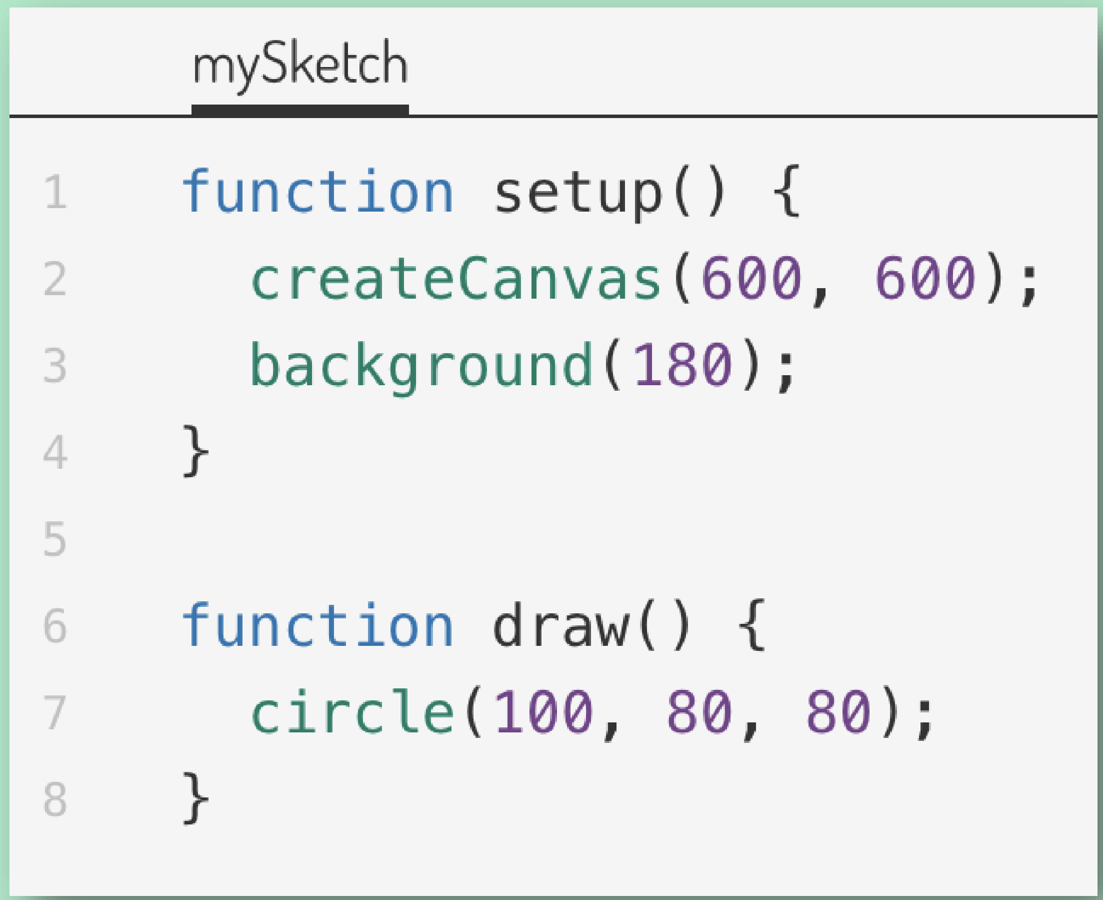
Luxembourg Tech School

LEVEL GO
2025-2026

3 – Movement with p5.js

Recap Quiz

1. What does this code do?

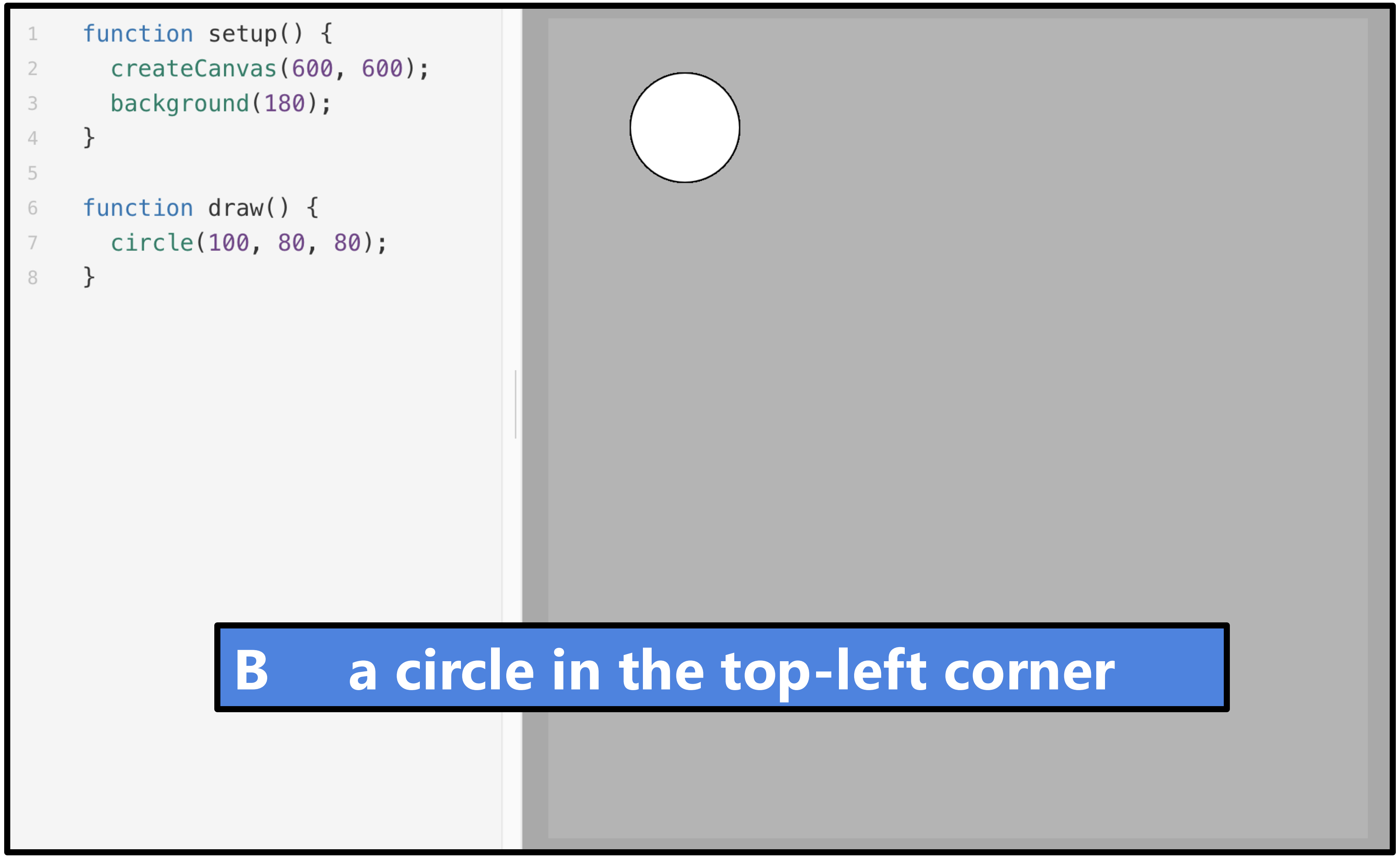


mySketch

```
function setup() {
  createCanvas(600, 600);
  background(180);
}

function draw() {
  circle(100, 80, 80);
}
```

- (A) a black rectangle
- (B) a circle in the top-left corner
- (C) a circle in the middle of the canvas
- (D) nothing; there is something missing in the code

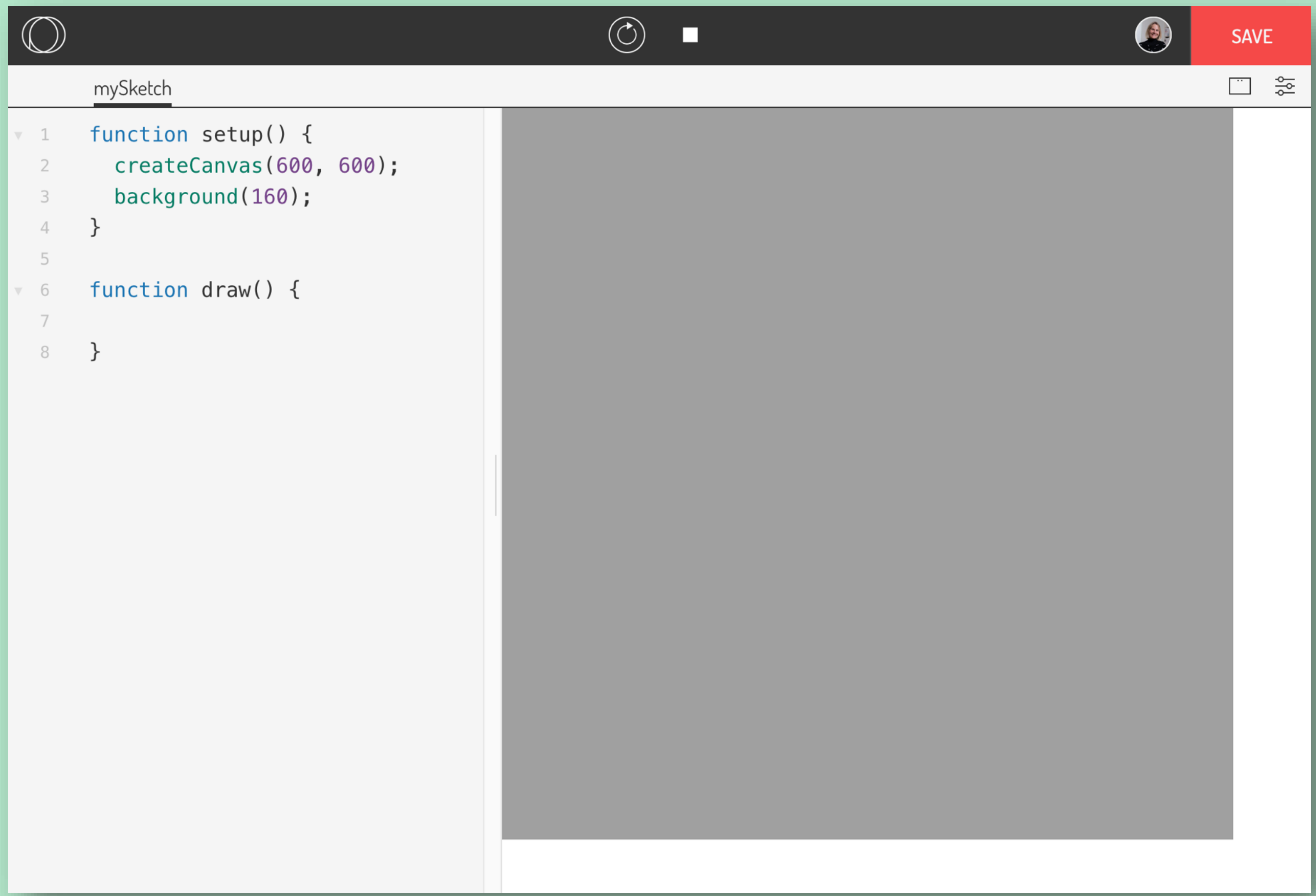
1. What does this code do?



```
function setup() {
  createCanvas(600, 600);
  background(180);
}

function draw() {
  circle(100, 80, 80);
}
```

B a circle in the top-left corner

2. Where is (0, 0) on the canvas?



```
function setup() {
  createCanvas(600, 600);
  background(160);
}

function draw() {

}
```

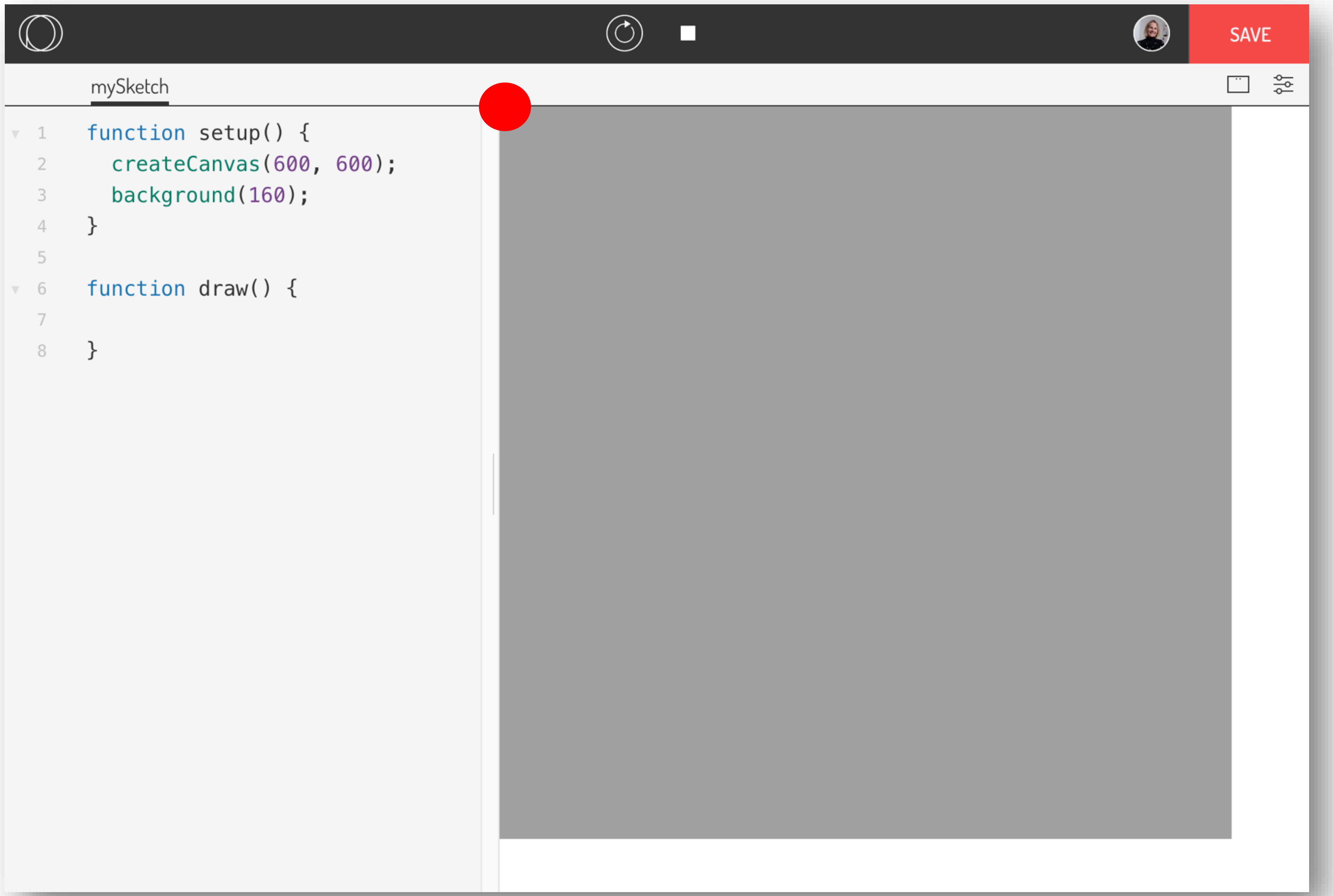
A bottom right

B top left

C center

D random position

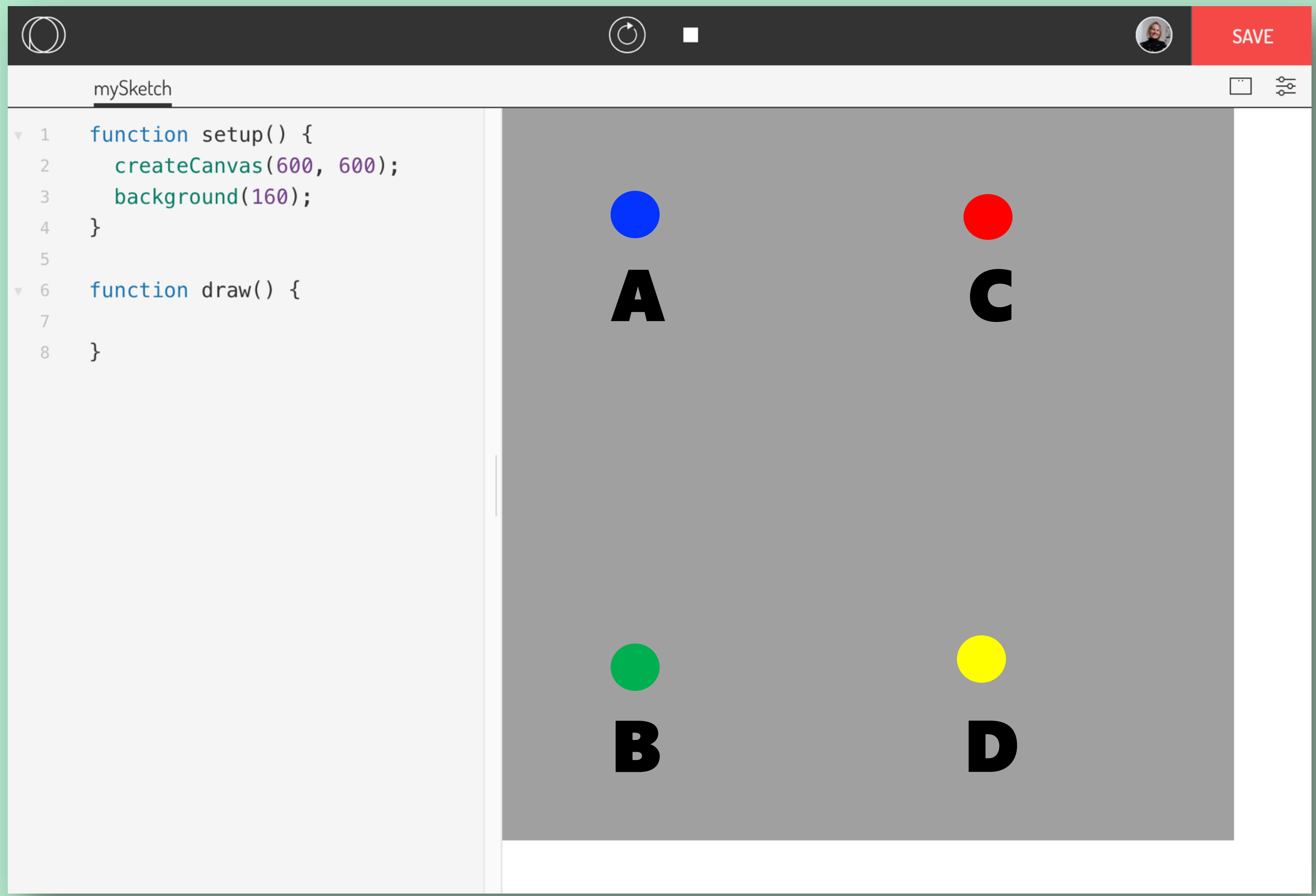
2. Where is (0, 0) on the canvas?



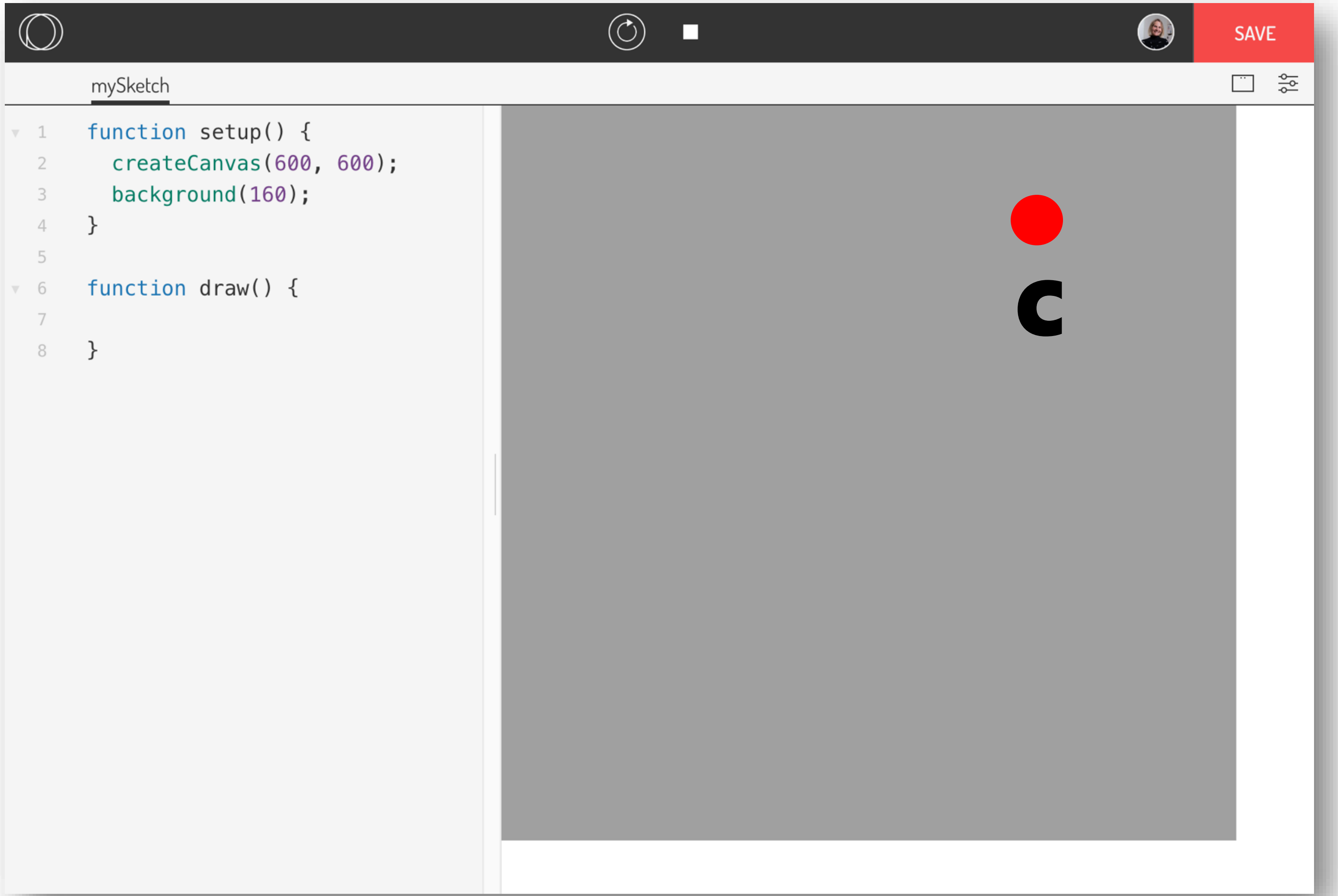
```
function setup() {
  createCanvas(600, 600);
  background(160);
}

function draw() {

}
```

- Ⓐ bottom right
- [✓ Ⓑ top left]
- Ⓒ center
- Ⓓ random position

3. Where is (400, 100) on the canvas?

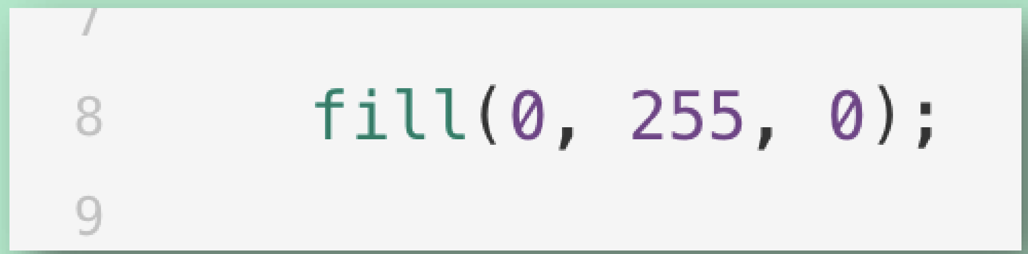

```
function setup() {
  createCanvas(600, 600);
  background(160);
}

function draw() {

}
```

2. Where is (400, 100) on the canvas?



```
function setup() {
  createCanvas(600, 600);
  background(160);
}

function draw() {

}
```

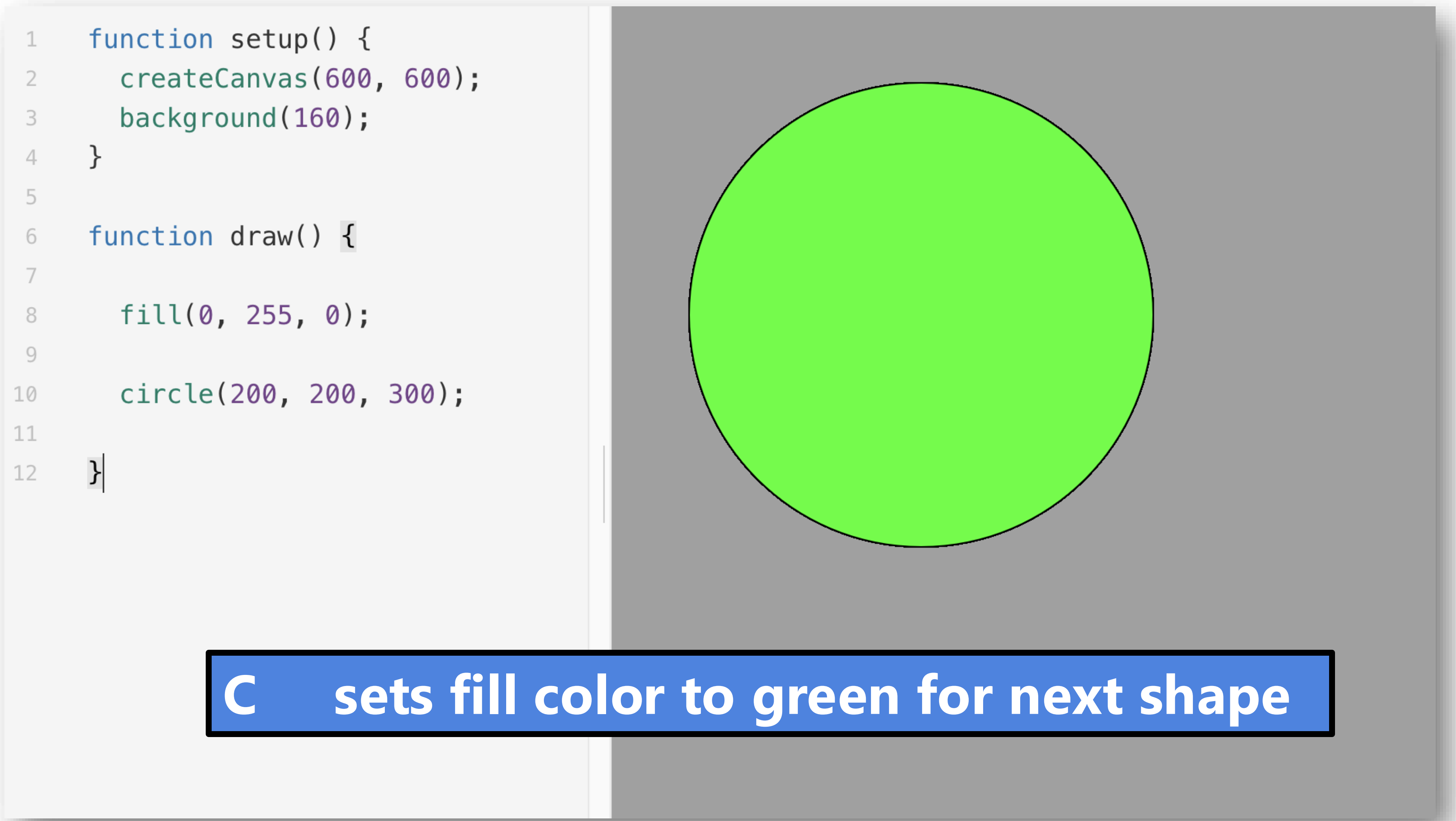
4. What does this line of code do?



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

- (A) sets background color to green
- (B) paints canvas red
- (C) sets fill color to green for next shape
- (D) deletes all shapes

4. What does this line of code do?

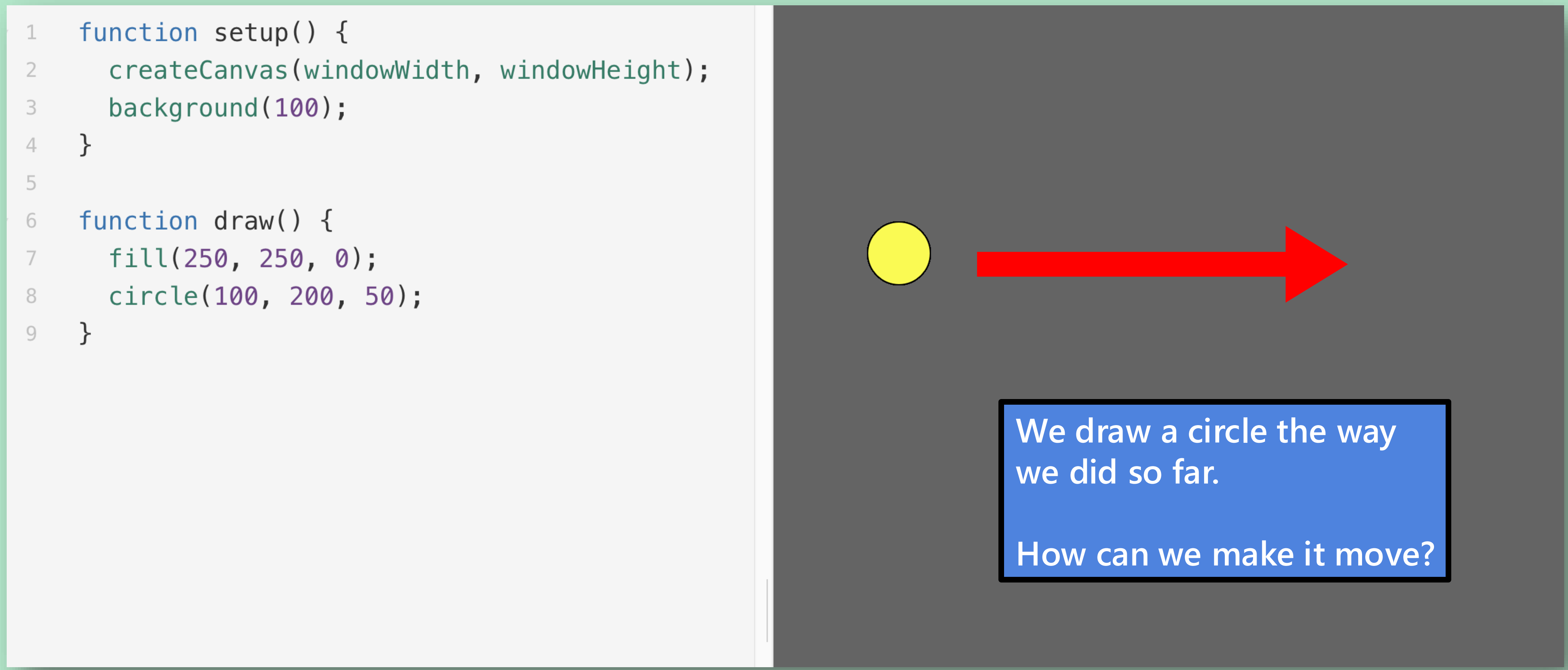


```
function setup() {
  createCanvas(600, 600);
  background(160);
}

function draw() {

  fill(0, 255, 0);

  circle(200, 200, 300);

}
```

C sets fill color to green for next shape

Let‘s start with some new things…

What if we want this circle to move?

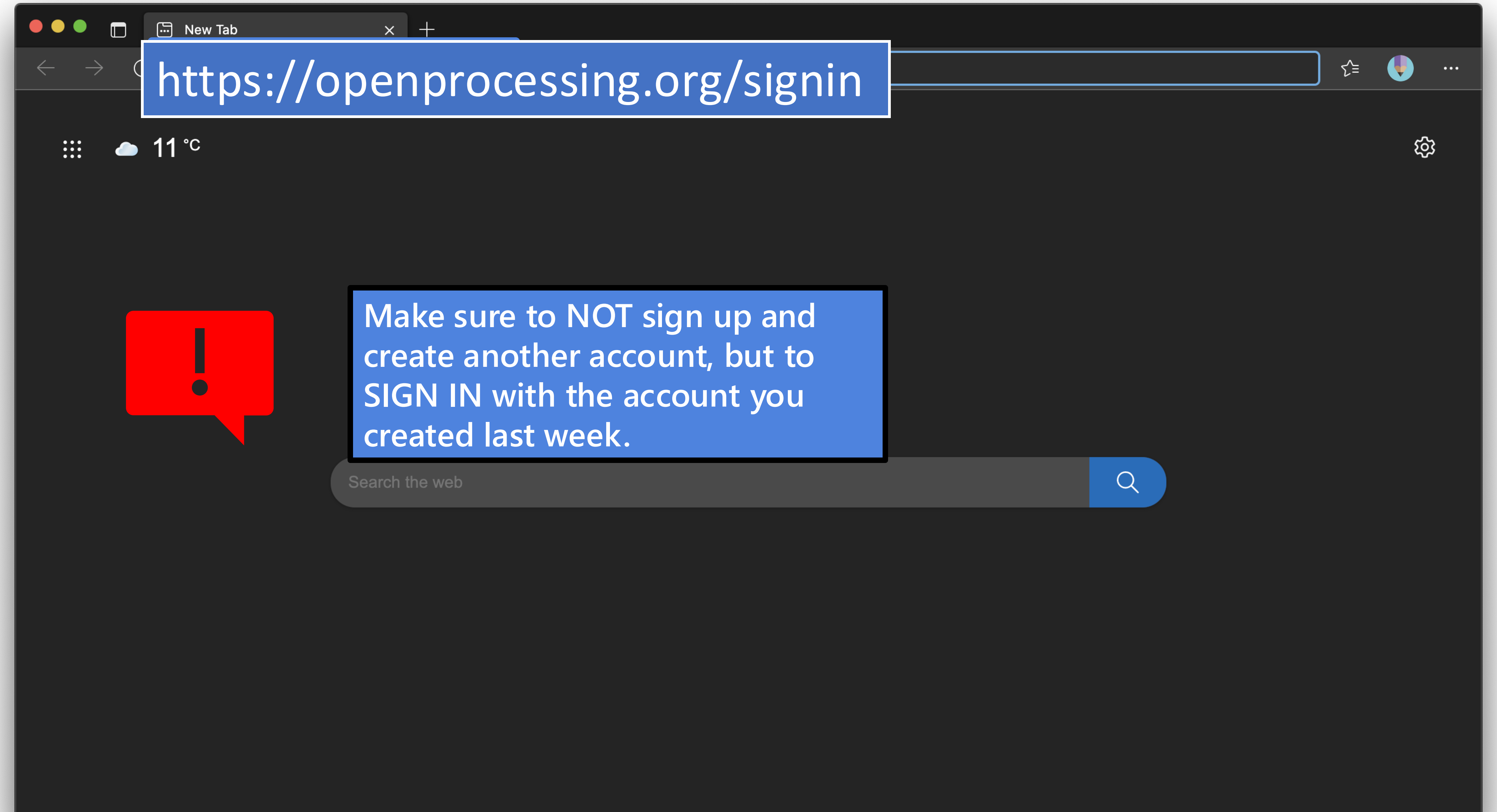


```
function setup() {
  createCanvas(windowWidth, windowHeight);
  background(100);
}

function draw() {
  fill(250, 250, 0);
  circle(100, 200, 50);
}
```

We draw a circle the way we did so far.

How can we make it move?

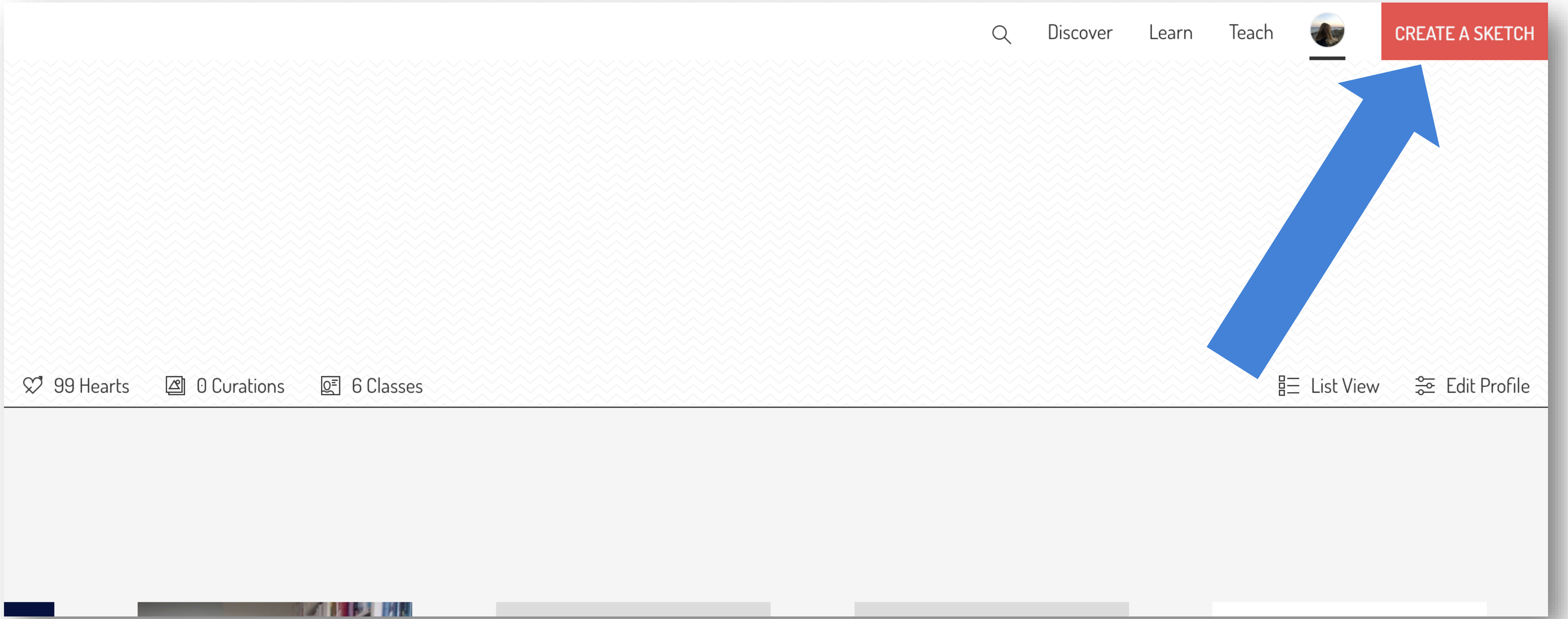
Open OpenProcessing in your browser



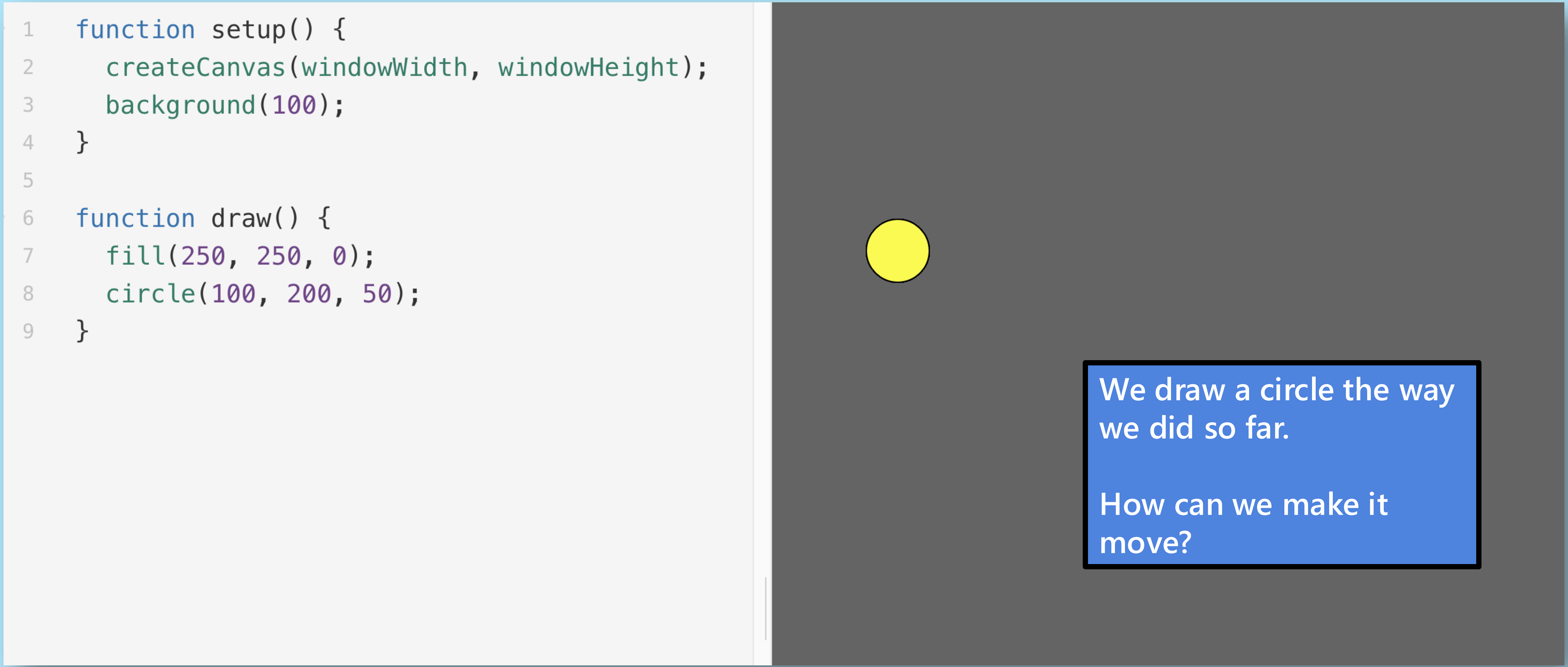
https://openprocessing.org/signin

Make sure to NOT sign up and create another account, but to SIGN IN with the account you created last week.

Create a new sketch



What if we want this circle to move?

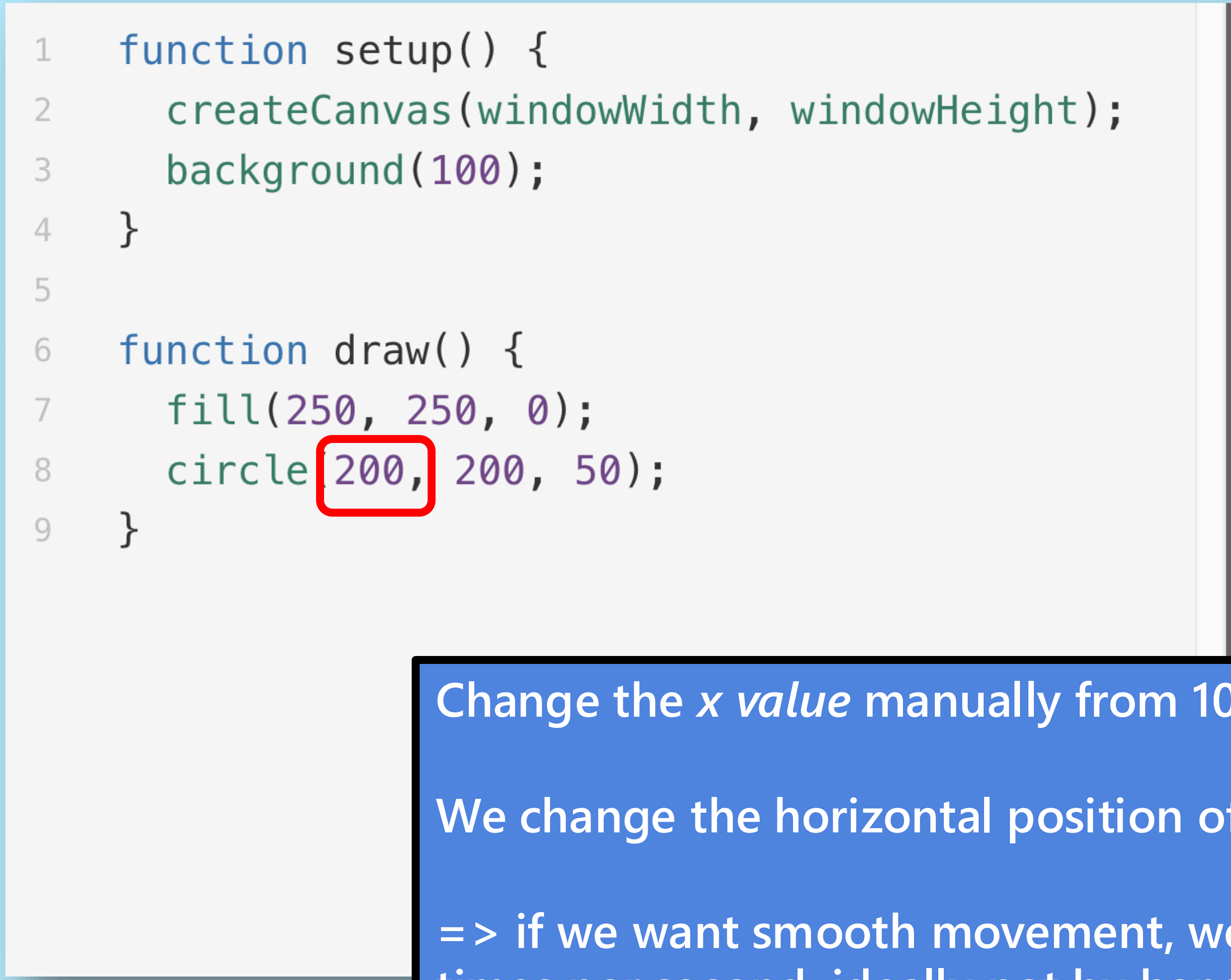


```
function setup() {
  createCanvas(windowWidth, windowHeight);
  background(100);
}

function draw() {
  fill(250, 250, 0);
  circle(100, 200, 50);
}
```

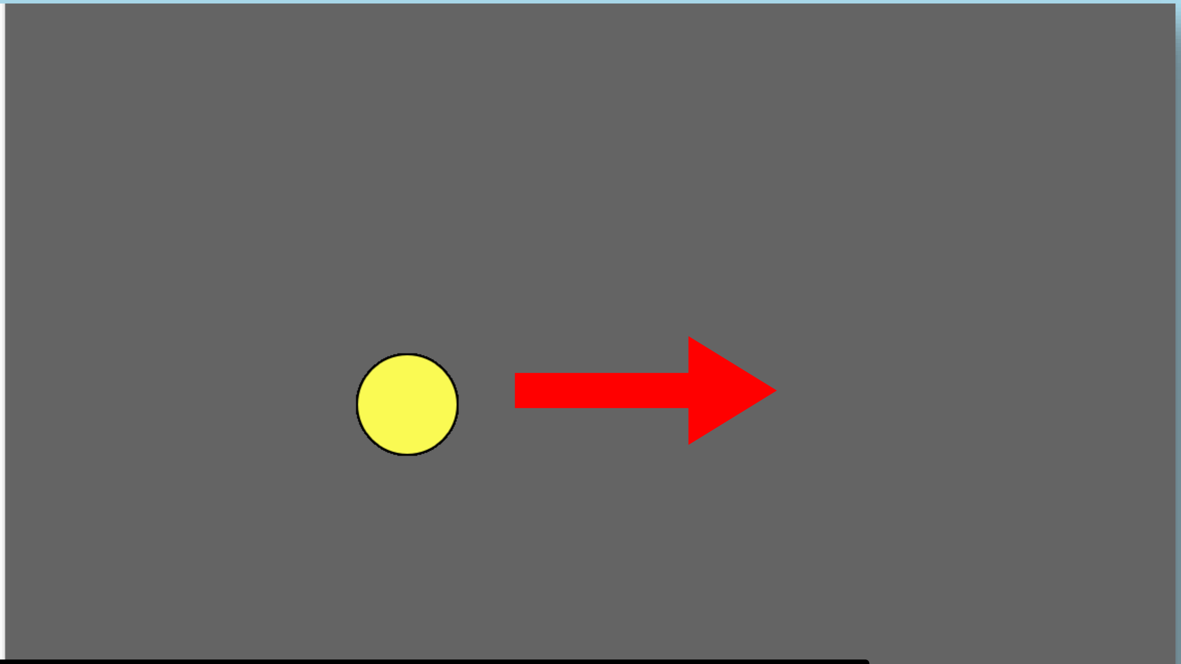
We draw a circle the way we did so far.

How can we make it move?

What if we want this circle to move?



```
function setup() {
  createCanvas(windowWidth, windowHeight);
  background(100);
}

function draw() {
  fill(250, 250, 0);
  circle(200, 200, 50);
}
```

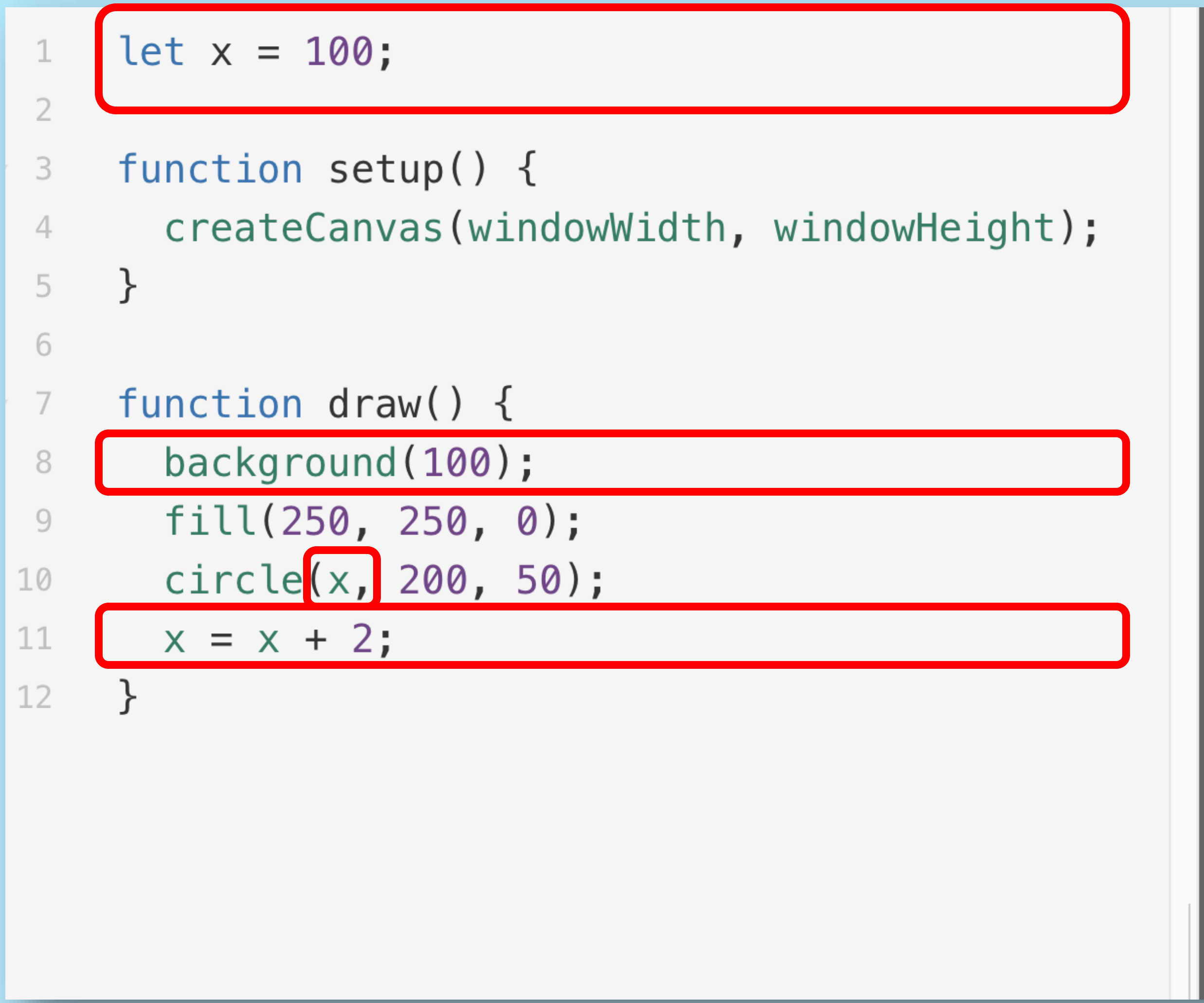


Change the *x value* manually from 100 to 200 to 300.

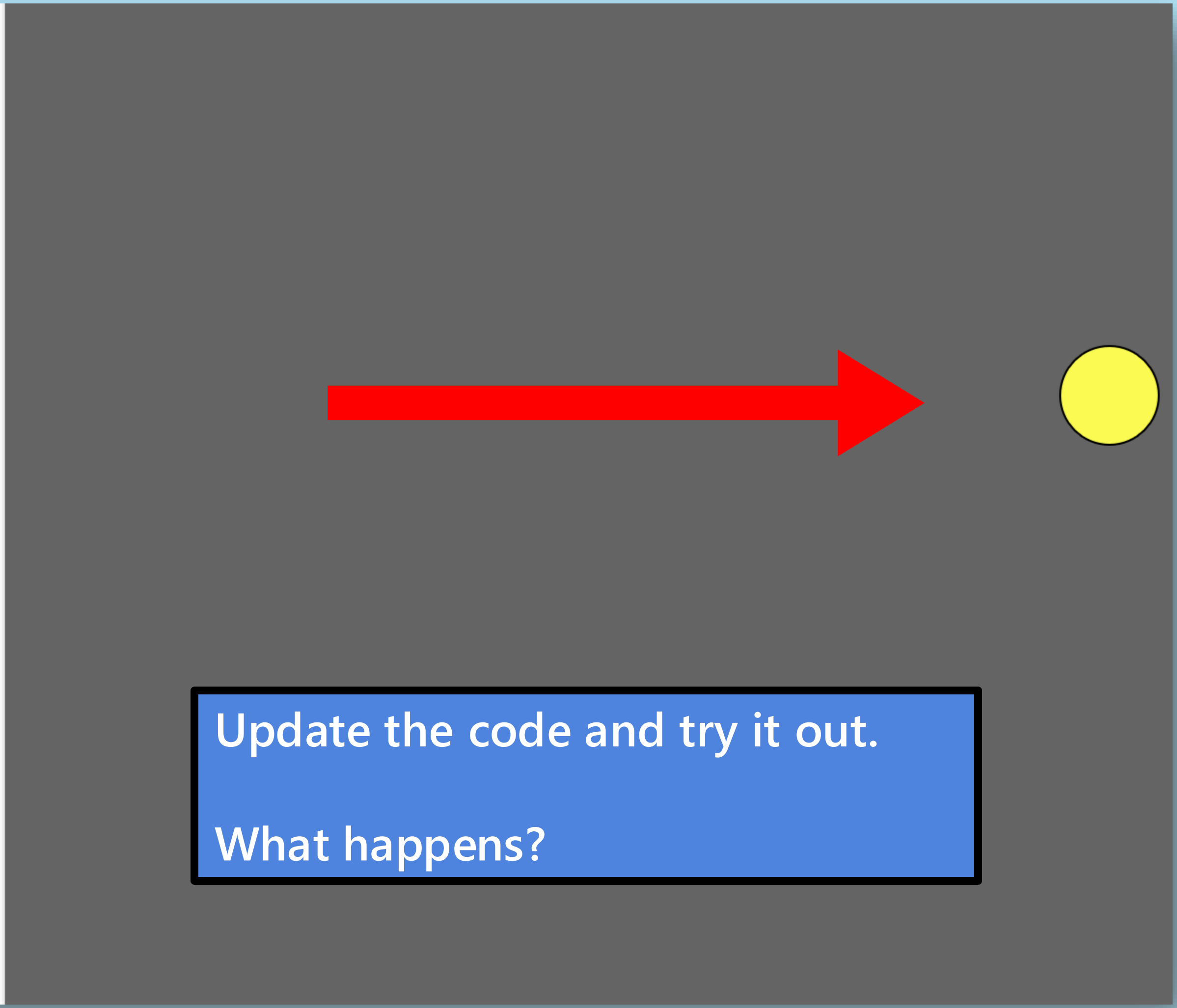
We change the horizontal position of the circle, but it is still static.

=> if we want smooth movement, we need to update the position many times per second, ideally not by hand….

Let‘s update our code to add movement



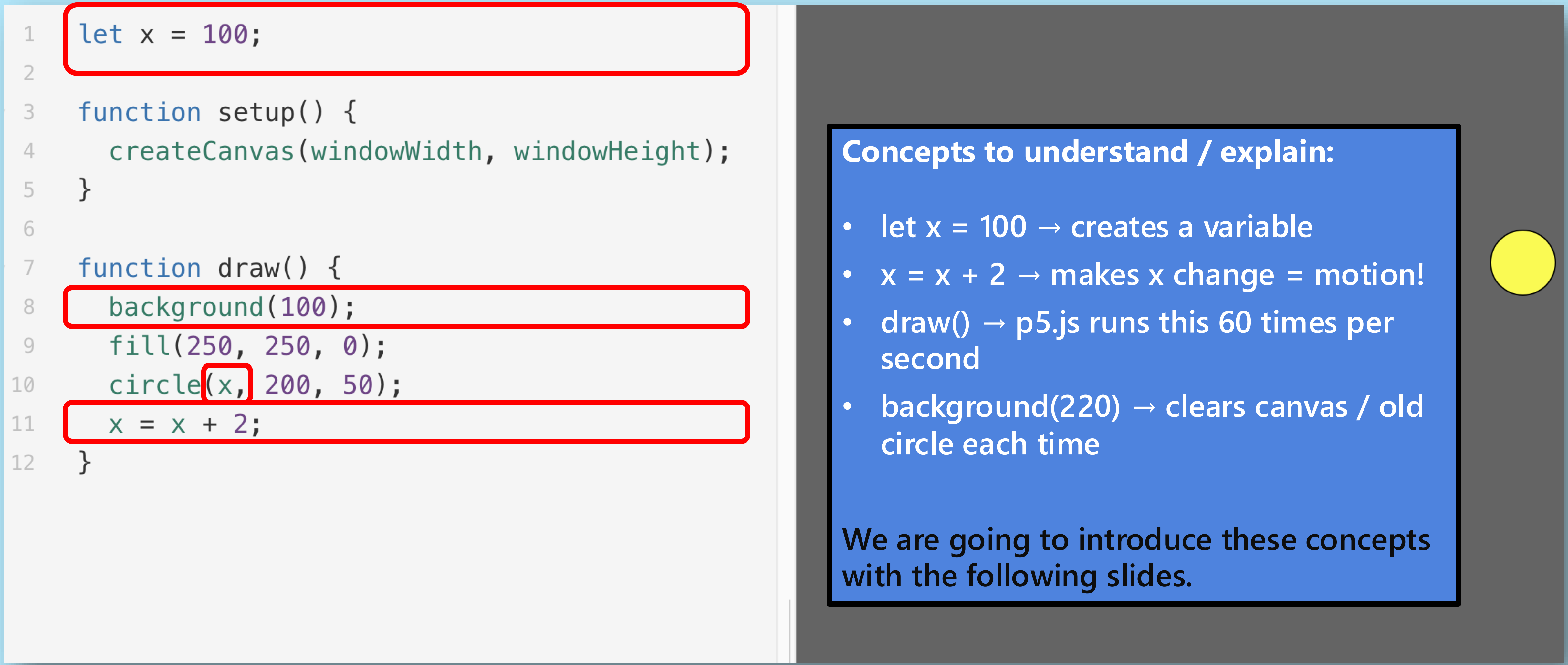
```
let x = 100;

function setup() {
  createCanvas(windowWidth, windowHeight);
}

function draw() {
  background(100);
  fill(250, 250, 0);
  circle(x, 200, 50);
  x = x + 2;
}
```



Update the code and try it out.

What happens?

Let's break it down…



```
let x = 100;

function setup() {
  createCanvas(windowWidth, windowHeight);
}

function draw() {
  background(100);
  fill(250, 250, 0);
  circle(x, 200, 50);
  x = x + 2;
}
```

Concepts to understand / explain:

- let x = 100 → creates a variable
- x = x + 2 → makes x change = motion!
- draw() → p5.js runs this 60 times per second
- background(220) → clears canvas / old circle each time

We are going to introduce these concepts with the following slides.

Variables & draw()

Introducing Variables

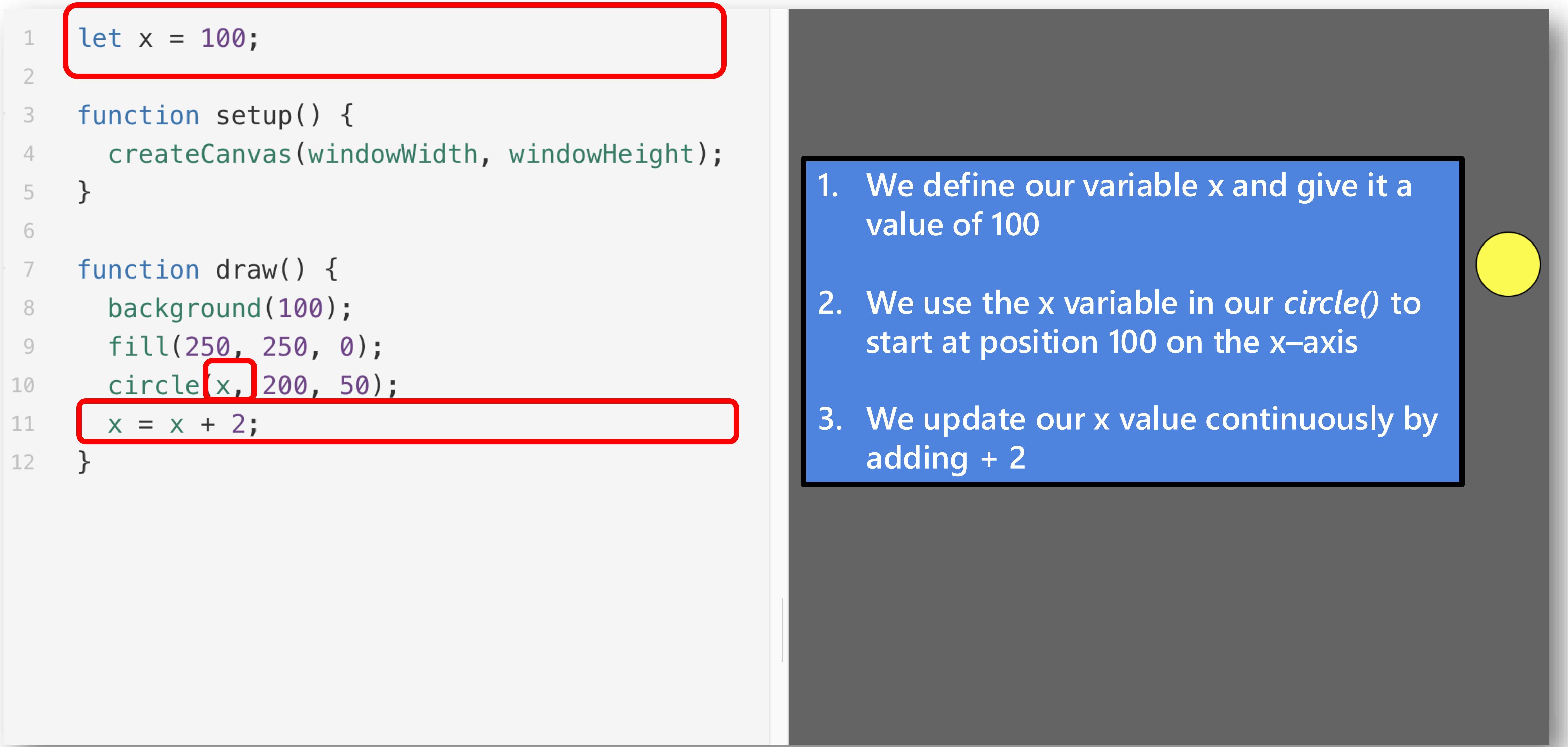
variable

value

Variables are like boxes that we can put values in.

We give the variable a name.

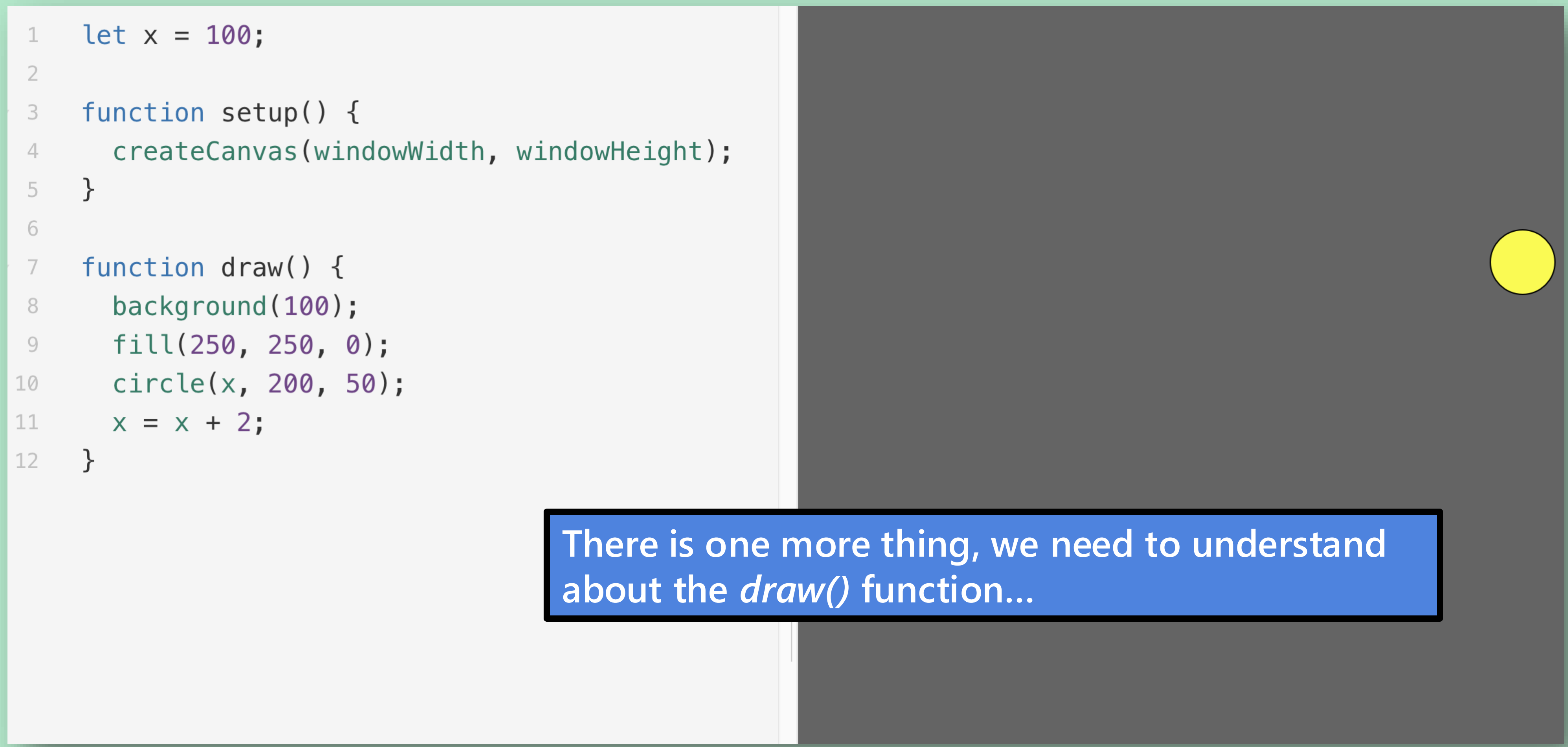
Variables to change position



```
let x = 100;

function setup() {
  createCanvas(windowWidth, windowHeight);
}

function draw() {
  background(100);
  fill(250, 250, 0);
  circle(x, 200, 50);
  x = x + 2;
}
```

1. We define our variable x and give it a value of 100
2. We use the x variable in our *circle()* to start at position 100 on the x–axis
3. We update our x value continuously by adding + 2

But why exactly does it move?



```
let x = 100;

function setup() {
  createCanvas(windowWidth, windowHeight);
}

function draw() {
  background(100);
  fill(250, 250, 0);
  circle(x, 200, 50);
  x = x + 2;
}
```

There is one more thing, we need to understand about the *draw()* function…

Introducing the magic of draw()

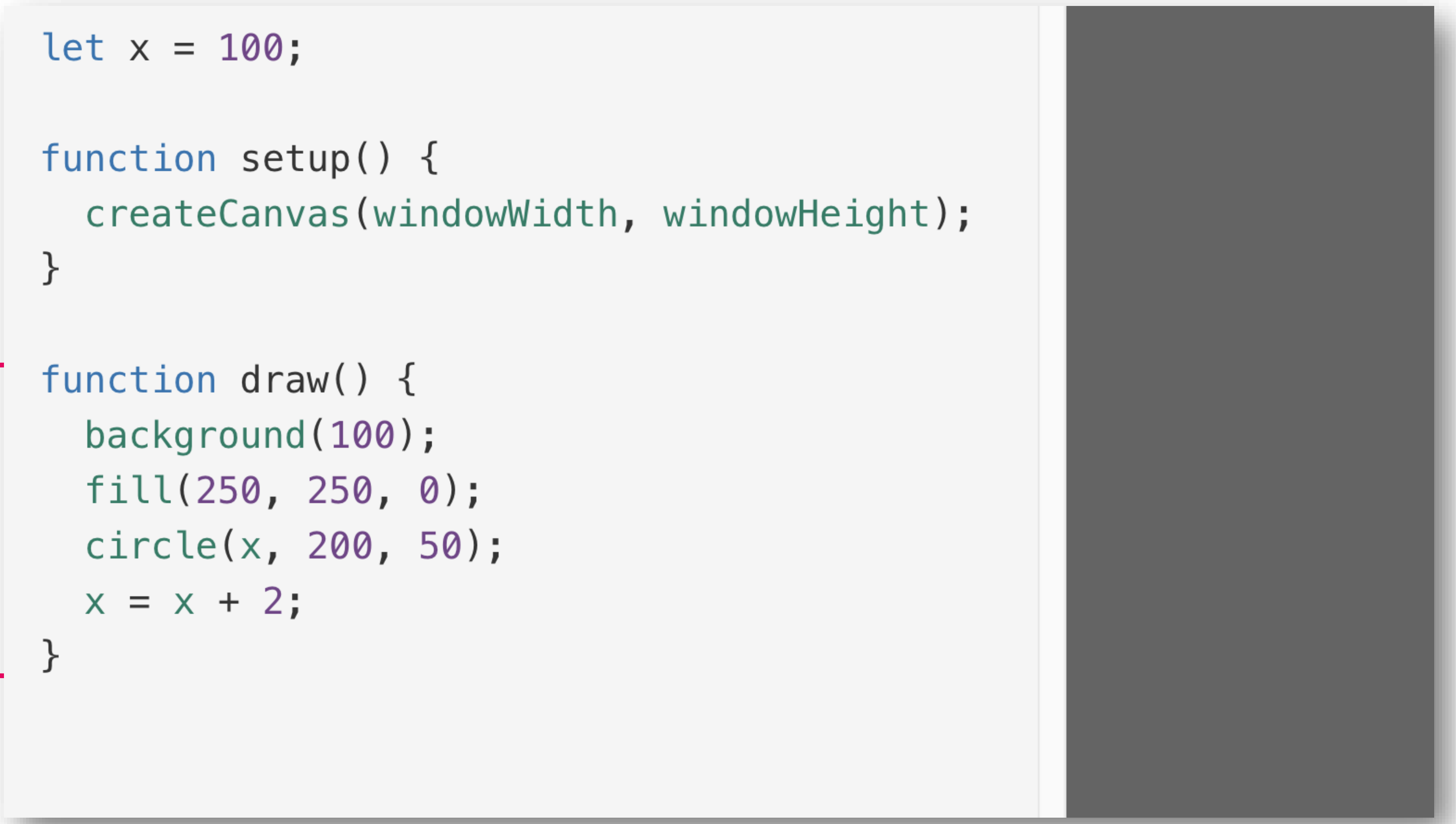
The computer does the commands inside ***draw()*** over and over again.

60 times per second.

It's like a flip book.

https://www.youtube.com/watch?v=JVzf9rtgf9Y

The Magic of draw()

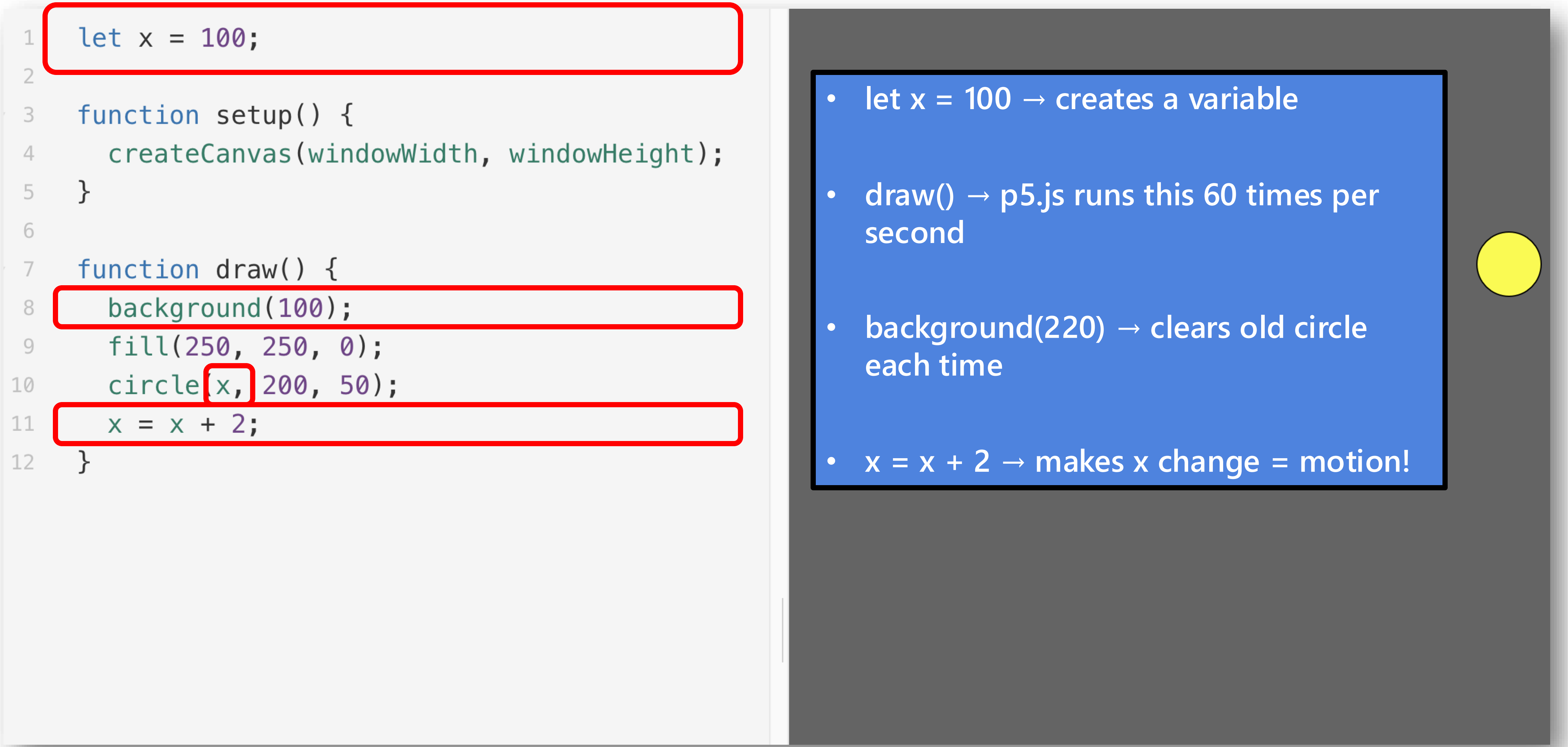


```
let x = 100;

function setup() {
  createCanvas(windowWidth, windowHeight);
}

function draw() {
  background(100);
  fill(250, 250, 0);
  circle(x, 200, 50);
  x = x + 2;
}
```

happens only 1x

happens 60x per second

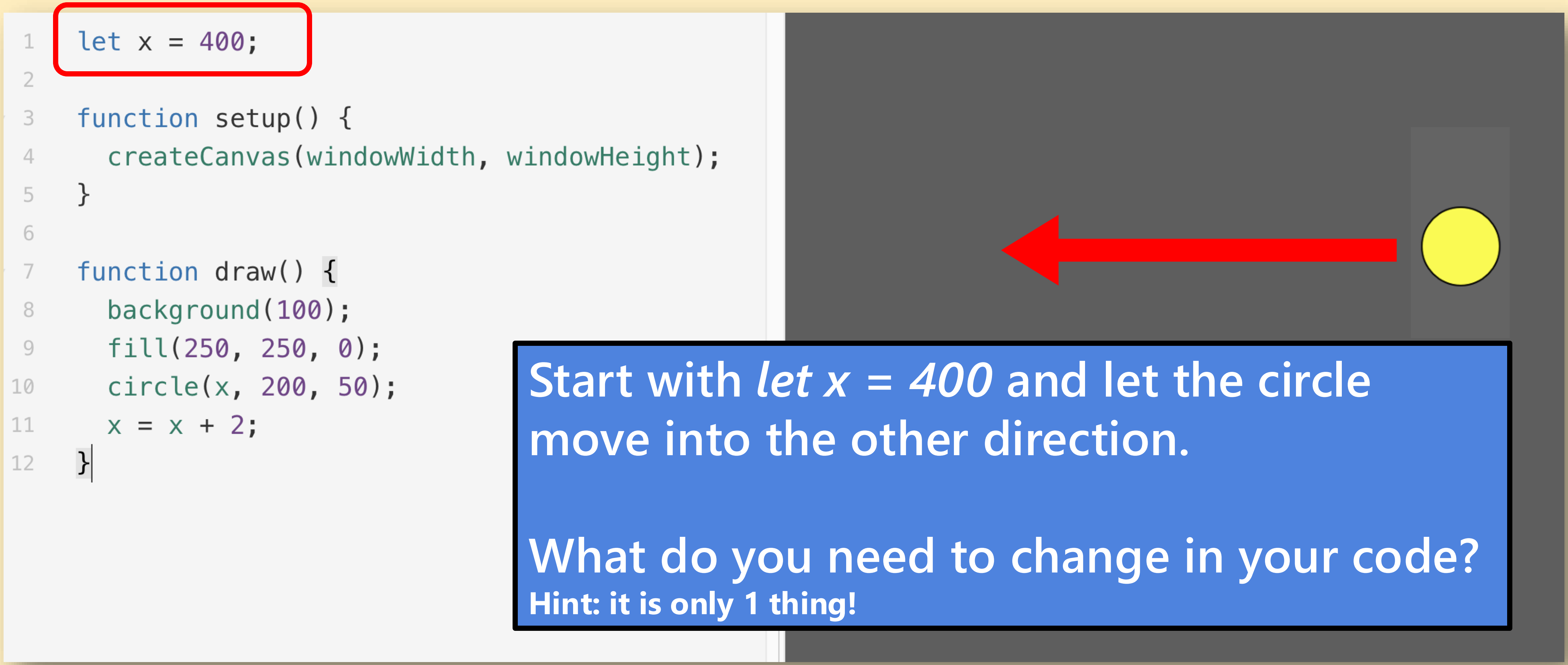
Let‘s break it down…



```
let x = 100;

function setup() {
  createCanvas(windowWidth, windowHeight);
}

function draw() {
  background(100);
  fill(250, 250, 0);
  circle(x, 200, 50);
  x = x + 2;
}
```

- let x = 100 → creates a variable
- draw() → p5.js runs this 60 times per second
- background(220) → clears old circle each time
- x = x + 2 → makes x change = motion!

Mini-Challenge: Change direction !



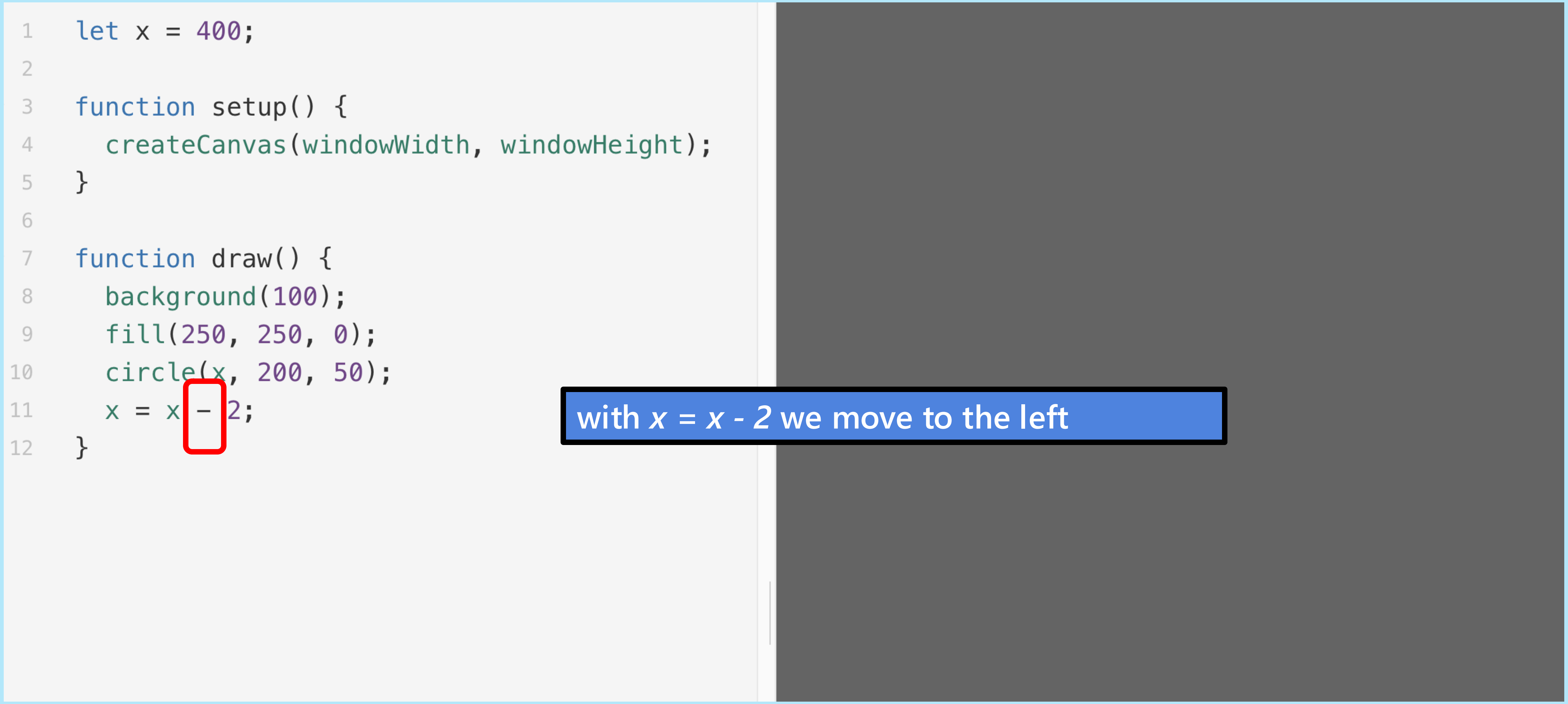
```
let x = 400;

function setup() {
  createCanvas(windowWidth, windowHeight);
}

function draw() {
  background(100);
  fill(250, 250, 0);
  circle(x, 200, 50);
  x = x + 2;
}
```

Start with *let x = 400* and let the circle move into the other direction.

What do you need to change in your code?

Hint: it is only 1 thing!

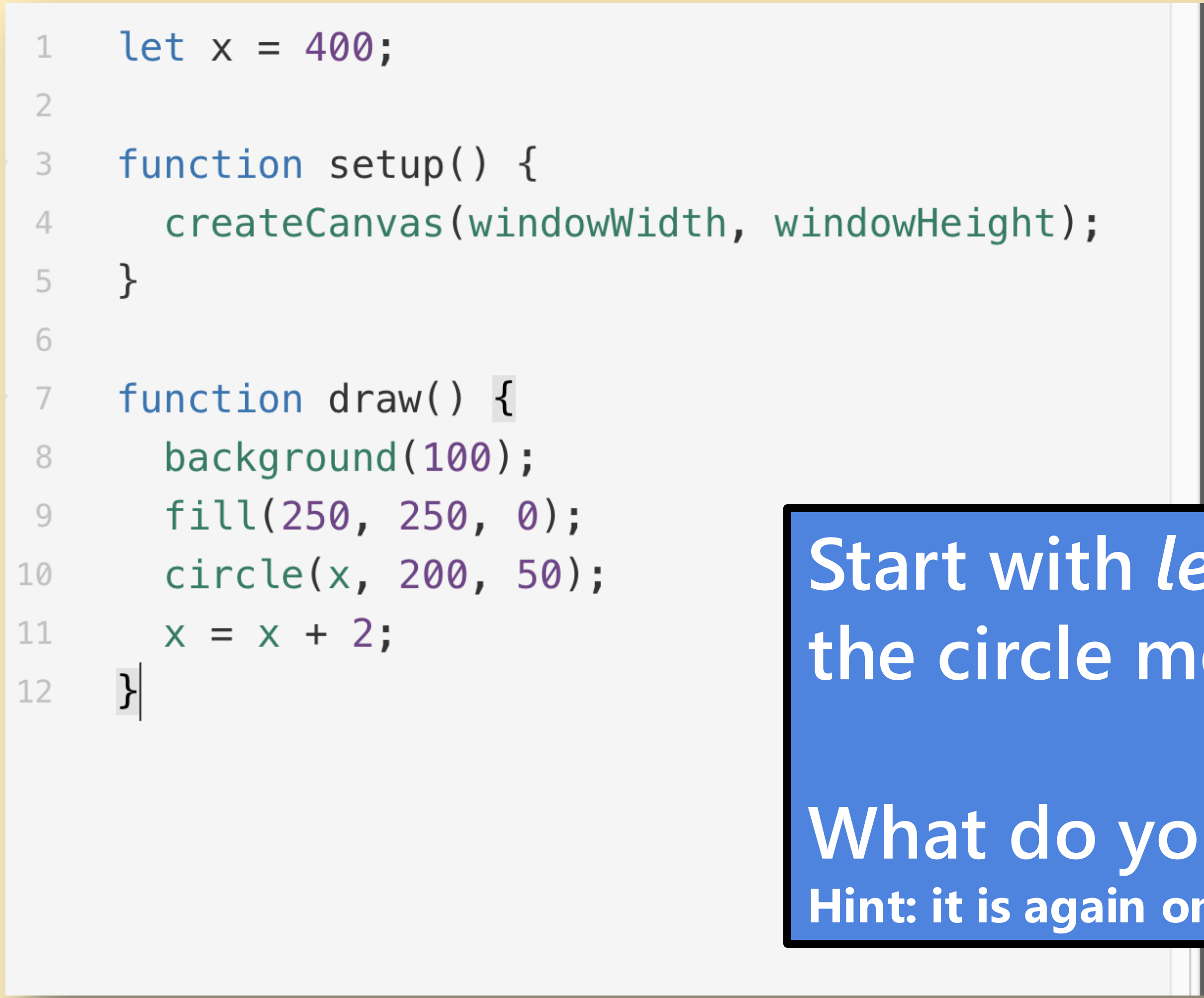
Change direction



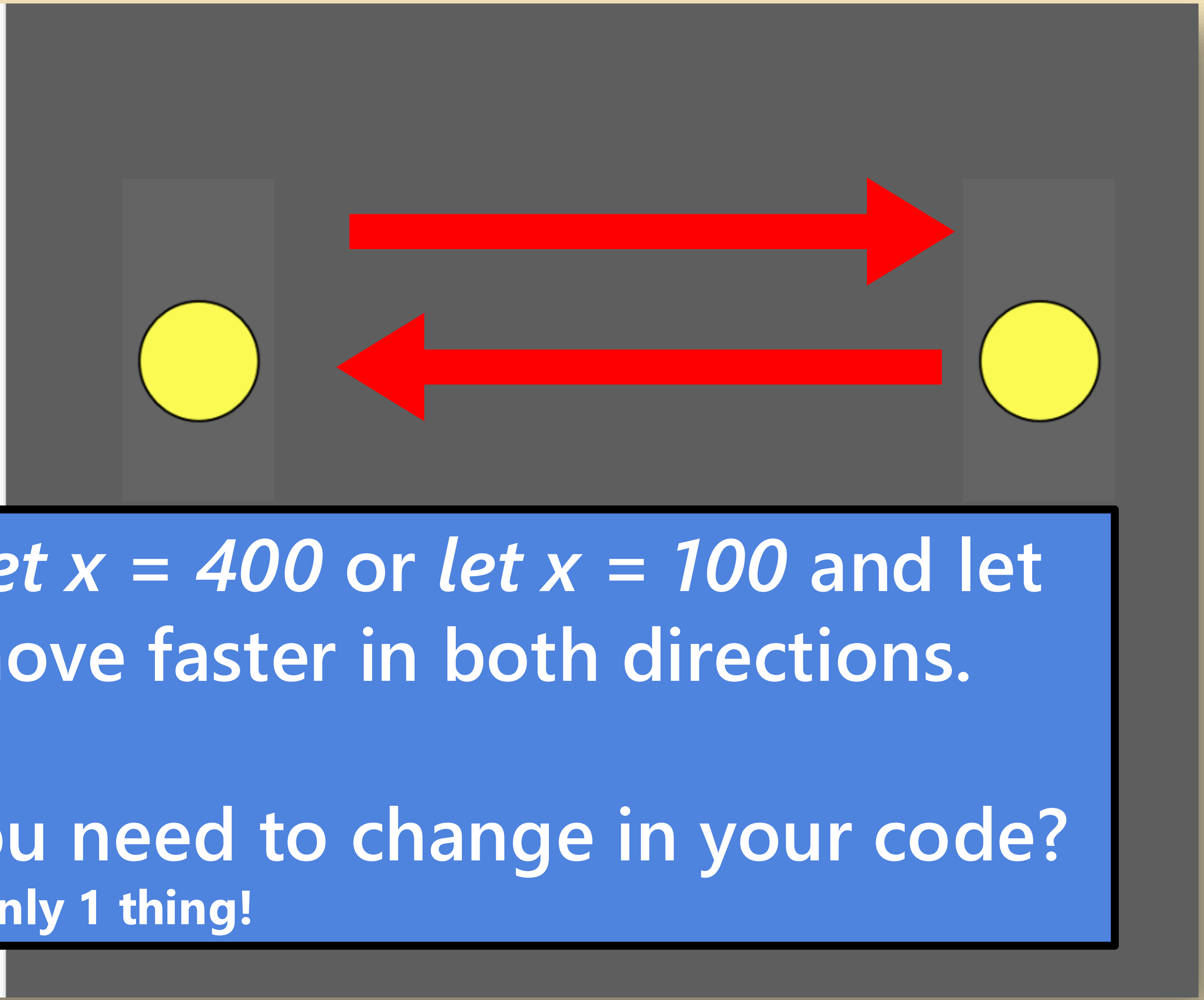
```
let x = 400;

function setup() {
  createCanvas(windowWidth, windowHeight);
}

function draw() {
  background(100);
  fill(250, 250, 0);
  circle(x, 200, 50);
  x = x - 2;
}
```

with *x* = *x - 2* we move to the left

Mini-Challenge: Move it faster!



```
let x = 400;

function setup() {
  createCanvas(windowWidth, windowHeight);
}

function draw() {
  background(100);
  fill(250, 250, 0);
  circle(x, 200, 50);
  x = x + 2;
}
```

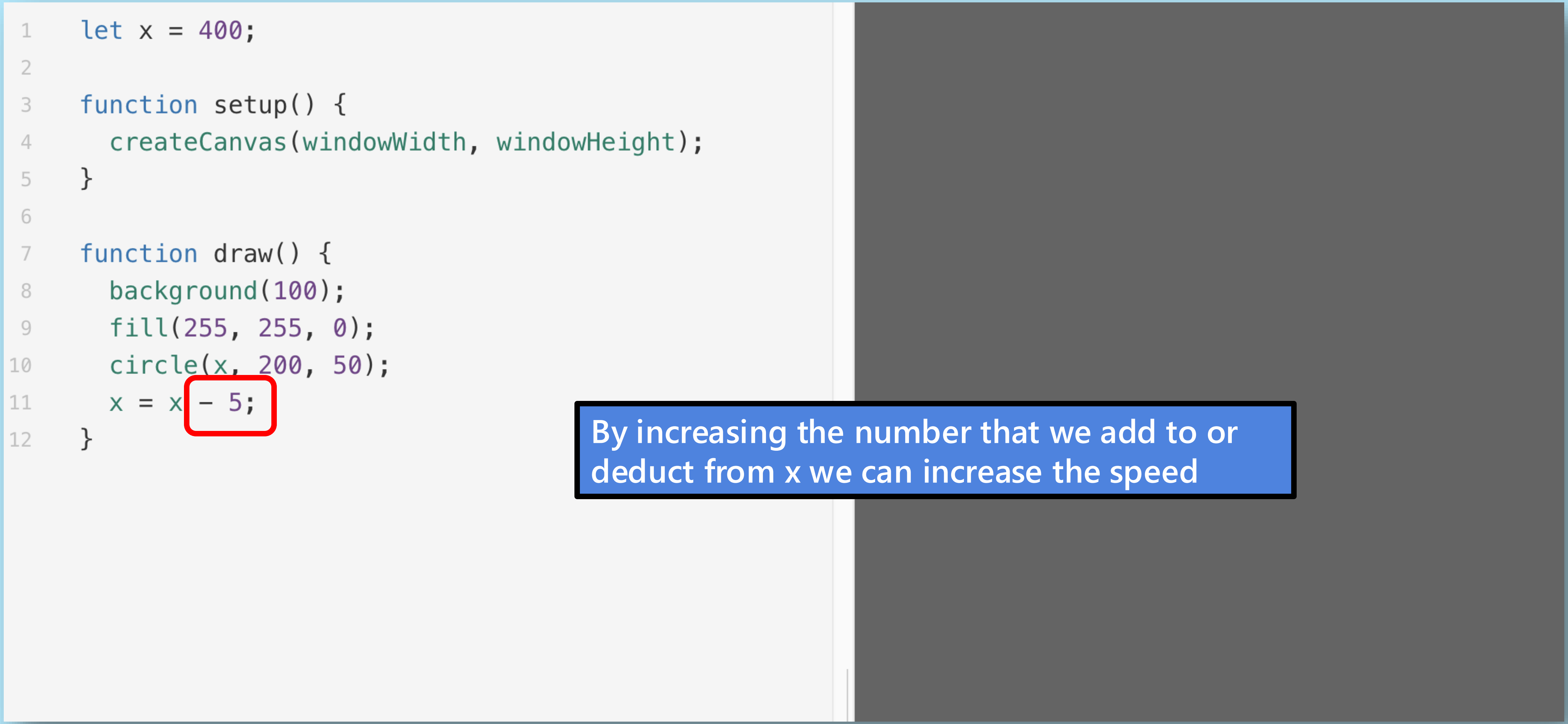


Start with *let x = 400* or *let x = 100* and let the circle move faster in both directions.

What do you need to change in your code?

Hint: it is again only 1 thing!

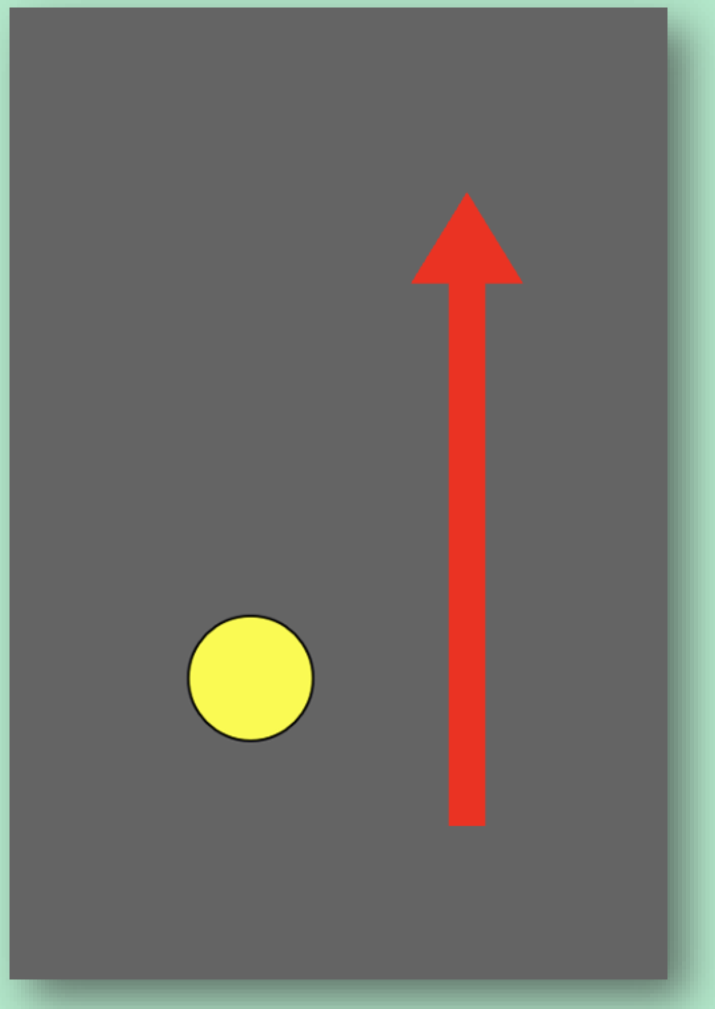
Move it faster

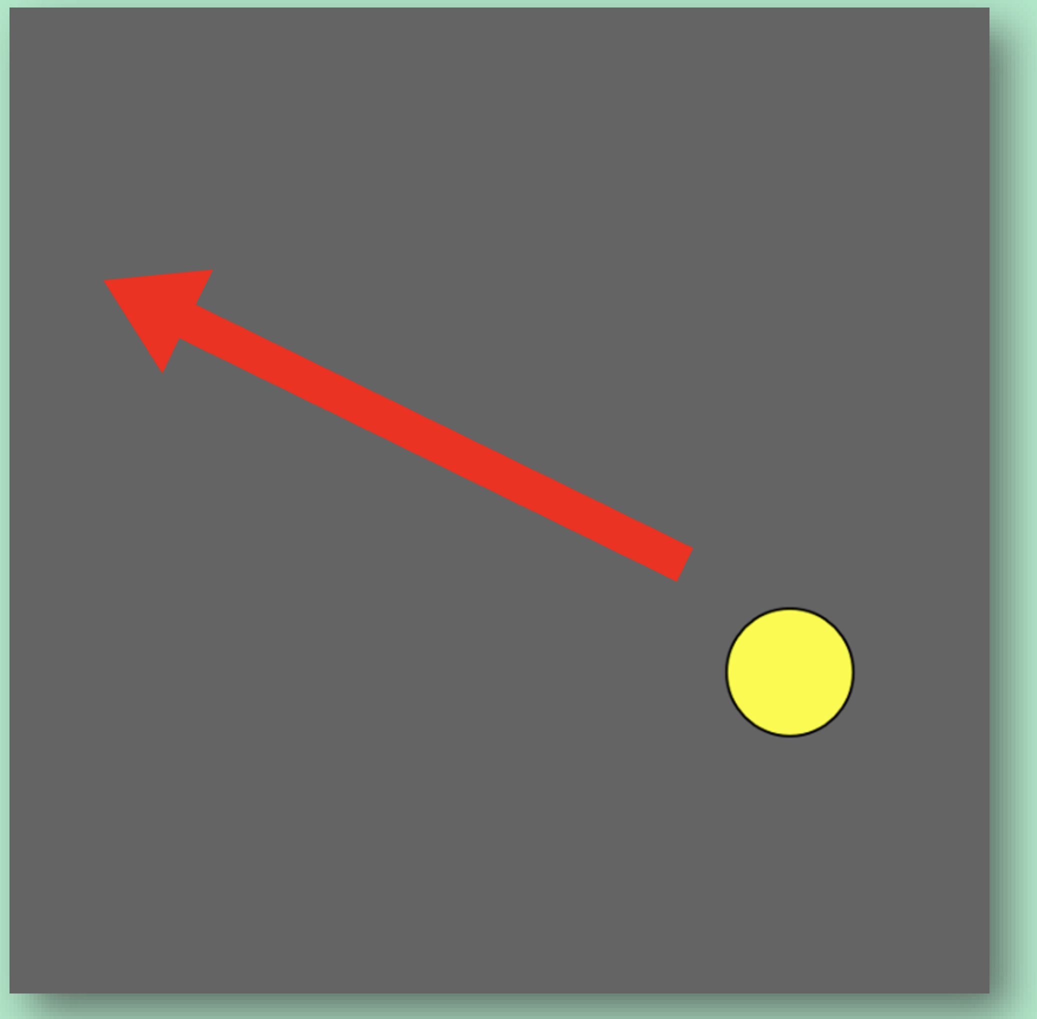


```
let x = 400;

function setup() {
  createCanvas(windowWidth, windowHeight);
}

function draw() {
  background(100);
  fill(255, 255, 0);
  circle(x, 200, 50);
  x = x - 5;
}
```

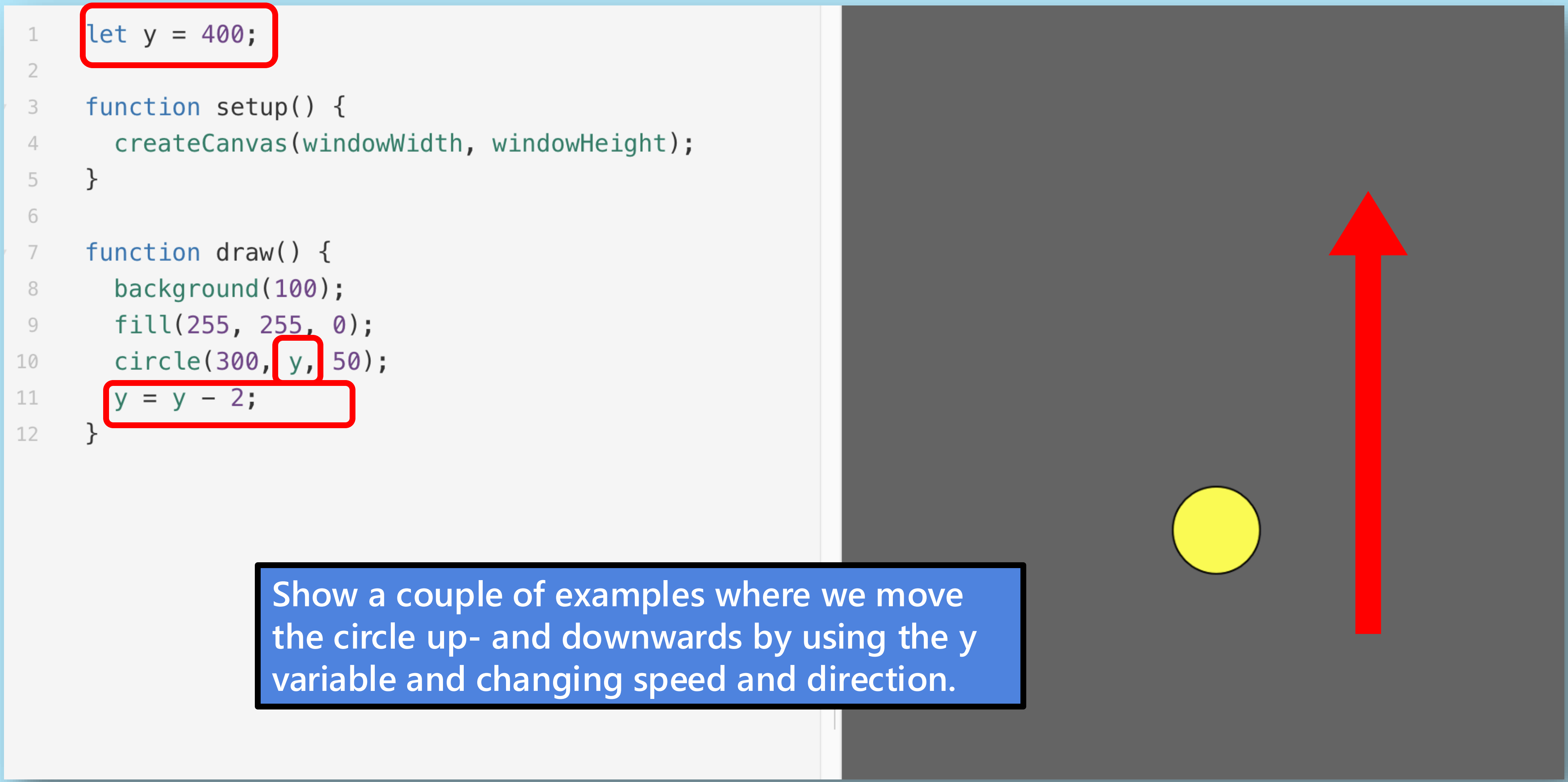
By increasing the number that we add to or deduct from x we can increase the speed

How can we move up & down & diagonally?





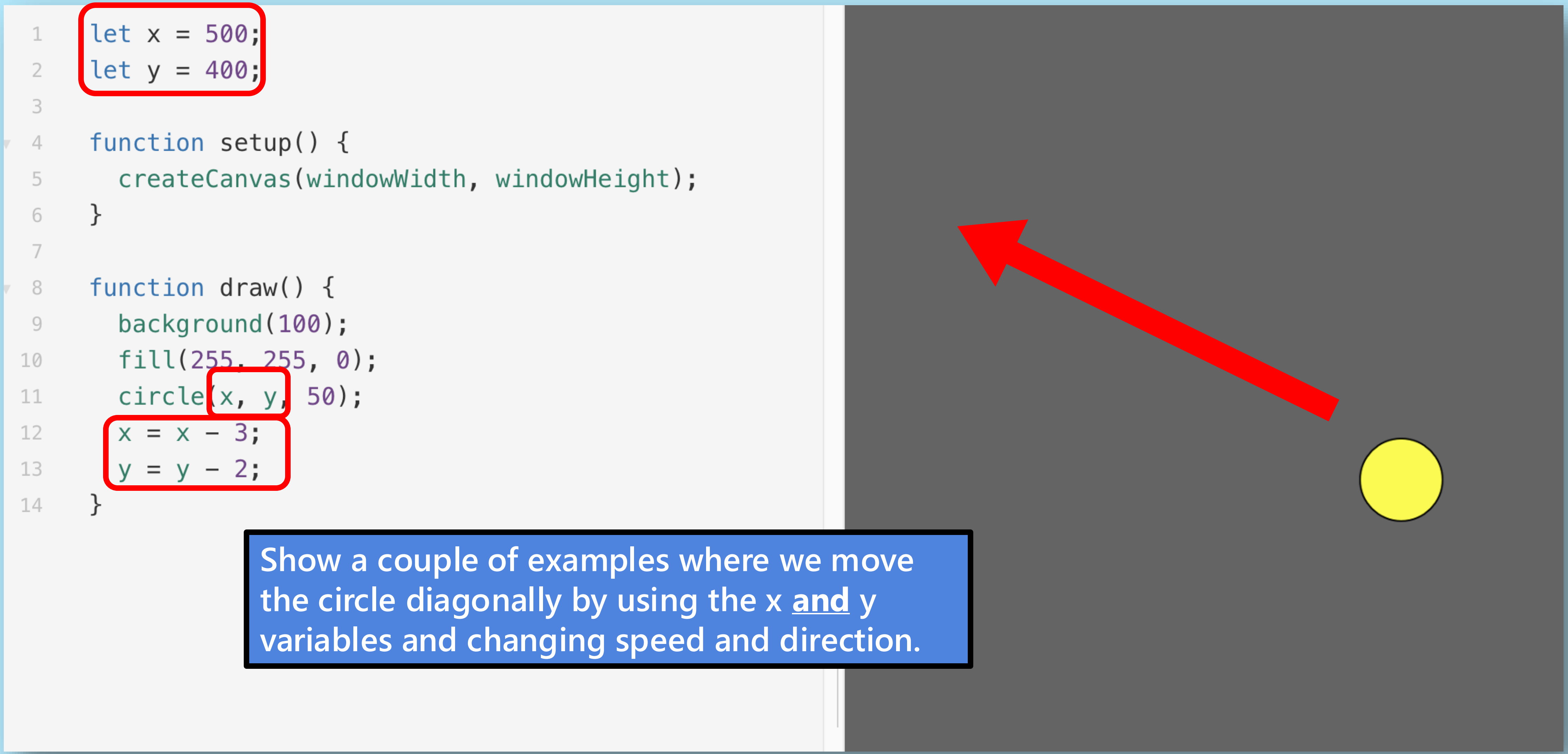
Up & down: change y direction



```
let y = 400;

function setup() {
  createCanvas(windowWidth, windowHeight);
}

function draw() {
  background(100);
  fill(255, 255, 0);
  circle(300, y, 50);
  y = y - 2;
}
```

Show a couple of examples where we move the circle up- and downwards by using the y variable and changing speed and direction.

Add x and y direction



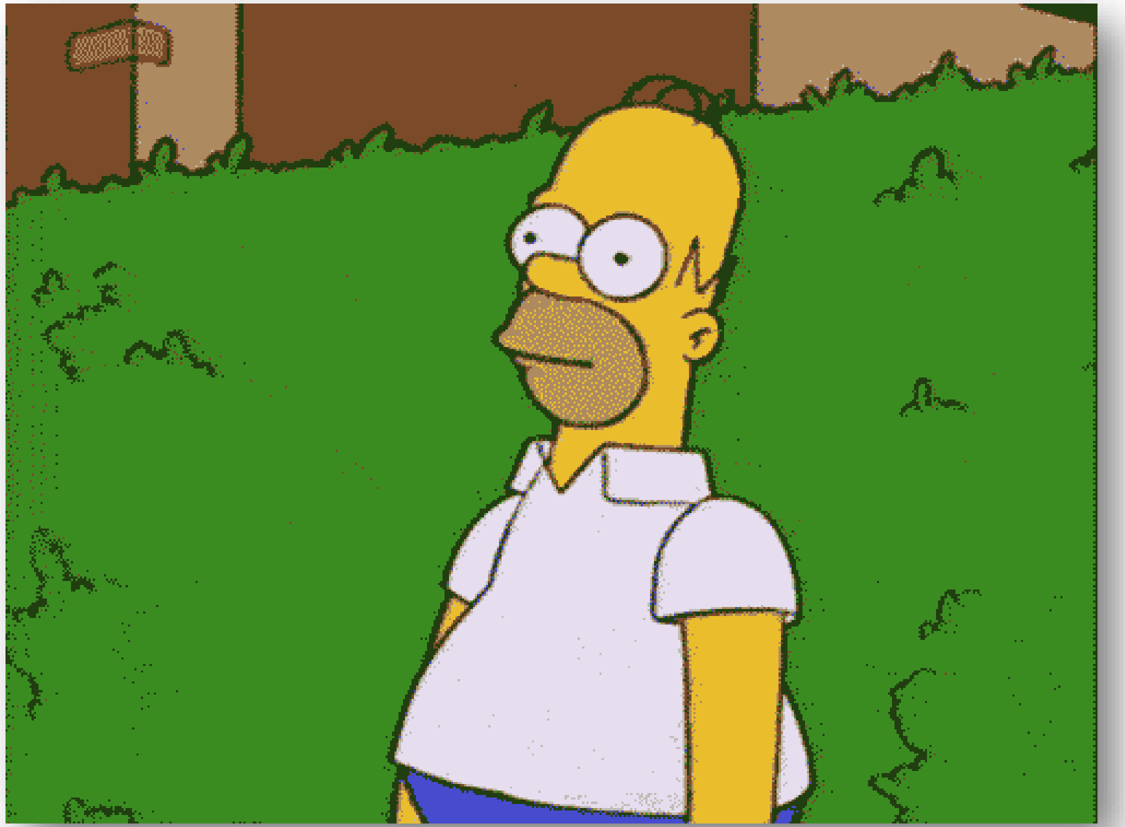
```
let x = 500;
let y = 400;

function setup() {
  createCanvas(windowWidth, windowHeight);
}

function draw() {
  background(100);
  fill(255, 255, 0);
  circle(x, y, 50);
  x = x - 3;
  y = y - 2;
}
```

Show a couple of examples where we move the circle diagonally by using the x **and** y variables and changing speed and direction.

But the circle always disappears

Now, we still have one major problem: the circle always disappears from the canvas…

To solve this, we need to know one more concept called **Conditionals**.



if.... else ....

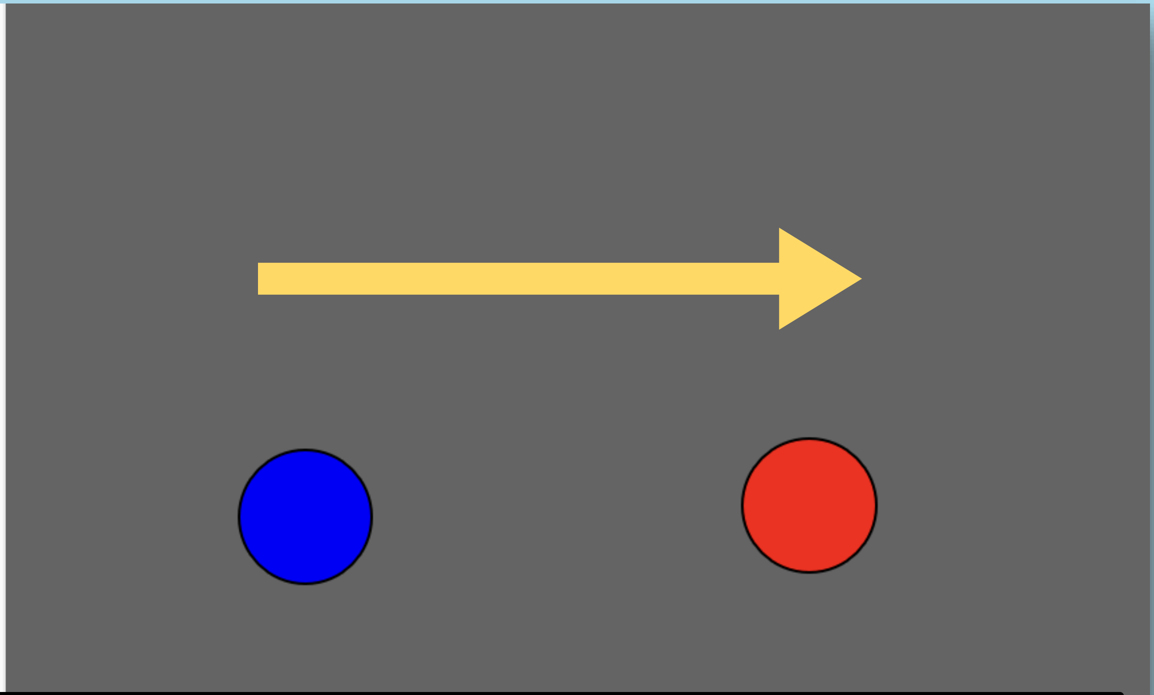
How to let the program decide?

Sometimes we want the computer to **make a decision**.

For example: “If the circle is on the left side, make it blue. Otherwise, make it red.”

Circle that changes colour based on position

```
let x = 100;

function setup() {
  createCanvas(windowWidth, windowHeight);
}

function draw() {
  background(100);

  if (x < 300) {
    fill(0, 0, 255);  // blue
  } else {
    fill(255, 0, 0);  // red
  }

  circle(x, 300, 50);
  x = x + 2;
}
```



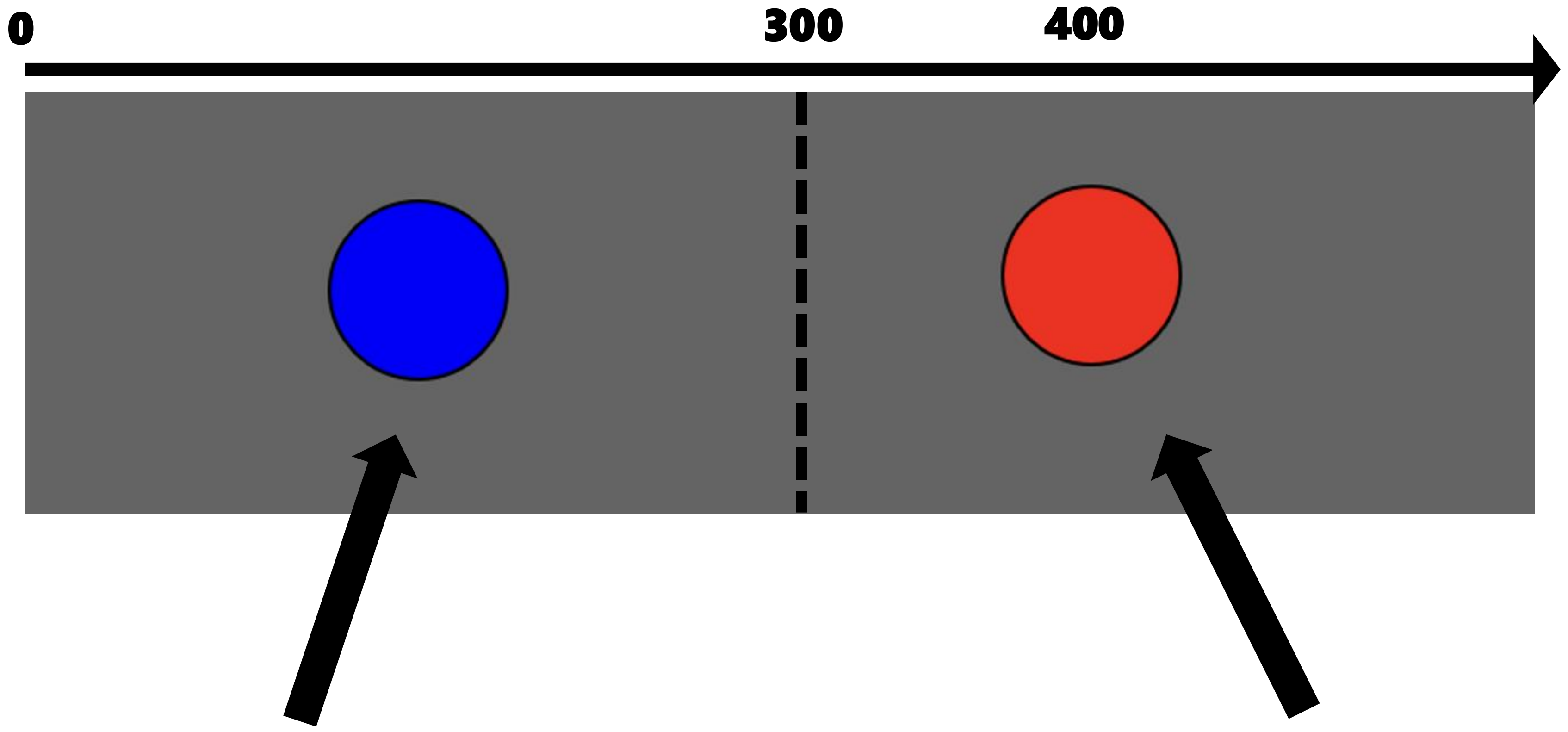
Circle starts blue and becomes red at 300px on the x axis.

Explain concepts:

- *if (...) { ... } else { ... }*
- only one of the blocks runs at a time
- Computer checks constantly "Is x less than 300?"

We check the x-position of the circle

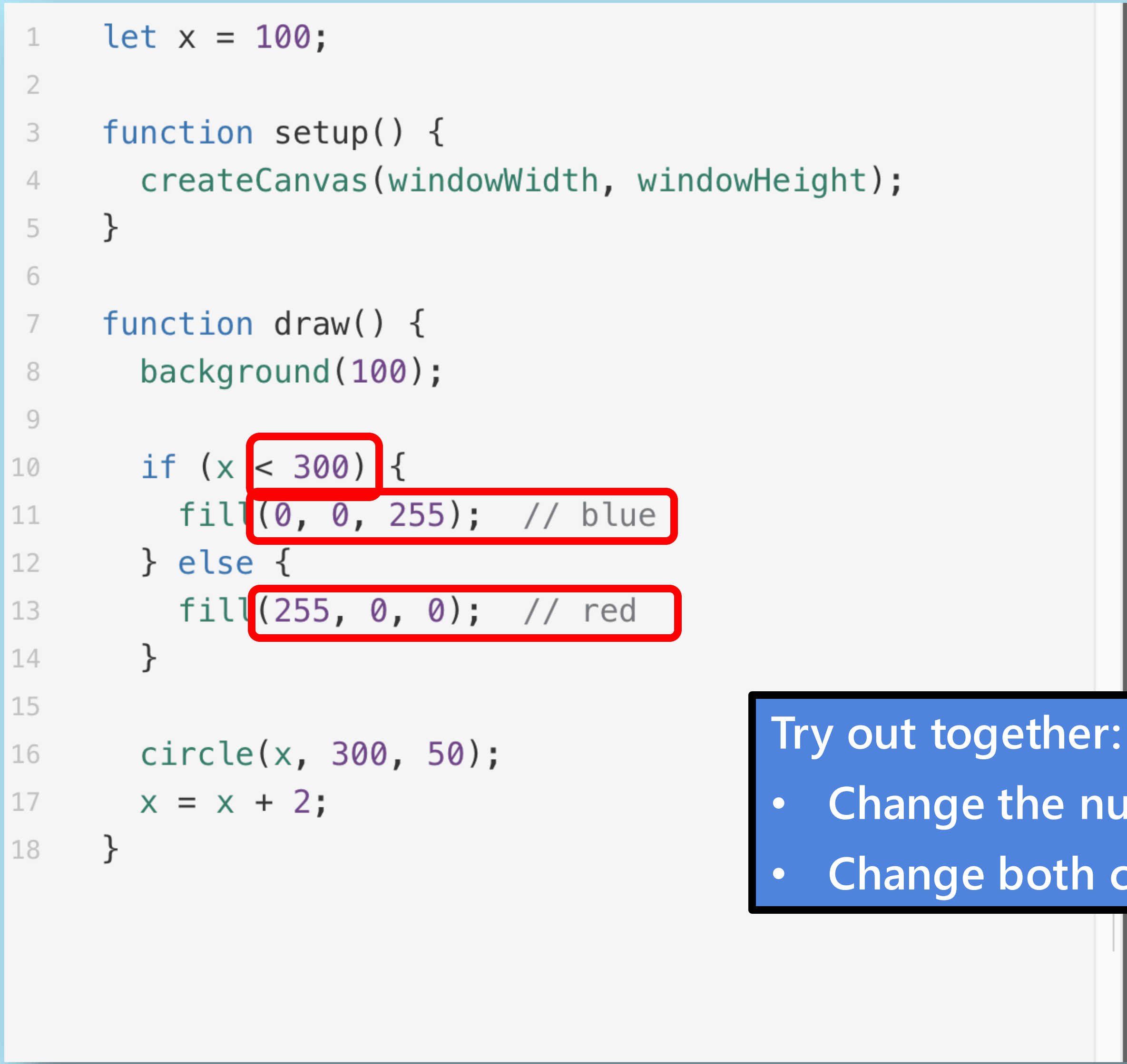
```
if (x < 300) {
  fill(0, 0, 255);  // blue
} else {
  fill(255, 0, 0);  // red
}
```



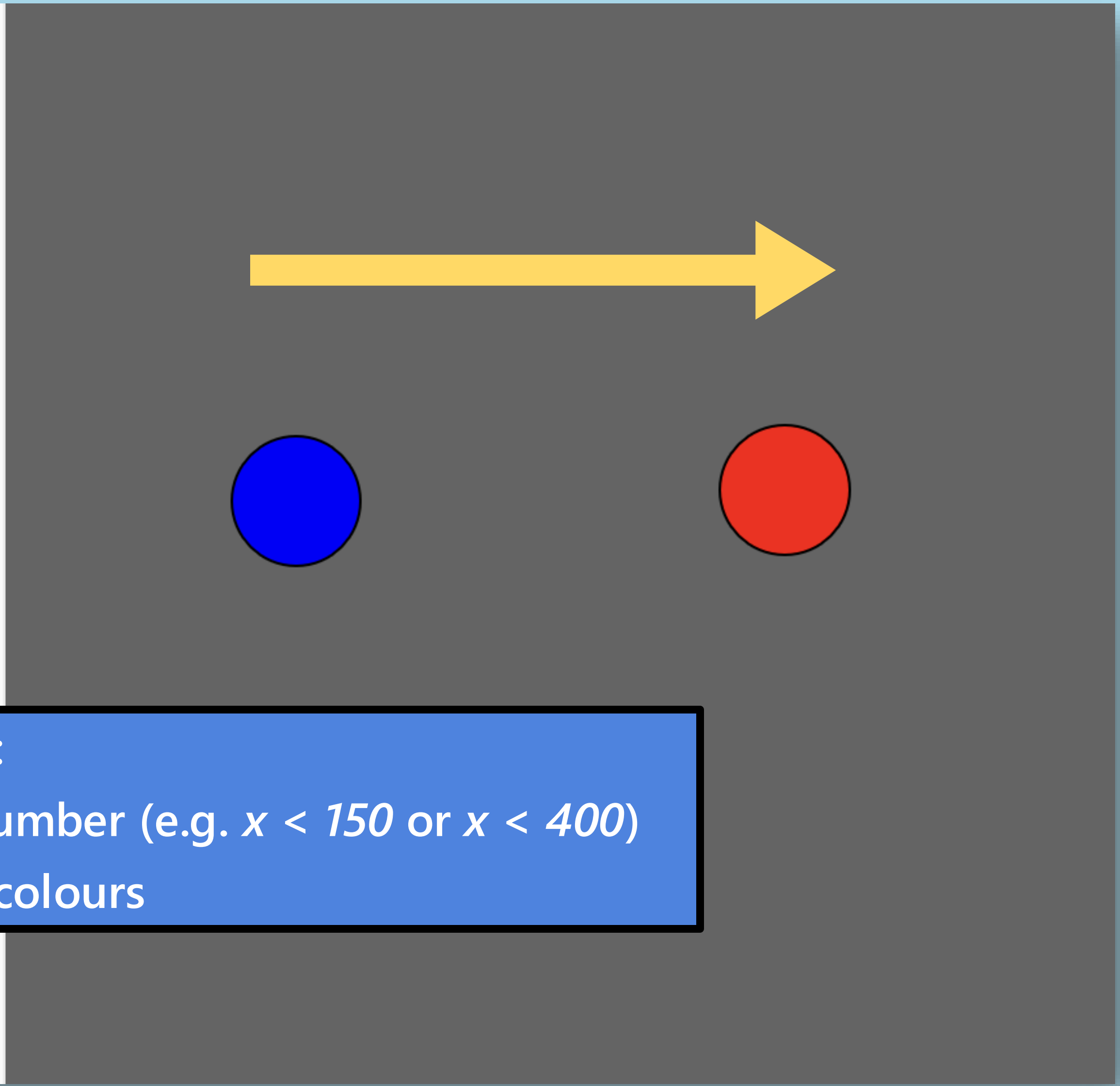
IF the x-value is smaller than 300: make the circle blue

ELSE = the x-value is 300 or larger: make the circle red

Circle that changes colour based on position



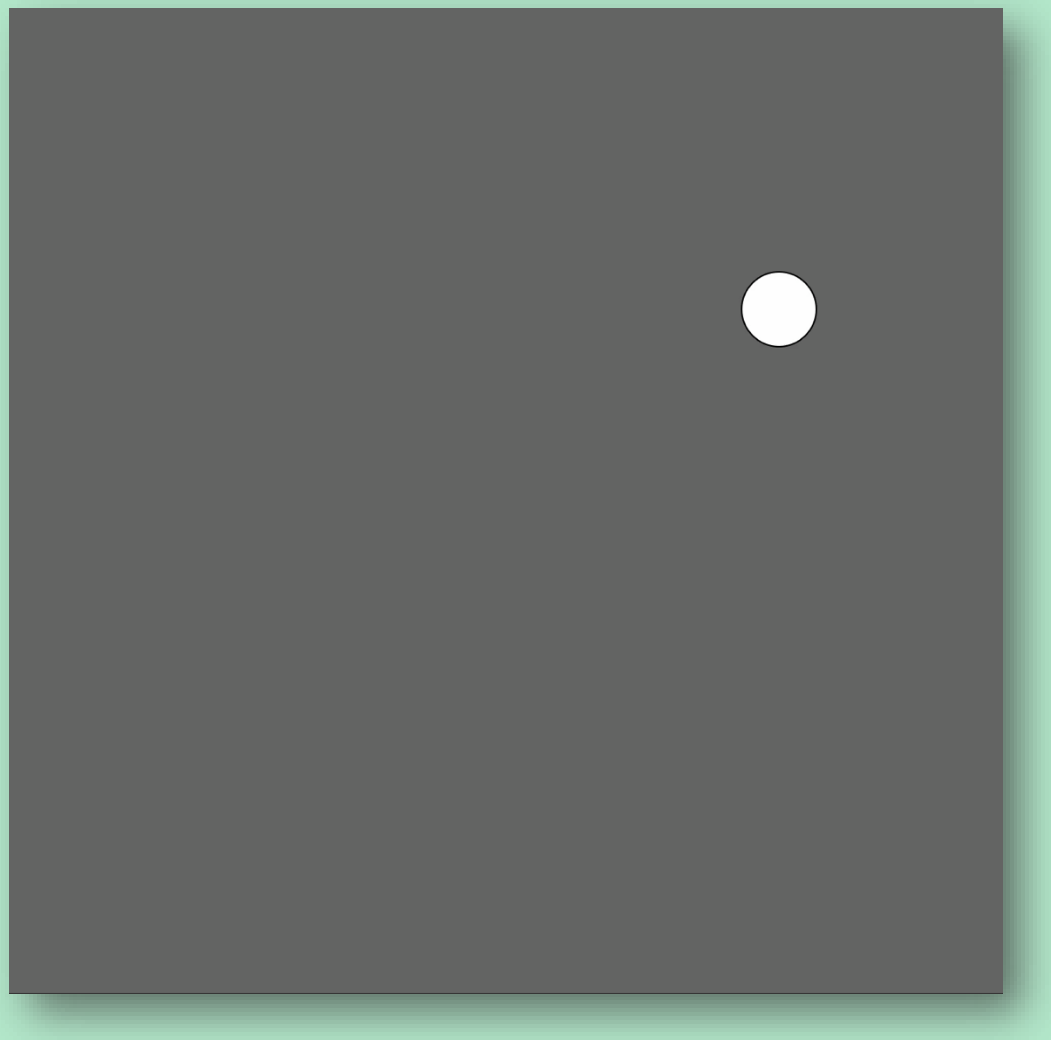
```
let x = 100;

function setup() {
  createCanvas(windowWidth, windowHeight);
}

function draw() {
  background(100);

  if (x < 300) {
    fill(0, 0, 255);   // blue
  } else {
    fill(255, 0, 0);   // red
  }

  circle(x, 300, 50);
  x = x + 2;
}
```



Try out together:

- Change the number (e.g. *x < 150* or *x < 400*)
- Change both colours

How can we apply this to bounce off the edges?



How to ask question in code

`>`	greater than
`<`	less than
`===`	equal to
`!==`	not equal
`&&`	AND (both most be true)
`||`	OR (either can be true)

Comparison Operators

Logical Operators

Examples

7 > 4	✅	true
3 > 10	❌	false
5 < 9	✅	true
100 < 50	❌	false
10 === 10	✅	true
7 !== 7.5	✅	true

>	greater than
<	less than
===	equal to
!==	not equal
&&	AND (both most be true)
||	OR (either can be true)

True or False?

50 > 10	✓	true
12 < 6	✗	false
130 > 129	✓	true
4 !== 4.1	✓	true

>	greater than
<	less than
===	equal to
!==	not equal
&&	AND (both most be true)
||	OR (either can be true)

Examples

true || false	✓	true
false || false	✗	false
false || true	✓	true

If either is true, result is true.

true && true	✓	true
false && false	✗	false
true && false	✗	false

Both must be true at the same time.

>	greater than
<	less than
===	equal to
!==	not equal
&&	AND (both most be true)
||	OR (either can be true)

True or False?

>	greater than
<	less than
===	equal to
!==	not equal
&&	AND (both most be true)
||	OR (either can be true)

5 > 3 && 5 < 9 ✓ true

5 > 3 && 5 > 9 ✗ false

75 > 100 || 90 < 100 ✓ true

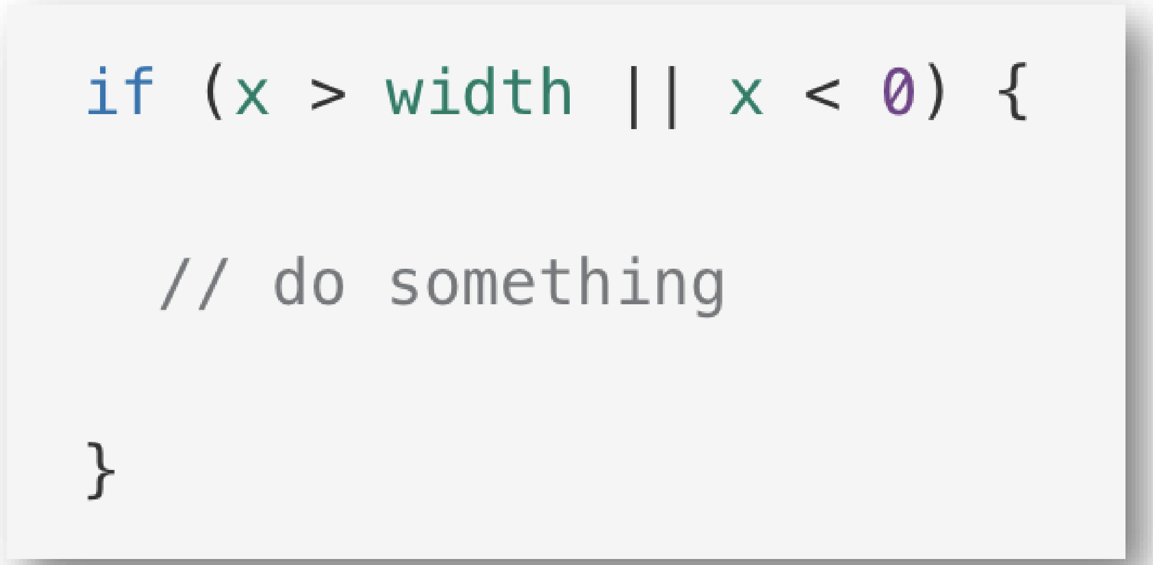
```
x = 50;
x > 100 || x < 100
```

✓ true

Why do we need operators?

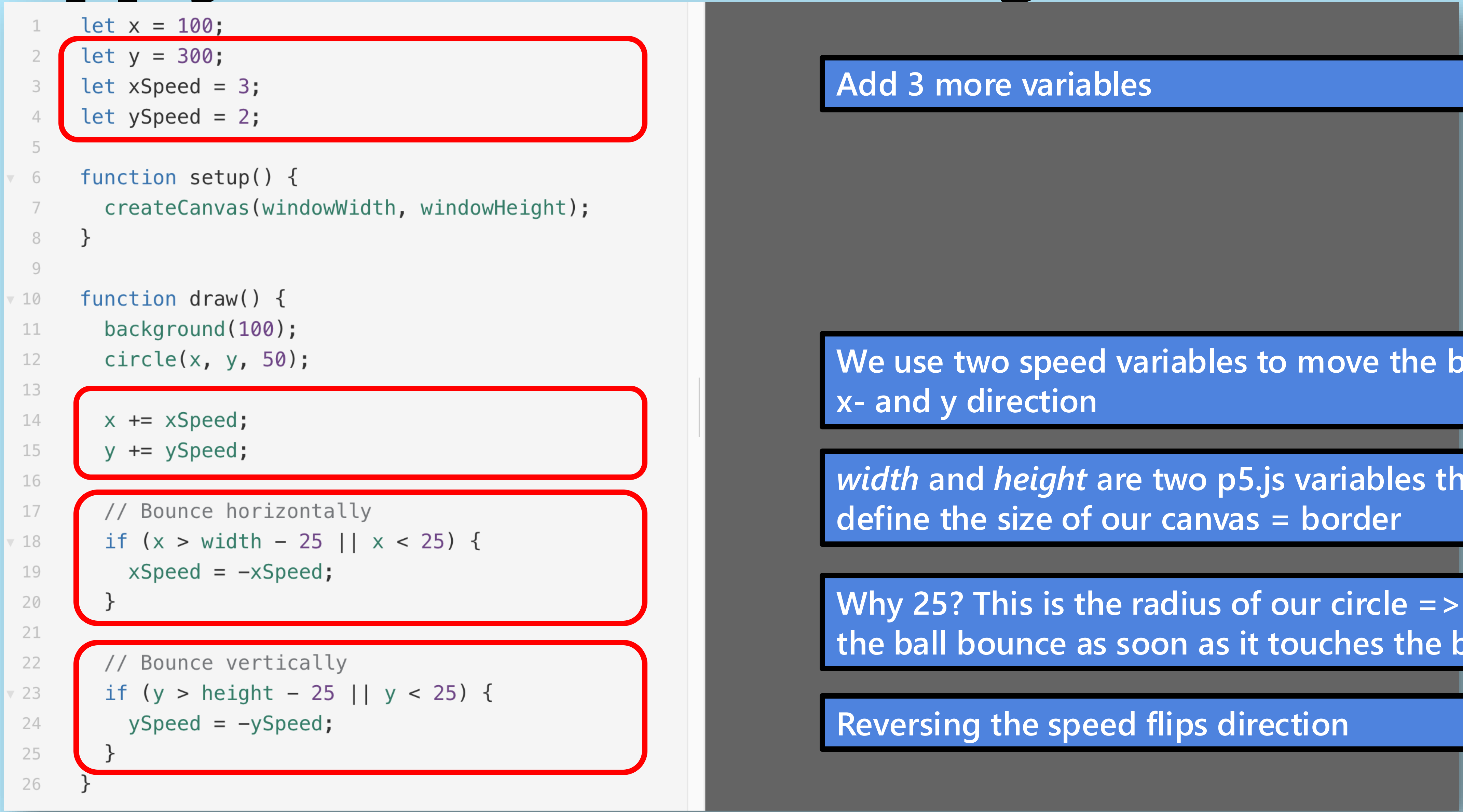
We use these logic checks to make the computer make decisions, like

- bouncing
- scoring points
- changing colors



```
if (x > width || x < 0) {

  // do something

}
```

Apply *if* to bounce off edges



```
let x = 100;
let y = 300;
let xSpeed = 3;
let ySpeed = 2;

function setup() {
  createCanvas(windowWidth, windowHeight);
}

function draw() {
  background(100);
  circle(x, y, 50);

  x += xSpeed;
  y += ySpeed;

  // Bounce horizontally
  if (x > width - 25 || x < 25) {
    xSpeed = -xSpeed;
  }

  // Bounce vertically
  if (y > height - 25 || y < 25) {
    ySpeed = -ySpeed;
  }
}
```

Add 3 more variables

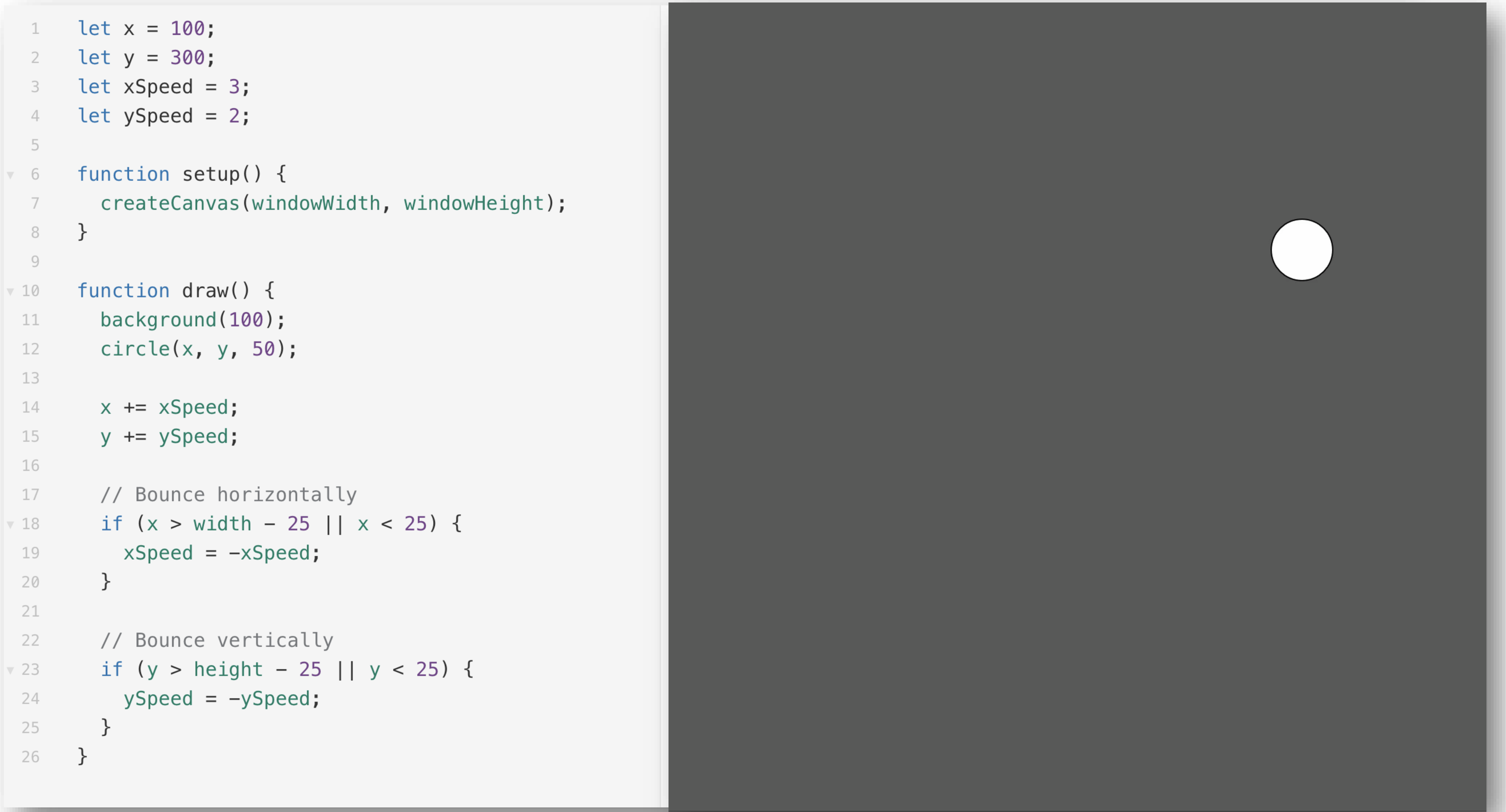
We use two speed variables to move the ball in x- and y direction

width and *height* are two p5.js variables that define the size of our canvas = border

Why 25? This is the radius of our circle => makes the ball bounce as soon as it touches the border.

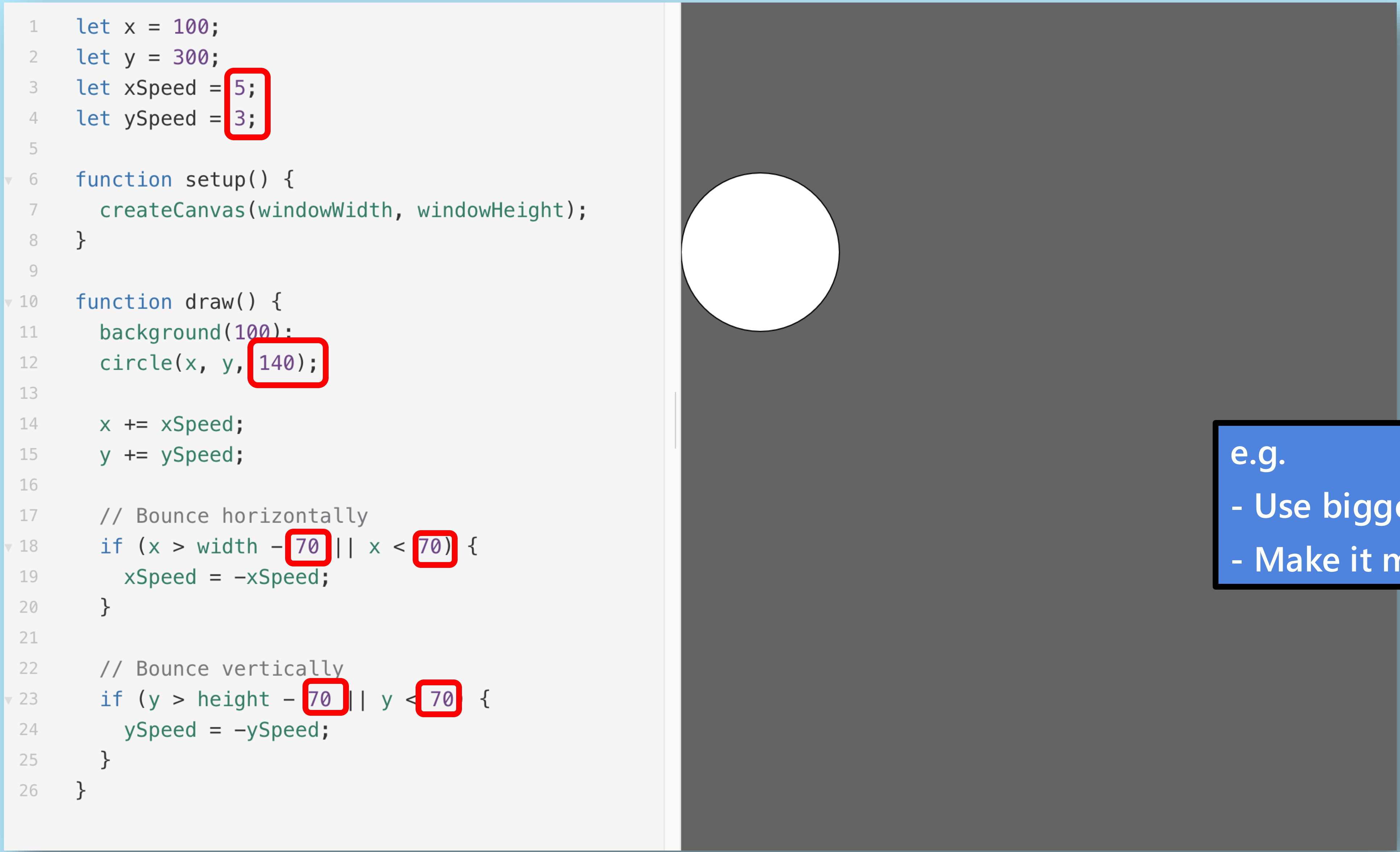
Reversing the speed flips direction

Apply *if* to bounce off edges



```
let x = 100;
let y = 300;
let xSpeed = 3;
let ySpeed = 2;

function setup() {
  createCanvas(windowWidth, windowHeight);
}

function draw() {
  background(100);
  circle(x, y, 50);

  x += xSpeed;
  y += ySpeed;

  // Bounce horizontally
  if (x > width - 25 || x < 25) {
    xSpeed = -xSpeed;
  }

  // Bounce vertically
  if (y > height - 25 || y < 25) {
    ySpeed = -ySpeed;
  }
}
```

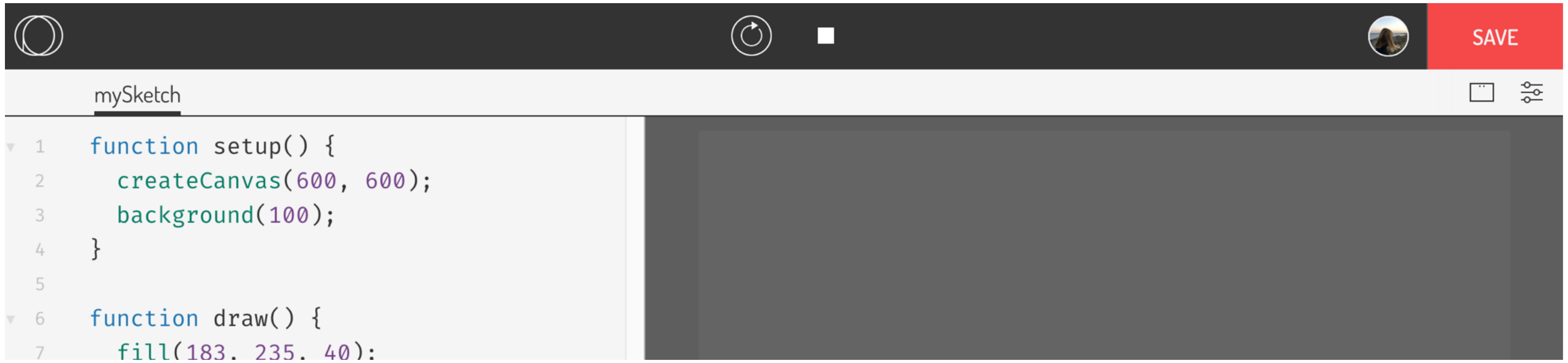
Variations



```
let x = 100;
let y = 300;
let xSpeed = 5;
let ySpeed = 3;

function setup() {
  createCanvas(windowWidth, windowHeight);
}

function draw() {
  background(100);
  circle(x, y, 140);

  x += xSpeed;
  y += ySpeed;

  // Bounce horizontally
  if (x > width - 70 || x < 70) {
    xSpeed = -xSpeed;
  }

  // Bounce vertically
  if (y > height - 70 || y < 70) {
    ySpeed = -ySpeed;
  }
}
```

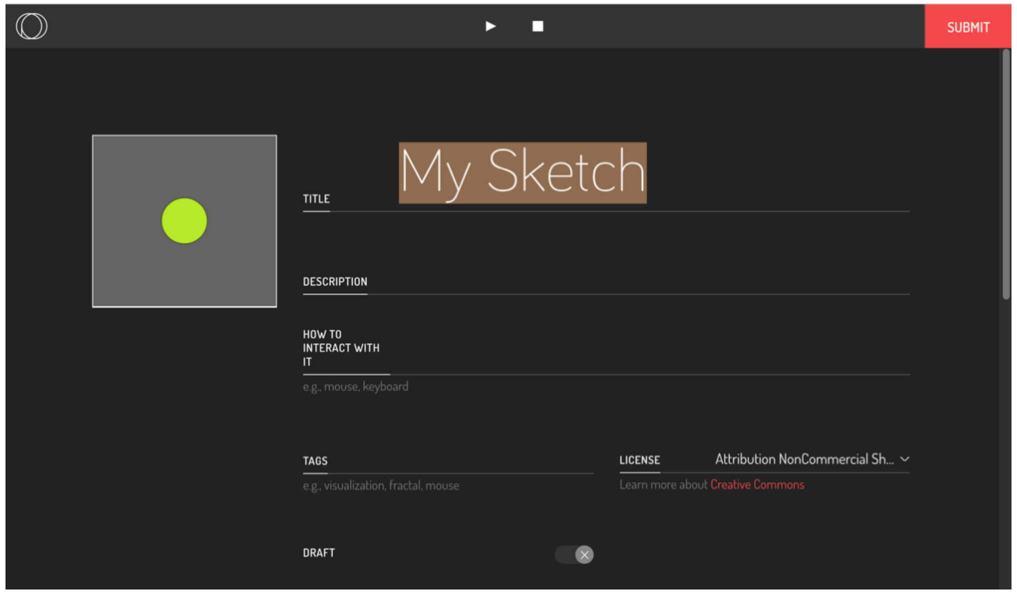
e.g.
- Use bigger / smaller circle
- Make it move faster / slower

Save



```
function setup() {
  createCanvas(600, 600);
  background(100);
}

function draw() {
  fill(183, 235, 40);
```

1 Click on *SAVE*


SUBMIT
TITLE
My Sketch
DESCRIPTION
HOW TO INTERACT WITH IT
e.g. mouse, keyboard
TAGS
e.g. visualization, fractal, mouse
LICENSE
Attribution NonCommercial Sh...
Learn more about Creative Commons
DRAFT

3 Click on *SUBMIT*

2 Give the sketch a new title instead of *„My Sketch"* e.g. *„Circles"*

Challenge – Customize your Sketch

1.🌈 Change colour on bounce:

e.g. with

**fill(random(255),
    random(255),
    random(255));**

```
fill(random(255), random(255), random(255));
```

2. 👻 Leave a trail (fading background):

e.g. with

background(220,20);

```
background(220, 20);
```

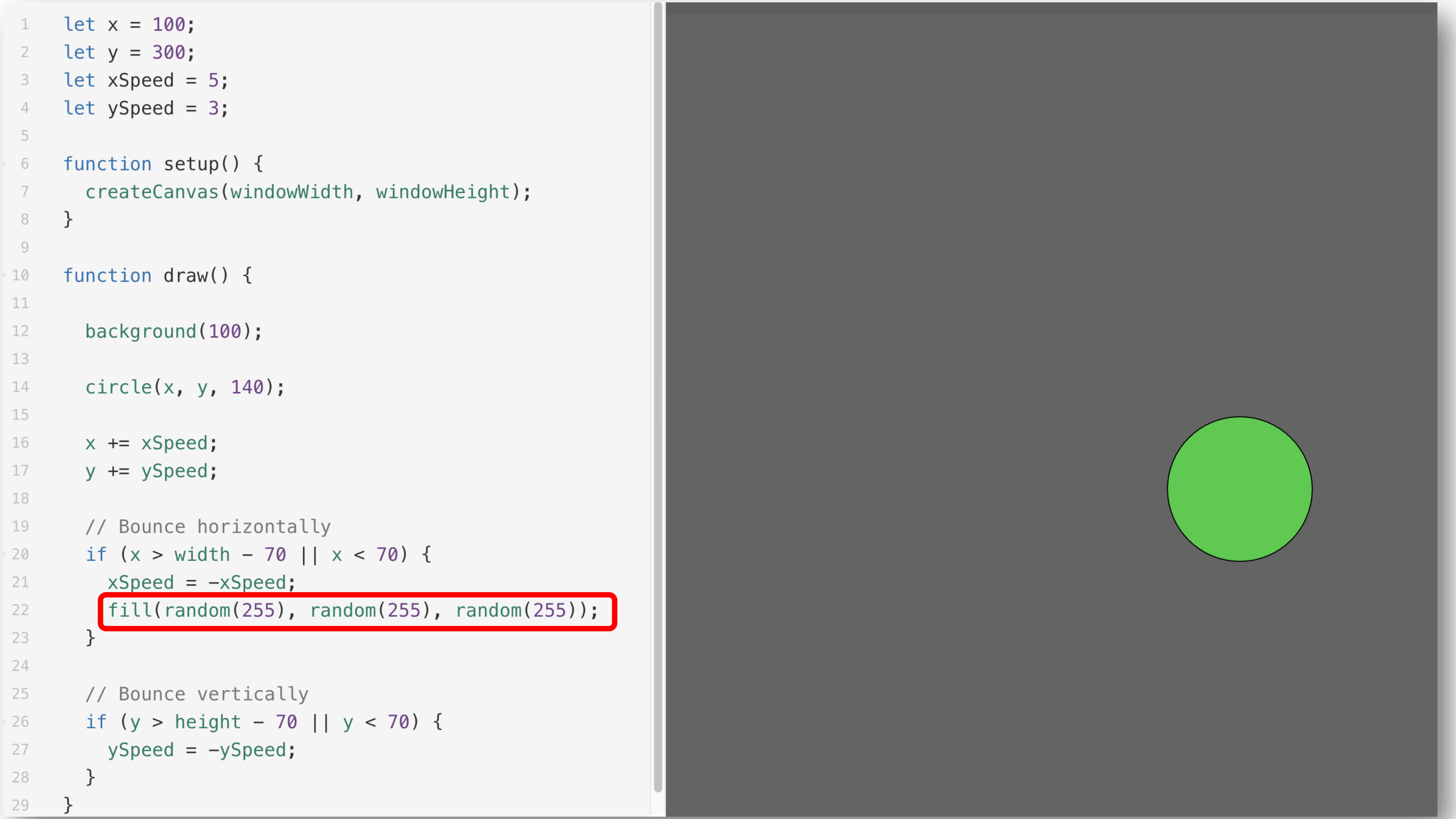
3. 👽 Replace circle with another shape:

e.g. with

```
fill(100, 255, 100);
arc(x, y, 80, 60, 0, PI); // body

fill(0);
circle(x - 20, y - 20, 10); // eye
circle(x + 20, y - 20, 10); // eye

line(x - 15, y - 5, x + 15, y - 5); // mouth
```

Examples for customization

1. Change colour on bounce



```javascript
let x = 100;
let y = 300;
let xSpeed = 5;
let ySpeed = 3;

function setup() {
  createCanvas(windowWidth, windowHeight);
}

function draw() {

  background(100);

  circle(x, y, 140);

  x += xSpeed;
  y += ySpeed;

  // Bounce horizontally
  if (x > width - 70 || x < 70) {
    xSpeed = -xSpeed;
    fill(random(255), random(255), random(255));
  }

  // Bounce vertically
  if (y > height - 70 || y < 70) {
    ySpeed = -ySpeed;
  }
}
```

random(*min, max*)

We will need the ***random( )*** function to generate random values.

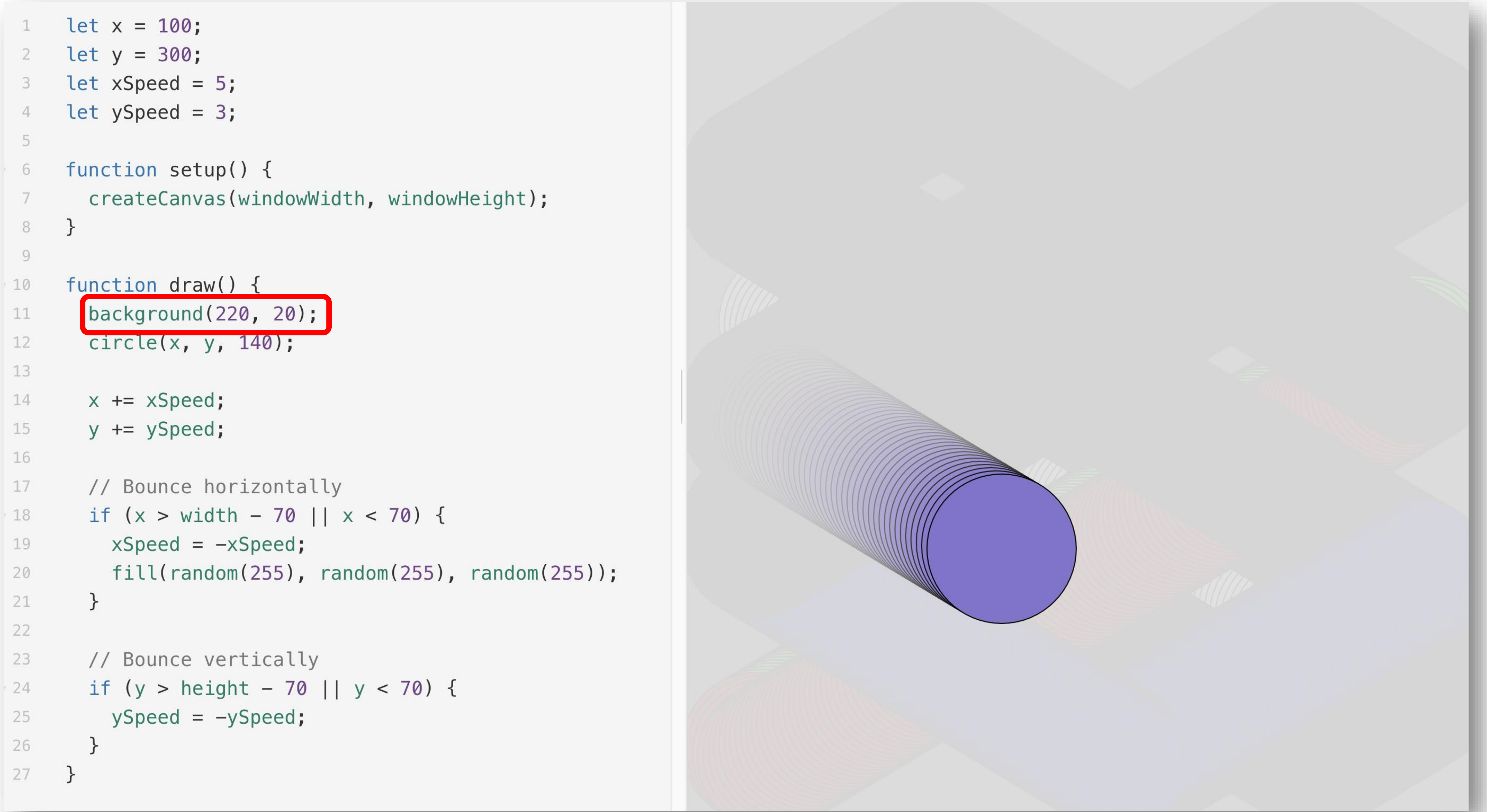
Example

```
random(0,255);
```

$\underbrace{\hphantom{random(0,255);}}$

generates a random value between 0 and 254

2. 👻 Leave a trail (fading background)



```
let x = 100;
let y = 300;
let xSpeed = 5;
let ySpeed = 3;

function setup() {
  createCanvas(windowWidth, windowHeight);
}

function draw() {
  background(220, 20);
  circle(x, y, 140);

  x += xSpeed;
  y += ySpeed;

  // Bounce horizontally
  if (x > width - 70 || x < 70) {
    xSpeed = -xSpeed;
    fill(random(255), random(255), random(255));
  }

  // Bounce vertically
  if (y > height - 70 || y < 70) {
    ySpeed = -ySpeed;
  }
}
```

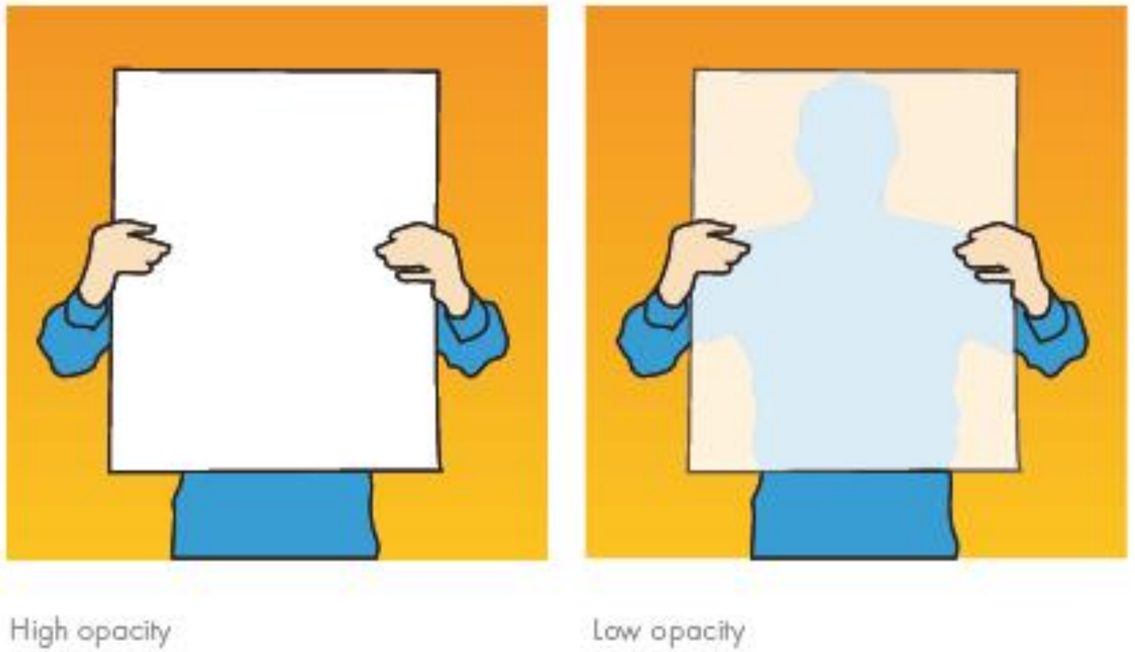
background(red,green,blue,opacity)

We already know the background() function to add a colour to an object.

Example

```
background(220,20);
```

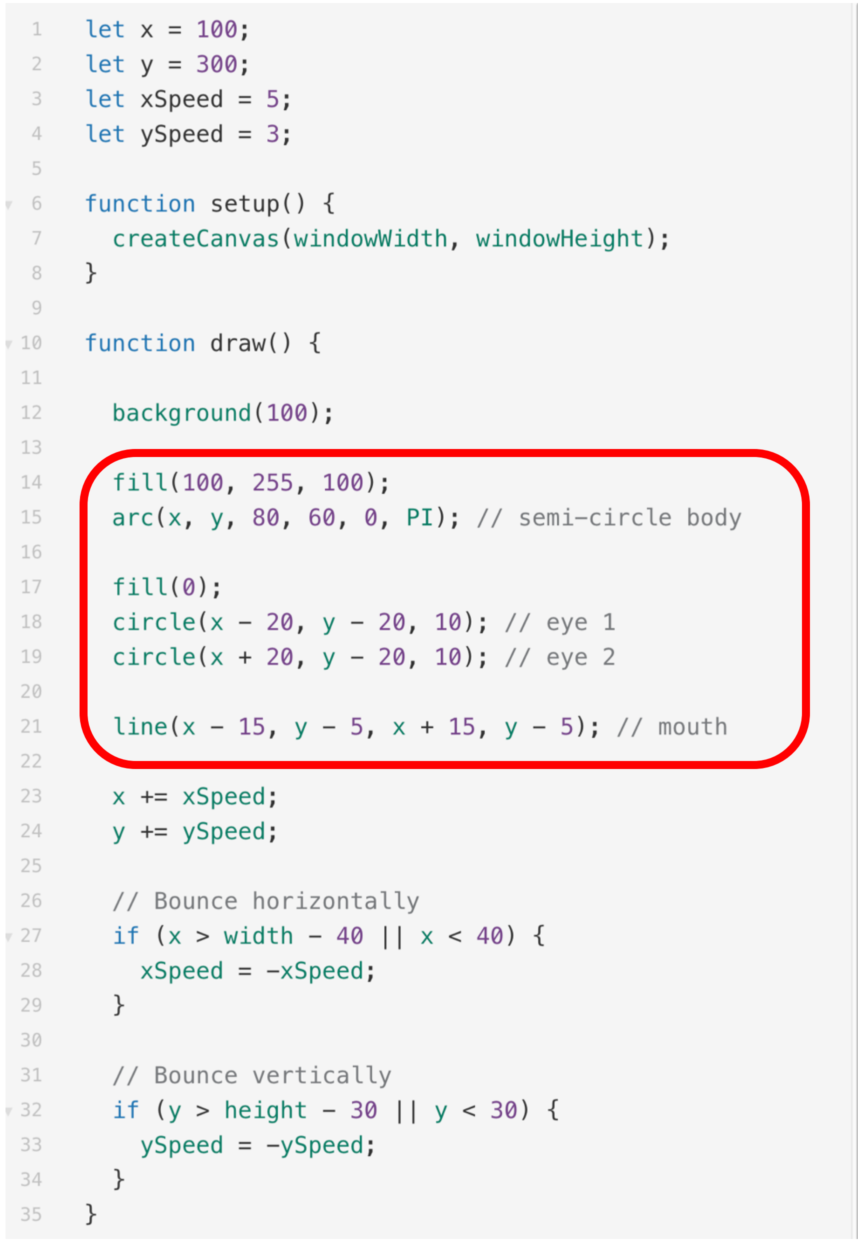
```
background(100,210,120,20);
```



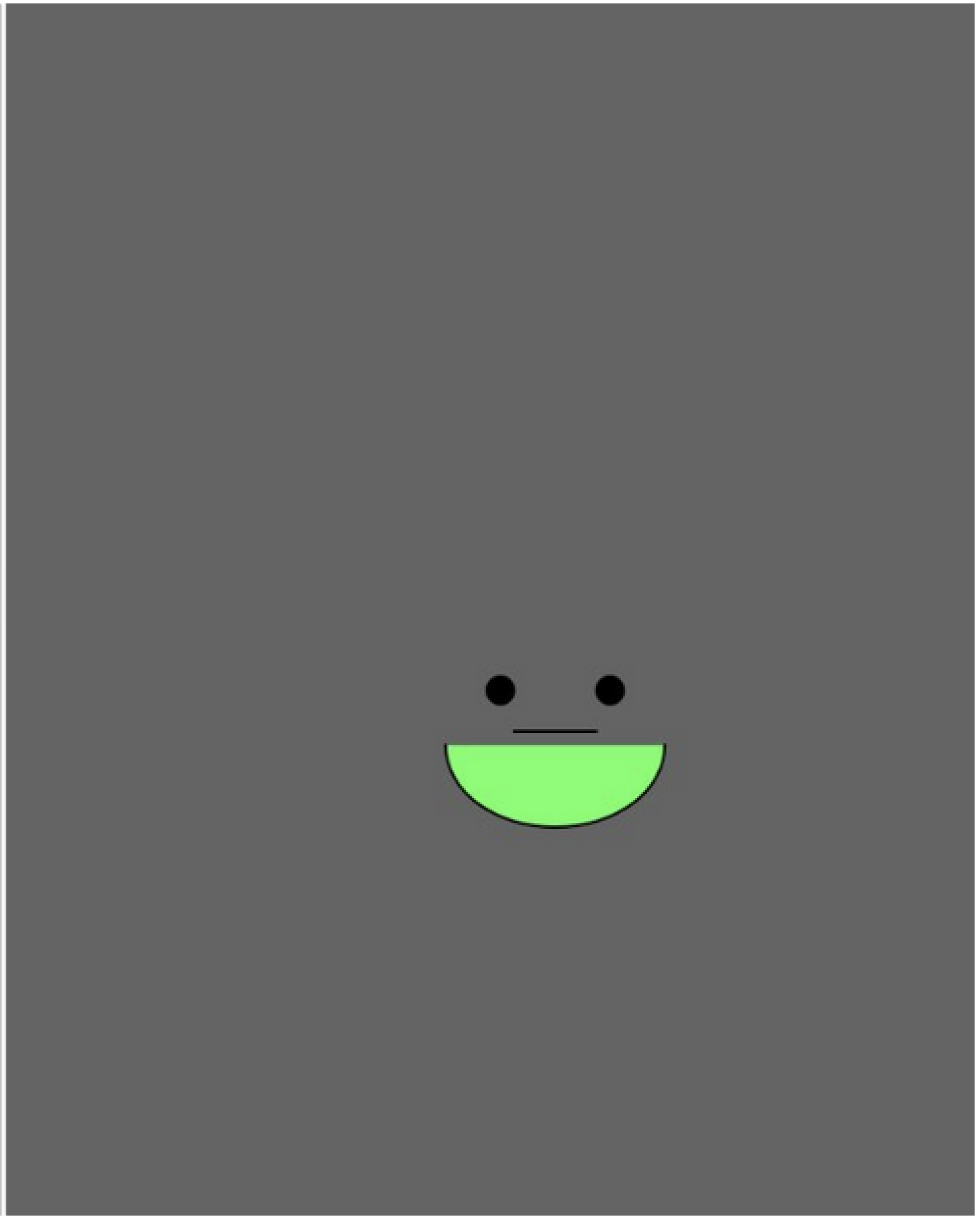
We can add a 2nd or 4th value between 0 (transparent) and 255 (opaque)

Source: https://www.precisecontinental.com/domtar-business-card-basics-1/

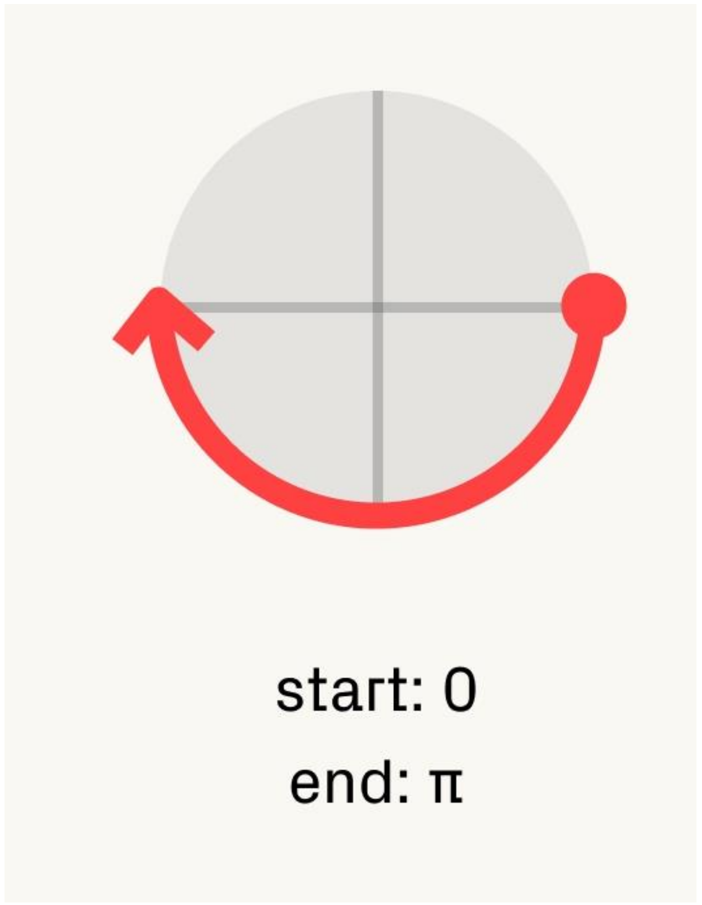
3. 👽 Replace circle with another shape



```
let x = 100;
let y = 300;
let xSpeed = 5;
let ySpeed = 3;

function setup() {
  createCanvas(windowWidth, windowHeight);
}

function draw() {

  background(100);

  fill(100, 255, 100);
  arc(x, y, 80, 60, 0, PI); // semi-circle body

  fill(0);
  circle(x - 20, y - 20, 10); // eye 1
  circle(x + 20, y - 20, 10); // eye 2

  line(x - 15, y - 5, x + 15, y - 5); // mouth

  x += xSpeed;
  y += ySpeed;

  // Bounce horizontally
  if (x > width - 40 || x < 40) {
    xSpeed = -xSpeed;
  }

  // Bounce vertically
  if (y > height - 30 || y < 30) {
    ySpeed = -ySpeed;
  }
}
```



arc()



e.g.

arc(x, y, 80, 60, 0, PI);

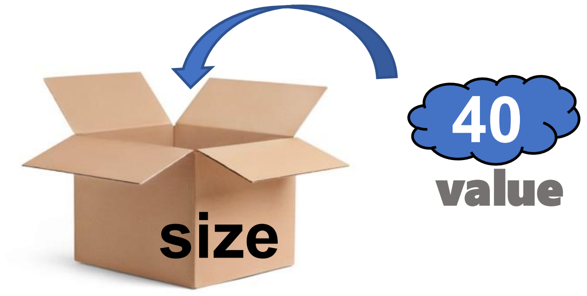
```
arc(x, y, w, h, start, stop, [mode], [detail])
```

Recap

3 ways of using variables

1. Create a Variable:

define name and value at the top of your code



```
let size = 40;
```

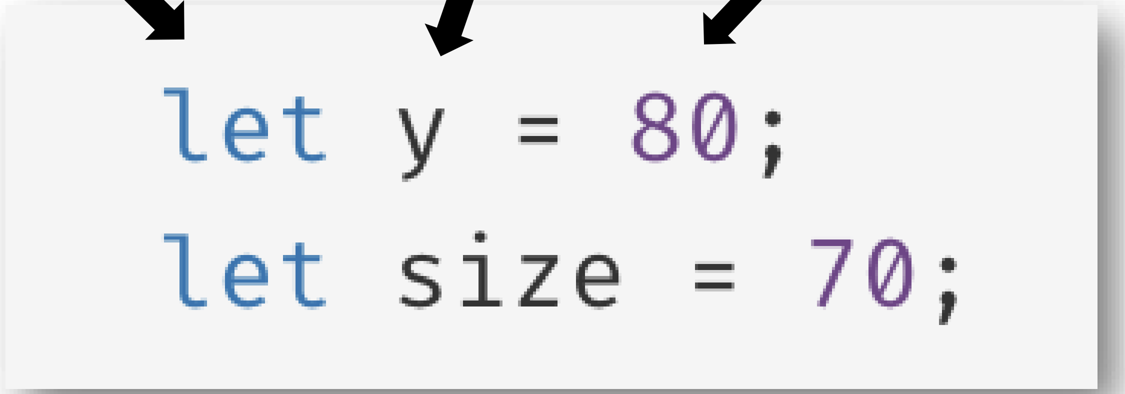
How to create variables

give it a name

let is written so that the computer knows we are creating a new variable

give it a value

This doesn't have to be a number, it could also be letters (string) or true/false.

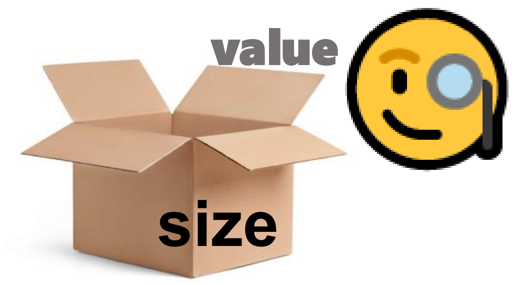


```
let y = 80;
let size = 70;
```

3 ways of using variables

2. Read a Variable:

use the variable to put its value into your code

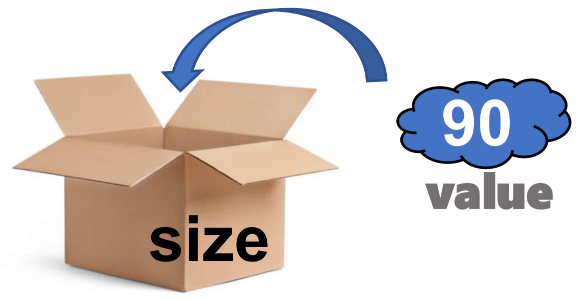


```
circle(100, 80, size);
```

3 ways of using variables

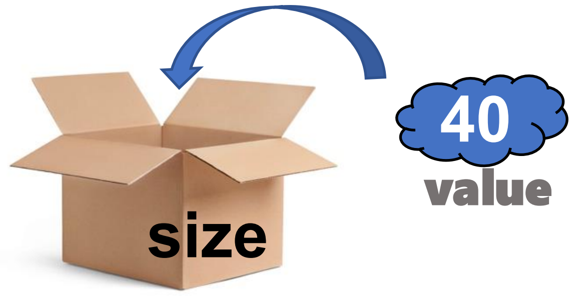
3. Change a Variable:

put a new value into the variable



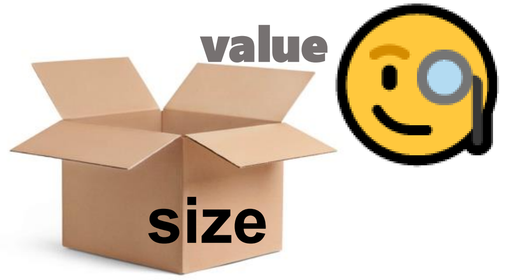
```
size = 90;
```

3 ways of using variables



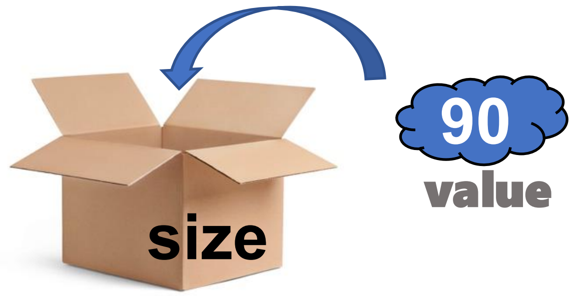
Create define name and value at the top of your code

```
let size = 40;
```



Read Use the variable to put its value into your code

```
circle(100, 80, size);
```



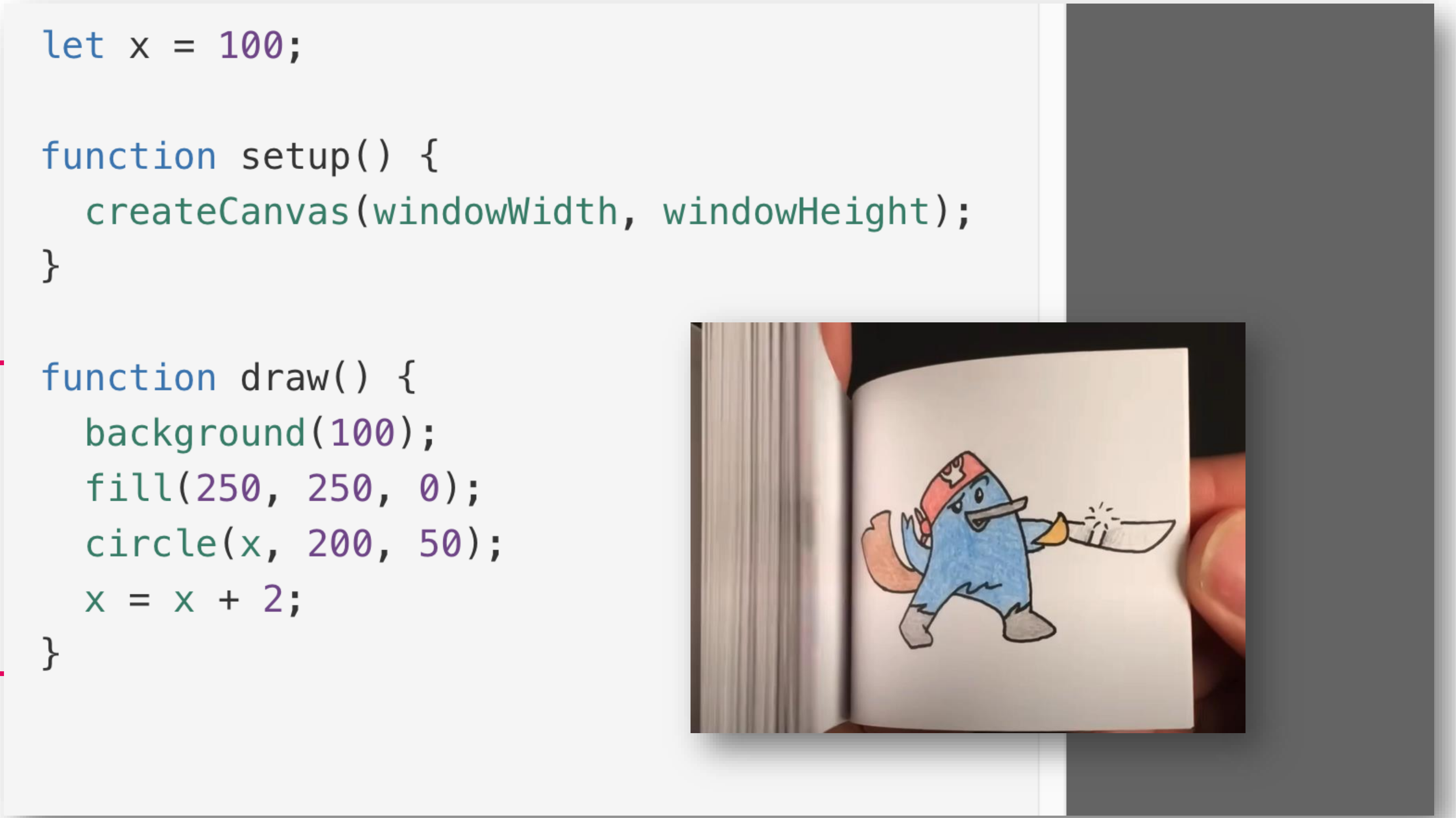
Change Put a new value into the variable

```
size = 90;
```

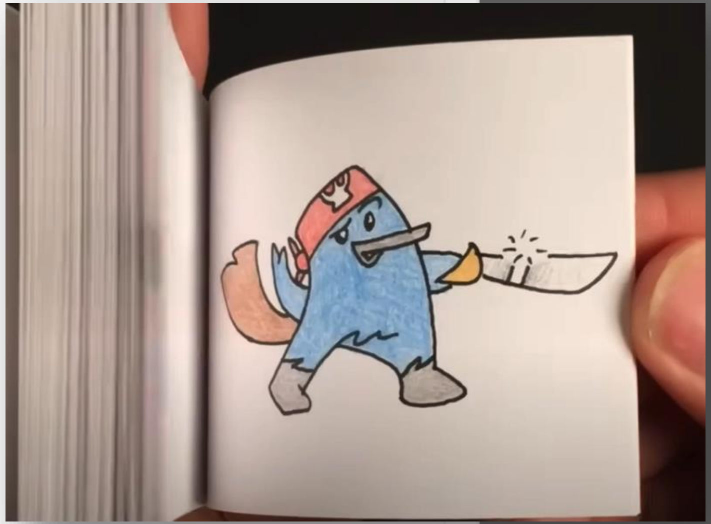
The Magic of draw()

happens only 1x

happens 60x per second

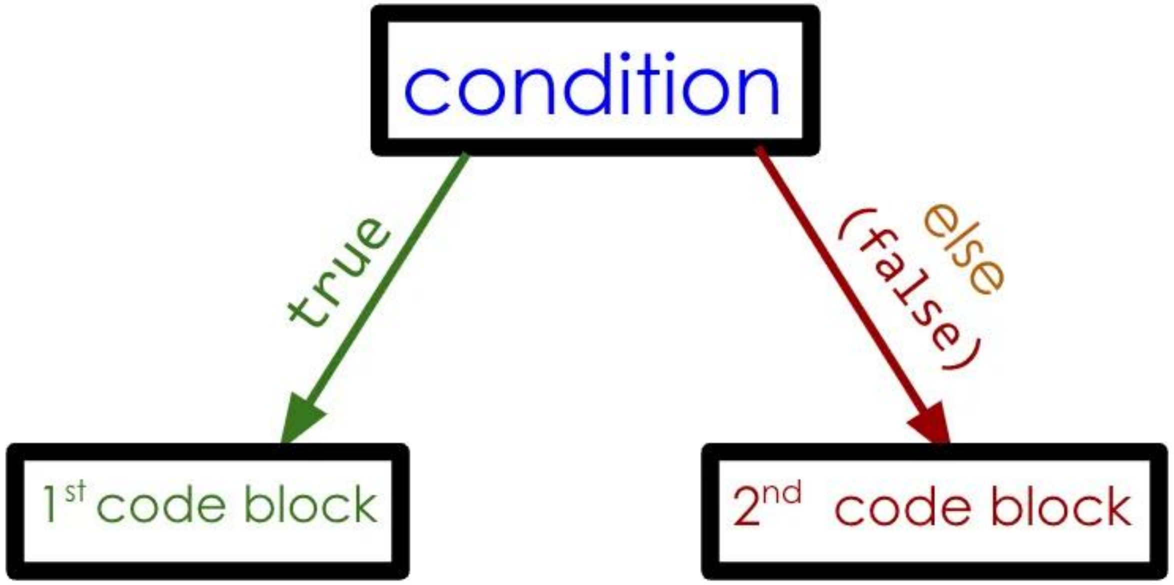


```
let x = 100;

function setup() {
  createCanvas(windowWidth, windowHeight);
}

function draw() {
  background(100);
  fill(250, 250, 0);
  circle(x, 200, 50);
  x = x + 2;
}
```

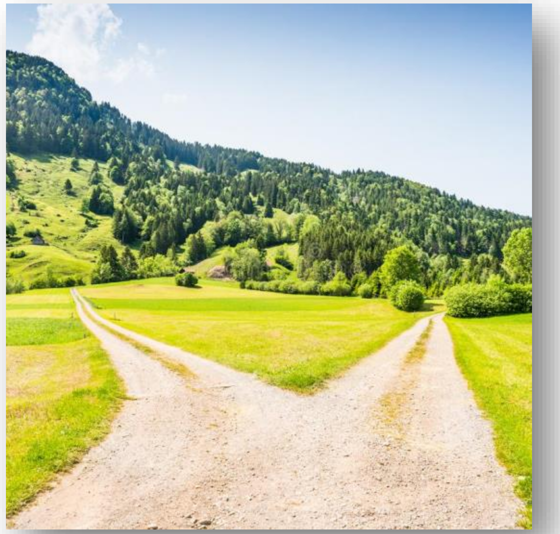


Different paths with *if...else...*

How our programs can follow different paths.



```
if (condition) {
  // code to run if the condition is true
} else {
  // code to run if the condition is false
}
```



Luxembourg Tech School

innovative initiatives
digital luxembourg.

SCRIPT
Service de Coordination de la Recherche et de l'Innovation pédagogiques et technologiques

cgié
centre de gestion informatique de l'éducation

LUXEMBOURG SPACE AGENCY

Luxembourg National Research Fund

andré losch
FONDATION

FONdation ABBL
pour l'éducation financière

STATE STREET

funded by
œuvre nationale

Luxair

the LHoFT

Luxembourg-City Incubator