today

due: student surveys, studio 1 sketches, html and css

topic: processing and p5.js, github

tuesday

due: studio 1

the bottom line

you have to write and understand your own code

what is programming?

needs human beings to provide instructions (write **algorithms**)

need **specificity** - syntax; condition associated with logic

pseudo-code: get use to doing, plan out the logic in your own language

philosophy of **incremental development**/modular system:
put everything into one set of code all at once is kind of impossible
need lots of little pieces-write mini programs one at a time
fit the pieces together!

what is an algorithm?

instructions for achieving a task

the idea is like making muffins

1. Whisk dry ingredients
2. In another bowl, mix wet ingredients
3. Pour wet on top of dry and fold together
4. Scoop into muffin tins
5. Bake at 400 F
6. Remove from oven

what is a program?

...collection of algorithms

what is syntax?

the grammatical rules and structural patterns governing the ordered use of appropriate words and symbols for issuing commands, writing code, etc., in a particular software application or programming language

in Processing, write a tiff file to a folder called frames

```
saveFrame("frames/####.tif");
```

processing basics

all sketches have **`setup`** and **`draw`** functions

```
void setup() {
  //code here
}

void draw() {
  //code here gets executed at 60fps
}
```

default frameRate in draw() is 60 fps

pre-defined functions

both of these words, setup and draw, are functions that have been pre-defined by Processing as what to do **once** and what to do **over and over again**

we are further defining these two functions within
what we specify inside the curly brackets

```
void draw() {
 background(0);
 fill(255,0,0);
 rect (100,100,200,200);
}
```

code elements

`//` comments...use them :)

`/* */` multiline comments

`;` statement terminator

`,` comma, separates parameters or arguments of functions

`print()` print to console

`println()` print to console as block

/////////////////

`command-t` tidies code

commenting

your future self will thank the present self, if you build a habit to comment

// Here is an explanation.

/*
Here I have a lot to say. I have so much to say.
What do you think about what I am saying?
*/

readability of code

group things together that belong

put paragraph break to separate them

```
size(600,400);
background(135);

stroke(255, 0, 0);
fill(100);
rect(100, 100, 200, 20);
```

processing basics

start a sketch with the size() method for window size

default (undeclared) is 100 x 100 (in pixels)

```
void setup() {
  size(300,300);
}
```

size();

should always have hard numbers, not variables

good:

```
size (640, 320);
```

not good:

```
size (float, float);
```

size(); should always be the first line code

case sensitive

this:

```
size(300,300);
```

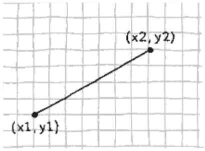
is not the same as:

```
Size(300,300);
```

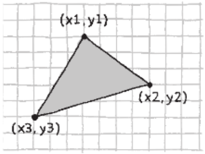
primitive shapes

point, line, triangle, quad, rect, ellipse, arc, bezier

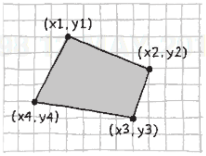
```
point(x,y);
```



```
line(x1, y1, x2, y2)
```

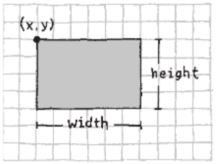


```
triangle(x1, y1, x2, y2, x3, y3)
```

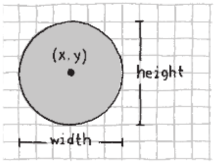


```
quad(x1, y1, x2, y2, x3, y3, x4, y4)
```

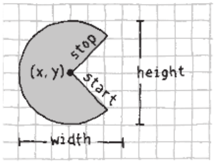
primitive shapes



```
rect(x, y, width, height)
```



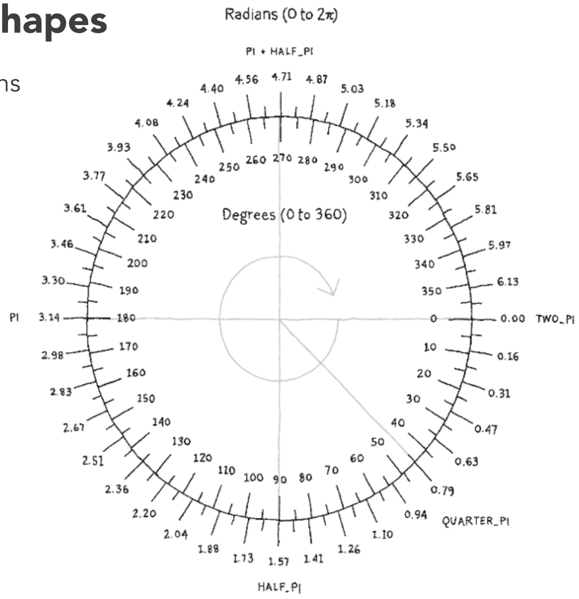
```
ellipse(x, y, width, height)
```



```
arc(x, y, width, height, start, stop)
```

primitive shapes

arcs & radians



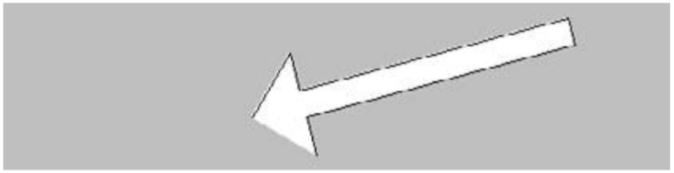
arcs & radians

If you prefer to use degree measurements, you can convert to radians with the *radians()* function. This function takes an angle in degrees and changes it to the corresponding radian value.

```
size(480, 120);
arc(90, 60, 80, 80, 0, radians(90));
arc(190, 60, 80, 80, 0, radians(270));
arc(290, 60, 80, 80, radians(180), radians(450));
arc(390, 60, 80, 80, radians(45), radians(225));
```

complex shapes

The *beginShape()* function signals the start of a new shape. The *vertex()* function is used to define each pair of x- and y-coordinates for the shape. Finally, *endShape()* is called to signal that the shape is finished.



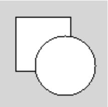
```
size(480, 120);
beginShape();
vertex(180, 82);
vertex(207, 36);
vertex(214, 63);
vertex(407, 11);
vertex(412, 30);
vertex(219, 82);
vertex(226, 109);
endShape();
```

drawing order

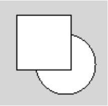
the program is drawn in the order it is written

default: what appears first in the code is the lowest layer

```
rect(15,15,50,50);
ellipse(60,60,55,55);
```

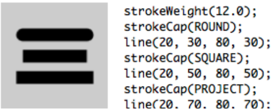


```
ellipse(60,60,55,55);
rect(15,15,50,50);
```

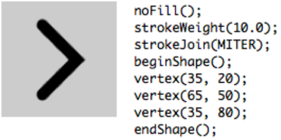


drawing attributes

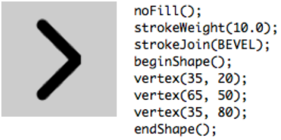
```
smooth(); // adds anti-aliasing

noSmooth(); // turns of anti-aliasing

ellipseMode(); // default

ellipseMode(CENTER); // draw from center

rectMode(); // default

rectMode(CENTER); // draw from center

strokeWeight(x); // x is amount in px
```



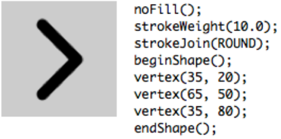
```
strokeWeight(12.0);
strokeCap(ROUND);
line(20, 30, 80, 30);
strokeCap(SQUARE);
line(20, 50, 80, 50);
strokeCap(PROJECT);
line(20, 70, 80, 70);
```



```
noFill();
strokeWeight(10.0);
strokeJoin(MITER);
beginShape();
vertex(35, 20);
vertex(65, 50);
vertex(35, 80);
endShape();
```



```
noFill();
strokeWeight(10.0);
strokeJoin(BEVEL);
beginShape();
vertex(35, 20);
vertex(65, 50);
vertex(35, 80);
endShape();
```

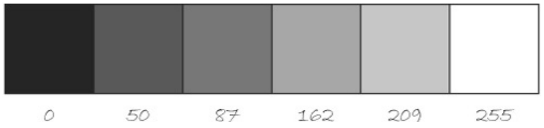


```
noFill();
strokeWeight(10.0);
strokeJoin(ROUND);
beginShape();
vertex(35, 20);
vertex(65, 50);
vertex(35, 80);
endShape();
```

color settings

```
stroke(x); // x is color

noStroke(); // turns off stroke

fill(x); // x is color

noFill(); // turns off fill

background(x); // x is background color

colorMode(); // accepts RGB or HSB
```

once an attribute is set, it remains active until set again



color

```
fill(x); // 1 value, grayscale

fill(x, a); // 2 values, grayscale and alpha

fill(x, y, z); // 3 values, rgb

fill(x, y, z, a); // 4 values, rgb and alpha

fill(#xxxxxx); // hexadecimal code

fill(#xxxxxx, a); // hexadecimal code and alpha
```

resetting color scale

each parameter has a default range of 0-255 representing the intensity of light

the range for alpha (transparency) is 0-255 (unless you change it)

```
colorMode(mode);
colorMode(mode, max);
colorMode(mode, max1, max2, max3);
colorMode(mode, max1, max2, max3, maxA);
```

```
colorMode(RGB, 100,100,100,100);
```

```
colorMode(HSB, 360, 100,100);
```

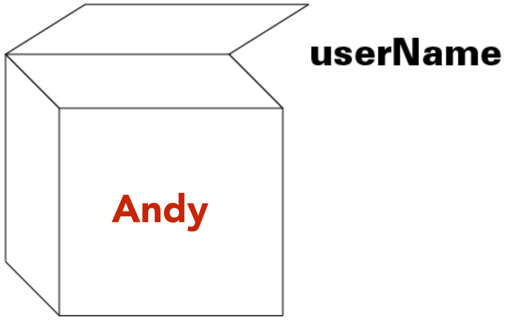
variables

variables store data for later recall (remember algebra)...

if x + y = 50, and we know y = 10, what is x?

variables

variables are like containers, or boxes, that store something (location in computer memory)



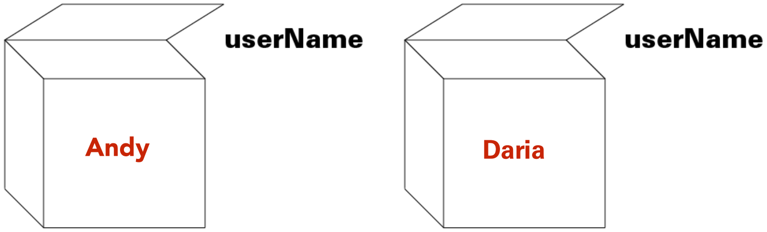
userName

Andy

variables

the contents can *vary*,

but each variable can only hold one piece of data at a time



userName

Andy

userName

Daria

variable (data) types

numbers **`(int, float, byte, short, long, double)`**

letters (**`char`**)

words, including spaces (**`String`**)

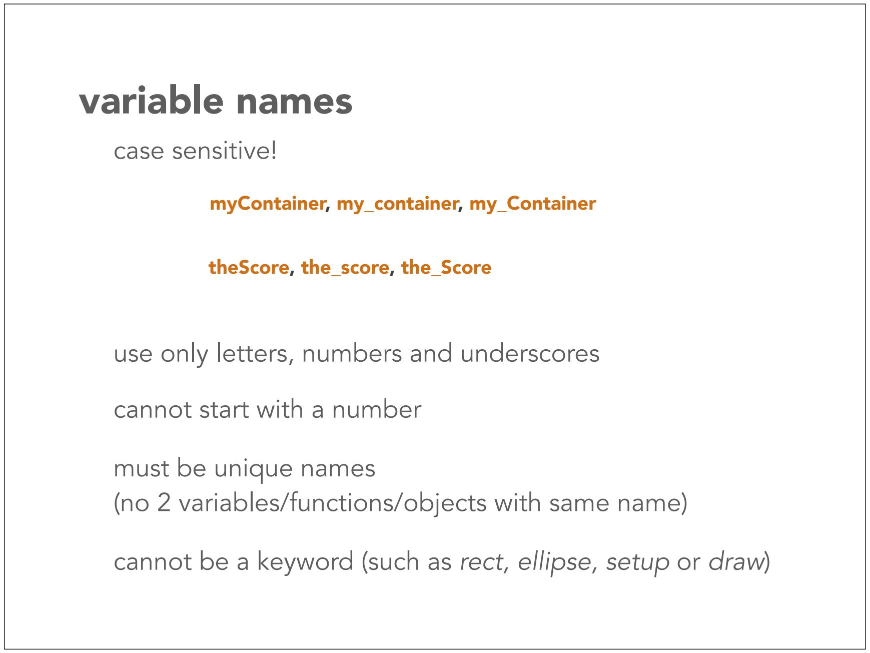
true or false (**`boolean`**)

colors

images

fonts

...and more



variable names

case sensitive!

myContainer, **my_container**, **my_Container**

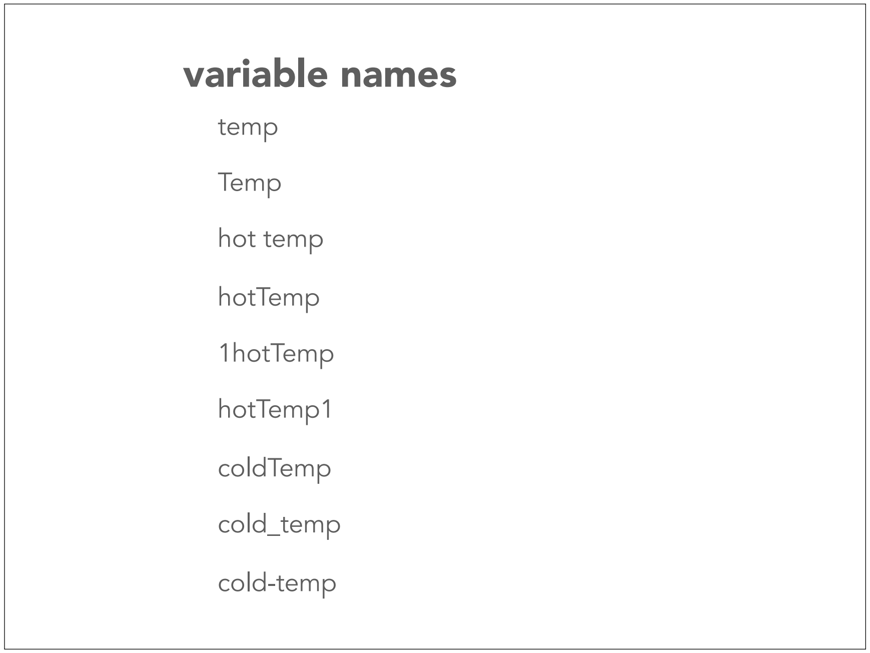
theScore, **the_score**, **the_Score**

use only letters, numbers and underscores

cannot start with a number

must be unique names
(no 2 variables/functions/objects with same name)

cannot be a keyword (such as *rect*, *ellipse*, *setup* or *draw*)



variable names

temp

Temp

hot temp

hotTemp

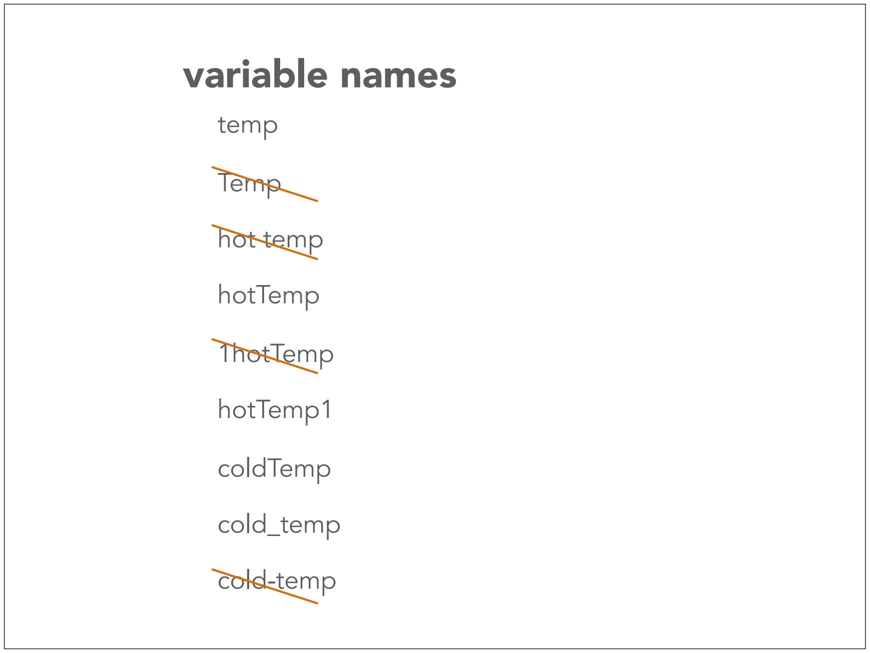
1hotTemp

hotTemp1

coldTemp

cold_temp

cold-temp



variable names

temp

~~Temp~~

~~hot temp~~

hotTemp

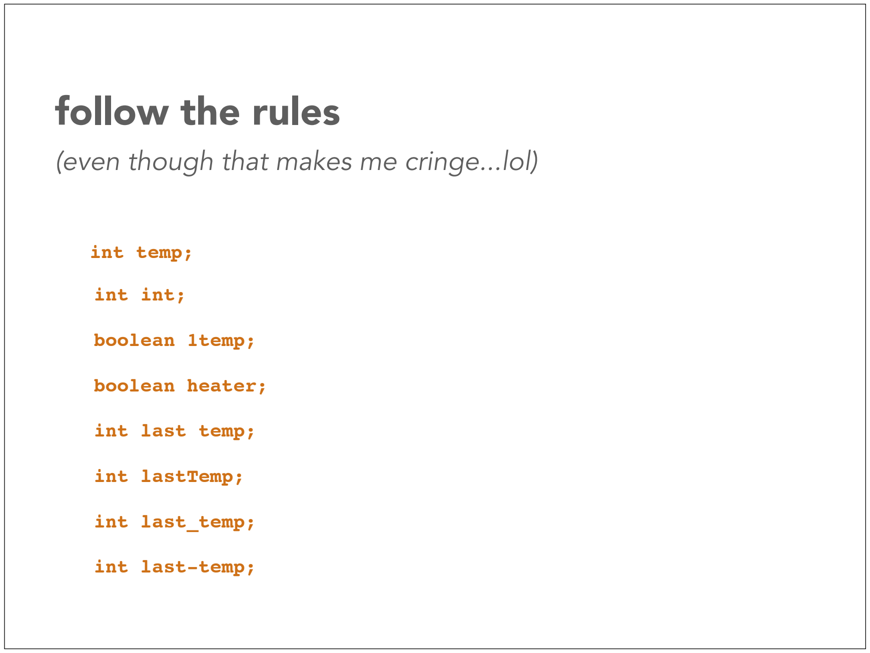
~~1hotTemp~~

hotTemp1

coldTemp

cold_temp

~~cold-temp~~



follow the rules

(even though that makes me cringe...lol)

```
int temp;
int int;
boolean 1temp;
boolean heater;
int last temp;
int lastTemp;
int last_temp;
int last-temp;
```

follow the rules

(even though that makes me cringe...lol)

```
int temp;
int int; //can't use a keyword
boolean 1temp; //can't start with number
boolean heater;
int last temp; //can't use space
int lastTemp;
int last_temp;
int last-temp; //can't use dash
```

follow the rules

(even though that makes me cringe...lol)

```
int temp;
~~int int;~~ //can't use a keyword
~~boolean 1temp;~~ //can't start with number
boolean heater;
~~int last temp;~~ //can't use space
int lastTemp;
int last_temp;
~~int last-temp;~~ //can't use dash
```

consistency

be consistent in naming conventions:
user_name or **userName** (called *camelCase*)

self-documentation

design your code so it is easy to understand (think visual communication)...think of others who will be reading your code

try to make it shorter rather than longer

self-documentation

what is a good variable name for temperature?

self-documentation

what is a good variable name for temperature?

```
t

temp

temperature

roomTemp

roomTemperature
```

variable declaration and initialization

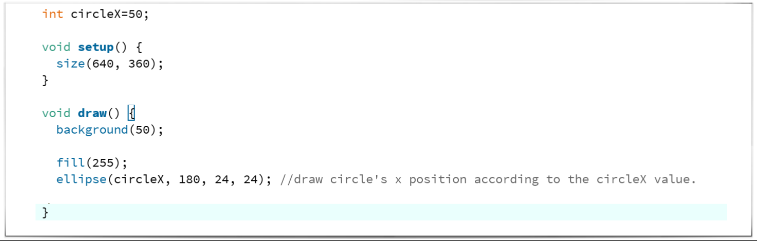
start with declaration (data type and unique name),

option to initialize

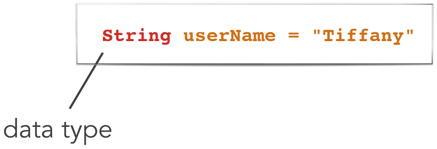
```
//delcaration only
int x;
```

```
//initialization
x = 5;
```

```
//delcaration and initialization
int x = 5;
```

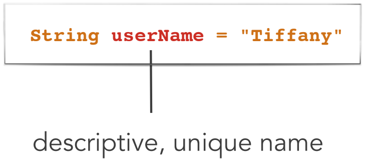


```
int circleX=50;

void setup() {
  size(640, 360);
}

void draw() {
  background(50);

  fill(255);
  ellipse(circleX, 180, 24, 24); //draw circle's x position according to the circleX value.

}
```



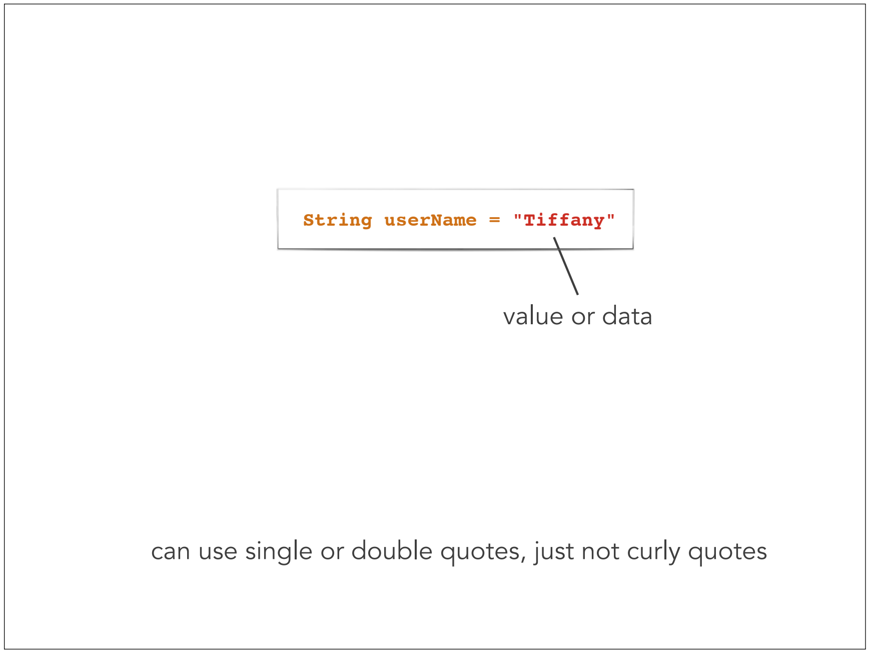
```
String userName = "Tiffany"
```

data type



```
String userName = "Tiffany"
```

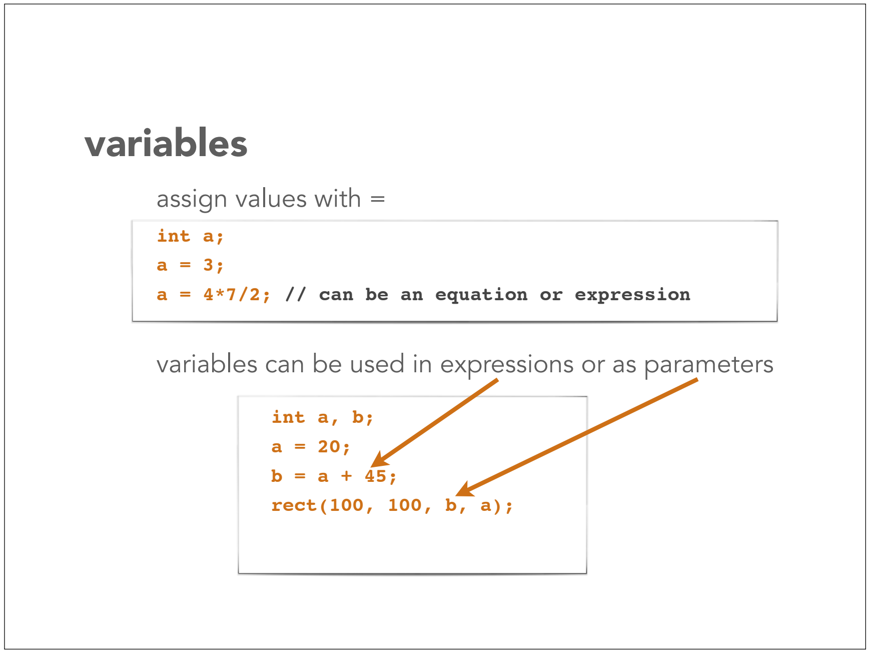
descriptive, unique name



```
String userName = "Tiffany"
```

value or data

can use single or double quotes, just not curly quotes



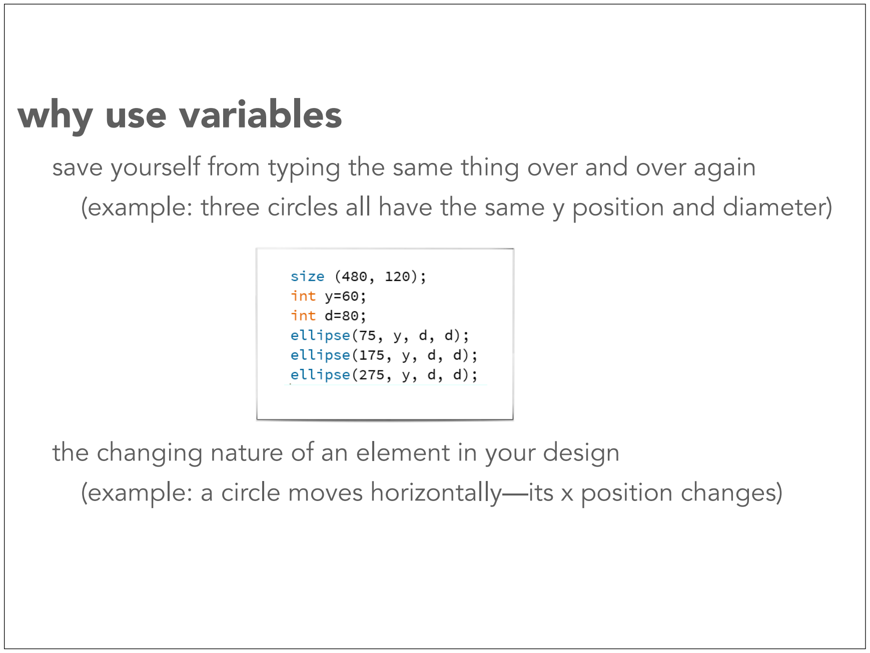
variables

assign values with =

```
int a;
a = 3;
a = 4*7/2; // can be an equation or expression
```

variables can be used in expressions or as parameters

```
int a, b;
a = 20;
b = a + 45;
rect(100, 100, b, a);
```



why use variables

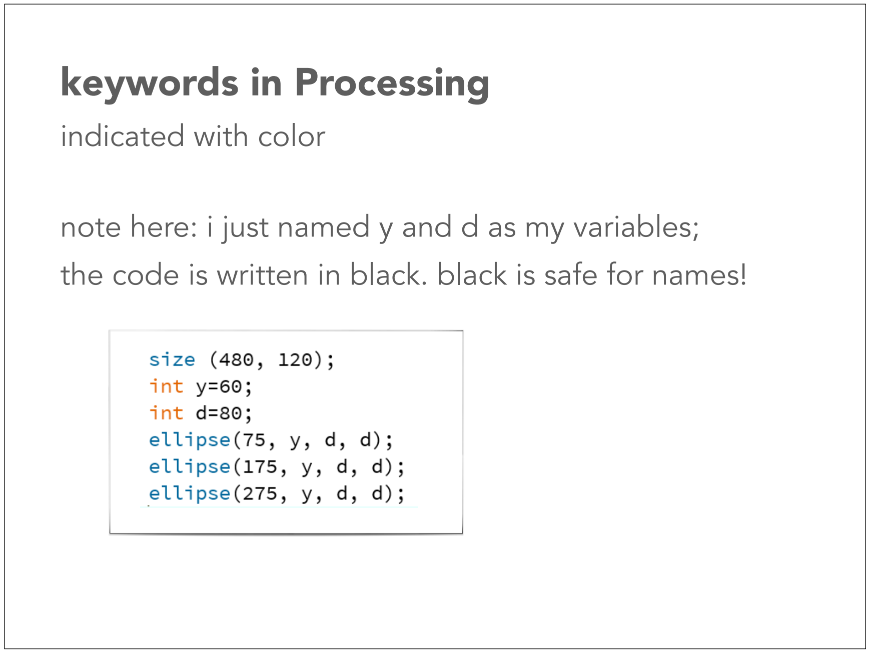
save yourself from typing the same thing over and over again

(example: three circles all have the same y position and diameter)

```
size (480, 120);
int y=60;
int d=80;
ellipse(75, y, d, d);
ellipse(175, y, d, d);
ellipse(275, y, d, d);
```

the changing nature of an element in your design

(example: a circle moves horizontally—its x position changes)



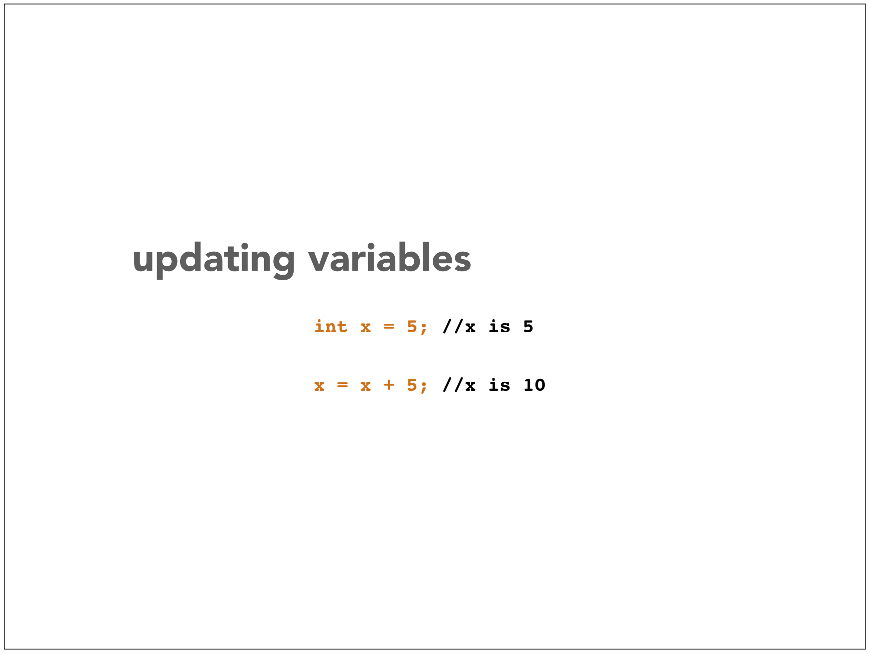
keywords in Processing

indicated with color

note here: i just named y and d as my variables;

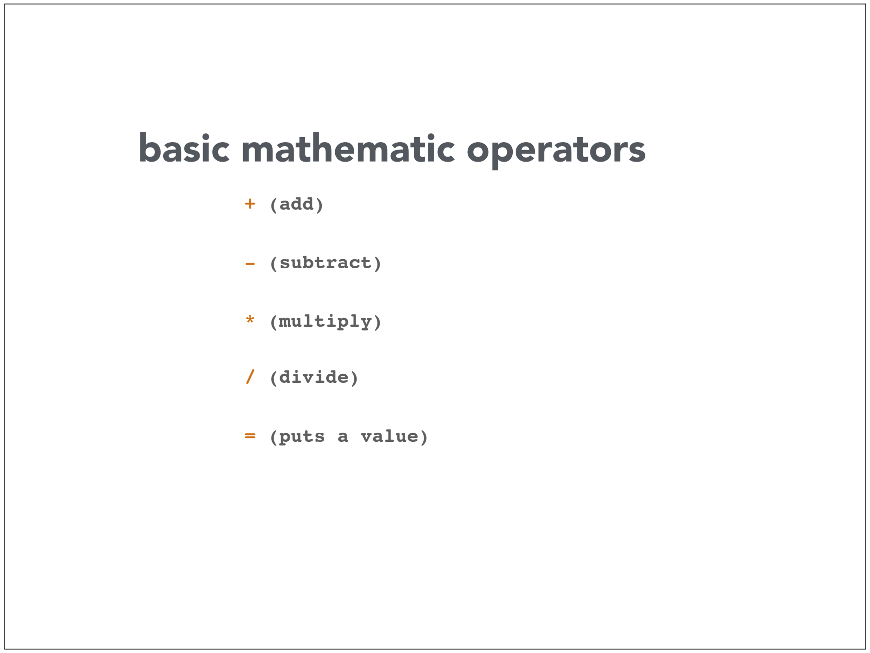
the code is written in black. black is safe for names!

```
size (480, 120);
int y=60;
int d=80;
ellipse(75, y, d, d);
ellipse(175, y, d, d);
ellipse(275, y, d, d);
```



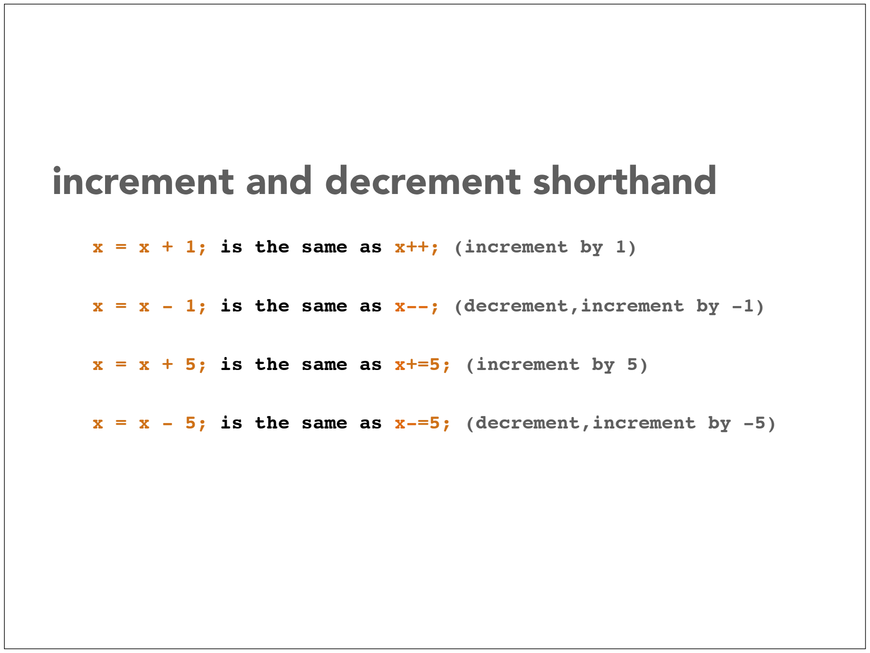
updating variables

```
int x = 5; //x is 5

x = x + 5; //x is 10
```



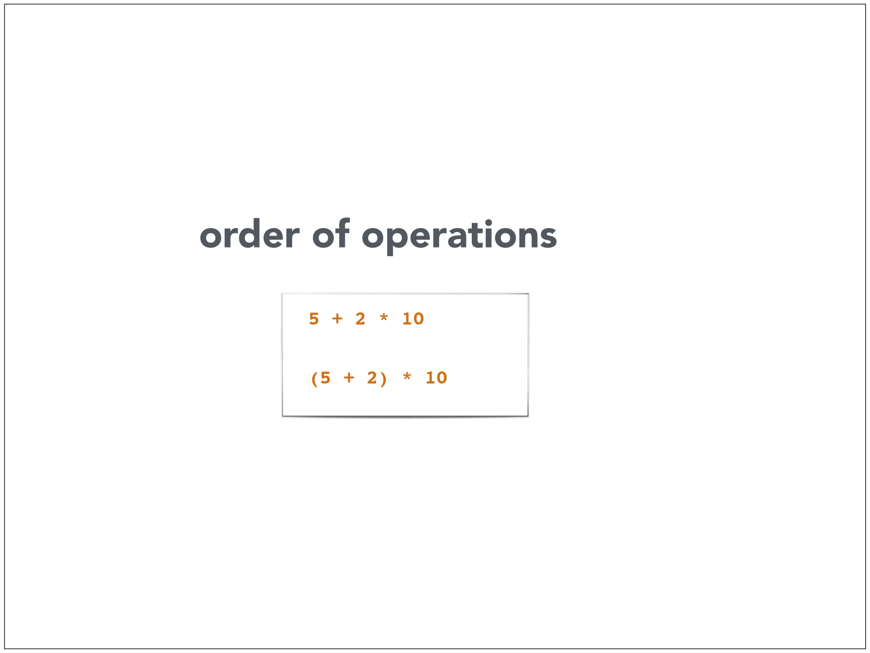
basic mathematic operators

```
+ (add)

- (subtract)

* (multiply)

/ (divide)

= (puts a value)
```



increment and decrement shorthand

```
x = x + 1; is the same as x++; (increment by 1)

x = x - 1; is the same as x--; (decrement,increment by -1)

x = x + 5; is the same as x+=5; (increment by 5)

x = x - 5; is the same as x-=5; (decrement,increment by -5)
```



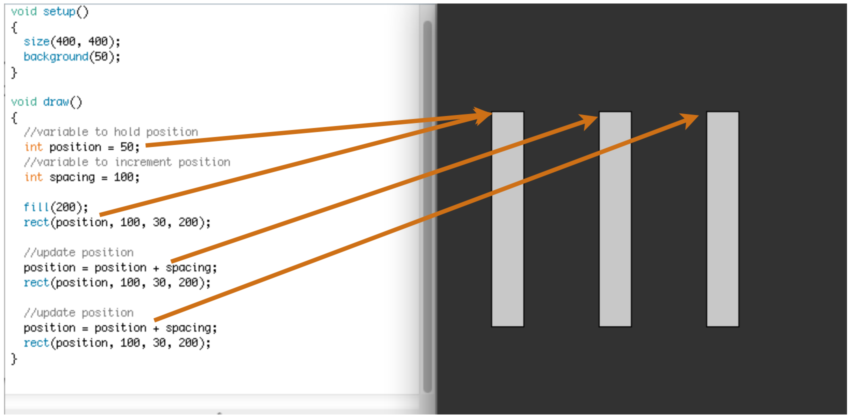
order of operations

```
5 + 2 * 10

(5 + 2) * 10
```

when to use a variable

some say any time you type a number (*that might be overkill...*)

look at your sketches for reused values that can be defined by a pattern or equation



```
void setup()
{
  size(400, 400);
  background(50);
}

void draw()
{
  //variable to hold position
  int position = 50;
  //variable to increment position
  int spacing = 100;

  fill(200);
  rect(position, 100, 30, 200);

  //update position
  position = position + spacing;
  rect(position, 100, 30, 200);

  //update position
  position = position + spacing;
  rect(position, 100, 30, 200);
}
```

variable scope

when a variable will change in each iteration of draw, declare it outside of `setup()` and `draw()`

this is called a ***global variable***

variable scope

when a variable is created within a block of code, such as a function such as `draw()`, it can be used only within that block

it will be destroyed with the program leaves the block

this is called a ***local variable***

conditional statements

by default Processing executes lines of a program one after the other (***procedural***)

sometimes we want to control which steps are executed depending on what else has happened, or ***conditional control flow***

necessary to make anything ***non-linear*** and ***interactive***...on our way to *object-oriented*

conditional statements

if

if I eat chocolate, then I will be happy

if, else

if I eat chocolate, then I will be happy,
else I will be very sad

if, else if

if I eat dark chocolate, then I will be happiest,
else, if I eat milk chocolate, then I will be less happy, but still happy
else, if I eat white chocolate, then I will feel sick

conditional statements

if

only do one thing, when there are two possibilities:
imagine you have 5 big apples and 5 small apples.
if it's a big apple, then put it in the basket.

if, else

can do two things, when there are two possibilities:
same scenario as above.
if it's a big apple, then put it in the basket, **else** compost the apple.

if, else if

more than two possibilities:
if it's a small apple, then compost,
else if it's a big red apple, then put it in the red basket,
else, put it in the green basket.

conditional statement syntax for if

```
if (condition) {
   statements;
}
```

```
if (x > 5) {
   background (0);
}
```

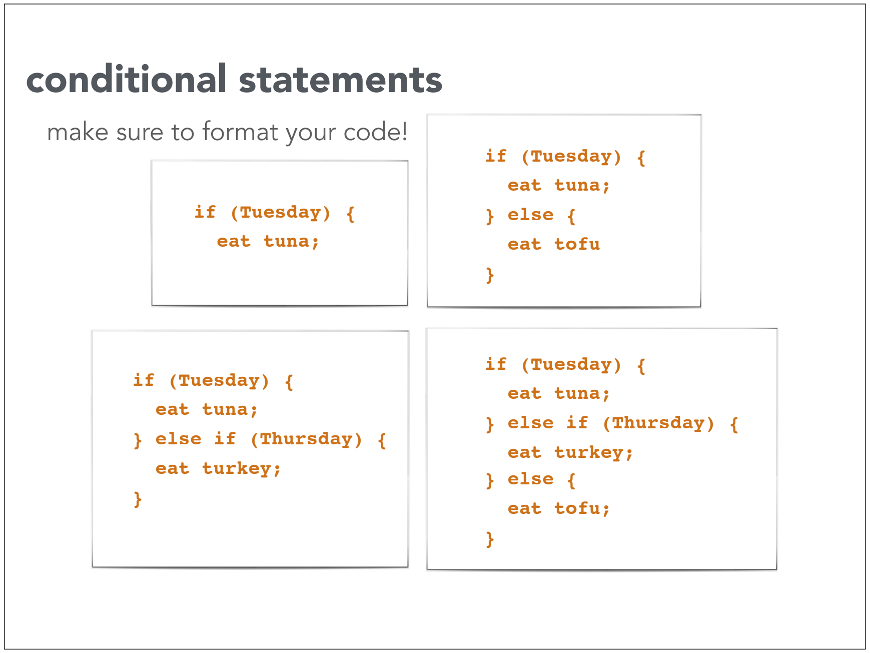
if the condition inside the ( ) evaluates to be true, then, execute the code inside the { }. If not, do nothing

conditional statement syntax for if else

```
if (condition) {
   statements
} else {
   statements
}
```

```
if (x > 5) {
   background (0);
} else {
   background (255);
}
```

if the condition inside the ( ) evaluates to be true, then, execute the code inside the { }. Else, execute the code inside the { } after the word "else"



conditional statements

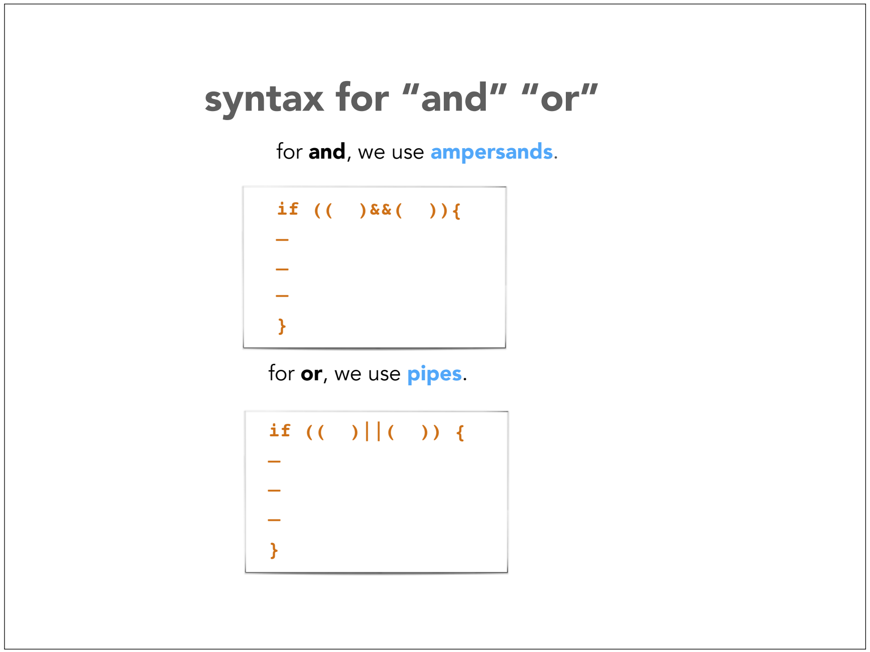
make sure to format your code!

```
if (Tuesday) {
  eat tuna;
```

```
if (Tuesday) {
  eat tuna;
} else {
  eat tofu
}
```

```
if (Tuesday) {
  eat tuna;
} else if (Thursday) {
  eat turkey;
}
```

```
if (Tuesday) {
  eat tuna;
} else if (Thursday) {
  eat turkey;
} else {
  eat tofu;
}
```



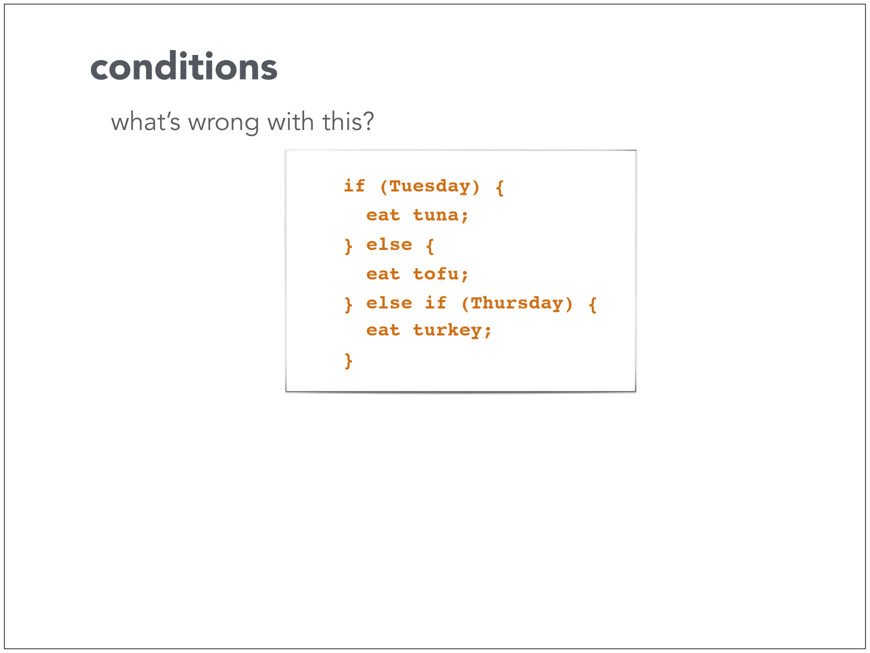
syntax for “and” “or”

for **and**, we use **ampersands**.

```
if ((   )&&(   )){
_
_
_
}
```

for **or**, we use **pipes**.

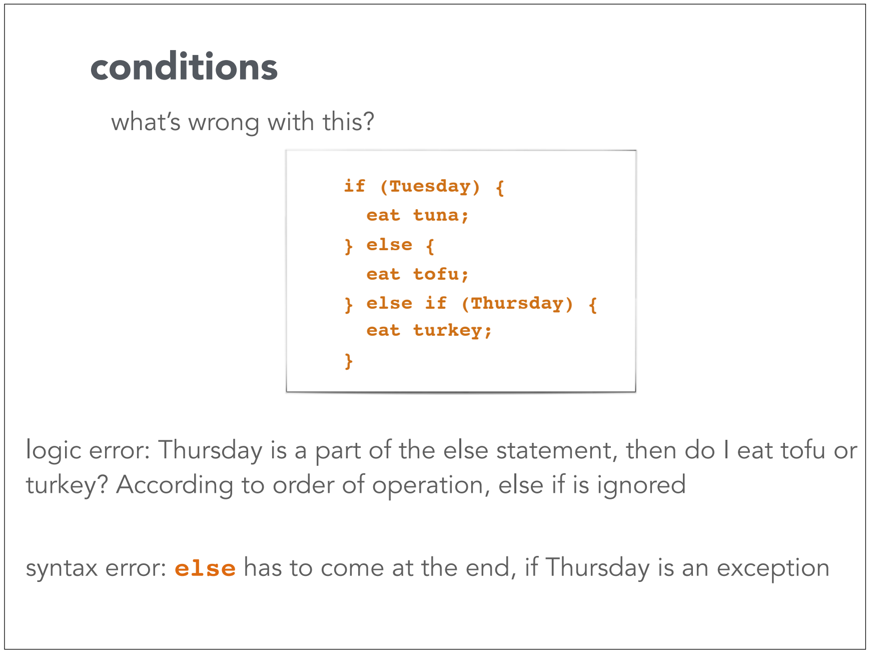
```
if ((   )||(   )) {
_
_
_
}
```



conditions

what’s wrong with this?

```
if (Tuesday) {
  eat tuna;
} else {
  eat tofu;
} else if (Thursday) {
  eat turkey;
}
```



conditions

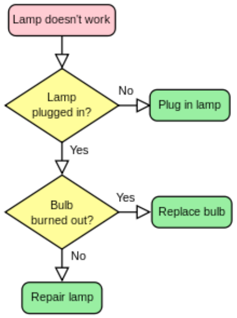
what’s wrong with this?

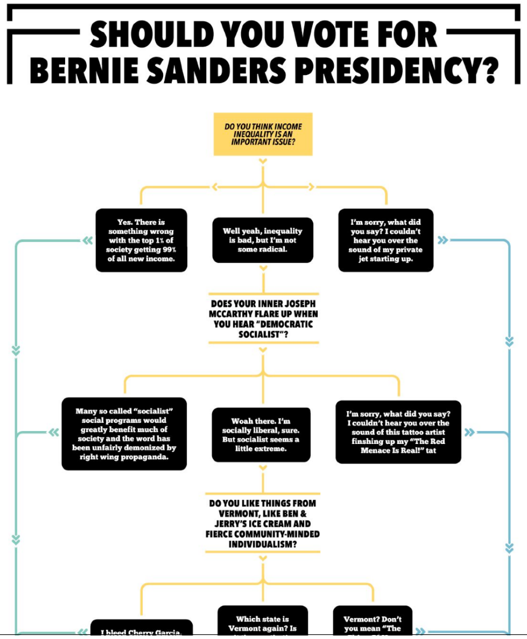
```
if (Tuesday) {
  eat tuna;
} else {
  eat tofu;
} else if (Thursday) {
  eat turkey;
}
```

logic error: Thursday is a part of the else statement, then do I eat tofu or turkey? According to order of operation, else if is ignored

syntax error: `else` has to come at the end, if Thursday is an exception

flowchart





SHOULD YOU VOTE FOR BERNIE SANDERS PRESIDENCY?

DO YOU THINK INCOME INEQUALITY IS AN IMPORTANT ISSUE?

Yes. There is something wrong with the top 1% of society getting 99% of all new income.

Well yeah, inequality is bad, but I'm not some radical.

I'm sorry, what did you say? I couldn't hear you over the sound of my private jet starting up.

DOES YOUR INNER JOSEPH MCCARTHY FLARE UP WHEN YOU HEAR "DEMOCRATIC SOCIALIST"?

Many so called "socialist" social programs would greatly benefit much of society and the word has been unfairly demonized by right wing propaganda.

Woah there. I'm socially liberal, sure. But socialist seems a little extreme.

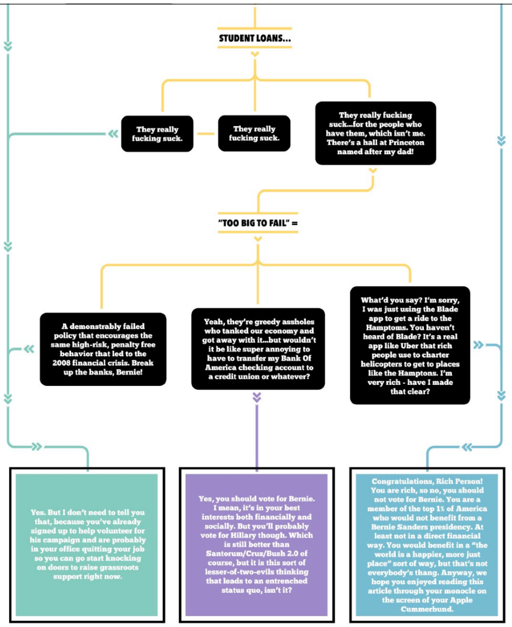
I'm sorry, what did you say? I couldn't hear you over the sound of this tattoo artist finshing up my "The Red Menace Is Real!" tat

DO YOU LIKE THINGS FROM VERMONT, LIKE BEN & JERRY'S ICE CREAM AND FIERCE COMMUNITY-MINDED INDIVIDUALISM?

I bleed Cherry Garcia.

Which state is Vermont again? Is

Vermont? Don't you mean "The



STUDENT LOANS...

They really fucking suck.

They really fucking suck.

They really fucking suck...for the people who have them, which isn't me. There's a hall at Princeton named after my dad!

"TOO BIG TO FAIL" =

A demonstrably failed policy that encourages the same high-risk, penalty free behavior that led to the 2008 financial crisis. Break up the banks, Bernie!

Yeah, they're greedy assholes who tanked our economy and got away with it...but wouldn't it be like super annoying to have to transfer my Bank Of America checking account to a credit union or whatever?

What'd you say? I'm sorry, I was just using the Blade app to get a ride to the Hamptoms. You haven't heard of Blade? It's a real app like Uber that rich people use to charter helicopters to get to places like the Hamptons. I'm very rich - have I made that clear?

Yes. But I don't need to tell you that, because you've already signed up to help volunteer for his campaign and are probably in your office quitting your job so you can go start knocking on doors to raise grassroots support right now.

Yes, you should vote for Bernie. I mean, it's in your best interests both financially and socially. But you'll probably vote for Hillary though. Which is still better than Santorum/Cruz/Bush 2.0 of course, but it is this sort of lesser-of-two-evils thinking that leads to an entrenched status quo, isn't it?

Congratulations, Rich Person! You are rich, so no, you should not vote for Bernie. You are a member of the top 1% of America who would not benefit from a Bernie Sanders presidency. At least not in a direct financial way. You would benefit in a "the world is a happier, more just place" sort of way, but that's not everybody's thang. Anyway, we hope you enjoyed reading this article through your monocle on the screen of your Apple Cummerbund.

for the purpose of evaluation...

relational operators

```
>  (greater than)

<  (less than)

>= (greater than or equal to)

<= (less than or equal to)
```

logical operators

```
== (checks for equality)

!= (checks for inequality)

|| (logical or)

&& (logical and)
```

what is the difference between = and ==?

tests

simple equality tests:

```
(5 == 6)                  //false
(5 == 5)                  //true
```

relational tests:

```
(5 < 6)                   //true
(5 > 5)                   //false
(5 <= 5)                  //true
```

logical tests:

```
(true || false)           //true
(!false)                  //true
```

combined tests:

```
!(15 > 20)                //true
((5 == 6) && (5 == 5))    //false
((5 == 6) || (5 == 5))    //true
```

loops are like the industrial revolution!

repetitive tasks are done by the machine

most common variables for generic counting

`i`, `j` and `k`

two types of loops

while loop

for loop

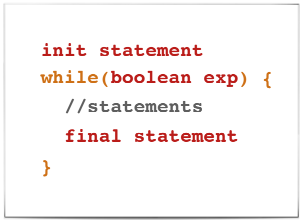
while loop

the idea of taking a single concept and repeat it many many times

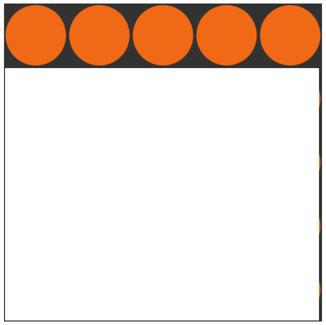
a while loop controls a sequence of repetitions…
ok, that’s beautiful, because it saves so much time!

however, loops **must** have an exist condition, or it will lock out!
it will get stuck and never get out!

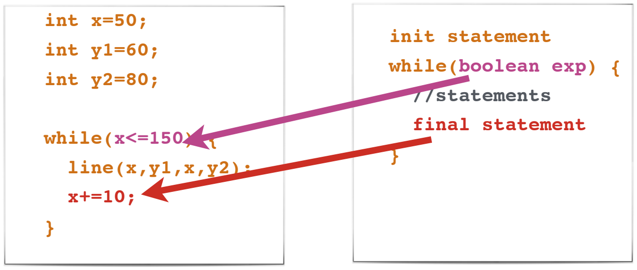
while loop



```
init statement
while(boolean exp) {
  //statements
  final statement
}
```



while loop



```
int x=50;
int y1=60;
int y2=80;

while(x<=150){
  line(x,y1,x,y2);
  x+=10;
}
```

```
init statement
while(boolean exp) {
  //statements
  final statement
}
```

compare "if" to "while"

in an if statement, if the evaluation inside the () is true, execute the statement **once**; if the evaluation is false, don't execute the statement

```
if ( boolean expression ){
  statement;
}
```

in a while statement, as long as the evaluation is true, the statement is executed **infinite times**

syntax:

```
while ( boolean expression ){
  statement;
}
```

the code will run, while the condition inside the () is true
once the condition is no longer met, it jumps out of the while loop

using if

```
float x=0;

void setup() {
  size (600, 400);
}

void draw() {
  background (0);
  fill(255);
  noStroke();

  if (x<width) {
    ellipse (x, 100, 25, 25);
    x=x+1;
  }
}
```

using while

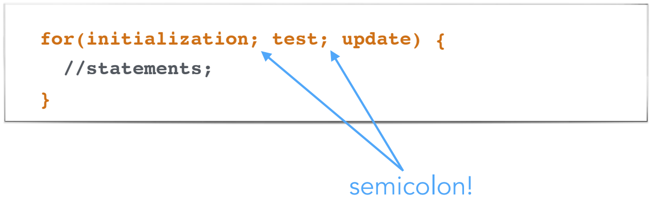
```
float x=0;

void setup() {
  size (600, 400);
}

void draw() {
  background (0);
  fill(255);
  noStroke();

  while (x<width) {
    ellipse (x, 100, 25, 25);
    x=x+1;
  }
}
```

for loop

```
for(init statement; boolean exp; final statement) {
  //statements
}
```

for loop

when you don't need it to be infinite and forever. Great for controlling repetition. Can be identical to the while loop, except shorter.



```
for(initialization; test; update) {
  //statements;
}
```

for loop

```
for (int i=0; i<width; i+=100) {

   //rect
   fill(50);
   rect(x+i, y, s, s);

   //ellipse
   fill(#F06916);
   ellipse(s/2+i, s/2, s-5, s-5);
 }
```

for loop

start at 0 and count up to 9 (10 iterations)

```
for (int i=0; i<10; i++)
```

start at 0 and count up to 100 by 10 (10 iterations)

```
for (int i=0; i<100; i+=10)
```

start at 100 and count down to 0 by 5 (20 iterations)

```
for (int i=100; i>=0; i-=5)
```

converting while loop to for loop

seeing how `for` loops can be converted to `while` loops helps you understand `for` loops

```
init statement
while(boolean exp) {
  //statements
  final statement
}
```

is the same as

```
for(init statement; boolean exp; final statement) {
  //statements
}
```

i'm feeling loopy and iffy!...which one to use?

use a `while` loop if you don’t know how many times you want the loop to execute (or based on an existing variable)

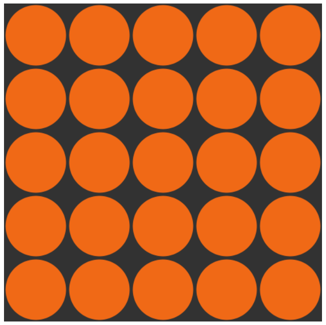
use a `for` loop if you know how many times you want to repeat

the `while` loop is considered a general purpose loop construct, but remember the test or boolean expression is evaluated outside the loop, so the loop body may not execute

nested loops

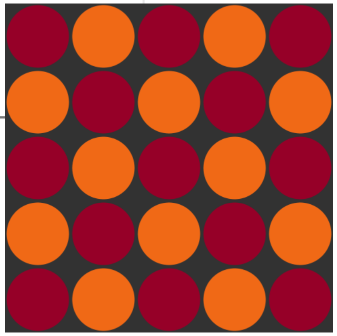
used in a situation where two or more variables needed to be evaluated

```
//add a loop to repeat design in a row
for (int i=0; i<width; i+=100) {
  //fill columns with design
  for (int j=0; j<height; j+=100){

  //rect
  fill(50);
  rect(x+i,y+j,s,s);

  //ellipse
  fill(#F06916);
  ellipse(s/2+i,s/2+j,s-5,s-5);
  }
}
```



adding a condition

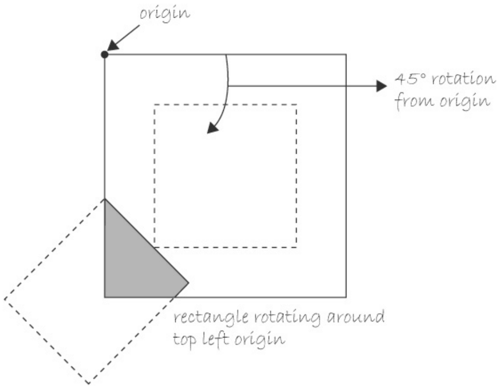
```
 //% is an operator called “mod"
 //it returns the remainder from division
 //e.g. x % y = the remainder of dividing x by y

if ((i+j)%200==0) {
   fill(150, 0, 40); //red
 } else {
   fill(#F06916); //orange
 }
```



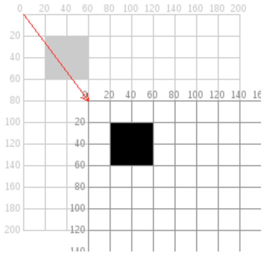
rotate function

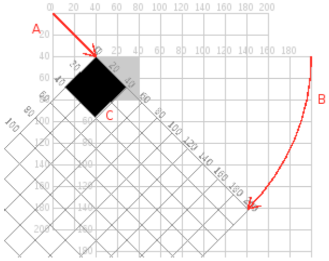
moves the matrix from 0,0 to a new point

```
rotate(radians(45));
rectMode(CENTER);
rect(width/2,height/2,100,100);
```



translate





```
translate(60,80);
rect(20,20,40,40);
```

```
translate(40,40);
rotate(radians(45));
rect(0,0,40,40);
```

scale

increases or decreases the size of a shape by expanding and contracting vertices

objects always scale from their relative origin to the coordinate system

scale values are specified as decimal percentages

for example, the function call scale(2.0) increases the dimension of a shape by 200%

```
rect(30, 20, 50, 50);
scale(0.5);
rect(30, 20, 50, 50);
```

```
rect(30, 20, 50, 50);
scale(0.5, 1.3);
rect(30, 20, 50, 50);
```

pushMatrix() and popMatrix()

```
pushMatrix() //saves current matrix

popMatrix() // restores last saved matrix
```

mouse interaction

mouseX and mouseY
position of the mouse in the sketch

pmouseX and pmouseY
previous position of the mouse (in the last frame)

mouseButton
which mouse button has pressed

mousePressed
true if the mouse button is pressed, false otherwise

mousePressed()
function that runs when the mouse is pressed

keyboard interaction

```
keyPressed() // function that runs once when any key is pressed

key // which key was pressed
```

```
void keyPressed() {
  if (key == 'j' || key == 'J') {
    d+=5;
  } else if (key == 'k' || key == 'K') {
    d-=5;
  }
}
```

functions

named blocks of code that make your code more:

modular: allow complex programs to be broken down into smaller, simpler tasks

readable: easier to understand, debug and maintain; one function call can replace many lines of code

reusable: allow commonly used code to be defined once and called repeatedly

function names

what would be a good name for a function that :

draws out a gradient in the background?

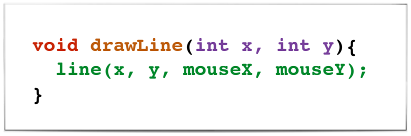
moves a pixel across the sketch?

draws rectangles across the sketch?

stores user information (such as name, dob)?

functions

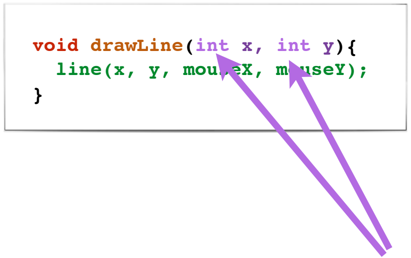
```
returnType functionName (arguments) {
  // instructions
}
```



```
void drawLine(int x, int y){
  line(x, y, mouseX, mouseY);
}
```

functions

```
returnType functionName (arguments) {
  // instructions
}
```



```
void drawLine(int x, int y){
  line(x, y, mouseX, mouseY);
}
```

arguments are optional, but **must include data type** if specified

add **arguments** in definition

```
void drawLine(int x, int y) {
  line(x, y, mouseX, mouseY);
}
```

add **parameters** in call

```
void draw() {
  background(255);
  drawLine(5, 0);
  drawLine(width/2, 0);
  drawLine(width-5, 0);
}
```

we've already been using built-in methods with parameters!

```
line(0, 10, 10, 20);
```

arrays

an array is a collection of variables of the same data type

arrays can be collections of ints, chars,
floats, booleans or any data type

the Design Department offers an **array** of courses…
too bad there are not more in the area of web design!

arrays

variable:

```
int x;
```

array:

```
int[] x;
```

beauty of array:

```
int[] x = new int[3000];
//create an array of 3000 integer variables
//length of the array goes inside [ ]
```

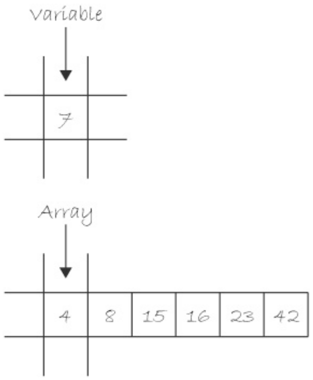
arrays

an int variable:

```
int number = 7;
```

an array of ints:

```
int[] numbers = {4,8,15,16,23,42};
```



declare, create, assign

step 1: declared array and define data type

```
float[] x;
```

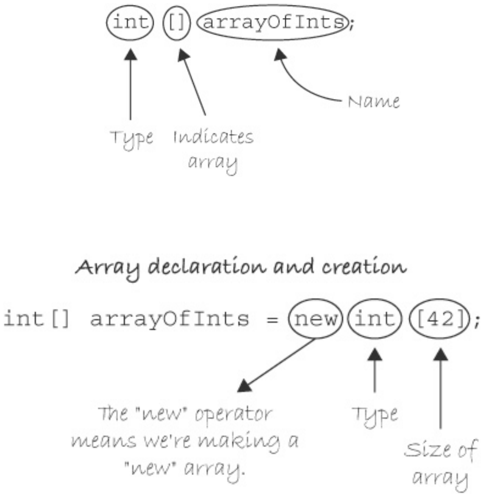
step 2: create array with keyword *new* and define length

```
float[] x = new float[3000];
```

step 3: assign values to each element

```
x[i] = random (0, 500);
```

arrays


int [] arrayOfInts;
Type
Indicates array
Name
Array declaration and creation
int [] arrayOfInts = new int [42];
The "new" operator means we're making a "new" array.
Type
Size of array

declaring, initializing and populating an array

declared, initialized and populated

```
int[] numbers = {2,4,6,8,10,12,14,16,18,20};
```

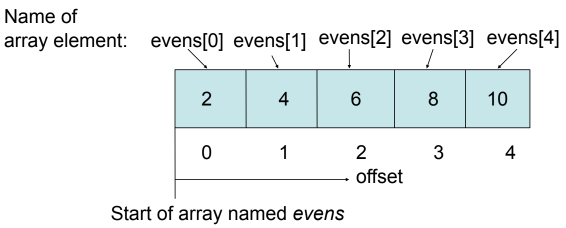
declared and initialized, then populated inside setup()

```
int[] numbers = new int[10];

void setup(){
  int[0]=2;
  int[1]=4;
  ...
}
```

accessing the array elements

each element in an array has a name—
it's the array plus an offset also called an ***index***

```
int[] evens = {2, 4, 6, 8, 10};
```



later on, you can retrieve the value according to the unit number!

sample array

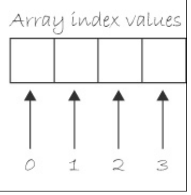
```
String[] family = {"Jesse", "Penelope", "Rocket", "Charlie"};
```

arrays begin counting position (index) at 0

what is **`family[0]`**?

`length` determines the number of objects in the array:

`family.length` *is 4*

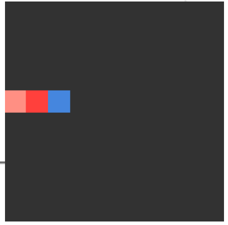


loops work great with arrays
index i used to generate x and y values

```
float[] x=new float[3000]; //3000 x location
float[] y=new float[3000]; //3000 y location

void setup() {
  size (500, 500);

  for (int i=0; i<x.length; i++) { /* dot operator refers to the
                                       length of x array */
      x[i]=random(-10, 200);
      y[i]=random(-10, 200);
  }


}
```

for loops work great with arrays

```
void showRectangles() {

  //array (list) of the color data type
  color[] palette = {
    #ff9086, #ff4343, #4988db
  };

  //width of rectangles
  int w=50;

  //for loop to draw rectangle for each color in the array
  for (int i=0; i<palette.length; i++) {

    //fill with the color in the palette array at the index
    fill (palette[i]);

    //draw the rectangles
    rect (i*w, 200, w, w);
  }
}
```



random

each time the random() function is called, it returns an unexpected value within the specified range. If only one parameter is passed to the function, it will return a float between zero and the value of the high parameter.

to generate a random number
between 0 and high and assign it to f

```
float f;
f = random(high);
```

to generate a random number between low and high

```
f = random(low, high);
```

coding strategies

break programs down into small chunks

write a little bit of code and test

write more and test

use many comments

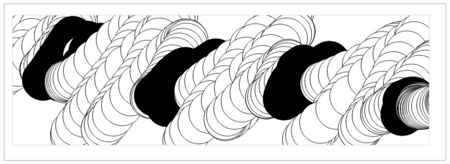
use print() and println() to debug

print your code and read away from the computer

rubber ducky method

post in “Discussions” and email glenda & Spencer

practice



size: 850 x 200

content: ellipse

features: ellipse follows the cursor
ellipse color changes on
mouse pressed

processing vs. p5.js

processing	p5.js
`void`	`function`
`size()`	`createCanvas()`
`mousePressed`	`mouseIsPressed`
`pushMatrix()`	`push()`
`popMatrix()`	`pop()`
`float, int`	`var`
`String`	`var`
`float[] x = new float[3]`	`var x = [0,5,10]`
	no 3D, `PShape` or `PFont`
	all variables declared with `var`

link p5.js to html

```
<script src=“http://cdnjs.cloudflare.com/ajax/libs/p5.js/0.4.21/p5.js”>
</script>
```

html link to js files

html head

```
<!--link to the p5.js library-->
<head>
  <meta charset="UTF-8">
  <title>processing</title>
  <script src=“http://cdnjs.cloudflare.com/ajax/libs/p5.js/0.4.21p5.js">
  </script>
</head>
```

html body

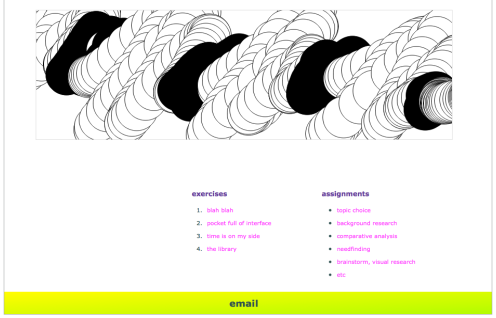
```
<body>
  <header id=“mySketch”></header>
  <!–put the rest of your html here–>

  <!--put the script link right before the end of the body tag-->
  <script src=“script.js”></script>
</body>
```

custom js file / capture to canvas

script.js

```
function setup() {
  //create a variable that references the html5 canvas
  //remember, what was called size() in Processing is called
  //createCanvas() is p5.js
  var myCanvas = createCanvas(800, 250);

  //parent the myCanvas variable to the html element titled “mySketch”
  myCanvas.parent('mySketch');
}

function draw() {
  //statements
}
```

practice



exercises

1. blah blah
2. pocket full of interface
3. time is on my side
4. the library

assignments

- topic choice
- background research
- comparative analysis
- needfinding
- brainstorm, visual research
- etc

email

add a simple sketch to your header for your portal

and then make it more cool and more interesting

github

post github link to the Canvas