CS 101
Variables in Processing

Lecture 6
Draw three circles to resemble this . . .
Draw three circles to resemble this . . .

size(400, 300);
ellipse(75, 60, 80, 80);
ellipse(175, 60, 80, 80);
ellipse(275, 60, 80, 80);
Variables

- A **variable** stores a value into the computer’s memory so that it can be used later-on in a program
- A single variable can be used many times
- The value that is stored in a variable can change over time
Storing Numbers

- A **variable** can store many types of values
- For the time being, we are only going to use them for storing whole numbers
  - These are called **integers**!
Variables

- What does a variable look like?
- How does one “create” a variable?
int name = 123;
All variables begin with the type!
Sticking to only integers for now...

Left of equals: the type and name
Right of equals: the value!
Every variable that you create should have the equals!

Every line ends with a semi-colon (with exceptions)

The name of the variable.
Whatever you want (with some restrictions)

Choose a number to “store” inside of the variable

int name = 123;
In programming, this is referred to as **declaring a variable**

The *name* can be anything, with a few restrictions

- Can have upper/lower case letters, numbers, and a few special symbols such as underscore and dashes.
- May *not* begin with a number
Using Variables

- You can use a variable (with an integer number stored in it) to position shapes!

```java
int xCoord = 57;
int yCoord = 92;
ellipse(xCoord, yCoord, 100, 200);
```
What values are needed?

size(300, 300);
int xCoord = ? ;
int yCoord = ? ;
int wVal = ? ;
int hVal = ? ;

rect(xCoord, yCoord, wVal, hVal);
What values are needed?

```plaintext
size(300, 300);
int xCoord = 50;
int yCoord = 150;
int wVal = 200;
int hVal = 75;

rect(xCoord, yCoord, wVal, hVal);
```
Back to the three circles

- What if I want to . . .
  - Center the circles?
  - Adjust the size of all three?
  - Move them to the bottom of the canvas?
Resize the circles

Take your program from earlier, and change so that the circles are larger and barely touch each-other
Resize the circles

size(400, 300);
ellipse(75, 60, 100, 100);
ellipse(175, 60, 100, 100);
ellipse(275, 60, 100, 100);
Resize the circles

```javascript
size(400, 300);
ellipse(75, 60, 100, 100);
ellipse(175, 60, 100, 100);
ellipse(275, 60, 100, 100);
```

How many arguments did you have to change to make it look like this?
Resize the circles

```plaintext
size(400, 300);
ellipse(75, 60, 100, 100);
ellipse(175, 60, 100, 100);
ellipse(275, 60, 100, 100);
```
Resize the circles

size(400, 300);
ellipse(75, 60, 80, 80);
ellipse(175, 60, 80, 80);
ellipse(275, 60, 80, 80);

OR ...
Resize the circles

int diameter = 80;
size(400, 300);
ellipse(75, 60, diameter, diameter);
ellipse(175, 60, diameter, diameter);
ellipse(275, 60, diameter, diameter);

int diameter = 100;
size(400, 300);
ellipse(75, 60, diameter, diameter);
ellipse(175, 60, diameter, diameter);
ellipse(275, 60, diameter, diameter);
Other variables?

```plaintext
int diameter = 100;
size(400, 300);
ellipse(75, 60, diameter, diameter);
ellipse(175, 60, diameter, diameter);
ellipse(275, 60, diameter, diameter);
```

How else should we use variables in this program?
Other variables?

```cpp
int diameter = 100;
int y = 60;
size(400, 300);
ellipse(75, y, diameter, diameter);
ellipse(175, y, diameter, diameter);
ellipse(275, y, diameter, diameter);
```
int diameter = 100;
int y = 60;
size(400, 300);
ellipse(75, y, diameter, diameter);
y = 100;
ellipse(175, y, diameter, diameter);
y = 140;
ellipse(275, y, diameter, diameter);
What will this program produce?

```
size(400, 400);
int r = 50;
int g = 100;
int b = 150;
fill(r, g, b, 100);
rect(50, 50, 200, 200);
r = r + 100;
fill(r, g, b, 100);
rect(100, 100, 200, 200);
g = g + 100;
fill(r, g, b, 100);
rect(150, 150, 200, 200);
```
What will this program produce?

```plaintext
size(400, 400);
int r = 50;
int g = 100;
int b = 150;
fill(r, g, b, 100);
rect(50, 50, 200, 200);
r = r + 100;
fill(r, g, b, 100);
rect(100, 100, 200, 200);
g = g + 100;
fill(r, g, b, 100);
rect(150, 150, 200, 200);
```
Math

- Processing has built-in capabilities to do basic math operations (Arithmetic) with integers and integer variables
- This is often useful when positioning shapes on the canvas
- The basic math operations are:
  - `+` addition
  - `-` subtraction
  - `*` multiplication
  - `/` division
Math - part I

After each line runs, what value will be stored in each of these variables?

```java
int number1 = 50 + 100 + 200;

int number2 = 500 / 100;

int number3 = 5 * 5 * 2;

int number4 = 100 - 70;
```
Math - Part II

int a = 200;
int b = 10;
int c = 2;

int valueX = a / b;

int valueY = b + a + c;

int valueZ = a * c;

After all of these lines run, what value will be stored in each of these variables?
Order of Operations

- What will the variable `sum` store in the expression below?
  ```
  int sum = 5 + 3 * 4 - 2;
  ```
- The order of operations matters!
- Order of operations: Parentheses, Exponents, Multiplication, Division, Addition, Subtraction
  - PEMDAS
Order of Operations

After all of these lines run, what value will be stored in each of these variables?

```plaintext
int valueA = 100 + 200 * 2 / 4;

int valueB = (5 + 5) * 3 - 2;

int valueC = 5 - 100 / (7 + 3) - 10;
```
Operators and Operands

- These arithmetic symbols are called **operators**
- The numbers and/or variables that are a part of the math equation are called the **operands**
- A little terminology review...

```c
int name = a + b + 100;
```
Draw a face

- Write a program that draws a very simple face, similar to the one pictured
- Should have face, eyes, mouth (colors are up to you)
- **We will revisit this, so save the code!**
Draw a face

size(400, 400);

fill(140, 100, 80);
rect(100, 100, 200, 200);
fill(0, 255, 0);
ellipse(160, 150, 20, 20);
ellipse(240, 150, 20, 20);
fill(100, 200, 255);
rect(150, 220, 100, 30);
Built-In Variables

- Processing has a number of variables that are automatically declared and modified/updated behind-the-scenes.
- You can use these variables too!
- A few that we’ll be using are:
  - `width`: The width of the canvas
  - `height`: The height of the canvas
  - `mouseX`: The current mouse X coordinate
  - `mouseY`: The current mouse Y coordinate
width and height

• What will this program produce?

```plaintext
textsize(200, 200);

int d = 100;

ellipse(0, 0, d, d);
ellipse(width, 0, d, d);
ellipse(0, height, d, d);
ellipse(width, height, d, d);
```
width and height

- What will this program produce?

```plaintext
size(200, 200);
int d = 100;
ellipse(0, 0, d, d);
ellipse(width, 0, d, d);
ellipse(0, height, d, d);
ellipse(width, height, d, d);
```
width and height

- What will this program produce?

```javascript
size(200, 200);
strokeWeight(10);
line(0, 0, width, height);
line(width, 0, 0, height);
ellipse(100, 100, 70, 70);
```
width and height

- What will this program produce?

```javascript
size(200, 200);
strokeWeight(10);
line(0, 0, width, height);
line(width, 0, 0, height);
ellipse(100, 100, 70, 70);
```
Setup and Draw

- From here-on-out, you should put your code within one of two sections of the processing program
- **setup**: a function where you put any code that should be run just once, right at the beginning
- **draw**: a function where you put code that you want to use to do all of your drawing, animating, and interacting
void setup() {
  size(480, 300);
}

void draw() {
  ellipse(140, 180, 160, 160);
  ellipse(120, 140, 60, 80);
  ellipse(110, 120, 30, 40);
}
Setup and Draw

- Typically, you will set the size of the processing canvas and the background color in `setup`
  - You’ll put other stuff in there too, as we learn more features
- Most of the “interesting” code will go inside of `draw`
  - All of the code you draw with!
Setup and Draw

- Everything that you put inside of `setup` is only run once, right when your program first begins.
- Everything you put in `draw` is run repeatedly, forever!
Setup and draw

What will this program produce?

```java
void setup() {
    size(500, 500);
    background(200, 230, 255);
}

void draw() {
    strokeWeight(2);
    fill(100, 255, 200);
    rect(100, 100, 50, 50);
}
```
Setup and draw

**OK, how about this program?**

**What have we changed?**

```cpp
int number = 50;

void setup() {
    size(500, 500);
    background(200, 230, 255);
}

void draw() {
    strokeWeight(2);
    fill(100, 255, 200);
    rect(number, number, 50, 50);
    number = number + 1;
}
```
Setup and draw

What will this program produce?

```java
void setup() {
    size(500, 500);
    background(200, 230, 255);
    stroke(0, 200, 255);
}

void draw() {
    strokeWeight(5);
    line(0, 0, mouseX, mouseY);
}
```
Setup and draw
Setup and draw

How do I not leave a trail of lines?
Setup and draw

```java
void setup() {
  size(500, 500);
  stroke(0, 200, 255);
}

void draw() {
  background(200, 230, 255);
  strokeWeight(5);
  line(0, 0, mouseX, mouseY);
}
```
Setup and draw
Following the mouse

- Write a program that behaves in this way:
void setup() {
    size(500, 500);
}

void draw() {
    strokeWeight(2);
    fill(100, 200, 200);
    rect(mouseX, mouseY, 50, 50);
}
Move the face

- Change your face program so that the face moves with the mouse
void setup() {
    size(400, 400);
}

void draw() {
    background(200);
    fill(140, 100, 80);
    rect(mouseX, mouseY, 200, 200);
    fill(0, 255, 0);
    ellipse(mouseX + 60, mouseY + 50, 20, 20);
    ellipse(mouseX + 140, mouseY + 50, 20, 20);
    fill(100, 200, 255);
    rect(mouseX + 50, mouseY + 120, 100, 30);
}
Checking the value stored in a variable

- How can we check what value is stored in a variable?
  - Very useful when program is not behaving the way we want
Checking the value stored in a variable

- How can we check what value is stored in a variable?
  - Very useful when program is not behaving the way we want
- Use `println(variable_name);`
  - As the name suggests, this *prints* the value onto its own
    *line* (ln)
Printing a variable

- Modify this program so that you print out the X and Y coordinates every frame
Floats

- We can also create variables that store numbers with decimal values
- This is a different type called float
- We’ll talk more about why these are useful later

```java
float height = 6.1;
float weight = 177.45;
float speed = 45.302;
```
Processing - Materials

- Required Materials
  - GSWP - First half of Chapter 4