Loops & Nested Loops

CSE 120 Winter 2020

Instructor: Sam Wolfson

Teaching Assistants: Yae Kubota, Eunia Lee, Erika Wolfe

The EARN IT Act: How To Ban End-to-end Encryption Without Actually Banning It

"There's a new bill afoot in Congress called the EARN IT Act. This bill is trying to convert your anger at Big Tech into law enforcement's long-desired dream of banning strong encryption. It is a bait-and-switch. Don't fall for it. I'm going to explain where it came from, why it's happening now, why it's such an underhanded trick, and why it will not work for its stated purpose. And I'm only going to barely scratch the surface of the many, many problems with the bill."

https://cyberlaw.stanford.edu/blog/2020/01/earn-it-act-how-ban-end-end-encryption-without-actually-banning-it

Administrivia

- Creativity Project this week!
  - Planning document due tomorrow
    - Discuss your ideas with the TAs during section!
  - Final submission due Friday (2/7) on Canvas
- Portfolio Update 1 due Wednesday (2/5)
  - Convert your existing projects to p5.js and upload them to your portfolio
- Guest lecture this Friday: Data Visualization!
  - Reading check & presentations in section on Thursday

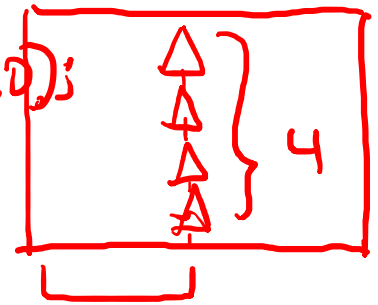
Quiz Recap

- Quiz 2 was more challenging than Quiz 1.

- Question 2 (Functions) was particularly difficult.

We have defined a function `drawTreeColumn(float x, color c)` that (1) draws a column of trees of color `c` across the height of the canvas, and (2) returns the total number of trees that it drew. Complete the code below to draw a *red* column of trees *centered horizontally* on the canvas and *print* the number of trees drawn to the console. Note that `x` specifies the *horizontal center* of a column.

```
void draw() {
  int n = drawTreeColumn(width/2, color(255,0,0));
  println(n);
}
```



- Our goal is to help you learn, and making mistakes is part of that process.

println(drawTreeColumn(width/2, color(255,0,0)));

Outline

- **Loops**
- Nested Loops

Looping

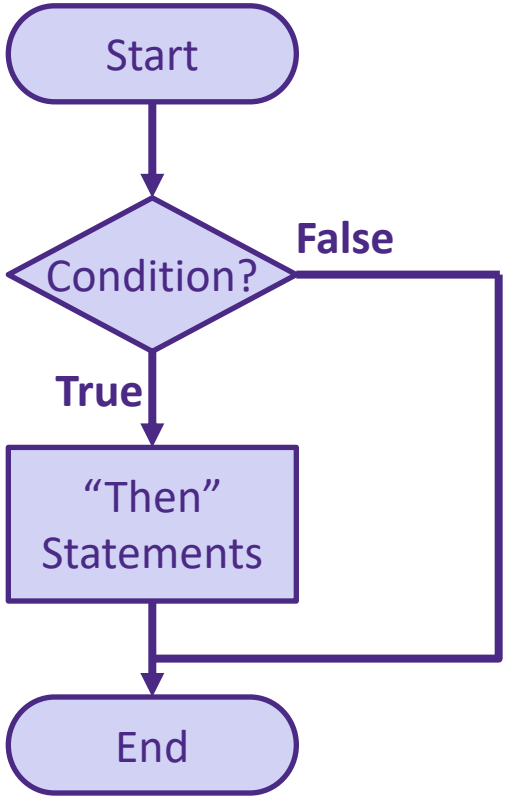
- Sometimes we want to do the same (or similar) things over and over again
  - Looping saves us time from writing out all the instructions

- Loops control a sequence of *repetitions*

But First: Remember If-Statements?

```
if (condition) {
    // "then"
    // statements
}
```

boolean



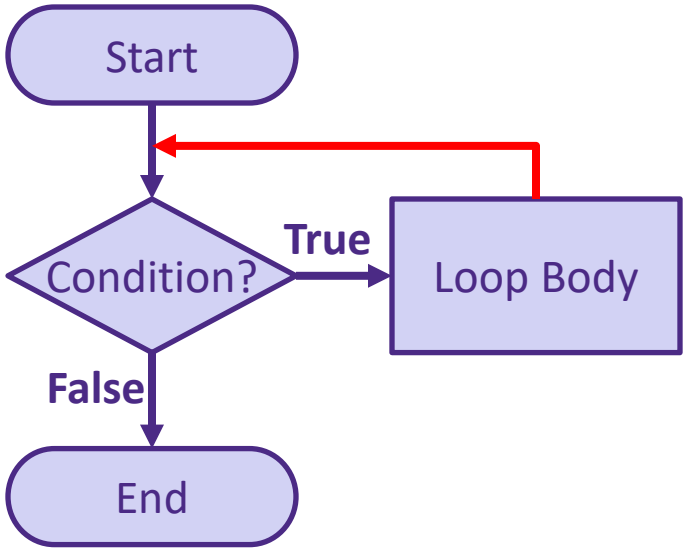
- Execute "then" statements if condition is `true`

While-Loop

- Basic form:

boolean

```
while (condition) {
    // loop
    // body
}
```



- Repeat loop body until condition is `false`
  - Must make sure to update conditional variable(s) in loop body, otherwise you cause an infinite loop

- **`draw()`** is basically a `while (true)` loop

While-Loop Example [Demo]

- Row of six animals:

```
void drawRow() {
  ???  // draw six mice
}

void drawMouse(float x, float y, color c) {
  ...  // drawing commands
}
```

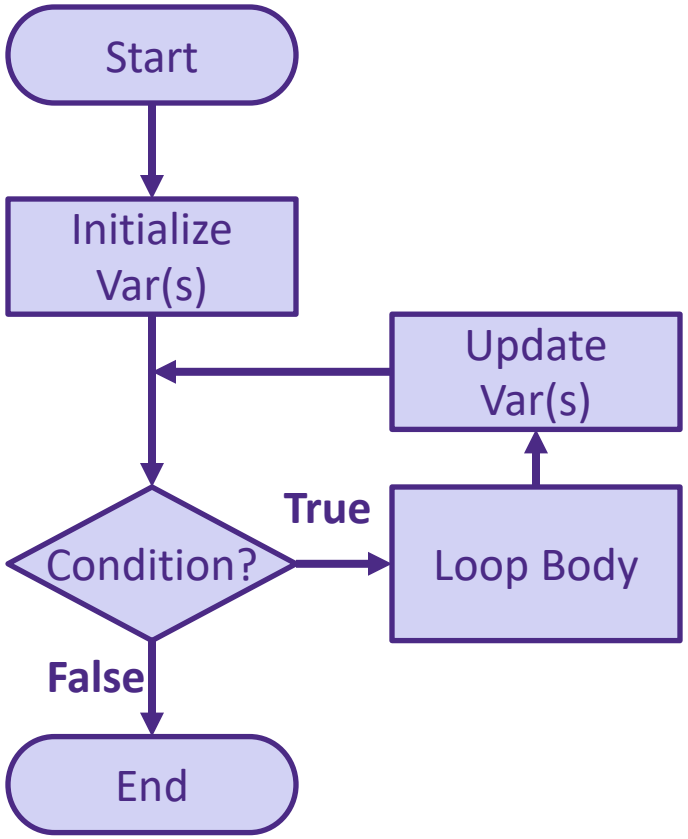
- Using a while-loop:

```
void drawRow() {
  int count = 0;
  while (count < 6) {
    drawMouse(80*count, 20, color(150));
    count = count + 1;
  }
}
```

While-Loop

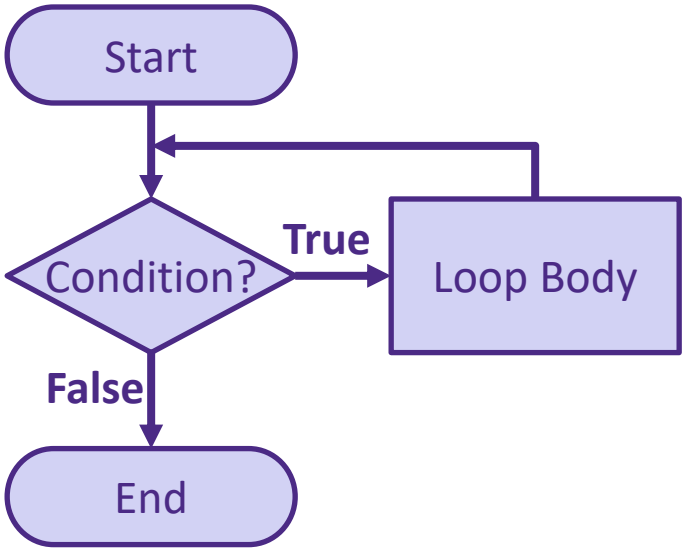
- More general form:

```
// init cond var(s)
int count = 0;
while (condition) {
    // loop body
    // update var(s)
    count = count + 1;
}
```

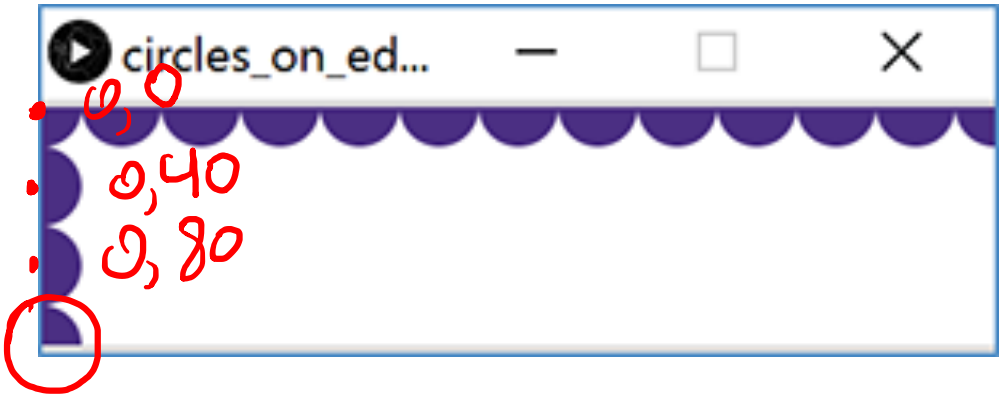


Loops Worksheet

```
while (condition) {
    // loop body
}
```



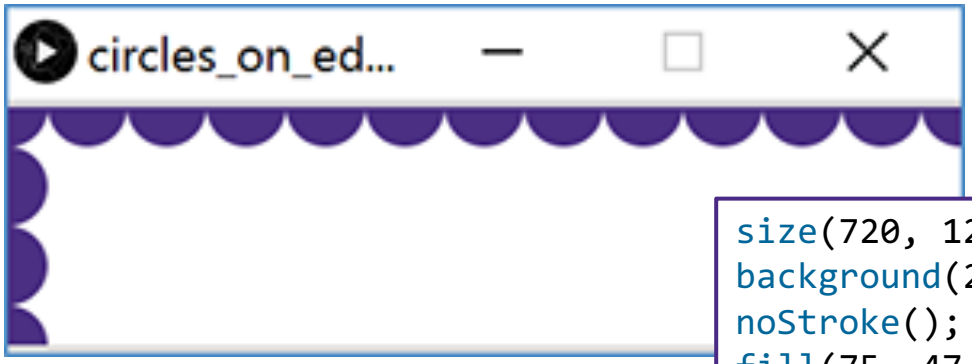
Processing Demo: Circles on Canvas Edge



40 px diameter

```
int i = 0;
while (i <= 3) {
   ellipse(0, i*40, 40, 40);
   i = i+1;
}
```

Processing Demo: Circles on Canvas Edge




```
size(720, 120);    // canvas size
background(255);   // white background
noStroke();        // no outline on circles
fill(75, 47, 131); // UW purple

int diam = 40;

// loop for circles along the top edge
int x = 0;
while (x <= width) {
  ellipse(x, 0, diam, diam);
  x = x+diam;
}

// loop for circles along the left edge
int y = 0;
while (y <= height) {
  ellipse(0, y, diam, diam);
  y = y+diam;
}
```

Outline

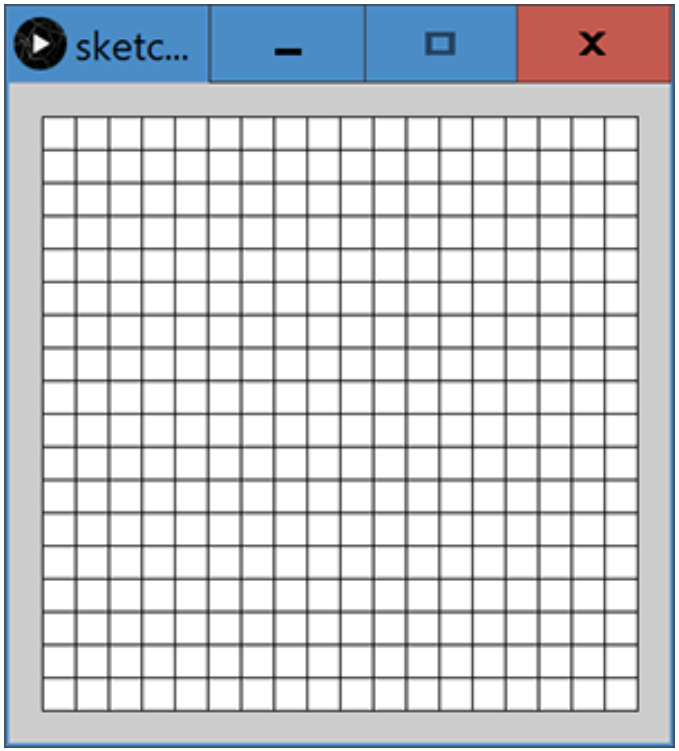
- Loops
- **Nested Loops**

Nested Loops

- Generally a loop has a single loop variable that changes with each iteration

- What if you need/want more things to change?
  - Can nest loops – *i.e.* put a loop inside of another loop

Processing Demo: Rectangle Grid




```
size(400, 400);

int y = 20;
while (y < height - 20) {
  int x = 20;
  while (x < width - 20) {
    rect(x, y, 20, 20);
    x = x + 20;
  }
  y = y + 20;
}
```

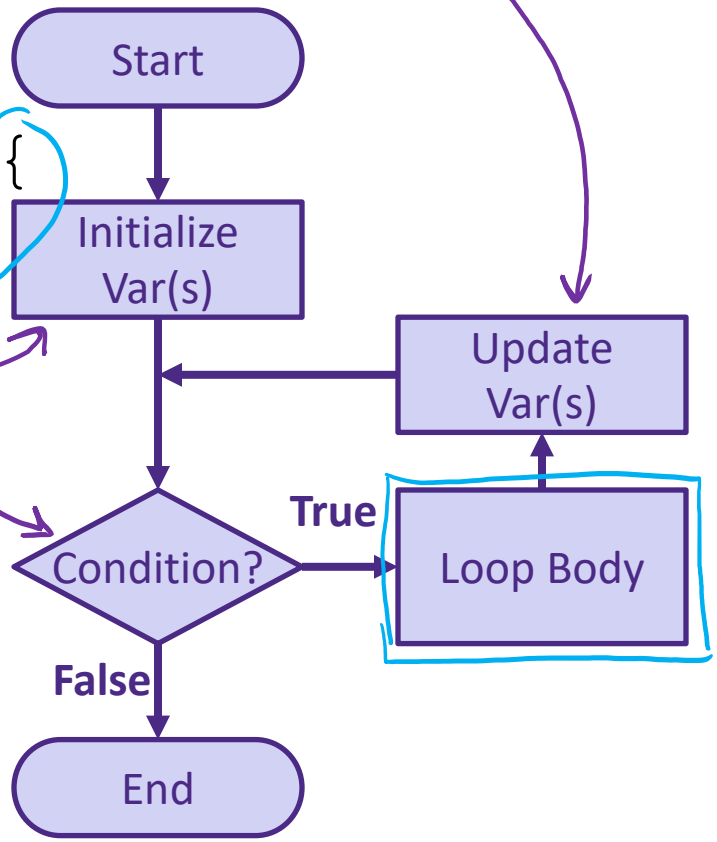
Additional Material For Loops

This material is optional, and you won't be tested on it, but it may help you to write more concise code.

For-Loop

separated by semicolons

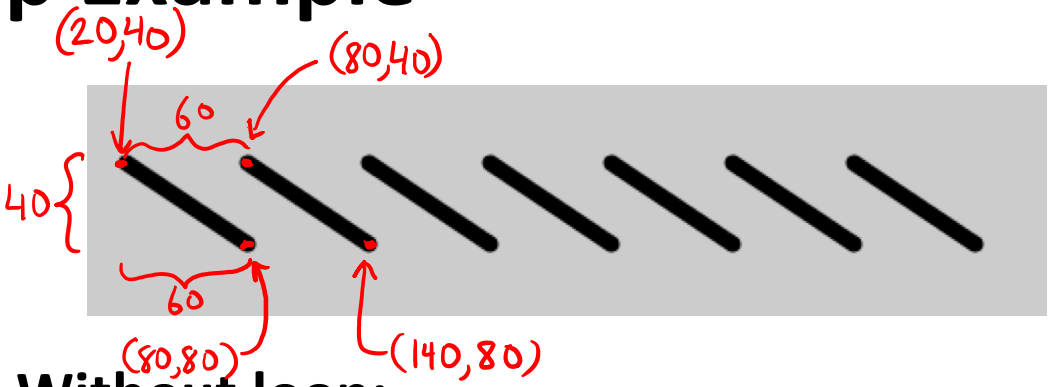
```
for (init; cond; update) {
    // loop body
}
```



- First runs *init* expression(s)
- Then checks *cond*
- If `true`, runs loop body followed by update statement(s)

- For-Loop can do exactly the same thing as a while-loop, but more compactly.

For-Loop Example



Without loop:

```
line( 20, 40,  80, 80);
line( 80, 40, 140, 80);
line(140, 40, 200, 80);
line(200, 40, 260, 80);
line(260, 40, 320, 80);
line(320, 40, 380, 80);
line(380, 40, 440, 80);
```

+60 +60 +60

always 40 always 80

With loop:

```
for (int i = 20; i < 400; i = i + 60) {
  line(i, 40, i + 60, 80);
}
```

init cond update stops once i=440

20 20
80 80

Understanding the For-Loop

initialization

```
for (int i = 20; i < 400; i = i + 60) {
  line(i, 40, i + 60, 80);
}
```

- Choice of variable name(s) is not critical
  - Represent the value(s) that vary between different executions of the loop body
  - Think of as temporary variable(s)

- If variable `i` is *declared* in the initialization statement, then it only exists *within this loop*

Understanding the For-Loop

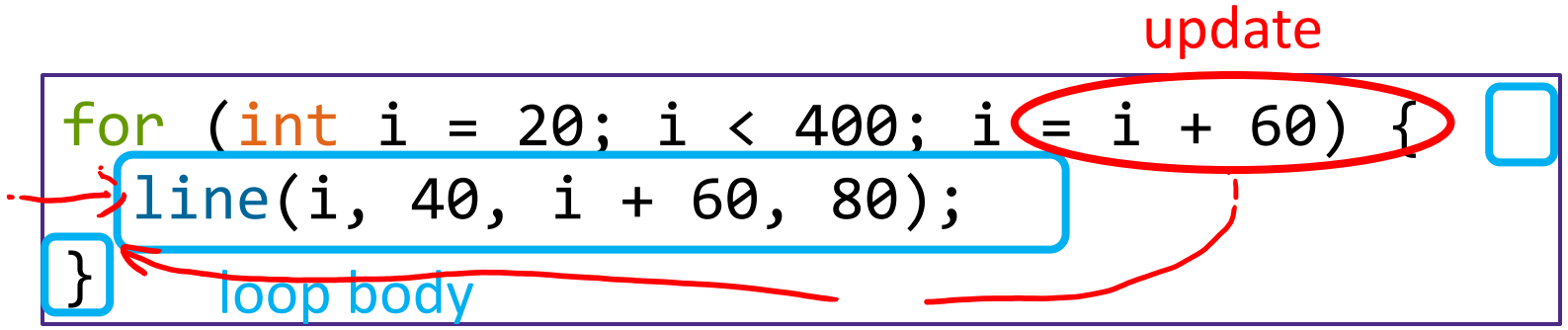
condition

```
for (int i = 20; i < 400; i = i + 60) {
  line(i, 40, i + 60, 80);
}
```

- Condition evaluated *before* the loop body and must evaluate to `true` or `false`
  - Reminder:

>	greater than
<	less than
>=	greater than or equal to
>=	less than or equal to
==	equal to
!=	not equal to

Understanding the For-Loop



- Update is an assignment that is executed *after* the loop body

- Loop body is enclosed by curly braces { } and should be *indented* for readability