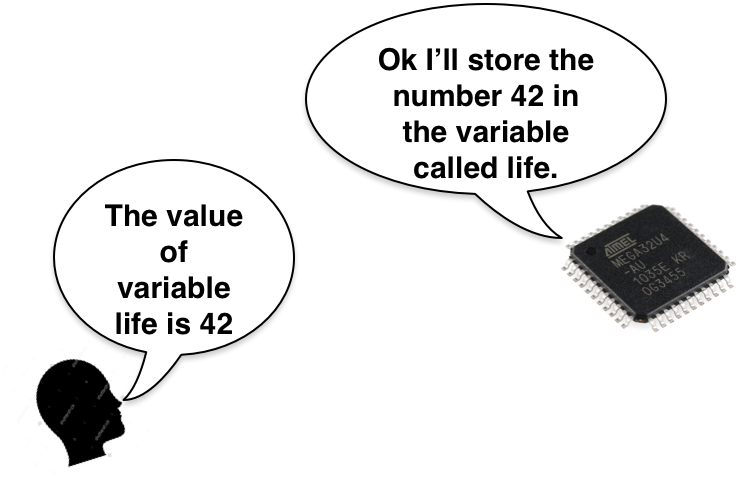
int life = 42;

**Declares a variable called “life” of type int, of value 42;**



**Writing**

**We can change the value of a variable by writing a value to it. The computer then stores our value in the variable. We can do this as many times as we like.**



**For example...**

life = 45;

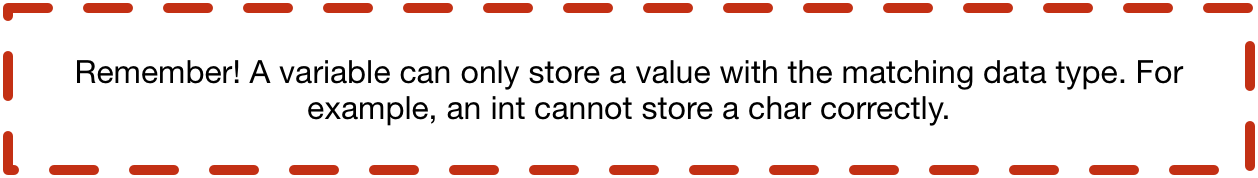
**The value 45 is stored in life. If we then write...**

life = 43;

**Now the value is 43. The previous value of 45 is lost. If we write...**

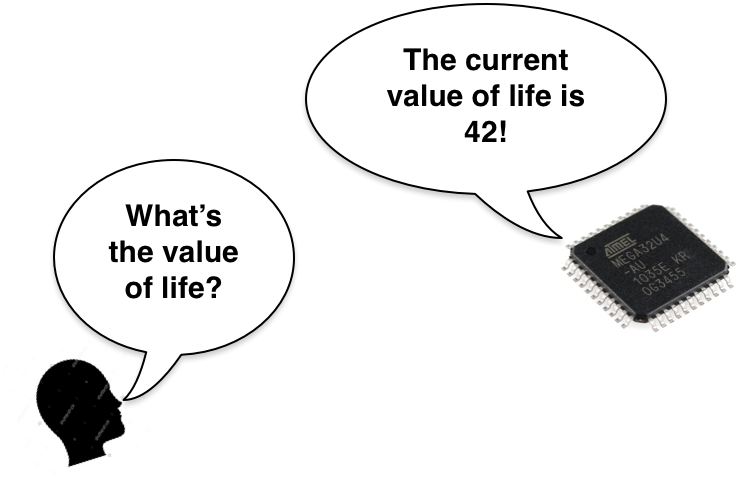
life = 41 + 1;

**life now has a value of 42.**



**Reading**

**We can access the value of a variable by reading it.**



**We read the value of a variable in our program. Reading a variable is simple, you just call it’s name.**

life;

**The computer reads the value of the variable, but it hasn’t actually done anything with useful with that variable. A better idea is to read that variable inside a function like this...**

print(life);

**The computer reads the value of life and then prints it for the user to see. We could also read the value of life and use it to initialize a different variable.**

int universe = life;

**The value of life is written to the integer called universe.**

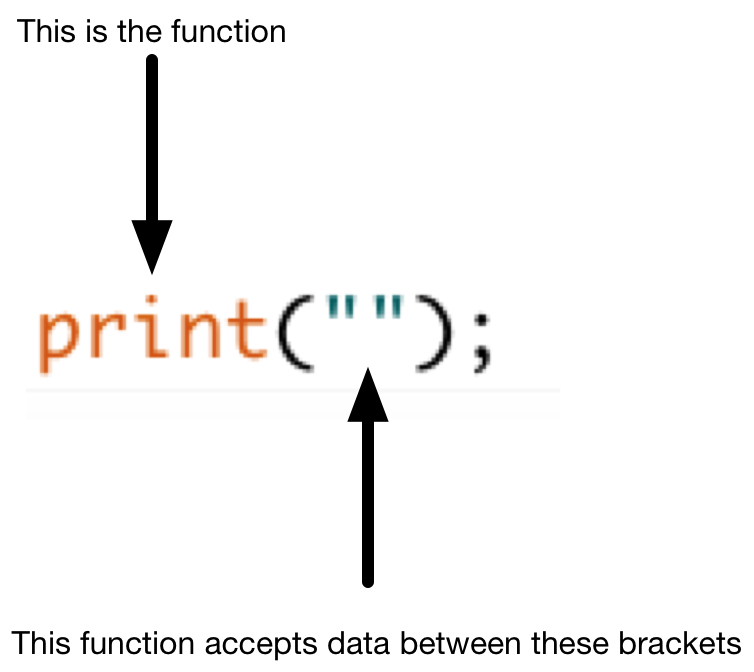
**That’s the basics of how computer programs store data. Let’s move on to functions…**

**What are Functions?**

**Functions do the heavy work in programs. They do the calculations, the input and output, and just about everything else. The print() statement we just explored was a function**

print(life);

**Functions can accept data as inputs and return data as an output. Some functions don’t accept or return any data. Others only either accept or return data. For our print function...**



**The delay function is another common one we’ll use.**

delay(5000);

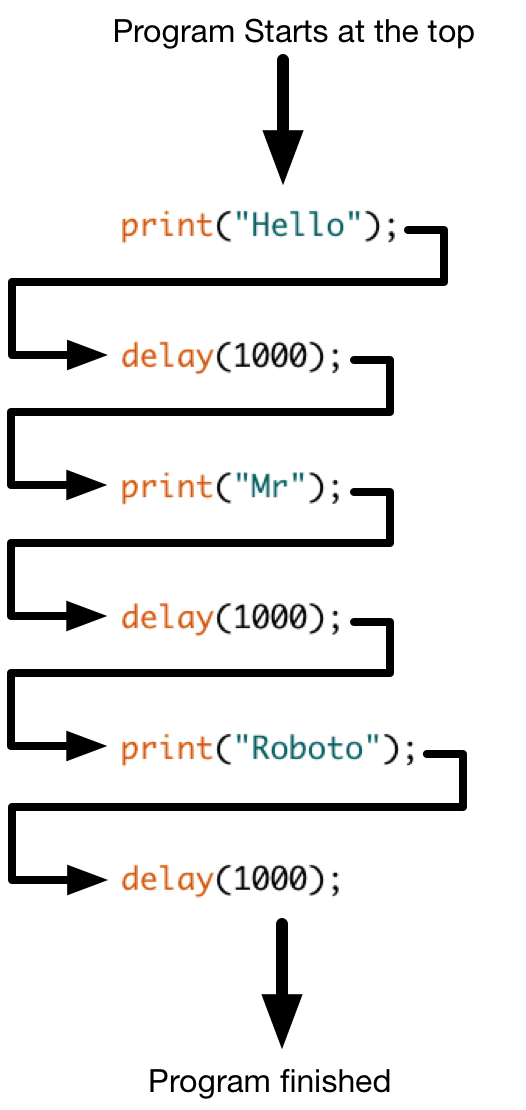
**This function tells the computer to wait for 5 seconds. It takes the delay time as an input, and does not return anything.**

**There are many functions we’ll use for QuadBot, but you won’t have to memorize them. You can find out how to use each one on the** [**Arduino reference page**](https://www.arduino.cc/en/Reference/HomePage) **and our QuadBot reference page.**

**What are Structures?**

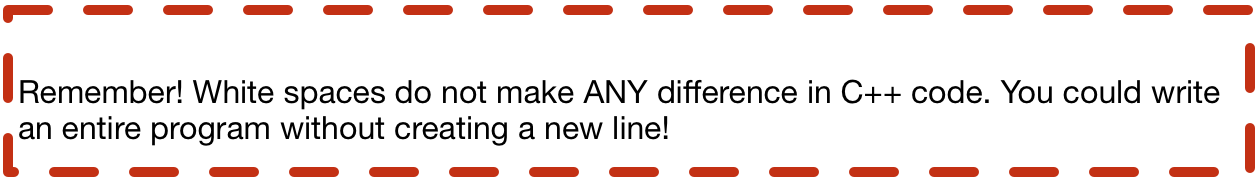
**How do programs run? It’s all in the structure…**

**The programs we’ll be writing are sequential; operations happen one after the other, from left to right, top to bottom.**



**This program starts at the top and runs through the functions one at a time. Believe it or not, we could write the same program like this…**





**A good coder will always add spaces in their code to make it easy for others to read.**

**Loops and how to use them**

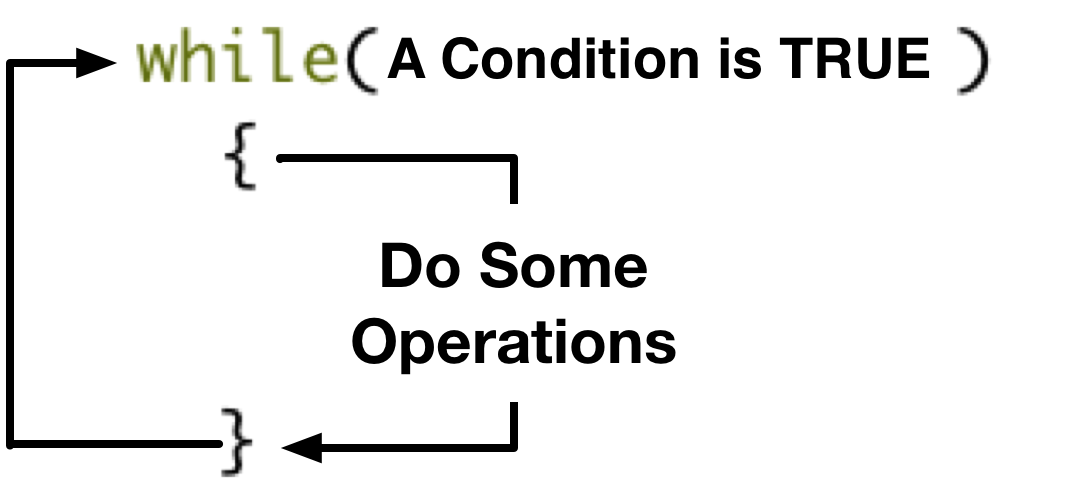
**In most programs we’ll want a computer to repeat itself over and over again. If we wanted the computer to print “Robots are awesome!” continuously, we could do so like this…**



**But this the slow way, a better way is to use a repeating loop.**

**While loop**

**A while loop is a structure that repeats any statements inside it while a condition is true.**



**For example…**

while(loopCount < 10){ //Repeat this code while loopCount < 10

print(“Robots are awesome!”); //Print the text to the user

}

**This code will keep printing “Robots are awesome!” until the variable loopCount becomes more than 10. The problem is, nothing is going to make loopCount increase, so it will never be more than 10. That means the loop will never stop.**

**We can fix this by initializing loopCount to 0 before the loop, and adding 1 to loopCount every time the loop runs.**

Int loopCount = 0; //Initialize loopCount to 0

while(loopCount < 10){ //Repeat this code while loopCount < 10

print(“Robots are awesome!”); //Print the text to the user

loopCount = loopCount + 1; //Add 1 to the value of loopCount

}

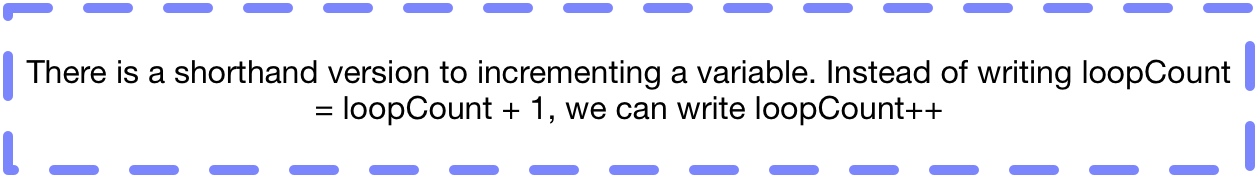
**TRY THIS ON YOUR QUADBOT**

**A special case of the while loop is the infinite loop. This is a loop that always evaluates as true, so never stops running. It looks like this…**

while(1){

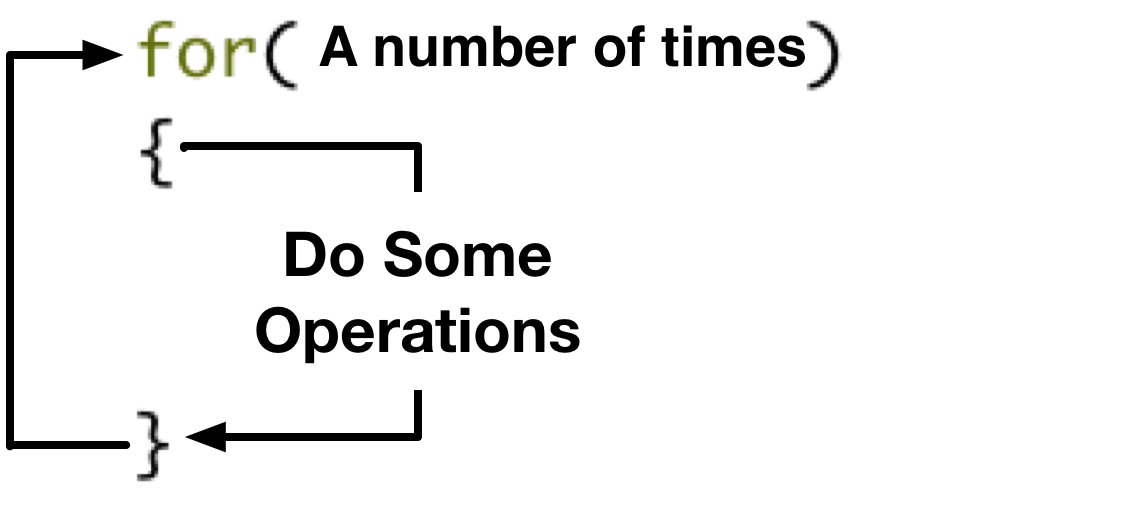
print(“I’ll never stop being awesome”);

}



**For loop**

**If we know for sure how many times we want a loop to run, for example in math calculations, we can use a For loop.**



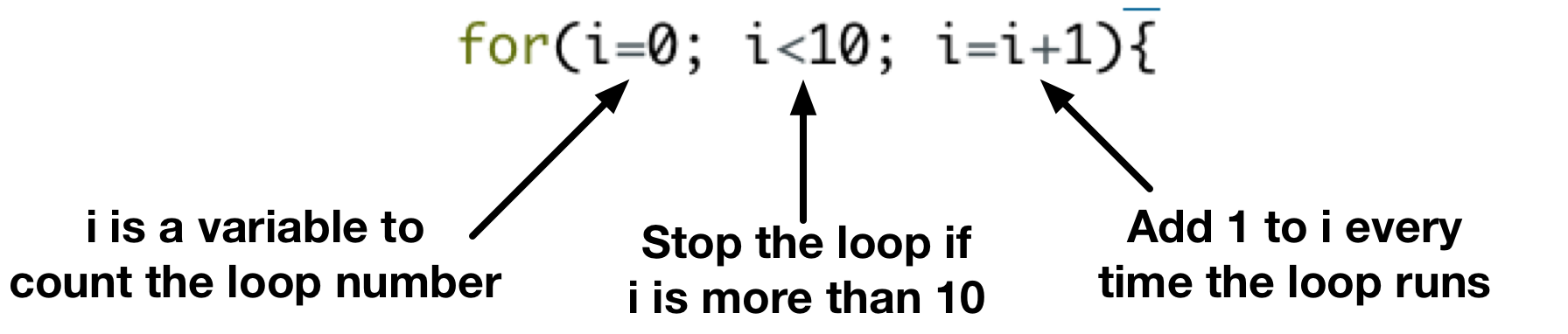
**Here’s what that actually looks like.**

for(int i=0 ; i<0 ; i=i+1){

Serial.println(“Still awesome”);

}

**To control the loop, we use the code inside the for() brackets.**

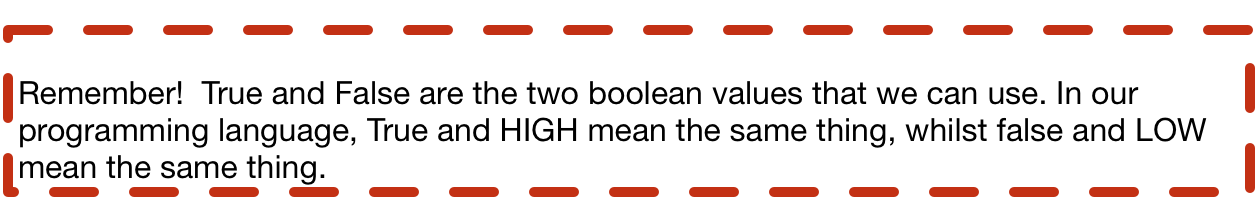


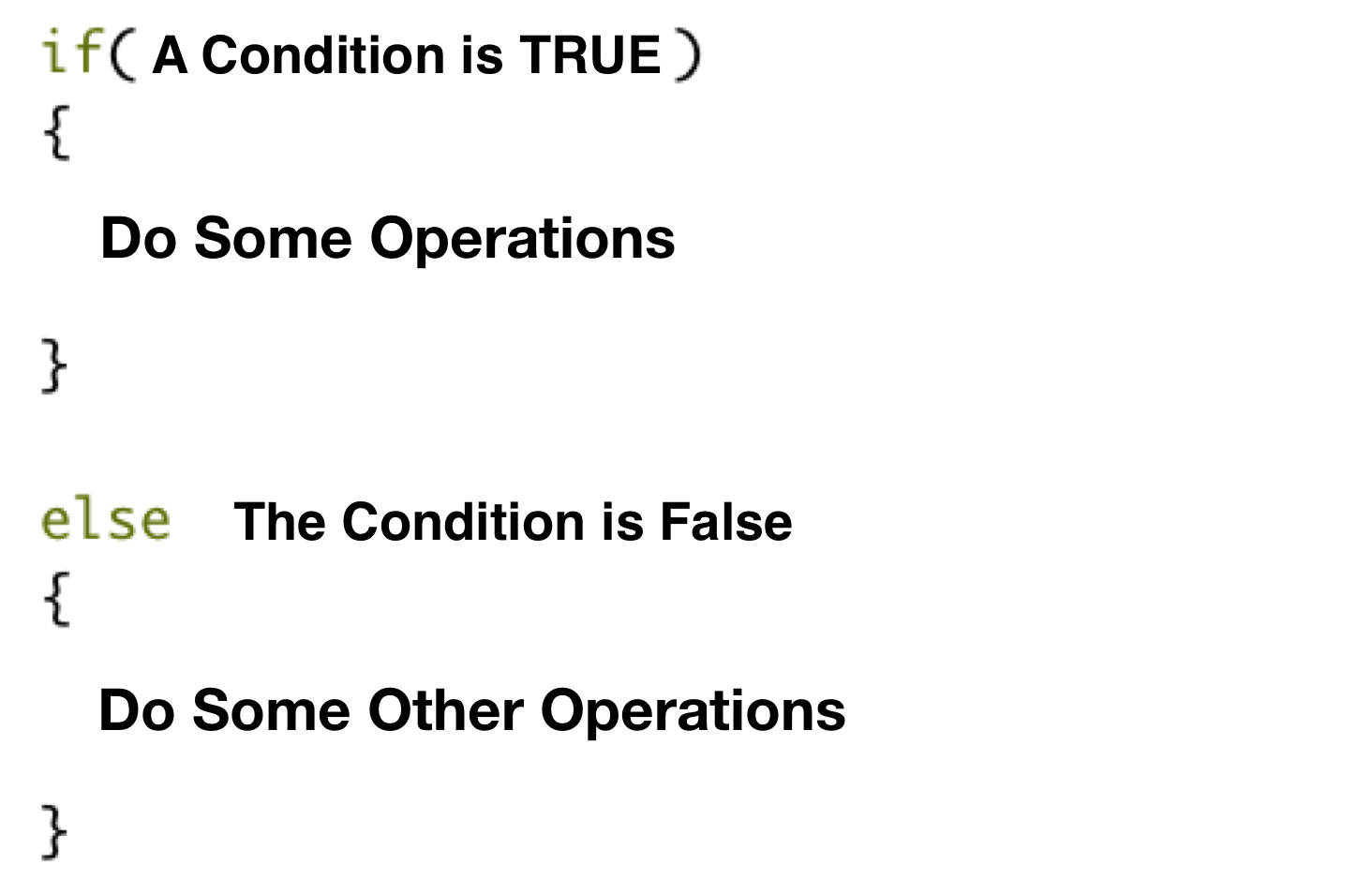
**i starts at 0 and increments every time the loop runs. When i is more than 10, the loop stops.**

**Like the while loop, this code will print “Still awesome” 10 times!**

**If-else statement**

**If-else are structures to make decisions in code. Based on whether a condition is true or false, a different piece of code will run.**





**For example…**

if(robots > 10){

println(“Awesome, over 10 robots”);

}

Else{

println(“Sad times”);

}

**Wrap Up**

**You should now be armed with the knowledge you need to understand basic programming…**

**Now move on to input and output...**