Loops

Loop Statements
Variable Scope
Remapping
Nested Loops

Textbook Readings:
Chapter 4, Examples 4-5, 4-6, 4-7, 4-8, 4-9, 4-10, 4-11, 4-12, 4-13
Chapter 5, Examples 5-22, 5-23

Coding Train Videos:
4.1: while and for Loops https://youtu.be/cnRD9o6odjk
4.2: Nested Loops https://youtu.be/1c1_TMdf8b8

Donald Judd
while (expression) {
  statements
}

Starter: https://editor.p5js.org/cs105/sketches/kP_6CK7fN

https://editor.p5js.org/cs105/sketches/eoRaCq9M

Four Loop Questions

1. What do I want to repeat?
   - a rect

2. What do I want to change each time?
   - the y position of the rect

3. Where does it start, how does it change?
   - start at y = 10, draw a rect every 20 pixels

4. How long should it repeat?
   - until it reaches the bottom of the canvas
Anatomy of a `while` Loop Statement

```c
y = 10;

while (y < height) {
    rect(50, y, 50, 10);
    y += 20;
}
```

code block

See “07 Loops (trace)”
Chrome Debugger Demo

- Watch the loop execute using an “interactive debugger”

Guide to using Chrome Debugger
https://javascript.info/debugging-chrome

Code to debug (but run this code from Processing IDE):
https://editor.p5js.org/cs105/sketches/VAeK60UBn

spaced-lines

draw lines all the way across canvas with equal spacing (20)

line position variable

Variations:
- margin
- right to left
- calculate spacing based on width to draw exact number of lines.

https://editor.p5js.org/cs105/sketches/OmmqVprVz
**number-of-lines**

draws an exact number of lines (7) with fixed spacing (20)

- counter variable
- line position variable

https://editor.p5js.org/cs105/sketches/ZyPAKsioE

**draw() frames vs. while loop iterations**

- the code block in `draw` is repeated every `frame`
- the code block in `while` is repeated every `loop iteration`

```javascript
function draw() {
  while (x < width) {
    // loop iterations
  }
}
```
What do you see after 5 frames?

```javascript
let x = 0;
function draw() {
    background(200);
    x = 20;
    while (x < width) {
        ellipse(x, 50, 20, 20);
        x = x + 20;
    }
}
```

What do you see after 5 frames?

```javascript
let x = 20;
function draw() {
    background(200);
    while (x < width) {
        ellipse(x, 50, 20, 20);
        x = x + 20;
    }
}
```
What do you see after 5 frames?

Common Logic Errors Leading to Infinite Loops

Adding a semicolon after the boolean expression means no code block. Without a code block, the loop variable won’t update.

```javascript
x = 0;
while (x < width) { 
    line(x, 10, x, height - 10);
    x = x + 10;
}
```

The while statement will never end, and the program will freeze (infinite loop).

Fix this by removing the semicolon after the boolean expression.

```javascript
x = 0;
while (x < width) {
    line(x, 10, x, height - 10);
    x = x - 10;
}
```

The loop variable is updated such that the loop condition will always be true.

Without a code block, the loop variable won’t update.

```javascript
x = 0;
while (x < width); { 
    line(x, 10, x, height - 10);
    x = x + 10;
}
```

The while statement will never end, and the program will freeze (infinite loop).

Fix this by ensuring that your code eventually makes the loop repetition expression false.
Debugging Loops

- Test a single loop operation first (without a loop)
- Simplify the loop operation
- Slow down draw frames with frameRate(1)
- Use print()

```javascript
x = 0;
print("start loop"); // debug
while (x < width) {
  print("loop", x); // debug
  line(x, 10, x, height - 10);
  x = x + 10;
}
print("done loop"); // debug
```

What do you see after 5 frames?
**gradient**

create a vertical grayscale gradient

eample of varying more than one variable

[Link](https://editor.p5js.org/cs105/sketches/U_fvzbRqx)

**(demos) convert while loop to for loop**

let i = 0;
function setup() {
  while (i < 10) {
    print("while loop ", i);
    i = i + 1;
  }
}

function setup() {
  for (let i = 0; i < 10; i++) {
    print("for loop ", i);
  }
}
Assignment Operator “Short Forms”

These all add 1 to x (they are all equivalent):
- \( x = x + 1; \)
- \( x += 1; \)
- \( x++; \)

These both add 5 to x (they are both equivalent):
- \( x = x + 5; \)
- \( x += 5; \)

Other examples
- // same as \( x = x + 10 \times y; \)
- \( x += 10 \times y; \)
- // same as \( x = x + \text{random}(-2, 2); \)
- \( x += \text{random}(-2, 2); \)

More Assignment Operator “Short Forms”

\( x--; \)

- \( x -= 10; // subtract 10 from x \)

- \( x *= 10; // multiply x by 10 \)

- \( x /= 10; // divide x by 10 \)
Anatomy of a For Loop

```javascript
for (let y = 10; y < height; y += 20) {
  rect(50, y, 50, 10);
}
```

code block

When to use for or while loop?

`for` is a short form for `while`, and they’re interchangeable.
(You can always use `while` if you want)

```javascript
y = 10;
while (y < height) {
  rect(50, y, 50, 10);
  y += 20;
}
```
When to use `for` or `while` loop?

Both are functionally equivalent, but sometimes one is easier to use in certain cases:

- Use `for` to **loop an exact number of repetitions**:
  - I want 10 pacmen
  - I want 3 lines

- Use `for` to **update by same amount**:
  - I want to count by 10 from 0 to 100
  - I want lines spaced 10 pixels apart the width of the canvas

- Use `while` to **update by different amounts and you don’t need a predetermined number of repetitions**:
  - I want lines spaced randomly 2 to 10 pixels apart over the whole width of the canvas

```
// position variable
x = 0;

for (let i = 0; i < num; i++) {
  line(x, 10, x, height - 10);
  x += spacing;
}
```

Starter: https://editor.p5js.org/cs105/sketches/ZyPAKsioE

https://editor.p5js.org/cs105/sketches/WWV7CJ-MI
**spaced-lines-for**

Use for to update by same amount
(a `for` loop version of spaced-lines)

```javascript
for (let x = 0; x < width; x += spacing) {
    line(x, 10, x, height - 10);
}
```

Starter: [https://editor.p5js.org/cs105/sketches/OmmqVprVz](https://editor.p5js.org/cs105/sketches/OmmqVprVz)
[https://editor.p5js.org/cs105/sketches/MtjMGylBD](https://editor.p5js.org/cs105/sketches/MtjMGylBD)

---

**What could you see after 2 frames?**
Scope

- What parts of your code have access to a variable
  - i.e. which variables can a code statement "use"
  - i.e. which variables can a code statement "see"

Global Scope

- We have mostly been declaring variables “outside” of functions
  - These are called global variables
- Global variables can be used anywhere in your program after they're declared
  - They have global scope
  - They "remember" their value between calls to setup(), draw(), etc.
  - Built-in Processing variables are global
    - e.g. width, frameCount, mouseX, key, mouseIsPressed

point(x, y)

CREDIT: http://www.biggamehunters.co.uk/
Local Scope

- You can declare variables “inside” any code block
  - These are called **local variables**
- Local Variables can be *used* anywhere in that code block *after* they're declared
- recall, a code block is anything in between { and }
  - loops have code blocks
  - conditionals have code blocks
  - functions have code blocks
- local variables *cease to exist after the code block is exited*
  - they have **local scope**

```javascript
function draw() {
  background(200);

  // x is a local variable only visible to draw()!
  let x = 0;

  while (x < width) {
    line(x, 10, x, height - 10);
    x += spacing;
  }
}
```

Starter: https://editor.p5js.org/cs105/sketches/OmmqVprVz
https://editor.p5js.org/cs105/sketches/sBVgkmfUT
Variable Scope Analogy

local (use it then throw it out)

global (remember it)

global (remember it and update it)

```javascript
let w = 100;
let y = w;

function setup() {
  let h = w * 3;
  createCanvas(w, h);
}

function draw() {
  background(240);
  line(0, y, w, y);
  let n = int(random(0, 500));
  for(let i = 0; i < n; i++) {
    let px = random(0, w);
    let py = random(0, y);
    point(px, py);
  }
  y--;
}
```
let w = 100;
let y = w;

function setup() {
  let h = w * 3;
  createCanvas(w, h);
}

function draw() {
  background(240);
  line(0, y, w, y);

  let n = let(random(0, 500));
  for(let i = 0; i < n; i++) {
    let px = random(0, w);
    let py = random(0, y);
    point(px, py);
  }

  y--;
}

let w = 100;
let y = w;

function setup() {
  let h = w * 3;
  createCanvas(w, h);
}

function draw() {
  background(240);
  line(0, y, w, y);

  let n = let(random(0, 500));
  for(let i = 0; i < n; i++) {
    let px = random(0, w);
    let py = random(0, y);
    point(px, py);
  }

  y--;
}
let w = 100;
let y = w;

function setup() {
  let h = w * 3;
  createCanvas(w, h);
}

function draw() {
  background(240);
  line(0, y, w, y);

  let n = let(random(0, 500));
  for(let i = 0; i < n; i++) {
    let px = random(0, w);
    let py = random(0, y);
    point(px, py);
  }
  y--;
}

let w = 100;
let y = w;

function setup() {
  let h = w * 3;
  createCanvas(w, h);
}

function draw() {
  background(240);
  line(0, y, w, y);

  let n = let(random(0, 500));
  for(let i = 0; i < n; i++) {
    let px = random(0, w);
    let py = random(0, y);
    point(px, py);
  }
  y--;
}
let w = 100;
let y = w;

function setup() {
  let h = w * 3;
  createCanvas(w, h);
}

function draw() {
  background(240);
  line(0, y, w, y);

  let n = let(random(0, 500));
  for(let i = 0; i < n; i++) {
    let px = random(0, w);
    let py = random(0, y);
    point(px, py);
  }
  y--;
}

let w = 100;
let y = w;

function setup() {
  let h = w * 3;
  createCanvas(w, h);
}

function draw() {
  background(240);
  line(0, y, w, y);

  let n = let(random(0, 500));
  for(let i = 0; i < n; i++) {
    let px = random(0, w);
    let py = random(0, y);
    point(px, py);
  }
  y--;
}
let w = 100;
let y = w;

global setup

let h = w * 3;
canvas(width: w, height: h);

for(let i = 0; i < n; i++) {
  let px = random(w);
  let py = random(h);
  point(px, py);
}
y--;
When to use local variables

- When you don't need to "remember" the value after the code block completes
  - a loop variable
  - an intermediate calculation
  - to re-use a common value in a function call

```
function draw() {
    ...

    // calculate start and end angles
    let start = radians(dir - mouthOpening/2);
    let end = radians(dir + mouthOpening/2);

    // use those angles to draw an arc
    arc(width/2, height/2,
        mouseX, mouseX,
        start, end);

    ...
}
```
function draw() {

... else if (shape == 3) {
   // half size
   let hs = size / 2;
   triangle(mouseX - hs, mouseY + hs,
           mouseX, mouseY - hs,
           mouseX + hs, mouseY + hs);

   }

...}

"out of scope" errors

function draw() {

... else if (shape == 3) {
   triangle(mouseX - hs, mouseY + hs,
           mouseX, mouseY - hs,
           mouseX + hs, mouseY + hs);

   // half size
   let hs = size / 2;

}

...}

(local variable for re-use)

function draw() {

... else if (shape == 3) {
   // half size
   let hs = size / 2;
   triangle(mouseX - hs, mouseY + hs,
           mouseX, mouseY - hs,
           mouseX + hs, mouseY + hs);

   }

...}

"out of scope" errors

function draw() {

... else if (shape == 3) {
   triangle(mouseX - hs, mouseY + hs,
           mouseX, mouseY - hs,
           mouseX + hs, mouseY + hs);

   // half size
   let hs = size / 2;

}

...}
("out of scope" errors)

```javascript
let x = 0;

function setup() {
    // local variable
    // only accessible to setup()
    let speed = 2;
}

function draw() {
    ellipse(x, 50, 20, 20);
    // this will cause an error: there is no
    // variable called 'speed' accessible to draw()
    x += speed;
}
```

"speed" is not defined

(speed is defined as a local variable in setup())

(logic errors with local variables)

```javascript
function draw() {
    let x = 0;
    ellipse(x, 50, 20, 20);
    x += 2;
}
```

x is declared as a local variable, it will have a value of 0 every frame.

the local variable x will be incremented, but then x will be thrown away as soon as draw finishes.
(logic errors with local variables)

```javascript
let x = 0;

function draw() {
    let x = 0;
    ellipse(x, 50, 20, 20);
    x += 2;
}
```

- `x` is declared as a local variable and it will "shadow" (hide) the global variable `x`.
- The local variable `x` will be assigned a value of 0 each frame.
- The local `x` will be incremented, but then `x` will be thrown away as soon as `draw` finishes.

What does this loop output?

```javascript
let a = 2;

function setup() {
    for (let i = 0; i < a; i++) {
        print("duck");
    }
    print("goose");
}
```

```
Duck Duck Duck Goose
Duck Goose Duck Goose
Duck Duck Goose Goose
Duck Goose Duck Goose
Duck Duck Goose Goose
```

```javascript
Duck Goose Duck Goose
```
What does this loop output?

```javascript
let a = 2;
function setup() {
    let a = 3;
    for (let i = 0; i < a; i++) {
      print("duck");
    }
    print("goose");
}
```

A B C
---
duck
duck
duck
goose
duck
goose
duck
goose
duck
goose
---
D

Changing Multiple “Things” in One Loop

Make a rainbow in a 100 by 400 canvas.
HINT: Use colorMode to switch to HSB colour model.

1. What do I want to repeat?
   - a line
2. What do I want to change each time?
   - the y position and the stroke hue
3. Where do they start, how do they change?
   - start y at 0, increment by 1
   - start hue at 0, increment by ???
4. How long should it repeat?
   - as long as y is less than the canvas height
p5* rainbow

Make a rainbow in a 100 by 400 canvas.

HINT: Use colorMode to switch to HSB colour model.

ideas:
1. [HACK] createCanvas(360, 100);

2. [GOOD] start hue at 0, increment so hue = 360 when y = height

3. [BEST] express hue as a “function of y”

Starter: https://editor.p5js.org/cs105/sketches/DB5GxfiW
https://editor.p5js.org/cs105/sketches/xgRuIstdFP

Remapping Height to Hue

![Diagram of y-coordinate and hue mapping](image)
Remapping with Height to Hue with Margin

\[ y \text{ coordinate} \]

\[ y_{\text{start}} \]
\[ y \]
\[ y_{\text{stop}} \]
\[ \text{height} \]

\[ \text{line hue.} \]
\[ 0 \]
\[ \text{hue} \]
\[ 360^\circ \]

\[ y_{\text{start}} = m \]
\[ y_{\text{stop}} = \text{height} - m \]

**p5** rainbow-margin

draw rainbow with top and bottom margin

https://editor.p5js.org/cs105/sketches/6JBfsBzM
**Remapping Different Scales**

- draw a gradient from hue 100° to 200°

\[
\text{value2} = \text{map(\text{value1}, \text{start1}, \text{stop1}, \text{start2}, \text{stop2})}
\]

- **value1**: the incoming value to be converted
- **start1**: lower bound of the value's **current** range
- **stop1**: upper bound of the value's **current** range
- **start2**: lower bound of the value's **target** range
- **stop2**: upper bound of the value's **target** range
remap variable using map()

let hue = map(y, yStart, yStop, hueStart, hueStop);

https://editor.p5js.org/cs105/sketches/kWxRVPKz9

What does this code print to the console?
Creating an infinite loop:

Typical causes:

- forgetting to change the counter variable
- changing the counter variable incorrectly
- using wrong test

An infinite loop is a logical error.

Note: This can almost always be avoided by using `for` loops!

infinite loop due to float precision

```javascript
let a = 0;

function setup() {
  frameRate(2);
}

function draw() {
  // infinite loop
  while (a !== 5) {
    // watch the console output!
    print(a);
    a += 0.1;
  }
}
```

[https://editor.p5js.org/cs105/sketches/3owVNGdti](https://editor.p5js.org/cs105/sketches/3owVNGdti)
What does this code print to the console?

```javascript
function setup() {
  let i = 0;
  for (let y = 0; y < 2; y++) {
    i = i + 1;
  }
  for (let x = 0; x < 3; x++) {
    i = i + 1;
  }
  print(i);
}
```

```
// draw many dots around the cursor
for (let i = 0; i < numDots; i++) {
  let x = mouseX + random(-spread, spread);
  let y = mouseY + random(-spread, spread);
  point(x, y);
}
```

[https://editor.p5js.org/cs105/sketches/xL4HuWhVz](https://editor.p5js.org/cs105/sketches/xL4HuWhVz)
... let x = 0;
while (x < width) {
    if (mouseX >= x && mouseX <= x + size &&
        mouseY >= y && mouseY <= y + size) {
        fill("#FF0000"); // red
    } else {
        fill("#FFFFFF");
    }
    rect(x, y, size, size);
    x += size;
}
...

https://editor.p5js.org/cs105/sketches/W71STybRF

Nested Loop

```javascript
for (let y = 0; y < height; y += 10) {
    for (let x = 0; x < width; x += 10) {
        rect(x, y, 10, 10);
    }
}
```
let s = 25; // size of each cell
let a = 0; // count the cells

// column
let x = 0;
for (let i = 0; i < 4; i++) {
    // row
    let y = 0;
    for (let j = 0; j < 3; j++) {
        // draw text at position x, y
        text(a, x, y);
        a = a + 1;
        y = y + s;
    }
    x = x + s;
}

https://editor.p5js.org/cs105/sketches/cpfJ9wHSd

Nested Loop Clock Analogy

for (let h = 0; h < 24; h++) {
    for (let m = 0; m < 60; m++) {
        for (let s = 0; s < 60; s++) {
            print(h, m, s);
        }
    }
}
for (let x = 0; x < width; x += 20) {
    // pick a hue for the column
    let hue = map(x, 0, width, 360, 0);
    fill(hue, 80, 80);
    // all cells in this column
    for (let y = 0; y <= height; y += 40) {
        rect(x, y, 20, 40);
    }
}

for (let y = 0; y <= height; y += 40) {
    // pick a hue for the row
    let hue = map(y, 0, width, 360, 0);
    fill(hue, 80, 80);
    // all cells in this row
    for (let x = 0; x < width; x += 20) {
        rect(x, y, 20, 40);
    }
}
What does this code output?

```javascript
function setup() {
  let i = 0;
  for (let y = 0; y < 2; y++) {
    for (let x = 0; x < 3; x++) {
      i++;
    }
  }
  print(i);
}
```