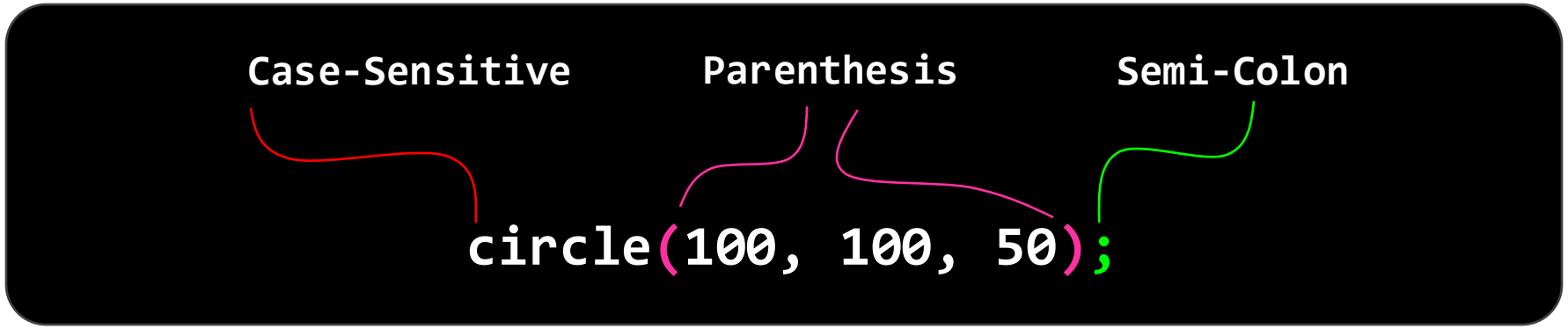
Luxembourg Tech School

LEVEL GO
2025-2026

2 - Draw with p5.js

Warm-Up: Writing Code

Writing Code

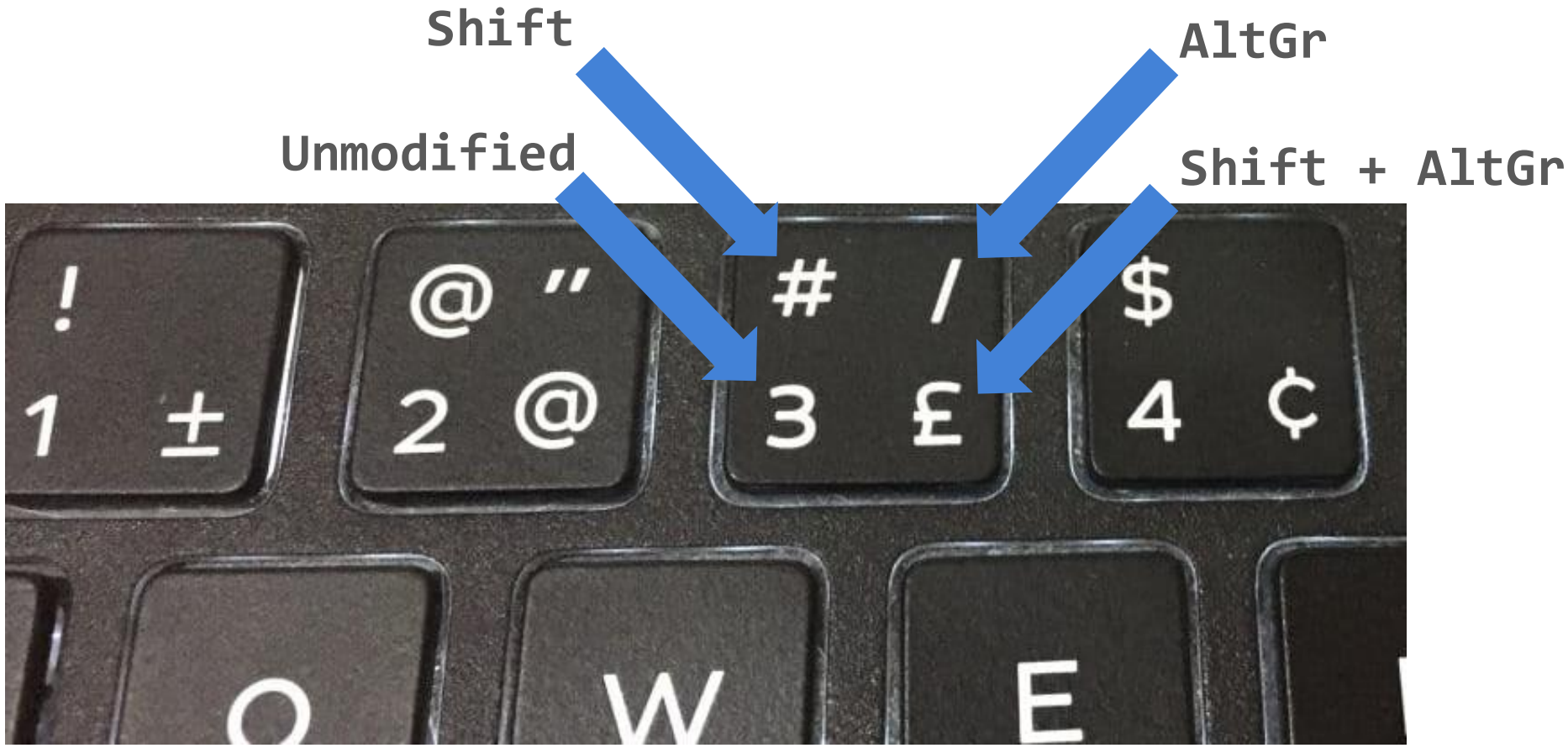


Every character, symbol, number has a meaning!

( ) = Parenthesis
{ } = Curly Braces
[ ] = Square Brackets
“” = quotes

: = Colon
; = Semi-Colon
_ = Underscore
/ = Slash
\ = Backslash
| = Pipe

Writing Code: modifier keys



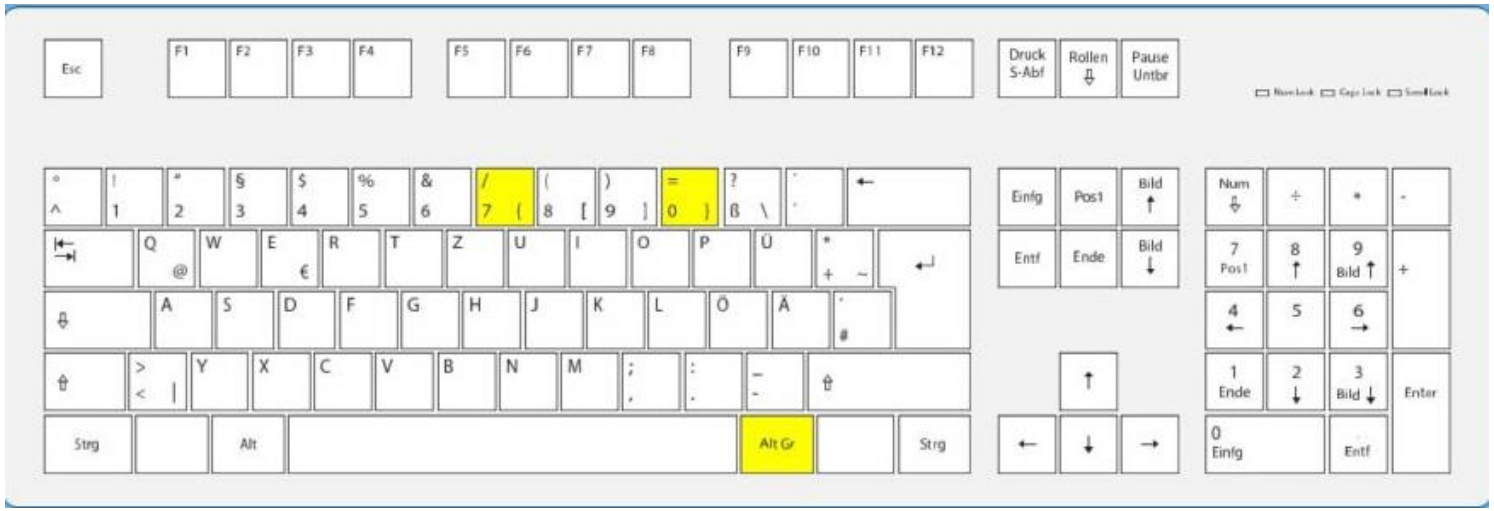
Try to find them on your keyboard

( ) = Parenthesis
{ } = Curly Braces
[ ] = Square Brackets

: = Colon
; = Semi-Colon
_ = Underscore
/ = Slash
\ = Backslash
| = Pipe

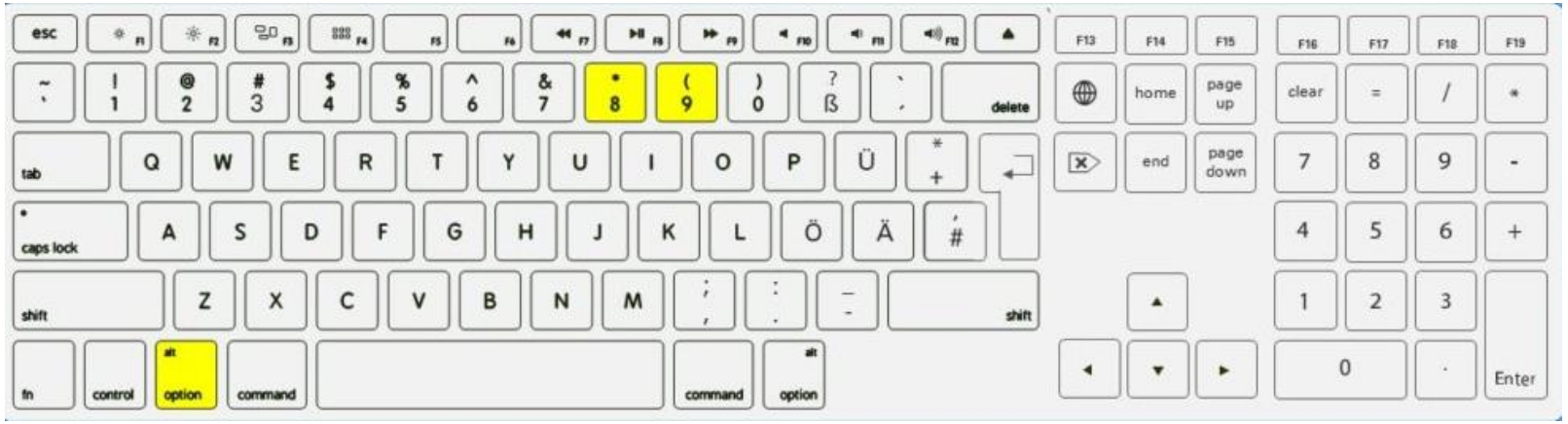
Try to find them on your keyboard

`{ } = Curly Braces`



Windows

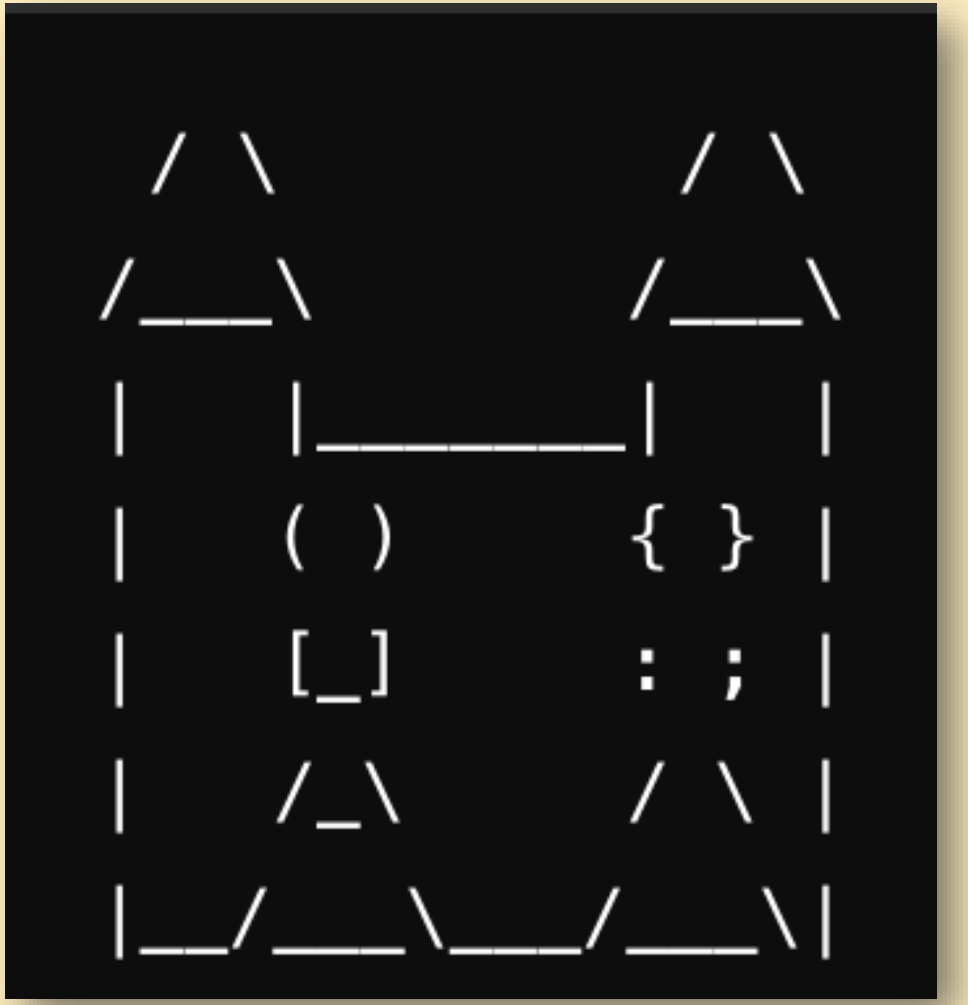
{ }



Mac

{ }

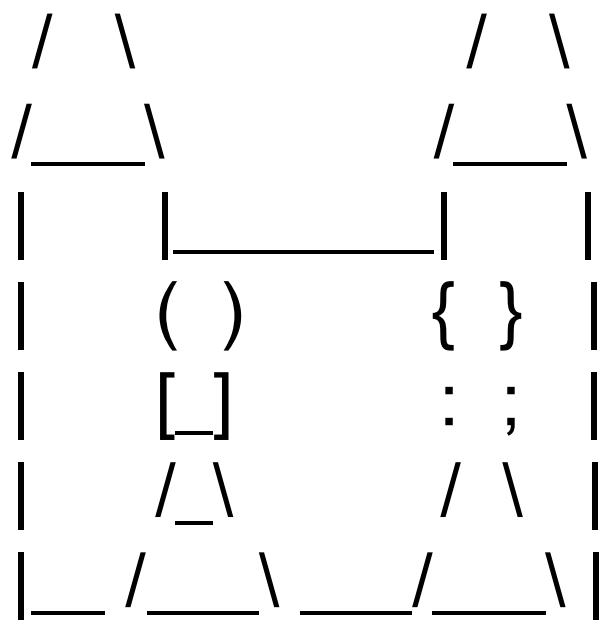
Try to create a castle

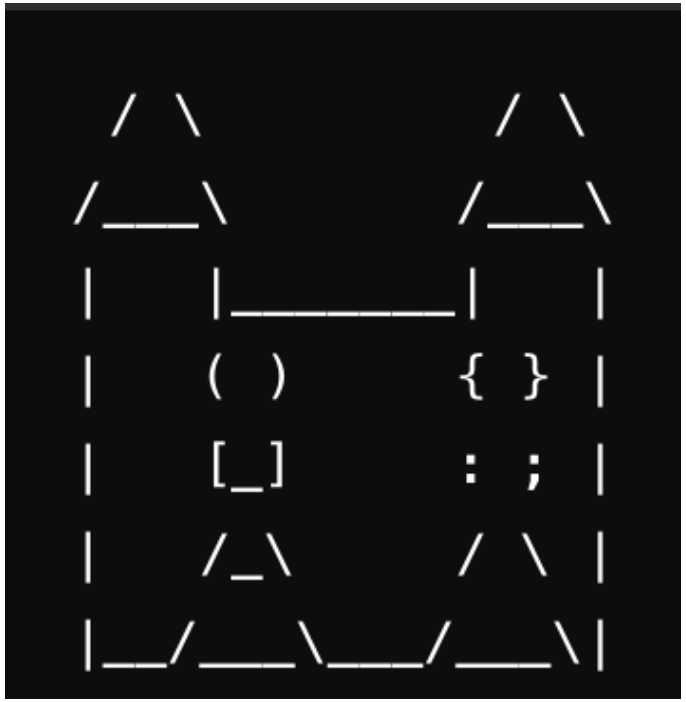


Open an empty text file and try to create a castle like the one on the left by finding and typing the following characters on your keyboard:

```
( ) = Parenthesis
{ } = Curly Braces
[ ] = Square Brackets
  : = Colon
  ; = Semi-Colon
  _ = Underscore
  / = Slash
  \ = Backslash
  | = Pipe
```

It could look something like this :)



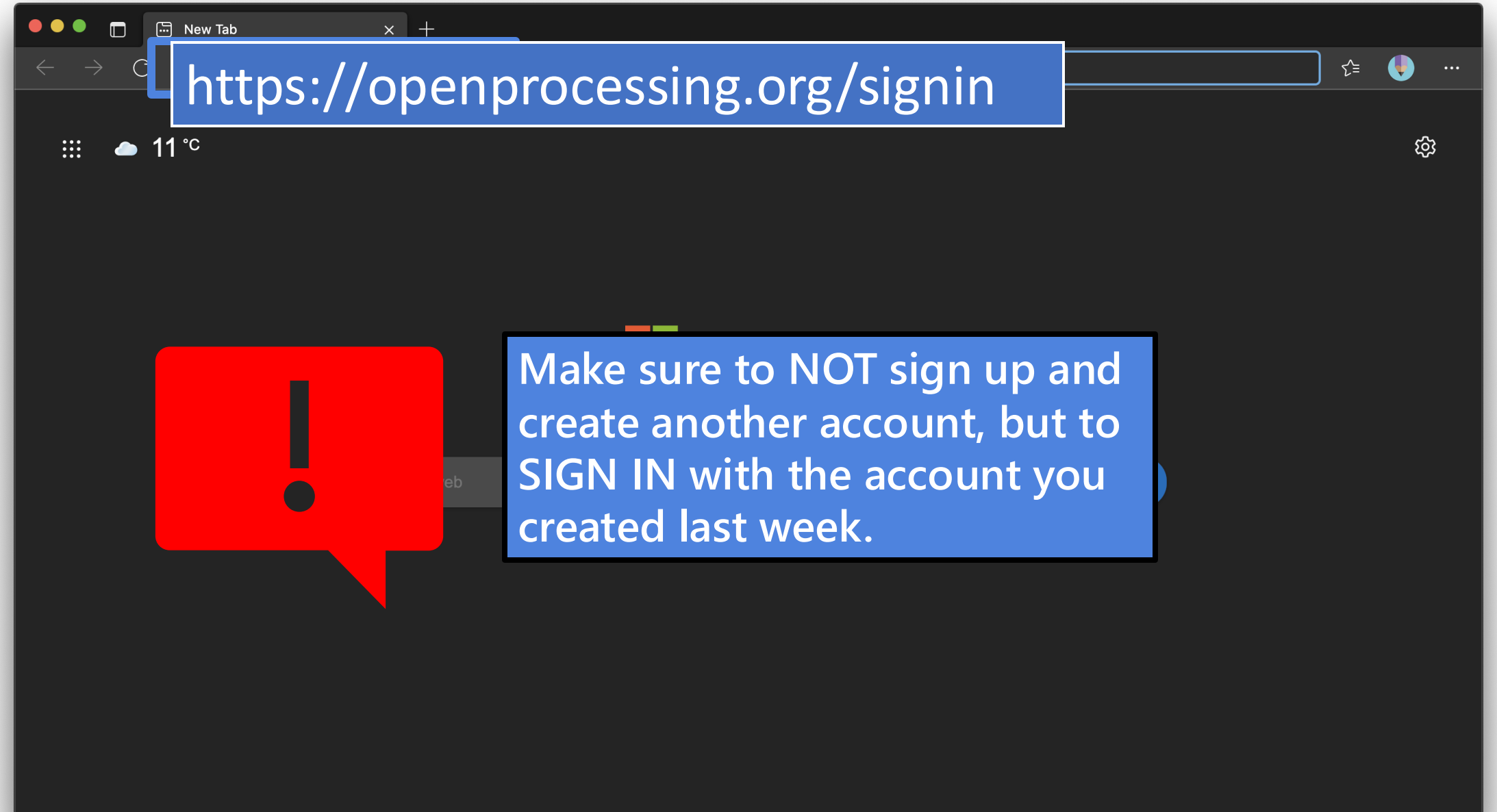


By the way, you just created a piece of **ASCII art.**

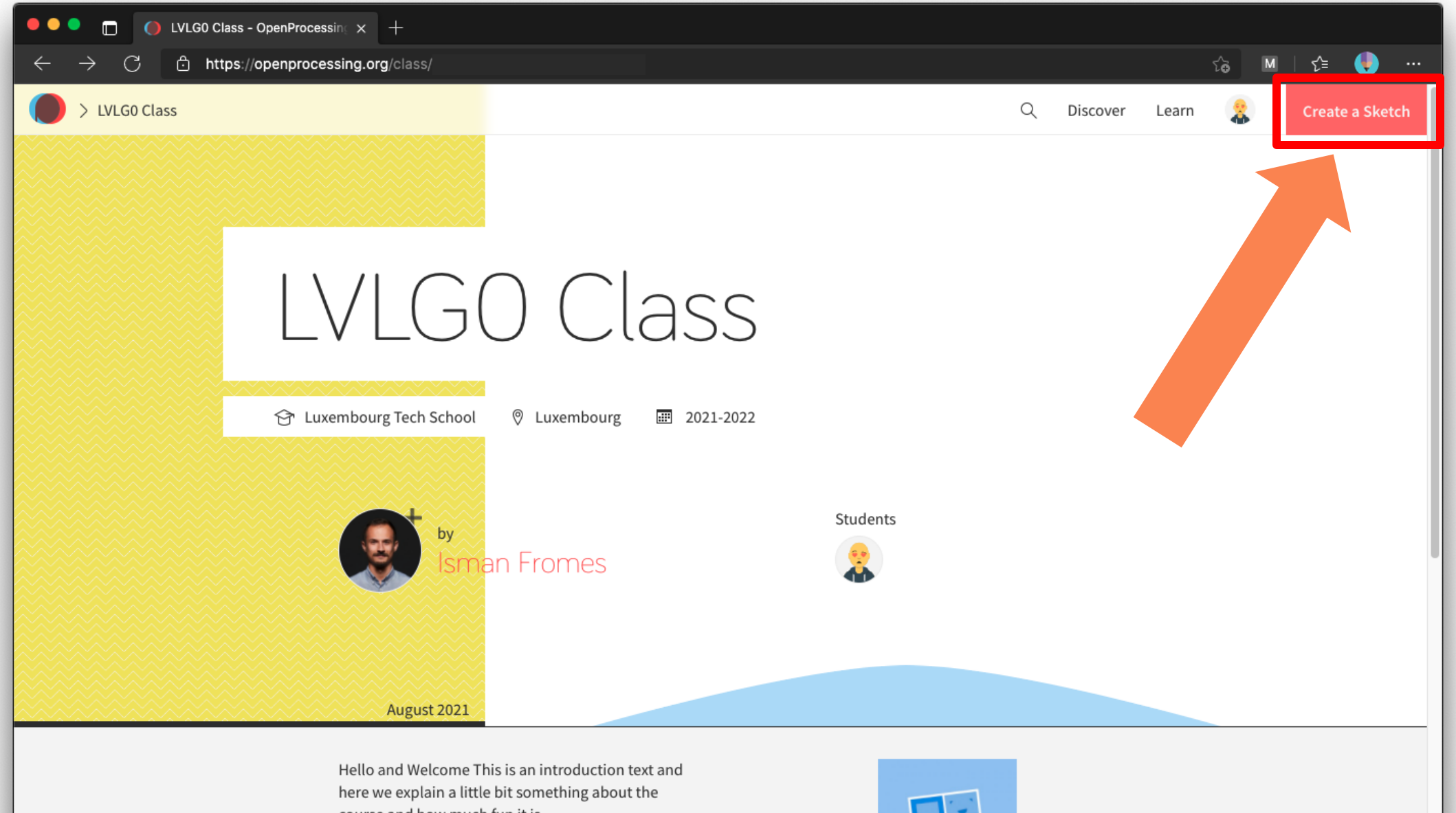
ASCII art is a design technique that consists of pictures pieced together from the 95 printable (from a total of 128) characters.

Let’s continue with Creative Coding

Open OpenProcessing in your browser

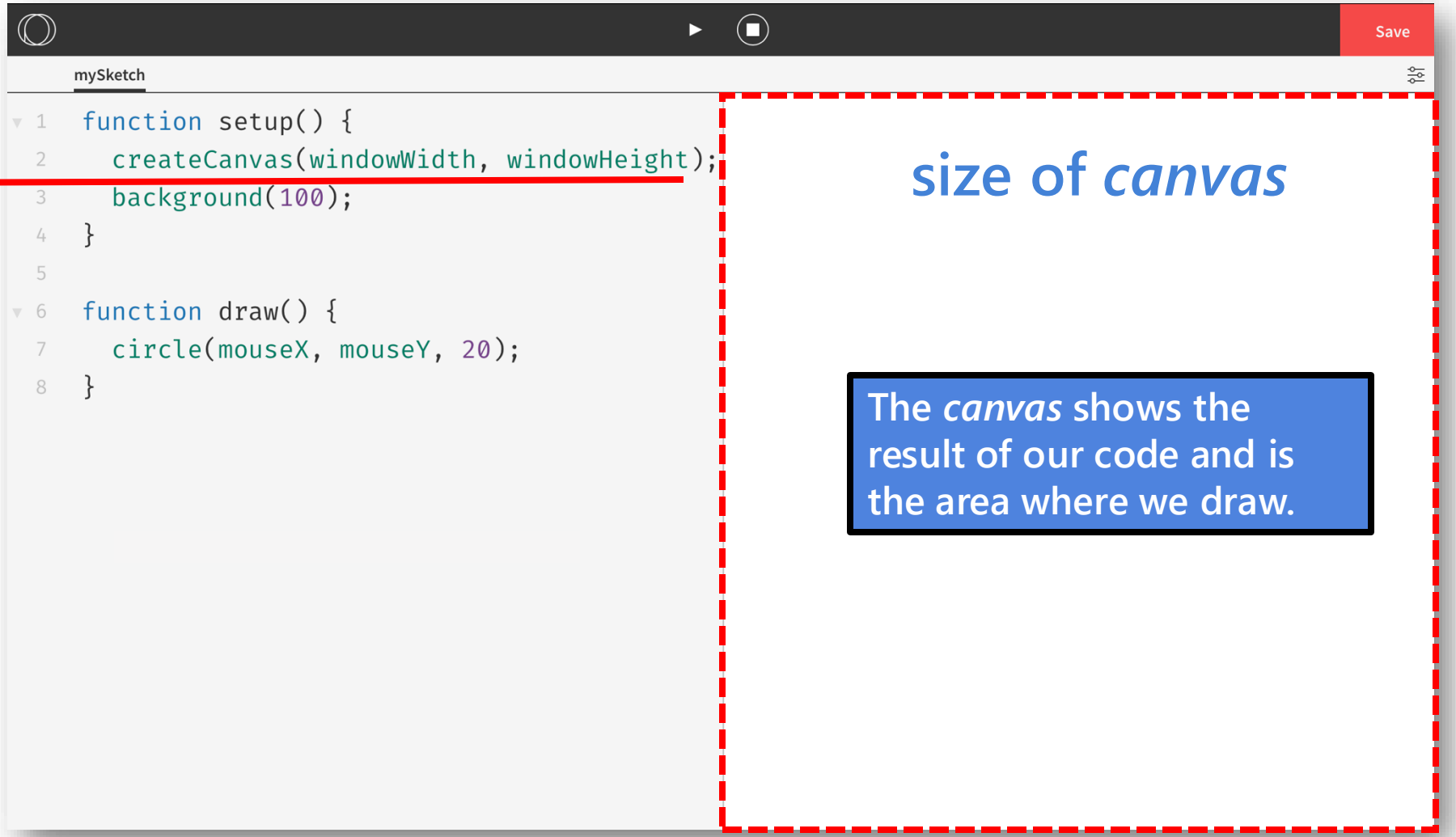


Click on “Create a Sketch”



Show change of canvas size

createCanvas(…)
defines the size of our canvas

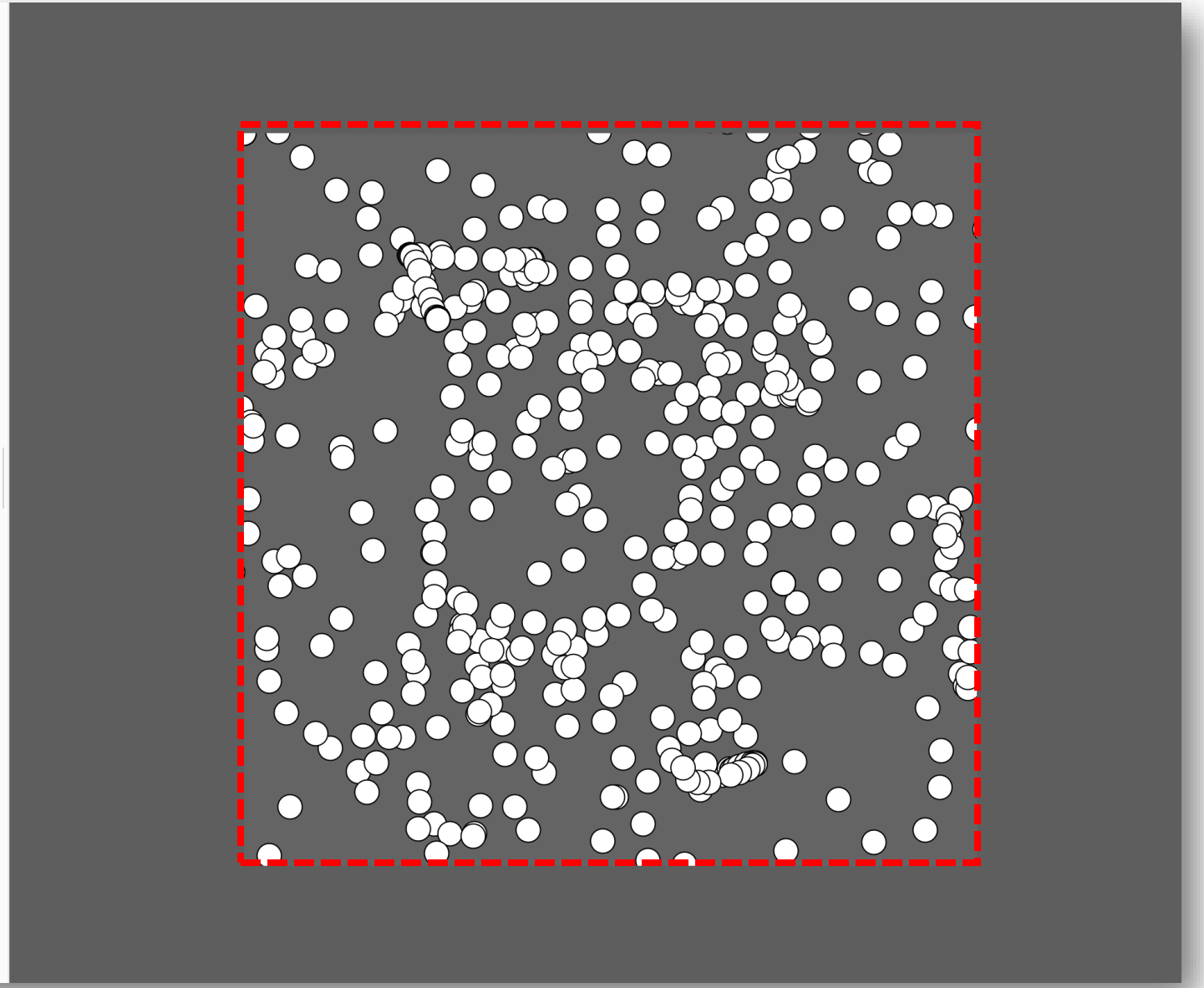


Show change of canvas size

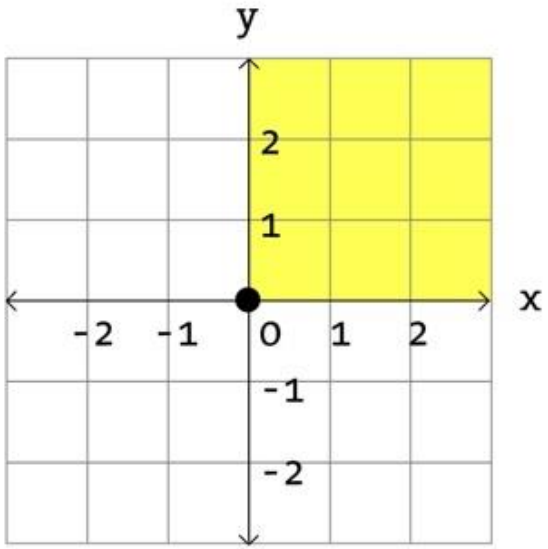
```
1  function setup() {
2    createCanvas(600, 600);
3    background(100);
4  }
5
6  function draw() {
7    circle(mouseX, mouseY, 20);
8  }
```

createCanvas(…) defines the size of our canvas

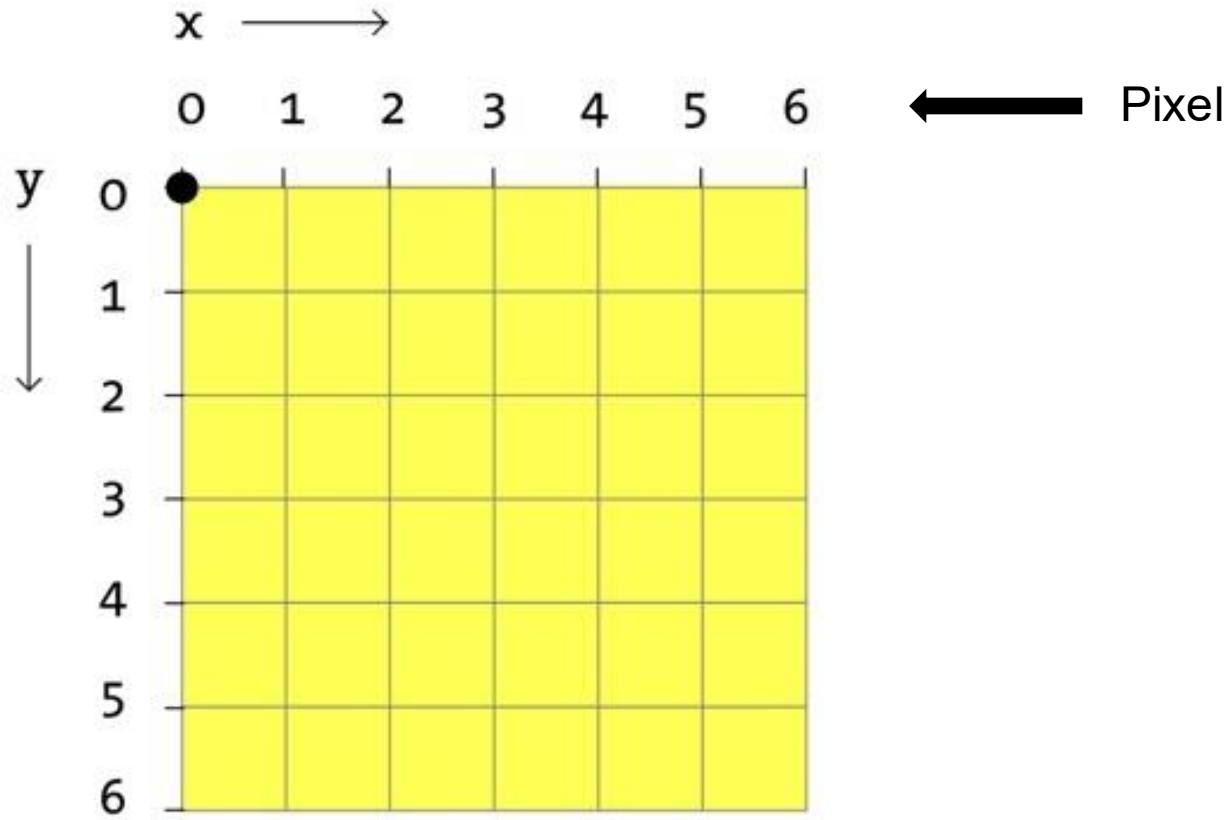
The *canvas* shows the result of our code and is the area where we draw.



Computer Coordinate System



Mathematical

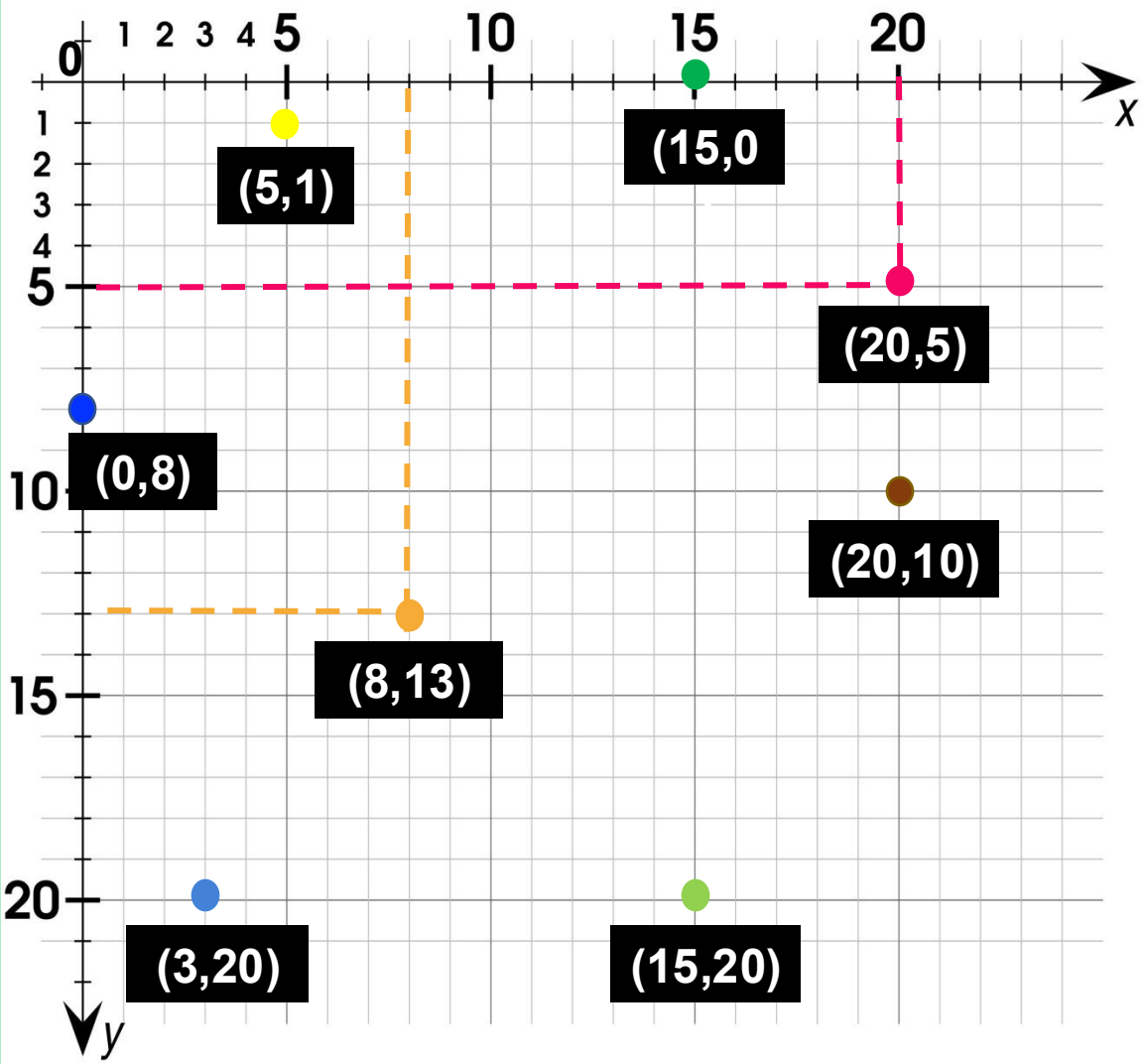


Pixel

Computer

p5*

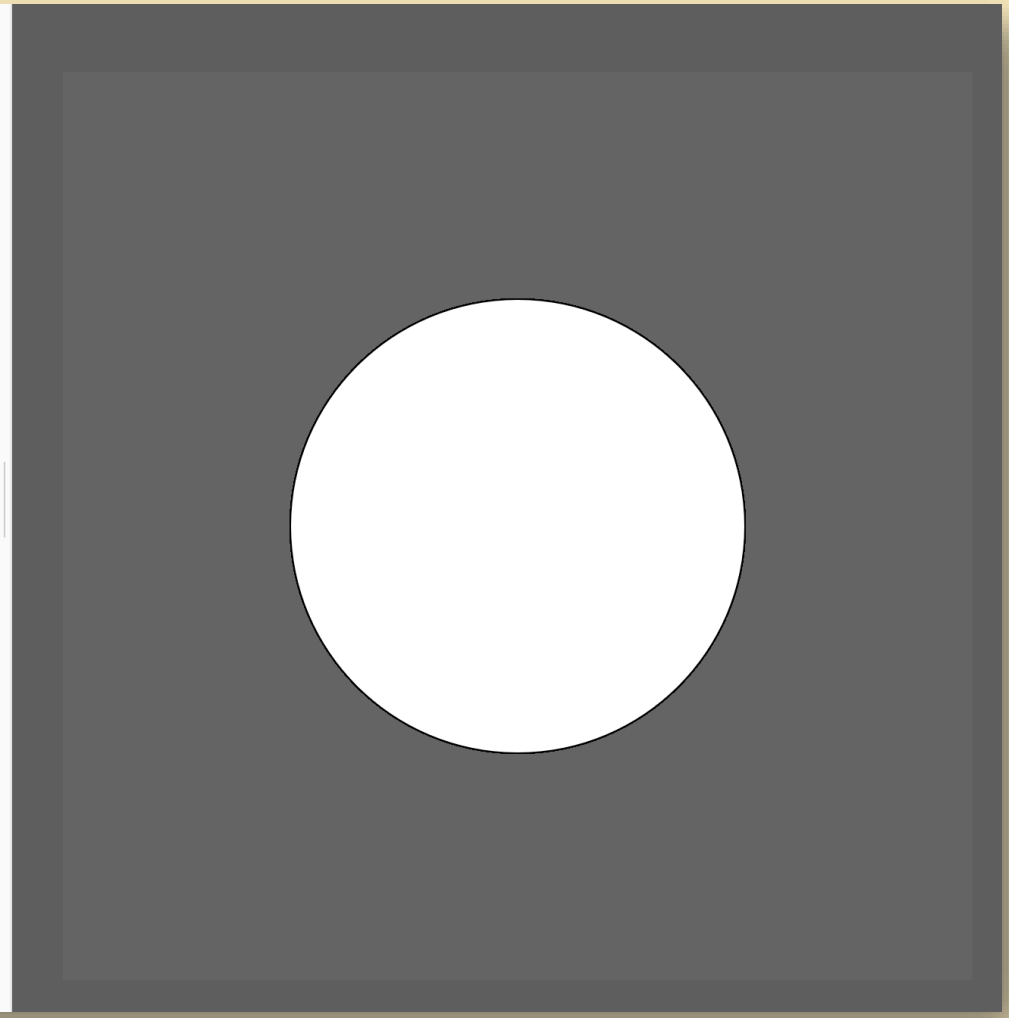
X and Y Coordinates



Can you draw a circle in the middle of the canvas?

Hint:
Have a look at the size of the canvas.

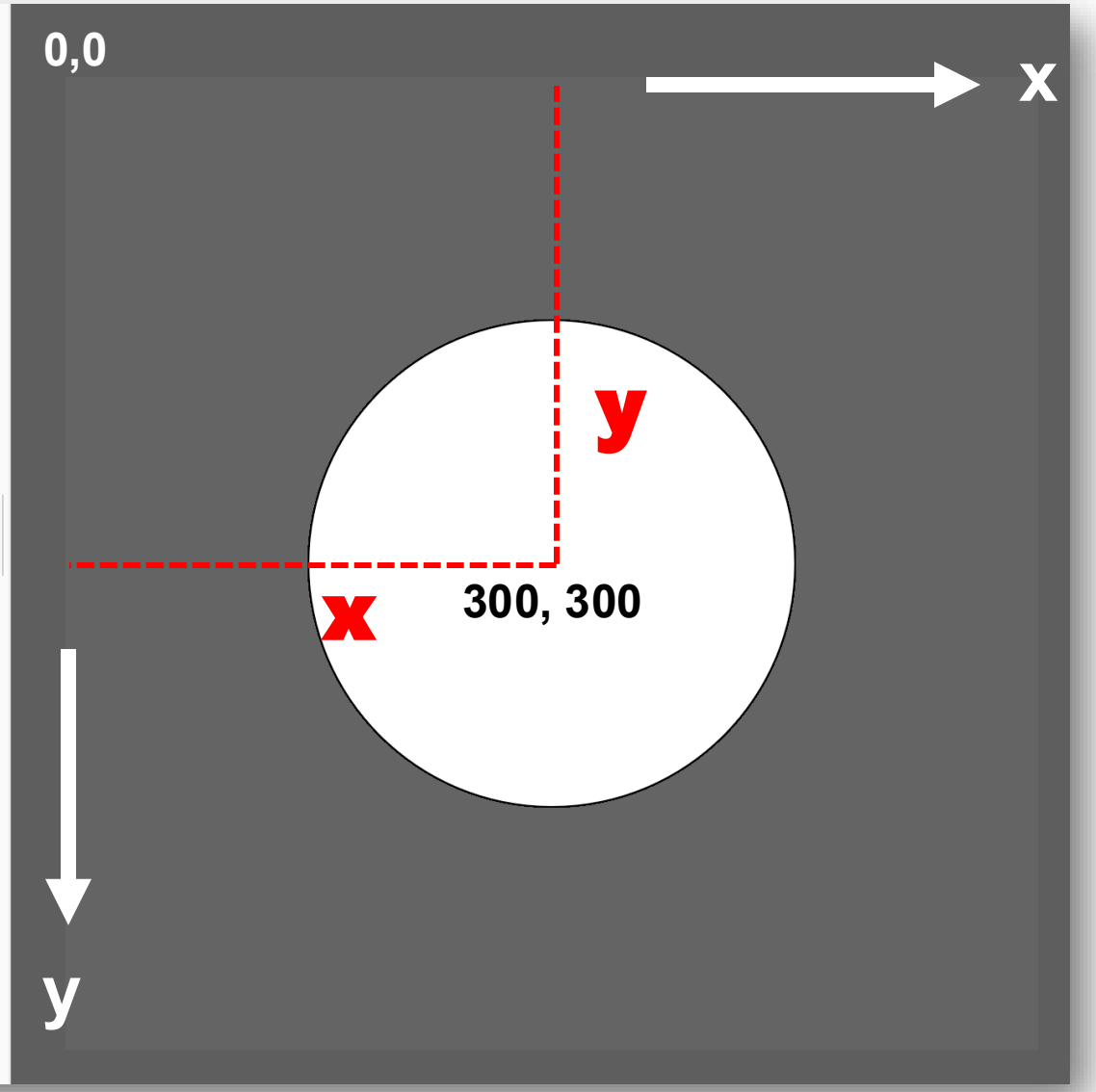
```
1  function setup() {
2    createCanvas(600, 600);
3    background(100);
4  }
5
6  function draw() {
7    circle([redacted], [redacted], 300);
8  }
```



Circle position

```
1  function setup() {
2    createCanvas(600, 600);
3    background(100);
4  }
5
6  function draw() {
7    circle(300, 300, 300);
8  }
```

x y

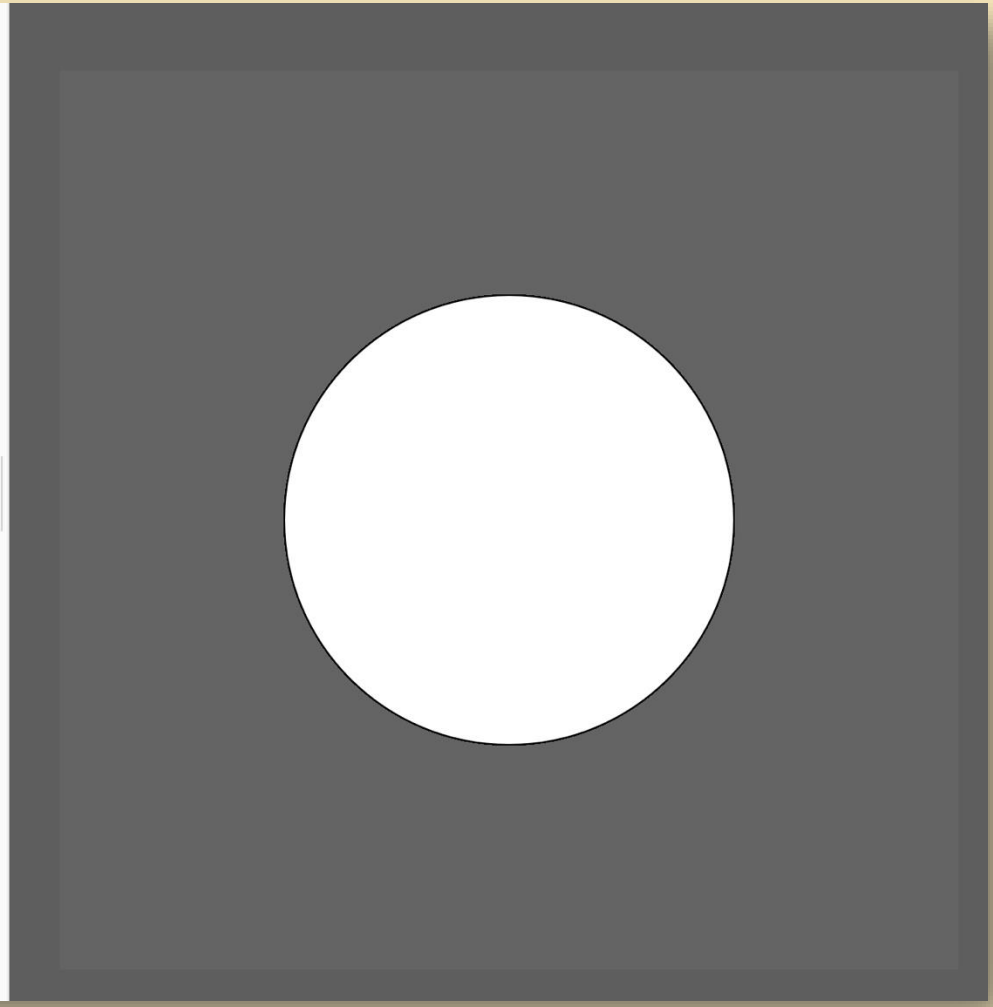


The first two values - ***x*** and ***y*** – show us WHERE exactly the circle is drawn on the *canvas*.

Which value do you need to change to move it up or down?

```
1  function setup() {
2    createCanvas(600, 600);
3    background(100);
4  }
5
6  function draw() {
7    circle(300, 300, 300);
8  }
```

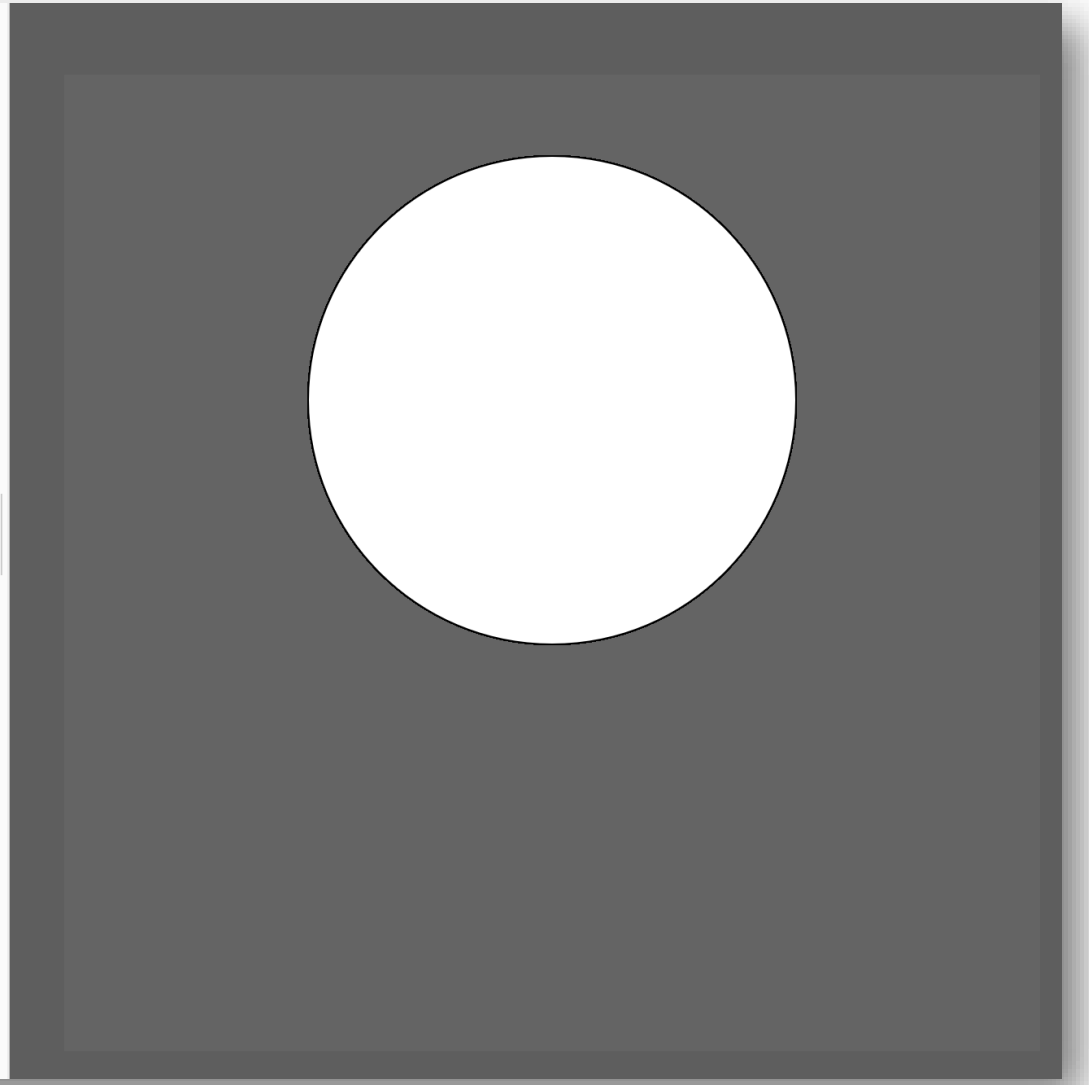
x y



Change y-value to move up/down

```
1  function setup() {
2    createCanvas(600, 600);
3    background(100);
4  }
5
6  function draw() {
7    circle(300, 200, 300);
8  }
```

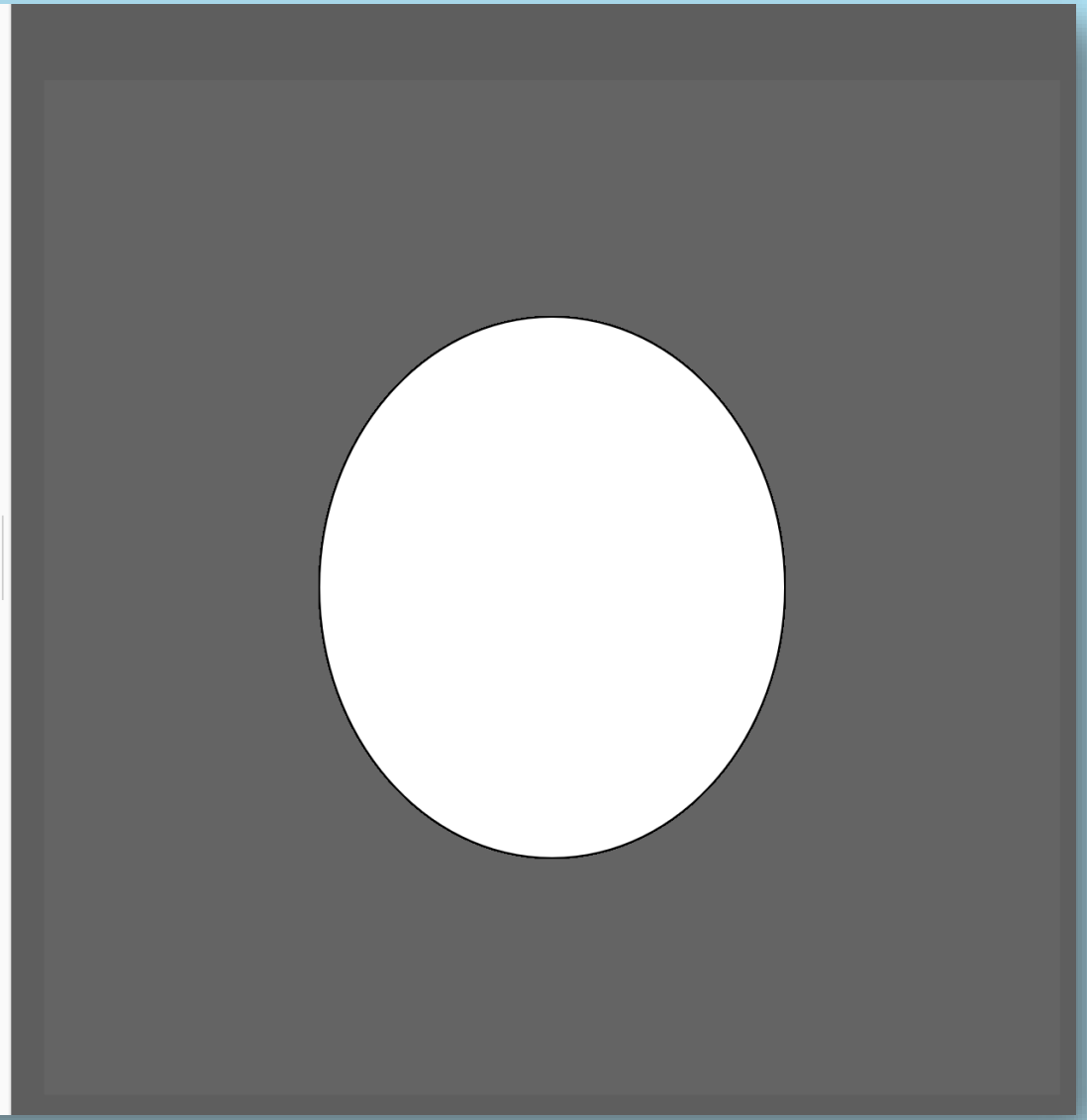
y



Let's make it a face

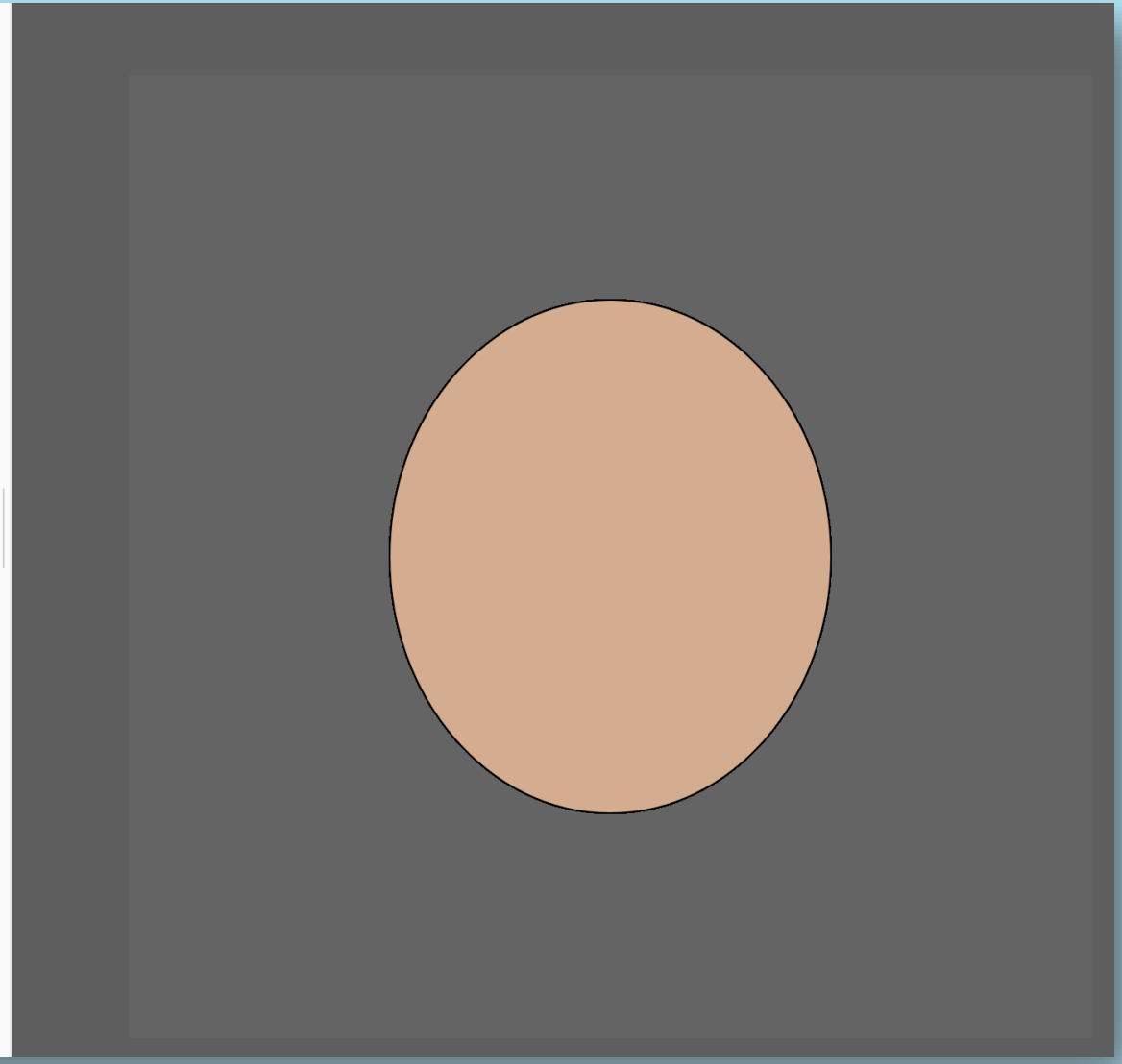
```
1  function setup() {
2    createCanvas(600, 600);
3    background(100);
4  }
5
6  function draw() {
7    ellipse(300, 300, 275, 320);
8  }
```

width height



Give it a colour

```
1  function setup() {
2    createCanvas(600, 600);
3    background(100);
4  }
5
6  function draw() {
7    fill(220, 170, 140);
8    ellipse(300, 300, 275, 320);
9  }
```



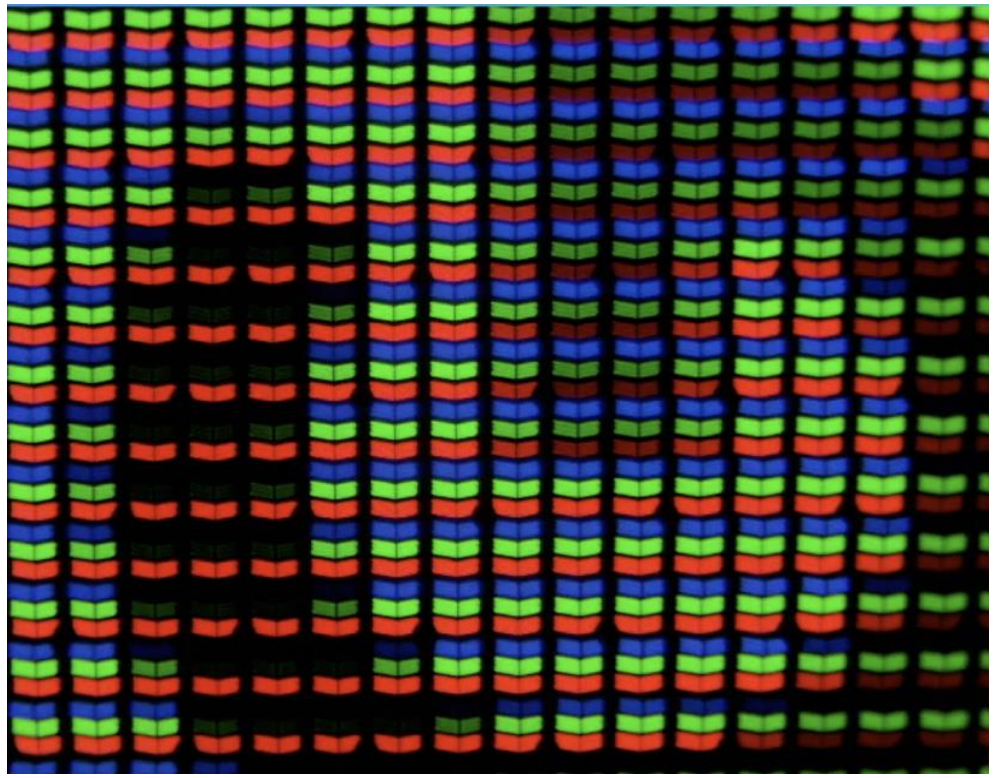
With ***fill(..., ..., ...);***
we can define the filling colour of the circle.

What do the 3 numbers mean?

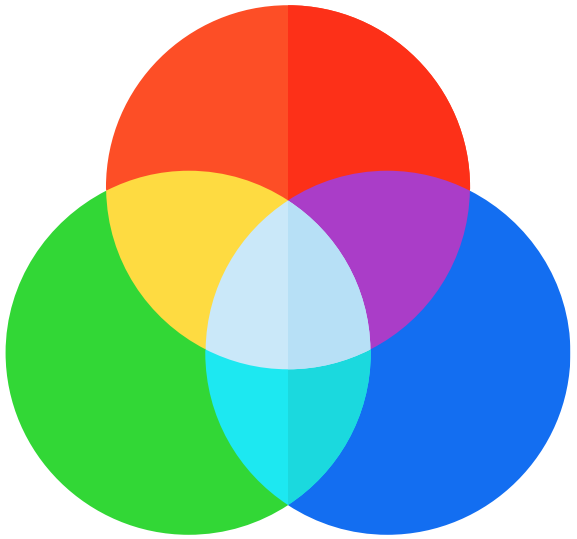
Seeing Pixels

When you hold a screen under a microscope you can see tiny little dots of lights called **pixels**. Each pixel is made of 3 different kind of colours: **red**, **green** and **blue**.

Each tiny pixel can shift between different **brightness**: from dark (0) to bright (255). From afar our eyes see an image with different kinds of colours.



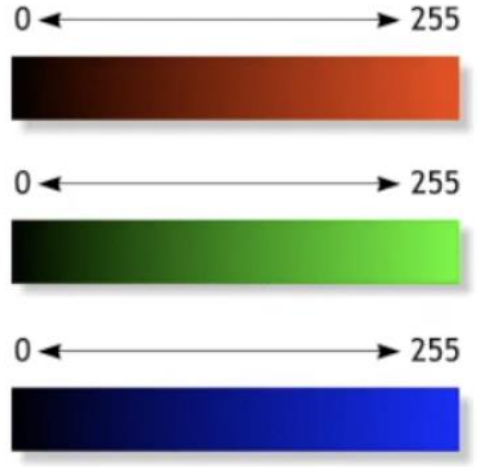
RGB - Adding Color



RGB (Red, Green, Blue) is the color space for digital images

We start with **0 = black**

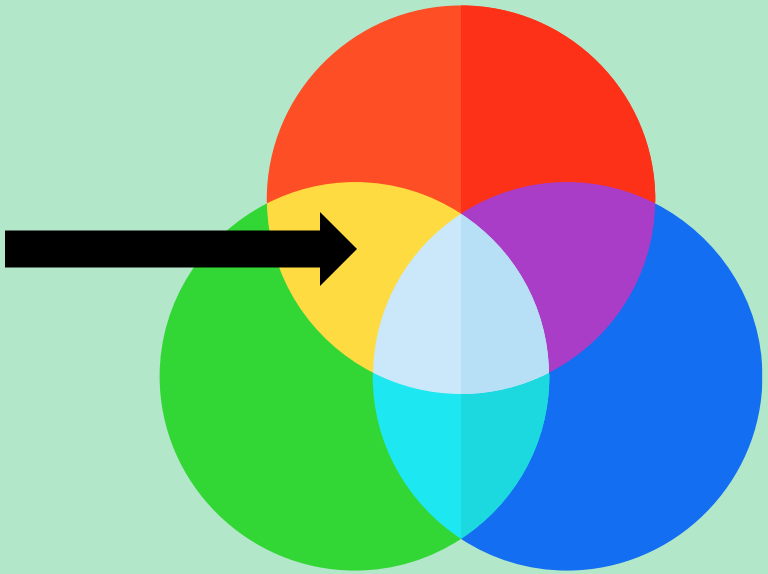
We can add **red**, **green** and **blue** from **0 to 255**



Red Green Blue

If we **add 255** for **red**, **green** and **blue** = **white**

RGB - Yellow



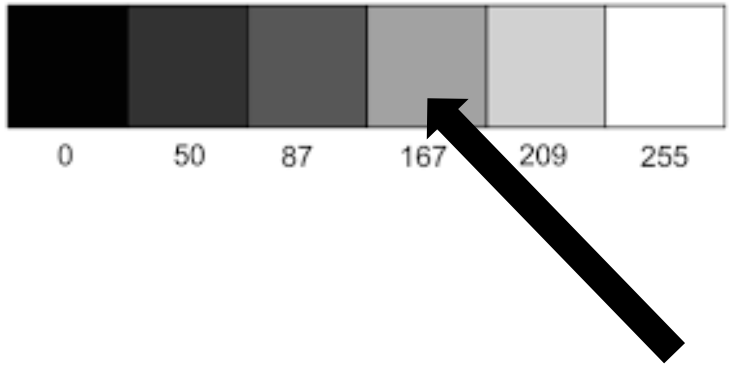
Rot Grün Blau

How do we get yellow ?

Yellow

```
fill(255, 255, 0);
```

White – Grey - Black

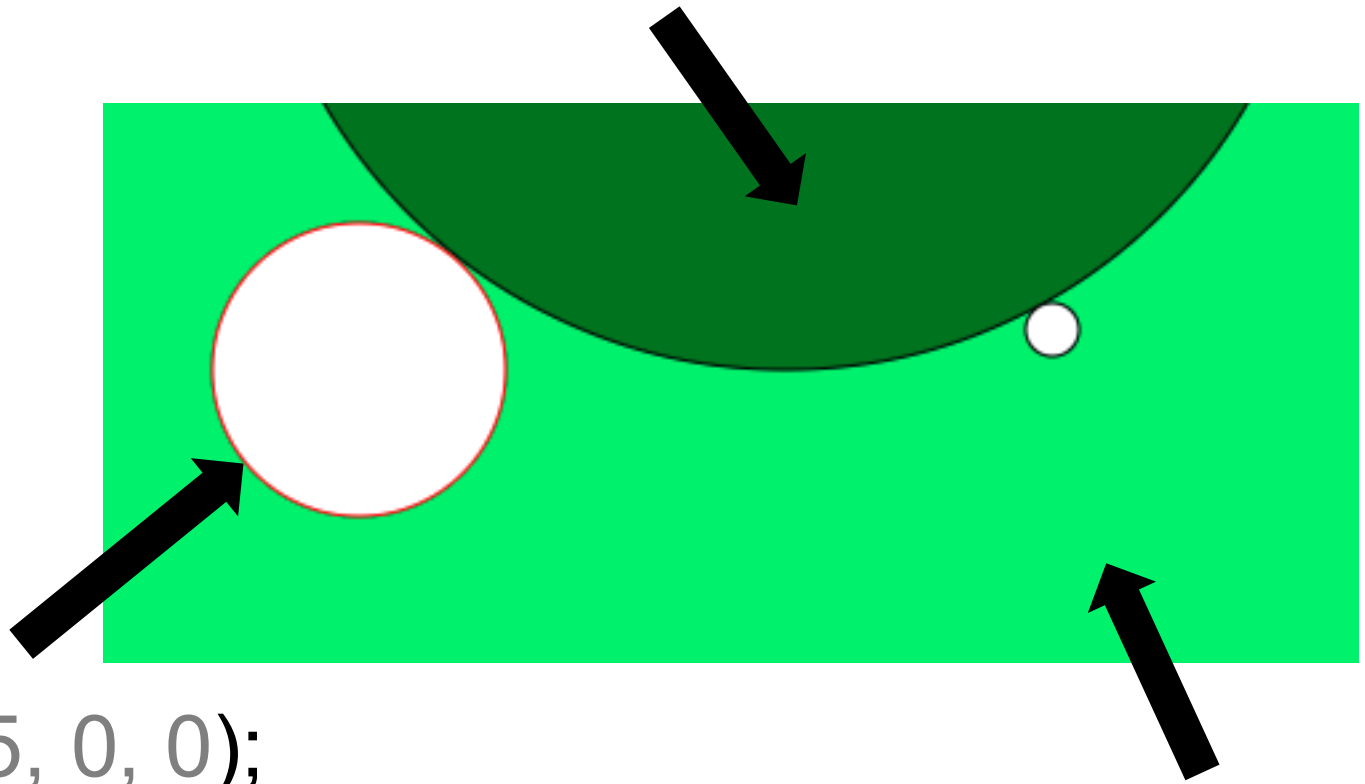


If we only use one value, we get different shades of grey.

```
fill(167);
```

Colour Functions

fill(30, 110, 45);

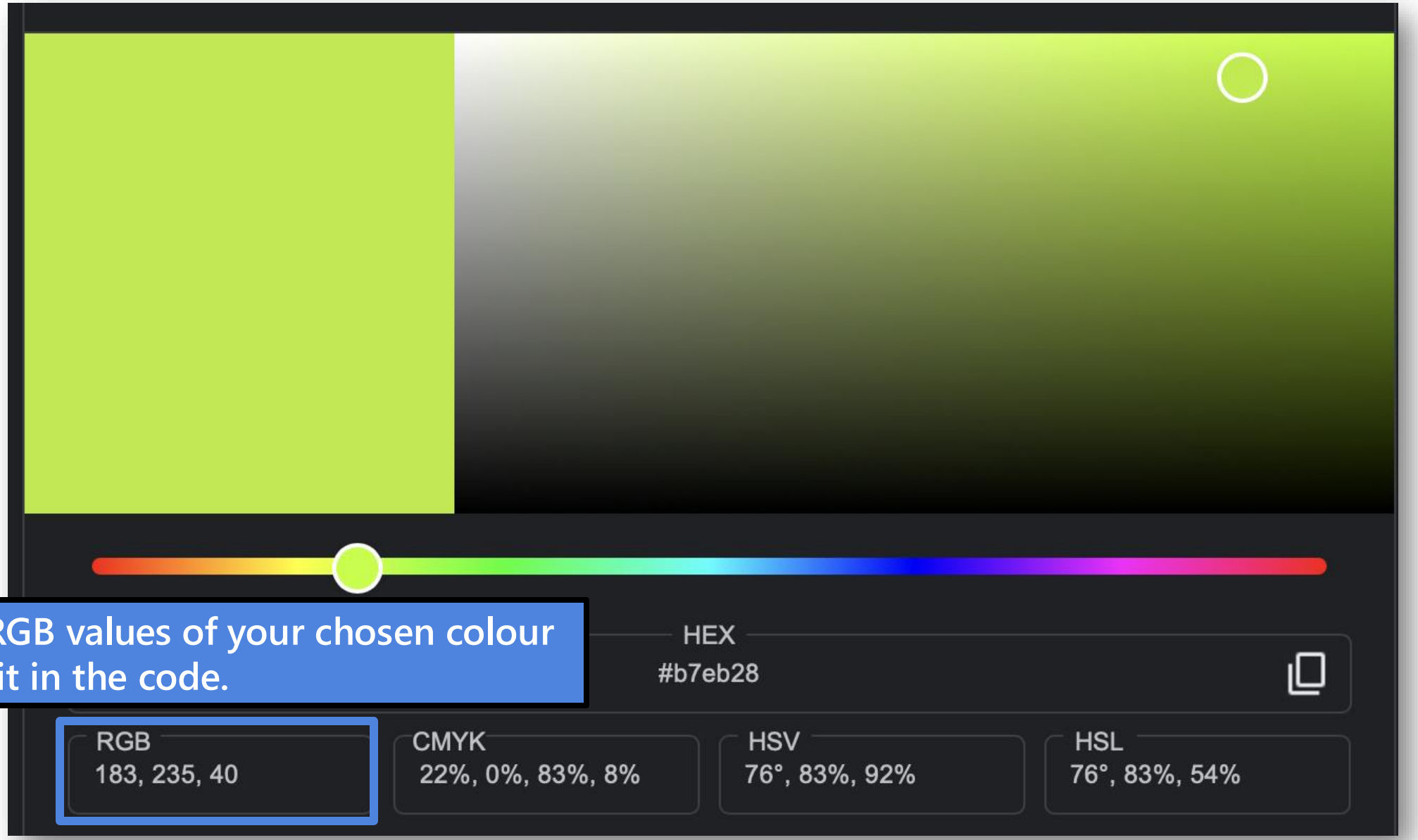


stroke(255, 0, 0);

background(85, 250, 110);

Where do we find new colours?

Colour Picker



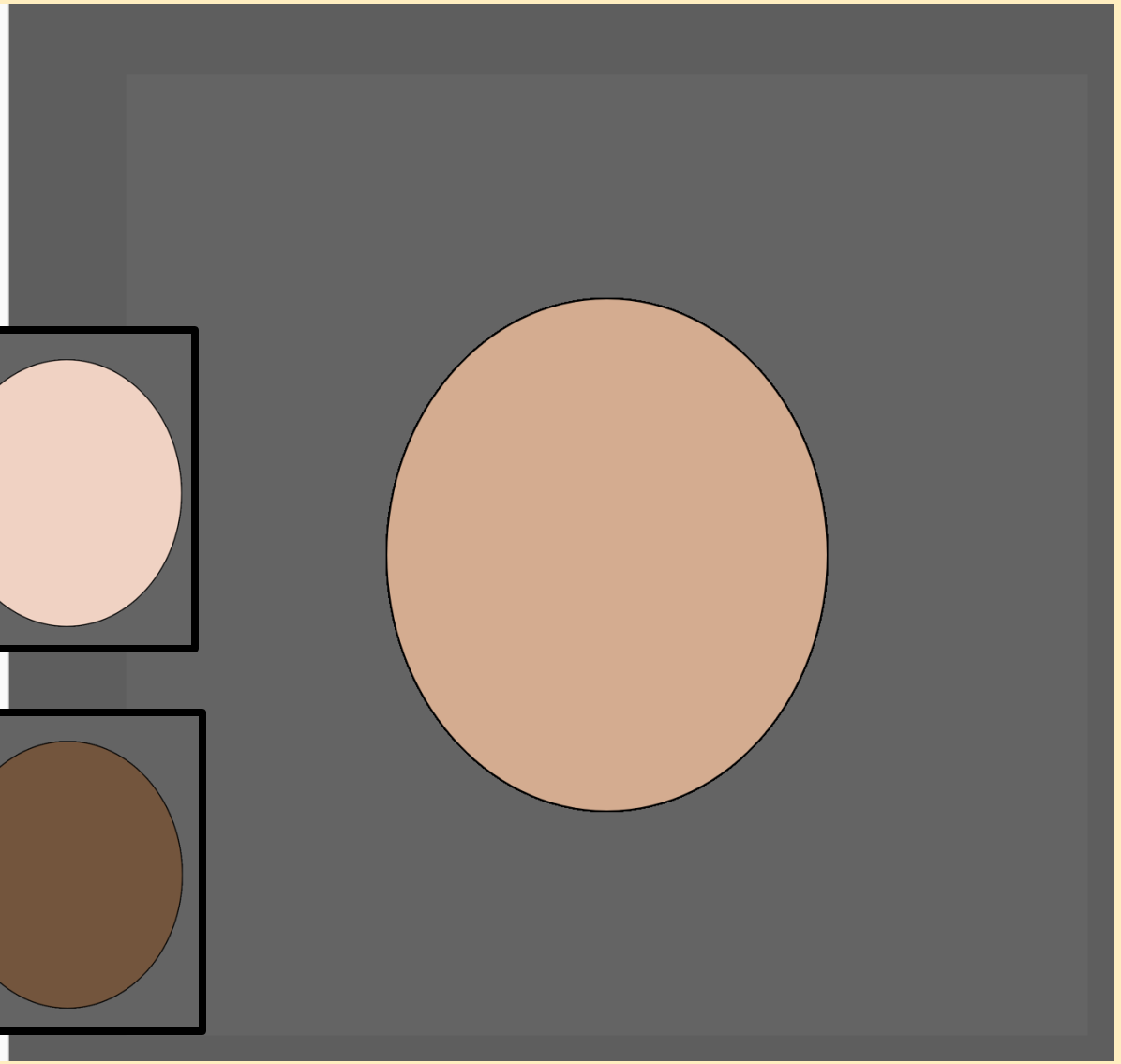
Copy the RGB values of your chosen colour and paste it in the code.

Change the colour of the face

```
1  function setup() {
2    createCanvas(600, 600);
3    background(100);
4  }
5
6  function draw() {
7    fill(220, 170, 140);
8    ellipse(300, 300, 275, 320);
9  }
10
11
```

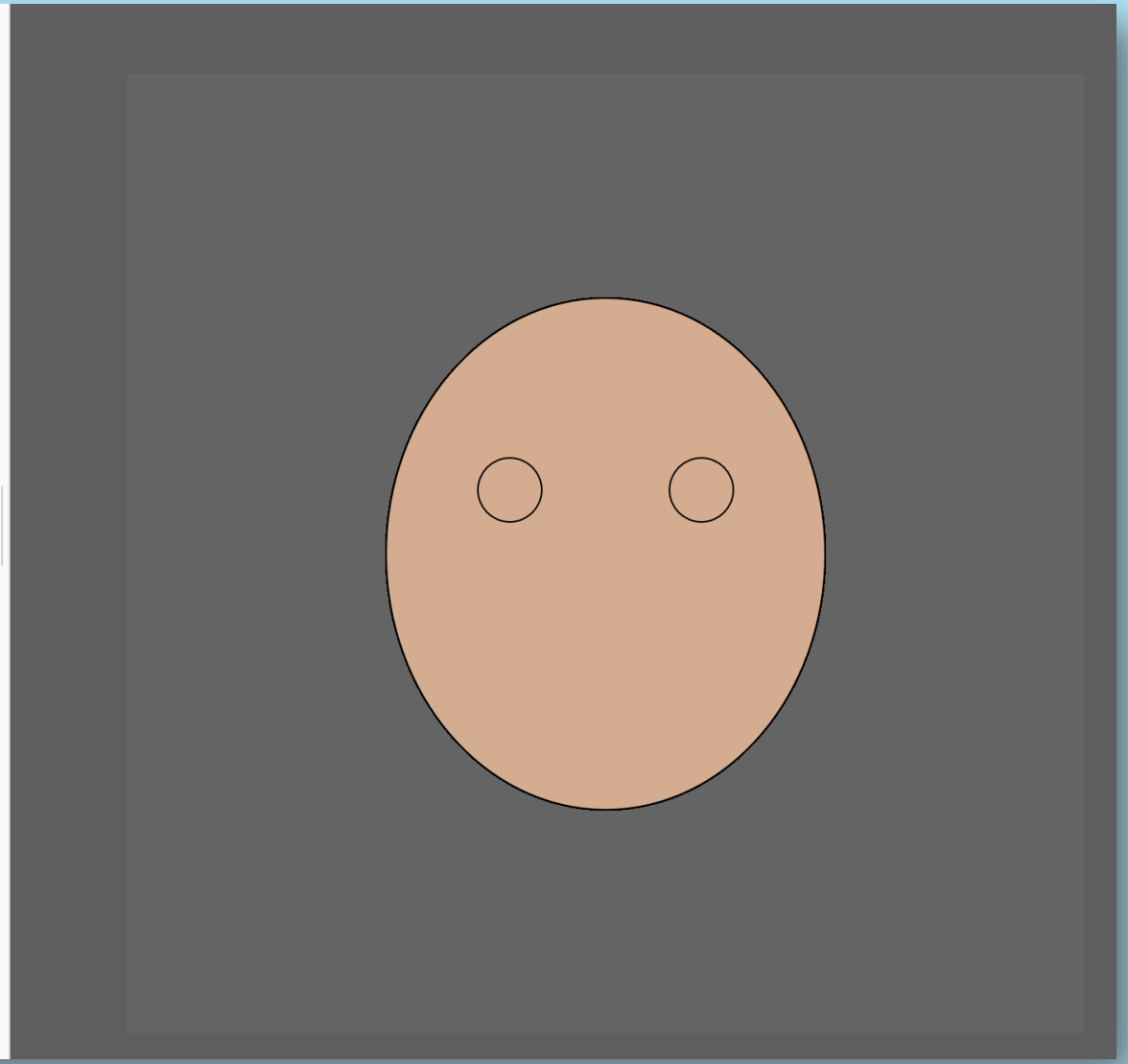
fill(240, 210, 195);

fill(115, 85, 60);



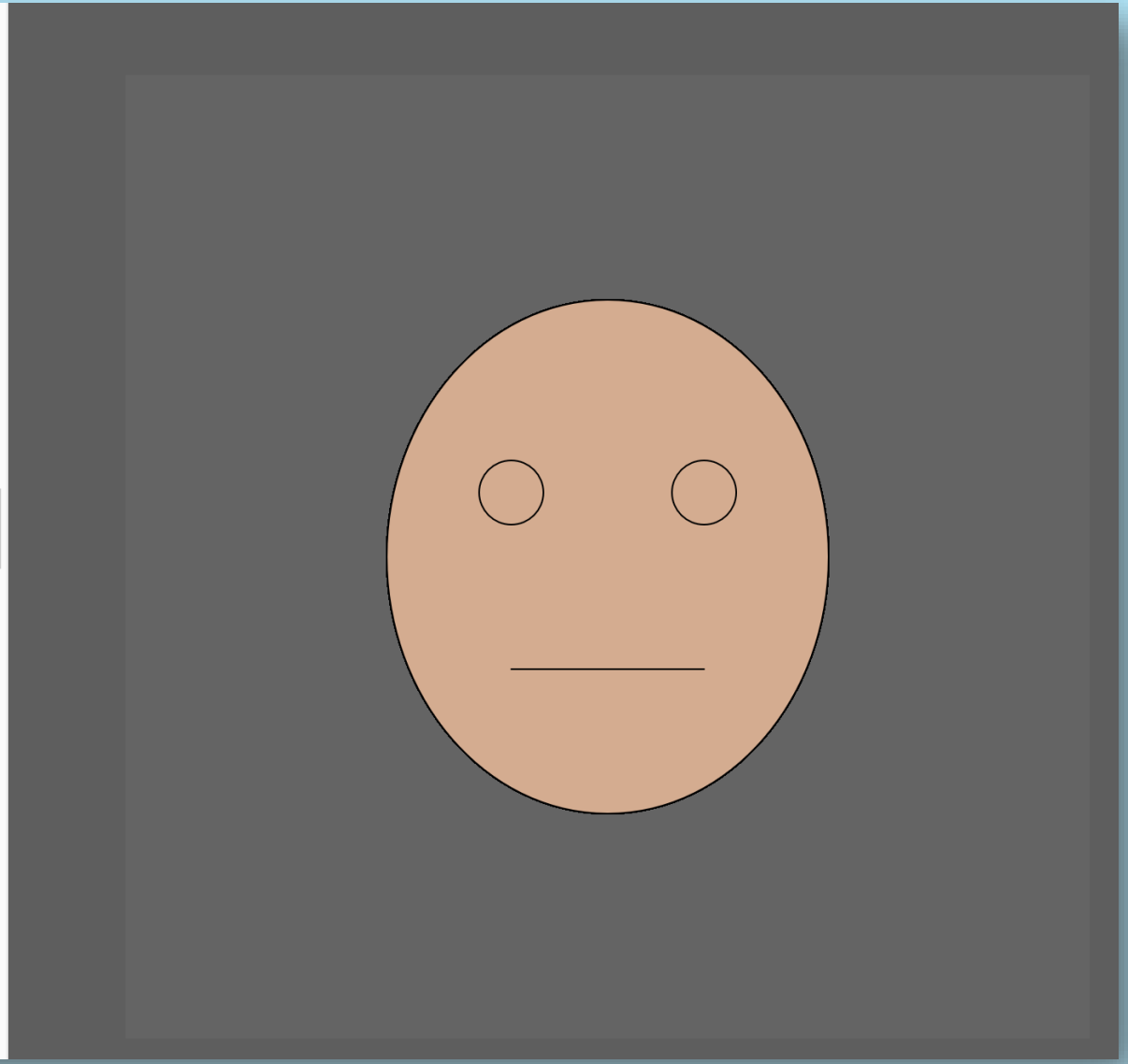
Add eyes

```
1  function setup() {
2    createCanvas(600, 600);
3    background(100);
4  }
5
6  function draw() {
7    fill(220, 170, 140);
8    ellipse(300, 300, 275, 320);
9
10   ellipse(240, 260, 40, 40);
11   ellipse(360, 260, 40, 40);
12 }
13
14
```



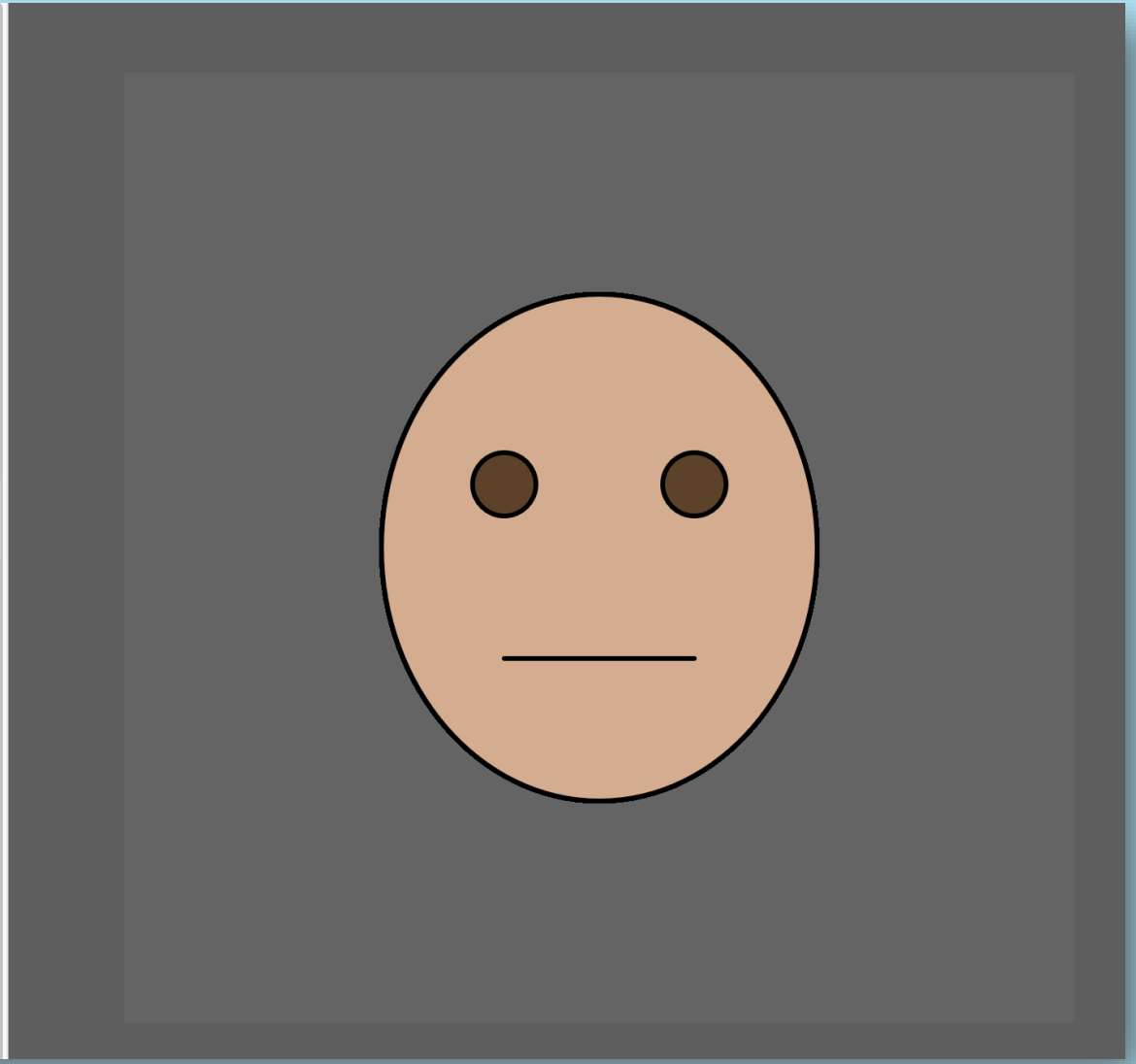
Add the mouth

```
1  function setup() {
2    createCanvas(600, 600);
3    background(100);
4  }
5
6  function draw() {
7    fill(220, 170, 140);
8    ellipse(300, 300, 275, 320);
9
10   ellipse(240, 260, 40, 40);
11   ellipse(360, 260, 40, 40);
12
13   line(240, 370, 360, 370);
14 }
15
16
```

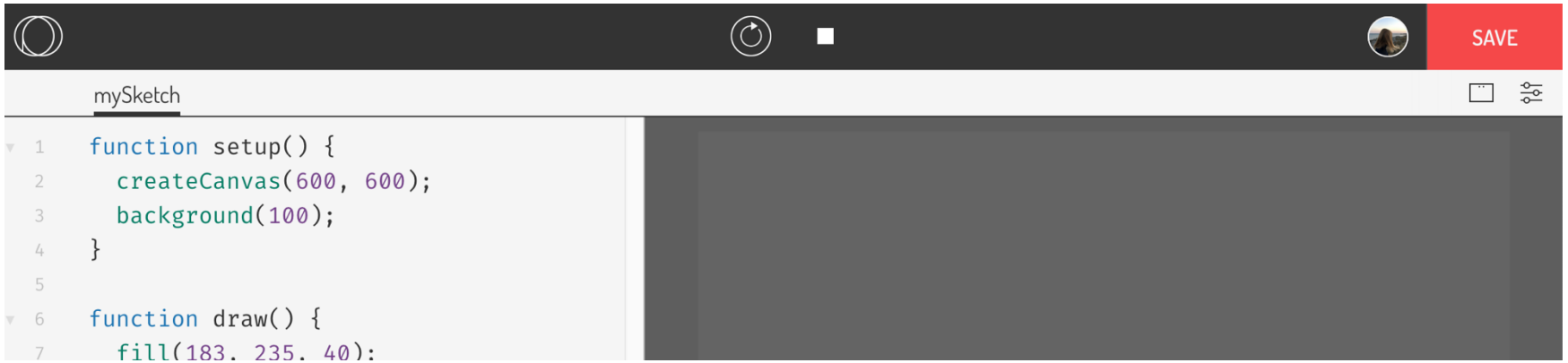


Add colour & thickness & comments

```
1  function setup() {
2    createCanvas(600, 600);
3    background(100);
4  }
5
6  function draw() {
7    fill(220, 170, 140);
8    ellipse(300, 300, 275, 320);
9
10   // Eyes
11   fill(95, 65, 35);
12   ellipse(240, 260, 40, 40);
13   ellipse(360, 260, 40, 40);
14
15   // Mouth
16   strokeWeight(3);
17   line(240, 370, 360, 370);
18 }
19
20
```



Save

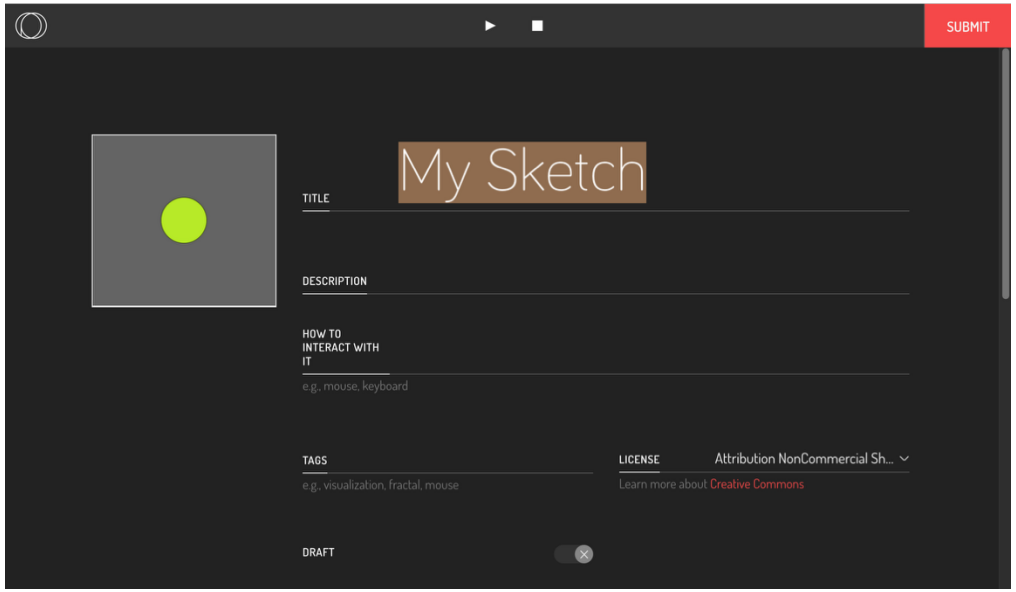


1

Click on ***SAVE***

2

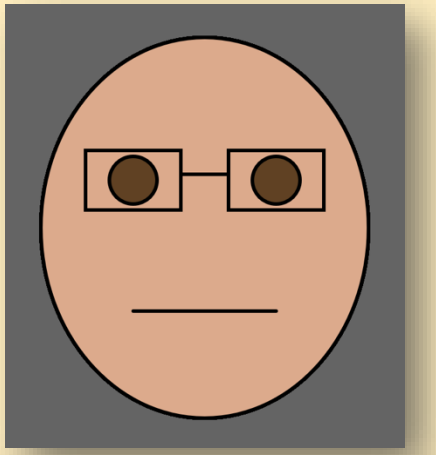
Give the sketch a new title instead of *„My Sketch“* e.g. *„Face“*



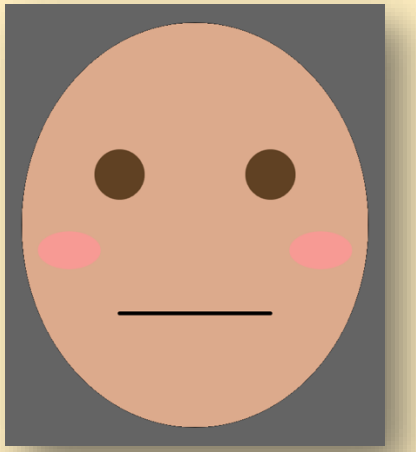
3

Click on ***SUBMIT***

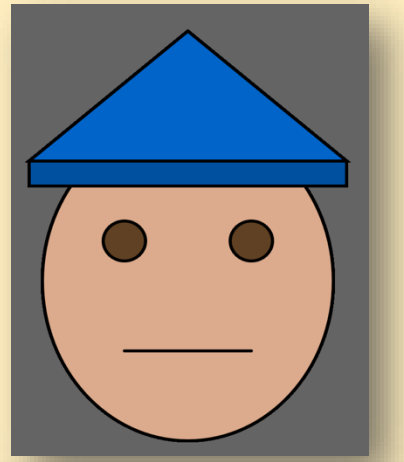
Personalize your face



```
    line(240, 370, 360, 370);

    // Glasses
    noFill();
    strokeWeight(3);
    // Frames
    rect(200, 235, 80, 50);
    rect(320, 235, 80, 50);
    // Bridge
    line(280, 255, 320, 255);
}
```



```
    // Mouth
    strokeWeight(3);
    stroke(0);
    line(240, 370, 360, 370);

    // Blush
    fill(255, 150, 150, 200);
    noStroke();
    ellipse(200, 320, 50, 30);
    ellipse(400, 320, 50, 30);
}
```



```
    // Mouth
    strokeWeight(3);
    line(240, 370, 360, 370);

    // Hat
    fill(0, 100, 200);
    triangle(150, 180, 450, 180, 300, 50);

    fill(0, 80, 160);
    rect(150, 180, 300, 25);
}
```

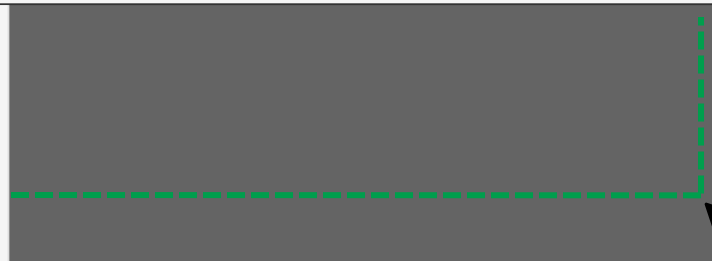
Hint: How to find the right position?

mySketch

```
1  function setup() {
2    createCanvas(windowWidth, windowHeight);
3    background(100);
4  }
5
6  function draw() {
7    //circle(mouseX, mouseY, 20);
8  }
9
10 function mouseClicked(){
11
12 print(mouseX)
13 print(mouseY)
14 print("---")
15 }
```

Die *mouseClicked()* Funktion wird nur aufgerufen, wenn man mit der Maus klickt.

the *print()* function allows us to write messages to the console.

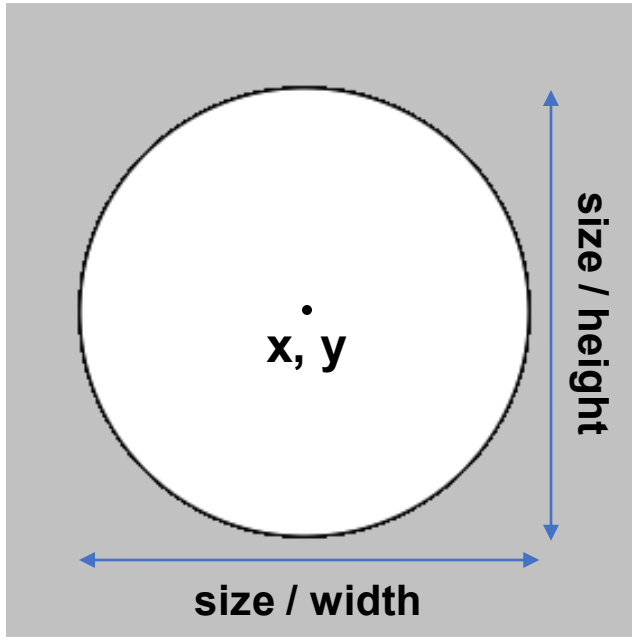


mouseX and *mouseY* are variables that contain the X and Y position of our mouse cursor on the canvas.

```
---
437
268.75
---
112
99.75
---
```

Basic Shapes

Circle/Ellipse

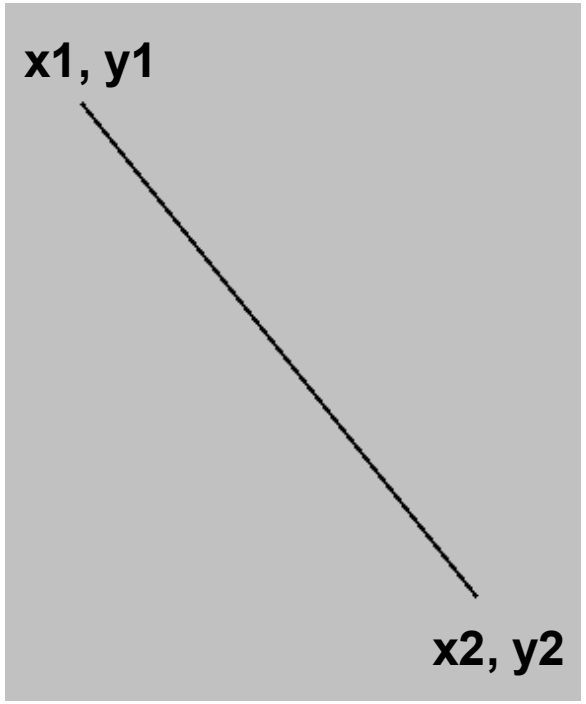



```
function setup() {
  createCanvas(windowWidth, windowHeight);
  background(200);
}

function draw() {
  circle(150,150,200);
  ellipse(150,150,200,200);
}
```

```
circle(x, y, size);
```

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

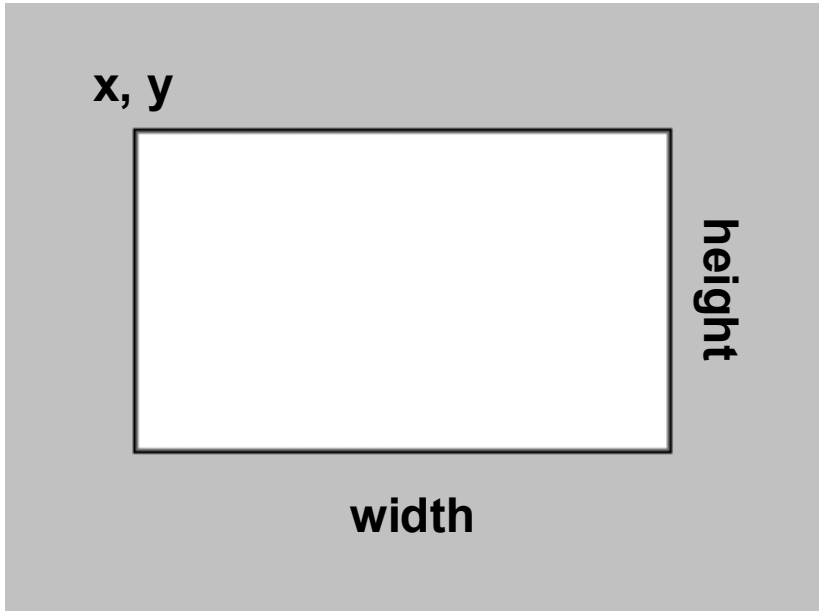
Line




```
function setup() {
  createCanvas(windowWidth, windowHeight);
  background(200);
}

function draw() {
  line(50,70,250,320);
}
```

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

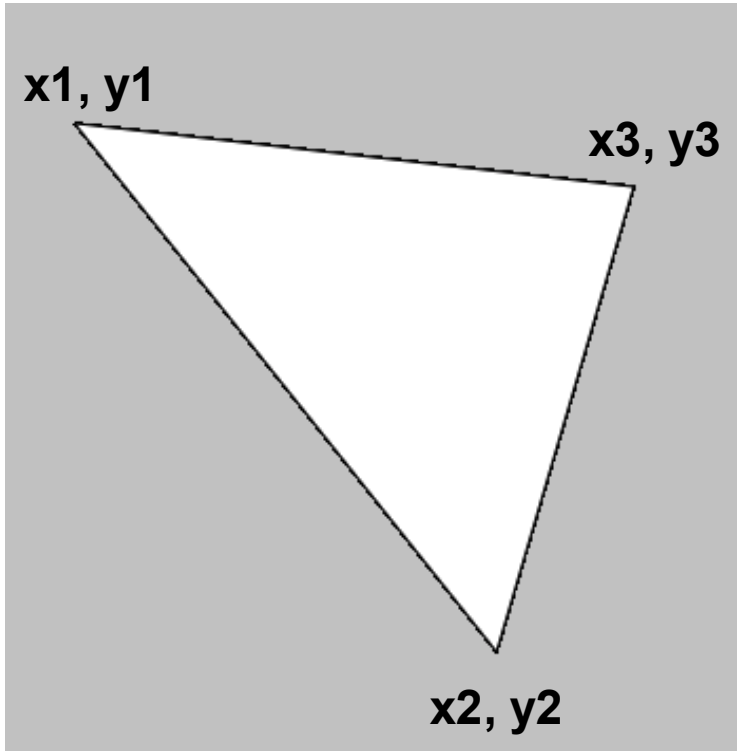
Rectangle




```
function setup() {
  createCanvas(windowWidth, windowHeight);
  background(200);
}

function draw() {
  rect(50,50,200,120);
}
```

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

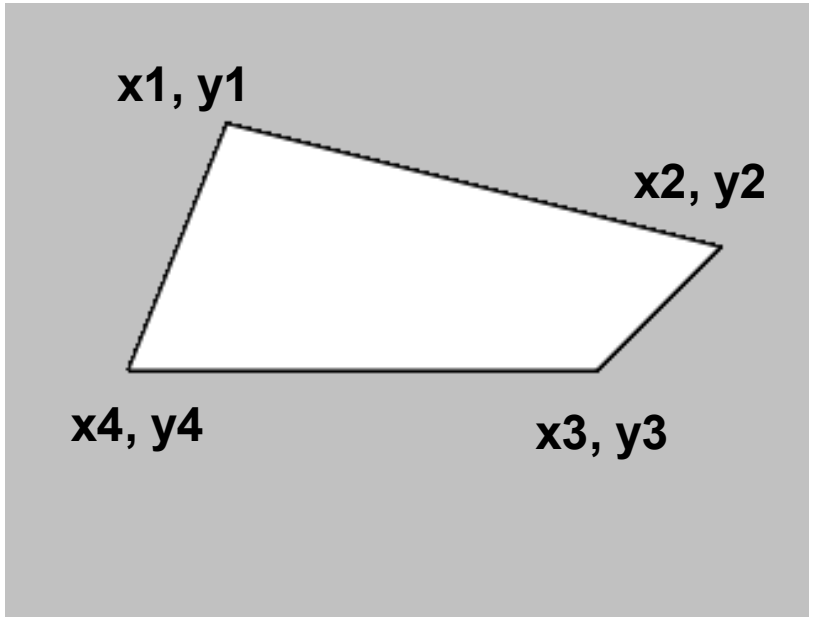
Triangle




```
function setup() {
  createCanvas(windowWidth, windowHeight);
  background(200);
}

function draw() {
  triangle(50,70,250,320,315,100);
}
```

```
triangle(x1, y1, x2, y2, x3, y3);
```

Quadrilateral




```
function setup() {
  createCanvas(windowWidth, windowHeight);
  background(200);
}

function draw() {
  quad(200,50,400,100,350,150,160, 150);
}
```

```
quad(x1, y1, x2, y2, x3, y3, x4, y4);
```

Benutzerdefinierte Formen mit *vertex()*

sketch.js *

```
function setup() {
  createCanvas(400, 400);
  background(200);
}

function draw() {
  stroke(255, 0, 0);
  strokeWeight(10);
  fill(255, 255, 50);
  beginShape();
  vertex(100, 300);
  vertex(70, 100);
  vertex(130, 180);
  vertex(198, 100);
  vertex(270, 180);
  vertex(330, 100);
  vertex(300, 300);
  endShape();
}
```

Preview

sketch.js *

```
function setup() {
  createCanvas(400, 400);
  background(200);
}

function draw() {
  stroke(255, 0, 0);
  strokeWeight(10);
  fill(255, 255, 50);
  beginShape();
  vertex(100, 300);
  vertex(70, 100);
  vertex(130, 180);
  vertex(198, 100);
  vertex(270, 180);
  vertex(330, 100);
  vertex(300, 300);
  endShape(CLOSE);
}
```

Preview

Nutze:

beginShape();

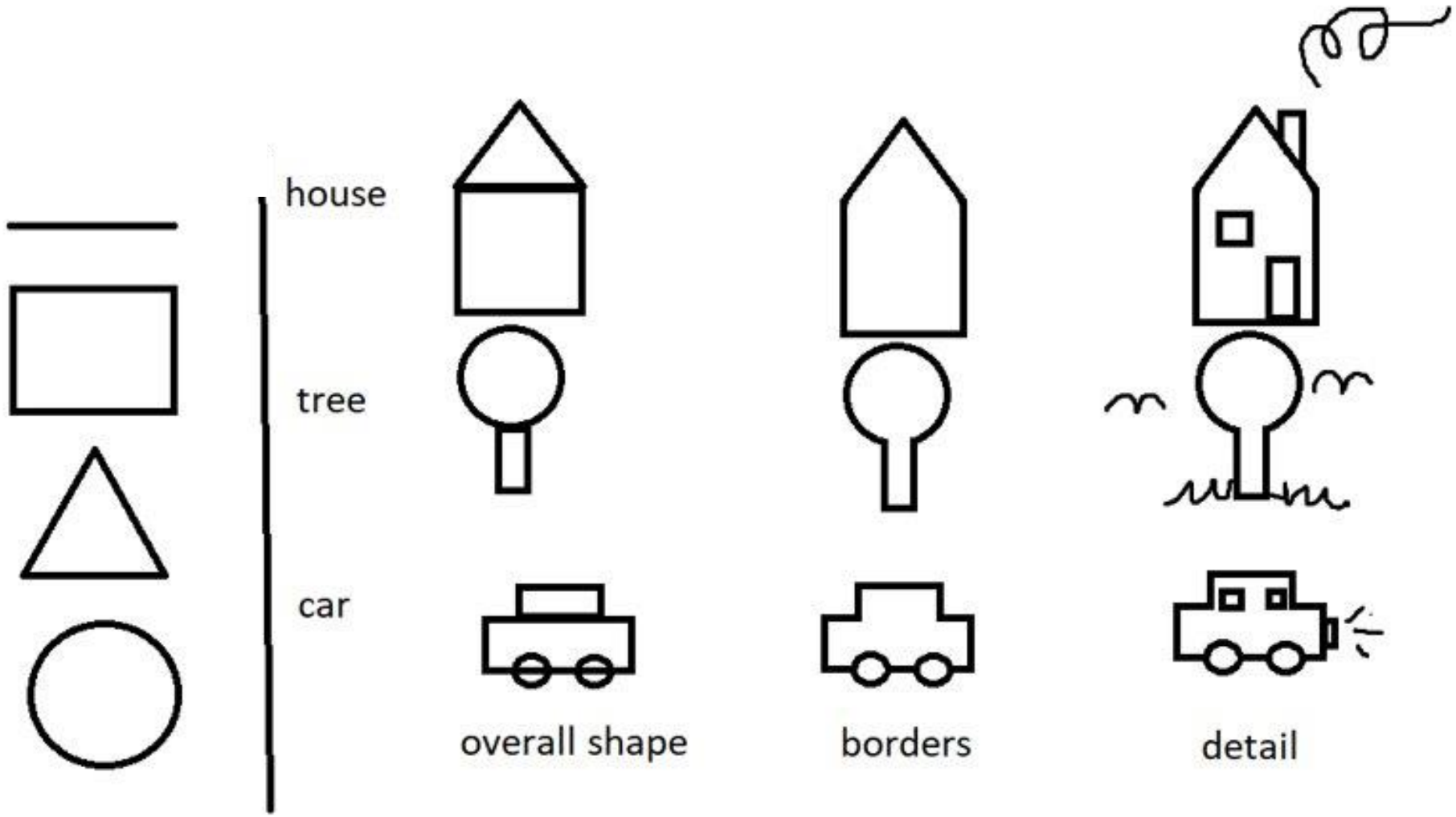
vertex(x,y);

endShape();

Ressourcen:

https://p5js.org/reference/#/p5/vertex

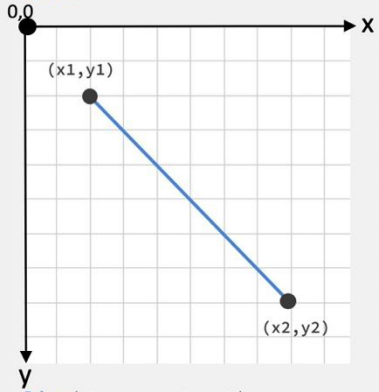
From shapes to figures



Use the Cheat Sheet

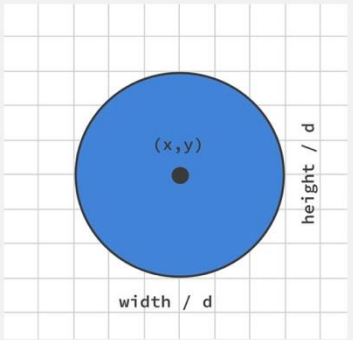
Basic Shapes and Color Functions

line



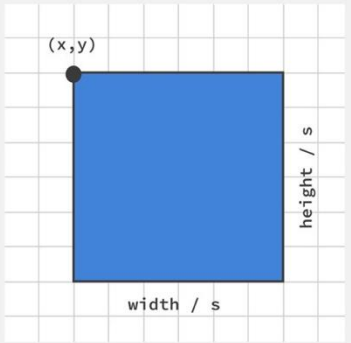

```
line(x1, y1, x2, y2);
```

ellipse / circle



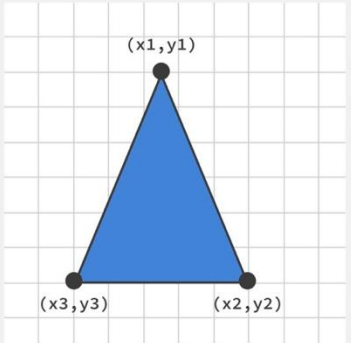

```
ellipse(x, y, width, height);
circle(x, y, d);
```

rectangle / square



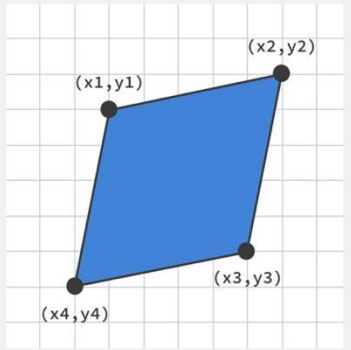

```
rect(x, y, width, height);
square(x, y, s);
```

triangle




```
triangle(x1, y1, x2, y2, x3, y3);
```

quad




```
quad(x1, y1, x2, y2, x3, y3, x4, y4);
```

color definitions

```
(120) // grayscale (0-255)
(100,125,255); // Red, Green, Blue (0-255)
```

background

```
background(100,125,255);  // set the background color
```

fill

```
fill(100,125,255);
// sets shape color
```

noFill()

```
// remove shape color
```

stroke

```
stroke(100,125,255);  // set stroke color
```

strokeWeight

```
strokeWeight(3);  // set the stroke thickness to 3px
```

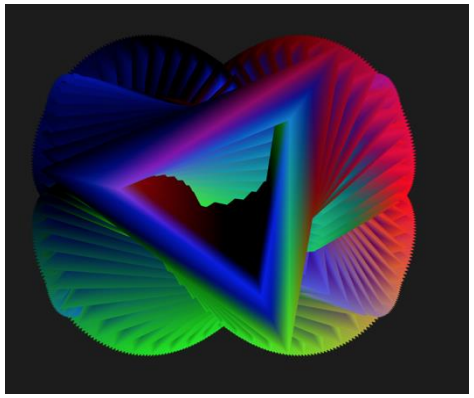
noStroke()

```
// remove stroke
```

You can go further with this…

Wrap Up

Creative Coding with p5.js



p5.js is a coding framework to create images, animations, games, interactions...

It is a tool to be digitally creative.

p5.js offers a fun and creative way of starting to learn to code.

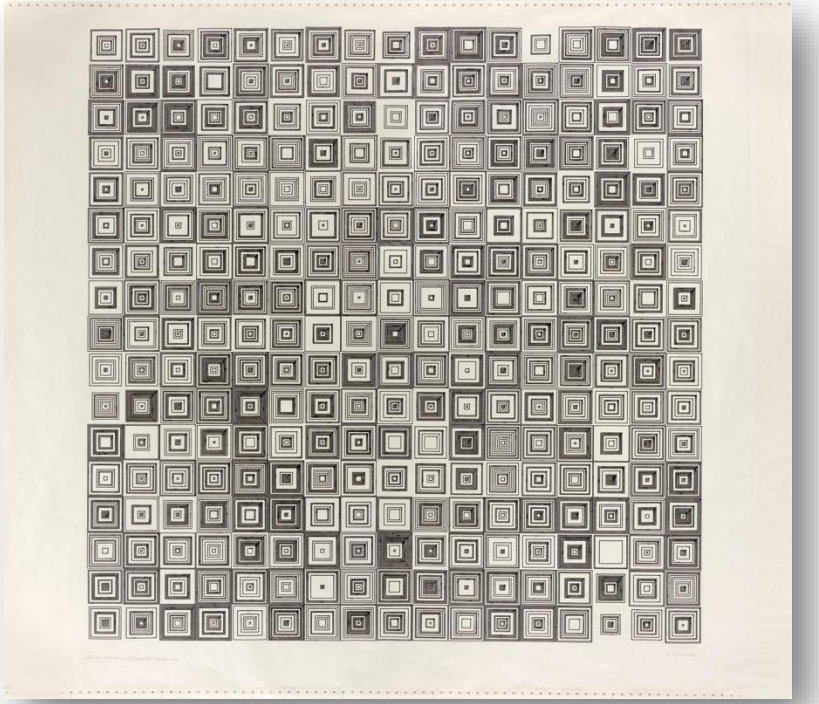
With **p5.js**, you can build visualizations using shapes, text, images, videos, and sounds.

Official Website: https://p5js.org

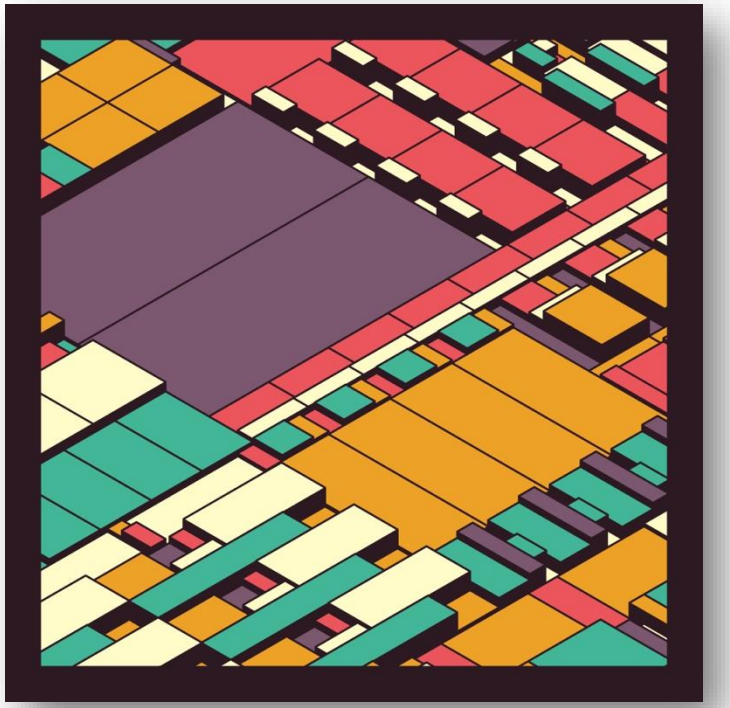
Generative Art

A computer or a programme helps artists to create works of art.
They follow certain rules and use a mixture of mathematics, chance and creativity.

Dmitri Cherniak, *Ringers #879*
(auctioned for $ 6,2 Mio)



Vera Molnar, *„DE LA SÉRIE (DES) ORDRES“*
(auctioned for € 25‘625)



Kjetil Golid, *Archetype #183*
(his works realized prices up to
$ 330‘200 at auctions)

Generative Art in Museums

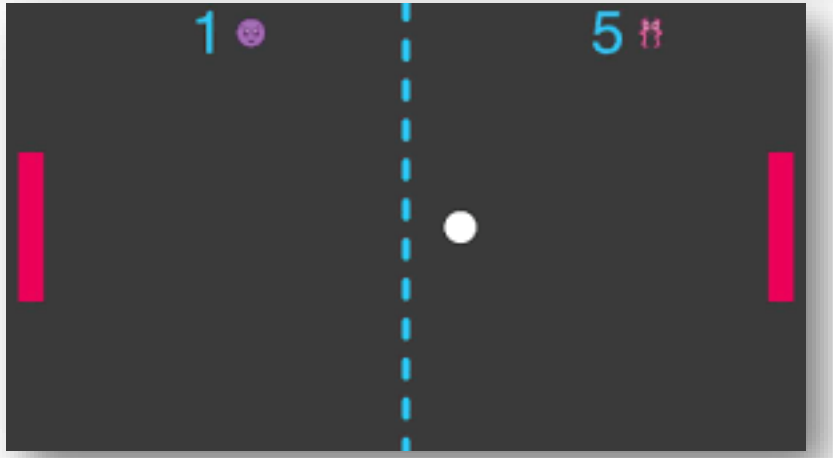
Refik Anadol, *Unsupervised*
(at Museum of Art MoMA in New York)

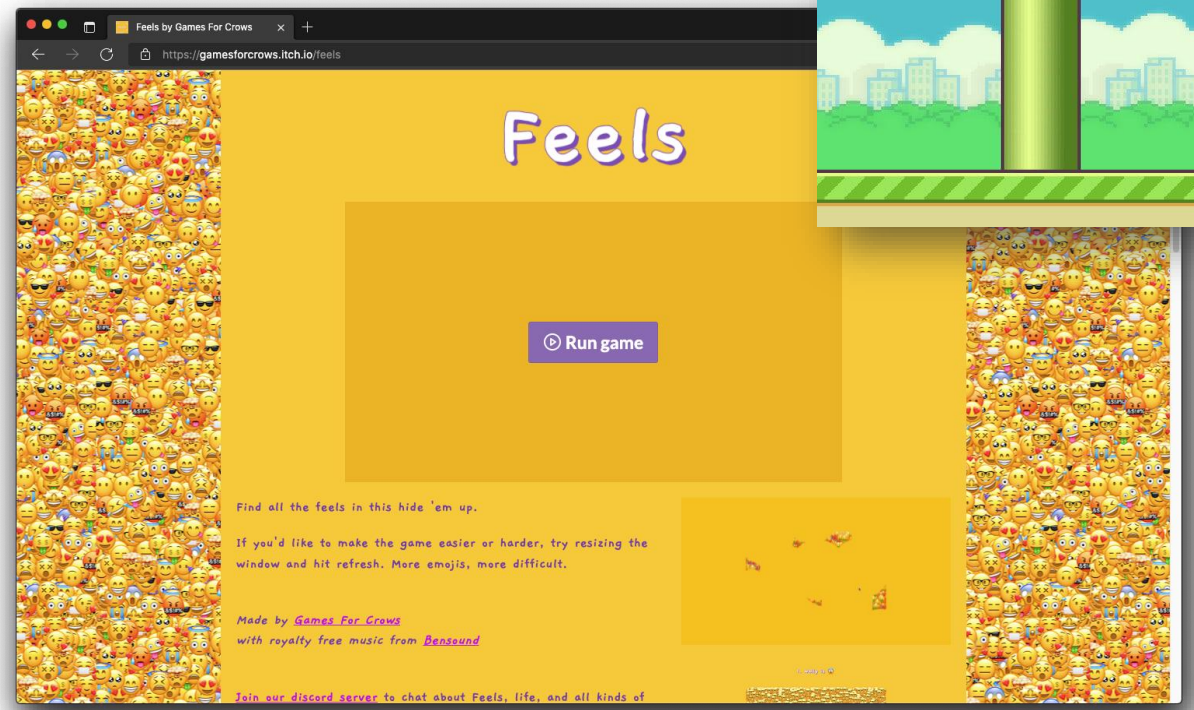
„I discovered generative art after stumbling upon Coding Train tutorials on YouTube. I immediately fell in love with the genre and taught myself the foundational techniques of creative coding and started producing an early body of work.“

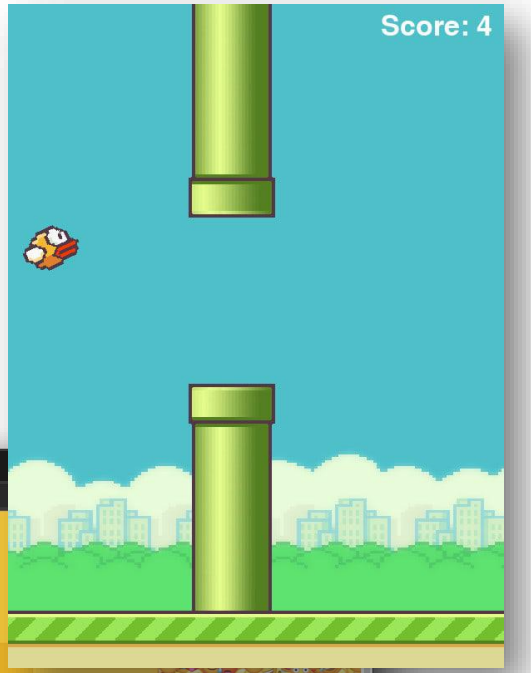
Emily Xie, *Memories of Qilin*
(generated 1024 variations issued as NFTs)

Games


score: 0
lives:
level: 1


1
5

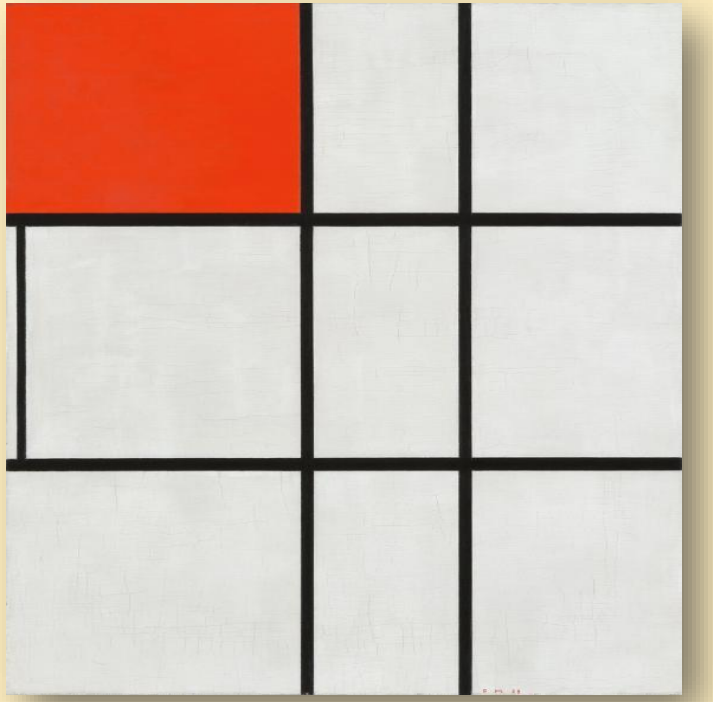

Feels by Games For Crows
https://gamesforcrows.itch.io/feels
Feels
Run game
Find all the feels in this hide 'em up.
If you'd like to make the game easier or harder, try resizing the window and hit refresh. More emojis, more difficult.
Made by Games For Crows
with royalty free music from Bensound
Join our discord server to chat about Feels, life, and all kinds of

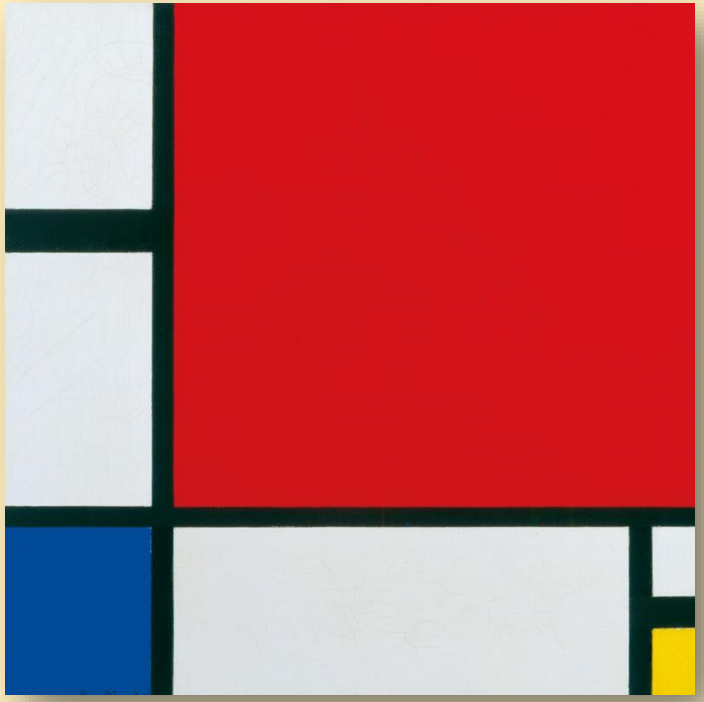

Score: 4

Luxembourg Tech School

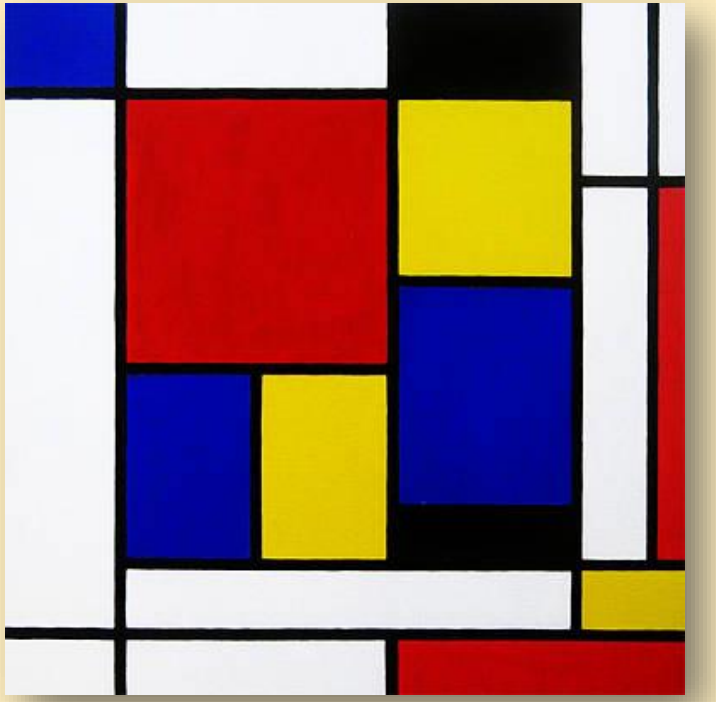
Extra Activity: Recreate an Artwork

Let's recreate an artwork!



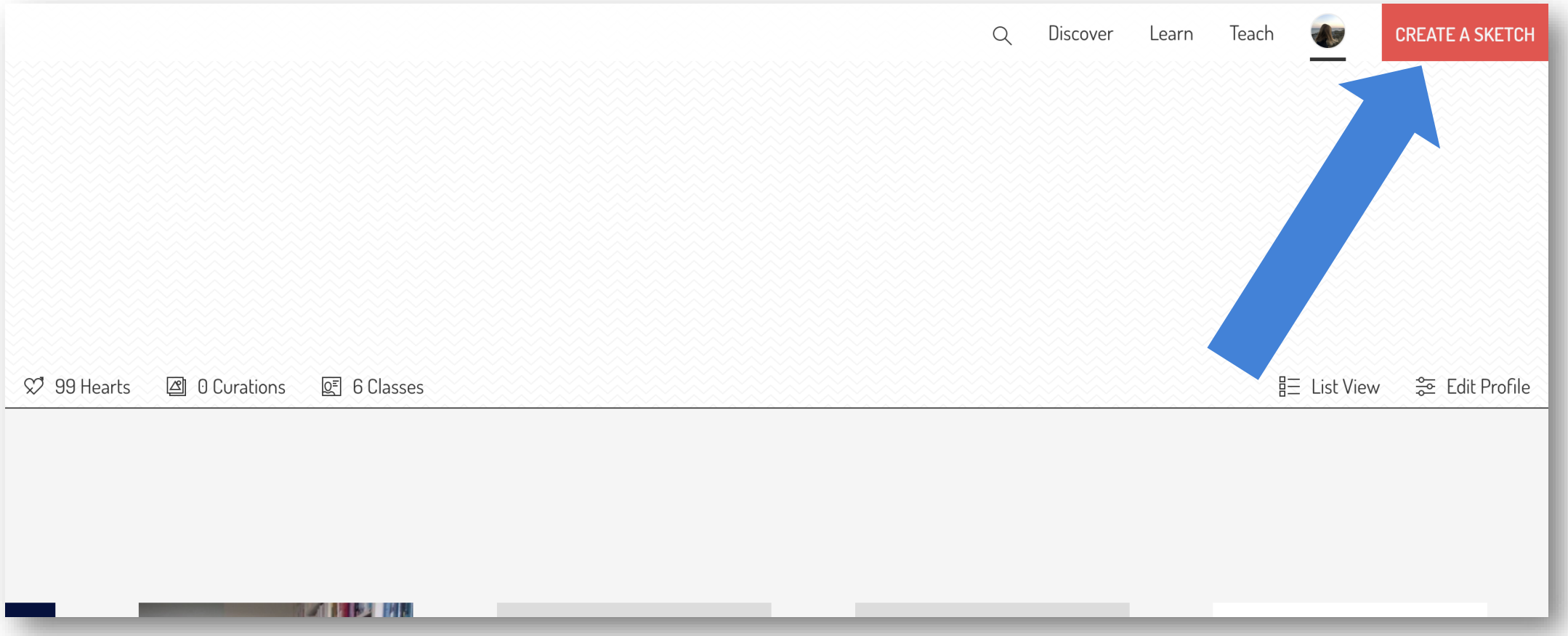


Composition No. II – was sold at Sotheby's New York in 2022 for **51 million dollar**



Piet Mondrian

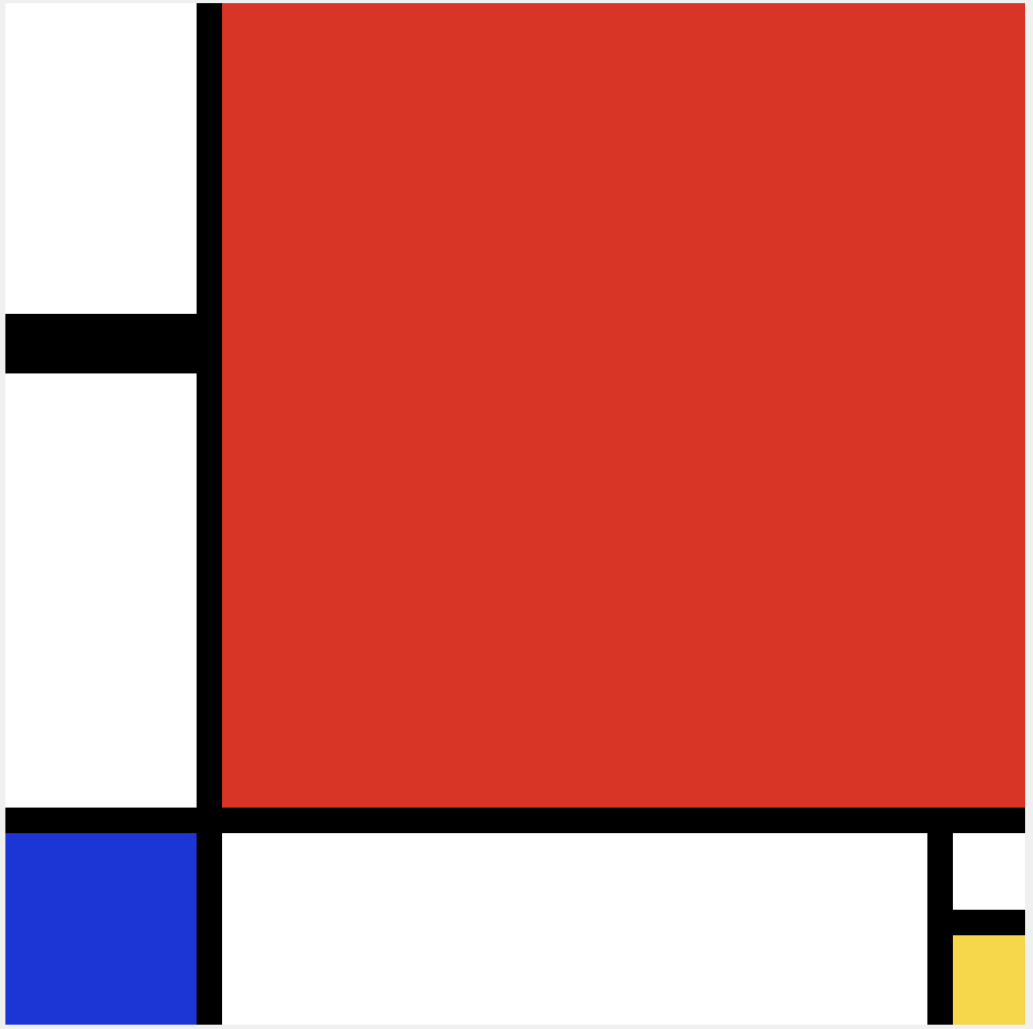
Open a new sketch



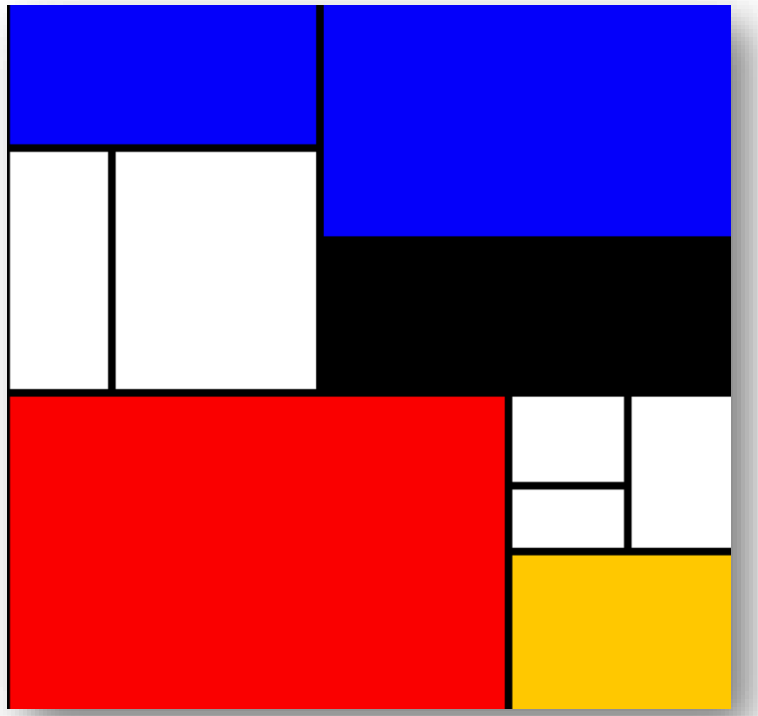
Piet Mondrian – one version

mySketch

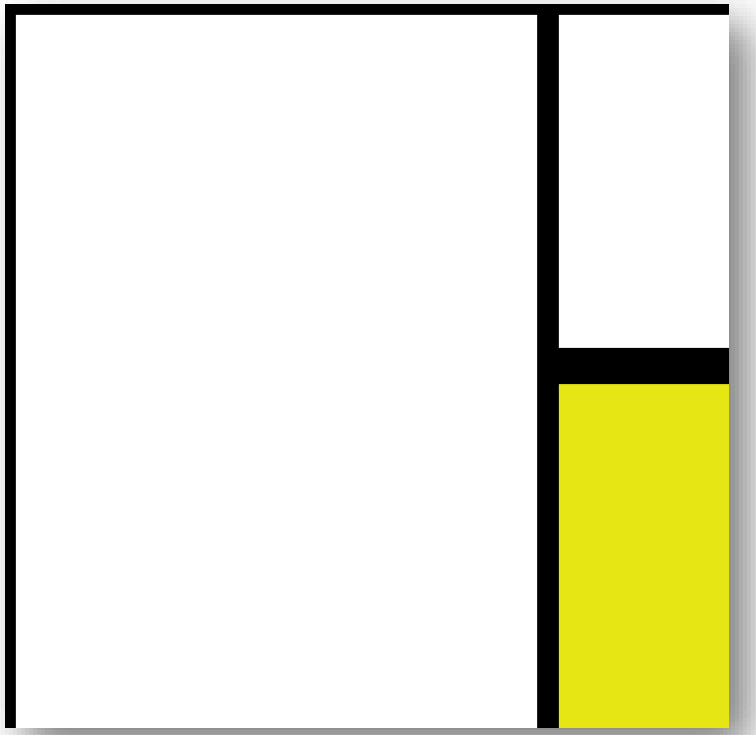
```
function setup() {
  createCanvas(600,600);
  background(255);
}

function draw() {
  fill(235, 22, 19);
  strokeWeight(15);
  square(120,-10, 490);
  fill(18, 55, 222);
  rect(-10,480,130,200);
  line(550,490,550,600);
  fill(252, 213, 15);
  square(550,540,80);
  strokeWeight(35);
  line(0,200,110,200);

}
```



Some Mondrian variations in p5.js



Mondrian Generator by Roni Kaufman
Sketch + Sketch



Interactive Mondrian by Emily
Tutorial