
Welcome back

If I were a...

- A Household Object
 - A Cartoon Character
-

—

Recap from yesterday

What did we learn?

Programming I

Samuel Bechara, PhD

—

What makes a computer (or phone) smart?



Computers aren't smart!
People are smart!

Computers are
dumb, they need
smart people to
tell them what to
do



**Vamos a levantarnos todos
ahora!**

**Everybody stand up right
now!**

The moral: **you have to speak
the language** of who/what
you want to give commands!

**That is what computer
programmers do.
They speak computer.**



Before we dive in...

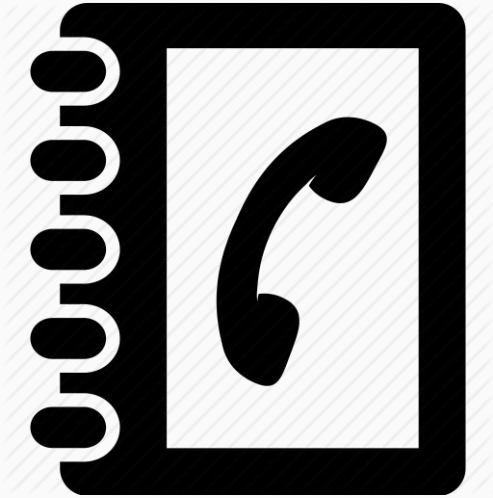
→ **Computers need PRECISE
commands**

You have to be VERY specific when talking to a computer (remember, they are dumb)

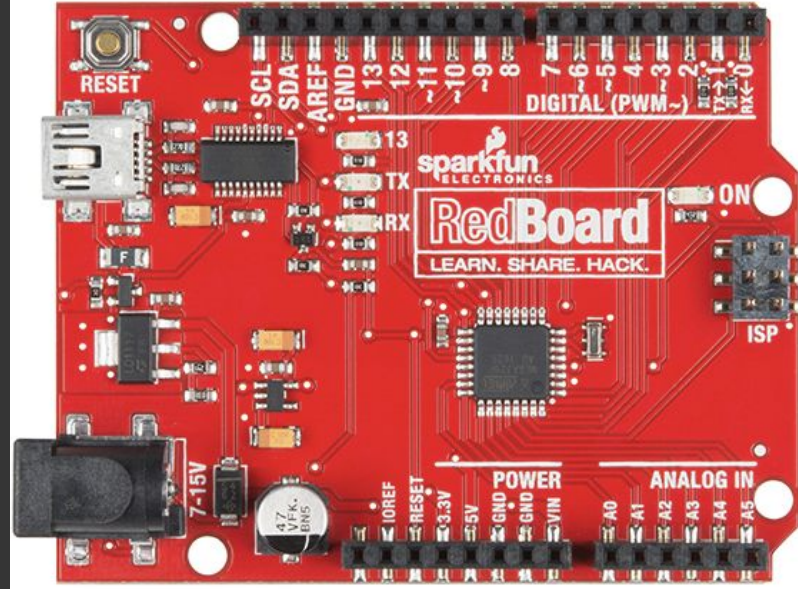
→ **They are very picky about case
and names**

Your first computer program - Phone Book “App”

Let's think about a phonebook app (you have one on your phone I am sure).



Recall:
What do **you**
need the
arduino for?





Arduino Programming Basics

Our Journal Process for Coding:

1. Describe program in words
2. Write Pseudocode
3. Sketch Diagrams and Flow of Code
4. Program
5. Test
6. Repeat

Arduino Programming Language Syntax

Syntax is concerned primarily with **word order** in a sentence and with the **agreement of words** when they are used together.

So in essence, syntax is a fancy way of saying the **rules of a language**. Just like the english language has a special syntax and rules, so do programming languages.

Spot the mistakes:

Yes, I was speaking to him yesterday.

Arduino Programming Language Syntax

Syntax is concerned primarily with word order in a sentence and with the agreement of words when they are used together.

So in essence, syntax is a fancy way of saying the **rules of a language**. Just like the english language has a special syntax and rules, so do programming languages.

Spot the mistakes:

✗ Yes, I was **speaking** to him yesterday.

✓ Yes, I **spoke** to him yesterday.

Her mother made her to call and thank him for the present.

Arduino Programming Language Syntax

Syntax is concerned primarily with word order in a sentence and with the agreement of words when they are used together.

So in essence, syntax is a fancy way of saying the **rules of a language**. Just like the english language has a special syntax and rules, so do programming languages.

Spot the mistakes:

✗ Yes, I was **speaking** to him yesterday.

✓ Yes, I **spoke** to him yesterday.

✗ Her mother made her **to call** and thank him for the present.

✓ Her mother made her **call** and thank him for the present.

If I took the bus, I will get there in 20 minutes.

Arduino Programming Language Syntax

Syntax is concerned primarily with word order in a sentence and with the agreement of words when they are used together.

So in essence, syntax is a fancy way of saying the **rules of a language**. Just like the english language has a special syntax and rules, so do programming languages.

Spot the mistakes:

✗ Yes, I was **speaking** to him yesterday.

✓ Yes, I **spoke** to him yesterday.

✗ Her mother made her **to call** and thank him for the present.

✓ Her mother made her **call** and thank him for the present.

✗ If I took the bus, I **will** get there in 20 minutes.

✓ If I took the bus, I **would** get there in 20 minutes.

Gary lives on the Elm Street.

Arduino Programming Language Syntax

Syntax is concerned primarily with word order in a sentence and with the agreement of words when they are used together.

So in essence, syntax is a fancy way of saying the **rules of a language**. Just like the english language has a special syntax and rules, so do programming languages.

Spot the mistakes:

✗ Yes, I was **speaking** to him yesterday.

✓ Yes, I **spoke** to him yesterday.

✗ Her mother made her **to call** and thank him for the present.

✓ Her mother made her **call** and thank him for the present.

✗ If I took the bus, I **will** get there in 20 minutes.

✓ If I took the bus, I **would** get there in 20 minutes.

✗ Gary lives on **the Elm Street**.

✓ Gary lives on **Elm Street**.

While watching a movie, people who text on their phone are very annoying.

Arduino Programming Language Syntax

Syntax is concerned primarily with word order in a sentence and with the agreement of words when they are used together.

So in essence, syntax is a fancy way of saying the **rules of a language**. Just like the english language has a special syntax and rules, so do programming languages.

Spot the mistakes:

✗ Yes, I was **speaking** to him yesterday.

✓ Yes, I **spoke** to him yesterday.

✗ Her mother made her **to call** and thank him for the present.

✓ Her mother made her **call** and thank him for the present.

✗ If I took the bus, I **will** get there in 20 minutes.

✓ If I took the bus, I **would** get there in 20 minutes.

✗ Gary lives on **the Elm Street**.

✓ Gary lives on **Elm Street**.

✗ **While watching a movie**, people who text on their phone are very annoying.

✓ People who text on their phone **while watching a movie** are very annoying.

Arduino Syntax Rules - // Commenting

Comments

Comments are just like they sound like. They are snippets of code for the programmer or other people inspecting the code of a program. The computer completely ignores them. That doesn't mean they aren't important! They make your code readable by humans.

Syntax

There are two types of comments. Single line and multi-line comments.

All **single line comments** start with two forward slash characters and the computer ignores everything on that line.

Example)

```
// This would be a commented line in arduino code
```

A **multi-line comment** starts with a forward slash and asterisk and will span multiple lines. The forward slash and asterisk are typed.

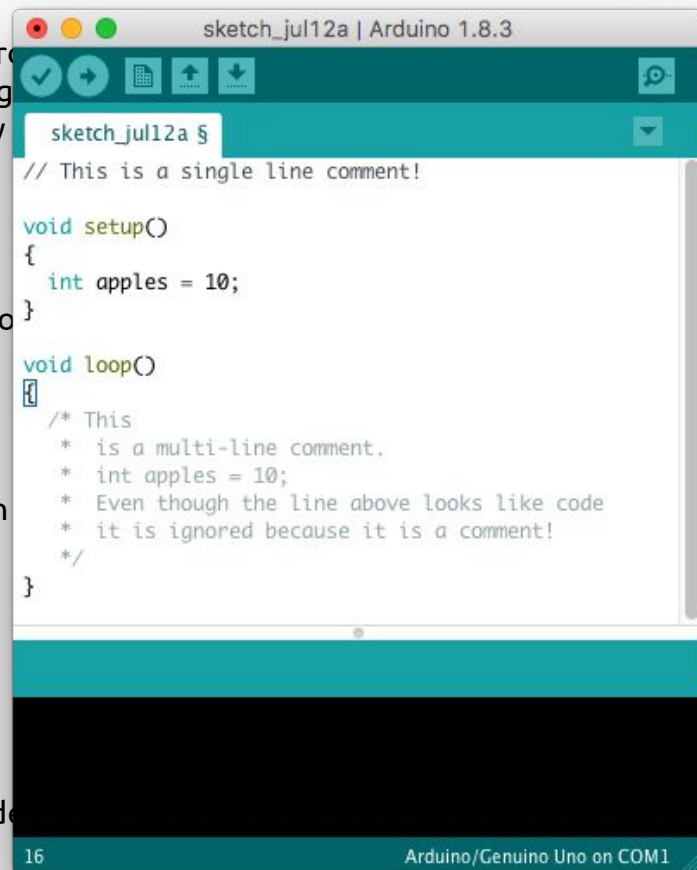
Example)

```
/* This
```

```
Is a multiline comment. Everything until  
the closing asterisk and slash is ignored --> */
```

Notes

The arduino IDE will color your comments in a light grey color to help you identify them easier!



Variables

A variable is a way of naming and storing a value for later use by the program, such as data from a sensor or an intermediate value used in a calculation.

Remember, computers are DUMB so you need to be very specific about what kind of variable you are using.

Kinds of variables (also called “data types”)

- char
- byte
- int
- long
- double

What are the variable naming rules?

Variables

A variable is a way of naming and storing a value for later use by the program, such as data from a sensor or an intermediate value used in a calculation.

Remember, computers are DUMB so you need to be very specific about what kind of variable you are using.

Kinds of variables (also called “data types”)

- char
- byte
- int
- long
- float
- double

What are the variable naming rules?

Variables

A variable is a way of naming and storing a value for later use by the program, such as data from a sensor or an intermediate value used in a calculation.

Remember, computers are DUMB so you need to be very specific about what kind of variable you are using.

Kinds of variables (also called “data types”)

- char
- byte
- int
- long
- float
- double

What are the variable naming rules?

Variable naming rules:

- Variables can consist of any letters (a to z and A to Z)
- Variables *can contain the numbers 0 to 9*, but **may not start with a number**, e.g. 3var is not allowed, but var3 is allowed
- Variables may not have the same names as Arduino language keywords, **e.g. you can not have a variable named int**
- Variables must have unique names i.e. you can not have two variables with the same name
- Variable names are case sensitive, so Count and count are two different variables
- Variables may not contain any special characters, except the underscore (_), e.g. top_score

Arduino Syntax Rules - Variables

Variables

Variables are just containers to store different types of information. Remember, the ones we will use the most often are `int` and `float`.

Use an `int` when you are just counting whole numbers.

Use a `float` for everything else.

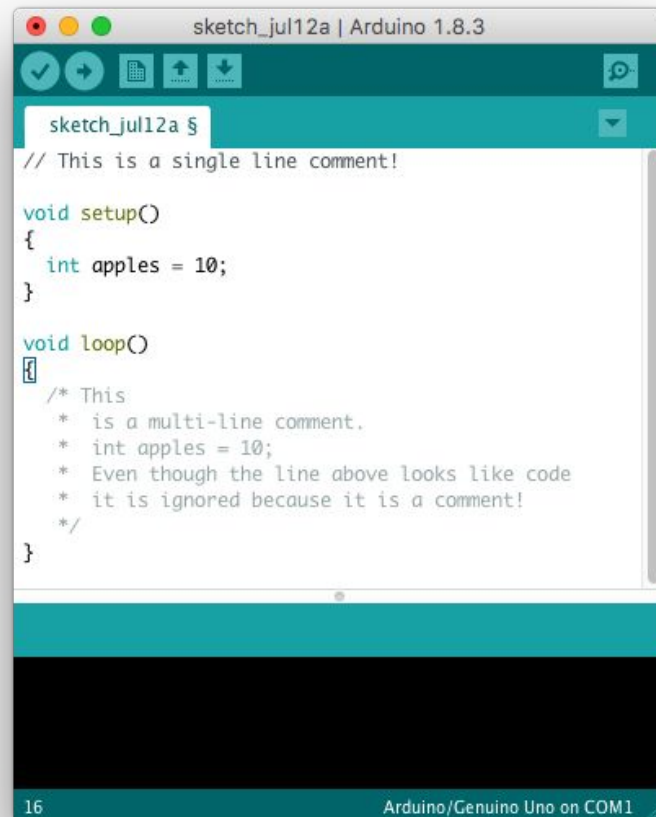
Syntax

All lines of code must end in a semicolon or you will get errors!

Example)

```
int variable1 = 10;
```

```
float variable2 = 20.3;
```



Arduino Syntax Rules - The Semicolon ;

The Semicolon ;

The Semicolon is kind of like the period in english. It signifies to the computer that the line of code that you are working on is finished. The actual white-space (spaces, tabs, etc) don't matter. Arduino waits until it sees a semicolon, THEN determines that the line is over.

Syntax

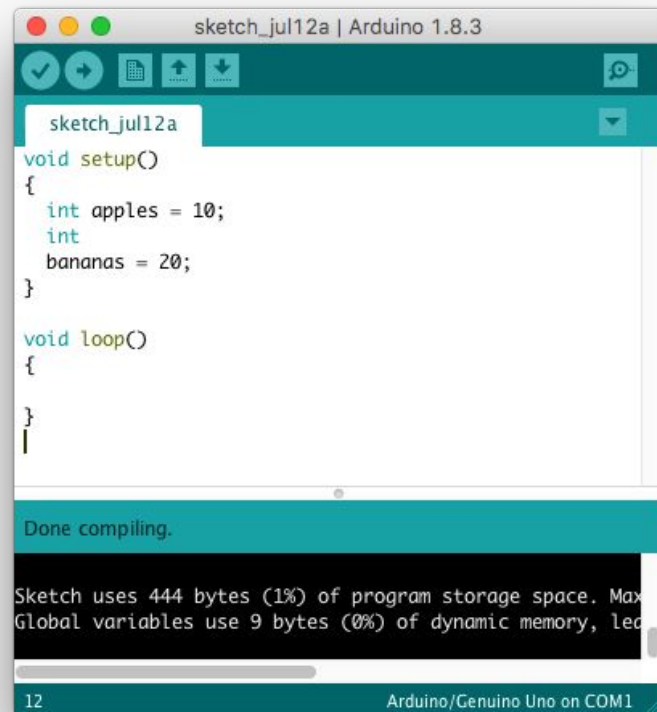
All lines of code must end in a semicolon or you will get errors!

Example)

```
int variable1 = 10;
```

Notes

The only exception to this rule is functions. We will look at that next.



Arduino Syntax Rules - The Function { }

The Function

All Arduino programs *must include two functions*, a **setup** function and a **loop** function. The **setup** function is a collection of code that runs ONCE when the arduino is powered on. The **loop** function runs indefinitely until the Arduino loses power, or the universe ceases to exist or whatever.

Syntax

The syntax for functions is kind of complicated, but for our purpose you need to remember that all code inside functions must be contained within curly braces.

Example)

```
void setup()
{
    // All code is tabbed
    // and must be contained within curlys
}
```

Notes

When arduino colors words in blue, it means they are special words and can't be used as variable names.



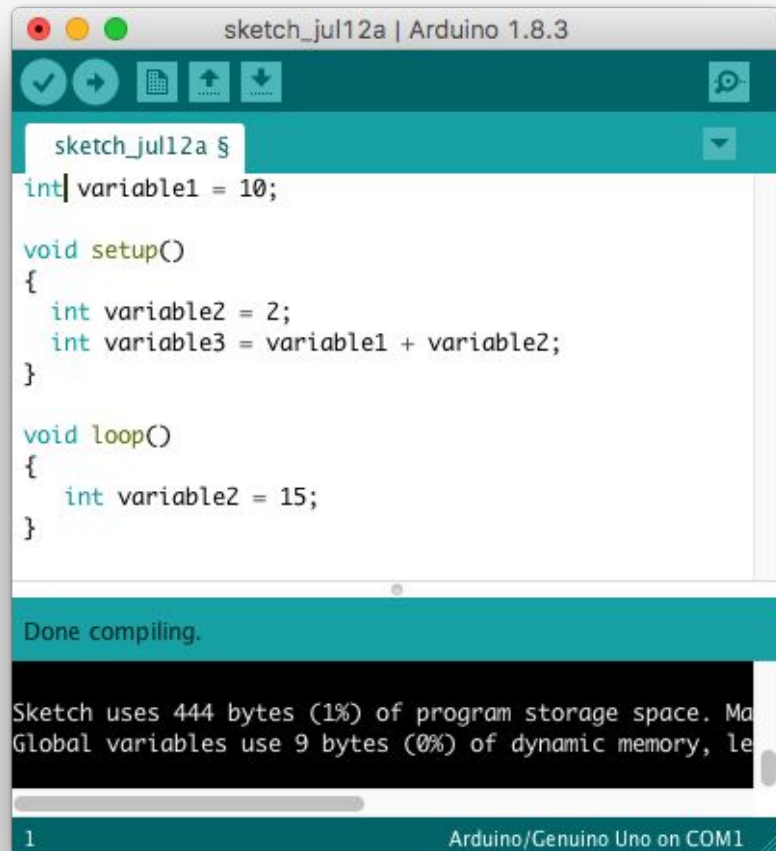
Arduino Syntax Rules - The Function { }

Function Rules

Variables can either be **global** or **local**.

Global variables are defined OUTSIDE of a function and can be used by ANY function.

Local variables are defined INSIDE a function and are only used by the function they are defined in.



The screenshot shows the Arduino IDE interface with a sketch named 'sketch_jul12a'. The code is as follows:

```
sketch_jul12a §
int variable1 = 10;

void setup()
{
  int variable2 = 2;
  int variable3 = variable1 + variable2;
}

void loop()
{
  int variable2 = 15;
}
```

Below the code editor, a status bar indicates 'Done compiling.' and a message box shows: 'Sketch uses 444 bytes (1%) of program storage space. Max allowed is 32256 bytes. Global variables use 9 bytes (0%) of dynamic memory, leaving 23040 bytes free.'

At the bottom, the board is identified as '1 Arduino/Genuino Uno on COM1'.

Arduino Syntax Recap

- Comments `//` or `/* */`
- Variables need a special data type and must be declared and defined
`float var1 = 10.23;`
- Semicolons end lines of code `;`
- Code in functions must be contained within `{ }`
- Functions determine if variables are “Global” or “Local”

Arduino Programming Setup

This is the “base” of every arduino program

```
/* Comment, write about what this program does */

// the setup function runs ONCE when you press reset or power the board
void setup()
{
    [code goes here]; // you can also write comments at the end of lines!
}

// the loop function runs over and over again... FOREVER!
void loop()
{
    [code goes here];
    delay(1000); // the delay is optional but we will be using it a lot today!
}
```

Let's make our first arduino program!

Your arduino has a built-in blue LED. Let's make it flash.

Pseudocode:

1. Turn on LED
2. Wait for 500 milliseconds
3. Turn off LED
4. Wait for 500 milliseconds
5. Return to step 1

Let's make our first arduino program!

Your arduino has a built-in blue LED. Let's make it flash.

Step 1: Write a comment about what the program does at the very top. This will help you remember what the program does easily. Make sure you save it somewhere you will remember and name it something that makes sense.

Example comment)

```
// This program blinks the built-in LED on the Arduino  
// Created by: Dr. Bechara  
// Date: 21 July 2017
```

Example name) arduino_led_blink.ino

Let's make our first arduino program!

In our `setup()` function we need to tell the arduino that we want to set the built in LED to be an **OUTPUT** (we want to talk to it not listen to it)

```
example) pinMode(LED_BUILTIN, OUTPUT);
```

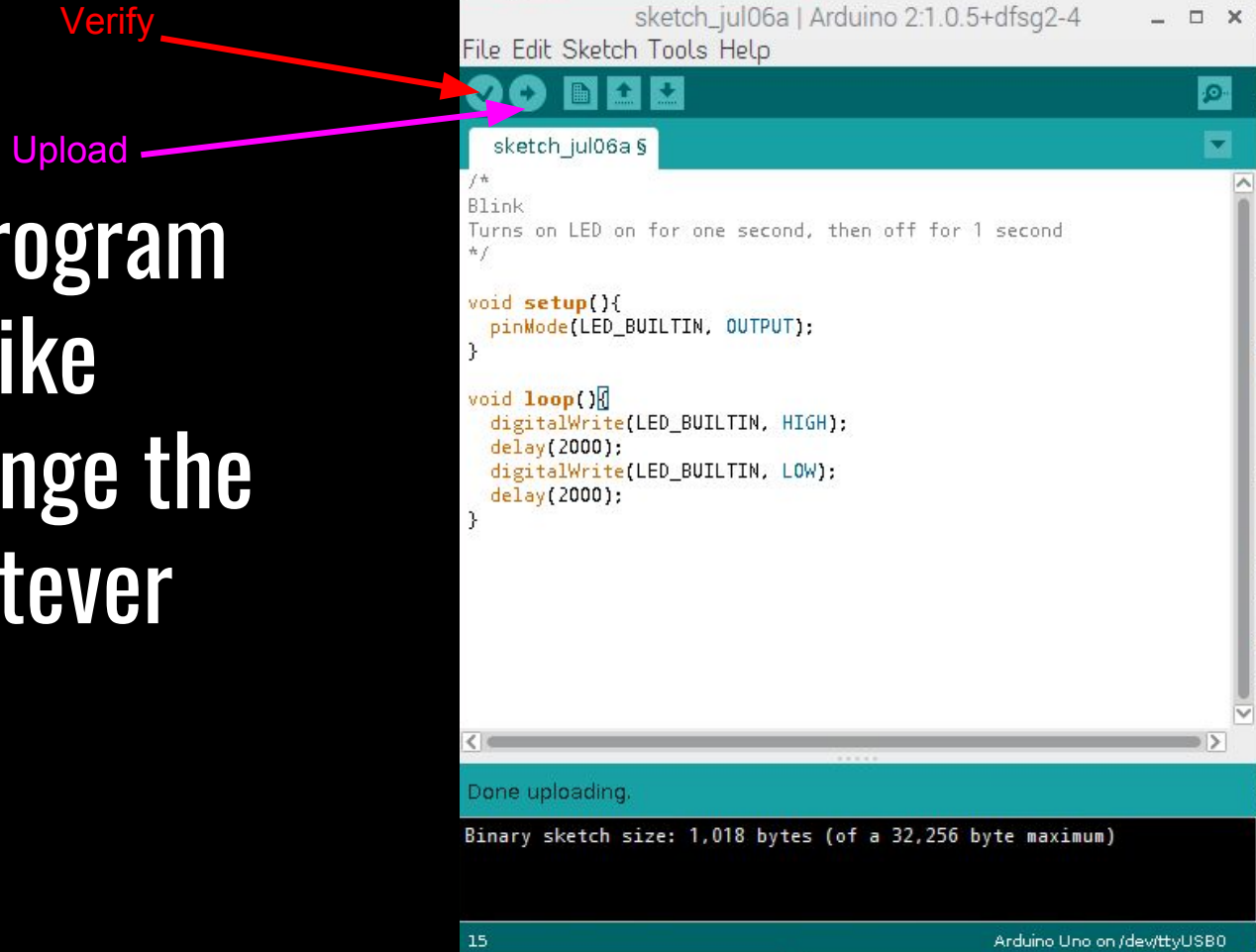
In our `loop()` function we want to change the state of the LED from **LOW** (off) to **HIGH** (on) and flip back and forth between the two. We do that with `digitalWrite(pin, value)`

```
example) digitalWrite(LED_BUILTIN, HIGH);
```

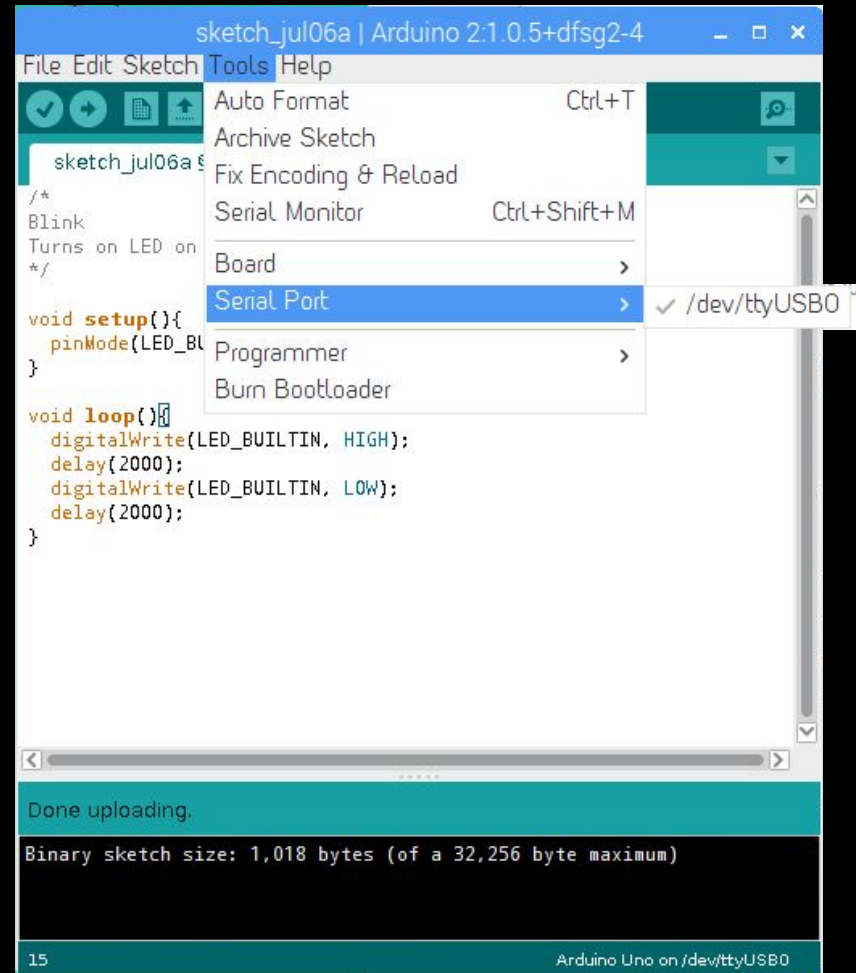
We can use our friend `delay(time_in_ms)` to change the interval the light is on / off.

```
example) delay(2000);
```

What your program
should look like
(you can change the
delay to whatever
you want)



Make sure you
change the serial
port appropriately
then upload the
program to the
arduino



Declaring Variables

You **MUST** declare a variable before you use it

Both ways shown are valid for declaring an “int”

```
int variable1;  
Variable1 = 42;
```

OR

```
int variable1 = 42;
```

Common variable mistakes

1

```
int 2chainz = 2;
```

4

```
double = 2;
```

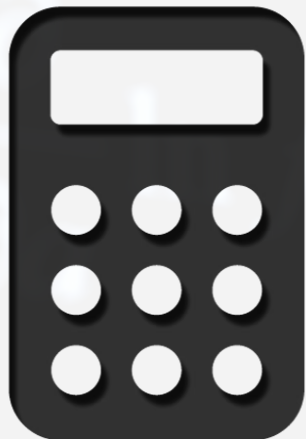
2

```
long voltage;  
Voltage = 23.03;
```

3

```
int bananaMan;  
bananaMan = 56.2;
```

What can you DO with variables?



All sorts of things! You can perform arithmetic just like you would with a calculator.

- + Addition
 - − Subtraction
 - / Division
 - * Multiplication
 - % Remainder
-



math_operation

theoryCIRCUIT.com

```
Serial.begin(9600);

Serial.print("Addition (a + b): ");
result = a + b;
Serial.println(result);

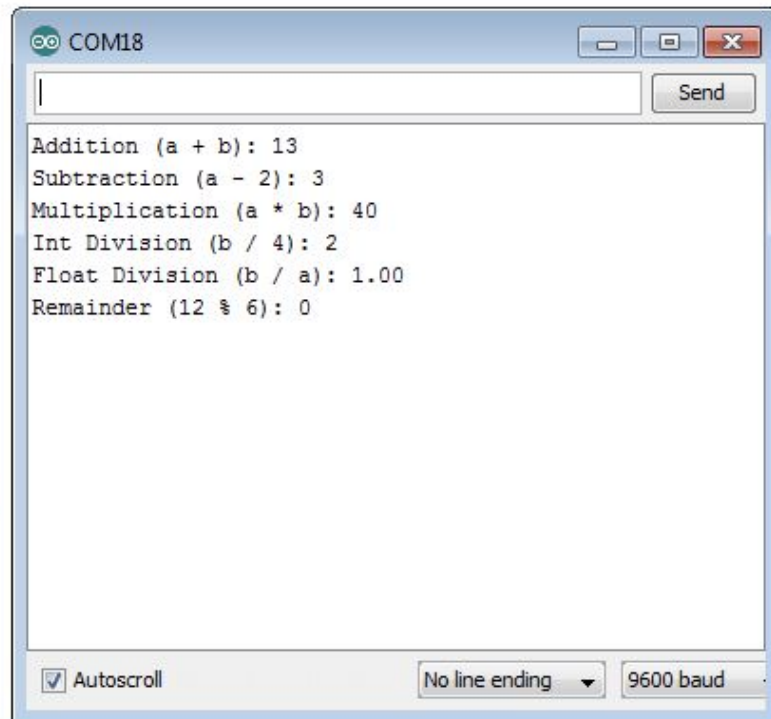
Serial.print("Subtraction (a - 2): ");
result = a - 2;
Serial.println(result);

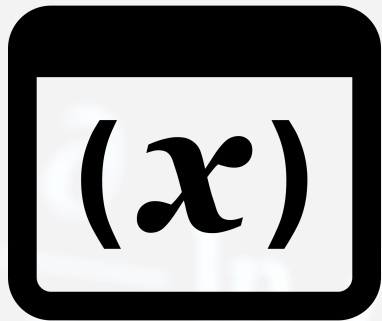
Serial.print("Multiplication (a * b): ");
result = a * b;
Serial.println(result);

Serial.print("Int Division (b / 4): ");
result = b / 4;
Serial.println(result);

Serial.print("Float Division (b / a): ");
result_fl = b / a;
Serial.println(result_fl);

Serial.print("Remainder (12 % 6): ");
result = 12 % 6;
Serial.println(result);
```





What can you DO with variables?

Variables are also good for helping clarify what things are doing in your program.

For example, it is good practice that instead of putting a *magic number* in, make it a variable with a good name and substitute that in for a number.

OK

```
digitalWrite(12, HIGH);
```

Better

```
int redLedPin = 12,  
digitalWrite(redLedPin, HIGH)
```

Let's make our **second arduino program!**

Let's attach an LED using the Arduino. Before we attach the LED, lets describe the program.

```
/*
```

```
Description: This program will alternate blinking between an  
external LED and the LED built into the Arduino
```

```
*/
```

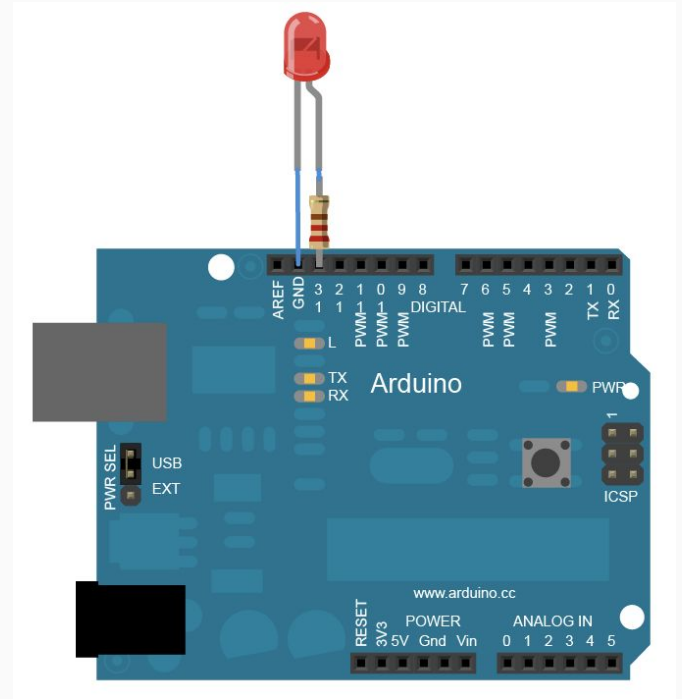
Let's make our second arduino program!

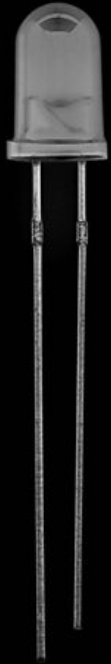
This program will alternate blinking between an external LED and the LED built into the Arduino

Pseudocode:

1. Turn on external LED
2. Turn off built in LED
3. Wait for 500 milliseconds
4. Turn off external LED
5. Turn on built in LED
6. Wait for 500 milliseconds
7. Return to step 1

Using the breadboard, connect one of the LEDs to the Arduino. Pick whichever color you want!





Diagram

Start writing the second **arduino** program

Step 1: Write a comment about what the program does at the very top. This will help you remember what the program does easily. Make sure you save it somewhere you will remember and name it something that makes sense.

Example comment)

```
// This program [does stuff]  
// Created by: Dr. Bechara  
// Date: 21 July 2017
```

Example name) arduinoBlinkAlternate.ino

Start writing the second **arduino** program

Since the pin numbers are just integers, let's set them as global variables. For example if I am using the **blue led** and it is plugged into **pin 10**, I would assign a global variable as follows.

```
int blueLED = 10
```

Just like before, in our **setup()** function we need to tell the arduino that we want to set the built in LED to be an **OUTPUT** (we want to talk to it not listen to it)

```
example) pinMode(LED_BUILTIN, OUTPUT);  
         pinMode(blueLED, OUTPUT);
```

In our **loop()** function we want to change the state of the LED from **LOW** (off) to **HIGH** (on) and flip back and forth between the two. We do that with **digitalWrite(pin, value)**

```
example) digitalWrite(LED_BUILTIN, HIGH);
```

We can use our friend **delay(time_in_ms)** to change the interval the light is on / off.

Start writing the second **arduino** program

In our `loop()` function we want to change the state of the LED from `LOW` (off) to `HIGH` (on) and flip back and forth between the two. We do that with `digitalWrite(pin, value)`

```
example) digitalWrite(redLED, LOW);
```

We can use our friend `delay(time_in_ms)` to change the interval the light is on / off.

```
example) delay(2000);
```

Now you figure out the rest!

What your program
should look like
(you can change the
delay to whatever
you want)

A screenshot of the Arduino IDE interface. The title bar shows the file name 'alternate_bulitin_singleLED' and the version 'Arduino 1.8.3'. The toolbar contains icons for checking, running, uploading, and downloading. The code editor shows the following code:

```
alternate_bulitin_singleLED
int blueLED=10;

void setup()
{
  pinMode(blueLED, OUTPUT);
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop()
{
  digitalWrite(blueLED, HIGH);
  digitalWrite(LED_BUILTIN, LOW);
  delay(1000);
  digitalWrite(LED_BUILTIN, HIGH);
  digitalWrite(blueLED, LOW);
  delay(1000);
}
```

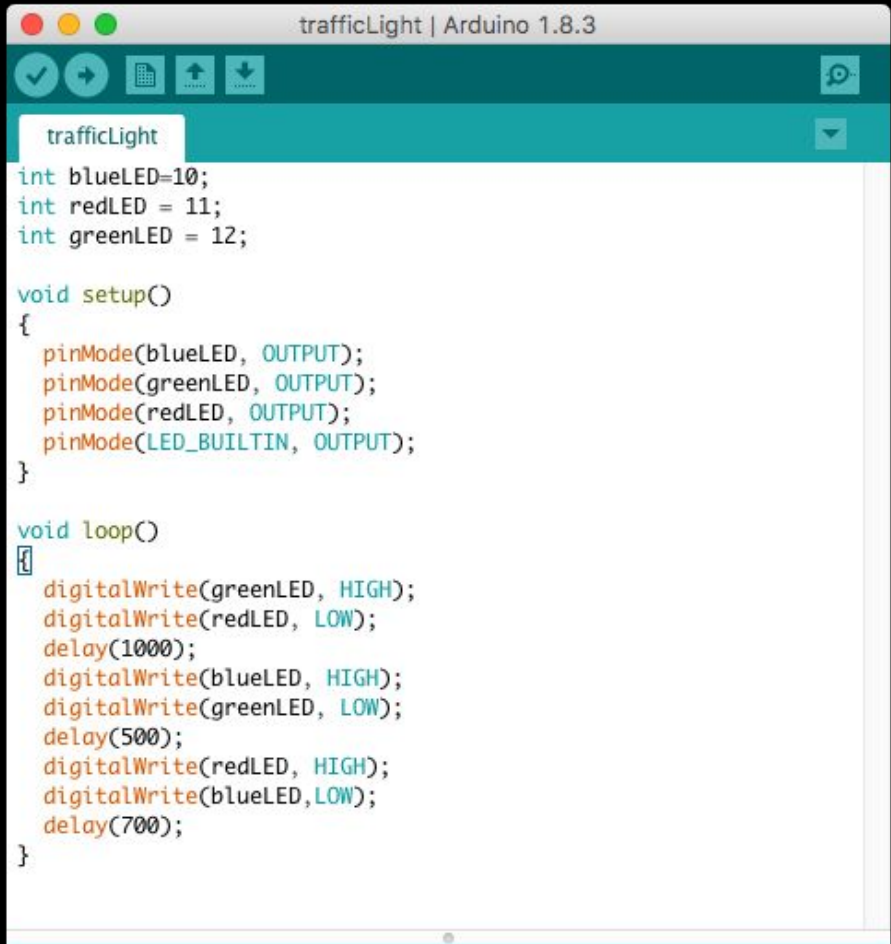

Your third arduino program

This time we are only going to give you the description. Make sure you go through the whole process. (Describe program in words, Write Pseudocode, Sketch Diagrams, Program, Test, Repeat). You can use what you have done so far to help! Be sure to save it and name it something that makes sense.

Description:

Hook up the other 2 LED lights. Alternate them like a traffic light. We don't have Red, Green, Yellow LEDs. Make due with what you have.

What your program
should look like
(you can change the
delay to whatever
you want)

A screenshot of the Arduino IDE interface. The title bar at the top reads "trafficLight | Arduino 1.8.3". Below the title bar is a toolbar with icons for checking, running, serial monitor, and file operations. The main text area contains the following code:

```
int blueLED=10;
int redLED = 11;
int greenLED = 12;

void setup()
{
  pinMode(blueLED, OUTPUT);
  pinMode(greenLED, OUTPUT);
  pinMode(redLED, OUTPUT);
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop()
{
  digitalWrite(greenLED, HIGH);
  digitalWrite(redLED, LOW);
  delay(1000);
  digitalWrite(blueLED, HIGH);
  digitalWrite(greenLED, LOW);
  delay(500);
  digitalWrite(redLED, HIGH);
  digitalWrite(blueLED, LOW);
  delay(700);
}
```

Your fourth arduino program

Description:

Let's count the number of times a certain LED is flashing and display that to the serial port.

Once the LEDs have been flashed 10 times, lets tell the user the program is done (via the serial port) and then STOP flashing

Start writing the fourth **arduino** program

Step 1: Like always, include a comment that describes your program

Step 2: As we have done before, let's set up int variables for all the pins. (the numbers to the right are just for illustration, yours are probably different!)

Step 3: Now we want to count. So setup an int for counting.

Step 4: Every time the LED flashes, we need to count it and display it to the serial port...

Step 5: Once LEDs have been flashed 10 times, the program should stop and all LEDs should be off

What are we missing?

We need to know how many times the LED has been flashed and then STOP after a certain number of times.

To do this we need conditionals and comparison operators!

Conditionals evaluate to either TRUE or FALSE

```
if (someVariable > 50)
{
    // do something here
}
```

Comparison Operators

`x == y` (x is equal to y)

`x != y` (x is not equal to y)

`x < y` (x is less than y)

`x > y` (x is greater than y)

`x <= y` (x is less than or equal to y)

`x >= y` (x is greater than or equal to y)

Consider the following. Let: `x = 4`, `y = 5`

`x == y` False

`x != y` True

`x < y` True

`x > y` False

`x <= y` True

`x >= y` False

What if you want something else to happen?

You can use an `if else` or an `else!`

```
if (someVariable > 50)
{}

else if (variable2 < 20)
{}

else
{}

```

Finish up the fourth
arduino program

Arduino Functions

Very similar to math class functions. Take an input and do something

Anatomy of a C function

Datatype of data returned,
any C datatype.

"void" if nothing is returned.

Function name

Parameters passed to
function, any C datatype.

```
int myMultiplyFunction(int x, int y){  
  int result;  
  result = x * y;  
  return result;  
}
```

Return statement,
datatype matches
declaration.

Curly braces required.

- You “call” functions in either the loop or setup function
- Functions do not HAVE to have inputs or outputs



Origins of communication

The History



Who: Named after Samuel F.B. Morse

When: 1836

What: Originally designed to transmit numbers, but was expanded to include the alphabet in 1840 by Alfred Vail.

Most commonly used letters were given the shortest sequences.

Why: Used to communicate before it was possible to transmit voice

How: System sent electric pulses along wires by closing a switch. Resulted in clicking sound on the receiving end

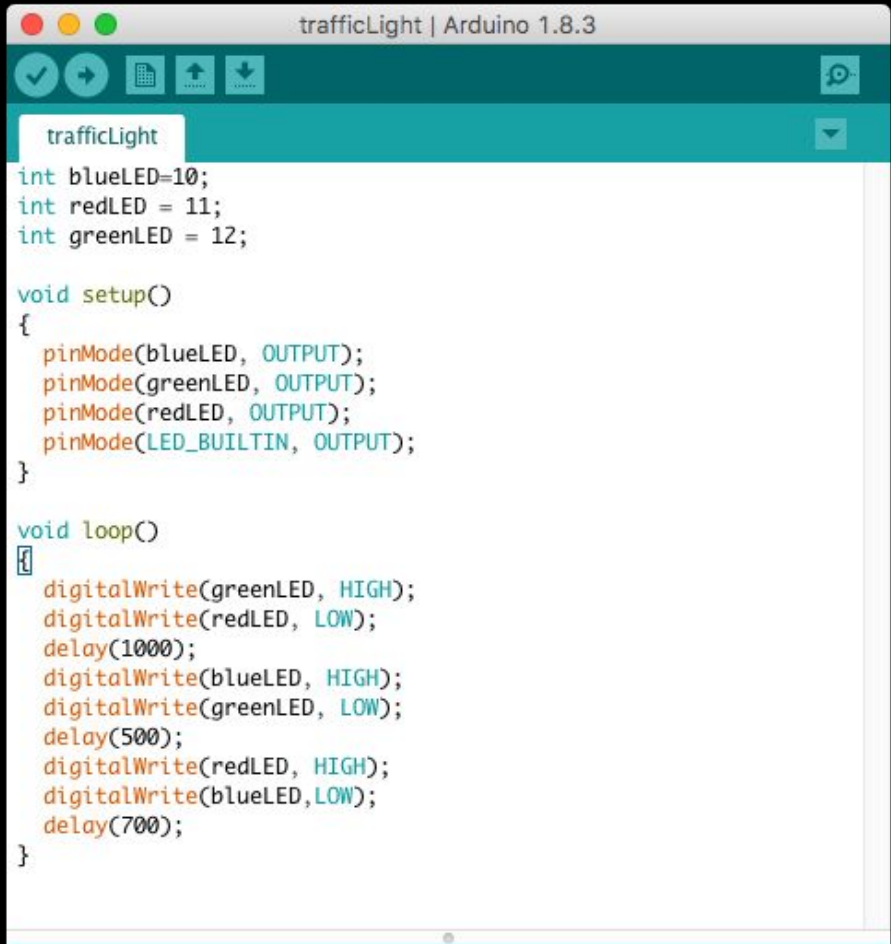
A	● ———	N	——— ●
B	——— ● ● ●	O	——— ——— ———
C	——— ● ——— ●	P	● ——— ——— ●
D	——— ● ●	Q	——— ——— ● ———
E	●	R	● ——— ●
F	● ● ——— ●	S	● ● ●
G	——— ——— ●	T	———
H	● ● ● ●	U	● ● ———
I	● ●	V	● ● ● ———
J	● ——— ——— ———	W	● ——— ———
K	——— ● ———	X	——— ● ● ———
L	● ——— ● ●	Y	——— ● ——— ———
M	——— ———	Z	——— ——— ● ●

TELEGRAPH !

Dit = .5 seconds
Dash = 1.5 seconds

Break between letters = 3 seconds
Break after the word = 5 seconds

What your program
should look like
(you can change the
delay to whatever
you want)

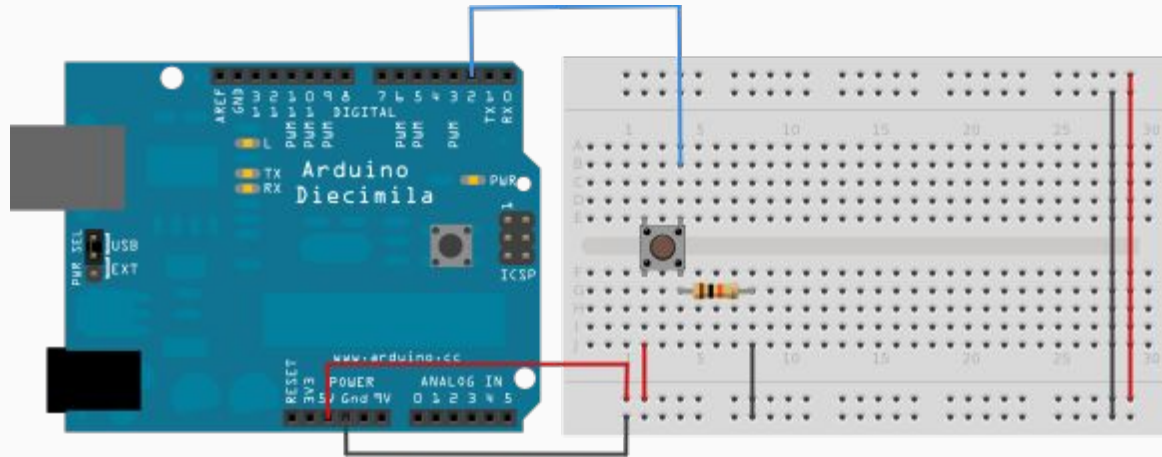
A screenshot of the Arduino IDE interface. The title bar at the top reads "trafficLight | Arduino 1.8.3". Below the title bar is a toolbar with icons for checking, running, serial monitor, and file operations. The main text area contains the following C++ code:

```
int blueLED=10;
int redLED = 11;
int greenLED = 12;

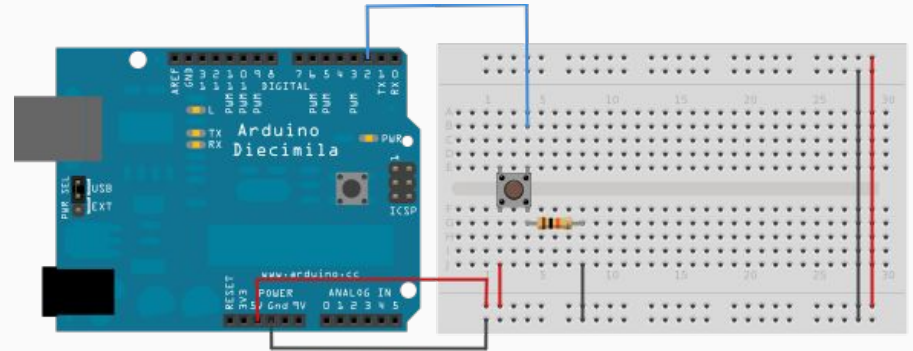
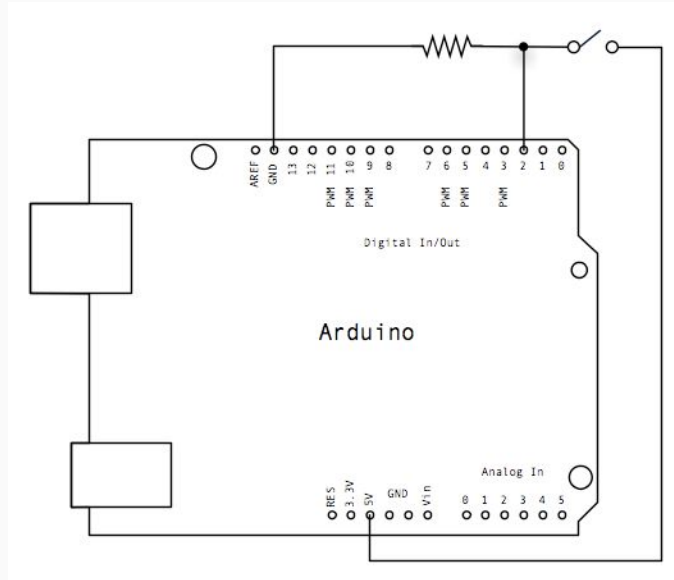
void setup()
{
  pinMode(blueLED, OUTPUT);
  pinMode(greenLED, OUTPUT);
  pinMode(redLED, OUTPUT);
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop()
{
  digitalWrite(greenLED, HIGH);
  digitalWrite(redLED, LOW);
  delay(1000);
  digitalWrite(blueLED, HIGH);
  digitalWrite(greenLED, LOW);
  delay(500);
  digitalWrite(redLED, HIGH);
  digitalWrite(blueLED, LOW);
  delay(700);
}
```

Leave everything where it is unless
you need to make room.
Let's add a button.



Diagram



Your third arduino program

This time we are only going to give you the description. Make sure you go through the whole process. (Describe program in words, Write Pseudocode, Sketch Diagrams, Program, Test, Repeat). You can use what you have done so far to help! Be sure to save it and name it something that makes sense.

Description:

Hook up the other 2 LED lights. Alternate them like a traffic light. Some of you will not have Red, Green, Yellow LEDs. Make due with what you have.

Your fourth arduino program

Description:

What do you want your button to do? You should have 3 LEDs connected. Your button just acts as an input. Once the arduino registers the input, it can do something.

Everyone's program should be a little different.

Pseudocode our fourth arduino program!

Everyone's program will be different. Mine flashes all the LED's when the button is pushed.

Pseudocode:

1. Turn all LEDs off
2. If the button is pressed
 - a. Turn on all LEDs
 - b. Wait 500 milliseconds
 - c. Turn off all LEDs
3. Return to step 1

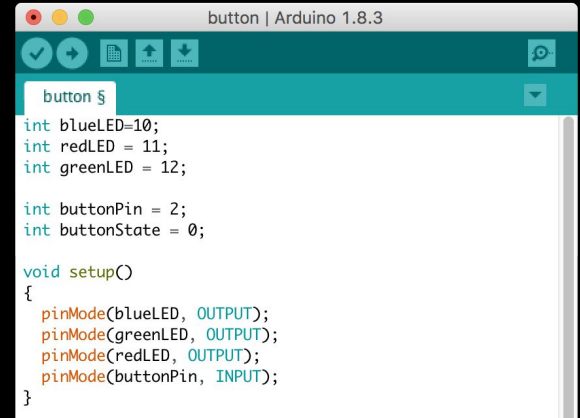
Start writing the fourth **arduino** program

Step 1: Like always, include a comment that describes your program

Step 2: As we have done before, let's set up int variables for all the pins. (the numbers to the right are just for illustration, yours are probably different!)

Step 3: Now we have a button. So let's add 2 new variables. One for the button pin, and one for its "state"

Step 4: In the setup, we make the LEDs OUTPUT as usual. But this time we need to make the button...

A screenshot of the Arduino IDE interface. The title bar reads "button | Arduino 1.8.3". The menu bar includes "File", "Edit", "Tools", and "Window". The toolbar shows icons for opening, saving, and running. The code editor contains the following code:

```
button §  
int blueLED=10;  
int redLED = 11;  
int greenLED = 12;  
  
int buttonPin = 2;  
int buttonState = 0;  
  
void setup()  
{  
  pinMode(blueLED, OUTPUT);  
  pinMode(greenLED, OUTPUT);  
  pinMode(redLED, OUTPUT);  
  pinMode(buttonPin, INPUT);  
}
```

What are we missing?

We need to have the Arduino know when the button is pressed. How can we do that?

Conditionals and comparison operators!

Conditionals evaluate to either TRUE or FALSE

```
if (someVariable > 50)
{
    // do something here
}
```

Comparison Operators

`x == y` (x is equal to y)

`x != y` (x is not equal to y)

`x < y` (x is less than y)

`x > y` (x is greater than y)

`x <= y` (x is less than or equal to y)

`x >= y` (x is greater than or equal to y)

Consider the following. Let: `x = 4`, `y = 5`

`x == y` False

`x != y` True

`x < y` True

`x > y` False

`x <= y` True

`x >= y` False

Lets look at the rest of the fourth arduino program

Now we have a statement that checks the state of the button
(HIGH corresponds to being pressed)

IF it is pressed, something happens.

A screenshot of the Arduino IDE interface. The title bar reads "button | Arduino 1.8.3". The code editor shows the following code:

```
int blueLED=10;
int redLED = 11;
int greenLED = 12;

int buttonPin = 2;
int buttonState = 0;

void setup()
{
  pinMode(blueLED, OUTPUT);
  pinMode(greenLED, OUTPUT);
  pinMode(redLED, OUTPUT);
  pinMode(buttonPin, INPUT);
}

void loop()
{
  buttonState = digitalRead(buttonPin);

  if (buttonState == HIGH)
  {
    digitalWrite(greenLED, HIGH);
    digitalWrite(redLED, HIGH);
    digitalWrite(blueLED, HIGH);
    delay(500);
    digitalWrite(greenLED, LOW);
    digitalWrite(redLED, LOW);
    digitalWrite(blueLED, LOW);
  }
```

What if you want something to happen if the button isn't pressed?

You can use an `if else` or an `else!`

```
if (someVariable > 50)
{
}

else if (variable2 < 20)
{
}

else
{
}
```

Finish up the fourth arduino program



```
button | Arduino 1.8.3

button $

int blueLED=10;
int redLED = 11;
int greenLED = 12;

int buttonPin = 2;
int buttonState = 0;

void setup()
{
  pinMode(blueLED, OUTPUT);
  pinMode(greenLED, OUTPUT);
  pinMode(redLED, OUTPUT);
  pinMode(buttonPin, INPUT);
}

void loop()
{
  buttonState = digitalRead(buttonPin);

  if (buttonState == HIGH)
  {
    digitalWrite(greenLED, HIGH);
    digitalWrite(redLED, HIGH);
    digitalWrite(blueLED, HIGH);
    delay(500);
    digitalWrite(greenLED, LOW);
    digitalWrite(redLED, LOW);
    digitalWrite(blueLED, LOW);
  }
}
```

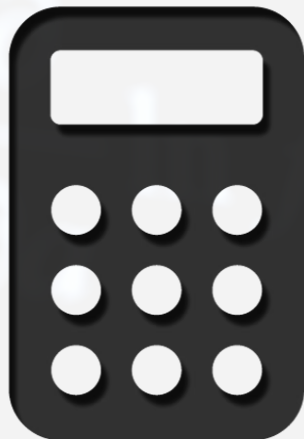

Your fifth arduino program

Description:

For this program lets tie everything together. Lets make the button change the lit LED to a random LED. Then, lets “count” the number of button presses. If the press is a multiple of 5 i.e. (5, 10, 15, 20, etc) let's turn on all three LEDs.

Let's also output the number of presses to the Serial port.

Remember this slide?



All sorts of things! You can perform arithmetic just like you would with a calculator.

- + Addition
- − Subtraction
- / Division
- * Multiplication
- % Remainder

Pseudocode our fifth arduino program!

Pseudocode:

Setup: Choose random LED and light. Start Serial.

1. Light up random LED
2. If the button is pressed, increment buttonCounter, print to serial
 - a. If (buttonPresses % 5 == 0)
 - i. Blink all LEDs*
 - b. Else
 - i. Turn off LED(s) that is on
 - ii. Turn on random LED ** for now, don't worry if it is the same LED as before*
 - iii. Remember what LED is on
3. Return to step 1

Arduino Random

The random() function generates pseudo-random numbers

Syntax

`random(max)` OR `random(min, max)`

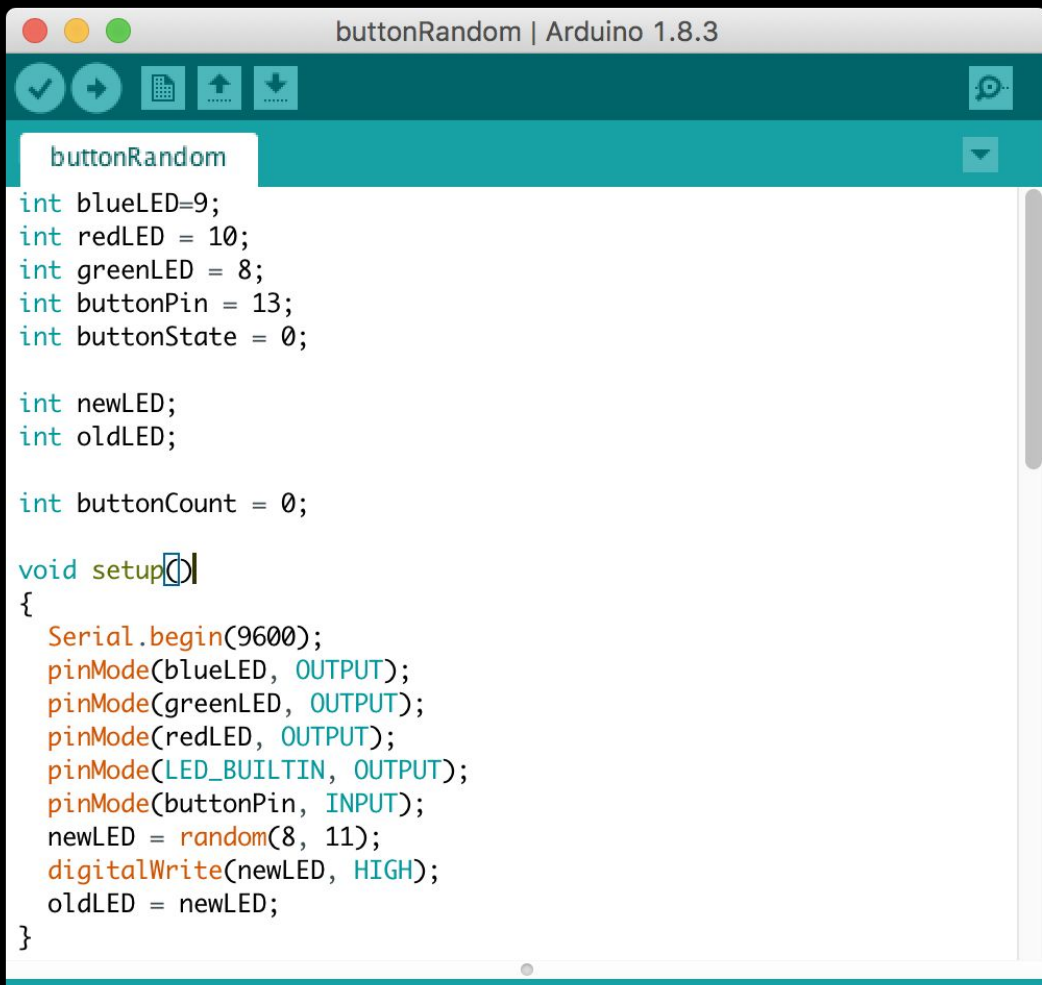
Returns

The function RETURNS a random number between `min` and `max - 1`

Start writing the fifth arduino program

This one is pretty tough, you can see what I called variables and how I setup the program on the right.

Don't forget to comment and sketch!

A screenshot of the Arduino IDE interface. The title bar at the top reads 'buttonRandom | Arduino 1.8.3'. Below the title bar is a toolbar with icons for checking, running, serial monitor, and file operations. The main text area contains the following code:

```
buttonRandom

int blueLED=9;
int redLED = 10;
int greenLED = 8;
int buttonPin = 13;
int buttonState = 0;

int newLED;
int oldLED;

int buttonCount = 0;

void setup()
{
  Serial.begin(9600);
  pinMode(blueLED, OUTPUT);
  pinMode(greenLED, OUTPUT);
  pinMode(redLED, OUTPUT);
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(buttonPin, INPUT);
  newLED = random(8, 11);
  digitalWrite(newLED, HIGH);
  oldLED = newLED;
}
```



Take a minute to reflect.



Tip

Write down whatever you want.

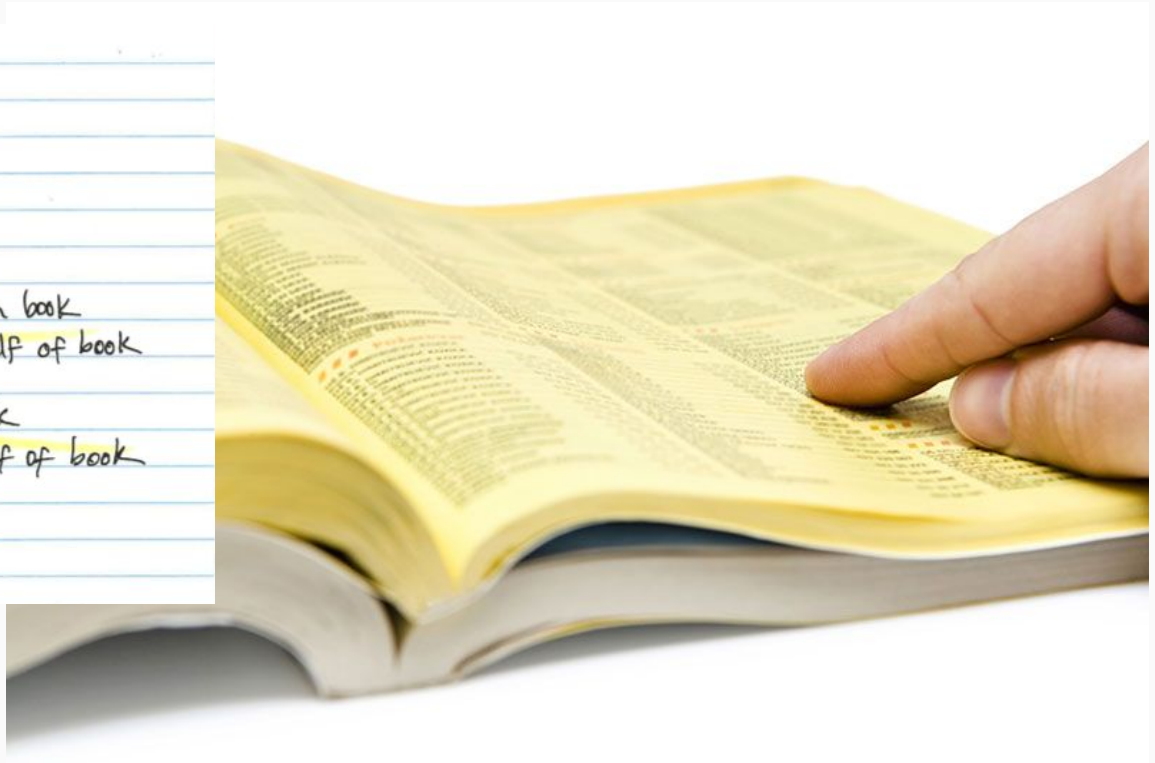
What did you learn today?

What do you hope to learn tomorrow?

Pseudocode of Phonebook App

pseudocode

- 1) Pick up phonebook
- 2) Open to middle of phonebook
- 3) Look at names
- CONDITIONS → 4) If "Smith" is among names
- decision pts ← 5) Call mike
- 6) else if "Smith" is earlier in book
- 7) open to middle of left half of book
- 8) go to line 3
- 9) else if Smith is later in book
- 10) open middle of right half of book
- 11) go to line 3
- 12) else
- 13) → give up



Write your own pseudocode

Write a program that switches an LED
light...



What are some things we need to know about
computer language to accomplish the pseudo
code you wrote about?

Arduino Programming Language Basics

Need slides on

- Commenting
- Data Types
- Functions
-

Program 1: Blink built in light

Show them how to hook up external LED

Program 2: Turn on external LED (alternate between built in and external)

Show them how to hook up rest of LED

Program 3: Make traffic light

Show how to hook up a button

Program 4: Use button

Show them the difference between advancedButton with without comments

Talk about for loops

Serial.begin

Program 5: Button Counter

For Loops

Program 6: Whatever they want. Go through FULL process, write it out in pseudocode, etc.

Every program should have notebook pages that coorespond for

- Pseudocode
- Diagram
- etc