

Week 1: Introduction to Processing

Join our Slack

<https://comm2753-cc.slack.com/>

Using your RMIT email address, join our Slack. This is where you will be able to communicate with the lecturer and each other. You can post or answer any questions to the group or chat 1-1.

Download and install Processing

<https://processing.org/download/>

This is where we will be writing all our sketches.

Download Processing. Processing is available for Linux, Mac OS X, and Windows. Select your choice to download the software below.



3.3.7 (13 March 2018)

Windows 64-bit
Windows 32-bit

Linux 64-bit
Linux 32-bit
Linux ARM
(running on Pi?)

Mac OS X

Join the openProcessing class

<https://www.openprocessing.org/class/58192>

Code: 46F760

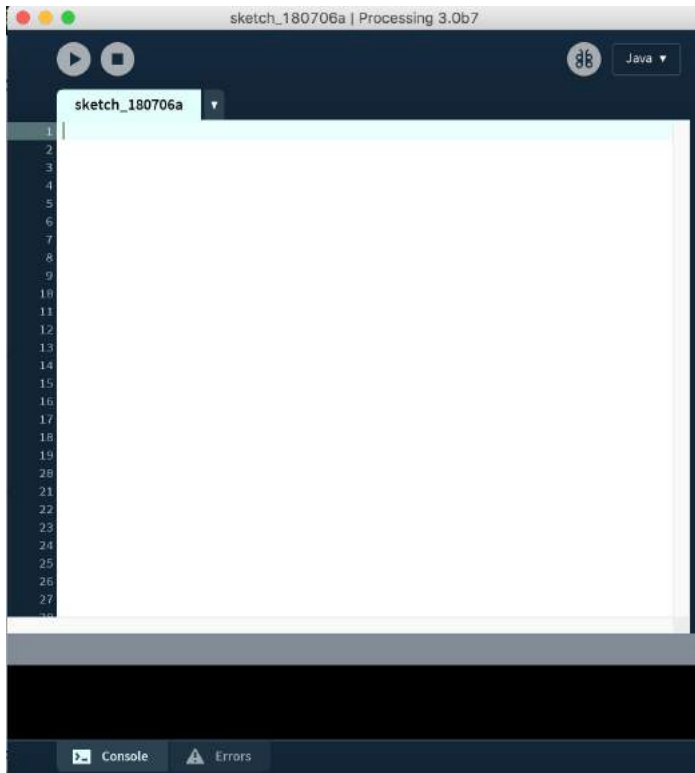
Please use an obvious name and your RMIT email when signing up.

This is where we will be submitting our weekly tasks and as a result, sharing what we've all been making.



Sketching with code

The concept of a “sketch” is fundamental to Processing. It introduces a new way of looking at code to encourage the ability to “sketch” with code.



When we save our sketch file, Processing will save that sketch inside a folder of the same name. For the sketch to run smoothly, you will always need to keep that sketch file (*.pde) inside the folder. Like this:

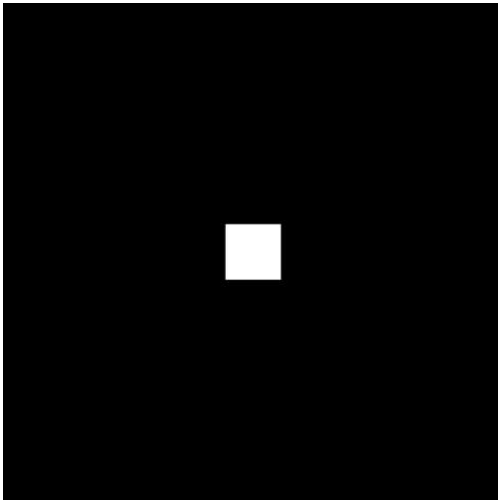


This is how the computer knows where to find your sketch and any files or parts you may reference in your sketch.

The first pretty square

Functions are like mini programs and the main two functions that exist in Processing are `setup()` and `draw()`. `setup()` runs once at the start of our sketch and `draw()` runs over and over again on repeat until you press stop.

Try the following:



```
//This runs once
void setup(){
  size(900,900);
  background(0);
}

//This runs on repeat
void draw(){
  rect(400,400,100,100);
}
```

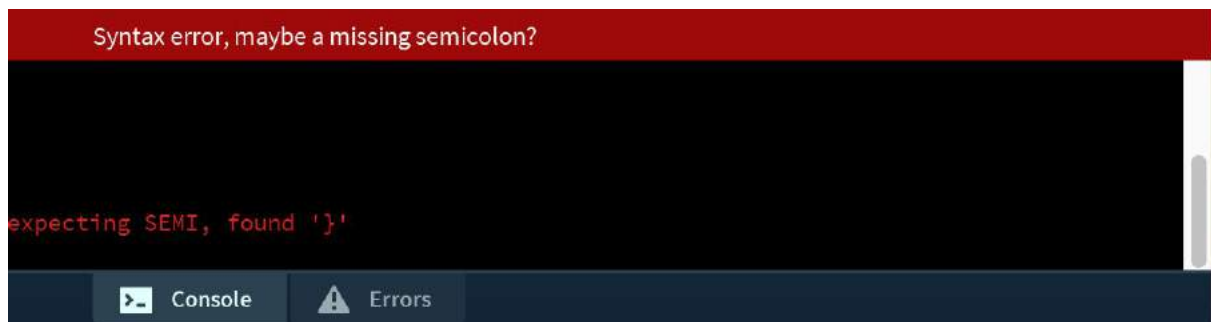
Can you describe what is happening in this sketch in plain english?

Change some numbers and it should become fairly obvious :)

You'll also notice here that I have included these lines: `//This runs once` and `//This runs on repeat`. This is what's known as commenting - the characters `//"` that precede the comment tell the computer to ignore this line. You will be using comments to add any inspiration or references you may have used for your sketches.

Try adding your own comment.

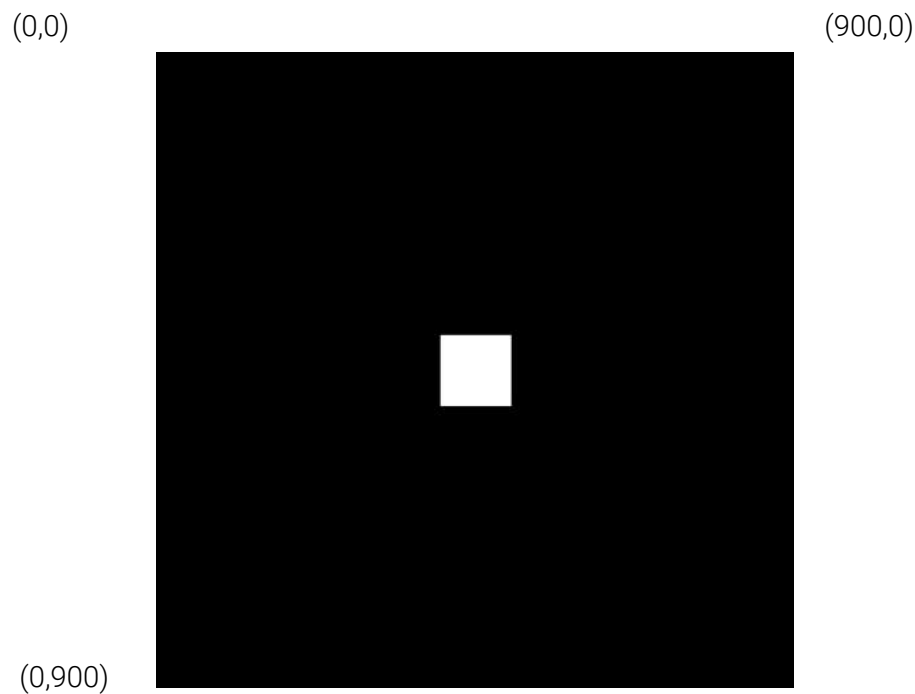
An important thing to also realise at this point is that the computer, unlike the English language, can't assume what you were *supposed* to say. Therefore, if there's a character out of place, the code won't run. Check those commas, brackets and semicolons! Don't worry too much about this, you will naturally get better at checking for these things the more you write code.



Co-ordinates

To tell the computer where to draw something, you will need to give it an x and y coordinate. You may remember these bad boys from maths class but in Computer Graphics, it's a little different.

The top left corner (x,y) is (0,0). The x coordinates increase to the right, the y coordinates **INCREASE** moving down the canvas.



Therefore `rect()` accepts the following 4 parameters:

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

By default, `rect()` accepts the x & y position of the rectangle's top left corner.

Let's add some colour!

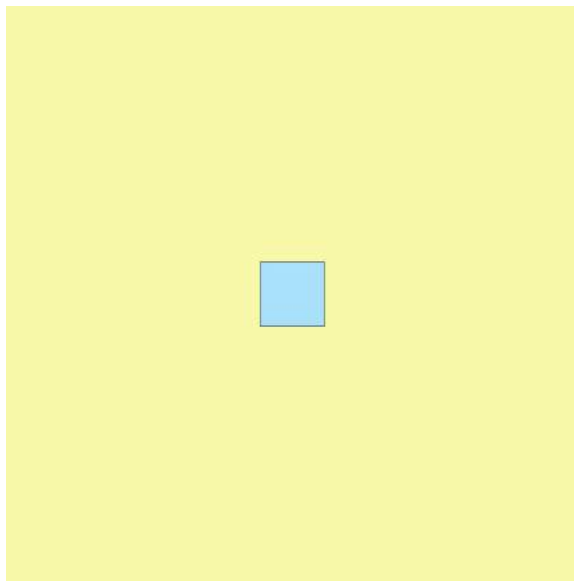
Sequence, colour and fill

So to add a fill to the rectangle, we need to add a line above our `rect()` function:

```
fill(169,225,250);
```

By default, Processing understands RGB colours. When we write something like `fill(255);` - the computer is actually interpreting that to be `fill(255,255,255);`

Try the following:



```
void setup(){
  size(900,900);
  background(248,250,169);
}

void draw(){
  fill(169,225,250);
  rect(400,400,100,100);
}
```

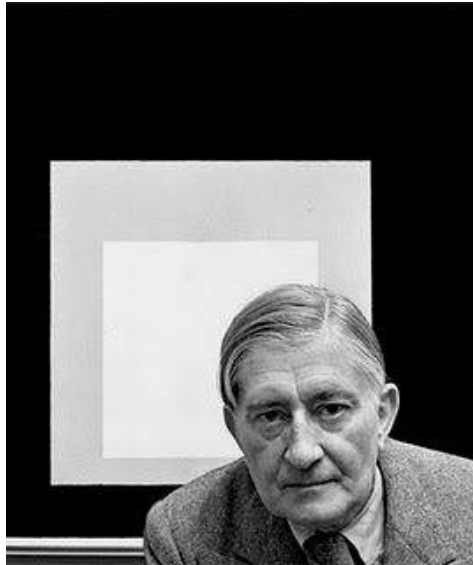
Under Tools > Colour Selector - you can choose your own RGB colours. We will delve more into colours and what the numbers all mean in Week 4. For now, just copy and paste the R, G and B values as comma-separated numbers like the example above.

In addition, let's remove that awful black stroke. By default, the `stroke(0);` is applied to all 2D Primitives. In the setup function, include the line:

```
noStroke();
```

What if we wanted to add another rectangle? One with a stroke and one without? Have a go.

Inspiration: Josef Albers






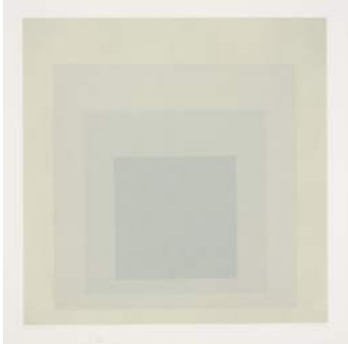

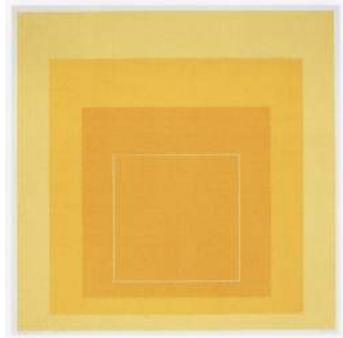
"If one says "Red" (the name of a color) and there are 50 people listening, it can be expected that there will be 50 reds in their minds. And one can be sure that all these reds will be very different."
—Josef Albers, *Interaction of Color* (1963)

Josef Albers was a German-born American artist and educator whose work, both in Europe and in the United States, formed the basis of modern art education programs of the twentieth century....Accomplished as a designer, photographer, typographer, printmaker, and poet, Albers is best remembered for his work as an abstract painter and theorist. He favored a very disciplined approach to composition. Most famous of all are the hundreds of paintings and prints that make up the series, *Homage to the Square*. In this rigorous series, begun in 1949, Albers explored chromatic interactions with nested squares... (he) often recorded the colors he used on the back of his works. Each painting consists of either three or four squares of solid planes of color nested within one another, in one of four different arrangements...

https://en.wikipedia.org/wiki/Josef_Albers

Homage to the Square, Josef Albers

		
<p>Study for Homage to the Square: Night Shades (1956)</p> <p>© 2018 The Josef and Anni Albers Foundation / Artists Rights Society (ARS), New York</p>	<p>Homage to the Square: Two Whites Between Two Yellows (1958)</p> <p>© 2018 The Josef and Anni Albers Foundation / Artists Rights Society (ARS), New York</p>	<p>Study for Homage to the Square: Early Rising A (1961)</p> <p>© 2018 The Josef and Anni Albers Foundation / Artists Rights Society (ARS), New York</p>

 <p>Full from Homage to the Square: Ten Works by Josef Albers (1962)</p> <p>© 2018 The Josef and Anni Albers Foundation / Artists Rights Society (ARS), New York</p>	 <p>Joy from Homage to the Square: Ten Works by Josef Albers (1962)</p> <p>© 2018 The Josef and Anni Albers Foundation / Artists Rights Society (ARS), New York</p>	 <p>Patina from Homage to the Square: Ten Works by Josef Albers (1962)</p> <p>© 2018 The Josef and Anni Albers Foundation / Artists Rights Society (ARS), New York</p>
 <p>Day + Night III from Day and Night: Homage to the Square (1963)</p> <p>© 2018 The Josef and Anni Albers Foundation / Artists Rights Society (ARS), New York</p>	 <p>Midnight + Noon IV from Midnight and Noon (1964)</p> <p>© 2018 The Josef and Anni Albers Foundation / Artists Rights Society (ARS), New York</p>	 <p>WLS I from White Line Squares (Series I) (1966)</p> <p>© 2018 The Josef and Anni Albers Foundation / Artists Rights Society (ARS), New York</p>

Week 1 Exercise

Create an ode to Josef Albers' "Homage to the Square". Start with finding a colour combination you like and then create a composition inspired by Albers with 3 or 4 different sized squares. Try different arrangements and submit your final piece. Add a comment with the initial Albers' work/s and any other references you were inspired by.

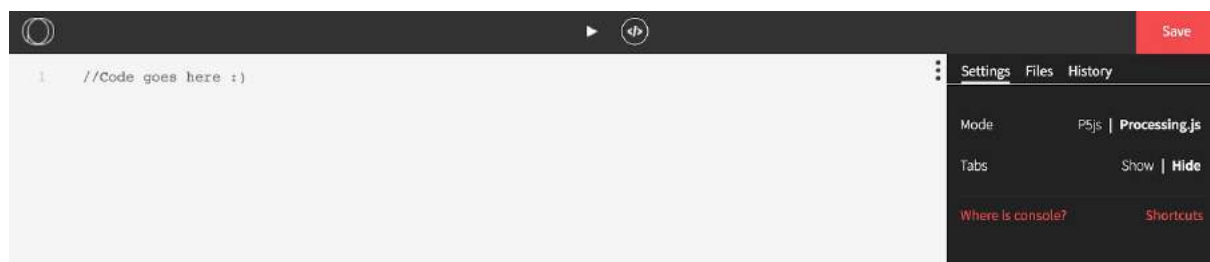
Uploading your weekly exercises

When you're ready, go to openProcessing and log-in.
Navigate to "Sketches" and select "Create a Sketch".



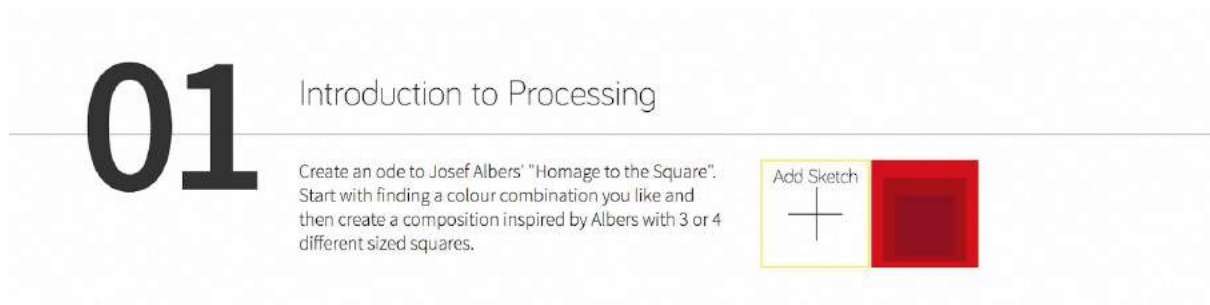
On the far right panel, make sure your "Mode" is set to "Processing.js" - this will save you from having to translate your code .

Copy and paste your code from Processing to the light grey box and hit Save.



Go to "My Feed" and navigate to "Classes"

Scroll down to the week that you want to submit for and hit "Add Sketch". Select your newly uploaded sketch and hey presto, we've submitted our first weekly exercise :)



Make sure that "Who can see it" is set to *Anyone* or *Me & My Classes* in addition to "Hide Source Code" being set to *Off*.

