Creative Coding

Intelligence Through Variables & Conditionals & Loops

COD 207 - Week 03 Class →

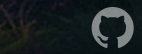
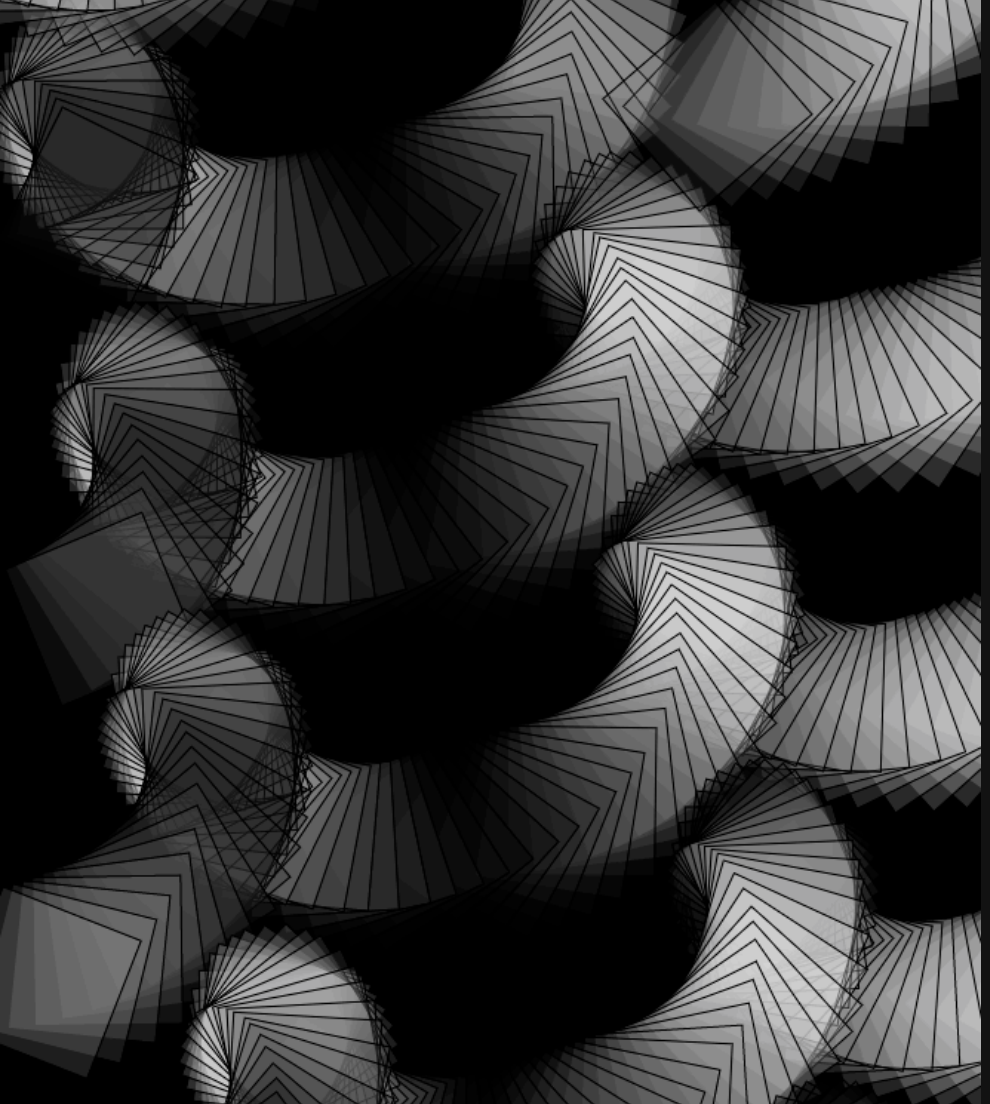


Table of Contents



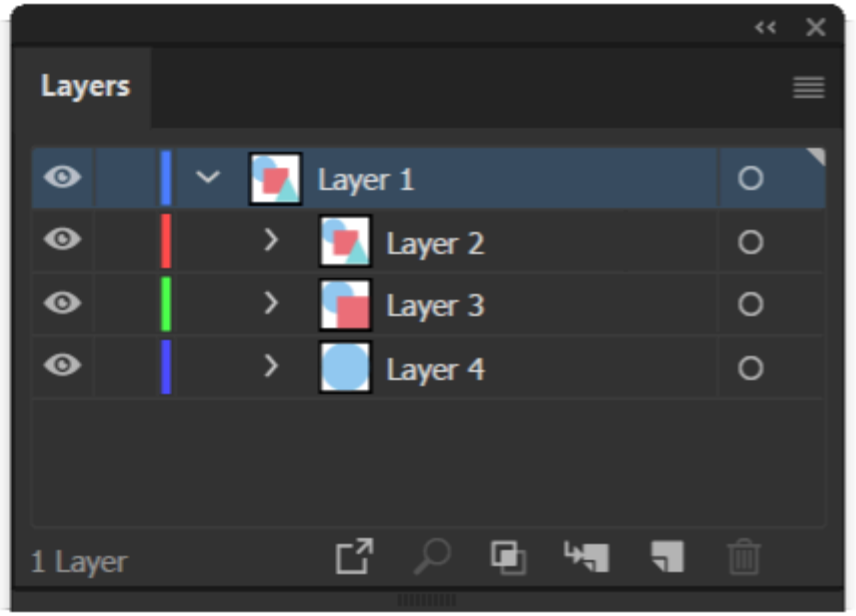
Wrap-up (Summary)

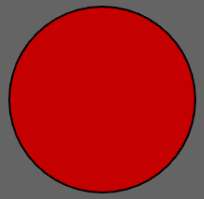
Things we learn about P5JS programming language.

- Cartesian Coordinate System (How canvas positioning works)
- Drawing simple shapes (`rect, ellipse, circle, triangle, arc...`)
- Styling shapes (`fill, noFill, stroke, noStroke...`)
- Drawing Order

Drawing Order

Drawing Order: Set the layer order of the shapes.





Structure

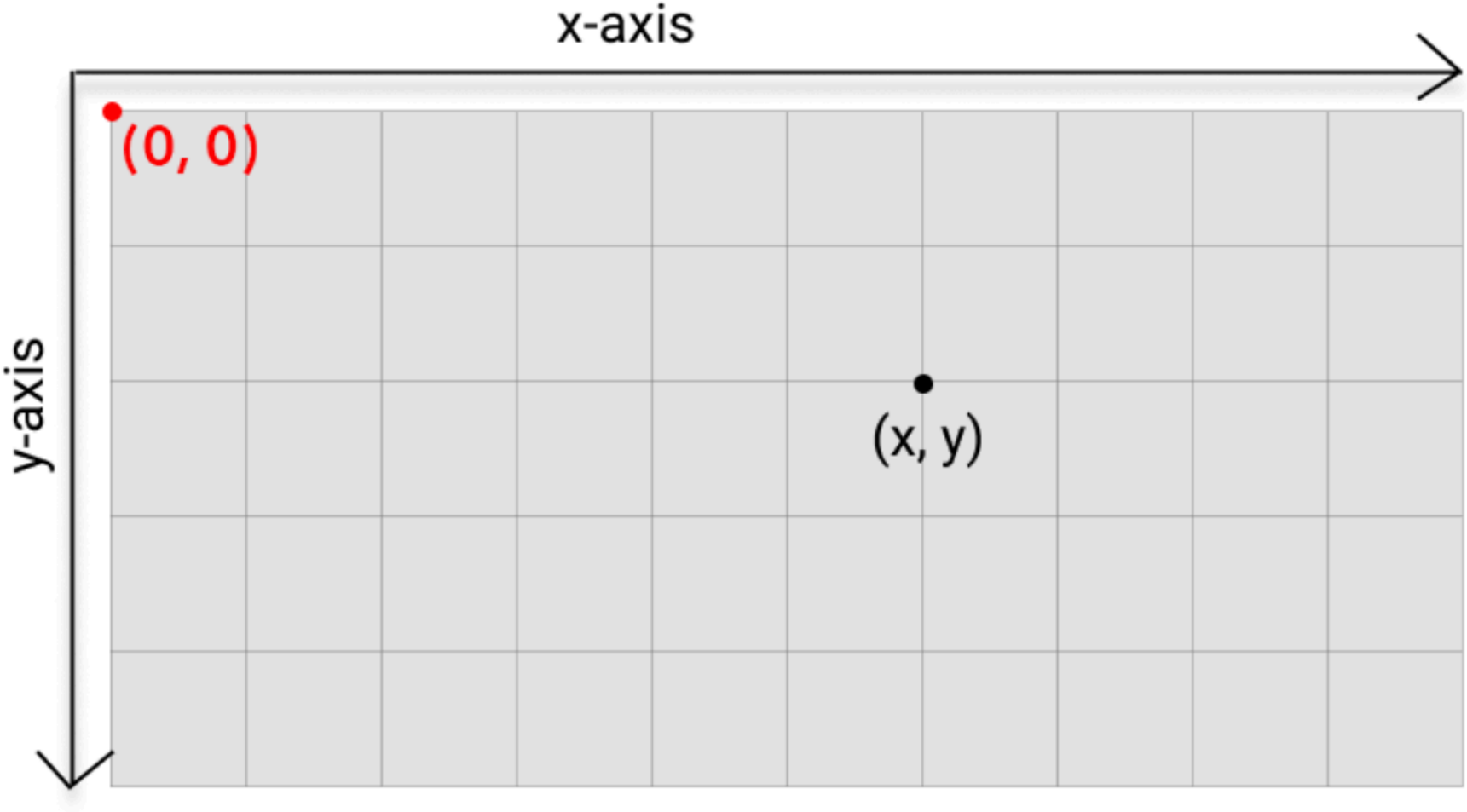
Inline functions

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

function draw() {
    background(100, 100, 100);

    fill(200, 0, 0);
    circle(300, 300, 100);
}
```

Coordinate System

Top left corner of the the canvas is the starting point of x and y coordinates.

The p5.js canvas uses a coordinate system to describe space. The origin, (0, 0), of the canvas is the top-left corner of the canvas.


x-axis
y-axis
(0, 0)
(x, y)

Color (fill function)

A color value can be represented in various ways with p5.js. It can be given as:

- Gray value as one numeric value between 0 and 255.
- RGB (Red, Green, Blue) walue as three numeric values between 0 and 255.

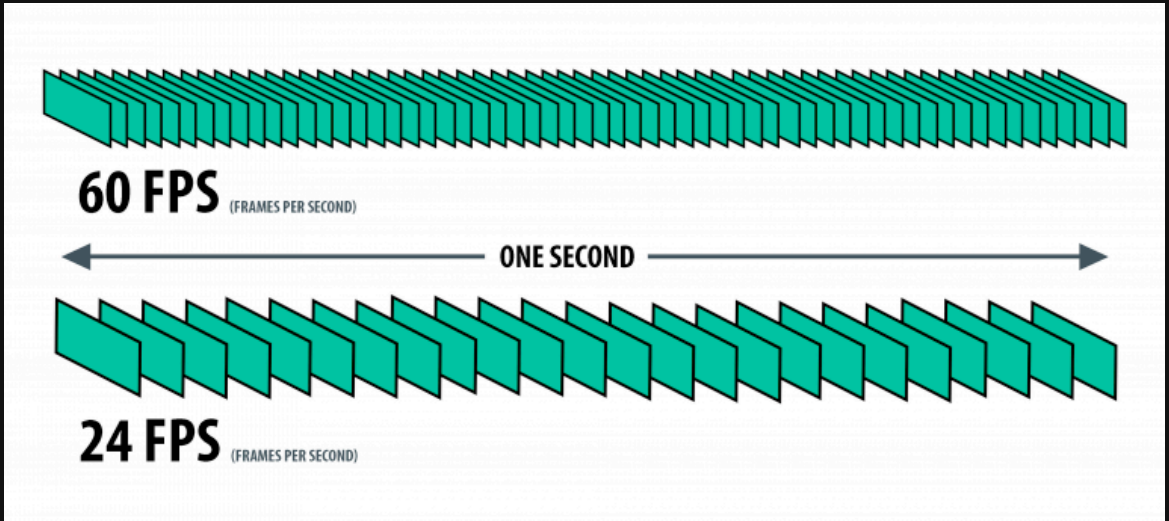
```
1   function setup() {
2       createCanvas(400,600);
3   }
4
5   function draw() {
6       //        (Red, Green, Blue)
7       background(71,   92,    122);
8
9       // Uncomment the following line to change
10      //fill(200, 10, 20);
11      circle(252,187,100);
12  }
```

Rendering

Rendering in P5JS involves generating images from input data such as vector graphics such as circle, rectangle, triangle, etc…

The resulting image, also called a render or frame, is displayed on the user's screen multiple times per second, typically ranging from 24 to 60 frames.

The graphics is rendered 60 times per second on your canvas by default in P5JS. Click on image to get more info.



Rendering, Built-in & Inline Functions

```
function setup() {
  // Creates the application window params: width=600, height=600
  createCanvas(600, 600);
}

function draw() {
  // Set the background color of the window params: Red: 100, Green: 20, Blue: 20
  // R,G,B values must be between 0 - 255
  background(100,20,20);
}
```

Variables

A variable stores a value in memory so that it can be used later in a program. A variable can be used many times within a single program, and the value is easily changed while the program is running.

> Declare a variable, if you type the same number more than twice.

💩 BAD

```
// position the elements in the middle
circle(200, 220, 300);
circle(200, 220, 250);
circle(200, 220, 200);
circle(200, 220, 150);
circle(200, 220, 100);
circle(200, 220, 50);
```

🥰 GOOD

```
// define position values for x and y axis
var xpos = 200;
var ypos = 220;

// position the elements in the middle
circle(xpos, ypos, 300);
circle(xpos, ypos, 250);
circle(xpos, ypos, 200);
circle(xpos, ypos, 150);
circle(xpos, ypos, 100);
circle(xpos, ypos, 50);
```

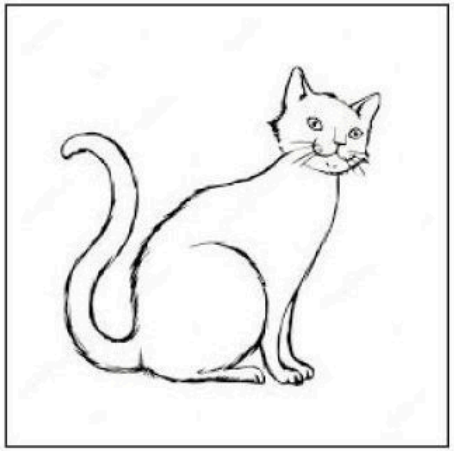
Reuse variables for relational purposes.

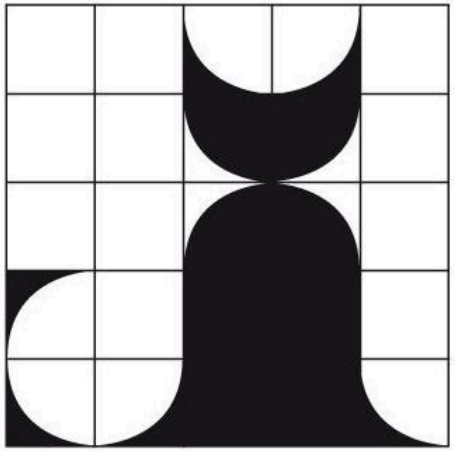
You can add values to the variables. E.g. Type the following into the `draw()` function.

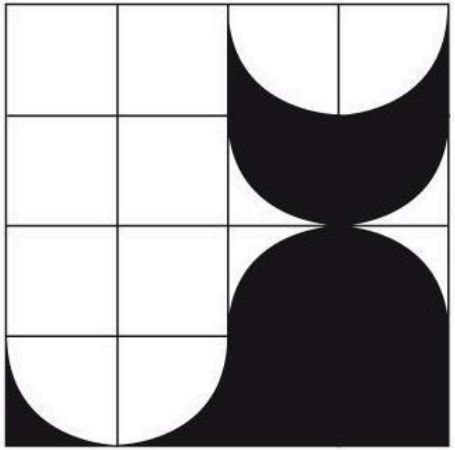
```
function draw() {
    var foo = 10;
    print(foo + 20);
    // output: 30
}
```

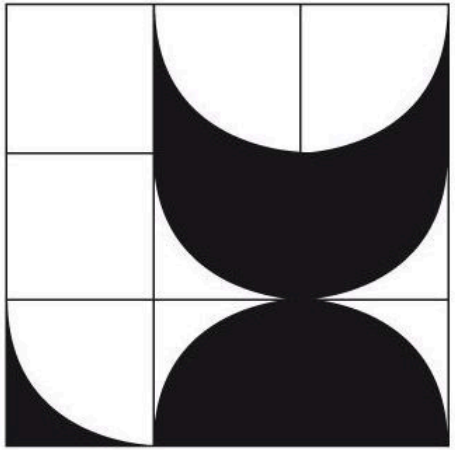
How to declare a variable for the 3rd parameter?

```
    // define position values for x and y axis
    var xpos = 200;
    var ypos = 220;

    // position the elements in the middle
    circle(xpos, ypos, 300);
    circle(xpos, ypos, 250);
    circle(xpos, ypos, 200);
    circle(xpos, ypos, 150);
    circle(xpos, ypos, 100);
    circle(xpos, ypos, 50);
```









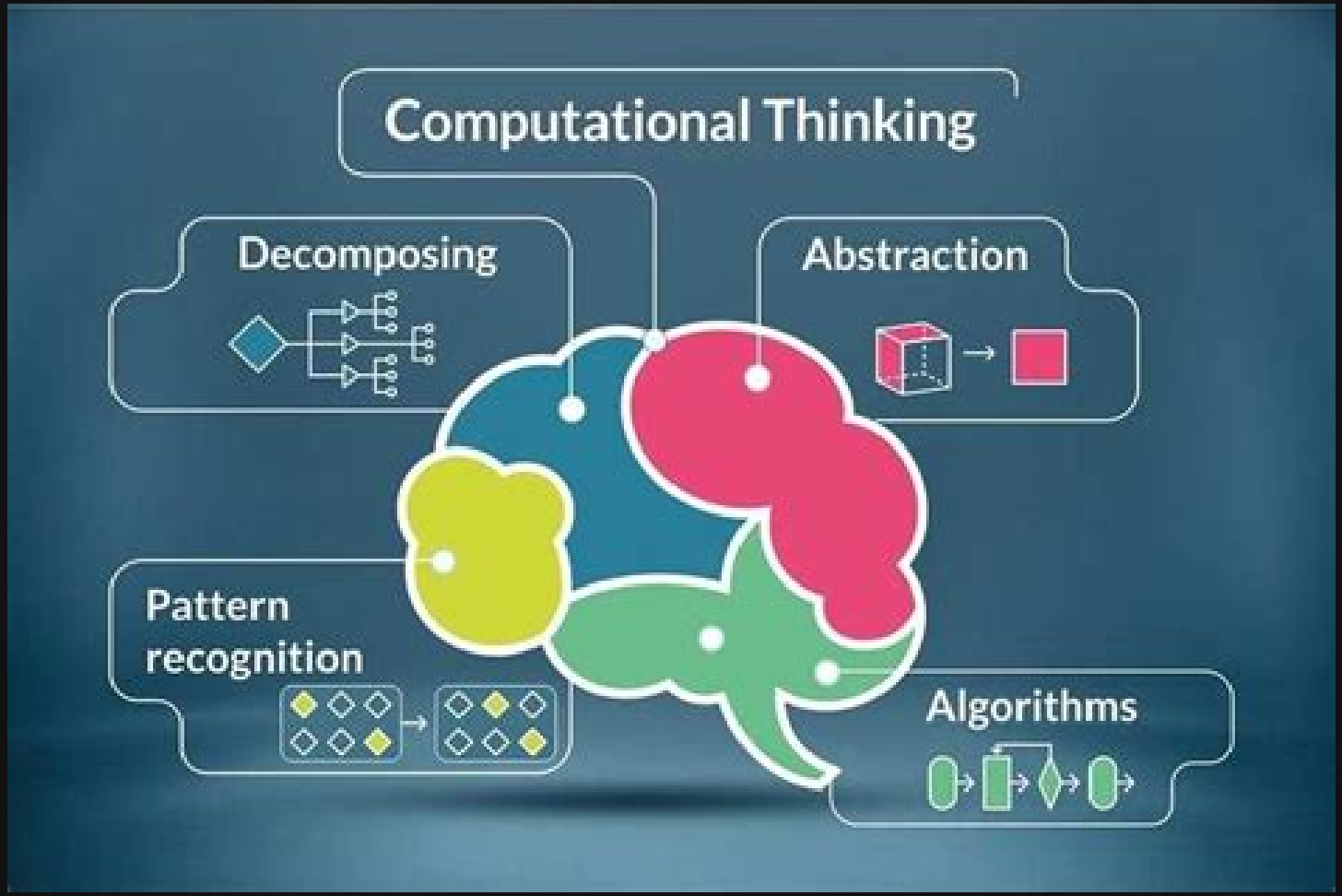
Abstraction

- Line Art
- Split into grids
- Analyze each grid
- Negative & Positive Space
- Shape Combinations (rect and arc)

Computational Thinking Framework

Computational thinking is a problem-solving process that includes; Decomposing, Abstraction, Pattern Recognition, Algorithms

- Formulating Problems & Analyze
- Represent data through Abstraction
- Algorithmic Thinking: Automate solutions
- Identifying, analyzing, and implementing possible solutions with the goal of achieving the most efficient and effective combination of steps and resources
- Generalizing and transferring this problem-solving process to a wide variety of problems


Computational Thinking
Decomposing
Abstraction
Pattern recognition
Algorithms

Conditionals

if/else statements

Introducing *if* statement

```
if ( condition )
{
    instruction 1
    instruction 2
    ...
}
```

```
if (a > 0)
{
    println("a is ", a);
    println("positive");
}
```

Don't type this yet. Just analyze the syntax.

`if` statement makes possible to execute a block of instructions (aka *code block*) only if a certain condition is valid.

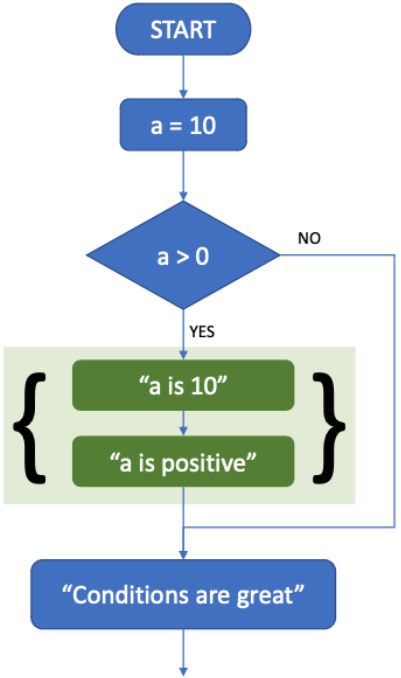
If the condition is not valid, the instructions between curly braces are not executed.

Set a Condition in p5JS

Deciding with *if*...

```
let a = 10;

if (a > 0)
{
    println("a is ", a);
    println("a is positive");
}

println("Conditions are great");
```

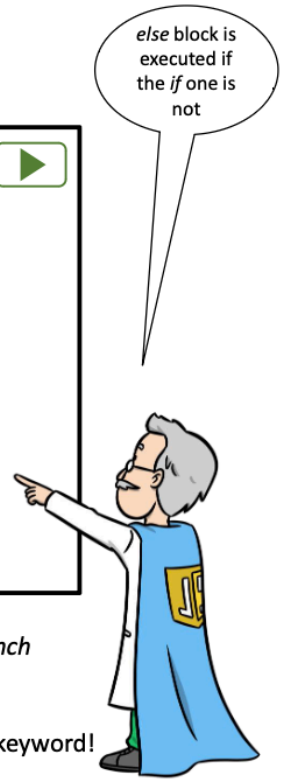
- Type carefully this small program and then run it. What is the output?
- Now modify the first line of code, and instead of 10 put there a negative number. What do you see now?

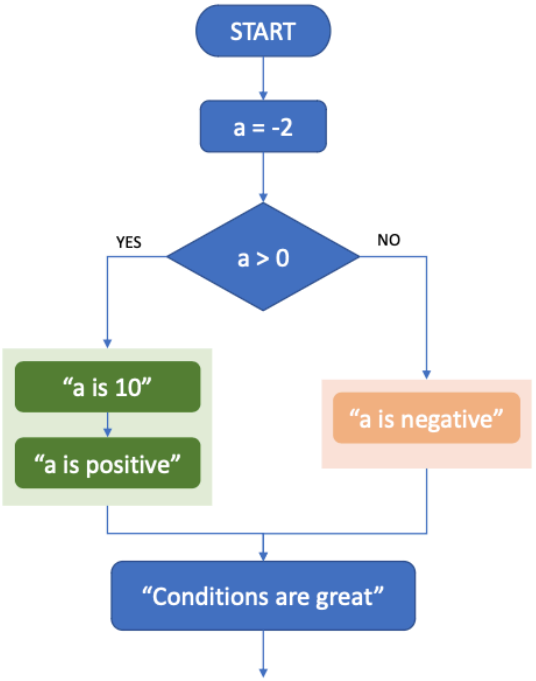


If condition == false

What *else*?

```
let a = -2;

if (a > 0)
{
    println("a is ", a);
    println("a is positive");
}

else
{
    println("a is negative");
}

println("Conditions are great");
```





- Modify the program to include also an *else branch* followed by a new code block
- Don’t use any parenthesis or symbol after else keyword!

Practice

Write the appropriate if conditions in place of `...` line in your code.

Exercise: Rating system

Let's build a simple rating system using *if* / *else-if* statements.
The program needs to display the appropriate message based on the actual rating from variable *rating*

```
let rating = 5;

...
println("Excellent!!!");
...
println("Good");
...
println("Average");
...
println("Below average");
```

If rating is 5!

If rating >= 4

If rating >= 3

Otherwise

★ ★ ★ ★ ★

Excellent!!!

"First, solve the problem.

Then, write the code."

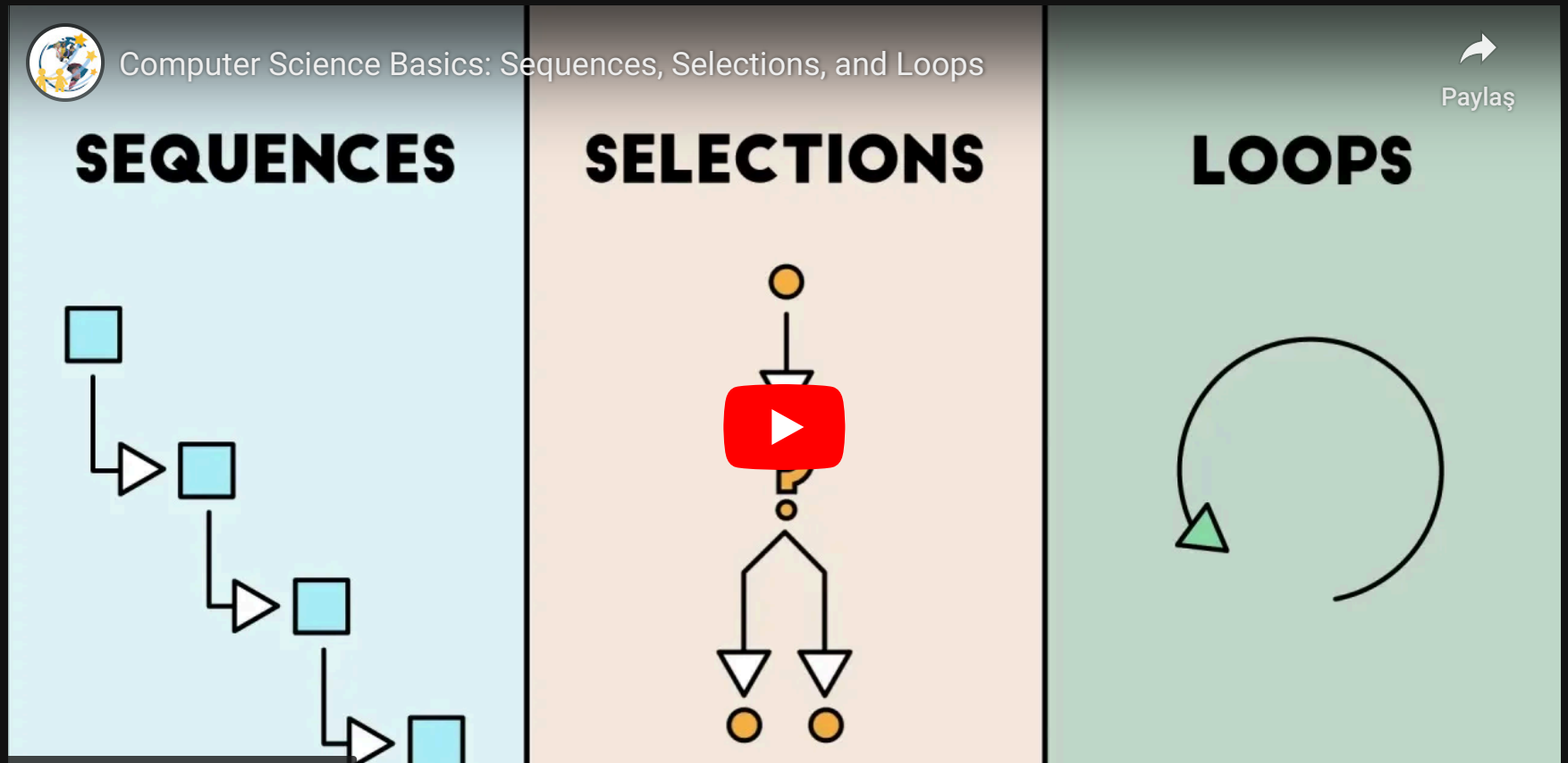
John Johnson

BREAK

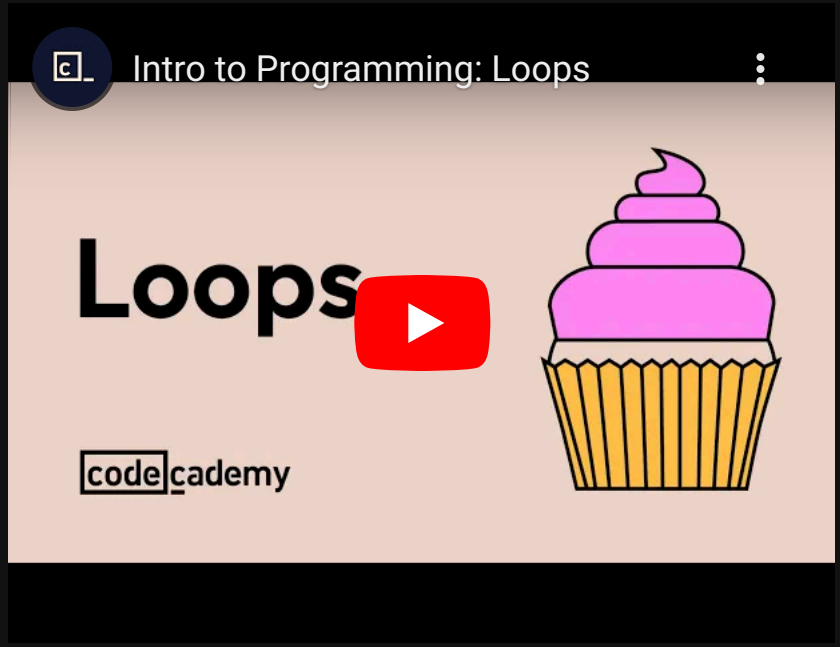
12 mins.

Computer Science Basics

1 Sequences, 2 Selection, and 3 Loops



What are loops in coding?



for loops

Repeats a section of code or code-block a limited number of times. Three steps of creating for loops;

1. Create a counter variable.
2. Set the repetition count.
3. Set the counter behaviour. Is it gonna increase one by one, two by two, etc...

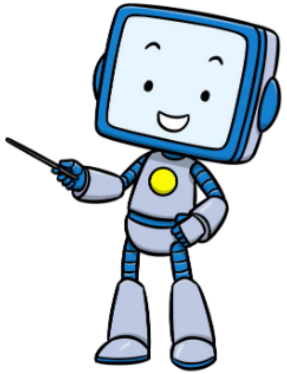
for loops in p5JS

Reading the for loop the easy way…

FOR | i = 0 | TO | 10 (exclusive)

Execute block { }

```
for(let i = 0; i < 10; i++)
{
    print("Hello World");
}
```



Replace the `print("...")` function with a shape

Do you see a single shape? Where are the other shapes?

Open the black box

for is executing the code block for i = 0 … 10 (exclusive)

```
for(let i = 0; i < 10; i++)
{
    ...
    ...
}
```

initialization

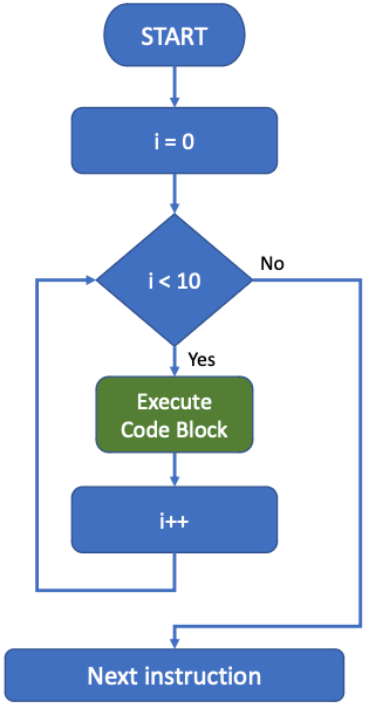
condition

variable update (increment)

repeat these lines as long as *i* is less than 10

i = 0 execute {…}
i = 1 execute {…}
i = 3 execute {…}
i = 4 execute {…}
i = 5 execute {…}
i = 6 execute {…}
i = 7 execute {…}
i = 8 execute {…}
i = 9 execute {…}

Open the black box



for works by utilizing a variable…

Step 1: JavaScript executes the initialization part just once for each for loop. Here we're declaring variable *i*.

Step 2: Then checks the condition, and if the condition is satisfied, it will execute the code block, otherwise the for loop is ending.

Step3: Code Block is executed

Step 4: After code block execution the variable is updated according to the statement in the 3rd position. Here we're incrementing *i*.

Step 5: JavaScript repeats from Step 2

Access the counter variable to modify the repeating code

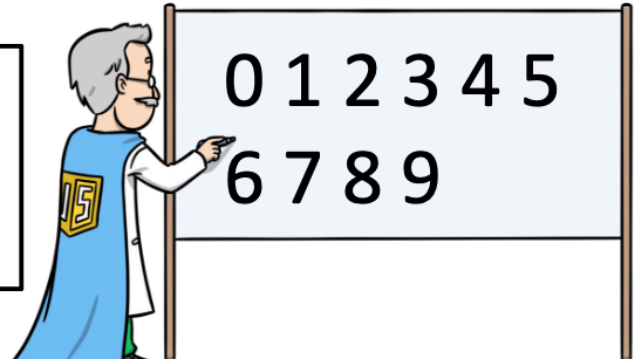
Use the counter variable `i` as a dynamic value.

Accessing for loop variable inside the code block

- Did you know that you can make use of the for variable inside the code block?
- The code block is executed *n* times, and each time *i* has a different value:

  i = 0 then execute {...}
  i = 1 then execute {...}
  i = 2 then execute {...}
  ...

```
for(let i = 0; i < 10; i++)
{
    println(i);
}
```



Tutorial

Loops & Repetation
by alptugan

Fork

Let's write an example to better illustrate the inefficient way of coding a concept. This sketch simply displays five circles, all in a row, aligned in the middle of the screen:

As you might see;

"There's a lot of repetition here between the line 19-23," you probably said to yourself.

The multiplication operator modifies the x

1/3

mySketch

```
1  var ypos = 0;
2  var xpos = 0;
3  var xstep = 0;
4
5  function setup() {
6    createCanvas(400, 400);
7
8    xpos = 20;
9    ypos = height / 2;
10   xstep = 90;
11   noLoop();
12 }
```

link to openProcessing 🔗

Assignments

1. Make a 6 x 6 grid. Canvas size must be 600 x 600 px.
2. Each grid must be a square with the same witdh and height in px.
3. Draw circles in the middle of each grid in random size between of your choice in min and max value.
4. You must use variables 10 PTS
5. You must use for loops 10 PTS
6. You must use comments in your code 10 PTS
7. ❗ Submit the openprocessing link 5 PTS
8. ❗ Submit the sketch source code as zip file as well 5 PTS

Foods for Source

📖 Read the tutorial 🔗

📺 Watch the video 🔗.

📺 Then watch the following tutorial 🔗

Assignment 2

Choose an object and redesign the figure with simple shapes in an abstract form.

1. Computational Thinking Analysis 20 PTS
2. Canvas size 600 x 600 px 10 PTS
3. Create 3 x 3 grid of squares in total 9 squares 20 PTS
4. Use only 2 different colors 10 PTS
5. Use variables 10 PTS
6. Use comments in your code 10 PTS
7. Clean code 10 PTS
8. ❗ Submit the openprocessing link 5 PTS
9. ❗ Submit the sketch source code as zip file as well 5 PTS

Assignment Reference Image

You can choose your own colors, and other shape properties.