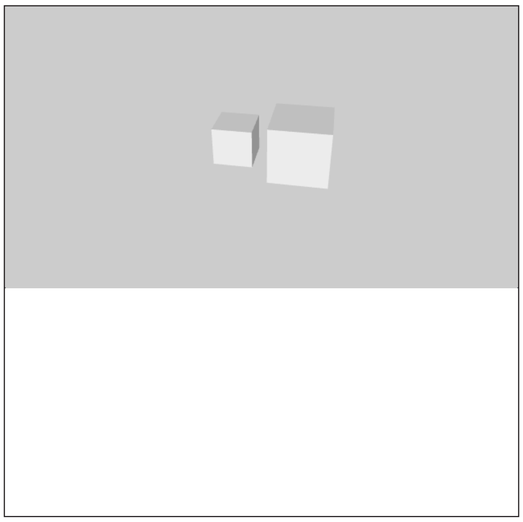
4 - Patterns in Processing

Overview: We are going to develop our understanding of using shapes to create repeating patterns. You will learn how we can save time and space by using as little code as possible.

STEP 1: RECAP SESSION 3 - INTERACTION - MOUSE & KEYBOARD

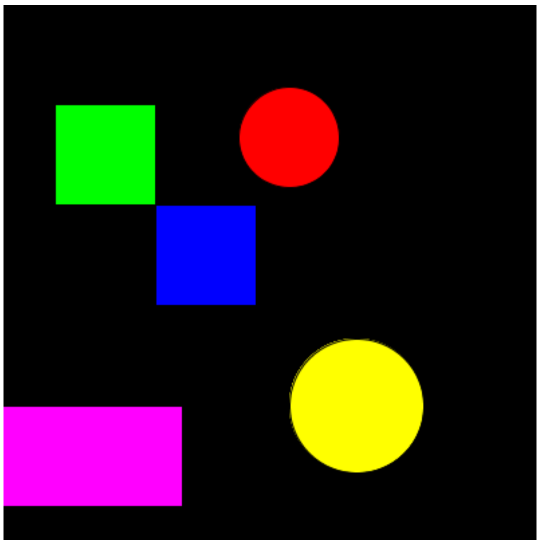
You can use the computer inputs to control objects on the screen. You can use the mouse and keyboard.

```
void setup() {
  size(500, 500);
  background(255);
  rectMode(CENTER);
}

void draw()
{

  fill(0, 255, 0);
  rect(mouseX, mouseY, 100, 100);
}
```



Now we will add lots of different shapes onto our screen. Last week we looked at using the different keys on your keyboard to draw different shapes.

```
void setup() {
  size(500, 500);
  background(0, 0, 0);
}
void draw() {
  if (keyPressed) {
    if (key == 'b') {
      fill(0, 255, 0);
      rect(75, 75, 75, 75);
    }
    if (key == 'c') {
      fill(0, 0, 255);
      rect(150, 150, 75, 75);
    }
  }

   else {
      if (key == 'p') {
        clear();
      }
    }
  }
```



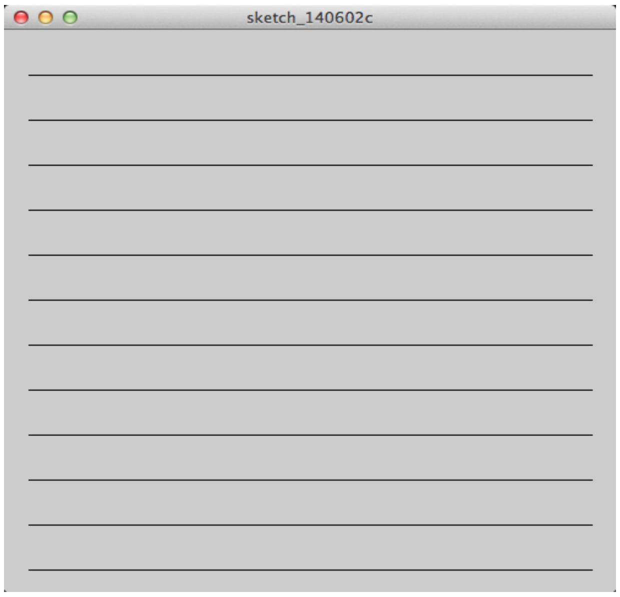
STEP 2: REPEATING PATTERNS - for Structure

So far we have looked at drawing shapes in processing individually each one on an separate line.

Lets start by creating a simple pattern using just lines,
we will start by making pattern using 12 lines of code.

Type the code bellow:

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

void draw()
{

 line(20, 40, 480, 40);
 line(20, 80, 480, 80);
 line(20, 120, 480, 120);
 line(20, 160, 480, 160);
 line(20, 200, 480, 200);
 line(20, 240, 480, 240);
 line(20, 280, 480, 280);
 line(20, 320, 480, 320);
 line(20, 360, 480, 360);
 line(20, 400, 480, 400);
 line(20, 440, 480, 440);
 line(20, 480, 480, 480);
}
```



Looking at the code above, there is a lot of lines of code for drawing just 12 lines. We are able to use iteration to create the same objective but with using only a few lines of code. At first it looks complicated, but we will explain it step by step. Type in the code below and run the program.

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

void draw()
{

  for (int i = 0; i < 500; i = i+40) {
   line(20, i+40, 480, i+40);
  }
}
```

What do you see when you run the program, is it the same as the first code?

Let’s break the code down into bits, so we can start to understand how it work.

General use of the for structure:

```
for (int, test; update) {
 statement
}
```

Specific broken down code:

```
1.          for (int i = 0;

2.          i < 500;

3.          i = i+40) {

4.           line(20, i+40, 480, i+40);
            }
```

1. Run.

2. Test - is i less than 500? If true continue to 3, if false continue to 5.

3. Run the statement in the block.

4. Run the update statement goto number 2.

5. Exit the block and run the program.

You will notice in the code the use of a new bracket, called an angle bracket. These are used in programming as an operator.

> a more-than sign. **<** a less-than sign.

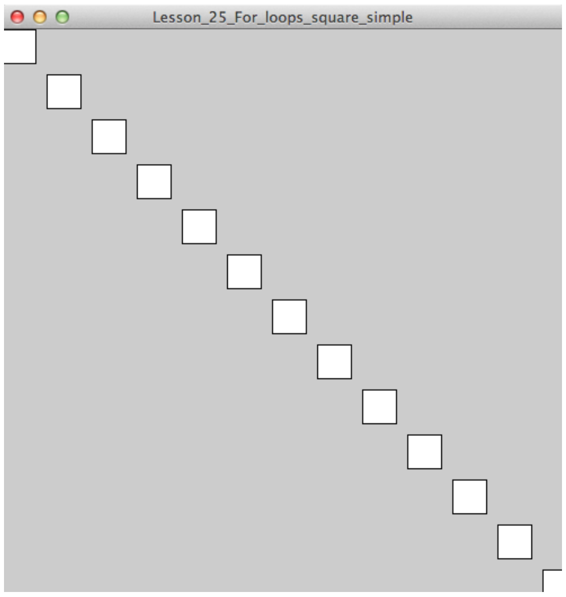
A quick sum for us to think about when we use the < operator, do you know what the equation below mean?

```
if i = 50;

i < 500;
```

STEP 3: COMPLICATED PATTERNS

The next stage is to use different shapes instead of lines. This is just as simple as the lines, but with adding a second dimension to our sketch. Lets start by using rectangles.

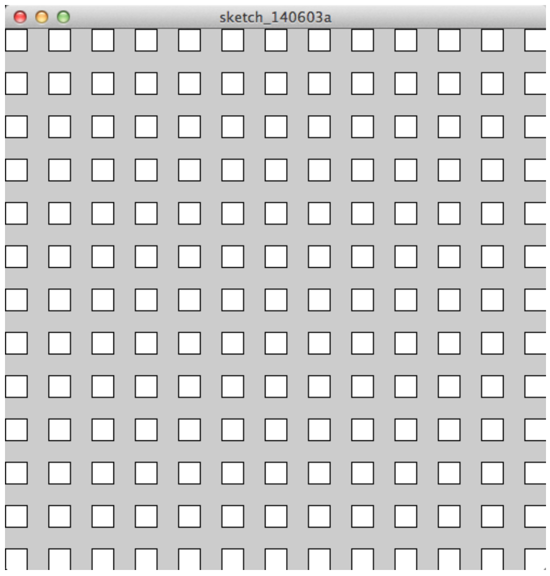
```
void setup() {
 size(500, 500);
}

void draw()
{

   for (int i = 0; i < 500; i = i+40) {
    rect(i, i, 40, 40);
  }
}
```



note Can you see how a repeated pattern as created a diagonal, can you see why?

What we want to do now is add a full grid of squares to our sketch. So far we only have one for structure, we will need to create two of them to create a full grid pattern.

Initially we had used i for both our x and y coordinates, now we will create a second for structure to change our y coordinate, we will call this j:

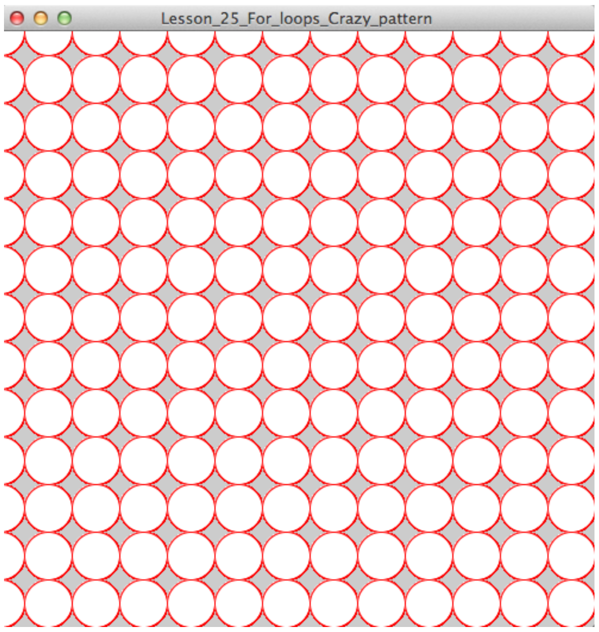


```
void draw()
{
  for (int i = 0; i < 500; i = i+40) {
   for (int j = 0; j < 500; j = j+40) {
    rect(i, j, 40, 40);
  }
 }
}
```

Do you see a grid of squares? If so can you change the size of your squares to be smaller than 40 x 40 pixels?

So far our shapes are very simple, white with a black border around them, in the next part lets add some colour.

We will start by deciding a colour for boarder. In processing this is called stroke. The works the same way as other RGB (RED, GREEN, BLUE) colour mode. In the example below I have chosen RED ellipses.

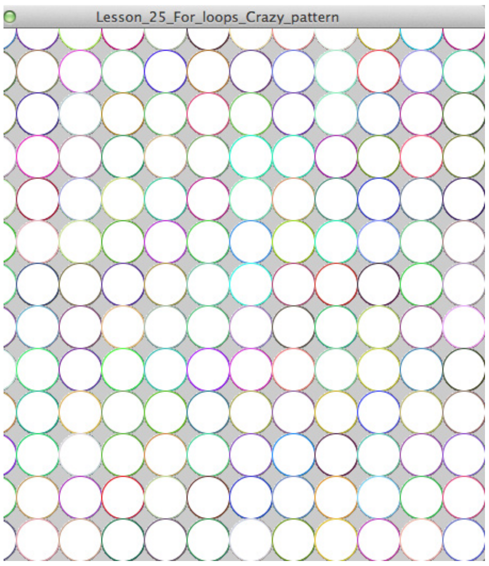


```
void setup() {
  size(500, 500);
  background(0, 0, 0);
}
void draw()
{

for (int i = 0; i < 500; i = i+40) {
    for (int j = 0; j < 500; j = j+40) {
      stroke(255, 0, 0);
      ellipse(i, j, 40, 40);
   }
  }
}
```

note Now its your chance to decide on your own shapes and colours, experiment with this values and change the size of your chosen shape.

Last session you learnt about using random function with colours. You need to tell the range for each colour shade. In the case of RGB colours the range is 0 to 255.

You can add this line to your stroke code:



```
stroke(random(0,255),random(0,255),random(0,255));
```

note When you run the code you might notice the loop is running too fast, if you would like to slow this down you can use

```
void setup() {
  size(500, 500);
  frameRate(1000);
}
```

This is a thousand frames a second, which is very fast, try a much lower number! What do you see?

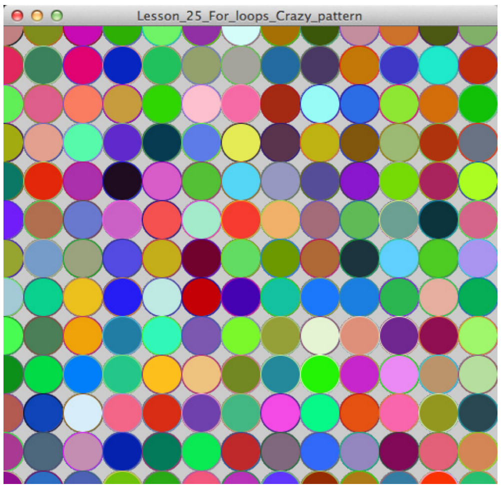
blue {
shift
}

STEP 4: MORE COMPLICATED PATTERNS

Now we have added our stroke () function, lets add more colour to our pattern. As before we have used fill().

Lets add a random colour to both the stroke and fill functions. Make sure to place this above the shape you would like to change.

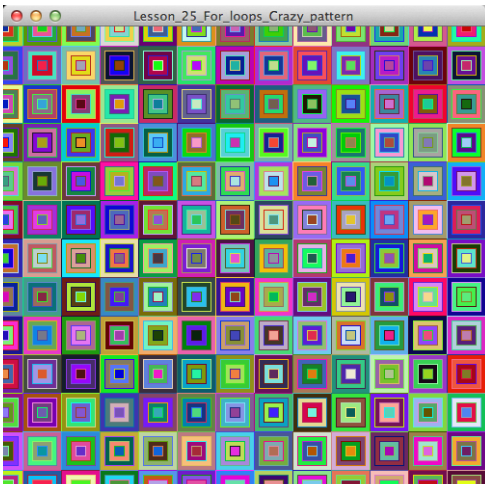
```
fill(random(0,255),random(0,255),random(0,255));
```



note   If you wanted to control just one of the RED, GREEN or BLUE colours what details would you change?

The final part of this session is allowing you the chance to experiment with adding multiple shapes and controlling each one with random colours. You can add as many as you want, but remember the shapes at the top of your code will be in the background and each line of code will place the shape in front.

Below is my code for just two of the squares in the image, in total I have 4 squares all with different random () stroke and fill functions.



```
void setup() {
  size(500, 500);
  frameRate(10);
  rectMode(CENTER);
}
void draw()
{
  for (int i = 0; i < 500; i = i+40) {
   for (int j = 0; j < 500; j = j+40) {
         fill(random(0,255),random(0,255),random(0,255));
         stroke(random(0,255),random(0,255),random(0,255));
         rect(i, j, 40, 40);
         fill(random(0,255),random(0,255),random(0,255));
         stroke(random(0,255),random(0,255),random(0,255));
         rect(i, j, 30, 30);
  }
 }
}
```

Can you take your program to the extreme and see how far and crazy you can make it?

DON'T FORGET TO SAVE YOUR WORK FROM THIS SESSION!