

Programming Fundamentals

Starting to Code in Processing

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

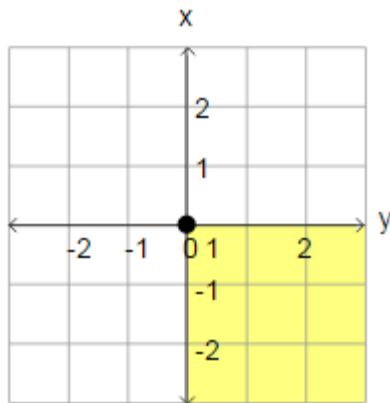


Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

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

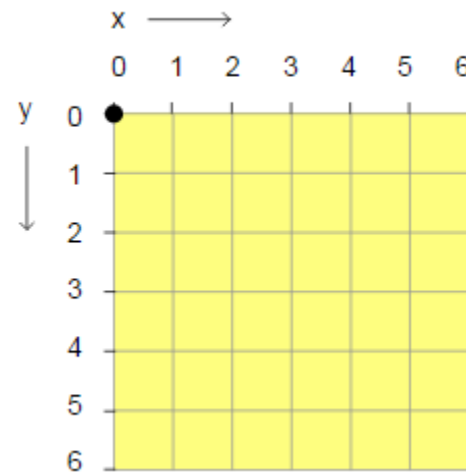
Coordinate System in Computing

In Geometry,
we use this type of
coordinate system:



point (0,0) is in the
centre.

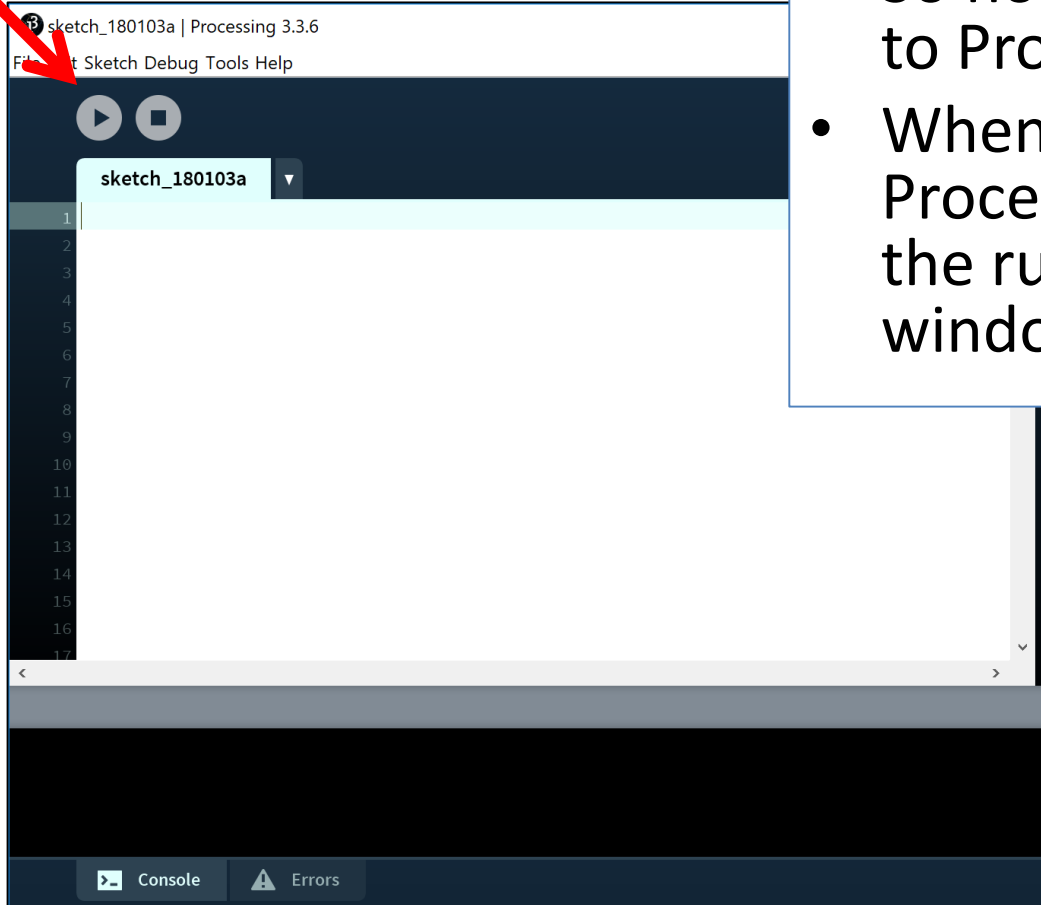
In Computing, we use this type of
coordinate system to represent the
screen:



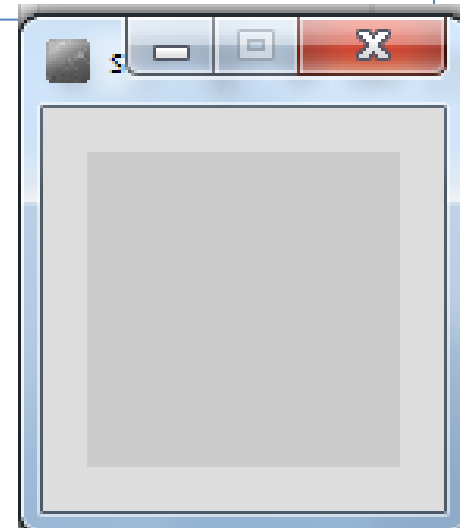
point (0,0) is in the top left hand
corner. Each number is a pixel.

Coordinate System in Computing

**Run
button**



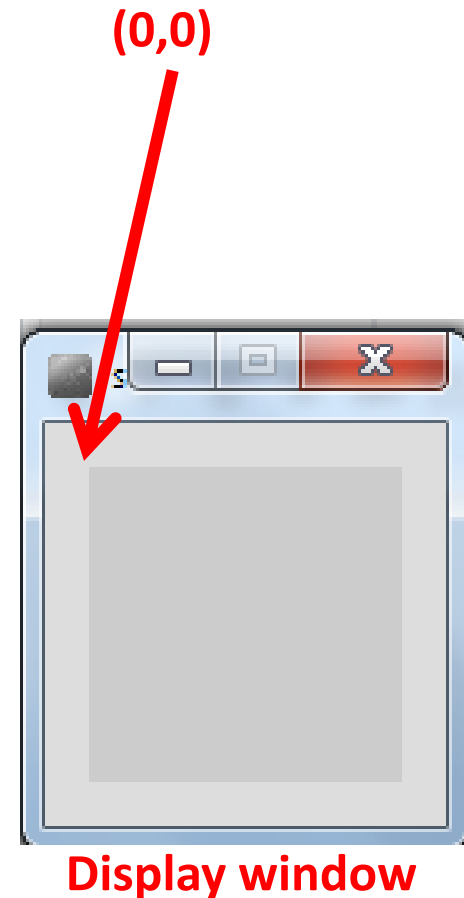
- So how does this relate to Processing?
- When you open Processing and click on the run button, a display window pops up.



Display window

Coordinate System in Computing

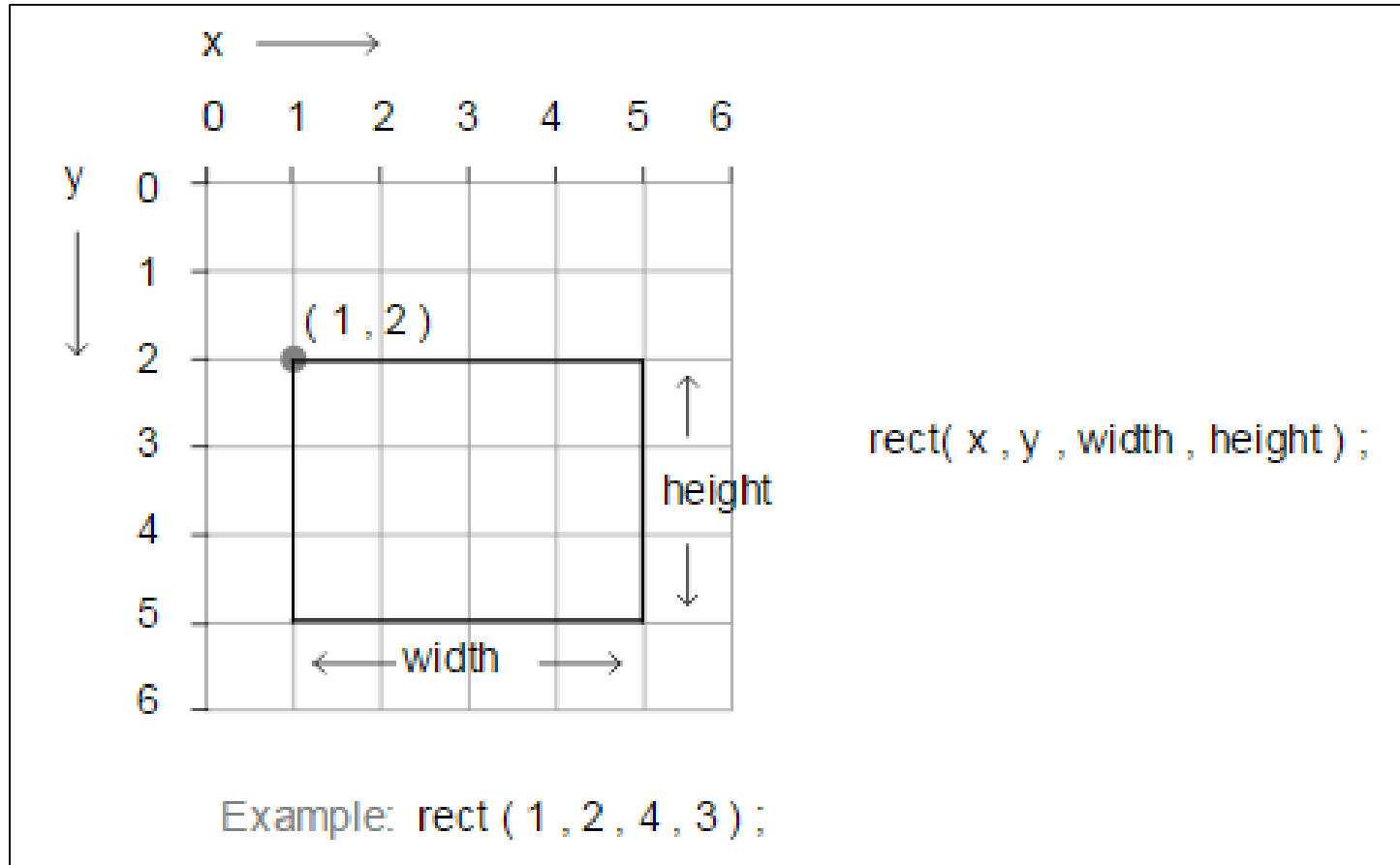
- The display window is where your code is run/ displayed.
- It follows the rules of the Computing coordinate system i.e. the top left hand corner is $(0,0)$.
- A point $(10,20)$ is 10 pixels to the right of $(0,0)$ and 20 pixels below $(0,0)$.



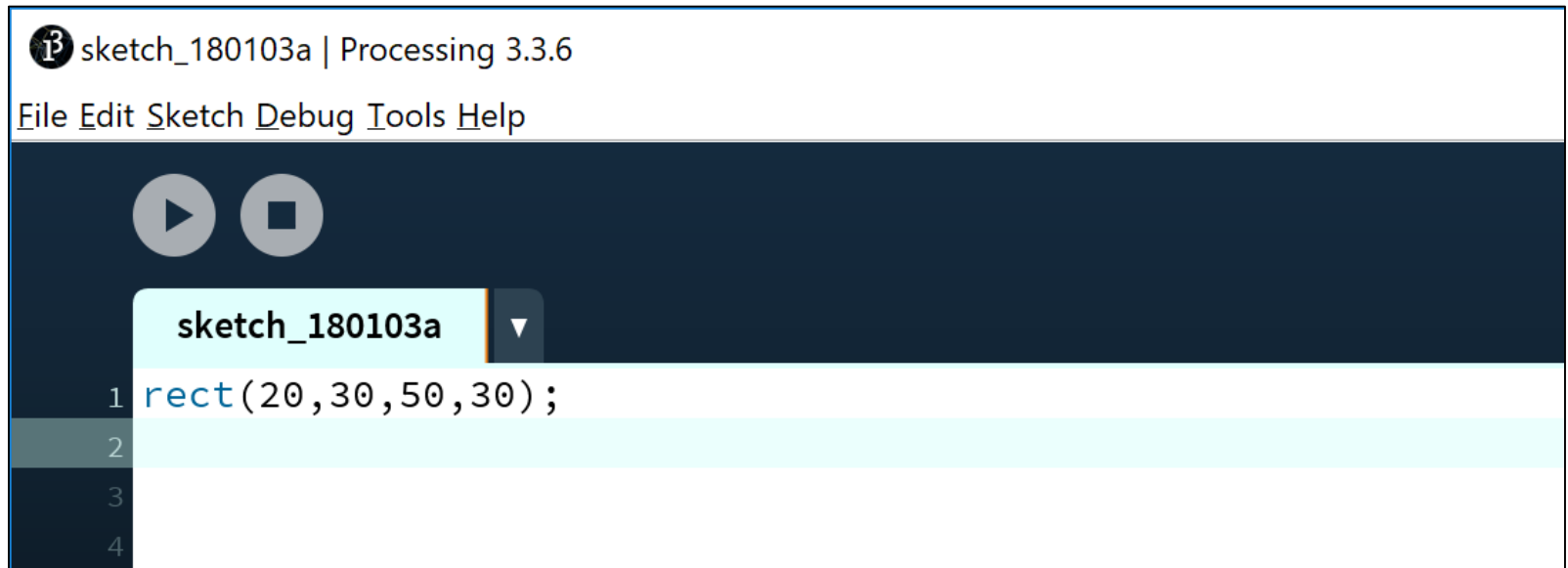
Functions in Processing

- Processing comes with several pre-written functions that we can use.
- A function comprises a set of instructions that performs some task.
- When you call the function, it performs the task.
- We will now look at functions that draw the following shapes:
 - Rectangle, square, line, oval and circle.

rect()



rect() – drawing a rectangle

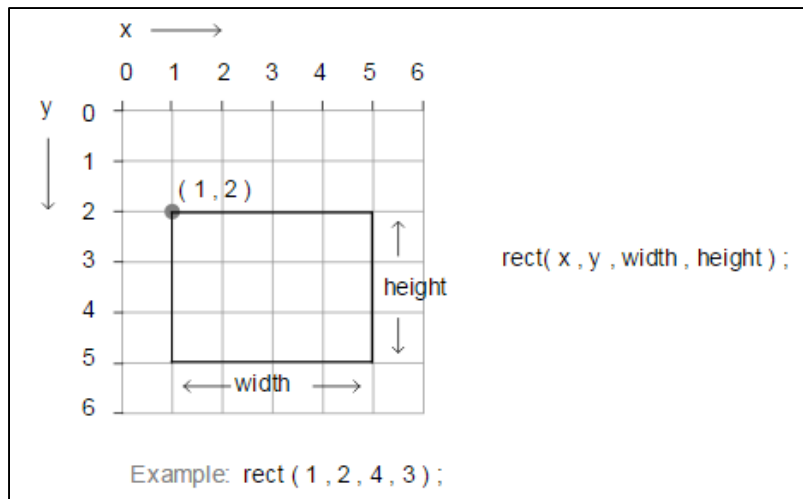


sketch_180103a | Processing 3.3.6

File Edit Sketch Debug Tools Help

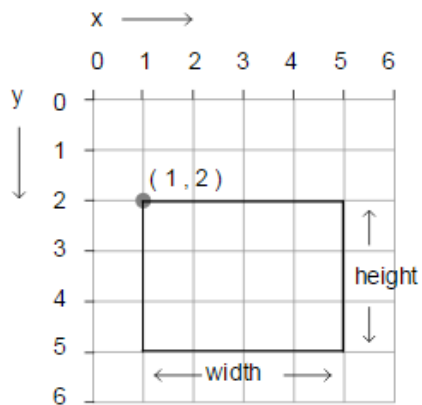
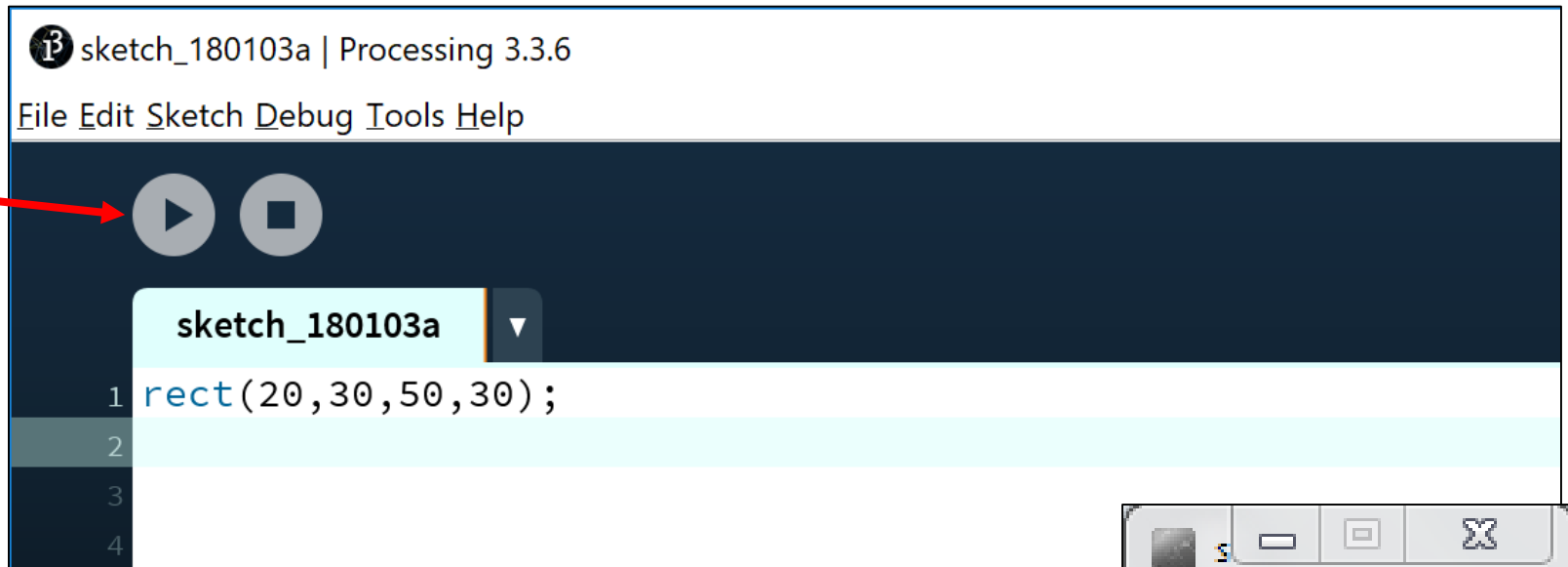
sketch_180103a

```
1 rect(20, 30, 50, 30);  
2  
3  
4
```



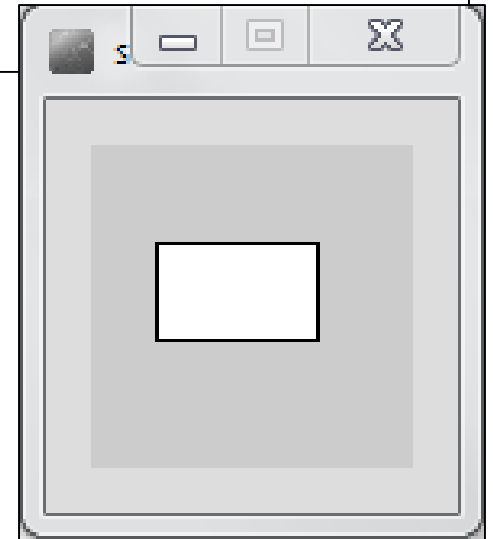
rect() – drawing a rectangle

Click
to
Run

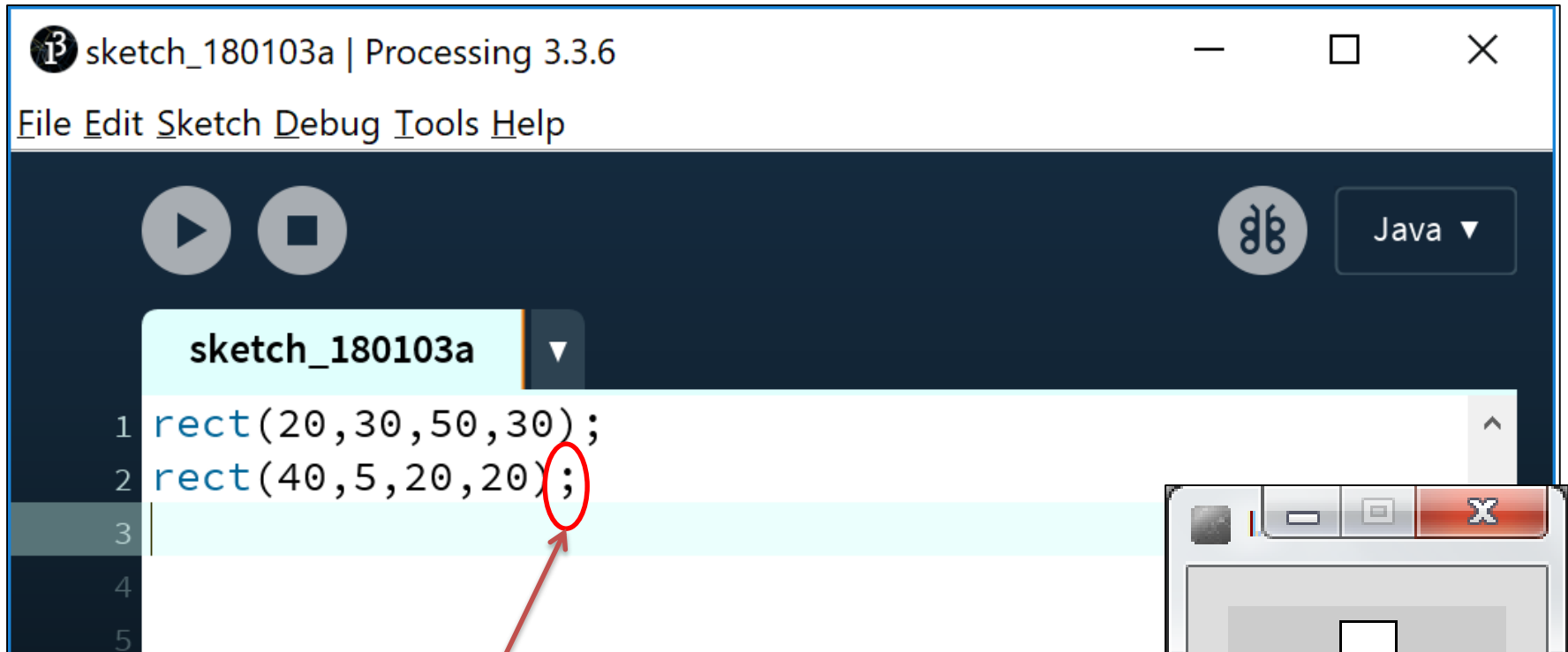


`rect(x, y, width, height);`

Example: `rect(1, 2, 4, 3);`

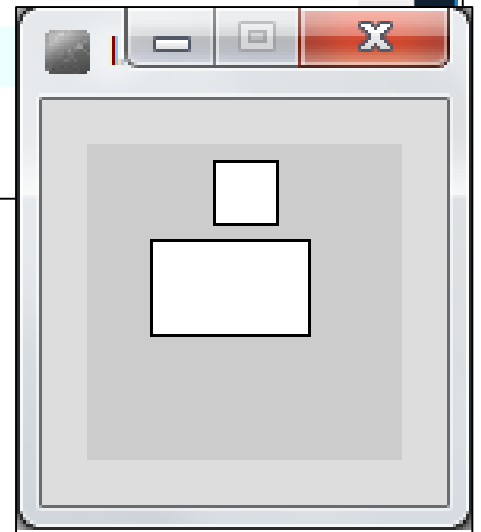


rect() – drawing a square

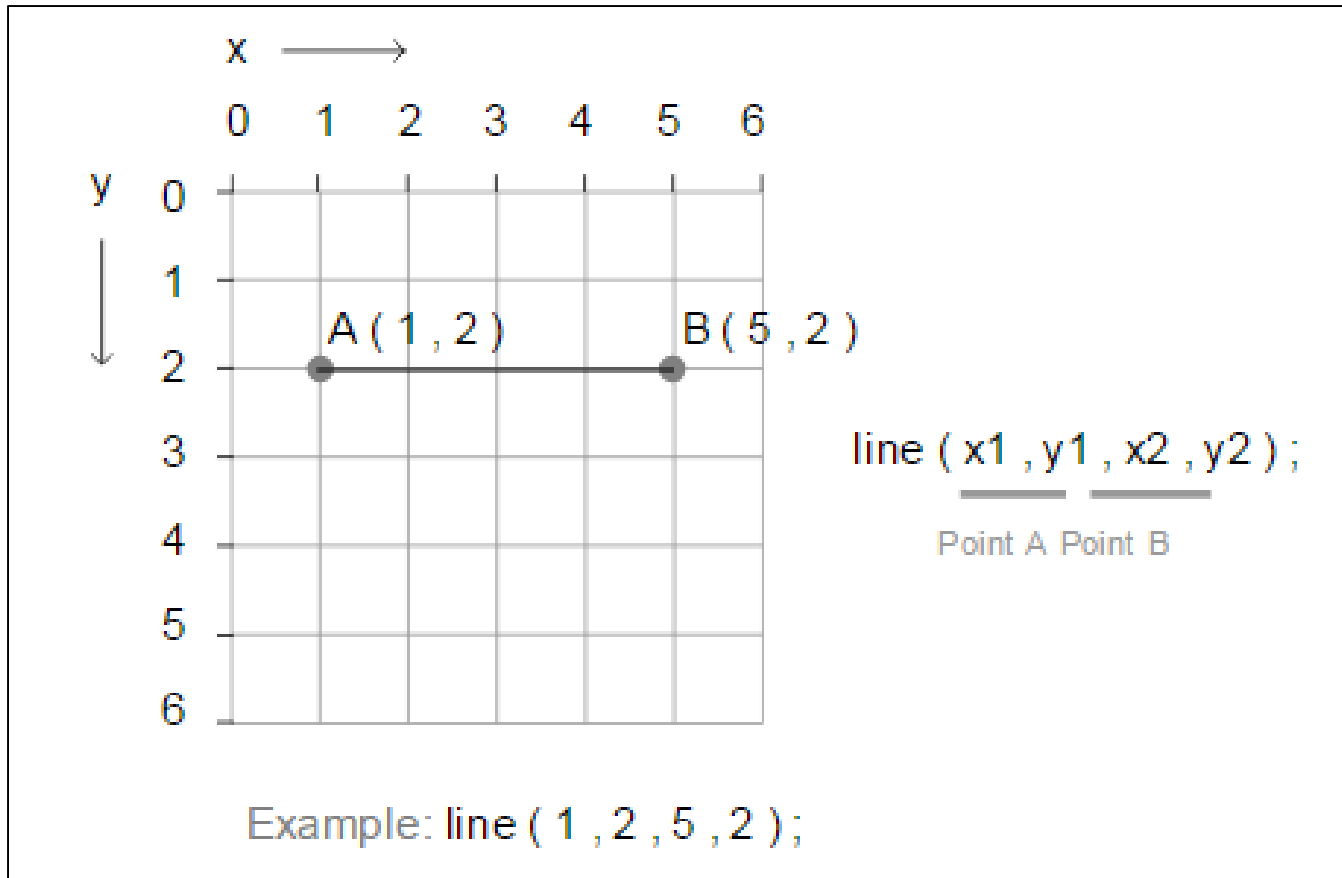


```
sketch_180103a
1 rect(20, 30, 50, 30);
2 rect(40, 5, 20, 20);
3
4
5
```

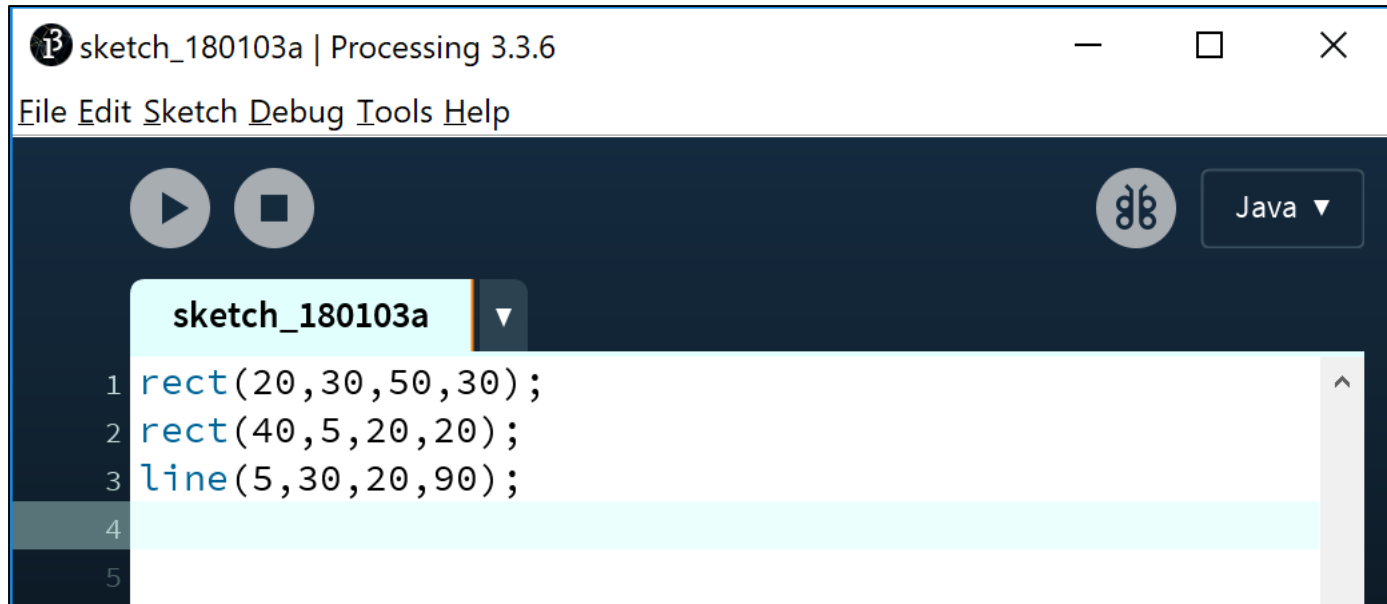
Note how each line of code has a semi-colon (;) at the end of it. This is called a **statement terminator** and must be included.



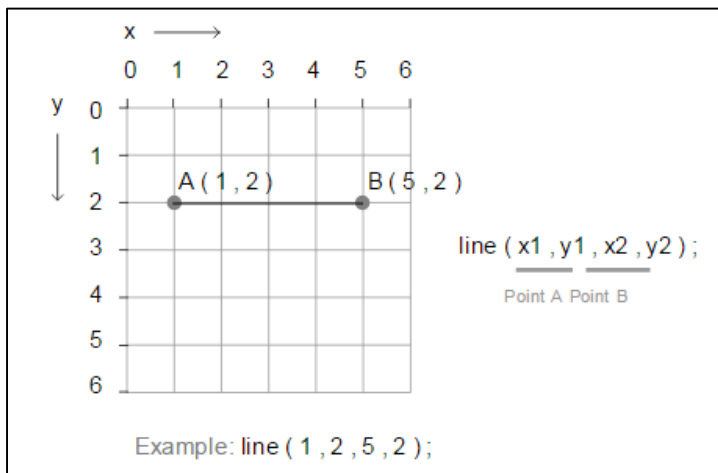
line()



line () – drawing a line

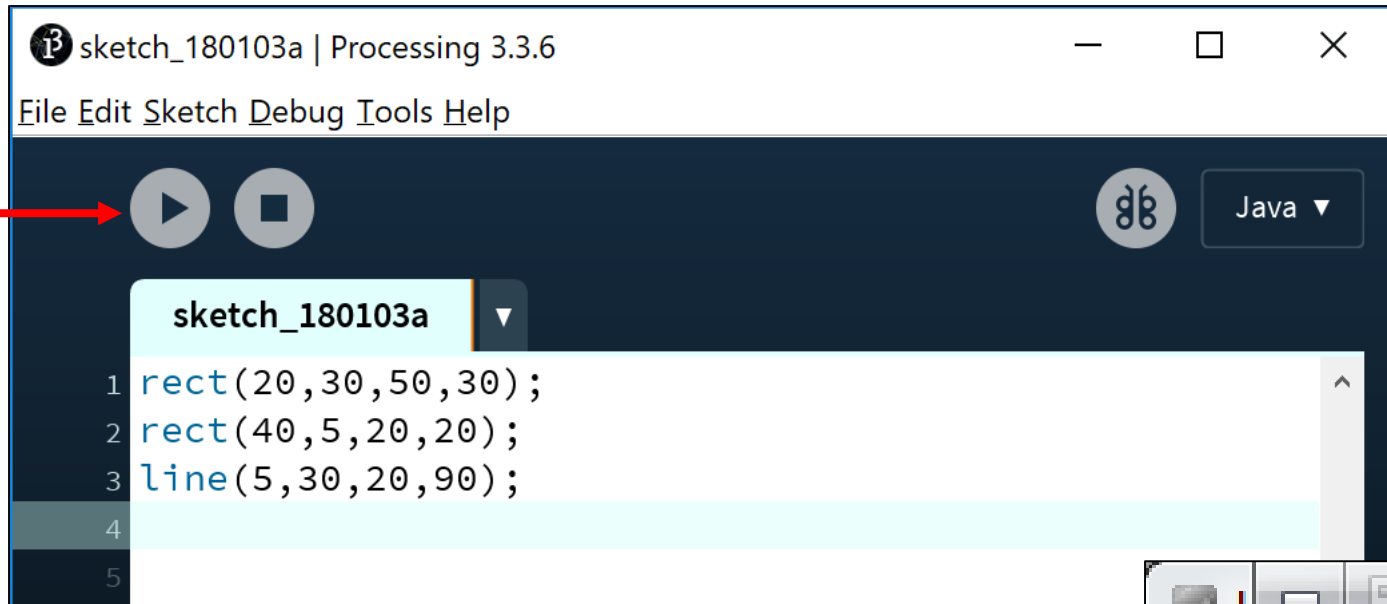


```
sketch_180103a | Processing 3.3.6
File Edit Sketch Debug Tools Help
sketch_180103a
1 rect(20,30,50,30);
2 rect(40,5,20,20);
3 line(5,30,20,90);
4
5
```

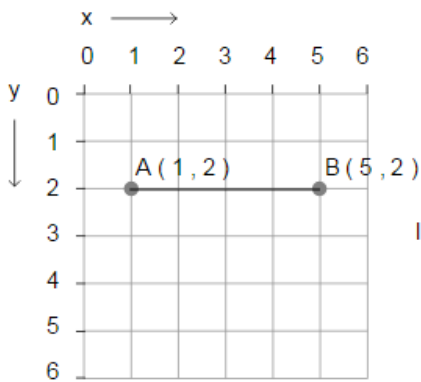


line () – drawing a line

Click
to
Run

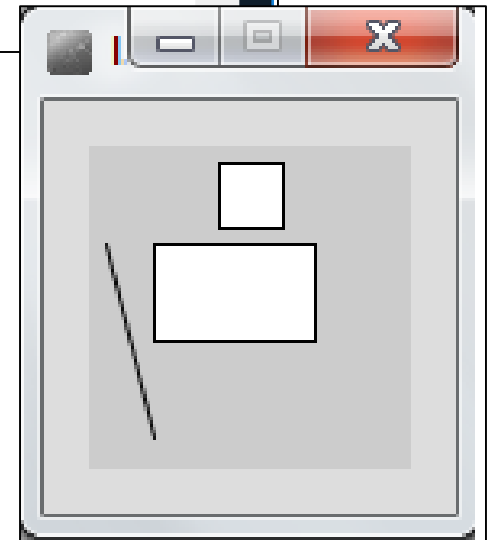


```
sketch_180103a | Processing 3.3.6
File Edit Sketch Debug Tools Help
sketch_180103a
1 rect(20,30,50,30);
2 rect(40,5,20,20);
3 line(5,30,20,90);
4
5
```

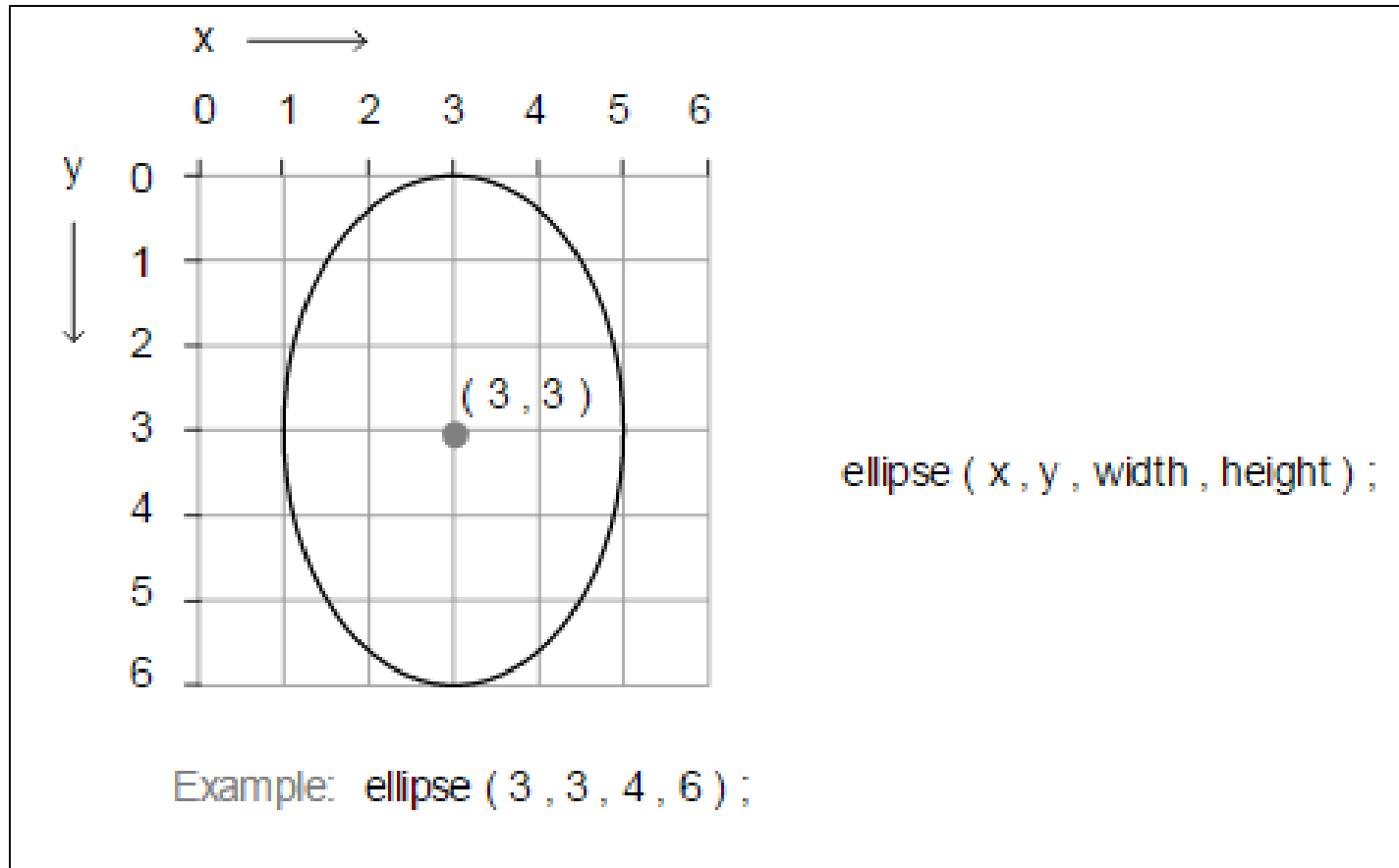


`line (x1 , y1 , x2 , y2);`
Point A Point B

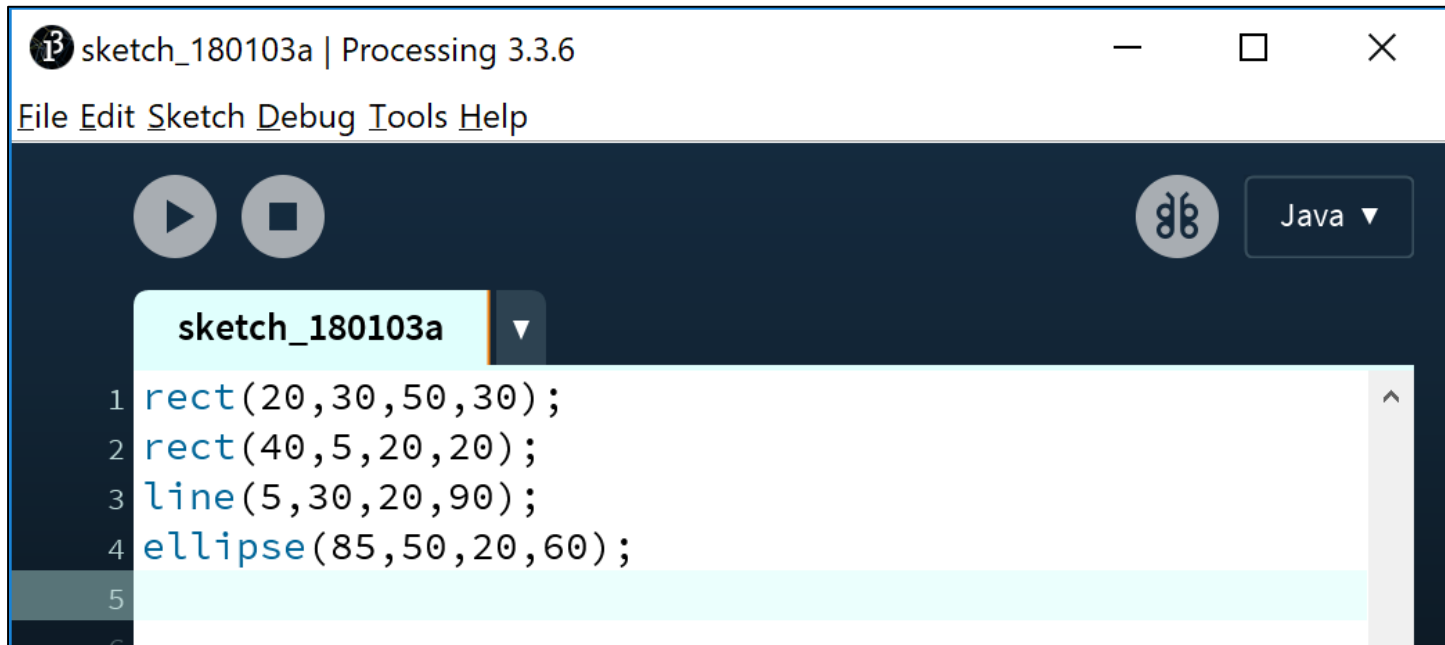
Example: `line (1 , 2 , 5 , 2);`



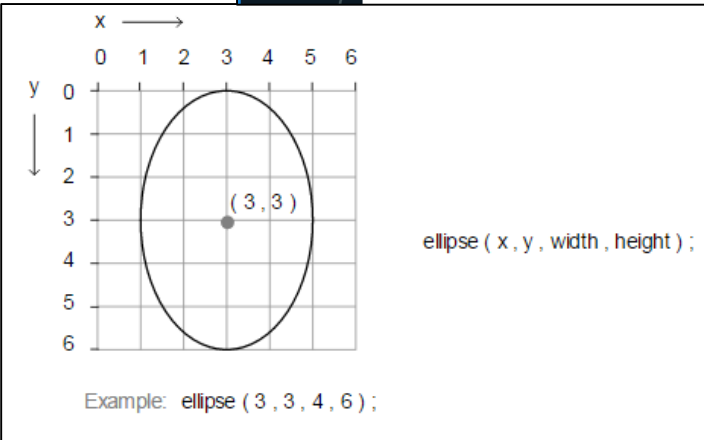
ellipse()



ellipse() – drawing an oval

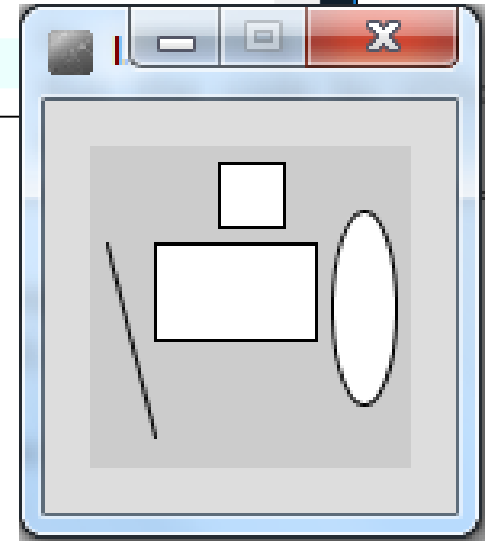
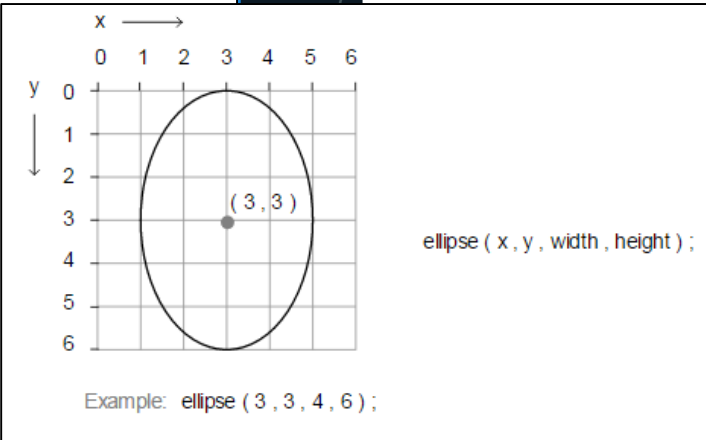
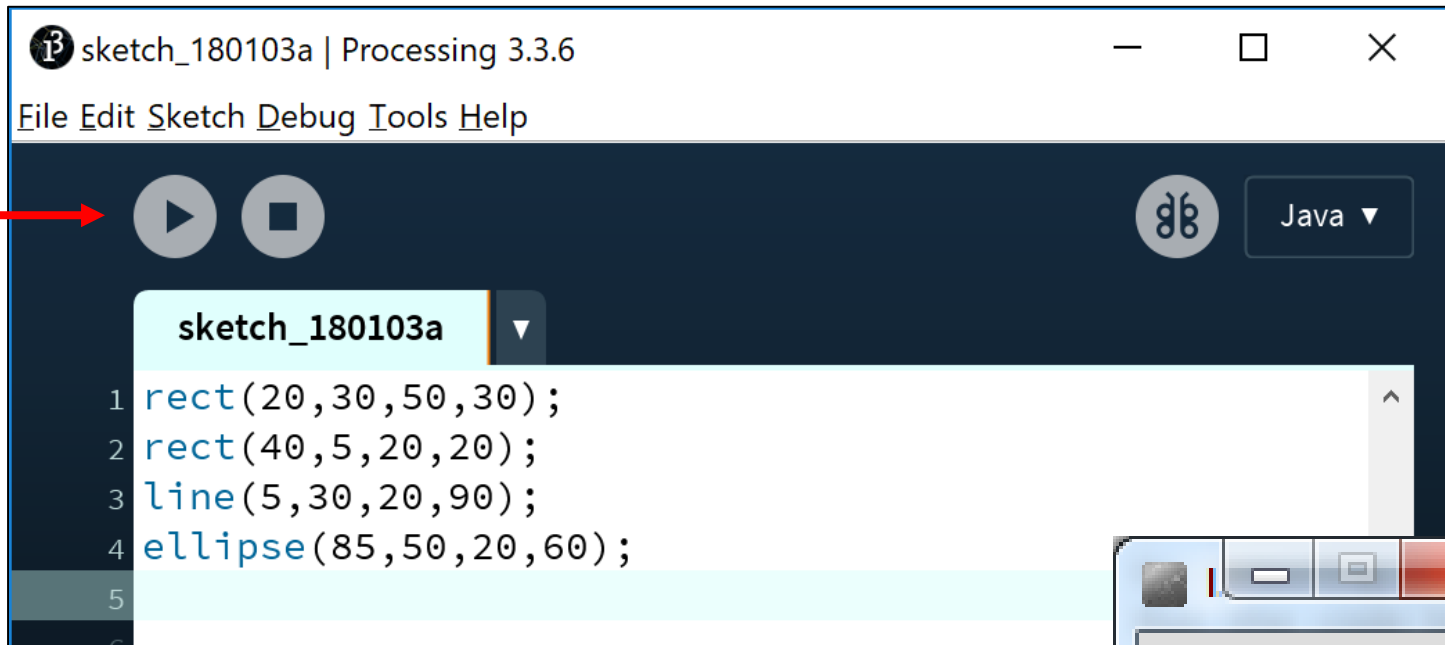


```
sketch_180103a | Processing 3.3.6
File Edit Sketch Debug Tools Help
Java
sketch_180103a
1 rect(20,30,50,30);
2 rect(40,5,20,20);
3 line(5,30,20,90);
4 ellipse(85,50,20,60);
5
```

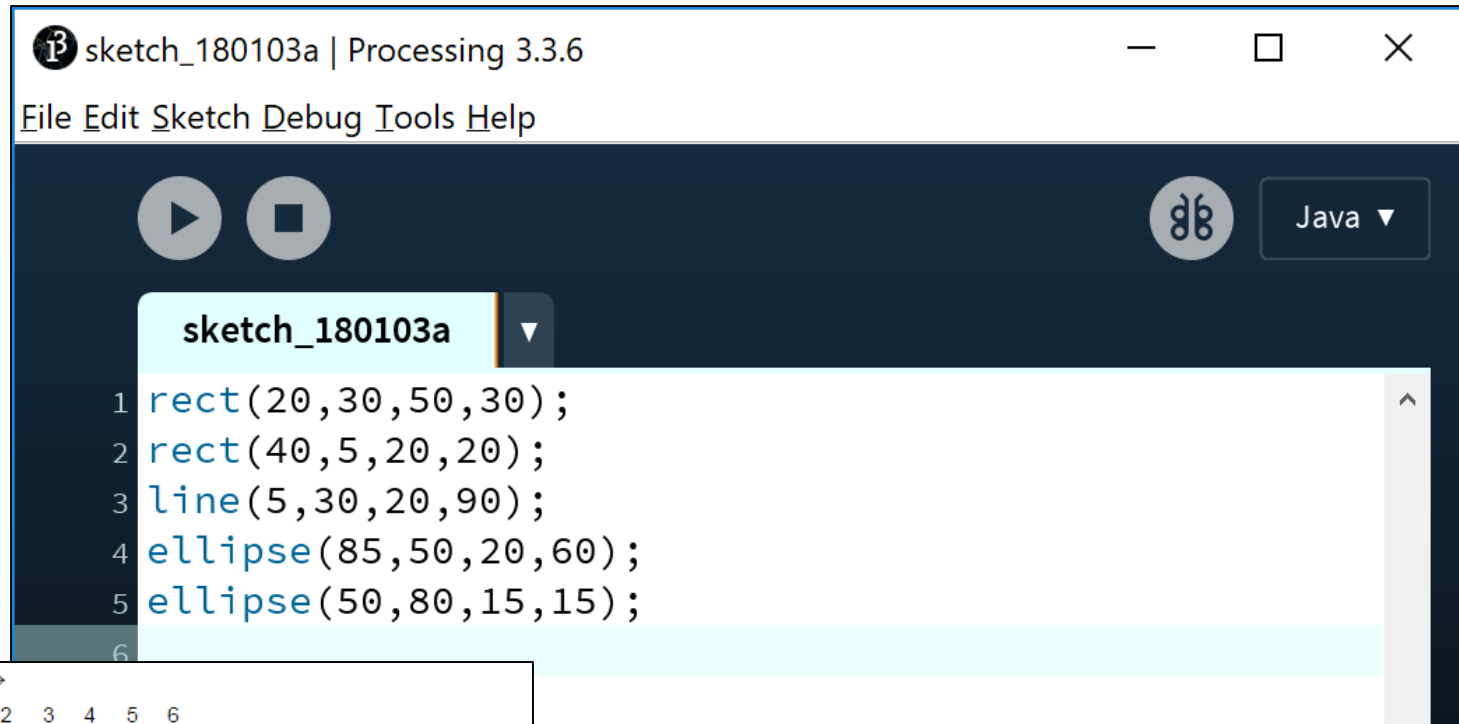


ellipse() – drawing an oval

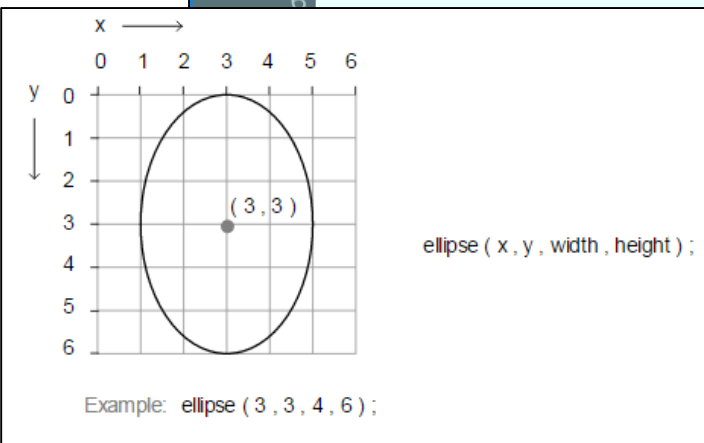
Click
to
Run



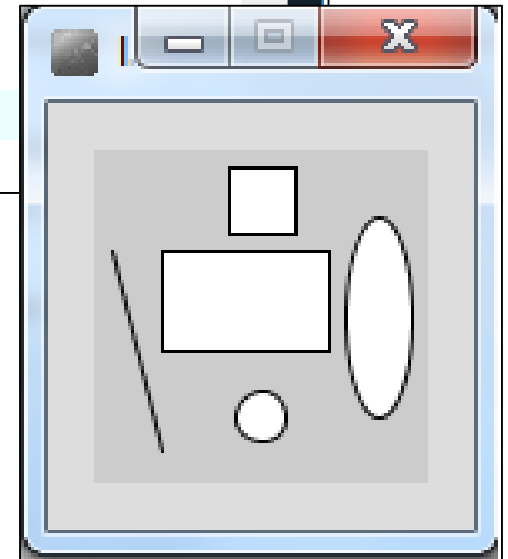
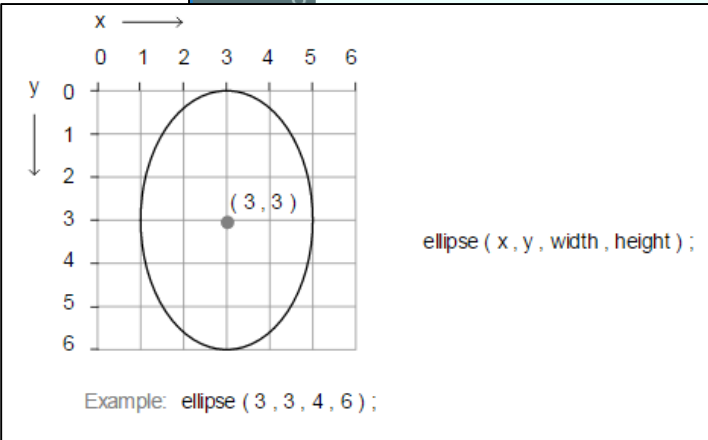
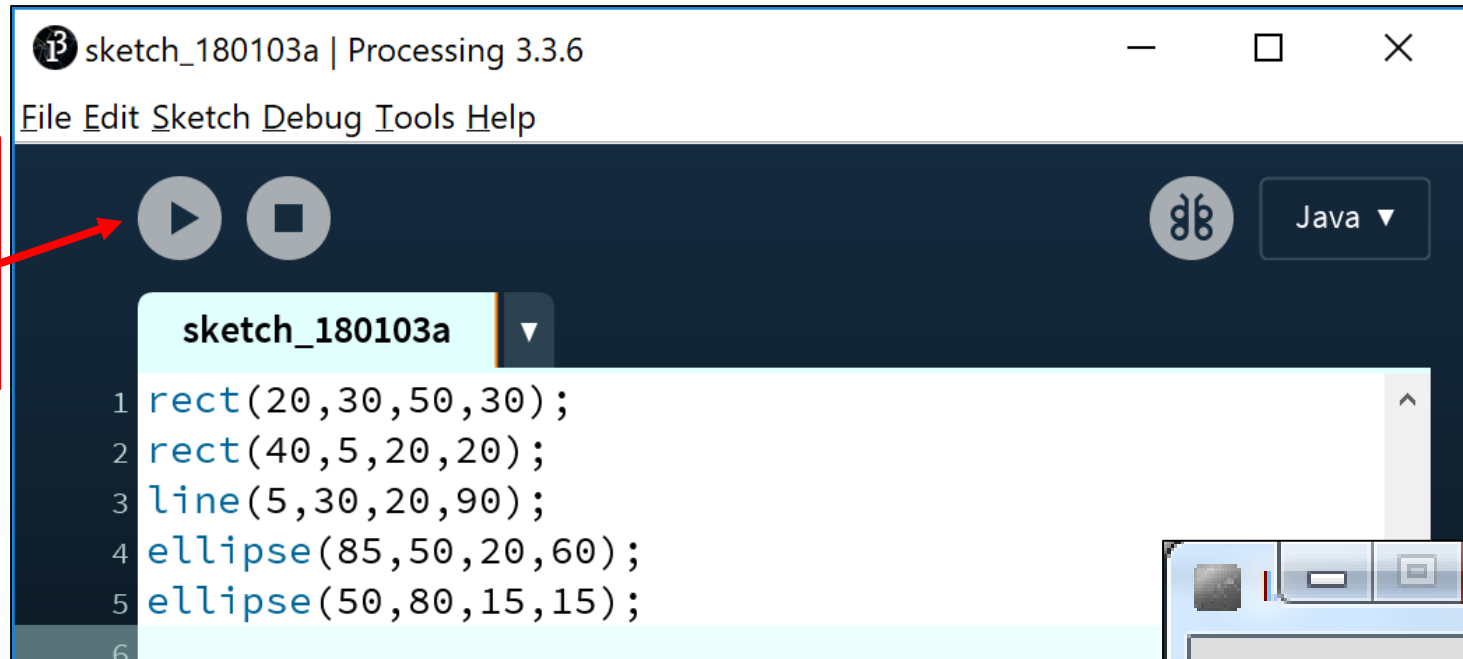
ellipse() – drawing a circle



```
sketch_180103a | Processing 3.3.6
File Edit Sketch Debug Tools Help
sketch_180103a
1 rect(20,30,50,30);
2 rect(40,5,20,20);
3 line(5,30,20,90);
4 ellipse(85,50,20,60);
5 ellipse(50,80,15,15);
6
```

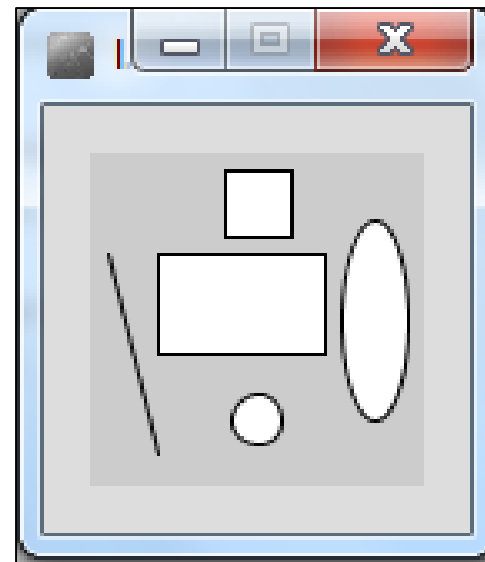


ellipse() – drawing a circle



Formatting the display window

- Our display window is looking fairly cramped.
- The default size of your display window is 100x100 pixels, which is quite small.



Formatting the display window

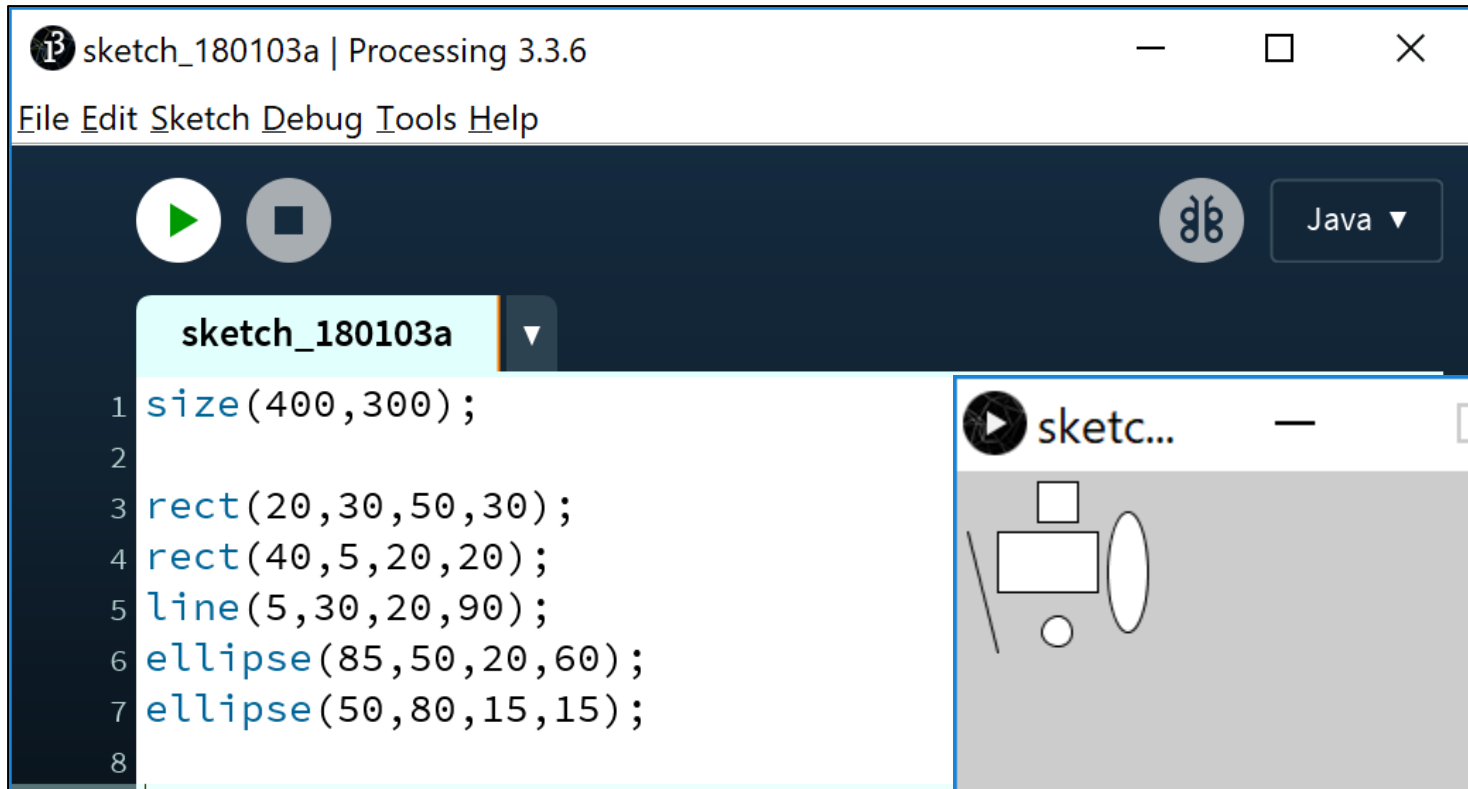
- We can change the size of the display window by calling the **size** function.
- When you use the size function in static drawings, it has to be the first line of code in your sketchbook.

```
size(w, h)
```

w = width of the display window

h = height of the display window

size()



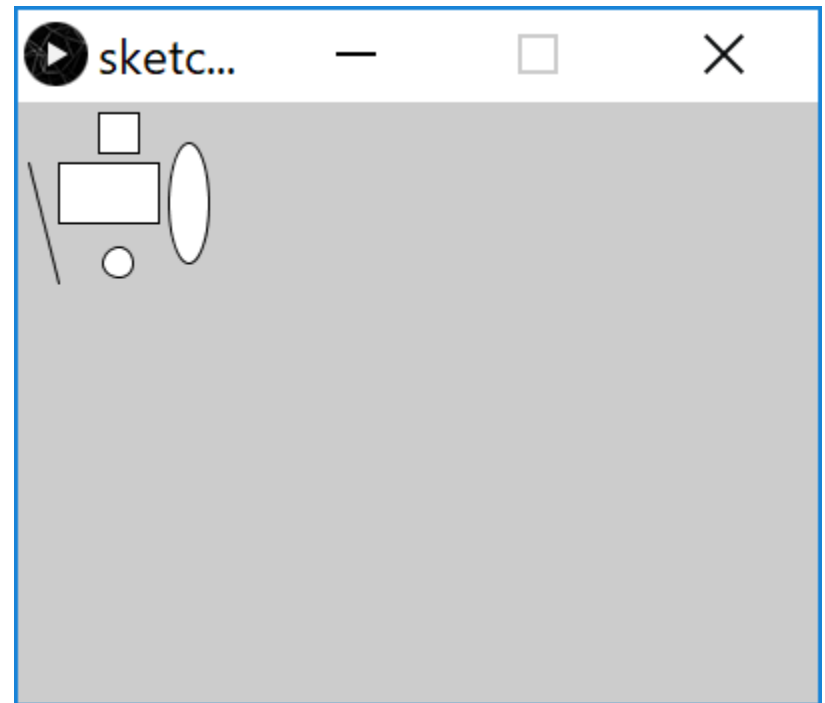
The image shows a screenshot of the Processing IDE. The main window is titled "sketch_180103a | Processing 3.3.6". The menu bar includes "File", "Edit", "Sketch", "Debug", "Tools", and "Help". The toolbar contains a play button, a stop button, a Java logo, and a "Java" dropdown menu. The code editor shows the following code:

```
1 size(400,300);  
2  
3 rect(20,30,50,30);  
4 rect(40,5,20,20);  
5 line(5,30,20,90);  
6 ellipse(85,50,20,60);  
7 ellipse(50,80,15,15);  
8
```

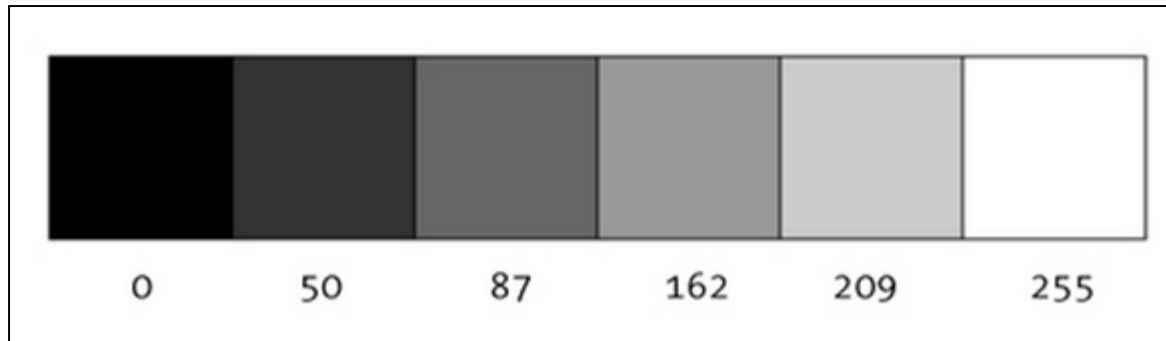
Below the code editor is a preview window titled "sketc...". The preview window displays a sketch with the following elements: a small square, a larger rectangle, a vertical line, a small circle, and an ellipse.

Formatting the display window

- Our display window looks less cramped now.
- But maybe we want to change the default gray colour?
- We could use the **background** function to set the colour to something else.



A note on colour first...Grayscale



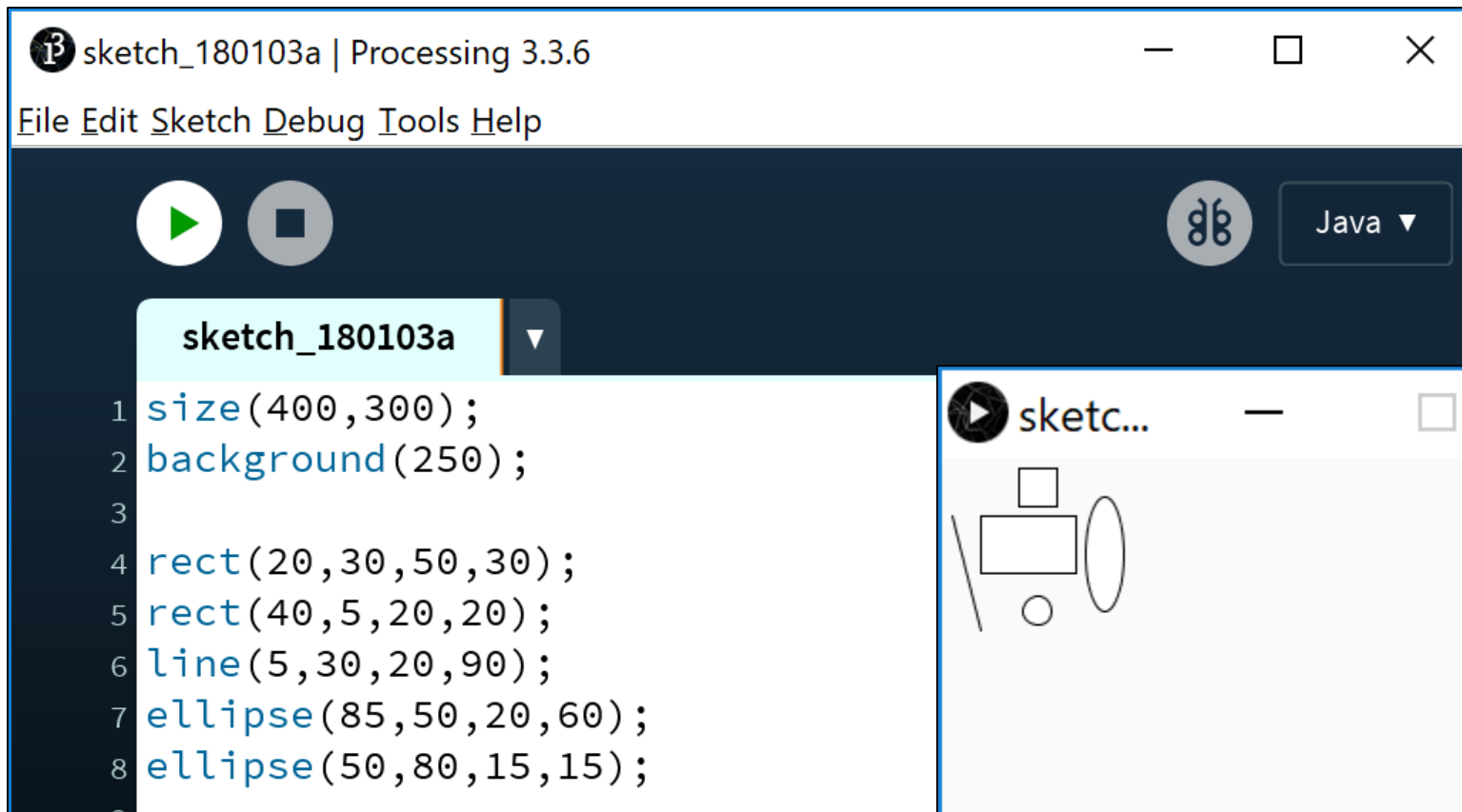
“0 means black, 255 means white. In between, every other number - 50, 87, 162, 209, and so on - is a shade of gray ranging from black to white.”

background() - syntax

background(grayScale)

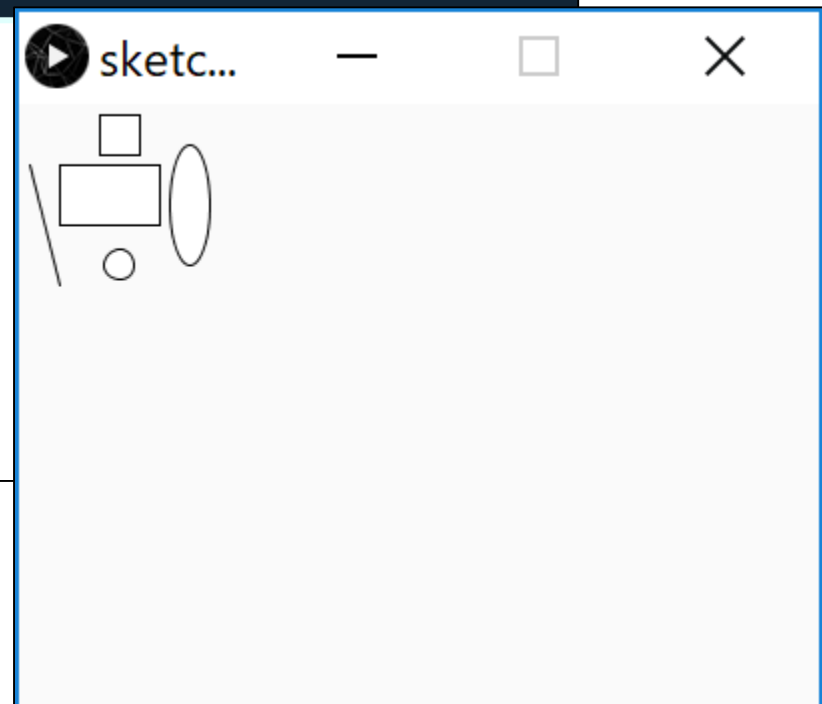
grayScale = grayscale colour (a number between
0 [black] and 255 [white] inclusive)

background()



The image shows a screenshot of the Processing IDE. The main window, titled "sketch_180103a | Processing 3.3.6", has a dark blue interface. It features a play button, a stop button, and a "Java" dropdown menu. Below the toolbar, a dropdown menu shows "sketch_180103a". The code editor displays the following code:

```
1 size(400,300);  
2 background(250);  
3  
4 rect(20,30,50,30);  
5 rect(40,5,20,20);  
6 line(5,30,20,90);  
7 ellipse(85,50,20,60);  
8 ellipse(50,80,15,15);  
9
```



The image shows a screenshot of a sketch window titled "sketc...". The window displays a simple line drawing of a robot-like figure. The drawing consists of a small square at the top, a larger rectangle below it, a vertical line extending downwards from the left side of the rectangle, a small circle at the bottom left, and a large vertical oval on the right side.

Questions?

