MSSM STEM Summer Camp
Learn to Code

Basic Language Review

Low Level – talk directly to the computer, hard to read (assembly)
High Level – easy to follow, but need a compiler (python)
Mid Level – Somewhere in between – C, C#, C++

Object Oriented – chunks or objects
Functional – linear, straight logic

Python	**Easy to read, HL, huge community, several uses**
C	**Created in 1972, structured mid level, industrial automation and system apps**
C#	**Combination of C, C++, and Java, making Windows apps**
C++	**OO ML, extension of C, Adobe and CGI (Jurassic Park, Avengers, etc)**
Java	**Slower than C++, very secure often used for banking apps and Android apps**
Javascript	**Client-side browser, HL OO, paired with HTML & CSS, no relation to Java**
Ruby	**Beginner friendly, uses a framework called Rails, NASA, GOOGLE**
PHP	**Dynamic webpages, databases, created by accident in C**
Swift	**Easier way to create Apple and iPhone apps, easier than Obj-C**
Go	**Google created for efficiency, combines many features**

Why Processing
It's easy to learn. Free, and simple to install. Instant results.
Other languages better, but longer to set up.
Programs for programming are usually called IDE.

If you continue to pursue programming, I suggest Python. It's growing in popularity, has tons of uses, and there is a huge community for learning. Many classes are free, but some of the paid ones are even better. The future data scientist shortage will need python-savvy programmers.

size(); *(white space is OK, but NOT capital letter, color change)*

Screen size x width, y is height

```
size(800, 600);

rect(100, 100, 200, 50);        (x location, y location, width, height)

// comments are great for remembering

rect(110,110, 200, 50);         linear down, it overlaps
```

very basic, not a real program. The core of a processing program has two structures

```
void setup() {          runs once and only once

}

void draw() {           loops at 60 frames per second (FPS)

}
```

Void means it will not send anything back. Functions usually perform a task and return a result. A Shipping Function would take in a person's address and calculate it with shipping rates and return the result. These are void functions that return nothing.

Setup does exactly that. Sets up some information. It is read once and only once each time a program runs.

The draw() function has many hidden benefits that control things so we don't have to worry.

```
void setup() {
        size(1000,800);
}

void draw() {
        rect(100,100,200,100);      // puts a 200 wide by 100 high rectangle 100 from left
                                    // and 100 from top
}
```

```
void draw() {
```

```
        rect(mouseX,100,200,100);    // follows the mouse location left and right only
}
```

```
void draw() {
        rect(mouseX,mouseY,200,100);  // follows the mouse everywhere, but never clears the
                                      // screen
}

void draw() {
        background(0);
        rect(mouseX,mouseY,200,100);        // follows the mouse and clears the screen
                                            // 60 times per second
}
```

4 squares side by side with comments and color filled

```
void setup() {
        size(400,400);
}

void draw() {
        background(128);
        fill(0);
        rect(100,100,100,100); //top left
        fill(255);
        rect(200,100,100,100); //top right
        rect(100,200,100,100); //lower left
        fill(0);
        rect(200,200,100,100); //lower right
}
```

Add variables to make changes easier

Int	Float	Boolean	Char	String
Integer Whole number	Floating Point Decimal number	True or False	Single text character	Words or sentences

```
Int wNum = 100;
Int hNum = 100;

void setup() {
	size(400,400);
}

void draw() {
	background(128);
	fill(0);
	rect(100,100, wNum, hNum); //top left
	fill(255);
	rect(200,100, wNum, hNum); //top right
	rect(100,200, wNum, hNum); //lower left
	fill(0);
	rect(200,200, wNum, hNum); //lower right
}
```

Make Things Move:

```
Void draw() {
	noStroke();
	fill(random(255), random(255), random(255));
	ellipse(mouseX, mouseY, 100,100);
}
```

Challenge:

5x5 checkboard with black squares in all corners

```
int blk = 0;
int wte = 255;
int skip = 0;

void setup() {
 size(700,700);
}
```

```
void draw() {

// first, the longer hard coded way.
 fill(blk);
 rect(100,100,100,100);
  fill(wte);
 rect(200,100,100,100);
  fill(blk);
 rect(300,100,100,100);
  fill(wte);
 rect(400,100,100,100);
  fill(blk);
 rect(500,100,100,100);
  fill(wte);
 rect(100,200,100,100);
 fill(blk);
 rect(200,200,100,100);
  fill(wte);
 rect(300,200,100,100);
 fill(blk);
 rect(400,200,100,100);
  fill(wte);
 rect(500,200,100,100);
 fill(blk);
 rect(100,300,100,100);
  fill(wte);
 rect(200,300,100,100);
 fill(blk);
 rect(300,300,100,100);
  fill(wte);
 rect(400,300,100,100);
  fill(blk);
 rect(500,300,100,100);
    fill(wte);
 rect(100,400,100,100);
  fill(blk);
 rect(200,400,100,100);
    fill(wte);
 rect(300,400,100,100);
  fill(blk);
 rect(400,400,100,100);
    fill(wte);
 rect(500,400,100,100);
    fill(blk);
 rect(100,500,100,100);
    fill(wte);
 rect(200,500,100,100);
  fill(blk);
```

```
 rect(300,500,100,100);
   fill(wte);
 rect(400,500,100,100);
  fill(blk);
 rect(500,500,100,100);

// for longer block comments use /*  */
// next the shorter more concise way, but how to change the colors?
/*
for (int i = 100; i < 600; i = i+100) {
  for (int j = 100; j < 600; j = j+100) {
    fill(skip);
     rect(i,j,100,100);
  }
}
*/
}
```