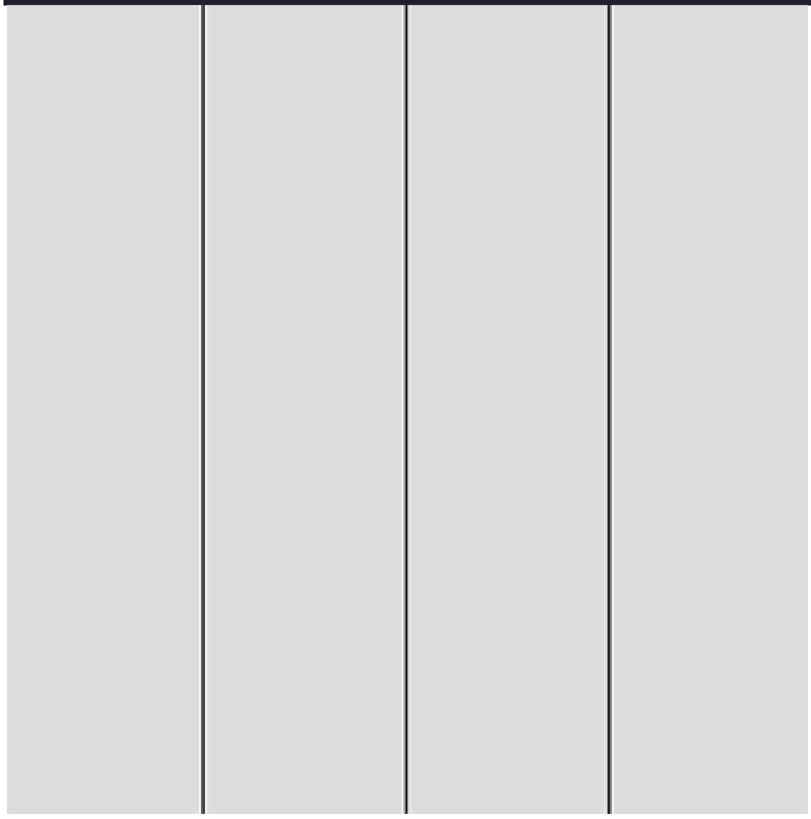
Creating Patterns with For Loop

18 min

Now that we know the basics of drawing with p5.js, we can begin to make complex designs based on algorithmic rules. So far, our code has worked by executing each line of code one after another. If we were to draw four lines, we would write four separate calls to the `line()` function. Instead, we can use the `for` loop to draw multiple shapes without having to repeat the code. We can also produce interesting patterns by modifying arguments of shape and style functions upon each iteration.

Let's take a look at the sketch below that draws three lines:

```
function draw() {
 line(75, 0, 75, height);
 line(150, 0, 150, height);
 line(225, 0, 225, height);
}
```

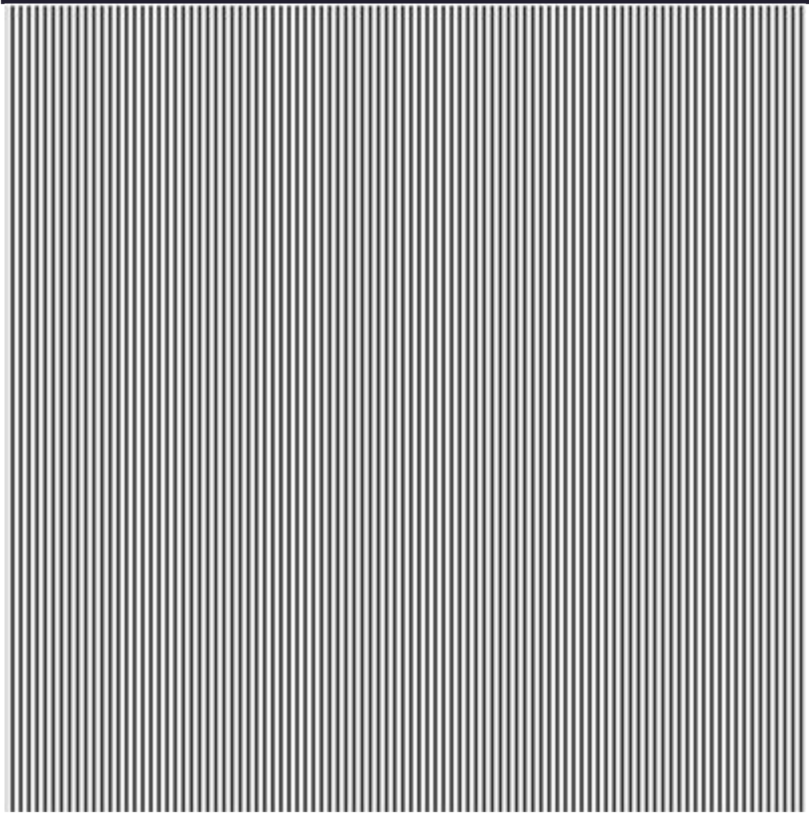


Notice that each line we draw uses the same y positions, but each x position increments by 75 pixels. We can simplify the above sketch by using a `for` loop and adding multiples of 75 to each iteration's x position.

```
function draw() {
  for(let lineX = 0; lineX < 4; lineX++){
    line(75 + lineX * 75, 0, 75 + lineX * 75, height);
  }
}
```

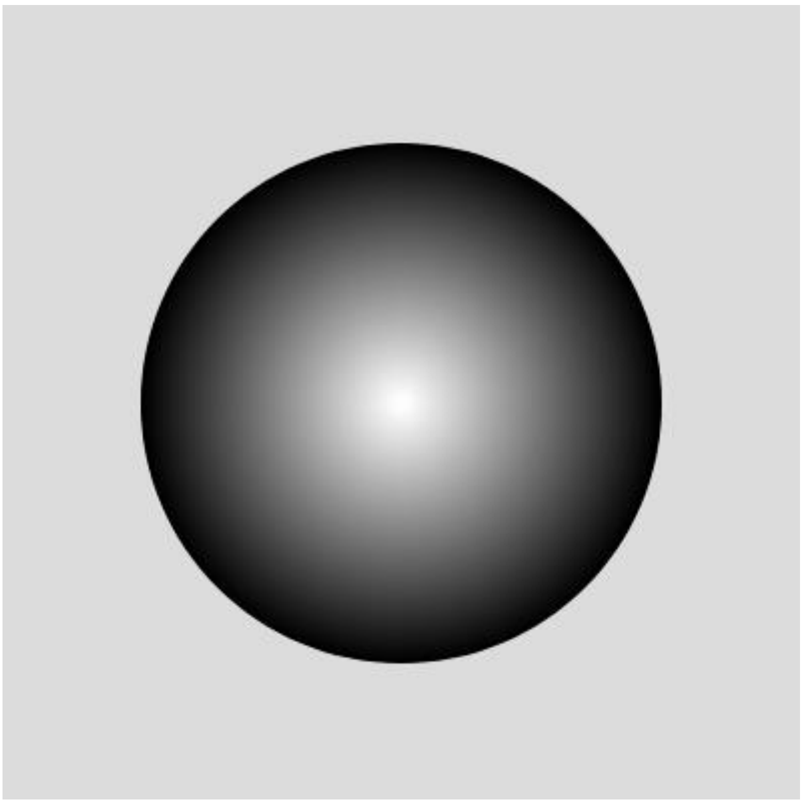
This might not seem like a lot now, but what if you want to draw 99 lines?

```
function draw() {
  for(let lineX = 0; lineX < 99; lineX++){
    line(4 + lineX * 4, 0, 4 + lineX * 4, height);
  }
}
```



You can use use `for` loops to change a number of parameters such as position, fill, stroke, color, size, and more! Take a look at the sketch below that draws 255 ellipses. With each iteration, the fill color increments from black to white, and the size decrements from 255 to 1.

```
function draw() {
  background(220);
  for(let i = 0; i < 255; i++){
    fill(i);
    circle(width/2, height/2, 255-i);
  }
}
```



You can put any code within a `for` loop, so why not another `for` loop? A common use case for nested `for` loops is to make grids of shapes. You use one `for` loop to form a row, and another `for` loop to repeat the row to form a grid!

```
function draw() {
  for(let posX = 0; posX < 4; posX++)  {
    for(let posY = 0; posY < 4; posY++)  {
      circle(posX * 25, posY * 25, 50);
    }
  }
}
```

Instructions

1.

The p5.js sketch to the right draws a row of rectangles made by calling the `rect()` function five times. Refactor the code to draw the row of rectangles using one `for` loop.
Hint
Write a `for` loop that calls the `rect()` function. To calculate the x positions of the rectangles, multiply the iterator variable by 75 and add 25 (the starting x position of the left-most rectangle).

2.

Let’s add another `for` loop inside the one you just created to draw a 5 by 5 grid of rectangles.

Hint

To calculate the y positions of the rectangles, multiply the iterator variable by 75 and add 25 (the starting y position of the left-most rectangle). Take a look at the nested `for` loop code example in the lesson above if you need a refresher!

sketch.js

```
function setup() {
  createCanvas(400, 400);
  background(200);
}

function draw() {
  // TODO: Comment out the below rectangle functions
  // rect(25, 25, 50, 50);
  // rect(100, 25, 50, 50);
  // rect(175, 25, 50, 50);
  // rect(250, 25, 50, 50);
  // rect(325, 25, 50, 50);

  // TODO: Draw a grid of rectangles using for loops
  for(let posX = 0; posX < 5; posX++){
    for(let posY = 0; posY < 5; posY++){
      rect(posX * 75 + 25, posY * 75 + 25, 50, 50);
    }
  }
}
```