Basic drawing techniques in p5:
ellipse(x, y, size, size)
rect(x, y, size, size)
point(x, y)
line(x1, y1, x2, y2)
stroke(color); strokeWeight(size);
fill(r, g, b, alpha(transparency))
setAlpha()
rotate(radians(angle))
beginShape(LINES/POINTS/TRIANGLES/QUADS);
vertex(x, y); -> adds vertices
endShape();

p5's built-in functions
function mousePressed(){}

User-controlled Characters
Keyboard
if (keyIsDown(DOWN_ARROW){}
Mouse
noCursor();
image(img, mouseX, mouseY);

Basic movement strategies
Wrap-around
if (x > width){ x = 0}
constrain to screen
y = constrain(y, min, max)
Acceleration
x += speed //or *=

computer controlled characters
Chase

```
//following the mouse
let xPos, yPos;
let xDesired, yDesired;

function setup() {
  createCanvas(500, 500);
  xPos = 250;
  yPos = 250;
  xDesired = 250;
  yDesired = 250;
  noStroke();
}
function draw() {
  background(0, 10);
  xDesired = mouseX;
  yDesired = mouseY;
  let distX = xDesired - xPos;
  let distY = yDesired - yPos;
  // move 5%
  xPos += 0.05 * distX;
  yPos += 0.05 * distY;
  fill(255);
  ellipse(xPos, yPos, 25, 25);
}
```

Parallax scrolling

sketch.js

```
//parallax scrolling
let bg, bg2;
let x = 0;
let x2 = 266;
function preload(){
  bg = loadImage("images/bg.jpg")
  bg2 = loadImage("images/bg-1.jpg")
}
function setup(){
  createCanvas(400, 266);
}
function draw(){
  image(bg, x, 0);
  image(bg2, x2, 0);

  if(x2 <= -266){
    x2 = x + 266
  }
  if(x <= -266){
    x = x2 + 266
  }
  //can be subsituted with speed
  x --
  x2 --
}
```

Basic HTML & CSS
Moving Canvas
let theCanvas = createCanvas(500, 500) //p5js
theCanvas.elt.style.display = 'absolute';
theCanvas.elt.style.top = '100px';
theCanvas.elt.style.left = '100px';
Re-Parenting
theCanvas.parent("#elementID"); //p5js
Input Elements
<form>input elements go here</form>
Button
<button id="button1" name="button1"
onclick="p5Function(); return false;">
Button Text Goes Here
</button>
Ranges
<input type="range" id="range1" min="0"
max="200" step="1" value="100"
onchange="updateRange(this);">
p5js:
function updateRange(theRange) {
let rangeData = int(theRange.value);}

Collision detection
if (dist(x1, y1, x2, y2) < 25) {//program}

Particle System (array, loop, oop)

```
//particle System
let particles = []
function setup() {
  createCanvas(400, 400);
}
function draw() {
  background(220);
  if (mouseIsPressed){
    for (let i = 0; i < 50; i++){
      particles.push(new Particle(mouseX, mouseY))
    }
  }
  for (let i = 0; i < particles.length; i++){
    particles[i].moveAndDisplay();
    let result = particles[i].checkIfDead()
    if (result){
      particles.splice(i, 1)
      i -= 1; //because splice moves the value
    }
  }
}
class Particle{
  constructor(x, y){
    this.x = x
    this.y = y
    this.size = random(20, 30);
    this.speedX = random(-3, 3);
    this.speedY = random(-3, 3);
  }
  moveAndDisplay(){
    this.x += this.speedX;
    this.y += this.speedY;
    this.size -= 0.2;
    this.size = constrain(this.size, 0, 30);
    noStroke();
    fill(128);
    ellipse(this.x, this.y, this.size, this.size);
  }
  checkIfDead(){
    if (this.size <= 0) {
        return true;
    }
    if (this.x < 0 || this.x > width) {
        return true;
    }
    if (this.y < 0 || this.y > height) {
        return true;
    }
    return false;
  }
}
```

Perlin noise

let noiseOffset = random(0, 1000);
let noiseValue = noise(this.noiseOffset);
//map(value, original x, original y, new x, new y)
let amountToMove = map(noise Value, 0, 1, -2, 2)
this.x += amountToMove;
this.noiseOffset += 0.01

Animated GIFs & PNG sequences

```
let spriteSheet;
let sprite1;

function preload() {
    spriteSheet = loadImage('bird_spritesheet.png');
}

function setup() {
    createCanvas(500, 500);
    sprite1 = new Sprite(width / 2, height / 2, 150, 100, spriteSheet);
}

function draw() {
    background(128);
    sprite1.display();
}

class Sprite {
    constructor(x, y, w, h, img) {
        this.x = x;
        this.y = y;
        this.w = w;//size
        this.h = h;
        this.img = img;//source img
        // compute how many frames we have
      //(img width/single img width)
        this.totalFrames = int(this.img.width / this.w);
        // keep track of which frame
        this.currentFrame = 0;
        // a pause counter to slow down the animation (optional)
        this.pauseCounter = 0;
        this.pauseCounterMax = 3;
    }
    display() {
        image(this.img, this.x, this.y, this.w, this.h,
            this.currentFrame * this.w, 0, this.w, this.h);
        // decrease our pause counter
        this.pauseCounter--;
        // frame cycle
        if (this.pauseCounter <= 0) {
            this.currentFrame += 1;
            if (this.currentFrame >= this.totalFrames) {
                this.currentFrame = 0;
            }
            this.pauseCounter = this.pauseCounterMax;
        }
    }

}
```

color detection

```
//get color
function setup() {
    createCanvas(500, 500);
    background(255);
    for (let i = 0; i < 50; i++) {
        fill(random(255), random(255), random(255));
        ellipse(random(10, width - 10),
                random(10, height - 10), 20, 20);
    }
}

function draw() {
    let c = get(mouseX, mouseY);
    let r = ■red(c);
    let g = ■green(c);
    let b = ■blue(c);
  console.log("■red:" + r, "■blue:" +
              b, "■green:" + g)

}
```

Off-screen graphics buffers

```
let buffer;
function setup() {
    createCanvas(500, 500);
    buffer = createGraphics(500, 500);
    buffer.background(255);
    buffer.strokeWeight(20);
    buffer.stroke(10);
}
function draw() {
    image(buffer, 0, 0);
    if (mouseIsPressed) {
        buffer.line(mouseX, mouseY, pmouseX, pmouseY);
    }
}
//in html
//<button onclick="drawMode();">Draw Mode</button>
//<button onclick="eraseMode();">Erase Mode</button>
function drawMode() {
    buffer.stroke(0);
}
function eraseMode() {
    buffer.stroke(255);
}
```