

Programming for Designers

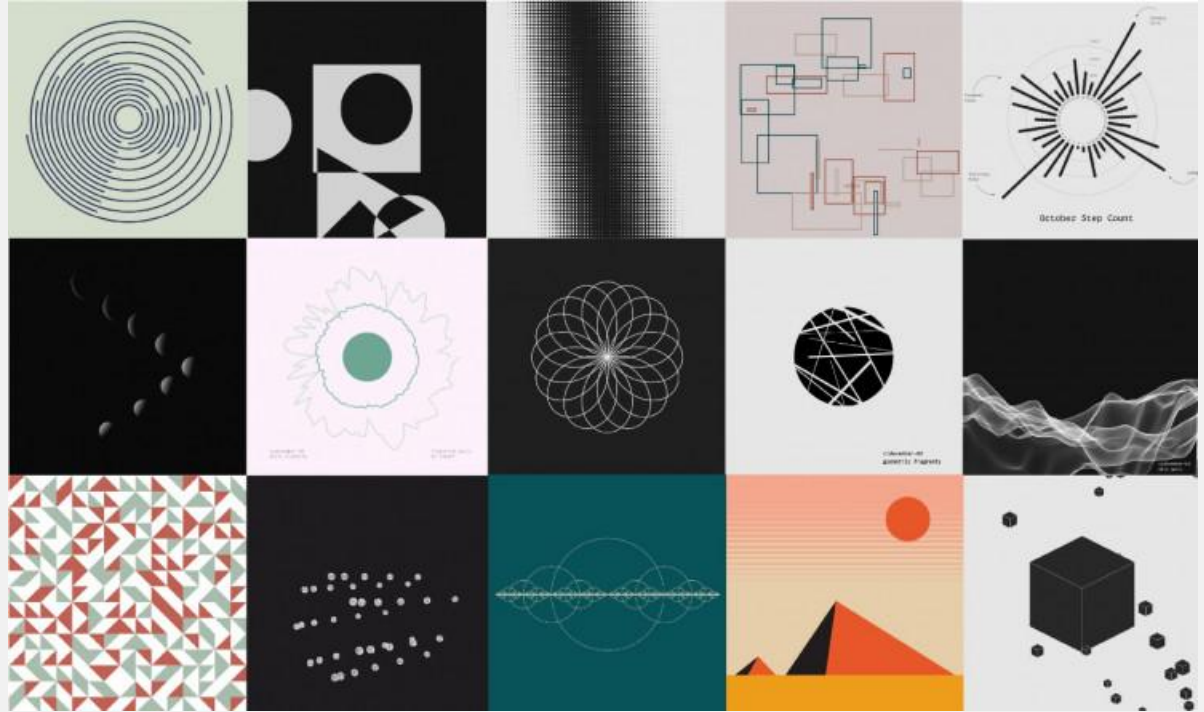
DECO1012 - Design Programming
Week 1



THE UNIVERSITY OF
SYDNEY

Creative coding is..

“... the act of using computer programming software to create works of art, design, architecture, and even fashion.” [1]



[1] <https://interestingengineering.com/everything-you-need-to-know-about-the-artistic-world-of-creative-coding>

Challenge 1: Step 1

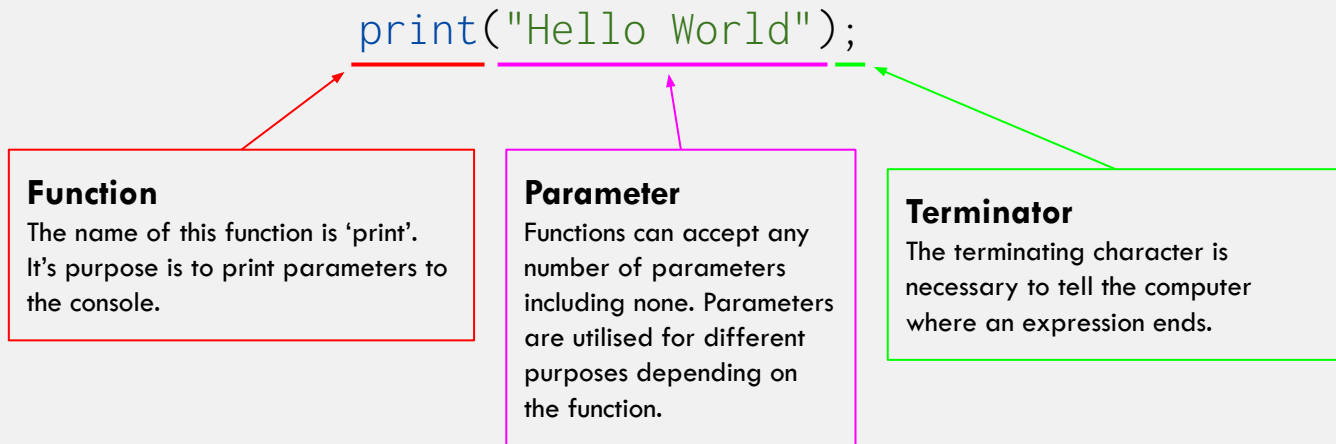
Hello World (Console Output)



THE UNIVERSITY OF
SYDNEY

Hello World

We can achieve this in p5.js very easily with a function call.



“Function Calls” are also known as “Function Invocations”. You can also say “Calling a Function” or “Invoking a Function”.

“Parameters” are also known as “Arguments”.

The terminating character in JavaScript is always a semicolon. JavaScript is a very forgiving language in that if you forget a terminating character your code might work fine. But it is good practice to terminate your expressions to avoid undesired behaviour. Other languages aren't as forgiving.



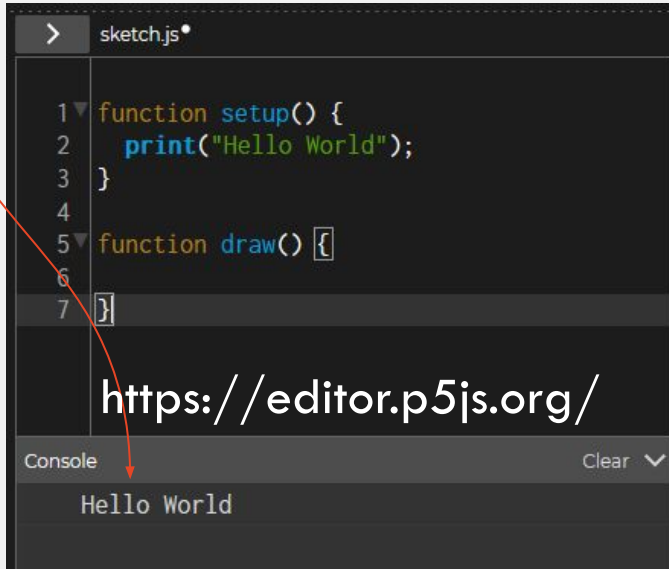
THE UNIVERSITY OF
SYDNEY

Step 1: Hello World (Console Output) [2 minutes]

Take **2 minutes** to complete steps 1 and 2. If everything is working well move on to steps 3 and 4.

If you don't see "Hello World" in the console, ask your tutor for help.

1. Try this yourself by replicating this sketch in your editor.
2. Run the sketch.
3. Try changing the parameter of the print function.
4. Try moving the print function call into the draw function and see what the difference is.



```
> sketch.js
1 function setup() {
2   print("Hello World");
3 }
4
5 function draw() {
6
7 }
```

<https://editor.p5js.org/>

Console Clear ▾

Hello World



THE UNIVERSITY OF
SYDNEY

Challenge 1: Step 2

Hello World (Canvas Output)



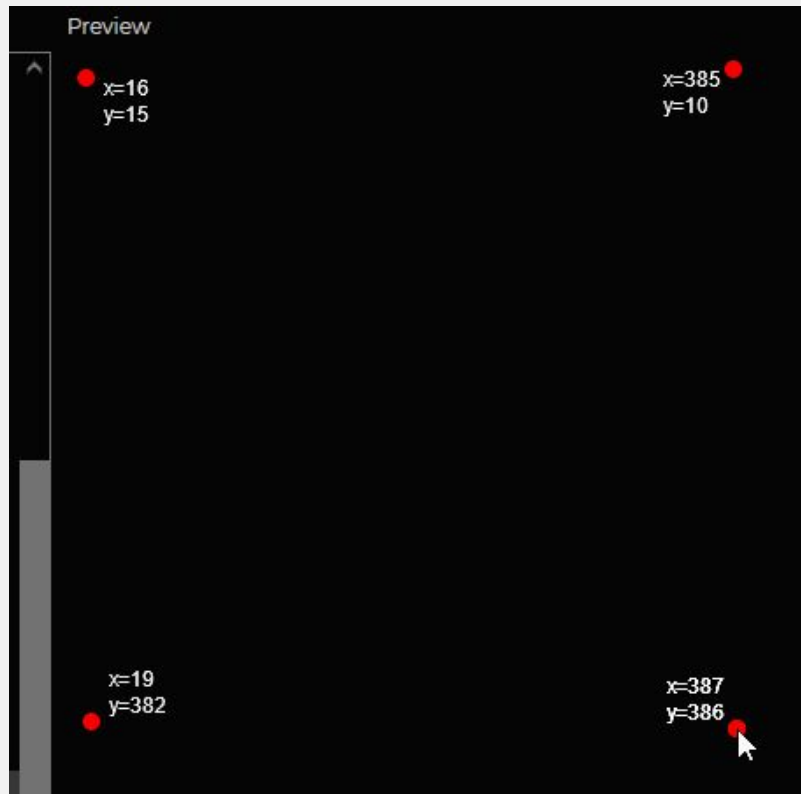
THE UNIVERSITY OF
SYDNEY

Canvas Coordinates

The p5.js canvas uses a cartesian coordinate system (x,y). When working in 2D, the origin point (0,0) of the coordinate system is in the top left.

The x value increases as you travel left to right.

The y value increases as you travel top to bottom.



Drawing functions

Many 2D drawing functions are available in p5.js and you can see examples of all of them in the p5 [reference](#) pages. The reference pages are an excellent resource which you should aim to become very familiar with.

Bookmark the reference page now. You will be using it a lot.

Two functions that I will show you are the [line\(\)](#) and [ellipse\(\)](#) functions. As you might have guessed, they are for drawing lines and ellipses.

Drawing functions: line()

A point on a 2D cartesian coordinate system has two values, x and y. To draw a line you need two points, a start point and an end point. So you will need a total of four parameters to draw a line.

Within the p5 reference page for [line\(\)](#) you will see two sections. **Syntax** and **Parameters**.

Syntax tells us how to call the function and **Parameters** tells us what each parameter is used for.

Syntax:

```
line(x1, y1, x2, y2);
```

Parameters:

x1 Number: the x-coordinate of the first point

y1 Number: the y-coordinate of the first point

x2 Number: the x-coordinate of the second point

y2 Number: the y-coordinate of the second point



THE UNIVERSITY OF
SYDNEY

Drawing functions: ellipse()

To draw a circle you need one point (x and y) and a width.

The fourth parameter is indicated as **optional** and denoted in the syntax by square brackets `[]`.

If this parameter is omitted, the function will assume the height is the same as the width and will draw a circle.

Syntax:

```
ellipse(x, y, w, [h]);
```

Parameters:

x Number: x-coordinate of the center of ellipse.

y Number: y-coordinate of the center of ellipse.

w Number: width of the ellipse

h Number: height of the ellipse. (Optional)

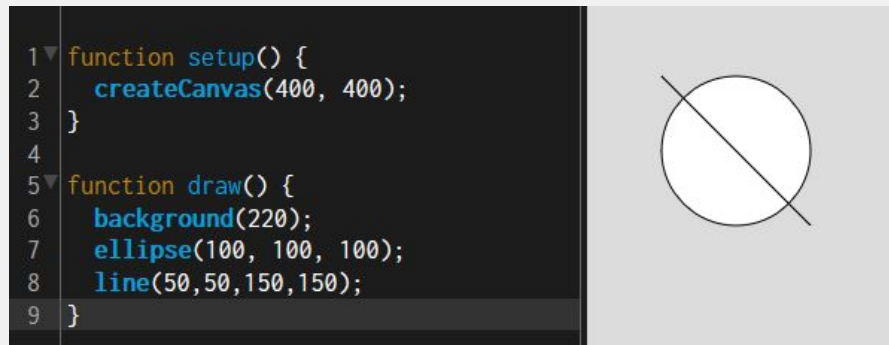
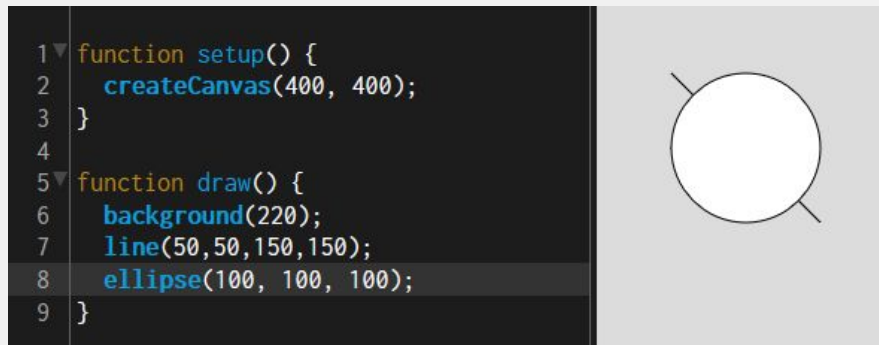


THE UNIVERSITY OF
SYDNEY

Order of execution

Each expression will be evaluated in succession from top to bottom. As drawing functions are called, they are drawn onto the canvas. This means that if you draw a shape after another in the same place, the second shape function will be in front of the first.

Below you can see two examples where line 7 and 8 of the sketch are swapped.



Step 2: Hello World (Canvas Output) [10 minutes]

You now have everything you need to attempt the step 2. Take **10 minutes** to replicate the picture in the embedded p5 editor (or your own editor in another window if you prefer).

- Consult the p5 [reference](#) pages for information on the [line\(\)](#) and [ellipse\(\)](#) functions and their parameters.
- Don't try to recreate the image perfectly. It's more important that you understand function calls, function parameters, canvas coordinates and the order of execution.

Make sure to ask for help if you need it.

The Basics: Styling Functions



THE UNIVERSITY OF
SYDNEY

Styling Functions

Styling functions allow you to set the colour of the fill and stroke of drawing commands as well as setting the thickness of line and turning off the fill and/or stroke entirely.

These functions are:

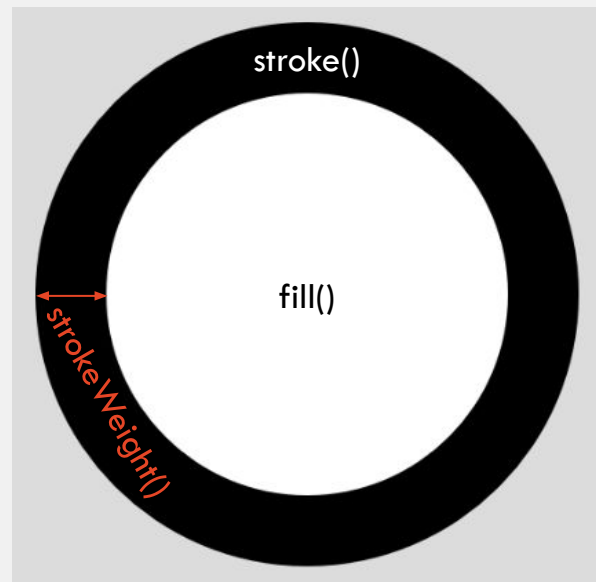
[fill\(\)](#) - for setting the colour of the fill.

[stroke\(\)](#) - for setting the colour of the stroke.

[strokeWeight\(\)](#) - for setting the thickness of the stroke.

[noFill\(\)](#) - for turning fill off.

[noStroke\(\)](#) - for turning stroke off.



Styling Functions: fill() and stroke()

Both fill() and stroke() accept colour parameters. Colours can be provided in a variety of ways.

All styling functions follow the order of execution and apply to shape functions drawn after styling functions.

Think of it like dipping your paintbrush into a colour before painting onto the canvas.



<https://giphy.com/JulieSmithSchneider>

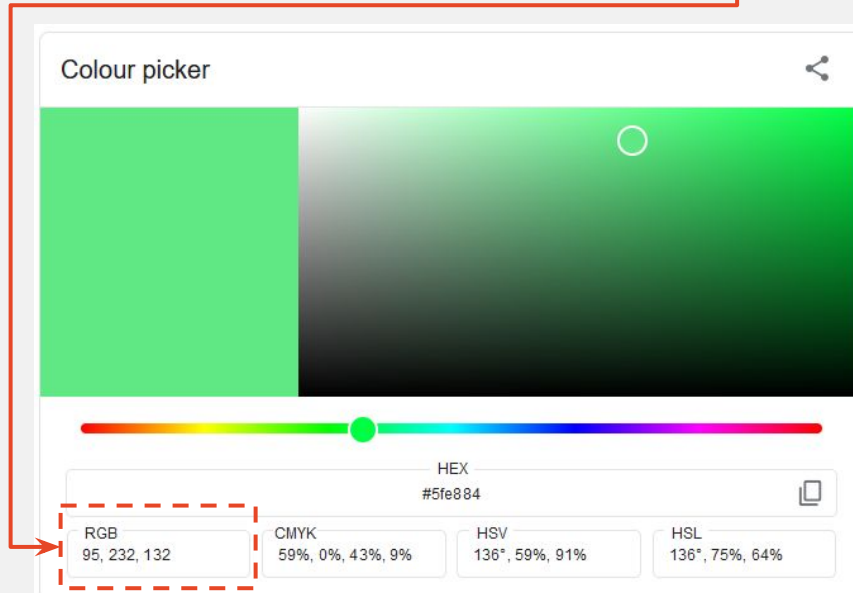
You already know the single parameter method for grayscale from background(). The next method I will teach uses three parameters in this order: **Red**, **Green** and **Blue**. By default, these values should be in a range between 0 and 255 just like the grayscale method.

```
background(200, 75, 30);  
fill(0, 0, 255);  
stroke(0, 255, 0);  
ellipse(100, 100, 50);
```

Preview

Using colours

Colour pickers can be very useful for quickly generating RGB values that can be used in `background()`, `fill()` and `stroke()`.



Styling Functions: `noFill()` and `noStroke()`

Both `noFill()` and `noStroke()` are very easy to use as they require no parameters. Shapes drawn after these functions will have either transparent fill or transparent stroke or both.

In order to undo these functions, you can simply set the fill and/or stroke again using `fill()` and `stroke()` with your desired colour parameters.

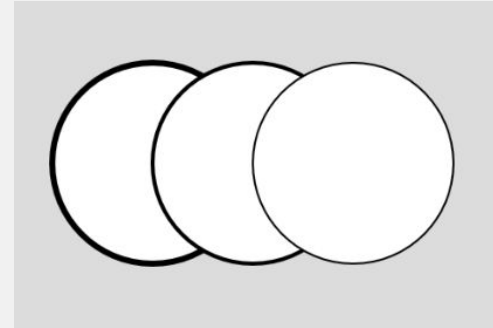


"The Joy of Painting with Bob Ross" (Television series)

Styling Functions: `strokeWeight()`

`strokeWeight()` sets the thickness of lines and shape borders. The function accepts a single parameter with a default of 1. The value of the parameter is the width of the line in pixels.

```
strokeWeight(3);  
ellipse(100,100,100);  
strokeWeight(2);  
ellipse(150,100,100);  
strokeWeight(1);  
ellipse(200,100,100);
```



It is possible to pass parameters with decimal places to `strokeWeight()` as well as values below 1. This usually results in a line of roughly the same width, although the stroke colour is lighter or darker depending on the value provided.

`strokeWeight(0)` does not have the same effect as `noStroke()`. Instead, it will result in almost transparent lines. This is not always immediately obvious but when transparent layers are added over the top the lines can become visible.



THE UNIVERSITY OF
SYDNEY

Challenge 1: Step 3

Basics of Drawing to Canvas



THE UNIVERSITY OF
SYDNEY

Step 3: Basics of Drawing to the Canvas [15 minutes]

You now have everything you need to attempt the step 3. Remember to consult the p5 [reference](#) pages.

Step 1: Don't put too much effort into recreating the image perfectly. Make sure you understand styling functions and how they affect shape functions paying close attention to the order of execution.

Step 2: Play around with colours. If you have time, try out some of the other methods to set colour found in the [fill\(\) reference page](#).

Make sure to ask for help if you need it.

Challenge 2: Using the Draw Loop

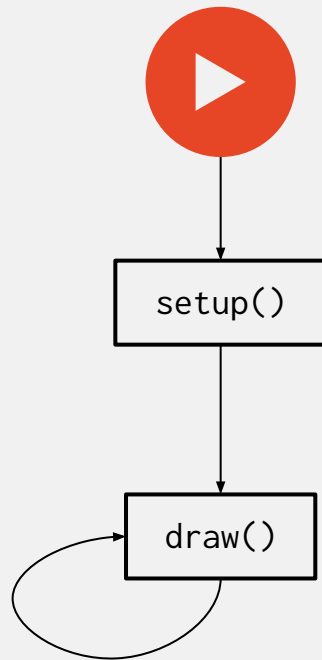


THE UNIVERSITY OF
SYDNEY

The anatomy of a sketch

p5.js is expecting us to define `setup()` or `draw()` and if they exist, then p5.js will call them.

When we run our sketch, p5 calls `setup()` **once** and then calls `draw()` **repeatedly** until the sketch is terminated. This is all done by p5 elsewhere, hidden inside the p5.js script file.



Challenge 2: Using the Draw Loop [10 minutes]

You now have everything you need to attempt the second challenge on [canvas](#).

Remember to consult the p5 [reference](#) pages.

Step 1: Make use of the built-in variables and try moving functions to different places in the sketch to gain a greater understanding of the anatomy of a sketch and the order of execution.

Step 2: You don't need to complete step 2 in the allotted time. If you don't finish this challenge, you can continue it in your own study time.

Make sure to ask for help if you need it.



Challenge 3: Relative Values

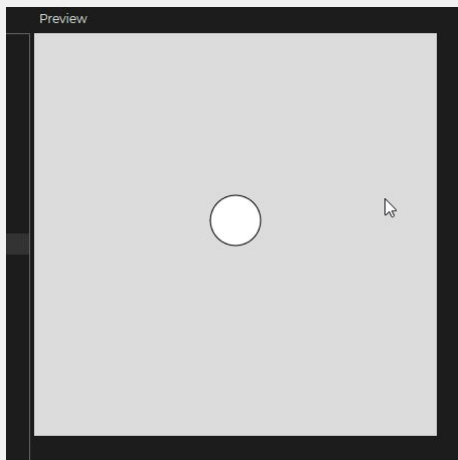
Absolute Values vs. Relative Values

An absolute value does not change based on any external influence.

A relative value is derived from another value.

The x coordinate of the circle is absolute.

The y coordinate of the circle is relative to the position of the mouse.



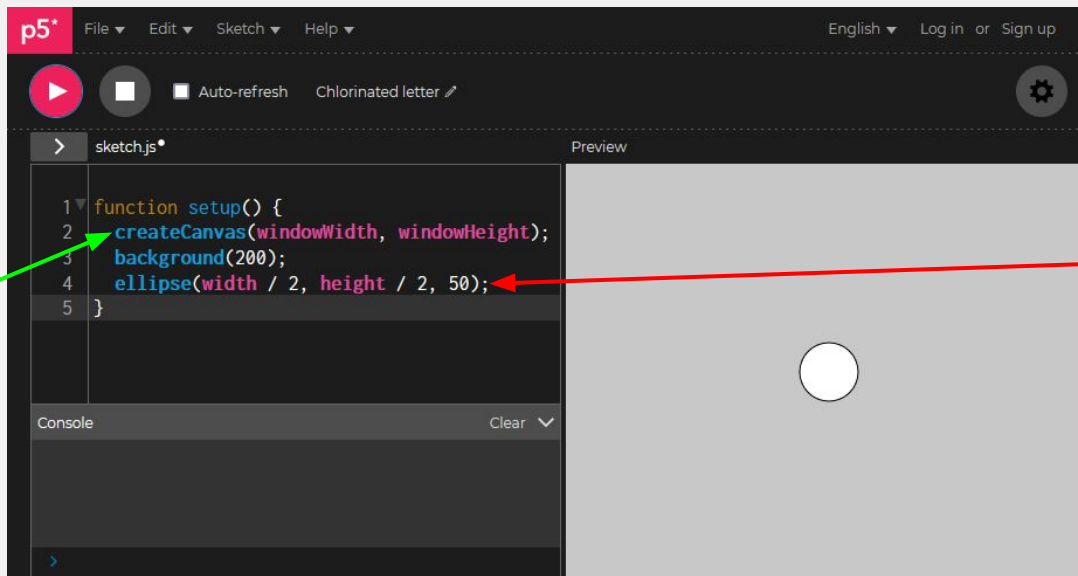
```
function setup() {  
  createCanvas(400, 400);  
}  
  
function draw() {  
  background(220);  
  ellipse(200, mouseY, 50);  
}
```

Relative Values

Relative values are very useful when we want attributes to adapt to unknown values.

Example:

Because the canvas size is based on the size of the browser window, we cannot predict the size of the canvas.



Whatever the canvas size may be, `width/2` and `height/2` will always calculate a coordinate in the center of the canvas.



THE UNIVERSITY OF
SYDNEY

Challenge 3: Using Relative Values [20 minutes]

You now have everything you need to attempt the third challenge on [canvas](#).

Remember to consult the p5 [reference](#) pages.

Step 1: Use arithmetic operators and in-built variables to solve the problem.

Step 2: Don't put too much effort into recreating the image perfectly. Instead, play around with your variable values to understand how they affect the image.

Make sure to [ask for help](#) if you need it.

Before we wrap up...

Try and re-try this week's challenges before the next class. Practice helps!

The ***practice*** quiz for Assessment 1 will be released on **23rd February 17:00**. Complete this quiz by **1st March 23:59**. This practice quiz is **not marked**.

If you have any questions and would like to talk to your tutor, arrange for a meeting using Canvas (option to “setup a meeting” available in the Home page).

See you next week!



THE UNIVERSITY OF
SYDNEY