Programming Fundamentals 1

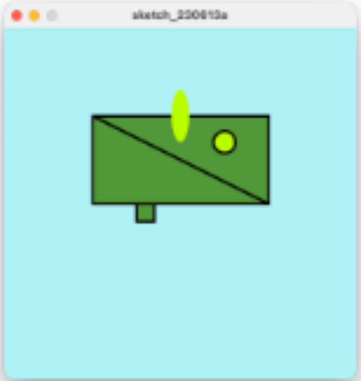
Produced by
Mr. Dave Drohan (david.drohan@setu.ie)
Dr. Siobhán Drohan
Ms. Mairead Meagher

Department of Computing & Mathematics
South East Technological University
Waterford, Ireland

Introduction to Processing

Static Drawings, Colour and more

Static Drawings

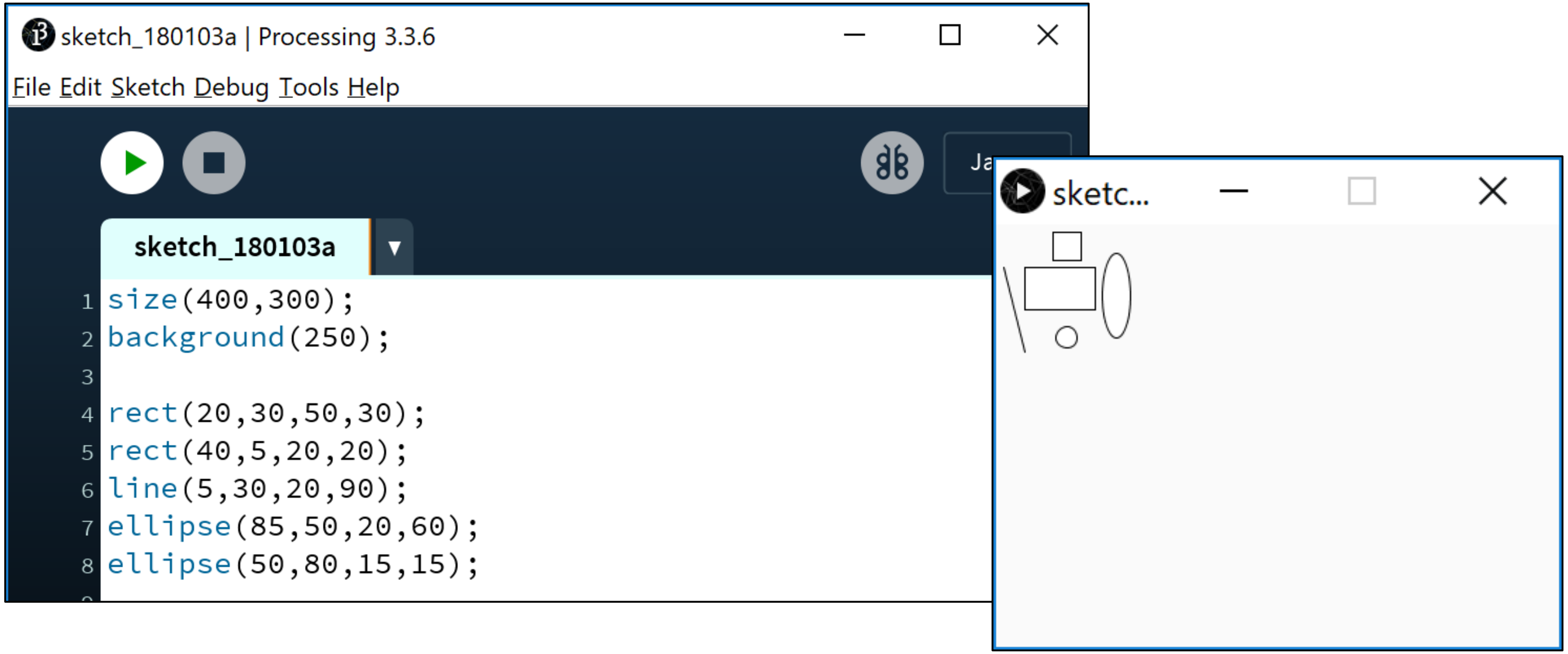


Colour · Stroke etc. · Errors · Comments

Agenda

- ❑ Static Drawing
- ❑ Colour & Filling Shapes
- ❑ Formatting Shape Outline
- ❑ Syntax and Logic Errors
- ❑ Comments

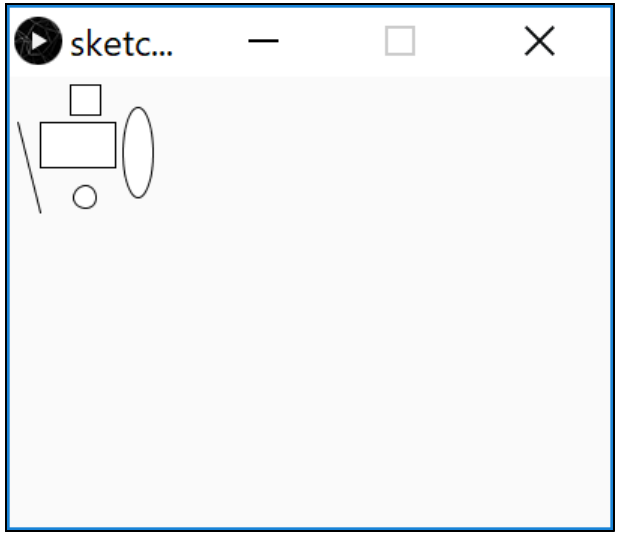
Static drawing – an example


sketch_180103a | Processing 3.3.6
File Edit Sketch Debug Tools Help
sketch_180103a
sketc...

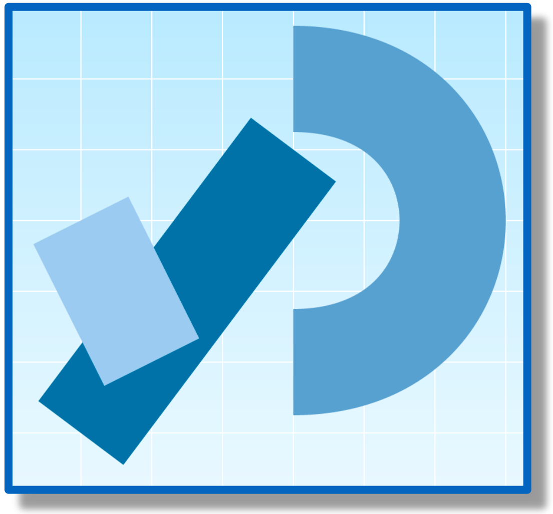
```
size(400,300);
background(250);

rect(20,30,50,30);
rect(40,5,20,20);
line(5,30,20,90);
ellipse(85,50,20,60);
ellipse(50,80,15,15);
```

Static drawing – an example

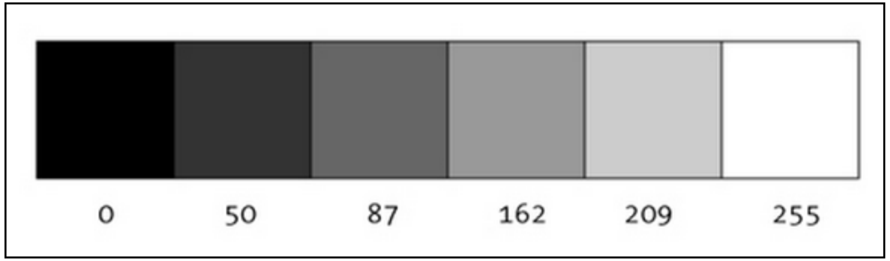
- Static drawings are those that don’t change over time:
  - Once they are drawn, they don’t change.
  - They don’t respond to events e.g. a mouse moving over the sketch, a key being pressed, etc.



Colour (Grayscale and RGB)



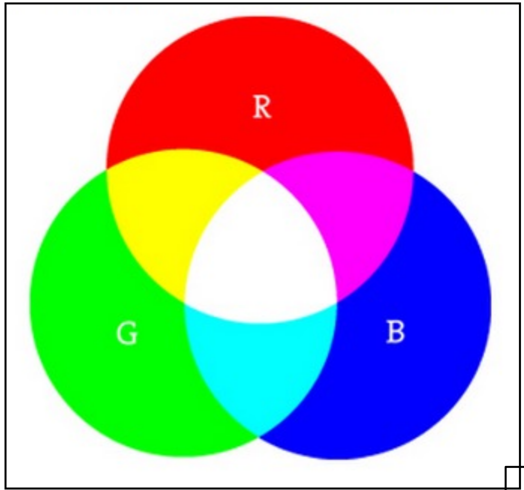
We looked at the Grayscale palette



"0 means black, 255 means white. In between, every other number - 50, 87, 162, 209, and so on - is a shade of gray ranging from black to white."

https://www.processing.org/tutorials/color/

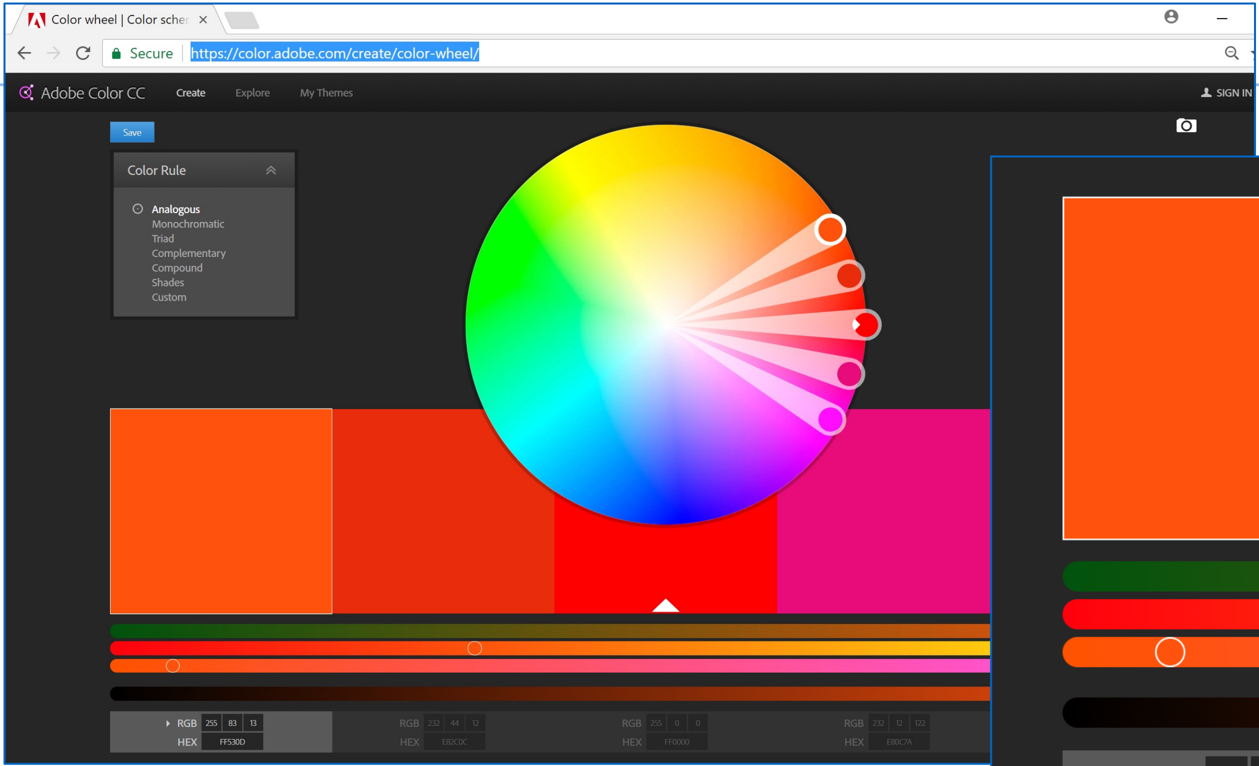
The RGB palette



"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."

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

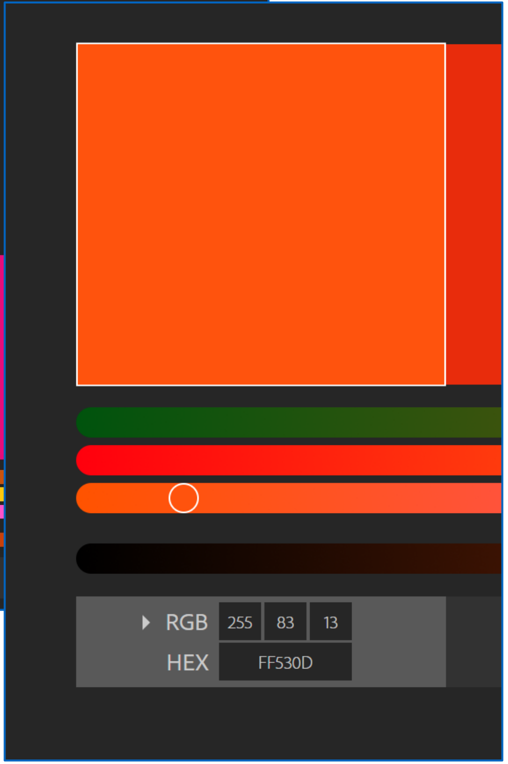
https://color.adobe.com/create/color-wheel/


Color wheel | Color scher
Secure
https://color.adobe.com/create/color-wheel/
Adobe Color CC
Create
Explore
My Themes
SIGN IN
Save

Color Rule

- [✓ Analogous]
- Monochromatic
- Triad
- Complementary
- Compound
- Shades
- Custom

RGB	255 83 13	232 44 12	255 0 0	232 12 122
HEX	FF530D	E82C0C	FF0000	E80C7A



RGB 255 83 13

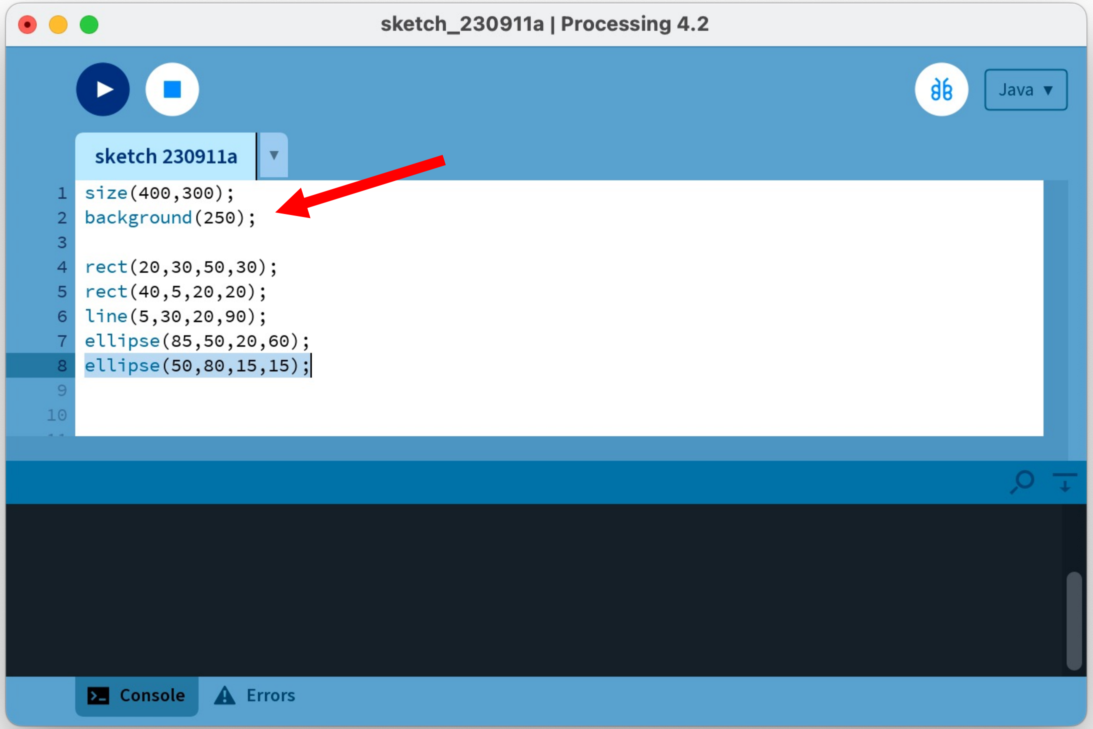
HEX FF530D

background() - syntax

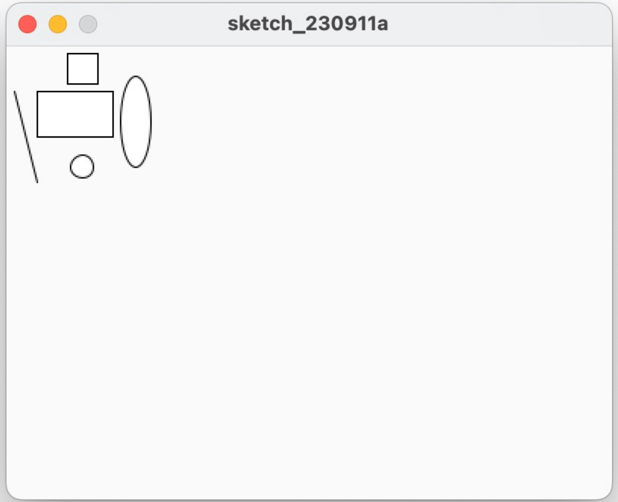
```
background(grayscale)
    grayscale = grayscale colour (a number between
                0 [black] and 255 [white] inclusive)
```

```
background(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)
```

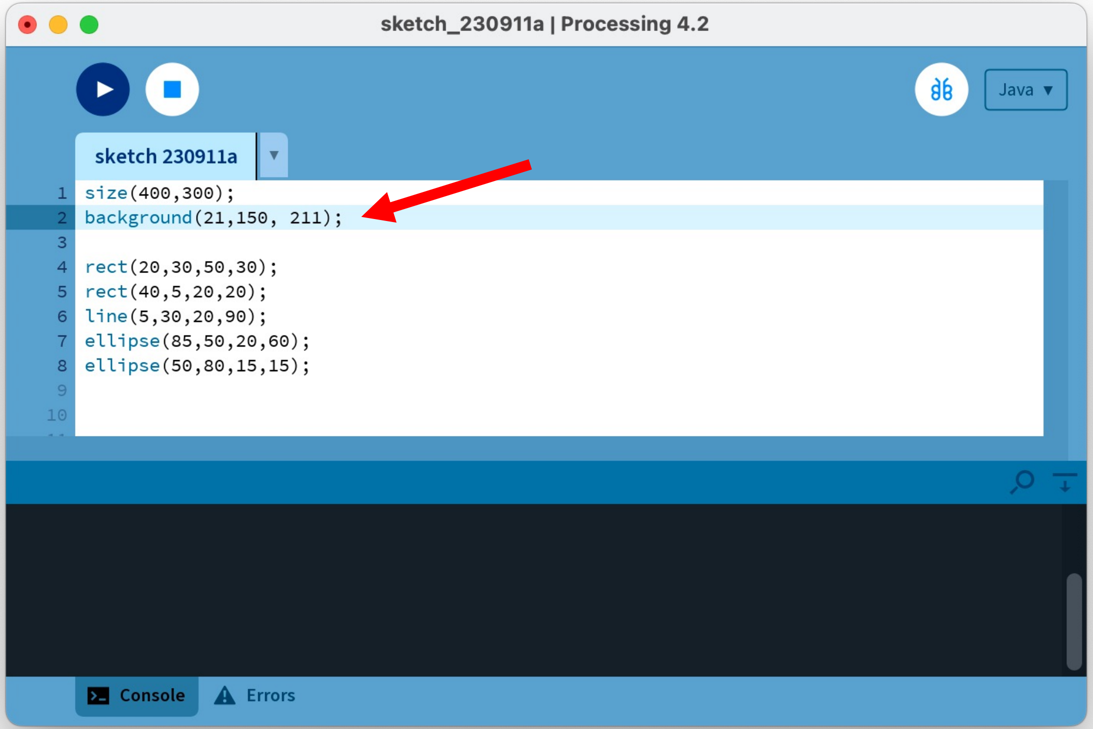
background() - grayscale



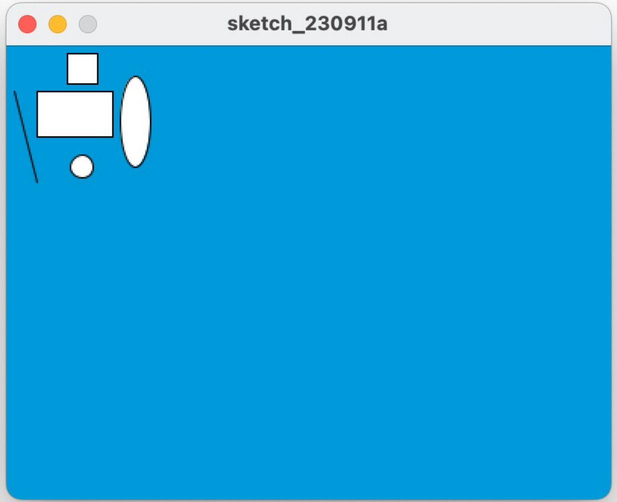
```
size(400,300);
background(250);

rect(20,30,50,30);
rect(40,5,20,20);
line(5,30,20,90);
ellipse(85,50,20,60);
ellipse(50,80,15,15);
```



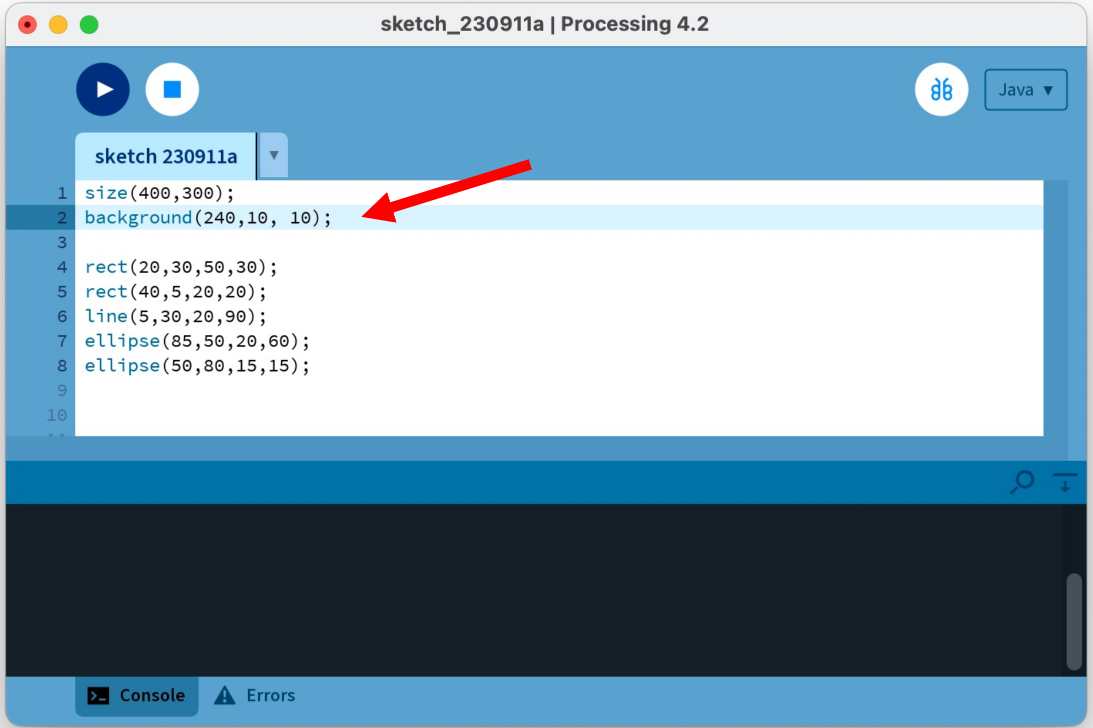
background() – R,G,B



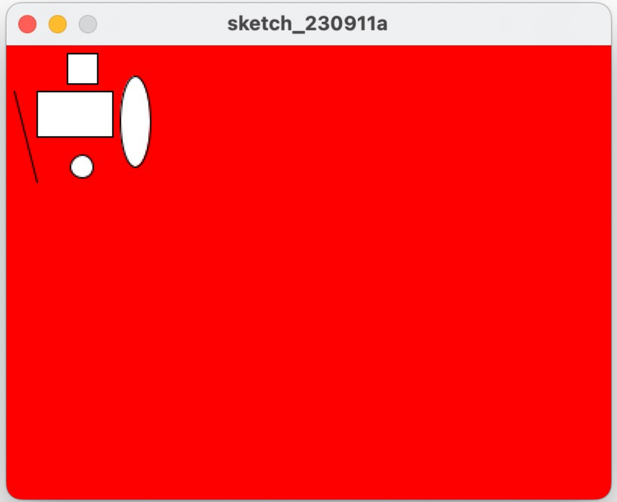
```
size(400,300);
background(21,150, 211);

rect(20,30,50,30);
rect(40,5,20,20);
line(5,30,20,90);
ellipse(85,50,20,60);
ellipse(50,80,15,15);
```



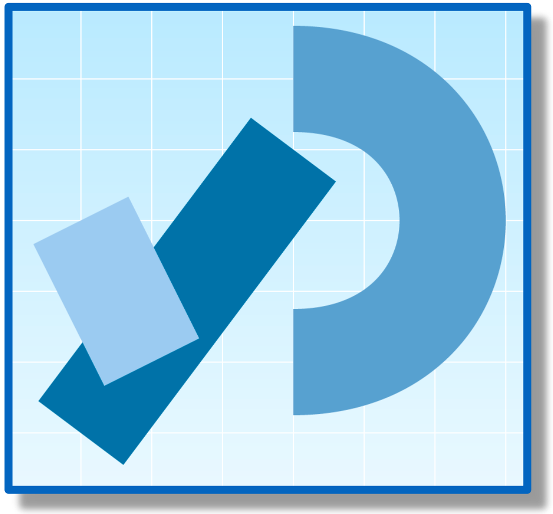
background() – R,G,B



```
size(400,300);
background(240,10, 10);

rect(20,30,50,30);
rect(40,5,20,20);
line(5,30,20,90);
ellipse(85,50,20,60);
ellipse(50,80,15,15);
```



Filling Shapes with Colour

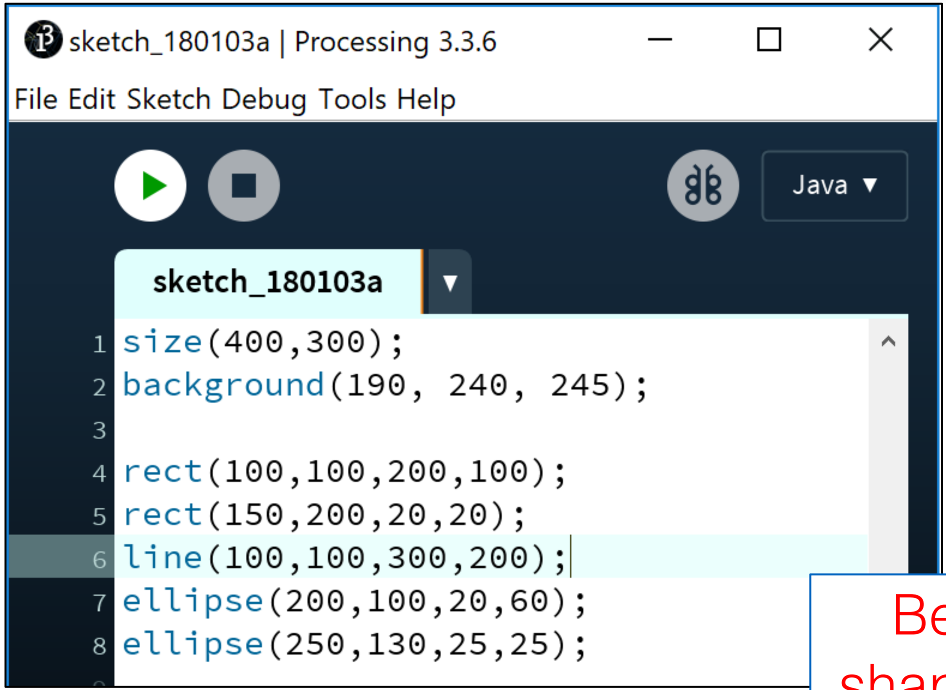


fill() - syntax

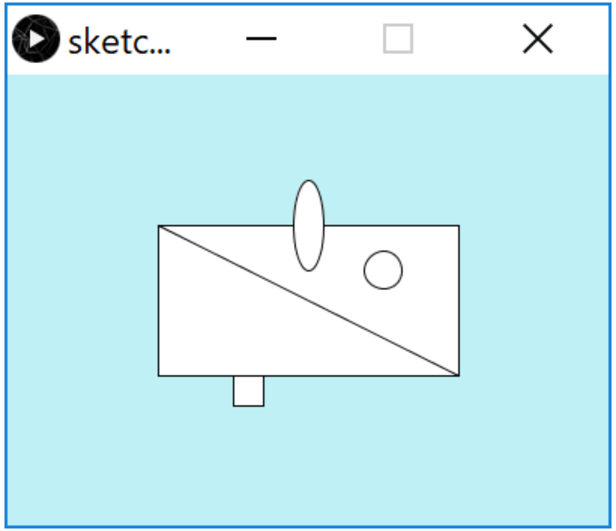
- ❑fills shapes with a chosen colour.
- ❑can use the RGB colours to select a colour.
- ❑all shapes drawn after the **fill** function is called, <u>will</u> be filled with the chosen colour.

```
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)
```

fill()

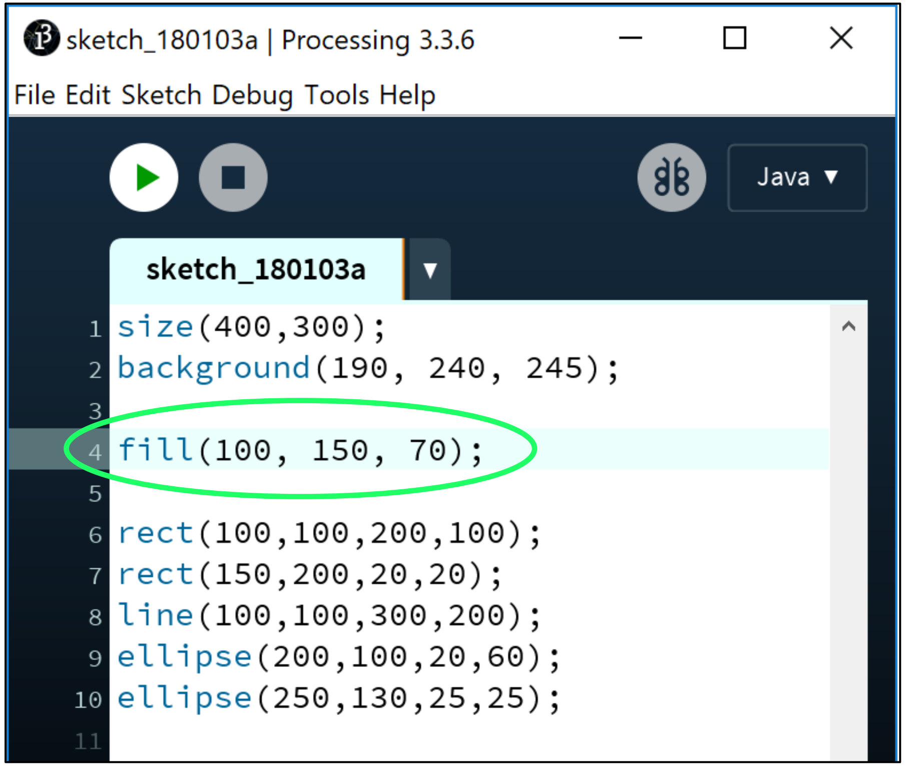

sketch_180103a | Processing 3.3.6

```
size(400,300);
background(190, 240, 245);

rect(100,100,200,100);
rect(150,200,20,20);
line(100,100,300,200);
ellipse(200,100,20,60);
ellipse(250,130,25,25);
```

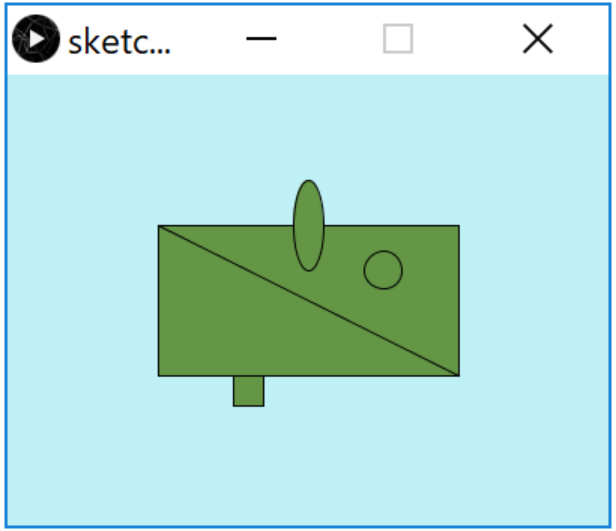


Before using fill(), all shapes fill with default of white.

fill()

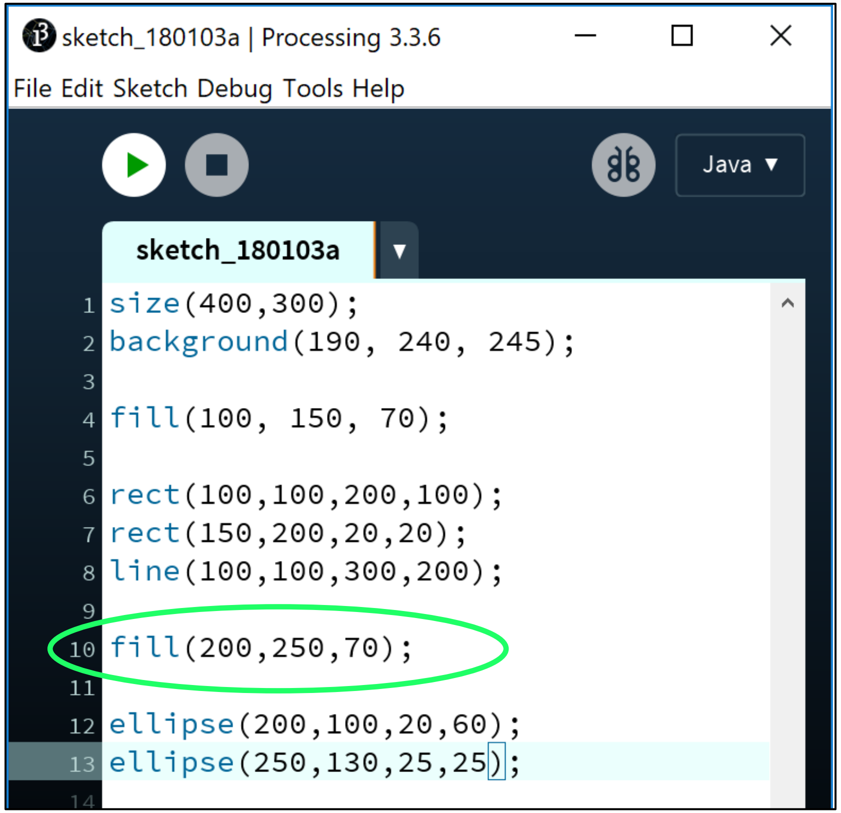

sketch_180103a | Processing 3.3.6
File Edit Sketch Debug Tools Help
Java
sketch_180103a

```
size(400,300);
background(190, 240, 245);

fill(100, 150, 70);

rect(100,100,200,100);
rect(150,200,20,20);
line(100,100,300,200);
ellipse(200,100,20,60);
ellipse(250,130,25,25);
```

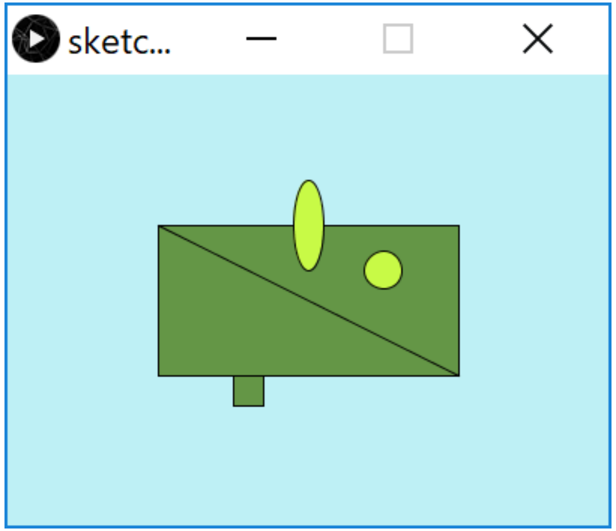


Now all shapes are filled with dark green.

fill()

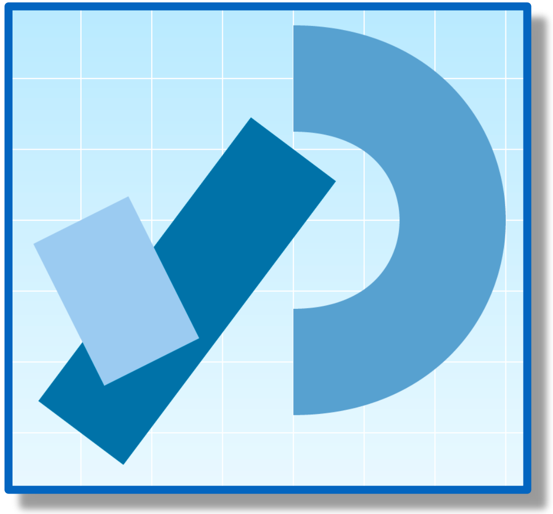

sketch_180103a | Processing 3.3.6
File Edit Sketch Debug Tools Help
Java
sketch_180103a

```
size(400,300);
background(190, 240, 245);

fill(100, 150, 70);

rect(100,100,200,100);
rect(150,200,20,20);
line(100,100,300,200);

fill(200,250,70);

ellipse(200,100,20,60);
ellipse(250,130,25,25);
```



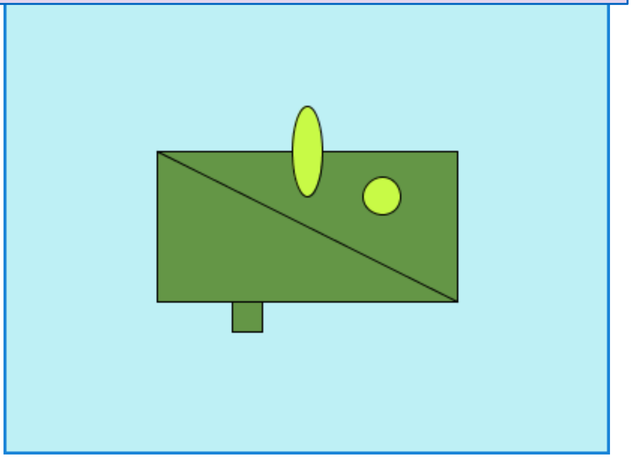
Rectangles filled with dark green.
Ellipses filled with light green;
order of statements matter!!!

Formatting the Shape Outline



Changing the outline (i.e. stroke)

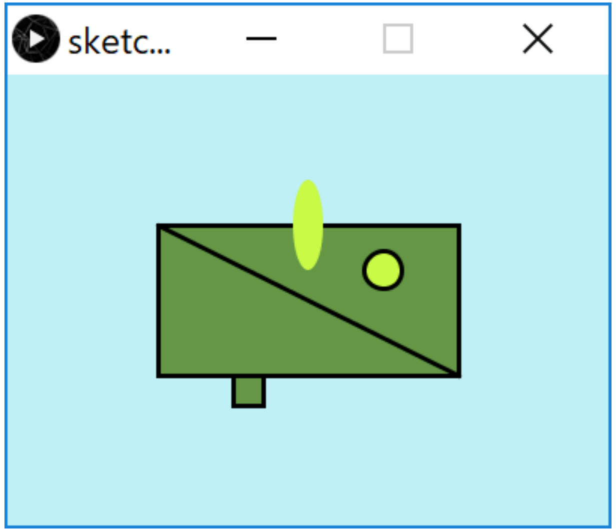
Before



After (changes):

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

We will now make those changes

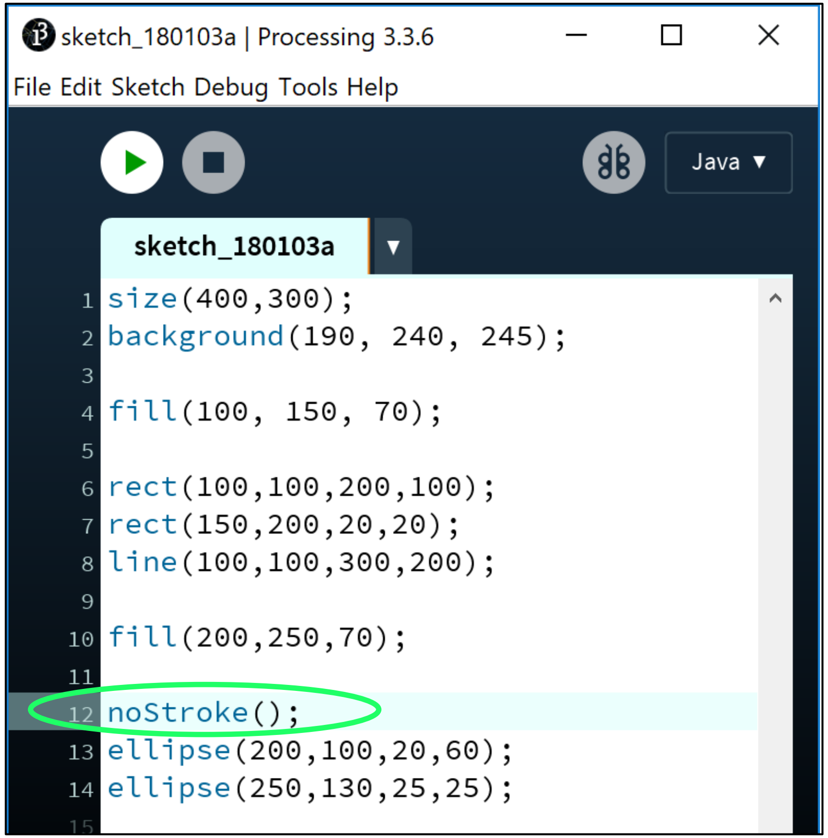


noStroke() - syntax

- ❑ 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.

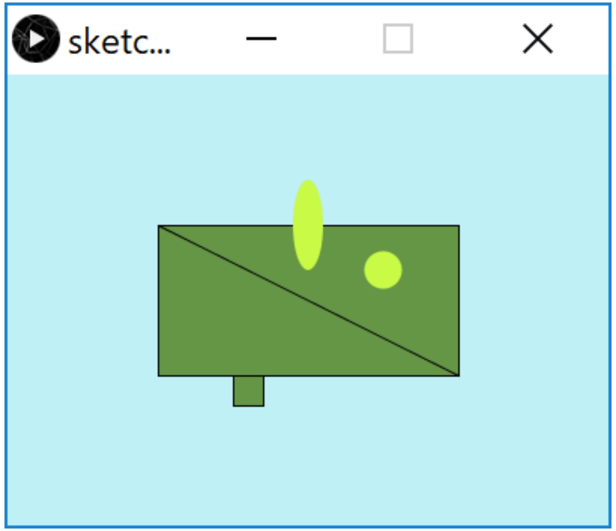
```
noStroke();
    //no parameters defined for this function.
```

noStroke()



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

```
size(400,300);
background(190, 240, 245);

fill(100, 150, 70);

rect(100,100,200,100);
rect(150,200,20,20);
line(100,100,300,200);

fill(200,250,70);

noStroke();
ellipse(200,100,20,60);
ellipse(250,130,25,25);
```



√ We have no border on the oval shape.

X But now our circle also has no border.

stroke() - syntax

- 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.

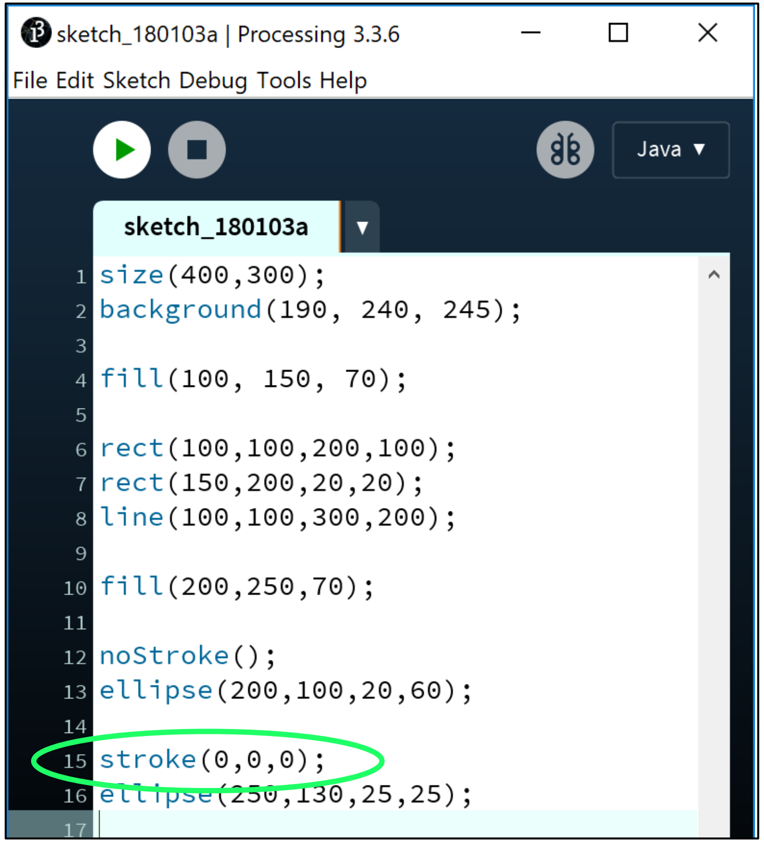
```
stroke (r, g, b)
```

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

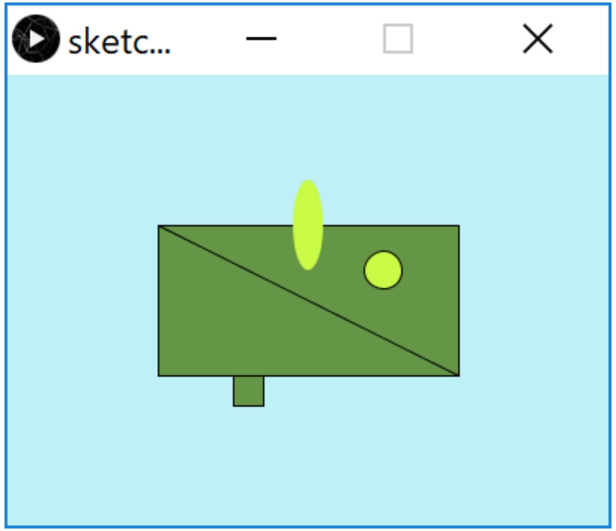
g = green colour (a number between 0 and 255 inclusive)

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

stroke()


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

```
size(400,300);
background(190, 240, 245);

fill(100, 150, 70);

rect(100,100,200,100);
rect(150,200,20,20);
line(100,100,300,200);

fill(200,250,70);

noStroke();
ellipse(200,100,20,60);

stroke(0,0,0);
ellipse(250,130,25,25);

```



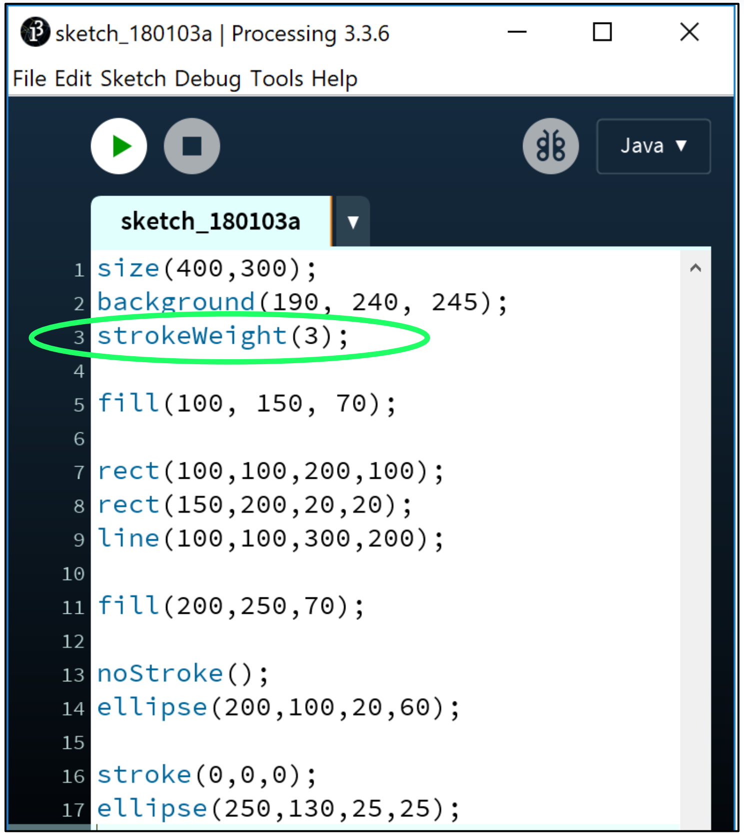
✓ Our circle now has a border.

strokeWeight() - syntax

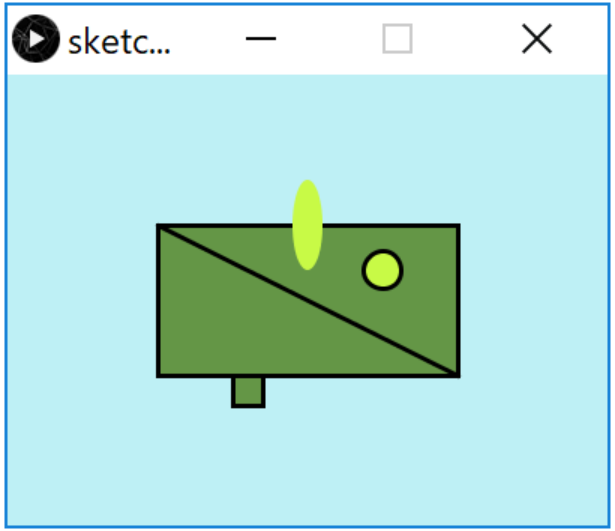
- ❑ The strokeWeight() function allows you to choose the thickness of a line/outline on shapes.
- ❑ The chosen thickness will apply to all lines/shapes that are drawn after the function is called.
- ❑ The thickness is specified in pixels.
- ❑ The default thickness is 1 pixel.

```
strokeWeight (pixels)
    pixels = thickness of the outline measures in pixels.
```

strokeWeight()


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

```
size(400,300);
background(190, 240, 245);
strokeWeight(3);

fill(100, 150, 70);

rect(100,100,200,100);
rect(150,200,20,20);
line(100,100,300,200);

fill(200,250,70);

noStroke();
ellipse(200,100,20,60);

stroke(0,0,0);
ellipse(250,130,25,25);
```



✓ Our outline is now heavier.

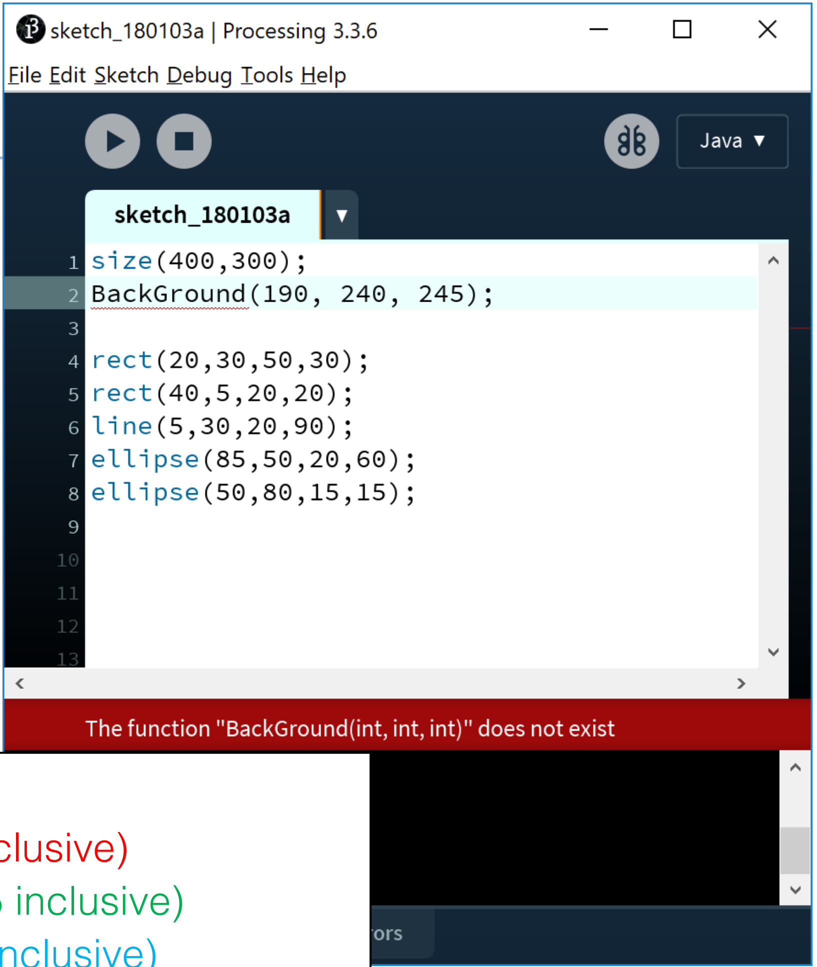
Syntax and Logic Errors

Syntax and Syntax Errors

- ❑ You will have seen the term Syntax mentioned above.
- ❑ Syntax are the rules you must follow when writing well-formed statements in a programming language.
- ❑ When you don’t follow the rules, Processing will not run your code; instead you will get an error.
- ❑ Some syntax error examples are on the upcoming slides.

Syntax Errors

The spelling of the background function must be identical to the spelling below (case sensitive!).


sketch_180103a | Processing 3.3.6
File Edit Sketch Debug Tools Help
Java
sketch_180103a
ors

```
size(400,300);
BackGround(190, 240, 245);

rect(20,30,50,30);
rect(40,5,20,20);
line(5,30,20,90);
ellipse(85,50,20,60);
ellipse(50,80,15,15);
```

The function "BackGround(int, int, int)" does not exist

background(r, g, b)

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

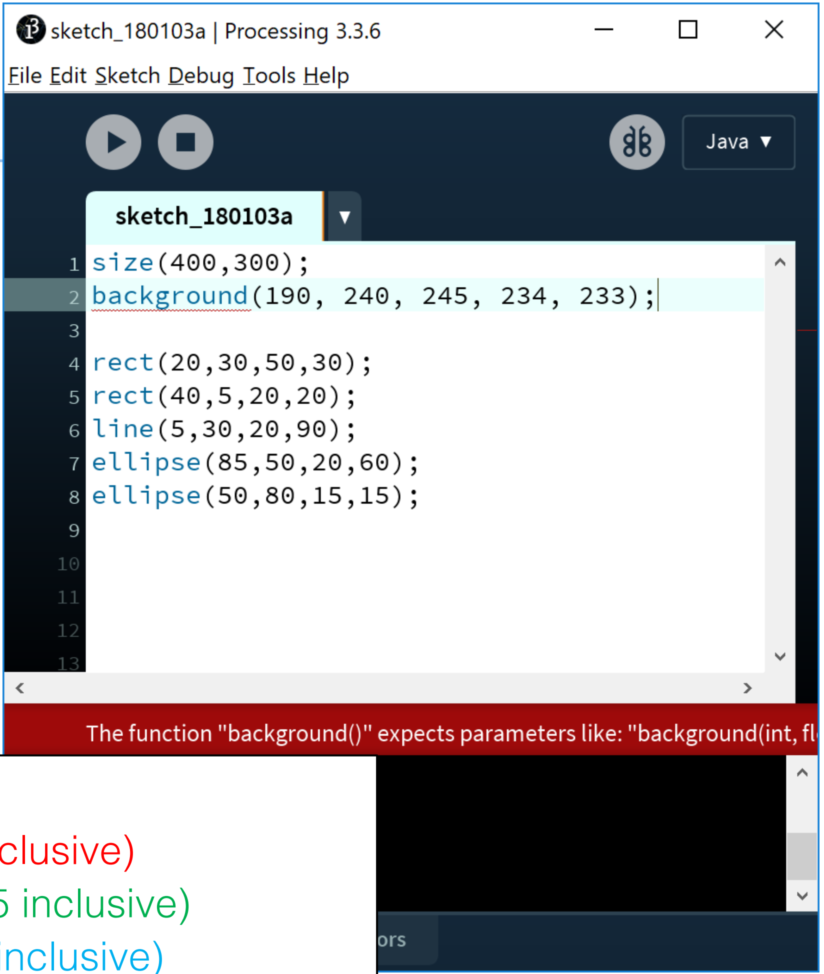
g = green colour (a number between 0 and 255 inclusive)

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

Syntax Errors

The background function has too many arguments passed to it i.e.

- ❏ *RGB version is defined with 3 parameters.*
- ❏ *Grayscale version is defined with 1 parameter.*


sketch_180103a | Processing 3.3.6
File Edit Sketch Debug Tools Help
Java
sketch_180103a
ors

```
size(400,300);
background(190, 240, 245, 234, 233);

rect(20,30,50,30);
rect(40,5,20,20);
line(5,30,20,90);
ellipse(85,50,20,60);
ellipse(50,80,15,15);
```

The function "background()" expects parameters like: "background(int, fl

background(r, g, b)

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

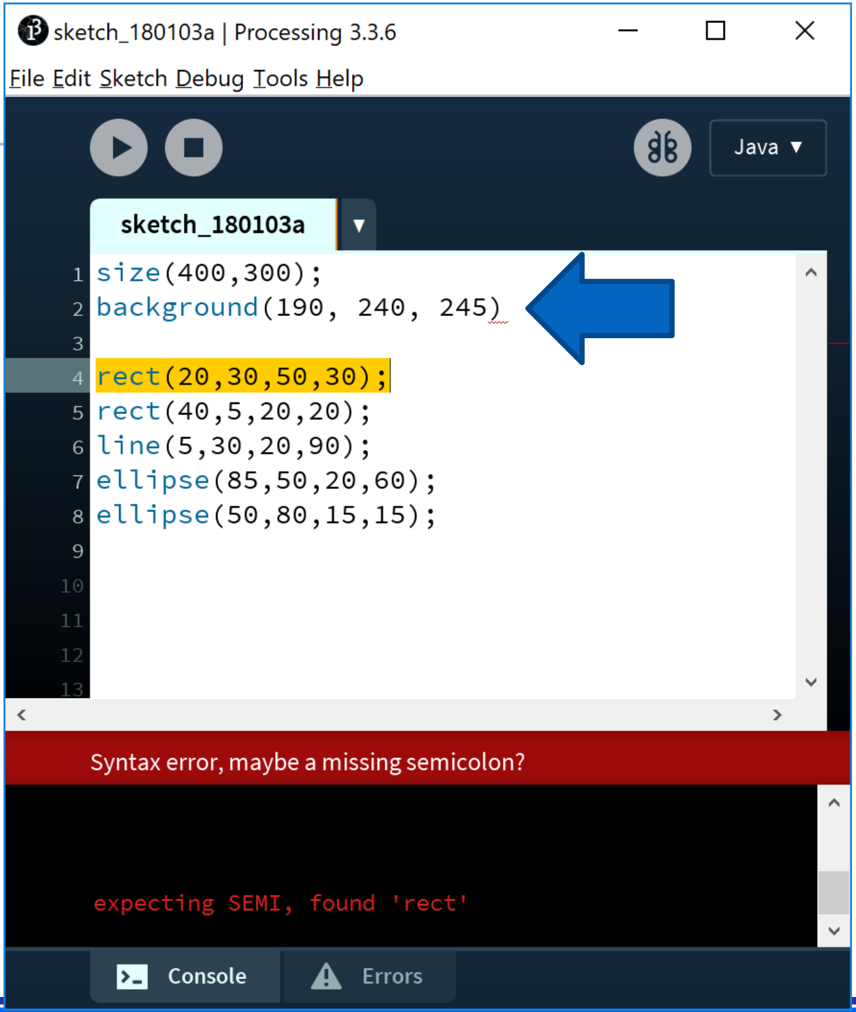
g = green colour (a number between 0 and 255 inclusive)

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

Syntax Errors

The semi-colon (;) is missing at the end of the statement.

Java needs a statement terminator for each line!

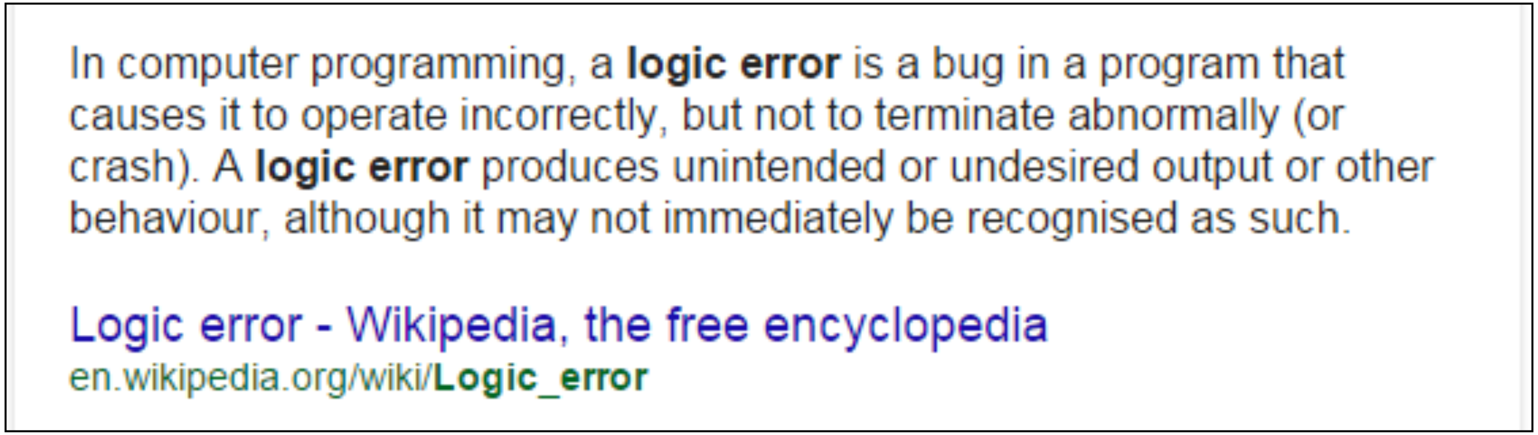

sketch_180103a | Processing 3.3.6
File Edit Sketch Debug Tools Help
Java
sketch_180103a
Console
Errors

```
size(400,300);
background(190, 240, 245)

rect(20,30,50,30);
rect(40,5,20,20);
line(5,30,20,90);
ellipse(85,50,20,60);
ellipse(50,80,15,15);
```

Syntax error, maybe a missing semicolon?

```
expecting SEMI, found 'rect'
```

Logic Errors

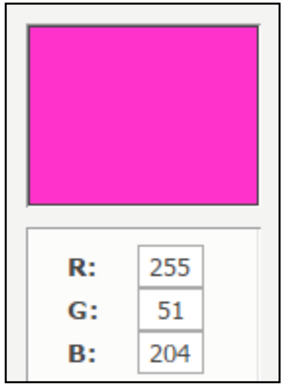


In computer programming, a **logic error** is a bug in a program that causes it to operate incorrectly, but not to terminate abnormally (or crash). A **logic error** produces unintended or undesired output or other behaviour, although it may not immediately be recognised as such.

Logic error - Wikipedia, the free encyclopedia
en.wikipedia.org/wiki/**Logic_error**

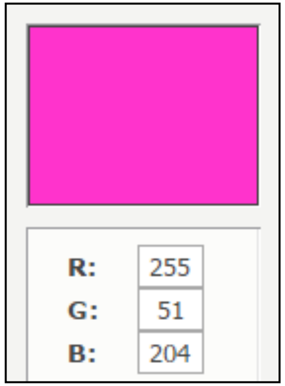
Logic Errors

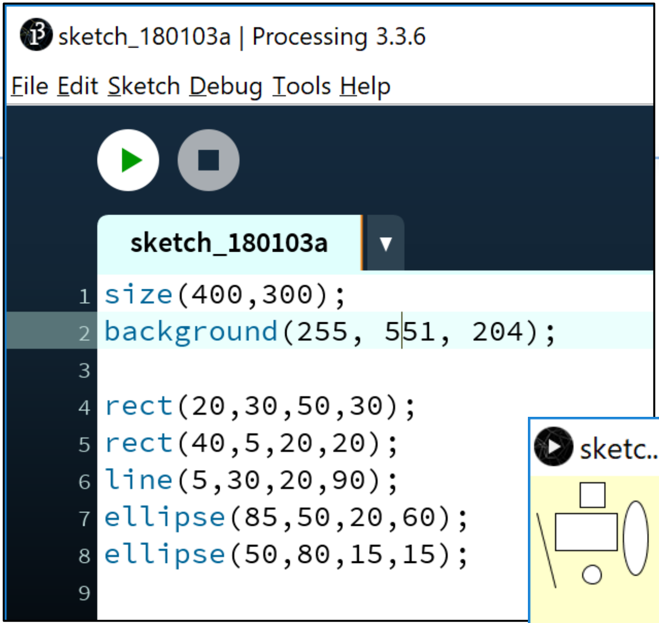
Say we wanted a pink background for our display window.


R: 255
G: 51
B: 204

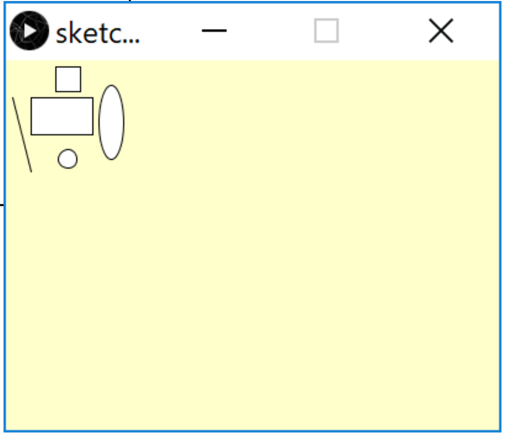
Logic Errors

Say we wanted a pink background for our display window.




sketch_180103a | Processing 3.3.6

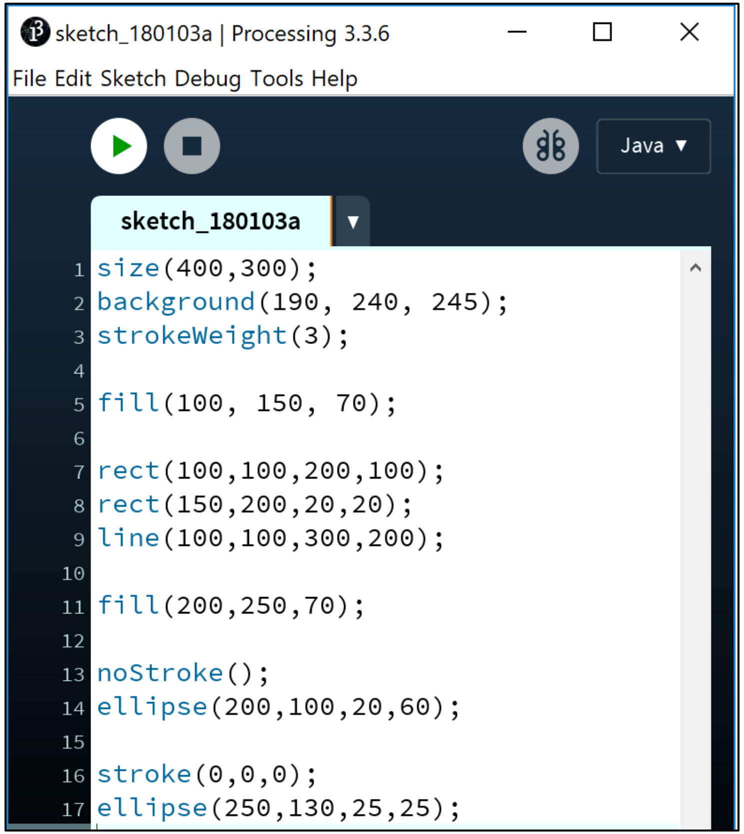
```
size(400,300);
background(255, 551, 204);

rect(20,30,50,30);
rect(40,5,20,20);
line(5,30,20,90);
ellipse(85,50,20,60);
ellipse(50,80,15,15);
```



- However, we incorrectly enter the **G** colour as 551 instead of 51.
- We now have a yellowish background.
- This is an example of a simple logic error.

Commenting your Code

Code so far…


sketch_180103a | Processing 3.3.6
File Edit Sketch Debug Tools Help
Java ▾
sketch_180103a

```
size(400,300);
background(190, 240, 245);
strokeWeight(3);

fill(100, 150, 70);

rect(100,100,200,100);
rect(150,200,20,20);
line(100,100,300,200);

fill(200,250,70);

noStroke();
ellipse(200,100,20,60);

stroke(0,0,0);
ellipse(250,130,25,25);
```

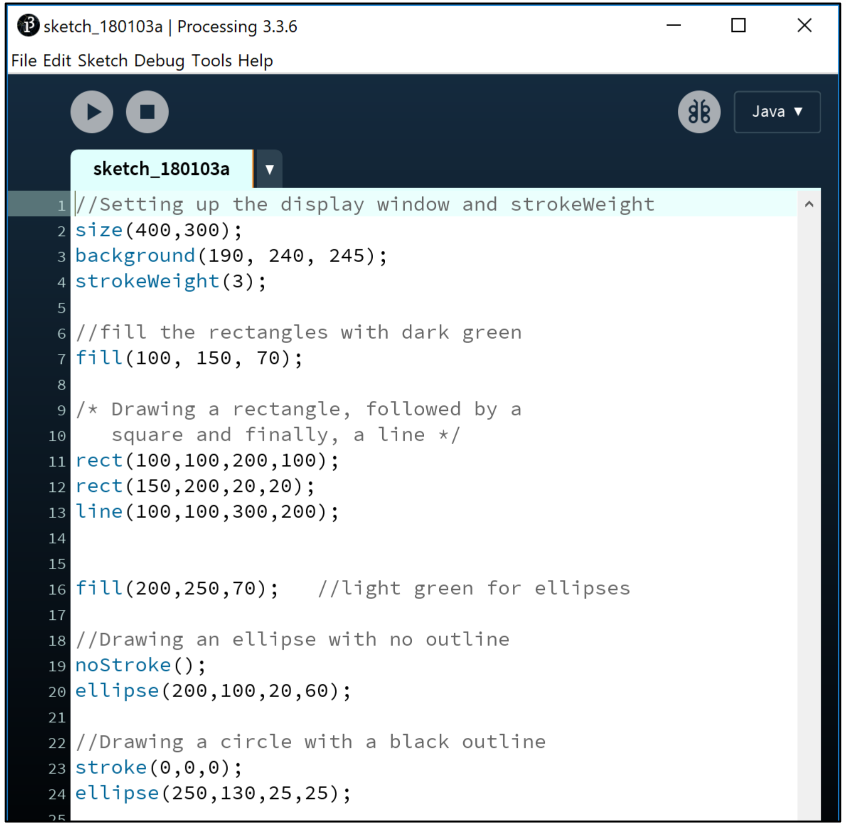
Can you tell, from looking at the code, what RGB colours you have chosen?

- ❑ 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


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

```
//Setting up the display window and strokeWeight
size(400,300);
background(190, 240, 245);
strokeWeight(3);

//fill the rectangles with dark green
fill(100, 150, 70);

/* Drawing a rectangle, followed by a
   square and finally, a line */
rect(100,100,200,100);
rect(150,200,20,20);
line(100,100,300,200);


fill(200,250,70);    //light green for ellipses

//Drawing an ellipse with no outline
noStroke();
ellipse(200,100,20,60);

//Drawing a circle with a black outline
stroke(0,0,0);
ellipse(250,130,25,25);
```

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?

Thanks.