Section A
Unit #12
10Print

Content

Section A
Unit #12 10PRINT

Unit #11 10PRINT

It comes from an abbreviation of a single line algorithm written in BASIC in the early 1980s: 10 PRINT CHR$(205.5+RND(1)); : GOTO 10

If you remember coding in BASIC then you are possibly older than you look. This single line of code produces a random pattern and this section uses this principle for creating random patterns.

This unit also introduces nested loops. There is such a huge scope to play with this creating interesting designs and patterns with different shapes, colour, thicknesses etc. so it a great place to experiment.

Sketch A12.1 a line!

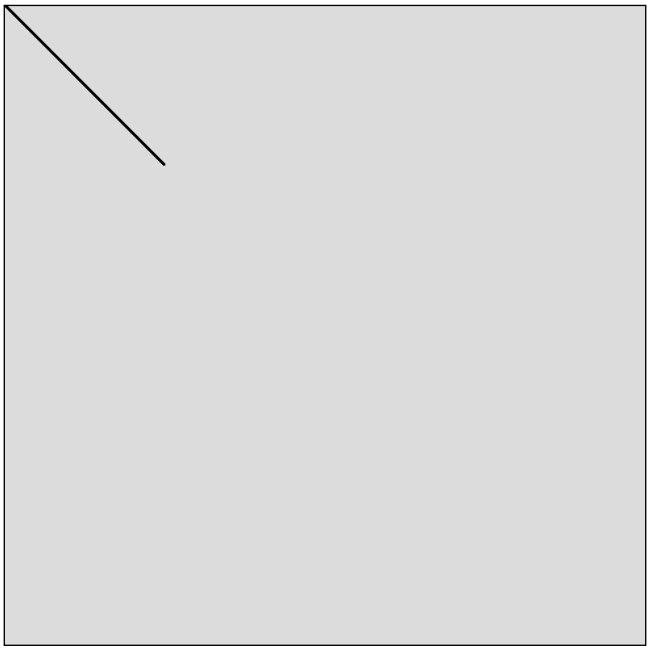
Drawing a line from the top left corner **(0, 0)** to a point **100** down and **100** to the left.

```
let x = 0
let y = 0

function setup()
{
  createCanvas(400, 400)
  background(220)
}

function draw()
{
  line(x, y, x + 100, y + 100)
}
```

Draw a line from (0, 0) to (100, 100)



Sketch A12.2 another variable

Instead of typing **100** we will create a variable called **spacing** and give that the value of **100** which will give us exactly the same result.

```
let x = 0
let y = 0
let spacing = 100

function setup()
{
  createCanvas(400, 400)
  background(220)
}

function draw()
{
  line(x, y, x + spacing, y + spacing)
}
```

Sketch A12.3 using the variable

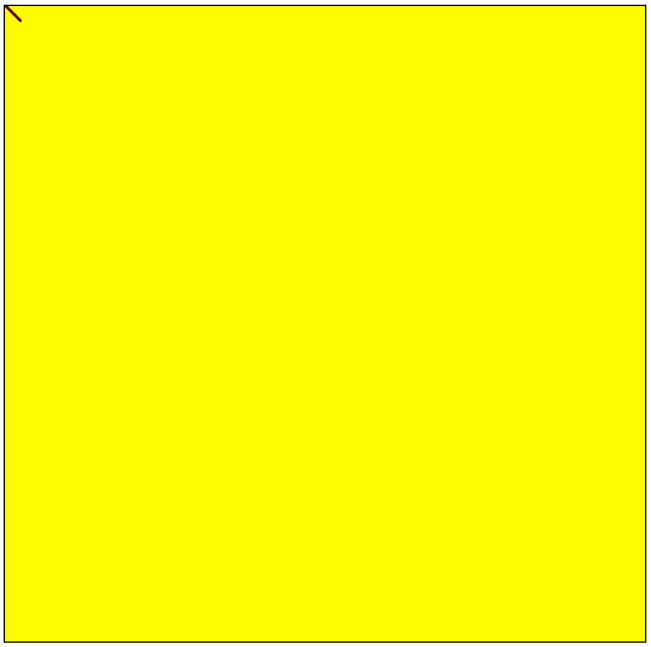
Let's now change the spacing variable and give the canvas a nice yellow background and make the line dark red

```
let x = 0
let y = 0
let spacing = 10

function setup()
{
  createCanvas(400, 400)
  background(255, 255, 0)
  stroke(100, 0, 0)
}

function draw()
{
  line(x, y, x + spacing, y + spacing)
}
```

Short red line and a nice yellow background



Challenges

1. Use a different colour by all means
2. Use a different value
3. Use a thicker line

Sketch A12.4 adding the spacing

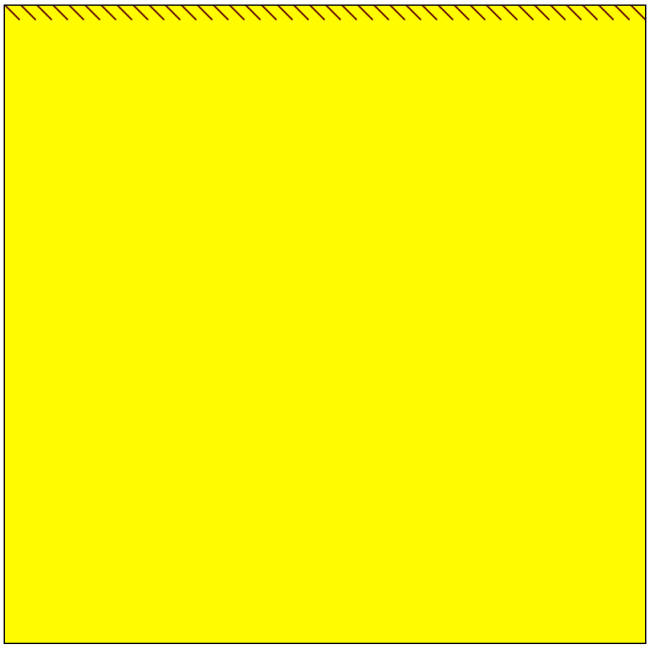
What we are going to do is now loop through to draw many of them evenly spaced out in a row using the variable **spacing**. You could just use the value **10** but having a variable means that you can change it once at the beginning and it will automatically change everywhere you used that variable.

```
let x = 0
let y = 0
let spacing = 10

function setup()
{
  createCanvas(400, 400)
  background(255, 255, 0)
  stroke(100, 0, 0)
}

function draw()
{
  line(x, y, x + spacing, y + spacing)
  x += spacing
}
```

A row of nicely spaced out lines



Sketch A12.5 stop at the edge

However the lines are being drawn beyond the canvas continually. What I would like to do is to start a new line when it gets to the right hand edge. To do this we use an **`if()`** statement. An **`if()`** statement checks to see if something has happened and if it has we tell it to do something different.

```
let x = 0
let y = 0
let spacing = 10

function setup()
{
  createCanvas(400, 400)
  background(255, 255, 0)
  stroke(100, 0, 0)
}

function draw()
{
  line(x, y, x + spacing, y + spacing)
  x += spacing
  if (x >= width)
  {
    x = 0
  }
}
```

Notes

The symbol **>=** means greater than or equal to. So if the value of **x** is greater than or equal to the width it goes back to zero. So what you are seeing is the red line starting again from the left hand edge every time it reaches the right hand edge. You can't see it happen because it is writing it over and over again on the same line.

Sketch A12.6 let's go down

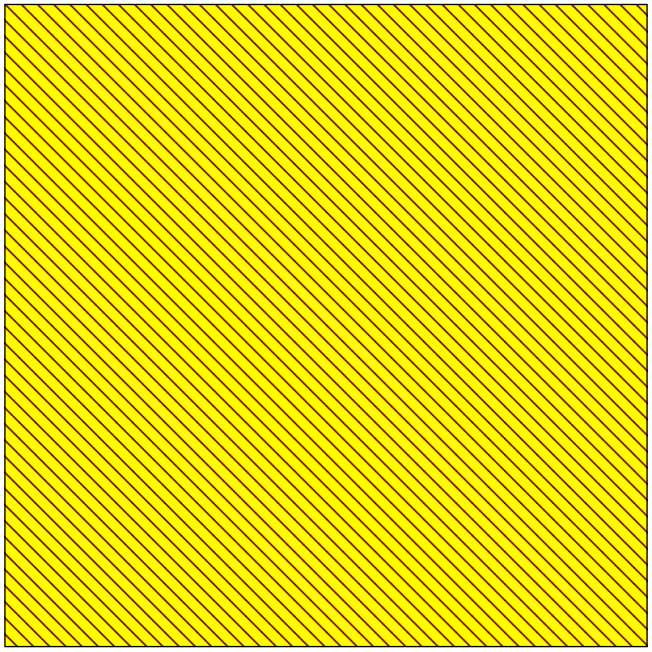
But what I would like is to start a new line after it gets to the edge and so we need to nudge it down by the same amount. To do this we add **spacing** value to **y**.

```
let x = 0
let y = 0
let spacing = 10

function setup()
{
  createCanvas(400, 400)
  background(255, 255, 0)
  stroke(100, 0, 0)
}

function draw()
{
  line(x, y, x + spacing, y + spacing)
  x += spacing
  if (x >= width)
  {
    x = 0
    y += spacing
  }
}
```

It draws that red line on each row



Sketch A12.7 stop when we get to the bottom

We use **noLoop()** when we get to the bottom, it stops the **draw()** function from looping round. You should get the same result as before.

```
let x = 0
let y = 0
let spacing = 10

function setup()
{
  createCanvas(400, 400)
  background(255, 255, 0)
  stroke(100, 0, 0)
}

function draw()
{
  line(x, y, x + spacing, y + spacing)
  x += spacing
  if (x >= width)
  {
    x = 0
    y += spacing
  }
  if (y >= height)
  {
    noLoop()
  }
}
```

Sketch A12.8 sloping the other way

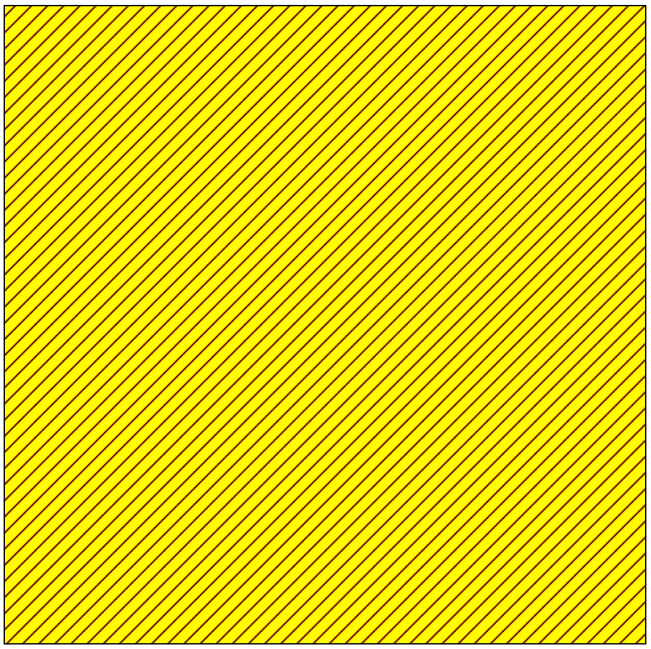
Let's change it so that it draws the line sloping the other way. We can use nearly the same code.

```
let x = 0
let y = 0
let spacing = 10

function setup()
{
  createCanvas(400, 400)
  background(255, 255, 0)
  stroke(100, 0, 0)
}

function draw()
{
  line(x + spacing, y, x, y + spacing)
  x += spacing
  if (x >= width)
  {
    x = 0
    y += spacing
  }
  if (y >= height)
  {
    noLoop()
  }
}
```

Now sloping the opposite way



Sketch A12.9 the other line 50%

We want to draw one line **50%** of the time and then the other **50%** in a random order, to do that we can simply pick a number at random from **0** to 1 and if it is less or more than **0.5** then do one or the other.

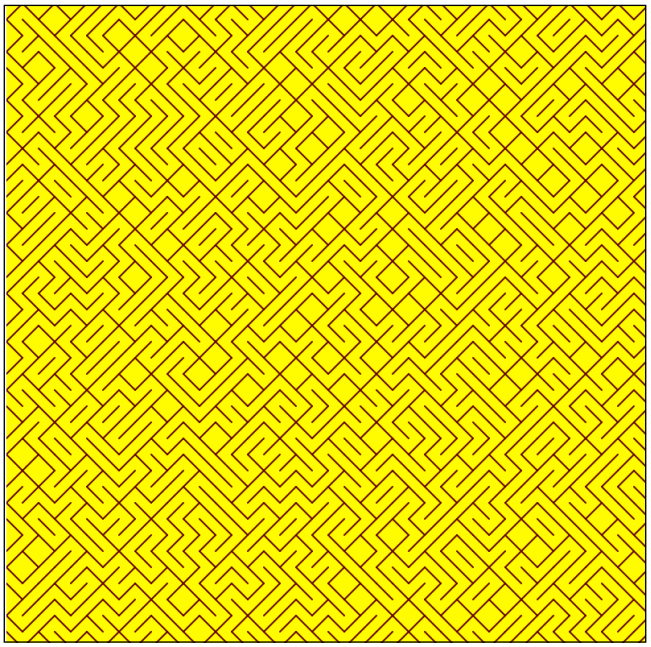
```
let x = 0
let y = 0
let spacing = 10

function setup()
{
  createCanvas(400, 400)
  background(255, 255, 0)
  stroke(100, 0, 0)
}

function draw()
{
  if (random(1) < 0.5)
  {
    line(x, y, x + spacing, y + spacing)
  }
  else
  {
    line(x + spacing, y, x, y + spacing)
  }
  x += spacing
  if (x >= width)
  {
    x = 0
    y += spacing
  }
```

```
  if (y >= height)
  {
    noLoop()
  }
}
```

A pleasing effect



Notes

Now we have a random pattern

Challenges

1. What happens when you change the percentage eg 0.5 to 0.8
2. Use other colours (random colours)
3. Thicker strokeWeight()

Sketch A12.10 Let's start from scratch

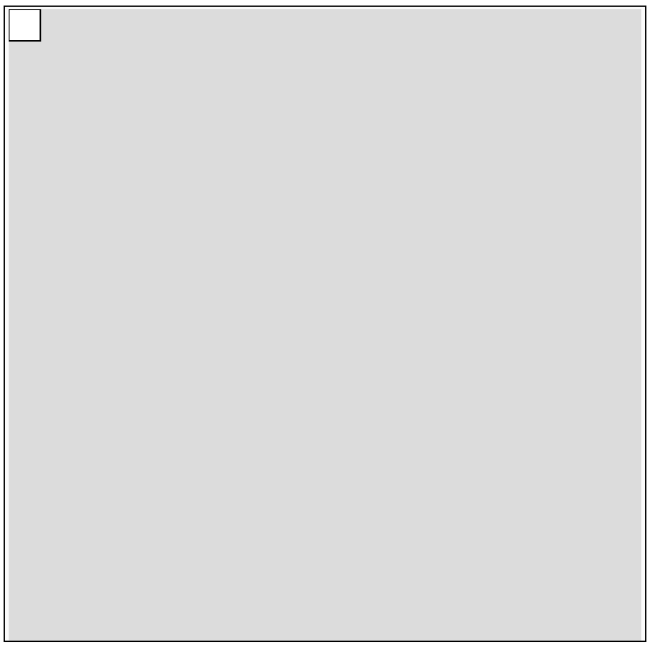
Start a completely new sketch. Instead of a line we are going to draw a square, we will draw it at the top corner. The square takes three arguments, the first two are the x and y ordinates and the third is the length of the side. We will use the variable spacing as it will come in handy. We will make it a little bigger

```
let x = 0
let y = 0
let spacing = 20

function setup()
{
  createCanvas(400, 400)
  background(220)
}

function draw()
{
  square(x, y, spacing)
}
```

A solitary square



Sketch A12.11 A row of squares

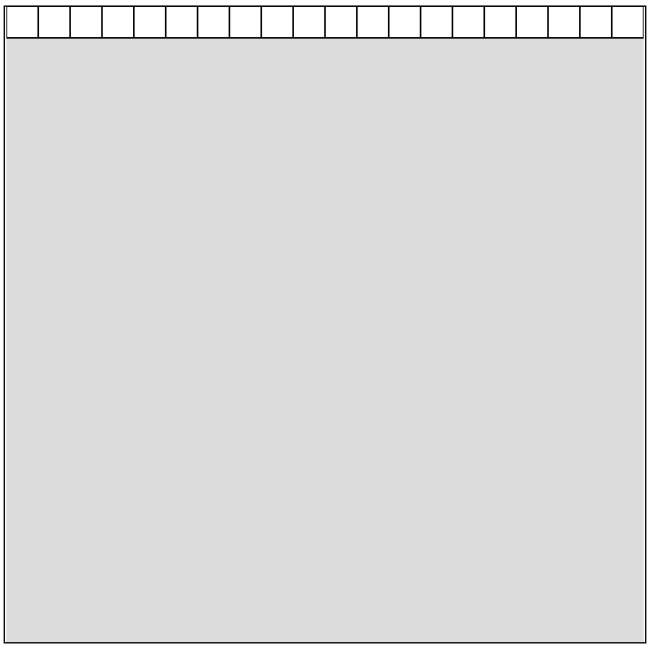
We are going to draw a row of them using a for loop. The for loop has three parts to it. The first part (**`let i = 0;`**) creates a variable called i and gives it a value of 0. The second part tells the loop when to stop, in this case when i gets to width (**`i < width;`**). The third part is how many steps (jumps) it takes (**`i += spacing`**). We set the spacing to 20, so we are jumping in 20's from 0 to 400 (the width).

```
let x = 0
let y = 0
let spacing = 20

function setup()
{
  createCanvas(400, 400)
  background(220)
}

function draw()
{
  for (let i = 0; i < width; i += spacing)
  {
    square(i, y, spacing)
  }
}
```

A loop to draw a row of squares



Notes

We change the x co-ordinate from x to i. This means that it shuffles along as each time i is increased by spacing (20)

Sketch A12.12 rows and columns

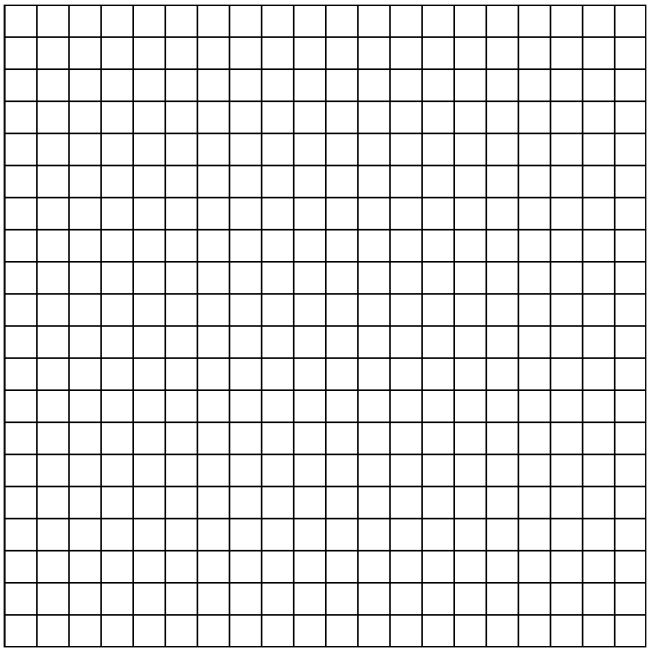
Adding columns as well as rows to fill the canvas

```
let x = 0
let y = 0
let spacing = 20

function setup()
{
  createCanvas(400, 400)
  background(220)
}

function draw()
{
  for (let i = 0; i < width; i += spacing)
  {
    for (let j = 0; j < height; j += spacing)
    {
      square(i, j, spacing)
    }
  }
}
```

Rows and columns of squares



Notes

We use the letters i and j for no other reason that it is convention. You will see it a lot if using nested loops, if you want a third one then k is used etc. You can use any later or variable name. We could've used x and y and sometimes it is better to do that but we had already used them.

Sketch A12.13 random size squares

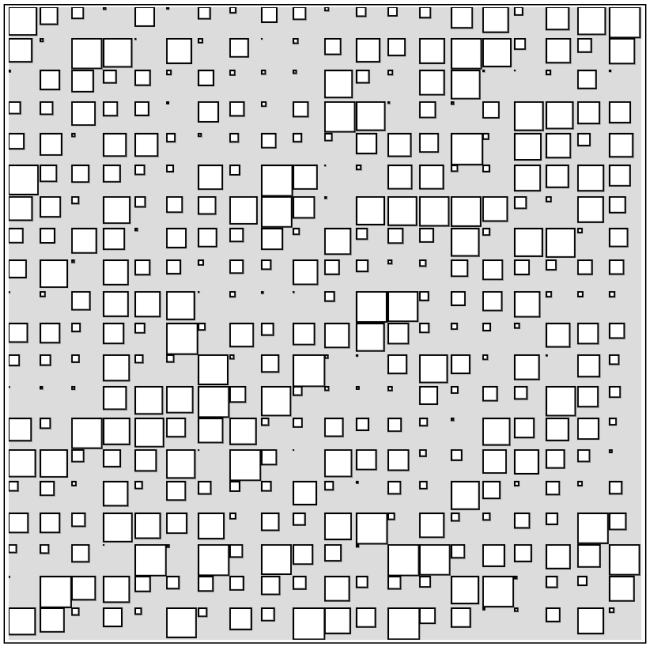
We can have some more fun with random, instead of having the same size squares what if we change the size every time we draw one! We will need a **noLoop()** function to stop it waggling all the time

```
let x = 0
let y = 0
let spacing = 20

function setup()
{
  createCanvas(400, 400)
  background(220)
}

function draw()
{
  for (let i = 0; i < width; i += spacing)
  {
    for (let j = 0; j < height; j += spacing)
    {
      square(i, j, random(spacing))
    }
  }
  noLoop()
}
```

Random sized squares



Sketch A12.14 using rectMode(CENTER)

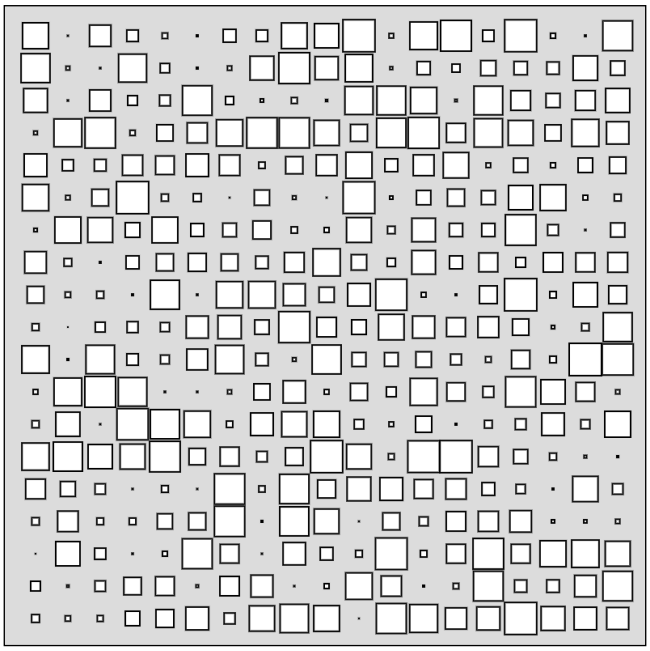
A little bit of refactoring. At the moment the co-ordinates of the square are at the top left hand corner of the square. To change that so that the co-ordinates are in the centre of the square we use a command called **rectMode(CENTER)**. Also to give it a border we start with spacing for **i** and **j** rather than zero.

```
let x = 0
let y = 0
let spacing = 20

function setup()
{
  createCanvas(400, 400)
  background(220)
  rectMode(CENTER)
}

function draw()
{
  for (let i = spacing; i < width; i += spacing)
  {
    for (let j = spacing; j < height; j += spacing)
    {
      square(i, j, random(spacing))
    }
  }
  noLoop()
}
```

Co-ordinates in the centre of the square
and a border



Notes

So much tidier and pleasing to the eye.

Challenge

Make the each square a different colour

Sketch A12.15 random colours

A bit more fun will be to colour them each randomly. We will make the background white to show up the colours, also create a small gap between squares

```
let x = 0
let y = 0
let spacing = 20

function setup()
{
  createCanvas(400, 400)
  background(255)
  rectMode(CENTER)
}

function draw()
{
  for (let i = spacing; i < width; i += spacing)
  {
    for (let j = spacing; j < height; j += spacing)
    {
      fill(random(255), random(255), random(255))
      square(i, j, random(spacing - 2))
    }
  }
  noLoop()
}
```

Random tiling