Helpful code:

Basic functions:

```
void setup () {
  //initialization
  //such as size, background
}

void draw () {
  //code that repeats forever
  //such as shapes and movements
}
```

System variables:

```
mouseX            //  x location of mouse
mouseY            //  y location of mouse
pmouseX           //  previous x location of mouse
pmouseY           //  previous y location of mouse
background(50);   //  background color
screen.width      //  width of entire screen
screen.height     //  height of entire screen
frameRate         //  rate frames are processed (fps)
frameCount        //  number of frames processed
```

Conditional s:

```
if (test) {
   statements
}
```

// Free Patterns!

In this exercise you will explore using variables to streamline code-based drawing. You will also add movement to one of the sketches.

Sketch each design on graph paper first. Look for values that have a mathematical pattern that can become variables.

Begin all sketches with comments that include a description of the sketch, name and date.

Comment all code.

Continue to work with basic parameters in this exercise, such as shape, stroke, fill and color (as well as design principles, such as form, repetition, placement, balance, proximity, etc). Push the limits within those constraints. Keep all techniques within the chapter topics (no loops, custom functions).

// Start Sketching!

* All sketches must be 300 x 300

* Create 3 initial sketches (without movement):

1. Create a sketch with at least five lines that are uniform but increase in weight (in space, not time). Use at least one variable, such as for the y1 position of the line (Ex2_1.pde)

2. Create a sketch with at least six rectangles that increase in dimension (in space, not time). Use at least two variables (Ex2_2.pde)

3. Create a sketch using at least four ellipses that overlap. Use at least three variables (Ex2_3.pde)

Helpful code (continued):

Operators:

==	(checks for equality)
!=	(checks for inequality)
ll	(logical or)
&&	(logical and)
++	(increment by 1)
+=	(add to the current value)
--	(decrease by 1)
-=	(decrease from current value)

Examples:
Simple equality tests:
(5 == 6) false
(5 == 5) true

Relational tests:
(5 < 6) true
(5 > 5) false
(5 <= 5) true

Logical tests:
(true ll false) true
(!false) true

Combined tests:
!(15 > 20) true
((5 == 6) && (5 == 5)) false
((5 == 6) ll (5 == 5)) true

// Make 'em move

Choose one of the above sketches and animate one shape, testing the bounding box of the sketch. If a shape hits the edge, it should bounce back. Try using alpha to create visual appeal of overlapping shapes. (Ex2_4.pde)

//If you are feeling comfortable

Take some inspiration from Shiffman's examples and add some incremental color changes

//Challenge (optional)

Add a random (but attractive) component to one of the above sketches and make sure to comment it (ex2_challenge.pde)

// Digital Submission

A folder to the class files (in SCC 2102) titled
FirstNameLastInitial_ex2
with all 4 (or 5) sketch folders (with pde files)

// Analog submission

Graph paper sketches, code printouts, screen shots of all sketches

// Note

We will view and discuss this exercise in class next week. Specific due date is in Canvas. Your work must be complete before class. Be prepared to discuss your ideas and results.