

An Introduction to Processing

Formatting shapes

Produced Dr. Siobhán Drohan
by: Mr. Colm Dunphy
 Mr. Diarmuid O'Connor



Waterford Institute of Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

Department of Computing and Mathematics
<http://www.wit.ie/>

Topics list

- Filling shapes with colour.
- Formatting the shape outline.
- Adding comments to your code.

fill() - syntax

`fill (r, g, b)`

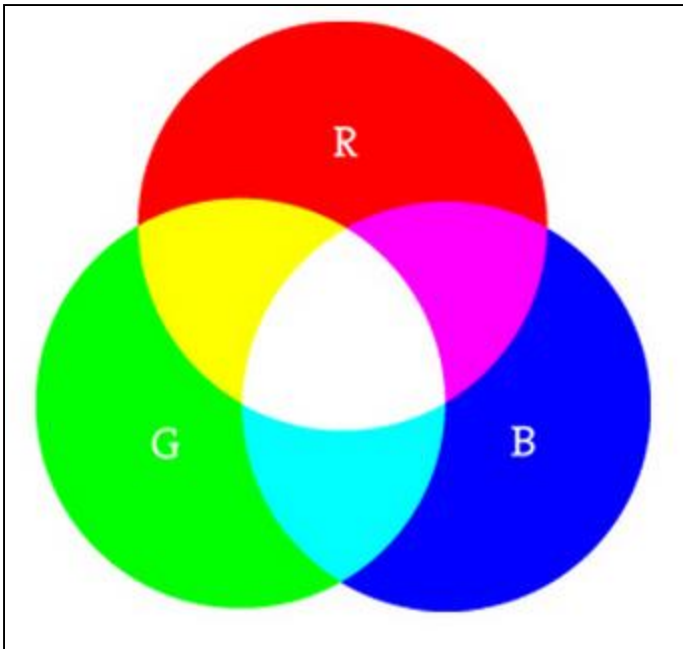
`r` = red colour (a number between 0 and 255 inclusive)

`g` = green colour (a number between 0 and 255 inclusive)

`b` = blue colour (a number between 0 and 255 inclusive)

- fills shapes with a chosen colour.
- can use the RGB colours to select a colour.
- all shapes drawn after the **fill** function is called, will be filled with the chosen colour.

A recap of RGB colours



Digital colours are made by mixing the three primary colours of light (red, green, and blue).

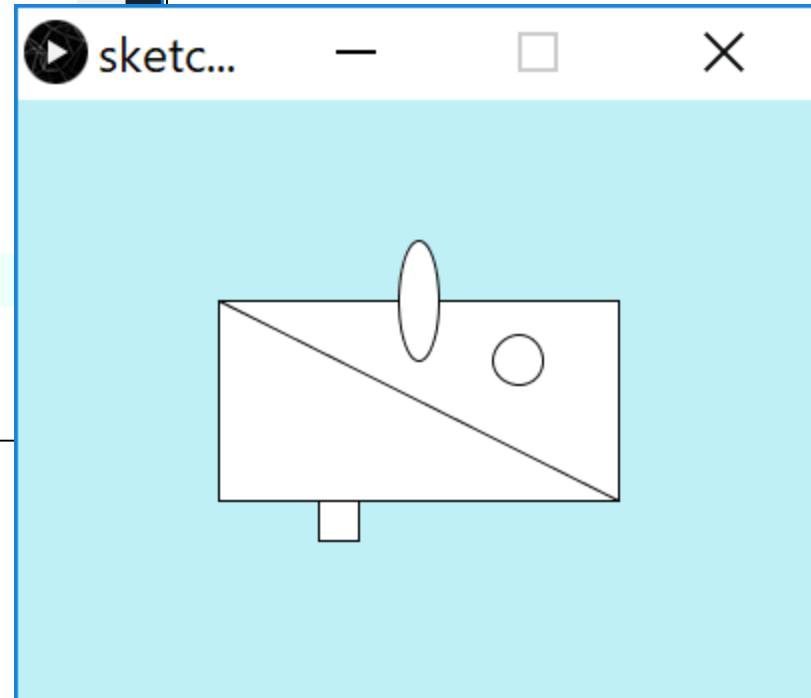
“As with grayscale, the individual color elements are expressed as ranges from 0 (none of that color) to 255 (as much as possible), and they are listed in the order R, G, and B.”

fill()

```
sketch_180103a | Processing 3.3.6
File Edit Sketch Debug Tools Help

[Play] [Stop] [Debugger] Java ▾

sketch_180103a ▾
1 size(400,300);
2 background(190, 240, 245);
3
4 rect(100,100,200,100);
5 rect(150,200,20,20);
6 line(100,100,300,200);
7 ellipse(200,100,20,60);
8 ellipse(250,130,25,25);
```



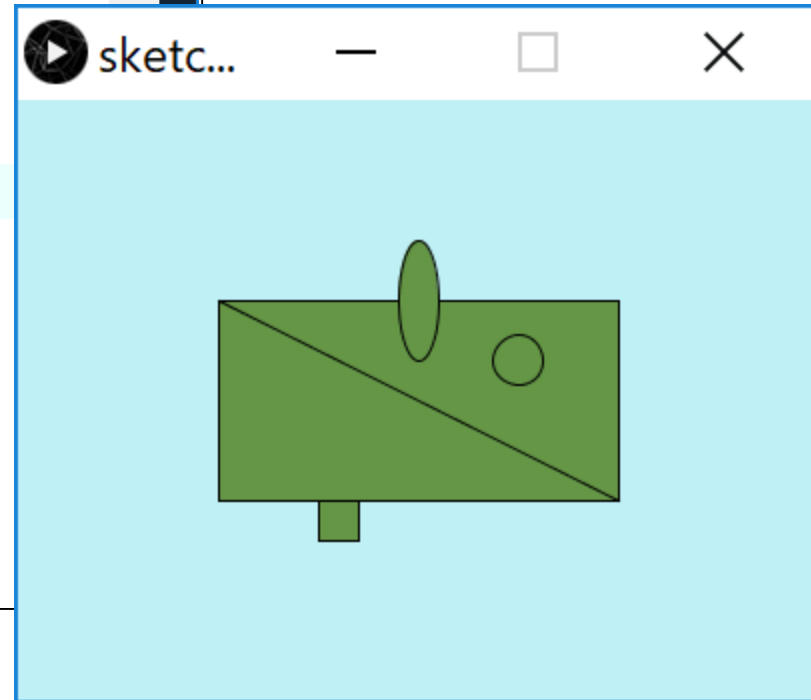
Before using fill(), all shapes filled with default of white.

fill()

```
sketch_180103a | Processing 3.3.6
File Edit Sketch Debug Tools Help

sketch_180103a
1 size(400,300);
2 background(190, 240, 245);
3
4 fill(100, 150, 70);
5
6 rect(100,100,200,100);
7 rect(150,200,20,20);
8 line(100,100,300,200);
9 ellipse(200,100,20,60);
10 ellipse(250,130,25,25);
11
```

Now all shapes are filled with dark green.

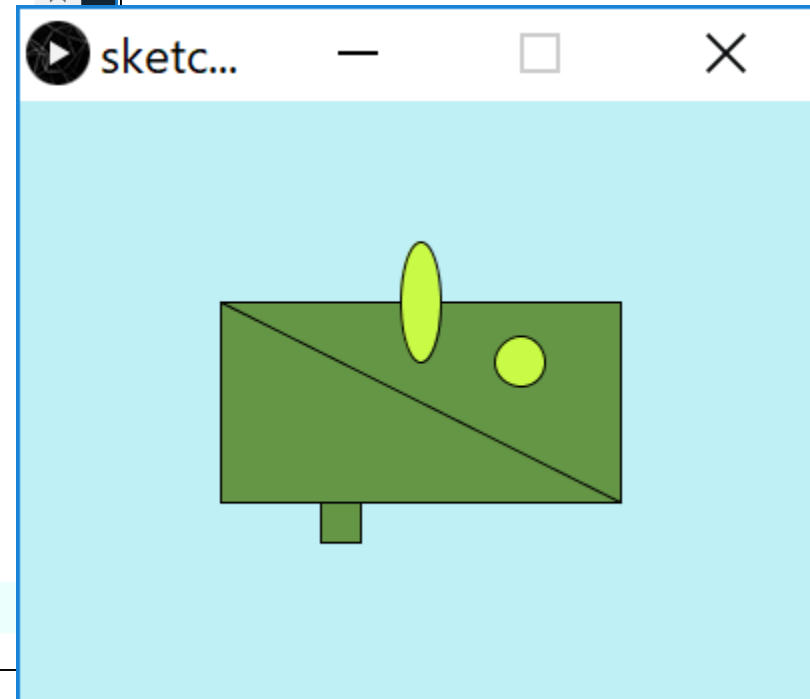


fill()

Rectangles filled with dark green. Ellipses filled with light green; order of statements matter!!!

```
sketch_180103a | Processing 3.3.6
File Edit Sketch Debug Tools Help

sketch_180103a
1 size(400,300);
2 background(190, 240, 245);
3
4 fill(100, 150, 70);
5
6 rect(100,100,200,100);
7 rect(150,200,20,20);
8 line(100,100,300,200);
9
10 fill(200,250,70);
11
12 ellipse(200,100,20,60);
13 ellipse(250,130,25,25);
14
```

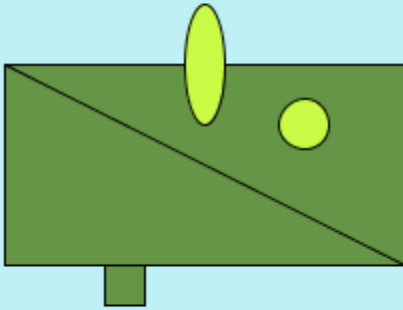


Topics list

- Filling shapes with colour.
- Formatting the shape outline.
- Adding comments to your code.

Changing the outline (i.e. stroke)

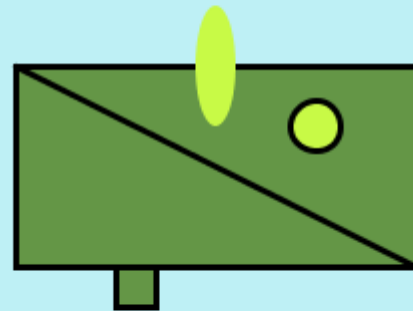
Before



After (changes):

- The oval has no border; all other shapes do.
- The outline is heavier.

We will now make those changes



noStroke() - syntax

```
noStroke();
```

```
//no parameters defined for this function.
```

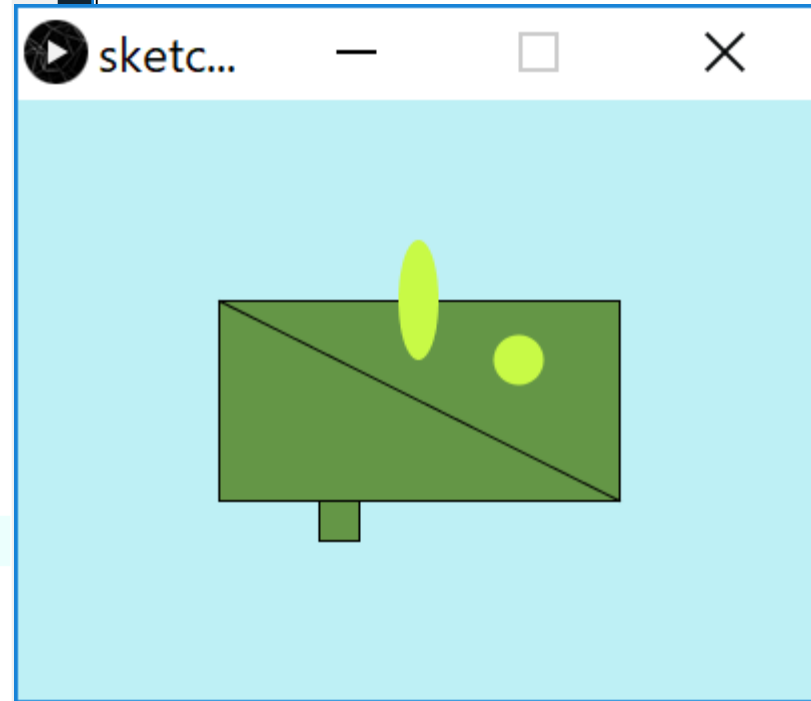
- A **stroke** is the outline of a shape.
- The noStroke() function disables the outline on shapes that are drawn after the function is called.
- All shapes drawn after the **noStroke** function is called, will have no outline.

noStroke()

```
sketch_180103a | Processing 3.3.6
File Edit Sketch Debug Tools Help

sketch_180103a
1 size(400,300);
2 background(190, 240, 245);
3
4 fill(100, 150, 70);
5
6 rect(100,100,200,100);
7 rect(150,200,20,20);
8 line(100,100,300,200);
9
10 fill(200,250,70);
11
12 noStroke();
13 ellipse(200,100,20,60);
14 ellipse(250,130,25,25);
15
```

✓ We have no border on the oval shape.
✗ But now our circle also has no border.



stroke() - syntax

stroke (r, g, b)

r = red colour (a number between 0 and 255 inclusive)

g = green colour (a number between 0 and 255 inclusive)

b = blue colour (a number between 0 and 255 inclusive)

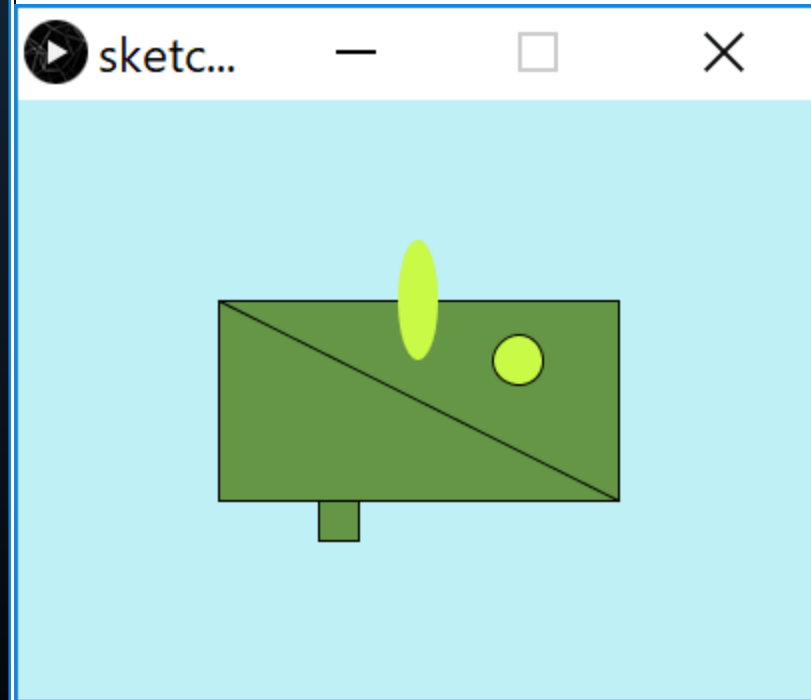
- The stroke() function enables the outline on all shapes that are drawn after the function is called.
- When you call stroke(), you need to specify a colour.

stroke()

```
sketch_180103a | Processing 3.3.6
File Edit Sketch Debug Tools Help

sketch_180103a
1 size(400,300);
2 background(190, 240, 245);
3
4 fill(100, 150, 70);
5
6 rect(100,100,200,100);
7 rect(150,200,20,20);
8 line(100,100,300,200);
9
10 fill(200,250,70);
11
12 noStroke();
13 ellipse(200,100,20,60);
14
15 stroke(0,0,0);
16 ellipse(250,130,25,25);
17
```

✓ Our circle now has a border.



strokeWeight() - syntax

strokeWeight (**pixels**)

pixels = thickness of the outline measures in pixels.

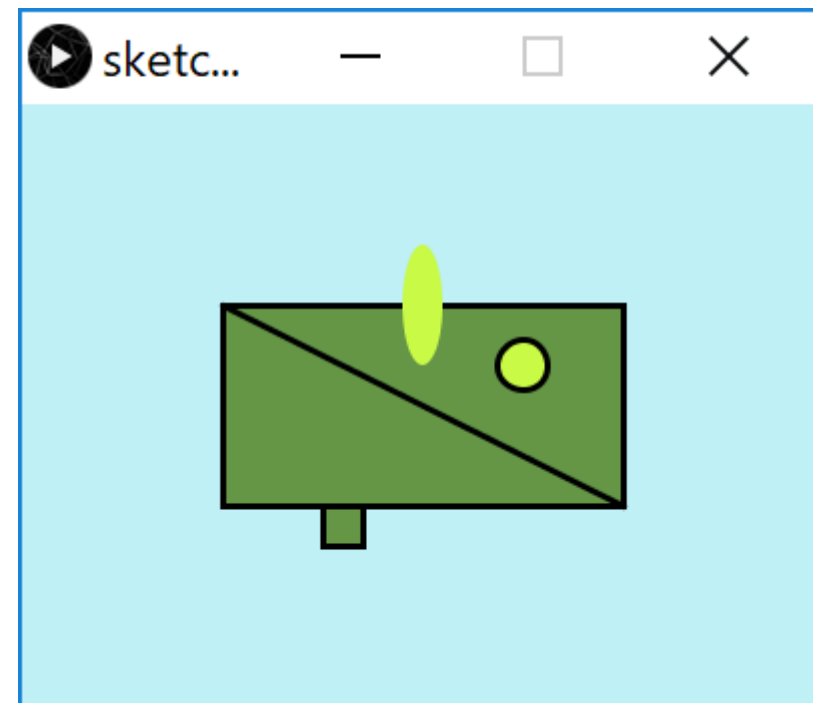
- The strokeWeight() function allows you to choose the thickness of a line/outline on shapes.
- The chosen thickness will apply to all lines/shapes that are drawn after the function is called.
- The thickness is specified in pixels.
- The default thickness is 1 pixel.

strokeWeight()

```
sketch_180103a | Processing 3.3.6
File Edit Sketch Debug Tools Help

sketch_180103a
1 size(400,300);
2 background(190, 240, 245);
3 strokeWeight(3);
4
5 fill(100, 150, 70);
6
7 rect(100,100,200,100);
8 rect(150,200,20,20);
9 line(100,100,300,200);
10
11 fill(200,250,70);
12
13 noStroke();
14 ellipse(200,100,20,60);
15
16 stroke(0,0,0);
17 ellipse(250,130,25,25);
```

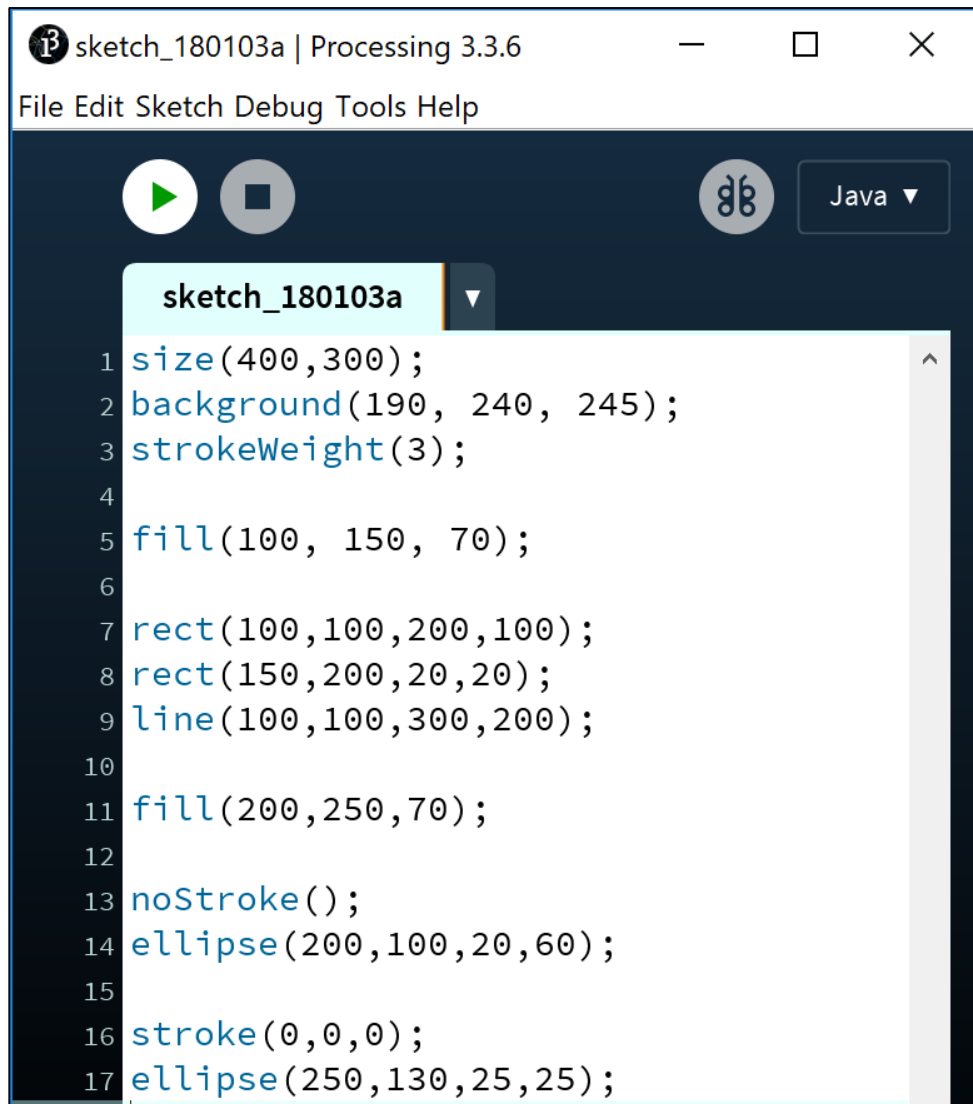
✓ Our outline is now heavier.



Topics list

- Filling shapes with colour.
- Formatting the shape outline.
- Adding comments to your code.

Code so far...



```
sketch_180103a | Processing 3.3.6
File Edit Sketch Debug Tools Help

sketch_180103a
1 size(400,300);
2 background(190, 240, 245);
3 strokeWeight(3);
4
5 fill(100, 150, 70);
6
7 rect(100,100,200,100);
8 rect(150,200,20,20);
9 line(100,100,300,200);
10
11 fill(200,250,70);
12
13 noStroke();
14 ellipse(200,100,20,60);
15
16 stroke(0,0,0);
17 ellipse(250,130,25,25);
```

Can you tell, from looking at the code, what RGB colours you have chosen?

- We can leave notes for ourselves and others in our code.
- This is called **commenting your code**.

Commenting your code...

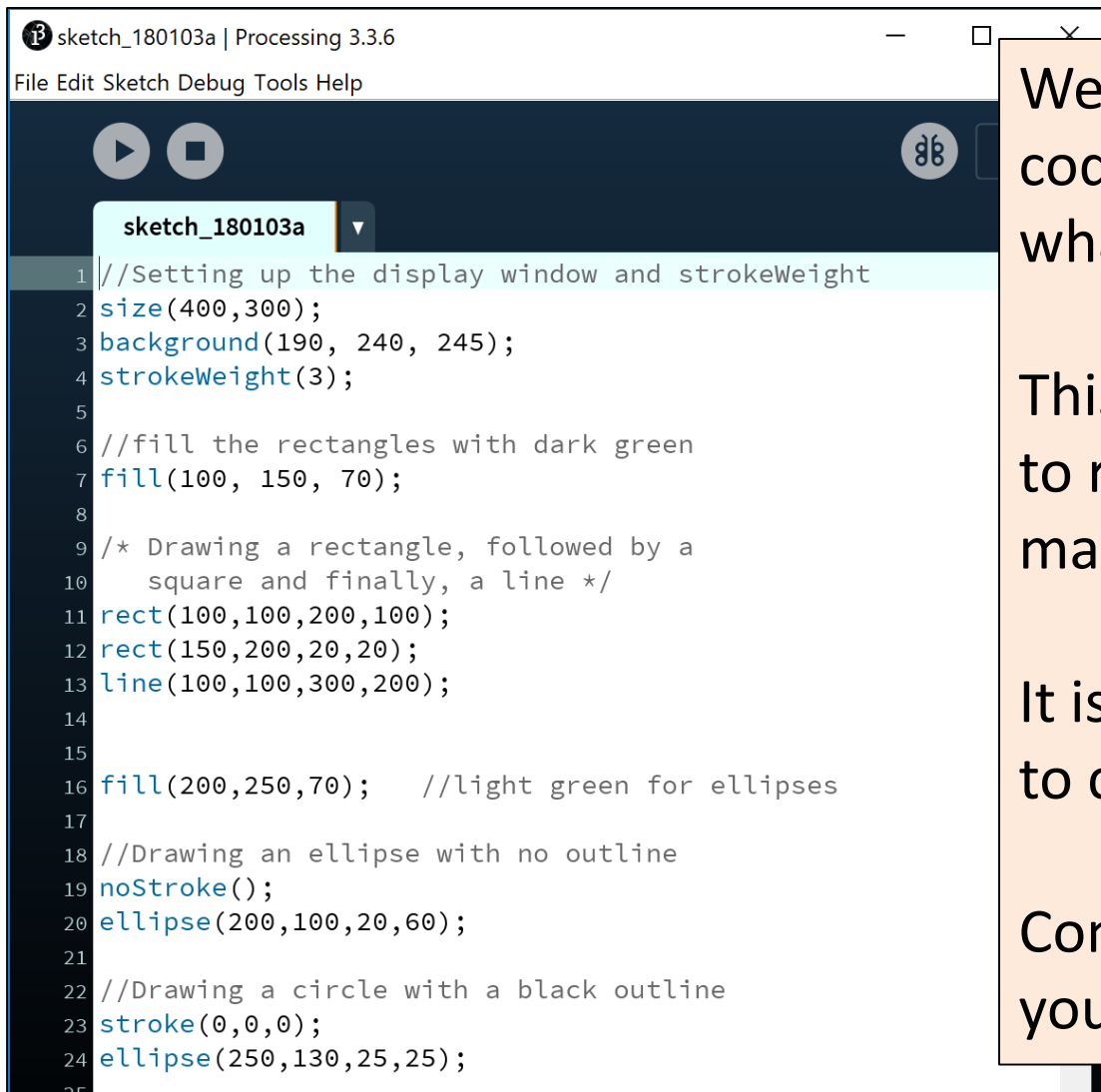
// This is a comment.

// Anything typed after the two slashes

// up to the end of the line, is ignored by Java.

/* This is a longer comment. As you can span more than one line with this comment style, it can be quite handy. */

Code so far...with commenting



The screenshot shows the Processing IDE interface. The title bar reads "sketch_180103a | Processing 3.3.6". The menu bar includes "File Edit Sketch Debug Tools Help". The toolbar contains a play button, a stop button, and a debug button. The code editor shows the following code:

```
1 //Setting up the display window and strokeWeight
2 size(400,300);
3 background(190, 240, 245);
4 strokeWeight(3);
5
6 //fill the rectangles with dark green
7 fill(100, 150, 70);
8
9 /* Drawing a rectangle, followed by a
10    square and finally, a line */
11 rect(100,100,200,100);
12 rect(150,200,20,20);
13 line(100,100,300,200);
14
15
16 fill(200,250,70); //light green for ellipses
17
18 //Drawing an ellipse with no outline
19 noStroke();
20 ellipse(200,100,20,60);
21
22 //Drawing a circle with a black outline
23 stroke(0,0,0);
24 ellipse(250,130,25,25);
25
```

We have commented our code with explanations of what is happening.

This makes our code easier to read, understand and maintain.

It is considered best practice to comment your code.

Comments do not affect your code at all.

Questions?

