http://dextro.org

Programming Platforms

VA345 Creative Coding

Course Instructor : Assoc. Prof. Dr. Selcuk ARTUT

Food for thought : Casey Reas

Computer Programming (Coding – sounds cool)

Software

Rule based systems

Earliest examples Modus ponens

If you believe that A is the case, and believe that A implies B, then you ought to believe that B is the case

Rituals, laws, political systems etc

Modus ponens

From Wikipedia, the free encyclopedia

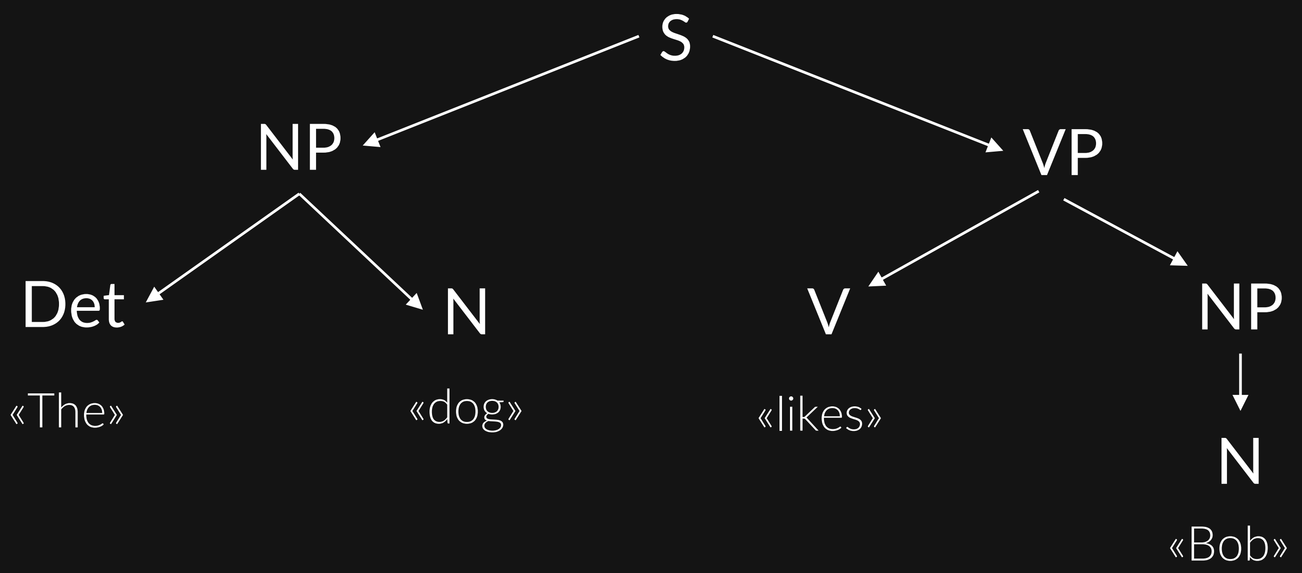
In propositional logic, ***modus ponendo ponens*** (Latin for "the way that affirms by affirming"; generally abbreviated to **MP** or ***modus ponens***[1]) or **implication elimination** is a rule of inference.[2] It can be summarized as "*P* implies *Q* and *P* are both asserted to be true, so therefore *Q* must be true." The history of *modus ponens* goes back to antiquity.[3]

Rule based systems

- Formal Languages vs Natural Languages
- It is easier to define a formal language than a natural language, because formal language is synthetic, expressive, compact etc.
- Language = grammar + dictionary
- Computer languages : Backus-Naur form (1959) - Algol60

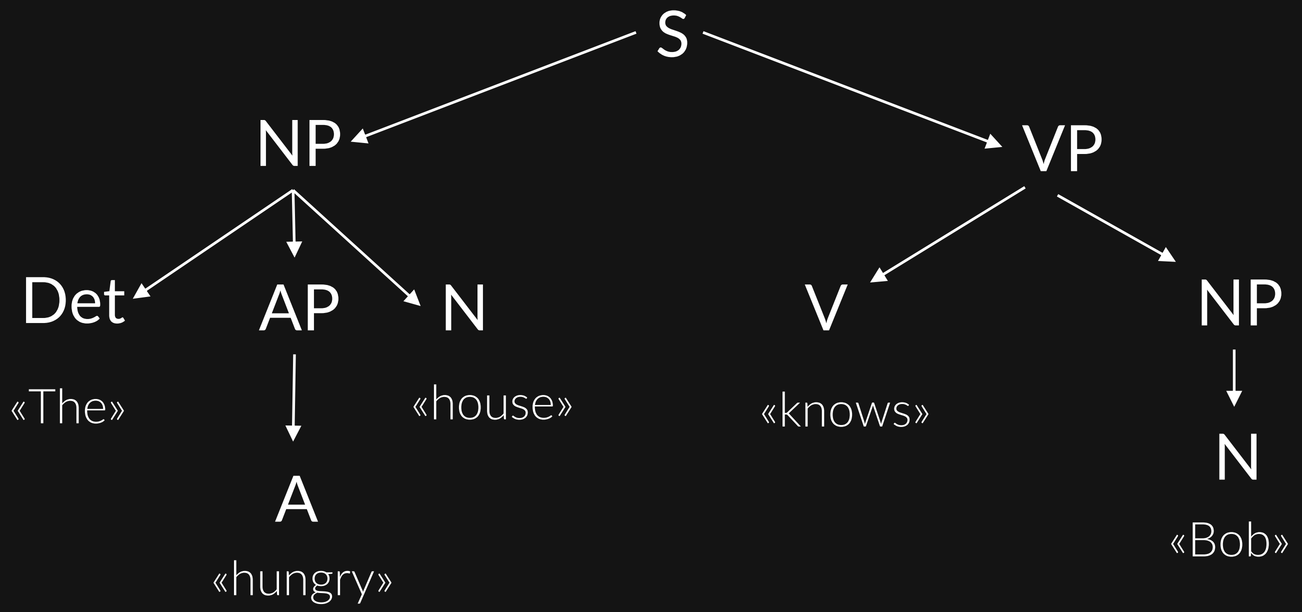
Generative Grammar for Syntax

- We can formalize a syntax of a language using a grammar



S : Sentence
NP : Noun phrase
Det : Determinant
N : Noun
V : Verb
VP : Verbal Phrase

A sentence can be syntaxically correct but semantically wrong



S

NP VP

Det AP N V NP

«The» «house» «knows»

A N

«hungry» «Bob»

S : Sentence
NP : Noun phrase
Det : Determinant
N : Noun
V : Verb
VP : Verbal Phrase
AP : Adjective Phrase

Most Common Creative Coding Platforms

Processing

Openframeworks

Cinder

p5.js

Touchdesigner

MaxMSP/Jitter

Vvvv

Watch: Creative Coding Toolkits Panel - LISA2012 (refer to SUCourse Resources)

Most Common Creative Coding Platforms

Processing

Openframeworks

Cinder

p5.js

Touchdesigner

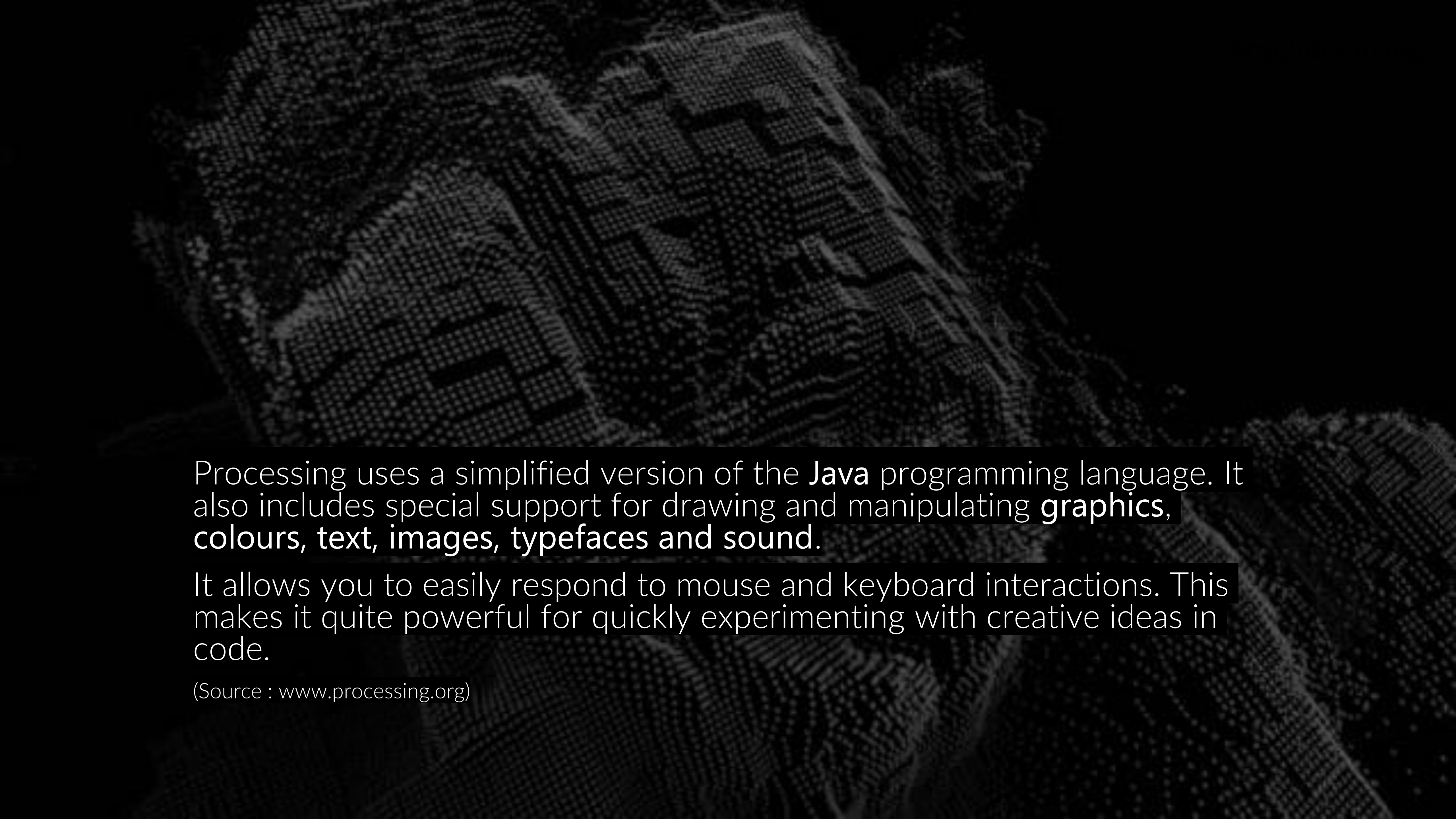
MaxMSP/Jitter

Vvvv

Watch: Creative Coding Toolkits Panel - LISA2012 (refer to SUCourse Resources)

Processing is a flexible software sketchbook and a language for learning how to code within the context of the visual arts. Since 2001, Processing has promoted software literacy within the visual arts and visual literacy within technology. There are tens of thousands of students, artists, designers, researchers, and hobbyists who use Processing for learning and prototyping.

www.processing.org



Processing uses a simplified version of the **Java** programming language. It also includes special support for drawing and manipulating **graphics, colours, text, images, typefaces and sound**.

It allows you to easily respond to mouse and keyboard interactions. This makes it quite powerful for quickly experimenting with creative ideas in code.

(Source : www.processing.org)

p5*js

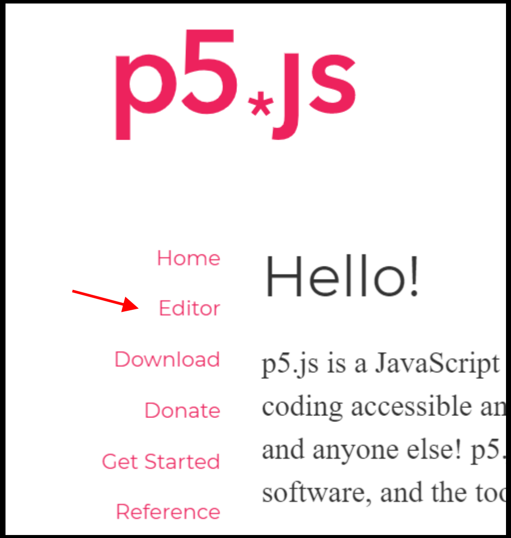
p5.js is a JavaScript library for creative coding, with a focus on making coding accessible and inclusive for artists, designers, educators, beginners, and anyone else! p5.js is free and open-source because we believe software, and the tools to learn it, should be accessible to everyone.www.processing.org

Intro p5.js by Daniel Schiffman ref:youtube

Using p5.js

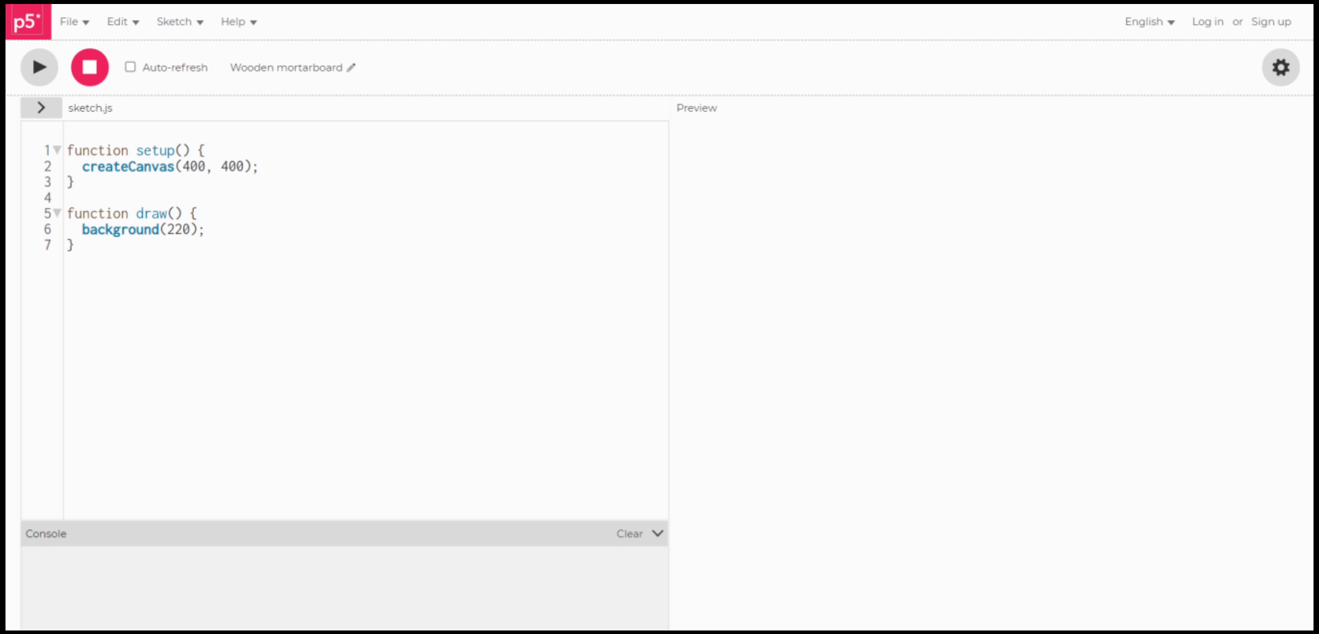
Method 1: Use online editor

Follow the link https://p5js.org/



Hello!

p5.js is a JavaScript
coding accessible an
and anyone else! p5.
software, and the too



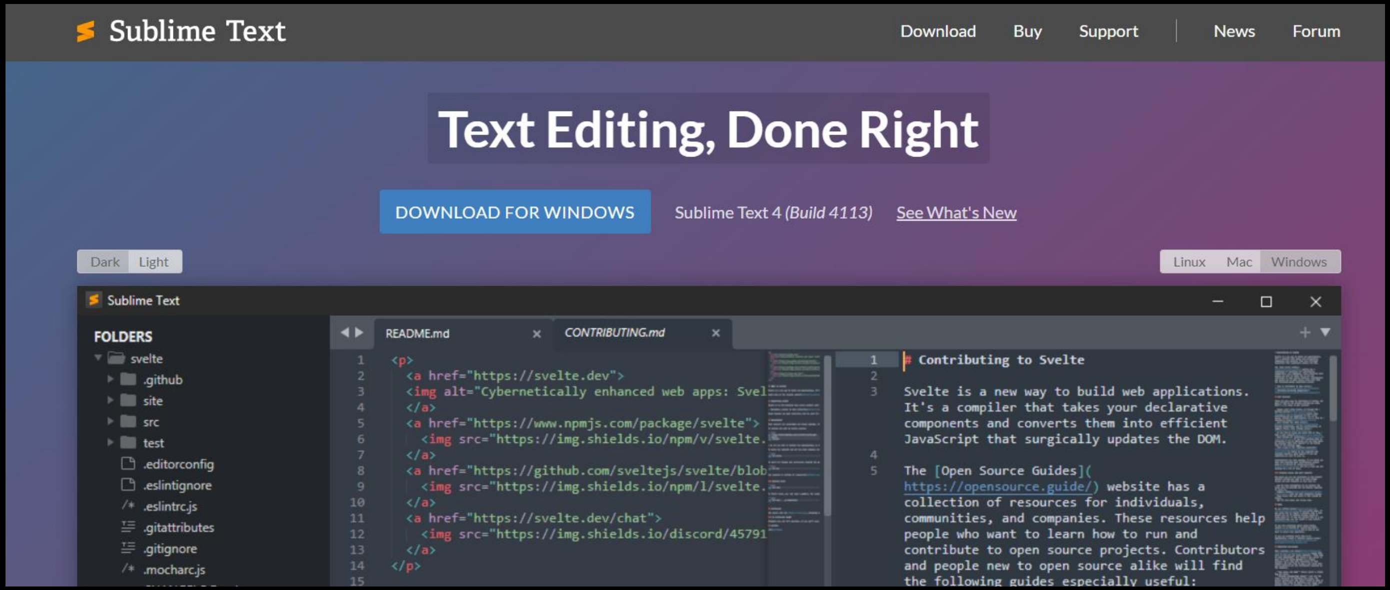
```
function setup() {
  createCanvas(400, 400);
}

function draw() {
  background(220);
}
```

Using p5.js

Method 2: Use offline editor

Follow the link https://www.sublimetext.com/

Download and Install Sublime Text Editor



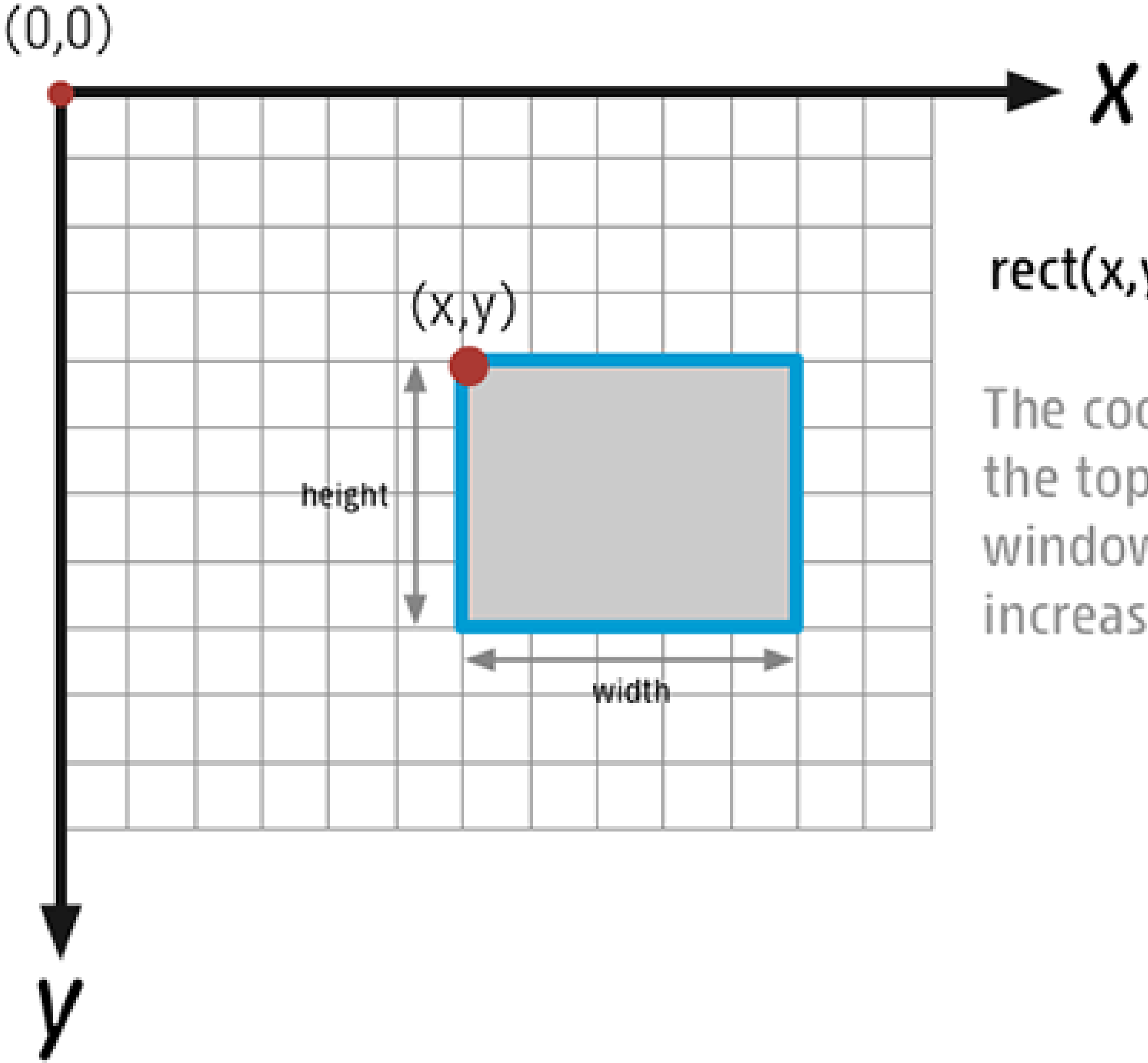
Text Editing, Done Right

Sublime Text 4 *(Build 4113)*

```
<p>
  <a href="https://svelte.dev">
  <img alt="Cybernetically enhanced web apps: Svel
  </a>
  <a href="https://www.npmjs.com/package/svelte">
    <img src="https://img.shields.io/npm/v/svelte.
  </a>
  <a href="https://github.com/sveltejs/svelte/blob
    <img src="https://img.shields.io/npm/l/svelte.
  </a>
  <a href="https://svelte.dev/chat">
    <img src="https://img.shields.io/discord/45791
  </a>
</p>
```

```
# Contributing to Svelte

Svelte is a new way to build web applications.
It's a compiler that takes your declarative
components and converts them into efficient
JavaScript that surgically updates the DOM.

The [Open Source Guides](
https://opensource.guide/) website has a
collection of resources for individuals,
communities, and companies. These resources help
people who want to learn how to run and
contribute to open source projects. Contributors
and people new to open source alike will find
the following guides especially useful:
```

SCREEN COORDINATES IN PROCESSING



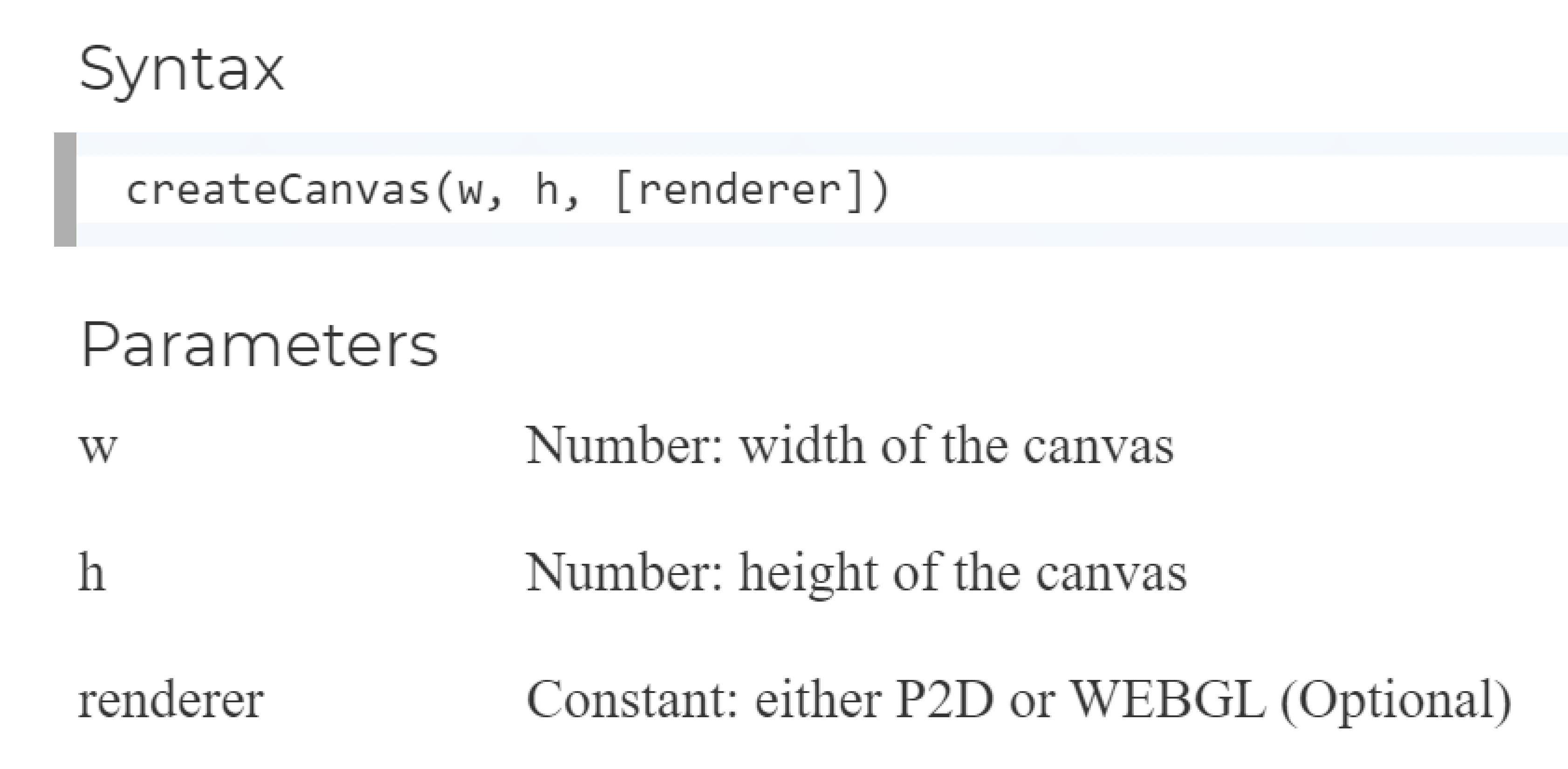
```
rect(x,y,width,height)
```

The coordinate origin (0,0) is at the top left of the display window. Increasing *x* moves right, increasing *y* moves down.

Analyzing p5.js Coding Environment

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

function draw() {
        background(220);
}
```

All Commands are Case Sensitive!!



Syntax

```
createCanvas(w, h, [renderer])
```

Parameters

w	Number: width of the canvas
h	Number: height of the canvas
renderer	Constant: either P2D or WEBGL (Optional)

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

Using Comments

Comments are to be used by humans only. So computer will ignore those lines, and compile the uncommented lines.

Comment Types

Block Comment : /* …………… */ (Using forward slash asterix …. asterix forward slash)

Line Comment : // (Using two forward slash)

Alternative usage : You may want to ignore a block or a line of code to check for debugging.

Drawing Basic Shapes

Some simple shapes;

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

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

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

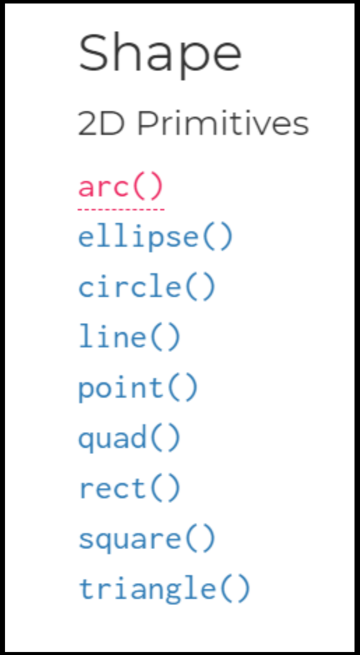
rect : https://p5js.org/reference/#/p5/rect

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

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

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

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



Shape

2D Primitives

arc()

ellipse()

circle()

line()

point()

quad()

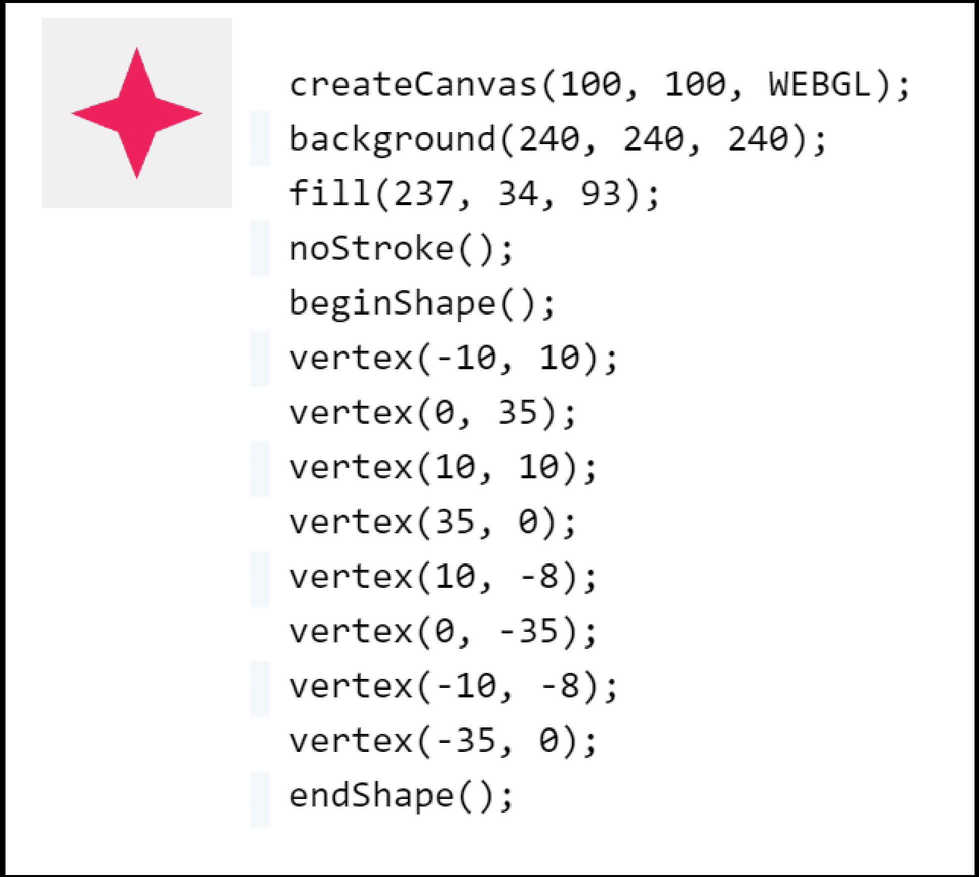
rect()

square()

triangle()

Important note: in 2D mode (i.e. when p5.Renderer is not set) the origin (0,0) is positioned at the top left of the screen. In 3D mode (i.e. when p5.Renderer is set to WEBGL), the origin is positioned at the center of the canvas.

Drawing Basic Shapes



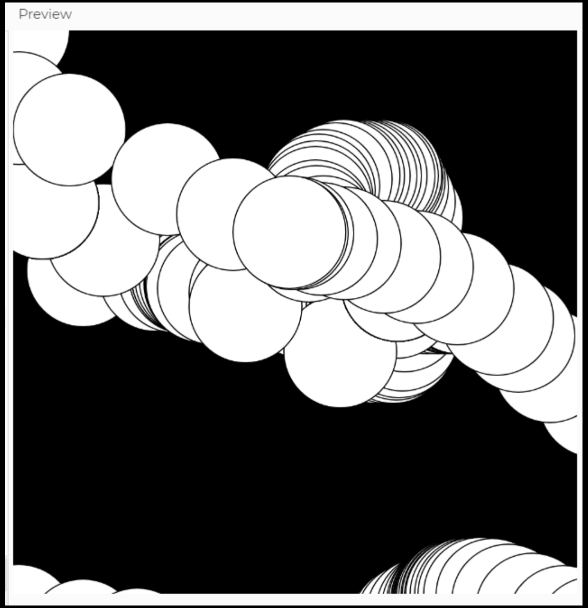
```
createCanvas(100, 100, WEBGL);
background(240, 240, 240);
fill(237, 34, 93);
noStroke();
beginShape();
vertex(-10, 10);
vertex(0, 35);
vertex(10, 10);
vertex(35, 0);
vertex(10, -8);
vertex(0, -35);
vertex(-10, -8);
vertex(-35, 0);
endShape();
```

Free Coding Time (40 minutes)

Play with the code!! Feel free to experiment.

Adding a bit of an Interaction

```
function setup() {
        createCanvas(500,500);
        background(0);
}
function draw() {
        //we do not call the background
        // function so that it adds each time
        // draw is called
        circle(mouseX,mouseY,100);
}
```



Colors

color() : Creates colors for storing in variables of the color datatype. The parameters are interpreted as RGB or HSB values depending on the current colorMode(). The default mode is RGB values from 0 to 255 and, therefore, color(255, 204, 0) will return a bright yellow color.

Syntax

color(gray)

color(gray, alpha)

color(v1, v2, v3)

color(v1, v2, v3, alpha)

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

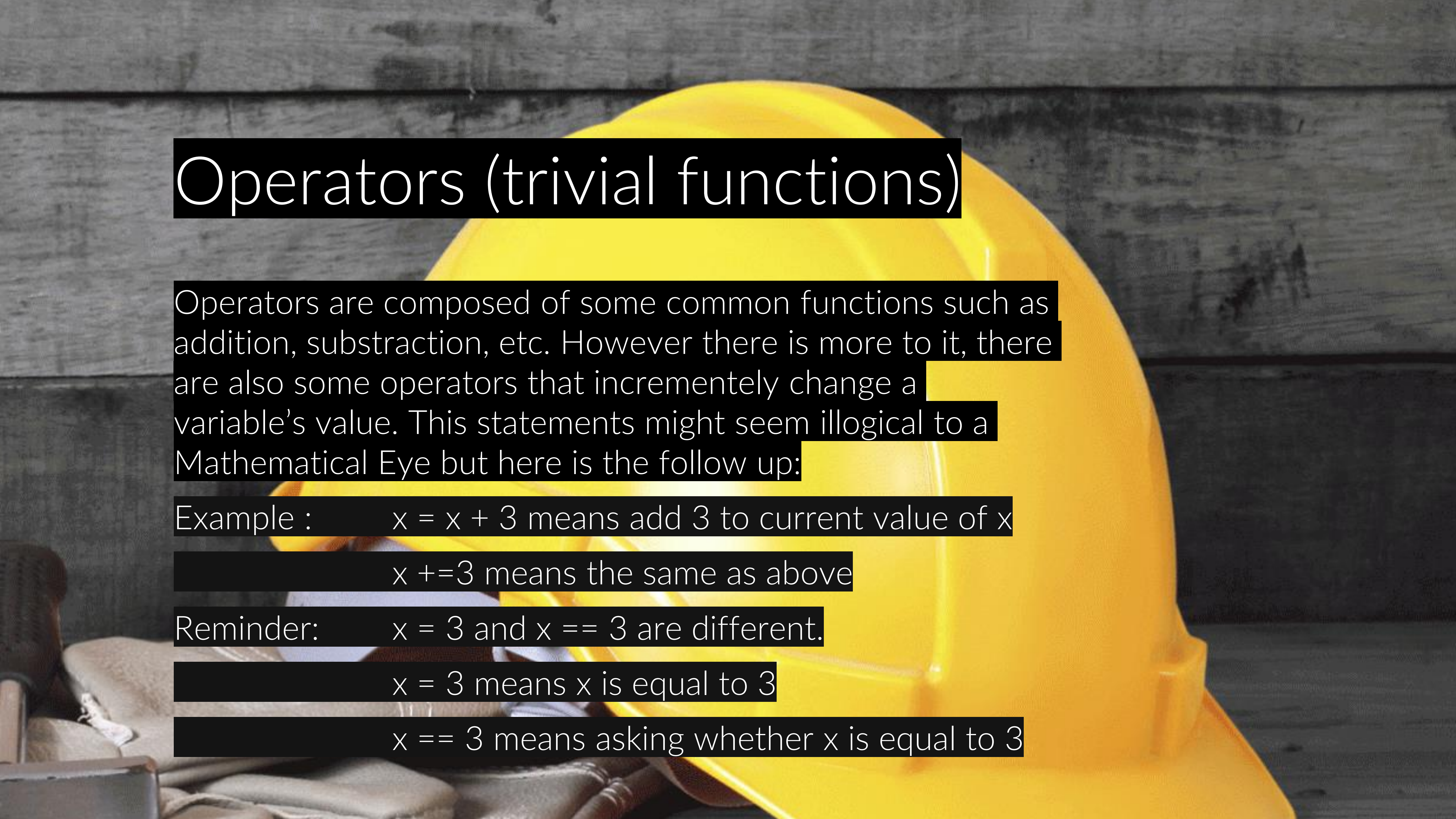
```
fill(255, 204, 0);
noStroke();
rect(30, 20, 55, 55);
```

Variables

In order to create dynamic structures, we are required to have data carriers that can change its value at a particular moment. So variables are **great** tools for **Creative Coding.**

Variable Examples

```
let a = 50;

let t = "hello";

let boo = true;

let col = color(255, 204, 0);
```



Operators (trivial functions)

Operators are composed of some common functions such as addition, substraction, etc. However there is more to it, there are also some operators that incremently change a variable's value. This statements might seem illogical to a Mathematical Eye but here is the follow up:

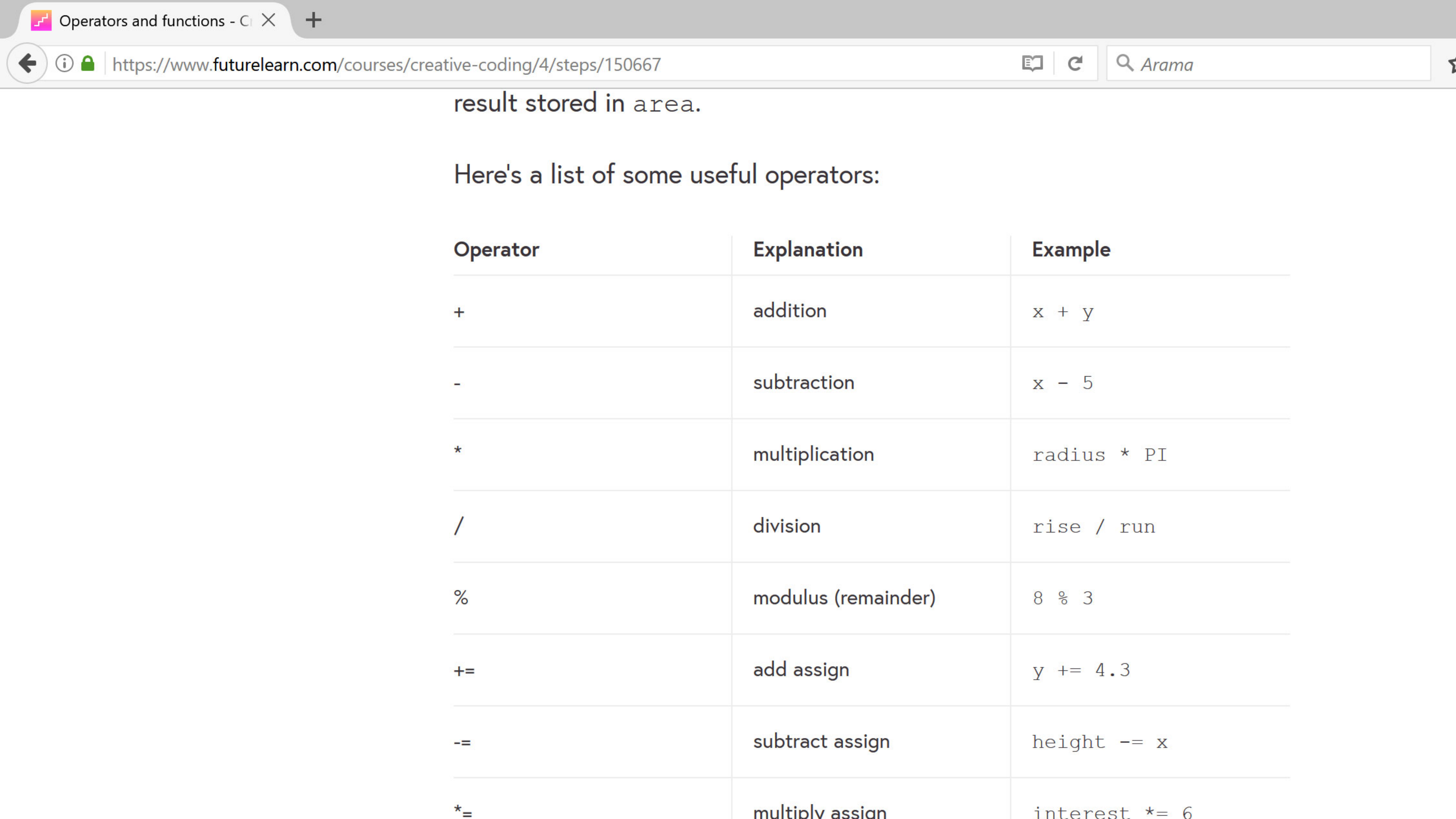
Example : x = x + 3 means add 3 to current value of x

x +=3 means the same as above

Reminder: x = 3 and x == 3 are different.

x = 3 means x is equal to 3

x == 3 means asking whether x is equal to 3


Operators and functions - C
https://www.futurelearn.com/courses/creative-coding/4/steps/150667
Arama

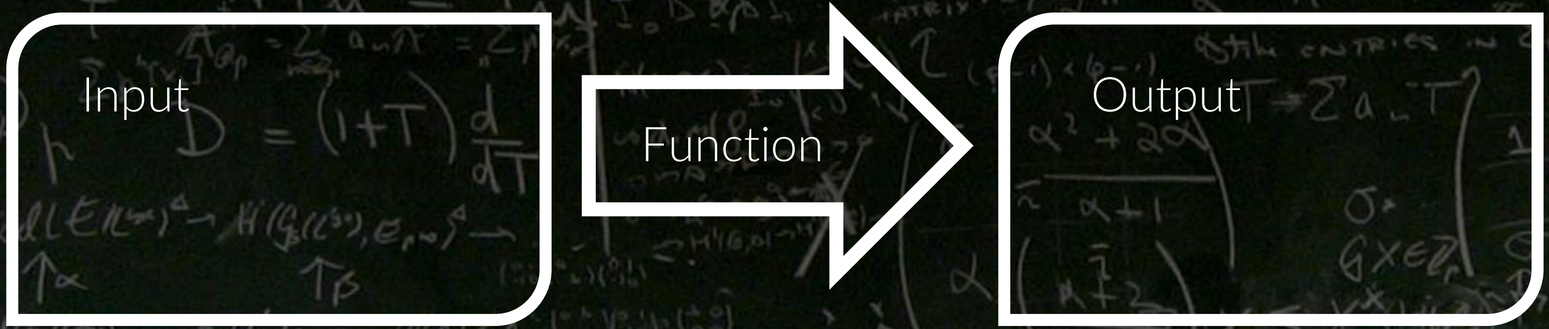
result stored in `area`.

Here's a list of some useful operators:

Operator	Explanation	Example
+	addition	`x + y`
-	subtraction	`x - 5`
*	multiplication	`radius * PI`
/	division	`rise / run`
%	modulus (remainder)	`8 % 3`
+=	add assign	`y += 4.3`
-=	subtract assign	`height -= x`
*=	multiply assign	`interest *= 6`

Functions

Functions are building blocks for achieving a specific purpose. They have inputs and outputs.



Input

Function

Output

Functions Example:

```
function setup() {
            createCanvas(400, 400);
            background(220);}


function draw() {
            drawStar(100,100,100,30,5);
}
function drawStar(x, y, radius1, radius2, npoints) {
      let angle = TWO_PI / npoints;
      let halfAngle = angle / 2.0;
      beginShape();
      for (let a = 0; a < TWO_PI; a += angle) {
      let sx = x + cos(a) * radius2;
      let sy = y + sin(a) * radius2;
      vertex(sx, sy);
      sx = x + cos(a + halfAngle) * radius1;
      sy = y + sin(a + halfAngle) * radius1;
      vertex(sx, sy);
      }
      endShape(CLOSE);
}
```

Here a function was defined that contains the drawing commands. This function can be called from anywhere in the program in order to execute the code it contains.

Functions Example:

```
function setup() {
        createCanvas(400, 400);
        background(220);}


function draw() {
        drawStar(100,100,100,30,5);
        drawStar(100,300,10,3,5);
}
function drawStar(x, y, radius1, radius2, npoints) {
    let angle = TWO_PI / npoints;
    let halfAngle = angle / 2.0;
    beginShape();
    for (let a = 0; a < TWO_PI; a += angle) {
    let sx = x + cos(a) * radius2;
    let sy = y + sin(a) * radius2;
    vertex(sx, sy);
    sx = x + cos(a + halfAngle) * radius1;
    sy = y + sin(a + halfAngle) * radius1;
    vertex(sx, sy);
    }
    endShape(CLOSE);
}
```

Loops Example:

```
for (let i = 0; i < 9; i++) {
  console.log(i);
}
```

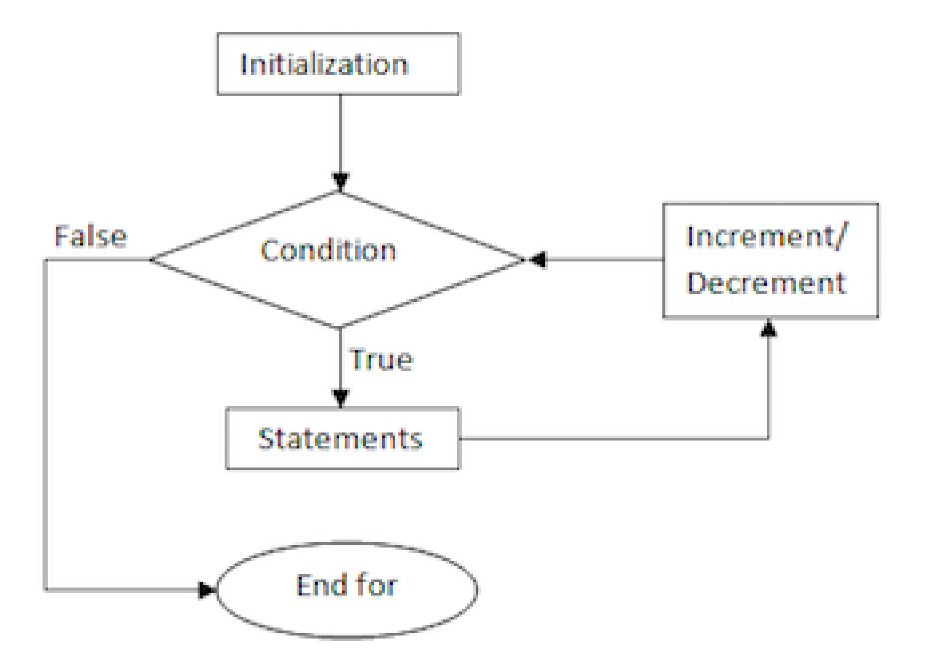
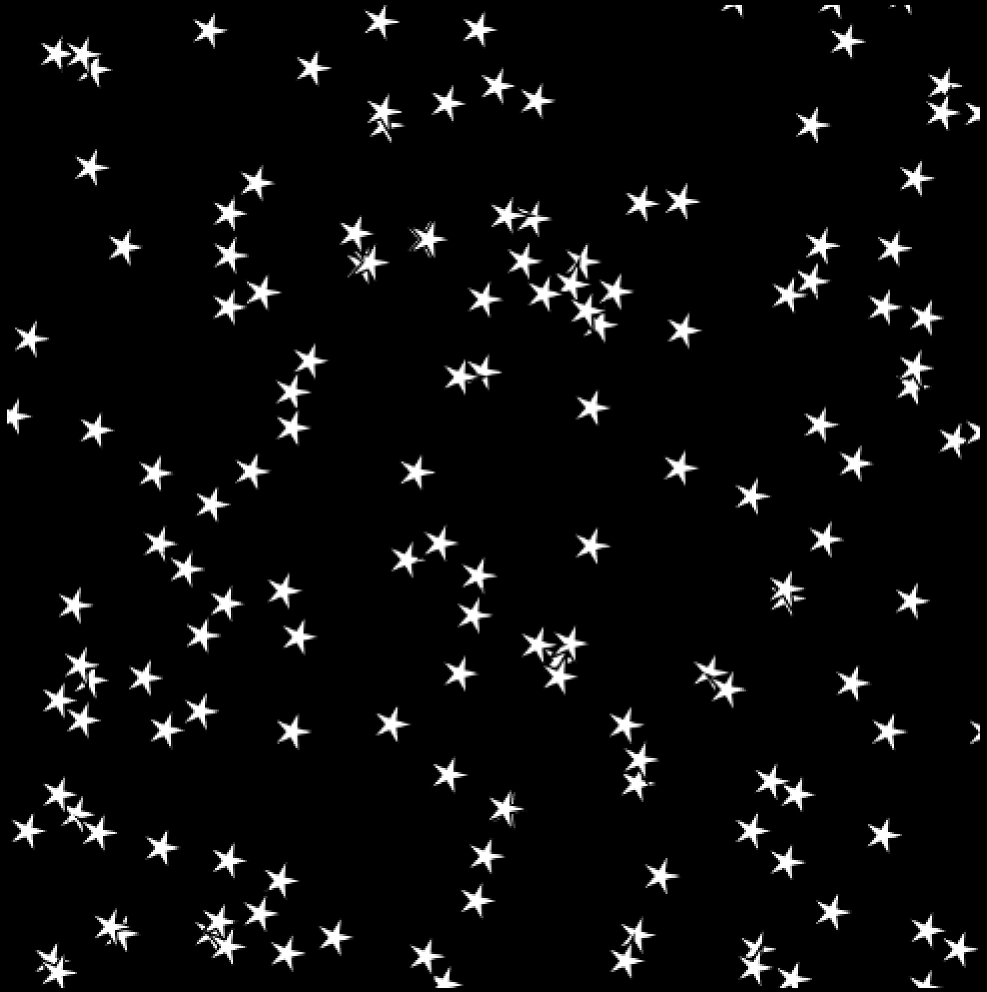
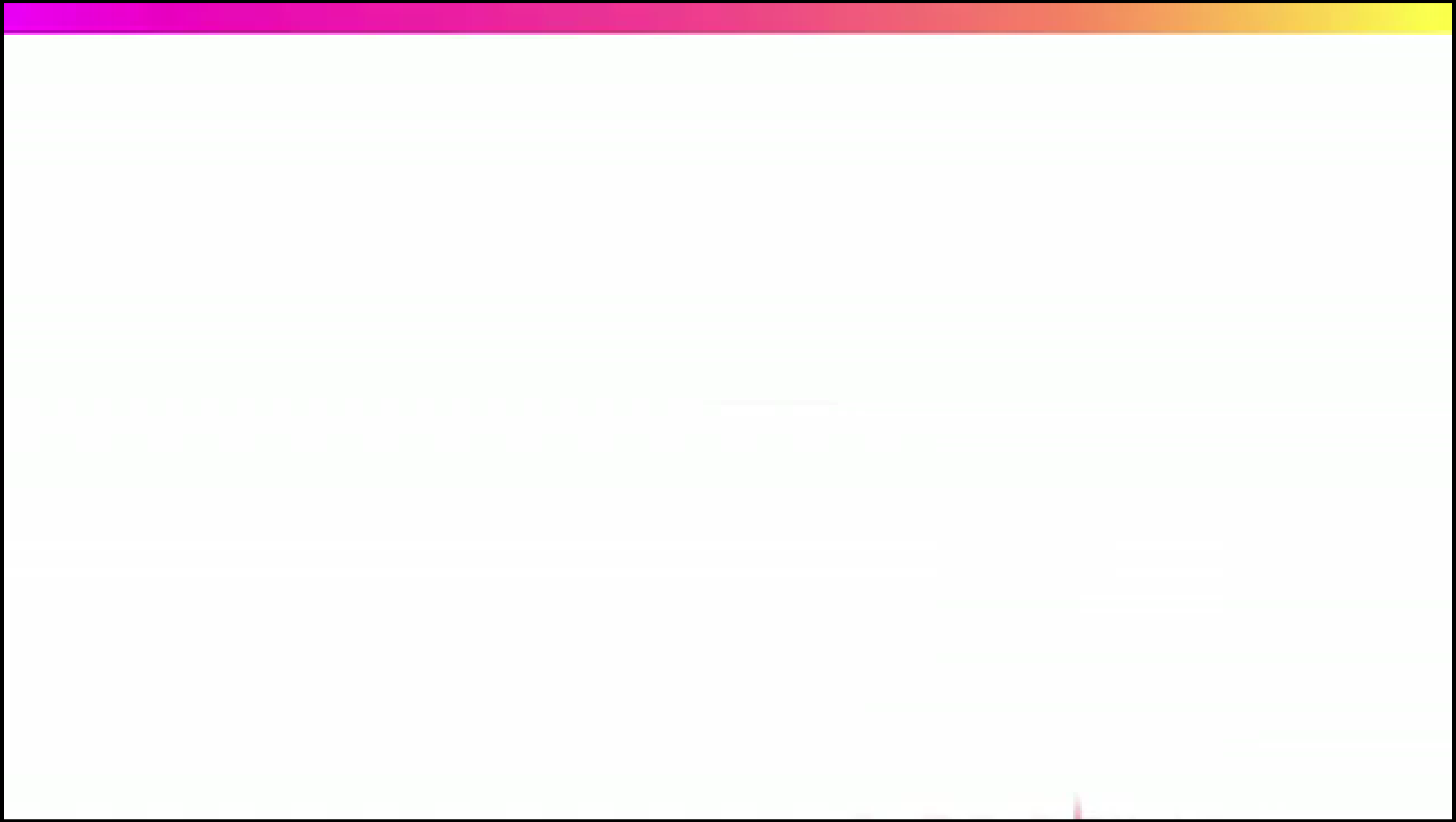


fig: Flowchart for for loop

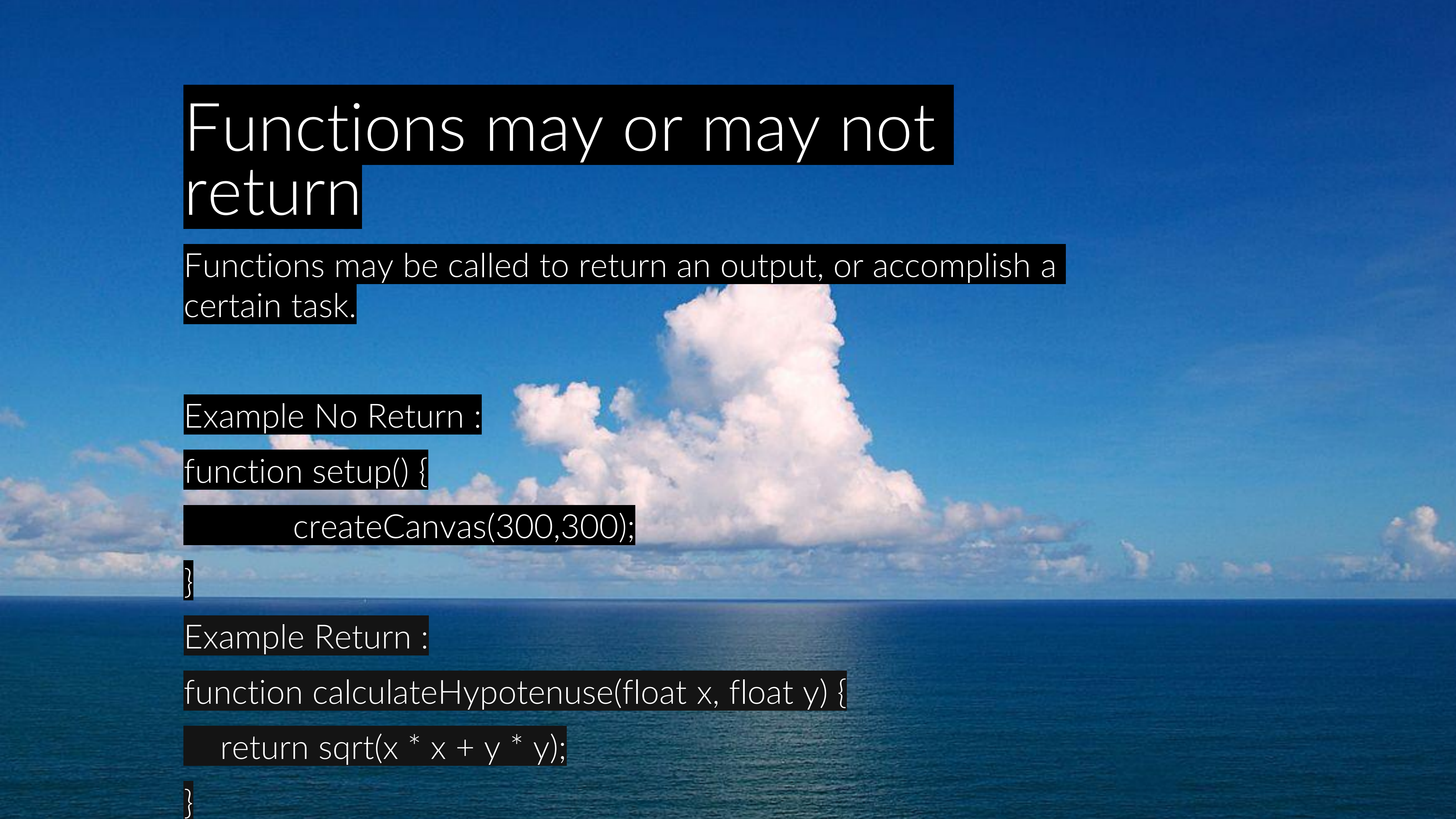
Functions Example:

```
function setup() {
        createCanvas(400, 400);
        background(0);
        for (let i = 0; i < 150; i++) {
            let x1 = random(width);
            let x2 = random(height);
            drawStar(x1, x2, 3, 10, 5);
        }
}


function draw() {
}
function drawStar(x, y, radius1, radius2, npoints) {
    let angle = TWO_PI / npoints;
    let halfAngle = angle / 2.0;
    beginShape();
    for (let a = 0; a < TWO_PI; a += angle) {
    let sx = x + cos(a) * radius2;
    let sy = y + sin(a) * radius2;
    vertex(sx, sy);
    sx = x + cos(a + halfAngle) * radius1;
    sy = y + sin(a + halfAngle) * radius1;
    vertex(sx, sy);
    }
    endShape(CLOSE);
}
```





https://www.futurelearn.com/courses/creative-coding



Functions may or may not return

Functions may be called to return an output, or accomplish a certain task.

Example No Return :

```
function setup() {
        createCanvas(300,300);
}
```

Example Return :

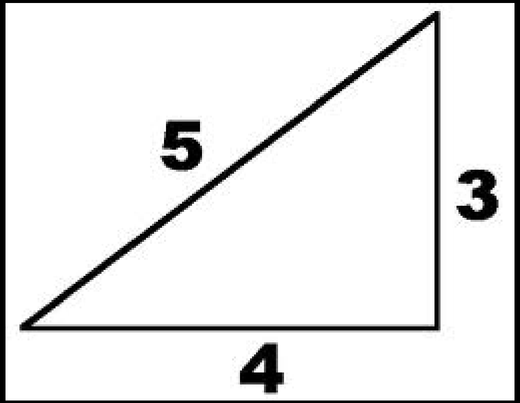
```
function calculateHypotenuse(float x, float y) {
    return sqrt(x * x + y * y);
}
```

Functions

Example:

```
function setup(){
	createCanvas(300,300);
	console.log(findHypothenuse(3,4));
}

function findHypothenuse(x,y){
	let hyp = sqrt(sq(x)+sq(y));
	return hyp;
}
```

Values can be passed in functions, which can also return a value as a result.



Chance Operations &Randomness

Artwork Credit : Candaş Şişman
Pattern of Possibilities
2015
Installation
23.716 Piece Backgammon Dices,
Wood, Video and Sound
130x130 cm

Randomness is observed in a sequence when order and patterns seem to be lacking (Philip Pasquier)

Given a variable X defined over a discrete set of options, its domain D_x, a random draw is giving equal probability to all the option from D_x (equiprobability)

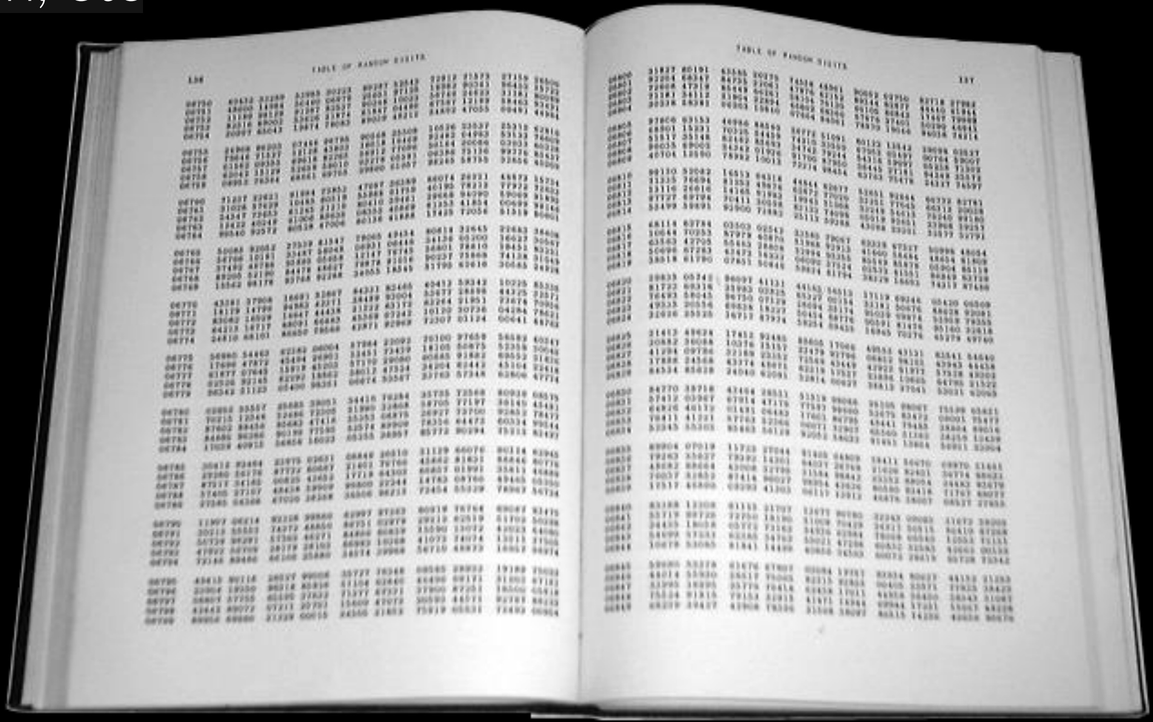
How to get random numbers

Randomness is not random at all.

One Million Random Book

Natural Sources of Entropy : Atmospheric Noise,

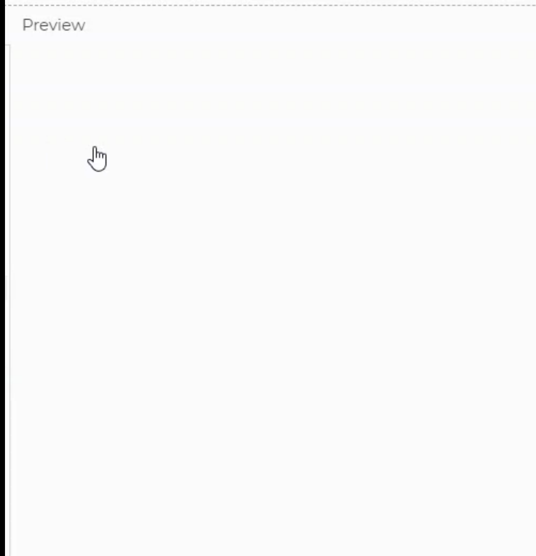
Thermal Noise, Cosmic Background Radiation, etc



Challenge (15 min)

Play with the same code, but draw circles with variable sizes.

- Hint: Try randomness https://p5js.org/reference/#/p5/random
- Ambitious Hint: Use mouseX, mouseY https://p5js.org/reference/#/p5/mouseX



Randomness in Computers

A pseudorandom process is a process that appears to be random but is not. Pseudorandom sequences typically exhibit statistical randomness while being generated by an entirely deterministic causal process. (wikipedia)

To generate truly random numbers would require precise, accurate, and repeatable system measurements of absolutely non-deterministic processes. Linux uses, for example, various system timings (like user keystrokes, I/O, or least-significant digit voltage measurements) to produce a pool of random numbers

Randomness in p5js

Let’s visit >
https://p5js.org/reference/#/p5/random

By default, random() produces different results each time the program is run. Set the seed parameter to a constant to return the same pseudo-random numbers each time the software is run.

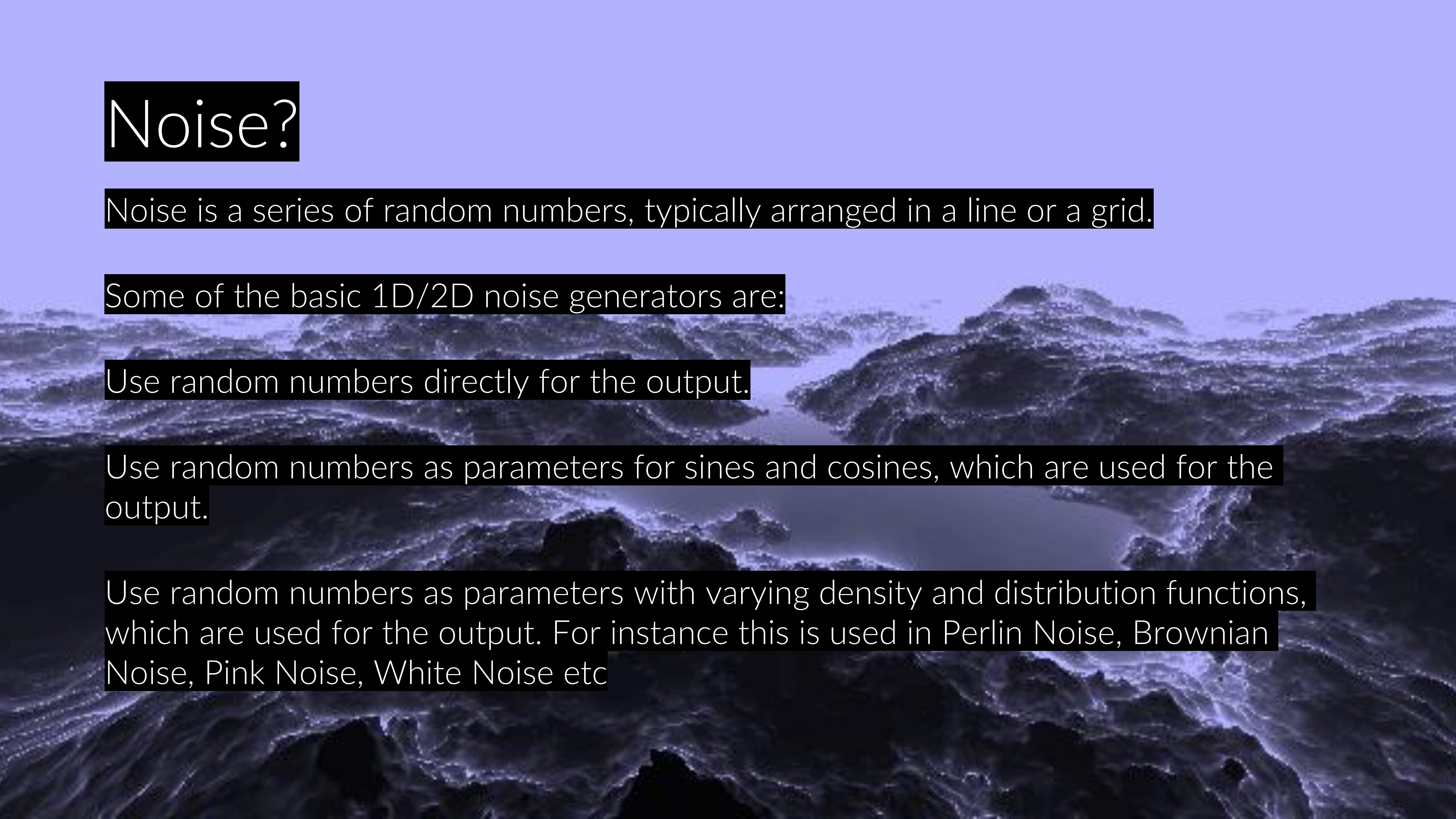
random([min], [max])

Random

```
function setup(){
        createCanvas(600,400);
}


function draw(){
    background(0);
    fill(255);
    let x = random(width);
    circle(x,height * 0.5, 40);
}
```

Pseudorandom number generators | Computer Science | Khan Academy

https://www.youtube.com/watch?v=GtOt7EBNEwQ



Noise?

Noise is a series of random numbers, typically arranged in a line or a grid.

Some of the basic 1D/2D noise generators are:

Use random numbers directly for the output.

Use random numbers as parameters for sines and cosines, which are used for the output.

Use random numbers as parameters with varying density and distribution functions, which are used for the output. For instance this is used in Perlin Noise, Brownian Noise, Pink Noise, White Noise etc

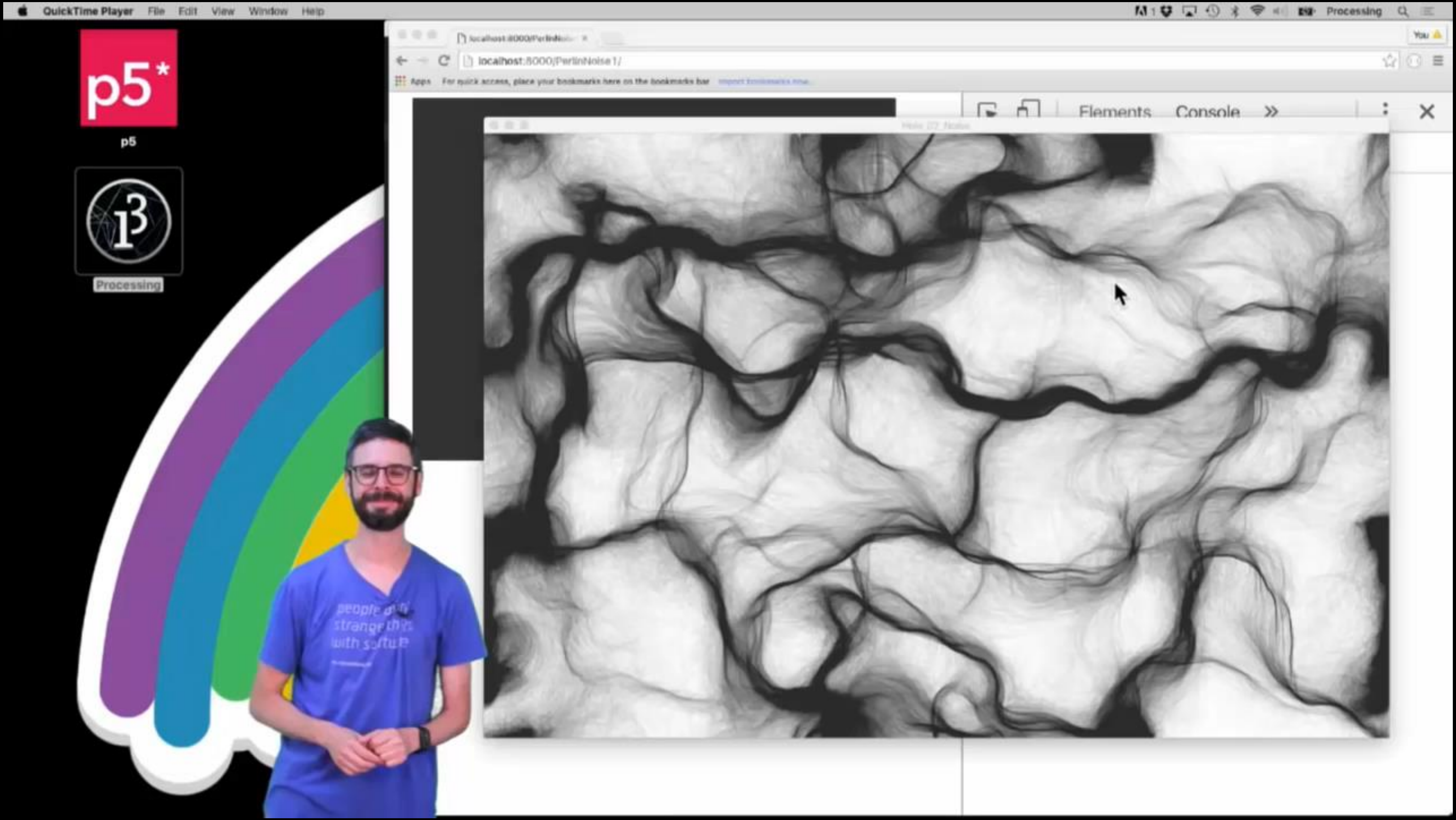
Perlin Noise?

An algorithm known as “Perlin noise,” named for its inventor Ken Perlin, takes this concept into account. Perlin developed the noise function while working on the original Tron movie in the early 1980s; it was designed to create procedural textures for computer-generated effects. In 1997 Perlin won an Academy Award in technical achievement for this work. Perlin noise can be used to generate various effects with natural qualities, such as clouds, landscapes, and patterned textures like marble.

Perlin noise has a more organic appearance because it produces a naturally ordered (“smooth”) sequence of pseudo-random numbers.

Random vs Noise

Recommended: https://www.youtube.com/watch?v=YcdldZ1E9gU


QuickTime Player
File
Edit
View
Window
Help
Processing
localhost:8000/PerlinNoise1/
Apps
For quick access, place your bookmarks here on the bookmarks bar
p5*
p5
Processing
Elements
Console
people ... strange thi... with softw...

Recommended: https://www.youtube.com/watch?v=YcdldZ1E9gU

Using Perlin Noise in p5.js

noise()

Returns the Perlin noise value at specified coordinates. Perlin noise is a random sequence generator producing a more natural, harmonic succession of numbers than that of the standard random() function.

https://processing.org/reference/noise_.html

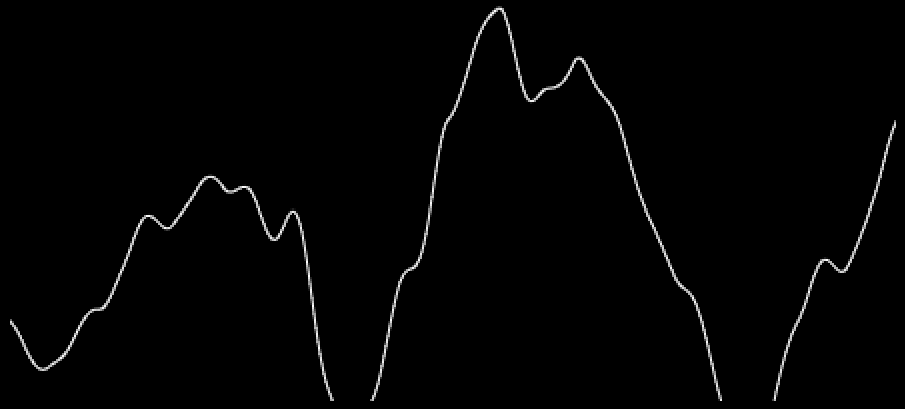
```
let xoff = 0.0;

function setup(){
        createCanvas(600,400);
}

    function draw(){
    background(0);
    fill(255);
    xoff = xoff + .01;
    let x = noise(xoff) * width;
    ellipse(x,height/2, 40,40);
}
```

Using Perlin Noise in p5.js 2D

```
let xoff = 0.0;
let yoff = 10000;
function setup(){
        createCanvas(600,400); }

function draw(){ background(0);
fill(255);
xoff = xoff + .01;
yoff = yoff + .01;
let x = noise(xoff) * width;
let y = map(noise(yoff), 0 , 1 , 0, height);
ellipse(x,y, 40,40);
}
```

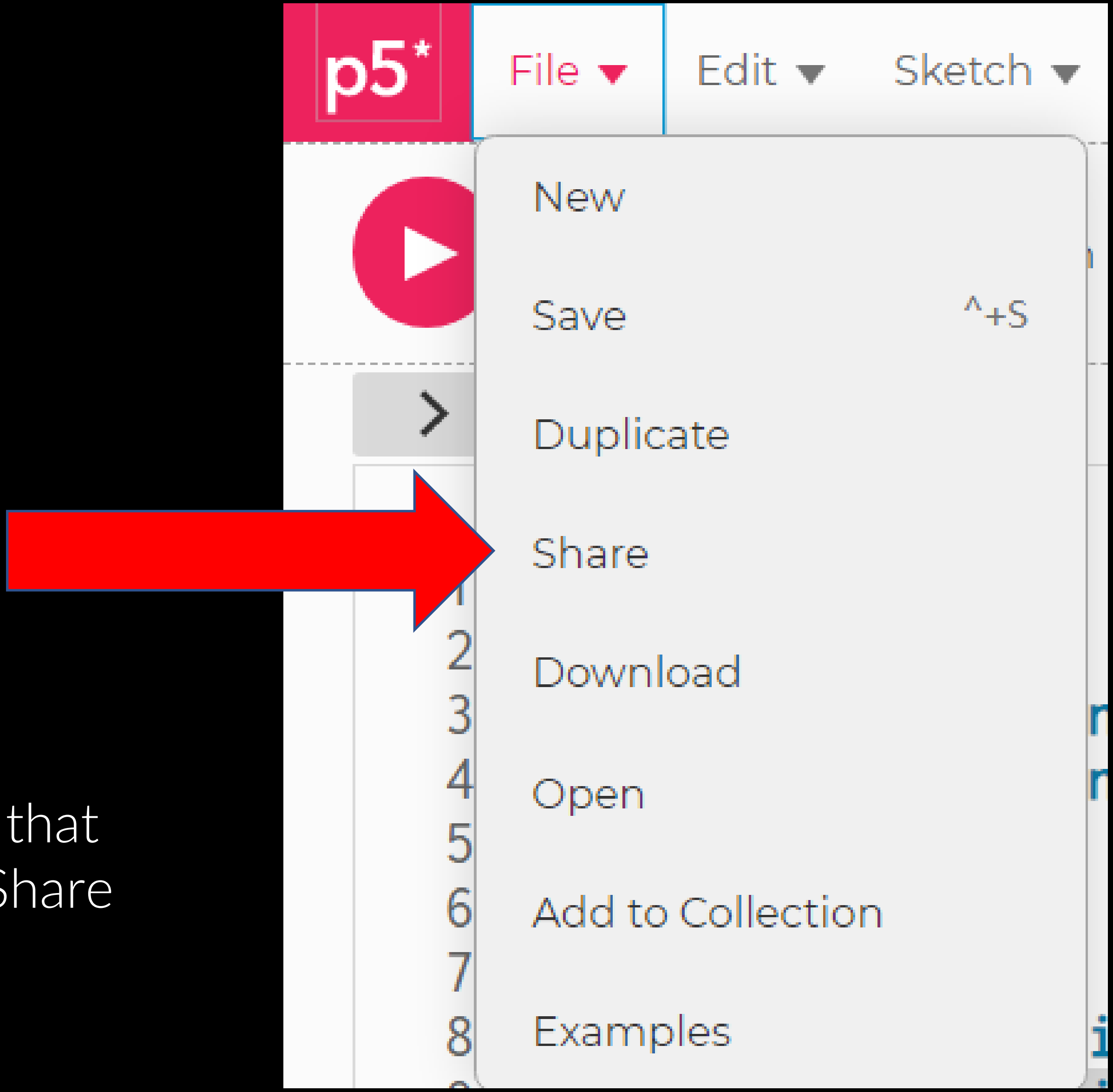
Graphing Perlin Noise in p5.js

```
let incr = 0.01;
let start = 0;

function setup() {
        createCanvas(600, 400);
}

function draw() {
        background(0);
        noFill();
        let xoff = start;
        beginShape();
        for(let x = 0; x < width ; x++){
                stroke(255);
                let y = noise(xoff) * width;
                vertex(x,y);
                xoff = xoff + incr;
        }
        endShape();
        start += incr;
}
```



Assignment 002

Use Perlin Noise to create a p5.js Sketch that moves basic shapes smoothly overtime. Share me a link (Present).


p5*
File
Edit
Sketch
New
Save ^+S
Duplicate
Share
Download
Open
Add to Collection
Examples