CS106 W21
Week 1

Introduction to Open Processing

and

Recap of CS105

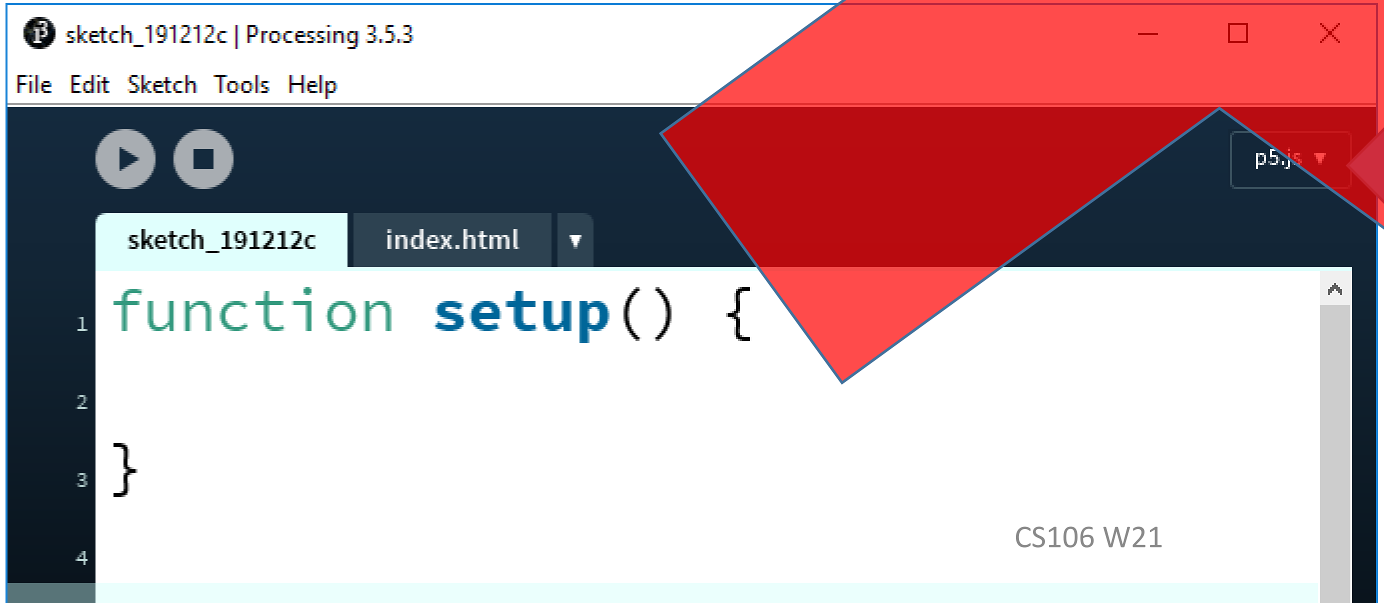
JavaScript p5

- JavaScript is a programming language
  - The programming language of the internet
- p5 is a JavaScript library (p5.js)
- In CS106, we create and edit JavaScript p5 files using Open Processing
- In CS106, we debug JavaScript p5 code using
  - Open Processing console
  - Your browser’s debugger

https://openprocessing.org

Processing IDE

- Integrated Development Environment
  - A JavaScript p5 editor and more
- Download from: https://processing.org/download/
- Must use p5.js mode (upper right corner)


sketch_191212c | Processing 3.5.3
File Edit Sketch Tools Help
p5.js
sketch_191212c
index.html

```
function setup() {

}
```

P5.js mode

Open Processing editor

- "Open Processing" is a web-based editor for JavaScript P5.
  - Your files are stored in the cloud
- This editor replaces the Processing IDE editor we used in CS105.
- CS106 students covered by the uWaterloo license called "Professor Plus+".
- You must sign up to get your CS106 student "Plus+" account.
- To sign up, go to the following url and enter code: 60EB5E

https://www.openprocessing.org/class/66897

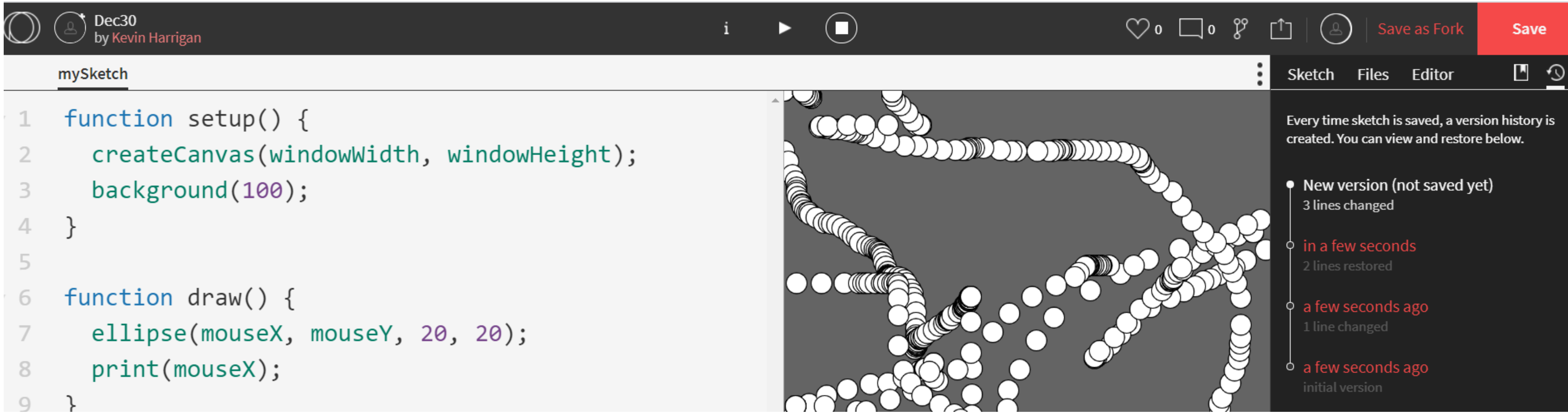
Open Processing editor

- “Introduction to Open Processing by Daniel Shiffman.
- First 3 minutes only:

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

Open Processing editor

- An Introductory video of the Open Processing editor.



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

function draw() {
  ellipse(mouseX, mouseY, 20, 20);
  print(mouseX);
}
```

Every time sketch is saved, a version history is created. You can view and restore below.

New version (not saved yet)
3 lines changed

in a few seconds
2 lines restored

a few seconds ago
1 line changed

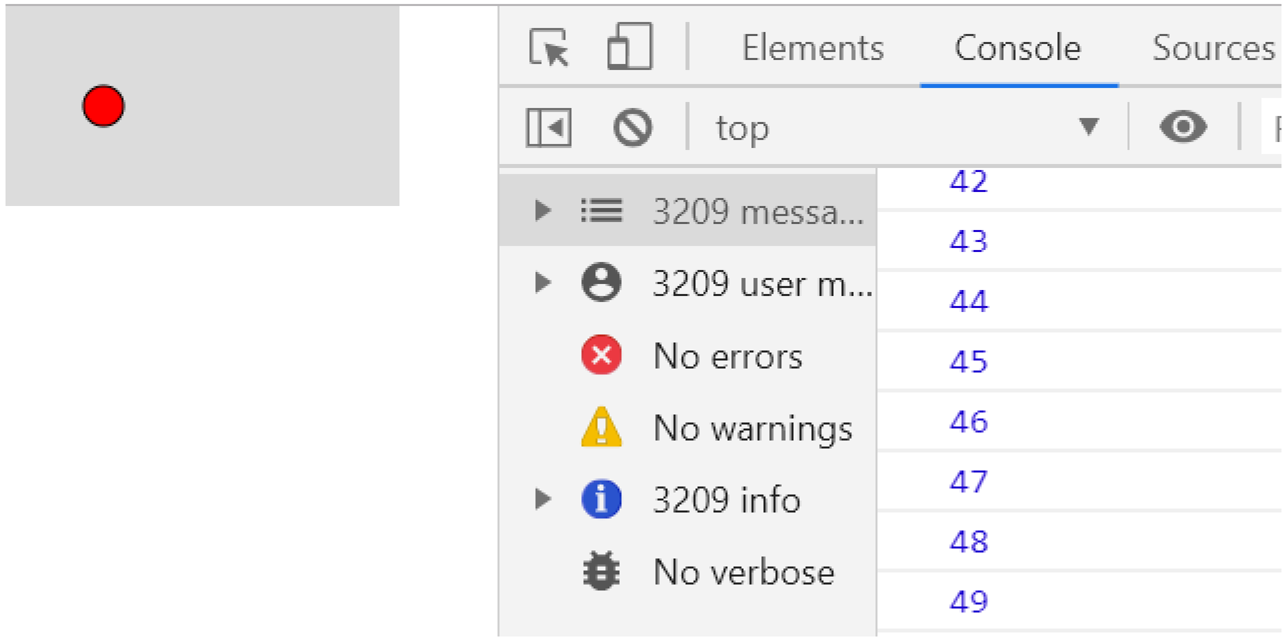
a few seconds ago
initial version

Open Processing editor

- Autoformat
  - CTRL + b
- Comment/uncomment each highlighted line
  - CTRL + /
- Use the Code Style Sheet

The Chrome Debugger

- Then, to open the developer **console** window **on Chrome**…
  - Use the keyboard **shortcut** Ctrl Shift J (**on** Windows) or
  - Ctrl Option J (**on** Mac) or
  - Using the **Chrome** menu, select "More Tools," and then "**Developer Tools**."



```
42
43
44
45
46
47
48
49
```

Review of topics from CS105

- Code Style Sheet
- Variables
- Conditions
- Loops
- Functions
- Arrays
- Program design
- Images
- Sound & Video

Variables

- Built-in (also called System) variables
  - width, mouseX
- Constants (all caps)
  - CENTER, PI, RIGHT
- User-defined variables
  - rectSize, count, i

User-defined variables

- Declared using “let”, In CS105/106, all variables must be declared.
- Variables in JavaScript p5 are not directly associated with a type
- A variable can be assigned (and re-assigned) values of all types such as:
  - integer, float, Boolean, string, color, and more

```
let b = 10;
let size = 10.3;
let gameOn = true;
let city = "Waterloo";
let ballColour = color(225, 0, 0);
```

Demo of Data Types and Functions

https://www.openprocessing.org/sketch/1050941#

```
let rectSize = 30;

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

function draw() {
  background(220);
  rectMode(CENTER);
  rect(width / 2, height / 2, rectSize, rectSize);
}

function keyPressed() {
  rectSize = random(20, 100);
}
```

Conditionals

https://www.openprocessing.org/sketch/1050942

```
let ballX = 0;

function setup() {
  createCanvas(200, 100);
}

function draw() {
  background(220);
  ellipse(ballX, height / 2, 20, 20);
  ballX = ballX + 1.0;
  if (ballX > width) {
    ballX = 0;
  }
}
```

Conditionals using &&

https://www.openprocessing.org/sketch/1050943

```
function setup() {
  createCanvas(600, 400);
  background(100);
}

function draw() {
  background(220);
  if (mouseX > width / 2 && mouseY > height / 2) {
    rect(mouseX, mouseY, 20, 20);
  }
}
```

else, nested conditions events, and Boolean

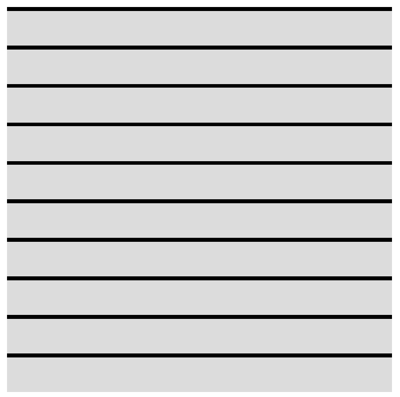
https://www.openprocessing.org/sketch/1050944

```
let gameOn = true;

function setup() {
  createCanvas(600, 400);
  background(100);
}

function draw() {
  background(220);
  if (gameOn) {
    if (mouseX > width / 2 && mouseY > height / 2) {
      rect(mouseX, mouseY, 20, 20);
    } else {
      ellipse(mouseX, mouseY, 10, 10);
    }
  }
}

function keyPressed() {
  gameOn = !gameOn;
}
```

While Loop

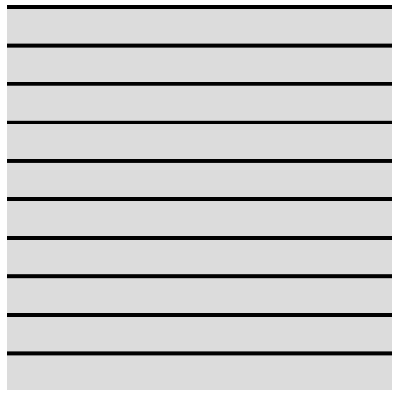
https://www.openprocessing.org/sketch/1050945

```
function draw() {
  background(220);
  let y = 0;
  while ( y < height ) {
    line( 0, y, width, y );
    y = y + 10;
  }
}
```



For Loop

https://www.openprocessing.org/sketch/1050946



```
function draw() {
  background(220);
  for (let y = 0; y < height; y += 10) {
    line(0, y, width, y);
  }
}
```

Built-In Functions

- setup()
- draw()
- keyPressed()
- mousePressed()
- Some built-in functions have parameters and return a value
  - random(2, 10)
  - dist(x1, y1, x2, y2)

User-Defined Functions

- Give a name to a block of code
- Benefits
  - Easy of reuse
  - Encapsulation – hides the messy details
  - Abstraction – think about problem solving at a higher level
  - Establish a point of connection between parts of your program
- Must be defined using “function”
  - function myFunc() {
- May have parameters
- May return a value

Demo of Functions and Hit Test

https://www.openprocessing.org/sketch/1050947

```
let hit;

function draw() {
  background(220);
  ellipse(width / 2, height / 2, 30, 30);
  hit = circleHittest(mouseX, mouseY, width / 2, height / 2, 30);
  text(hit, 10, 10);
}

function circleHittest(x1, y1, cx, cy, s) {
  return (dist(x1, y1, cx, cy) < s / 2);
}
```

Arrays

- A sequence of values
- For example, monthly max temperatures
  - Ontario

Month	Max Temp
Jan	-1
Feb	0
Mar	5
Apr	12
May	18
Jun	24
Jul	27
Aug	26
Sep	21
Oct	14
Nov	8
Dec	2

Arrays – Loop through array of Strings

https://www.openprocessing.org/sketch/1050948

```
let month = ["Jan", "Feb", "Mar", "Apr",
  "May", "Jun", "Jul", "Aug",
  "Sep", "Oct", "Nov", "Dec"];

function setup() {
  createCanvas(100, 400);
  background(220);
  textSize(25);
  for (let i = 0; i < month.length; i++) {
    text(month[i], 10, (i * 30) + 30);
  }
}
```

Find the Largest in an Array

https://www.openprocessing.org/sketch/1050949

```
let hiMonth = [-1, 0, 5, 12, 18, 24, 27, 26, 21, 14, 8, 2];
function setup() {
  createCanvas(400, 100);
  background(220);
  textSize(25);
  let largest = hiMonth[0];
  for (let i = 1; i < hiMonth.length; i++) {
    if (hiMonth[i] > largest) {
      largest = hiMonth[i];
    }
  }
  text("Temperature of hottest month: " + largest, 10, 50);
}
```

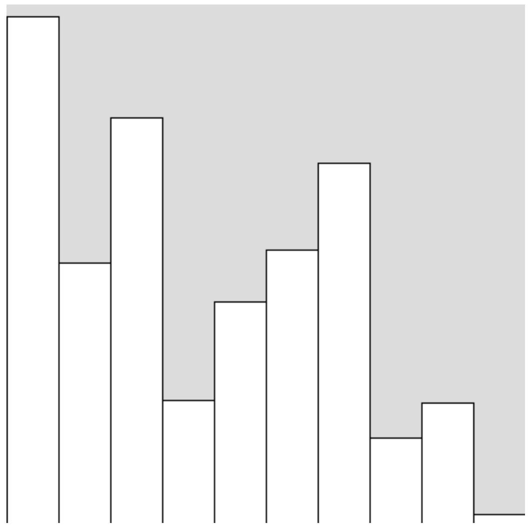
Initialize a large Array

https://www.openprocessing.org/sketch/1050950

```
let arr = [];
let numBars = 10;

function setup() {
  createCanvas(400, 400);
  background(220);
  for (let i = 0; i < numBars; i++) {
    arr[i] = floor(random(0, height));
  }
}
```

Visualize the data as a Bar Graph

https://www.openprocessing.org/sketch/1050951

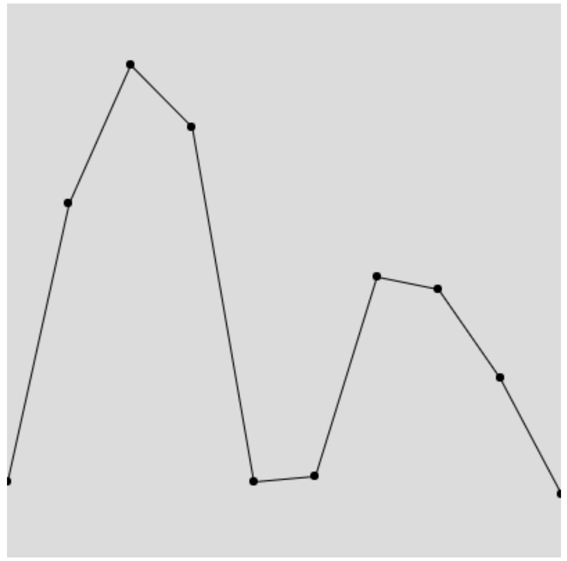


```
let arr = []; // declare array
let barWidth;
let numBars = 10;

//… setup() goes here, as on previous slide …

  barWidth = width / numBars;
  for (let i = 0; i < arr.length; i++) {
    rect(i * barWidth, height - arr[i], barWidth, arr[i]-1);
  }
}
```

Visualize the same data as a line graph

https://www.openprocessing.org/sketch/1050952

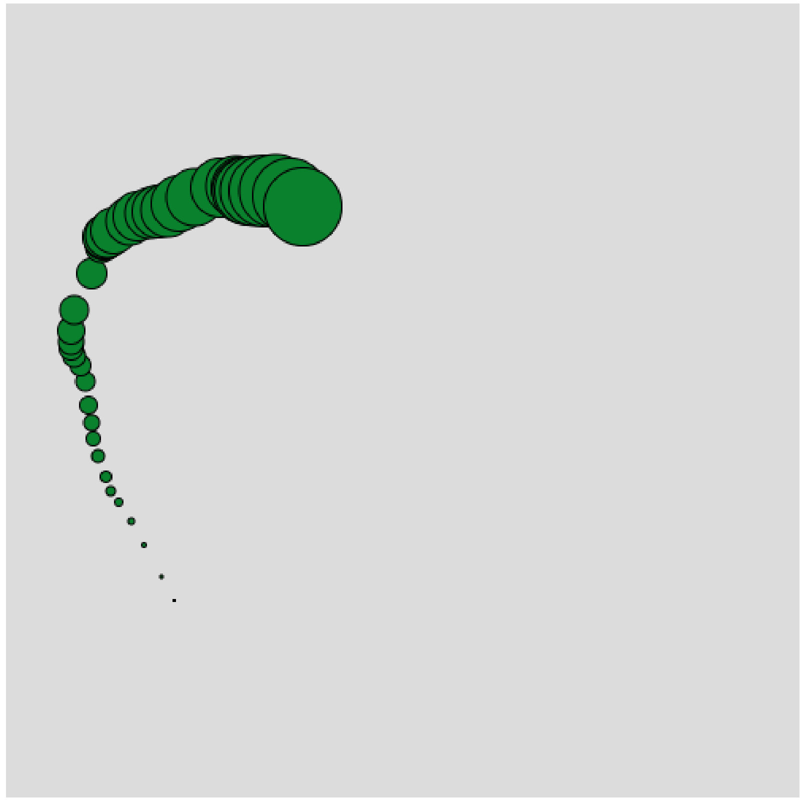
```
for (let i = 0; i < numBars; i++) {
   arr[i] = floor(random(0, height));
 }
 pointsWidth = width / (numBars - 1);
 for (let i = 0; i < arr.length; i++) {
   // draw the points
   let x = i * pointsWidth;
   let y = height - arr[i];
   strokeWeight(6);
   point(x, y);
   // draw the connecting lines
   if (i > 0) {
     let px = (i - 1) * pointsWidth;
     let py = height - arr[i - 1];
     strokeWeight(1);
     line(px, py, x, y);
   }
 }
```



Snake: Modular Design

https://www.openprocessing.org/sketch/1050953

```
let snakeX = [];
let snakeY = [];

function setup() {
  createCanvas(500, 500);
  initialization();
}

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



Snake: Modular Design con’t

```
function initialization() {
  for (i = 0; i < 50; i++) {
    snakeX[i] = 0;
    snakeY[i] = 0;
  }
}
```

Snake: Modular Design con’t

```
function updateSnake() {
  for (let i = 0; i < snakeX.length - 1; i++) {
    snakeX[i] = snakeX[i + 1];
    snakeY[i] = snakeY[i + 1];
  }
  snakeX[snakeX.length - 1] = mouseX;
  snakeY[snakeY.length - 1] = mouseY;
}

function drawSnake() {
  for (let i = 0; i < snakeX.length; i++) {
    ellipse(snakeX[i], snakeY[i], i, i);
  }
}
```

Creating a Timer

- Let’s say we want something to happen every three seconds
- Using snake from previous slides
- Change fill every 3 seconds
- Built-in function millis()
    - Returns # of milliseconds since program started

Demo: Snake with a Timer to change Fill

https://www.openprocessing.org/sketch/1050955

- Add these variables

```
let savedTime;
let changeFillTime = 3000;
```

- Initialize saveTime to the current millis()

```
savedTime = millis();
```

- Determine when 3 seconds have passed

```
if (millis() - savedTime > changeFillTime) {
    fill(random(255), random(255), random(255));
    savedTime = millis();
  }
```

Demo: Snake ends after 10 seconds

https://www.openprocessing.org/sketch/1050956

- Add variables

```
let gameoverTime = 10000;
let gameOn = true;
```

- Check to determine if 10 seconds have passed

```
 if (millis() > gameoverTime) {
    gameOn = false;
  }
```

Demo: Snake ends after 10 seconds con’t

- Modify draw()

```
if (gameOn) {
   background(220);
   updateSnake();
   drawSnake();
 }
```

Demo: Animated

https://www.openprocessing.org/sketch/1050954

```
let counter = 0;

function setup() {
  createCanvas(500, 500);
}
```

Demo: Animated

```
function draw() {
  background(0);
  fill(200);
  stroke(255);
  strokeWeight(3);

  counter = counter + 1;
  if (counter === 61) {
    counter = 0;
  }
```

Demo: Animated

```
let aa = map( counter, 0, 60, 100, 400 );

  ellipse( aa, 100, 80, 80 );
  ellipse( 100, 500 - aa, 80, 80 );
  ellipse( 500 - aa, 400, 80, 80 );
  ellipse( 400, aa, 80, 80 );
}
```

Image Filters (1 of 3)

https://www.openprocessing.org/sketch/1051152

```
// press keys 1 to 9 to see different filters

let img;

// preload is an event function called before setup
function preload() {
  img = loadImage("data/bird.jpg");
}

function setup() {
  createCanvas(img.width, img.height);
}
```

Image Filters (2 of 3)

```
function draw() {
  // draw image first
  image(img, 0, 0);
    // then apply a filter
  if (key === "1") {
    filter(INVERT);
    label("INVERT");
  } else if (key === "2") {
    filter(THRESHOLD);
    label("THRESHOLD");
  } else if (key === "3") {
    filter(GRAY);
    label("GRAY");
  } else if (key === "4") {
    filter(DILATE);
    label("DILATE");
  } else if (key === "5") {
    filter(ERODE);
    label("ERODE");
```

```
  } else if (key === "6") {
    filter(POSTERIZE, 2);
    label("POSTERIZE 2");
  } else if (key === "7") {
    filter(POSTERIZE, 4);
    label("POSTERIZE 4");
   } else if (key === "8") {
    filter(BLUR, 3);
    label("BLUR 3");
  }  else if (key === "9") {
    filter(BLUR, 12);
    label("BLUR 12");
  }
}
```

Image Filters (3 of 3)

```
function label(s) {
  fill(0);
  rectMode(CENTER);
  rect(width/2, height - 20, 120, 20);
  textAlign(CENTER, CENTER);
  fill(255);
  textSize(16);
  text(s, width/2, height - 20);
}
```

Sound: Honk and Horn

https://www.openprocessing.org/sketch/1051155

```
let honk;
let horn;

function preload() {
  // load sound files from data directory
  honk = loadSound("honk.wav");
  horn = loadSound("horn.wav");
}

function setup() {
  background(220);
}

function mousePressed() {
  if (mouseX < 50) {
    honk.play();
  } else {
    horn.play();
  }
}
```

Video

https://www.openprocessing.org/sketch/1051157

```
let camera;

function setup() {
  createCanvas(320, 240);

  // start video capture
  camera = createCapture(VIDEO);
  // set size of capture frame
  camera.size(width, height);
  // hide the original HTML video object
  camera.hide();

  background(220);
}

function draw() {
  image(camera, 0, 0);
}
```

What is printed to the console ?

https://www.openprocessing.org/sketch/1050957

```
let bar = [];


function setup() {
  bar[0] = 5;
  bar[1] = 4;
  bar[2] = 3;
  bar[3] = 2;

  print(bar[2]);
}
```

A. 1
B. 2
C. 3
D. 4
E. 5

What is printed to the console ?

https://www.openprocessing.org/sketch/1050958

```
let bar = [];


function setup() {
  bar[0] = 5;
  bar[1] = 4;
  bar[2] = bar[0] - 1;
  bar[3] = 2;


  print(bar[2]);
}
```

A. 1
B. 2
C. 3
D. 4
E. 5

What is printed to the console ?

https://www.openprocessing.org/sketch/1050959

```
let bar = [];

function setup() {
  bar[0] = 5;
  bar[1] = 4;
  bar[2] = 3;
  bar[3] = 2;
  bar[4] = bar[bar[2]];

  print(bar[4]);
}
```

A. 1
B. 2
C. 3
D. 4
E. 5

What is printed to the console?

https://www.openprocessing.org/sketch/1050960

```
let a = 1;

function setup() {
  go(a);
  go(a + 1);
  print(a);
}

function go(b) {
  b = b + 1;
}
```

A. 1

B. 2

C. 3

D. 4

E. undefined

What is printed to the console?

https://www.openprocessing.org/sketch/1050961

```
let a = [1, 2, 3, 4];

function setup() {
  let v = 1;
  for (let i = 0; i < a.length;
i++)
  {
    v = v * a[i];
  }
  print(v);
}
```

A. 0

B. 1

C. 4

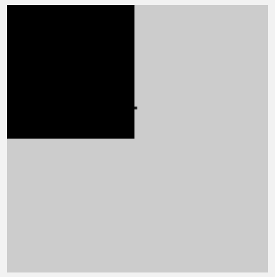
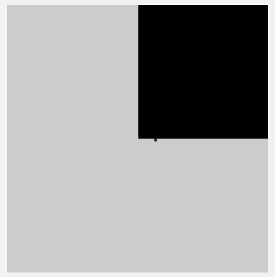
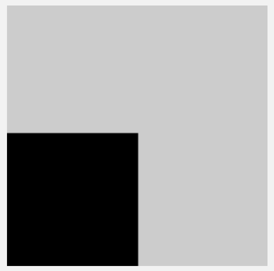
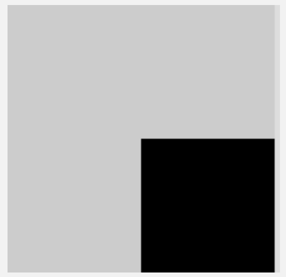
D. 10

E. 24

What does this draw after 1,000,000 frames?

https://www.openprocessing.org/sketch/1050962

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

function draw() {
  point(max(random(0, width), 50),
    max(random(0, height), 50));
}
```

What does this draw after 1,000,000 frames?

https://www.openprocessing.org/sketch/1050963

```
function draw() {
    point(min(50, random(0, width)),
          max(random(0, height), 50));
}
```