Week 5: Loopy loops

So we've been drawing shapes, changing colours and composing our designs. It's time to look at how the computer can help *us* with the things that *the computer* is great at and our human math-ing is not so great at. Let's start by simply drawing a circle:

```
color[] colorList = new color[5];

void setup(){
  size(900,900);
  noStroke();
  colorMode(HSB,360,100,100);
  colorList[0] = color(199,100,90);
  colorList[1] = color(186,58,90);
  colorList[2] = color(9,78,94);
  colorList[3] = color(322,34,100);
  colorList[4] = color(13,24,100);
}

void draw(){
}

void mousePressed(){
  background(colorList[int(random(5))]);
  fill(colorList[int(random(5))]);
  ellipse(50,50,100,100);
}

void keyPressed(){
  saveFrame("image-###.jpg");
}
```

So now let's say we want to draw a row of circles, we could copy and paste line 19 and 20 over and over again, calculating the new coordinates and filling up the row. Bearable, but now what if you want to draw 100 smaller circles?! So this is where the **for loop** comes to the rescue. The for loop is denoted in the following way:

```
for (init; test; update) {
  statements
}
```

1. The `init` statement is run.
2. The `test` is evaluated to be true or false.
3. If the `test` is true, jump to step 4. If the `test` is false, jump to step 6.
4. Run the statements within the block.
5. Run the `update` statement and jump to step 2.
6. Exit the loop.

technecolour

If this looks overwhelming, don't worry. It's quite straight forward once we talk in human words.

`init =` "Computer, start the variable's value here,..."
`test =` "...keep running the statements until this is false"
`Update =` "...and each time, update the value."

Therefore, let's step through what we know:
1. Our canvas is `900px` wide
2. Our circles are `100px` wide
3. Therefore, we need `9` circles to fill a row.

`init` - "Computer, start the variable `i` at the value `0` "

```
for (int i = 0; test; update){
  statements
}
```

`test` - "...keep running the statements until `i < 9` is false..."

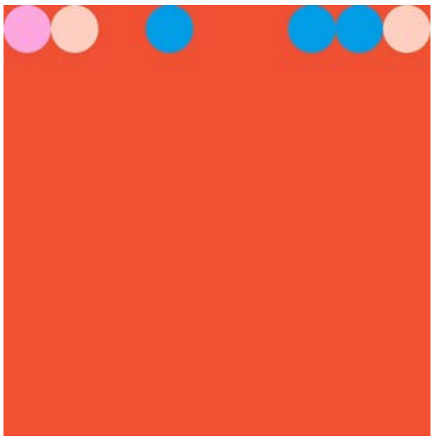
```
for (int i = 0; i < 9; update){
  statements
}
```

If we want `9` circles in a row, each time we run the loop of statements, we want to increment the circle's `x` value by `100px` so that we're not drawing 9 circles on top of one another. Therefore, we need to add a little maths to the circle's `x` value.

`update` - "...and each time, increment `i` by `1`."

```
for (int i = 0; i > 9; i++){
  ellipse(50+(100*i),50,100,100);
}
```

So let's give it a shot:



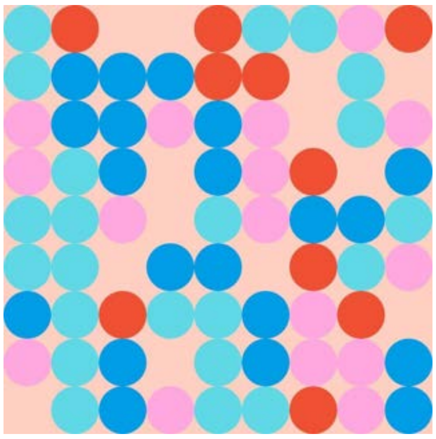
```
color[] colorList = new color[5];

void setup(){
  size(900,900);
  noStroke();
  colorMode(HSB,360,100,100);
  colorList[0] = color(199,100,90);
  colorList[1] = color(186,58,90);
  colorList[2] = color(9,78,94);
  colorList[3] = color(322,34,100);
  colorList[4] = color(13,24,100);
}

void draw(){
}

void mousePressed(){
  background(colorList[int(random(5))]);
  for (int i = 0; i < 9; i++){
    fill(colorList[int(random(5))]);
    ellipse(50+(100*i),50,100,100);
    println(i);
  }
}

void keyPressed(){
  saveFrame("image-###.jpg");
}
```

1. Can you work out why sometimes some circles are missing?

2. Now, can you work out how to repeat this row of 9 circles nine times down the canvas?

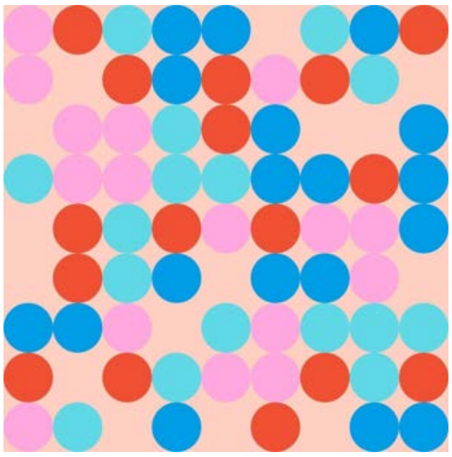
Hint: A loop can contain a loop!

How'd you go?



```
color[] colorList = new color[5];

void setup(){
  size(900,900);
  noStroke();
  colorMode(HSB,360,100,100);
  colorList[0] = color(199,100,90);
  colorList[1] = color(186,58,90);
  colorList[2] = color(9,78,94);
  colorList[3] = color(322,34,100);
  colorList[4] = color(13,24,100);
}

void draw(){
}

void mousePressed(){
  background(colorList[int(random(5))]);
  for (int i = 0; i < 9; i++){
    for (int j = 0; j < 9; j++){
      fill(colorList[int(random(5))]);
      ellipse(50+(100*i),50+(100*j),100,100);
    }
  }
}

void keyPressed(){
  saveFrame("image-###.jpg");
}
```

Let's have a look at a way we can improve this loop. Let's say we want to be able to change the size of the circles and the size of the canvas, but always draw enough circles to fill the canvas.



```
color[] colorList = new color[5];
int ellipseSize = 20;

void setup(){
  size(400,650);
  noStroke();
  colorMode(HSB,360,100,100);
  colorList[0] = color(199,100,90);
  colorList[1] = color(186,58,90);
  colorList[2] = color(9,78,94);
  colorList[3] = color(322,34,100);
  colorList[4] = color(13,24,100);
}

void draw(){
}

void mousePressed(){
  background(colorList[int(random(5))]);

  for (int i = 0; i < width; i+=ellipseSize){
    for (int j = 0; j < height; j+=ellipseSize){
     fill(colorList[int(random(5))]);
     ellipse((ellipseSize/2)+i, (ellipseSize/2)+j,
     ellipseSize,ellipseSize);
    }
  }
}

void keyPressed(){
  saveFrame("image-###.jpg");
}
```

See if you can explain the new loop in plain English:

```
for (int i = 0; i < width; i+=ellipseSize){
  for (int j = 0; j < height; j+=ellipseSize){
    fill(colorList[int(random(5))]);
    ellipse((ellipseSize/2)+i,(ellipseSize/2)+j,ellipseSize,ellipseSize);
  }
}
```

`init` - "Computer, start the variable _____ at the value _____"
`test` - "...keep running the statements until ___________________"
`update` - "...and each time, increment/decrement ____ by ____."

To test the sketch, change the size of the ellipse and the size of the canvas.
Isn't that nicer than specifying and respecifying every coordinate, every time?

Week 5 Exercise

Have a look at some of your inspiration from The Field. Can you spot any repetition?

Repeat a 2D primitive(other than an ellipse) using a for loop that is not dependant on a particular canvas size. Borrow colours from your inspiration image. Bonus props for nested loops(that’s the loop within a loop) and fun randomness that breaks the grid!