Nested Loops & Multi-dimensional Arrays

```
for (...) {

  doSomething();

}
```

As we know, a `for` loop lets us repeat tasks easily in code.

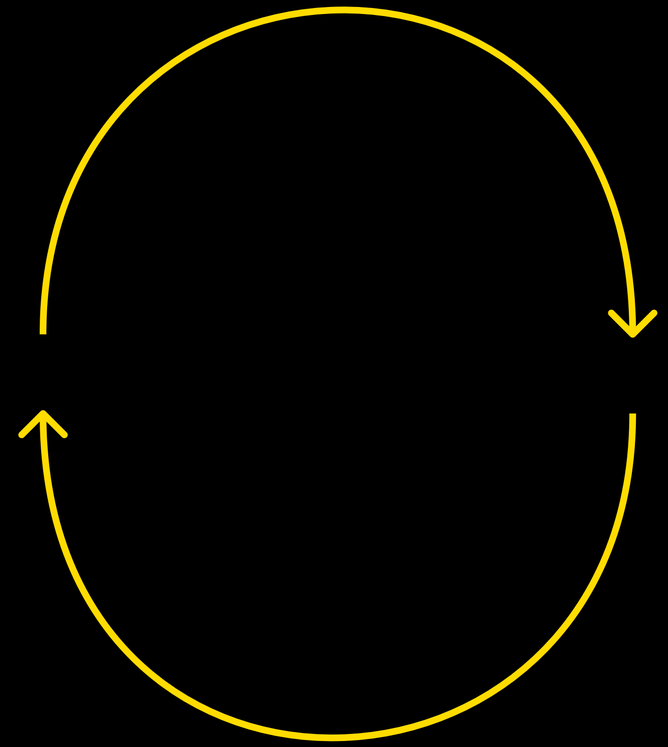
Just like the `draw()` function in p5, loops repeat some block of code, the *body* of the loop, until some condition is met.

```
for (...) {
   for (...) {
      doSomething();
   }
}
```

But we can also put loops inside of other loops - this is called *nesting*. With two loops inside of each other, for example, we're now repeating ... a repeated task.

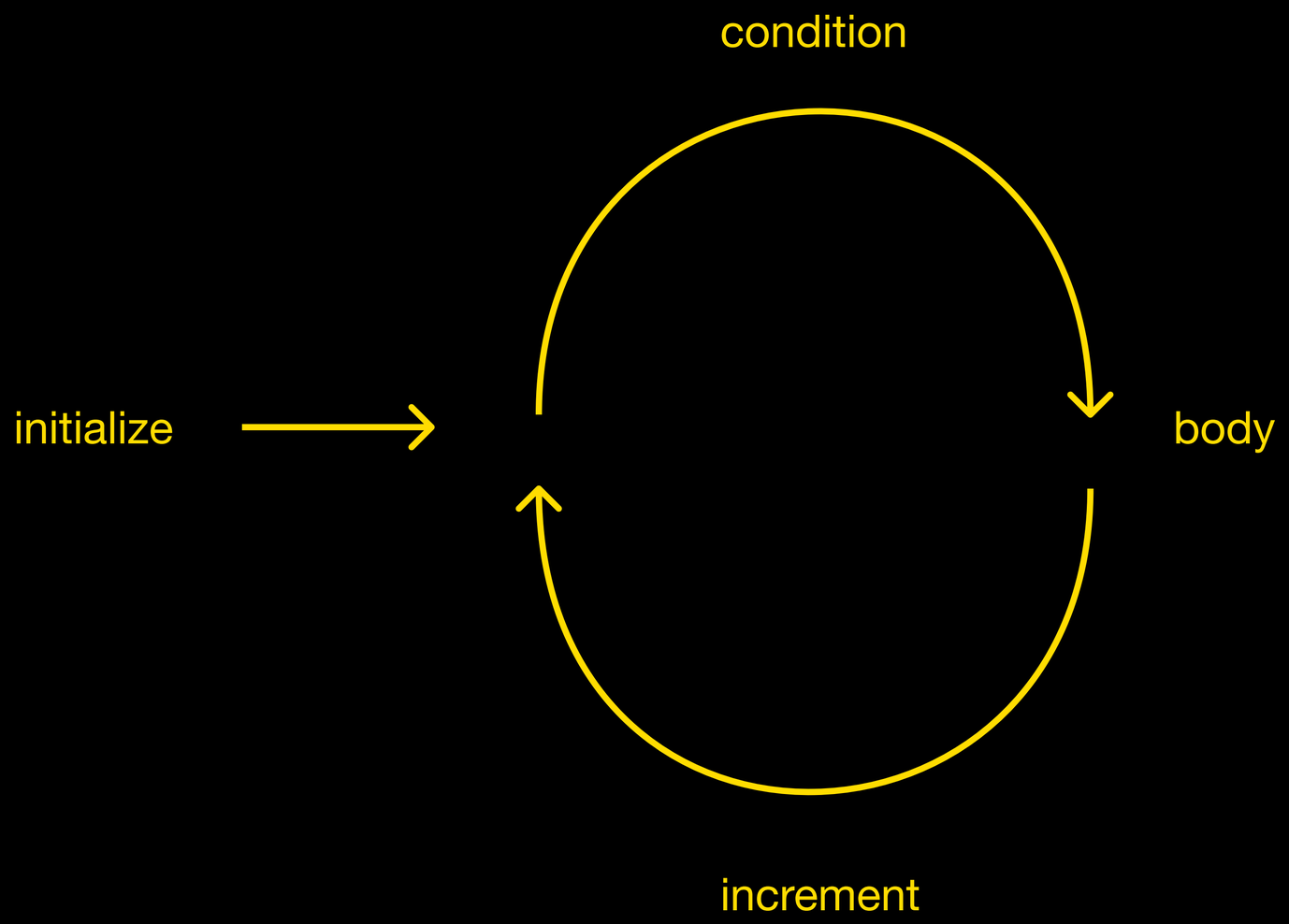
Why is this useful? Imagine you have 1920x1080 pixels on a screen. Would you know which pixel 307200 is? Probably not, but you would know how to find pixel 640x480. **Multiple for loops help us organize our repetitions in the right way for the task at hand.**

```
for (...) {

  doSomething();

}
```

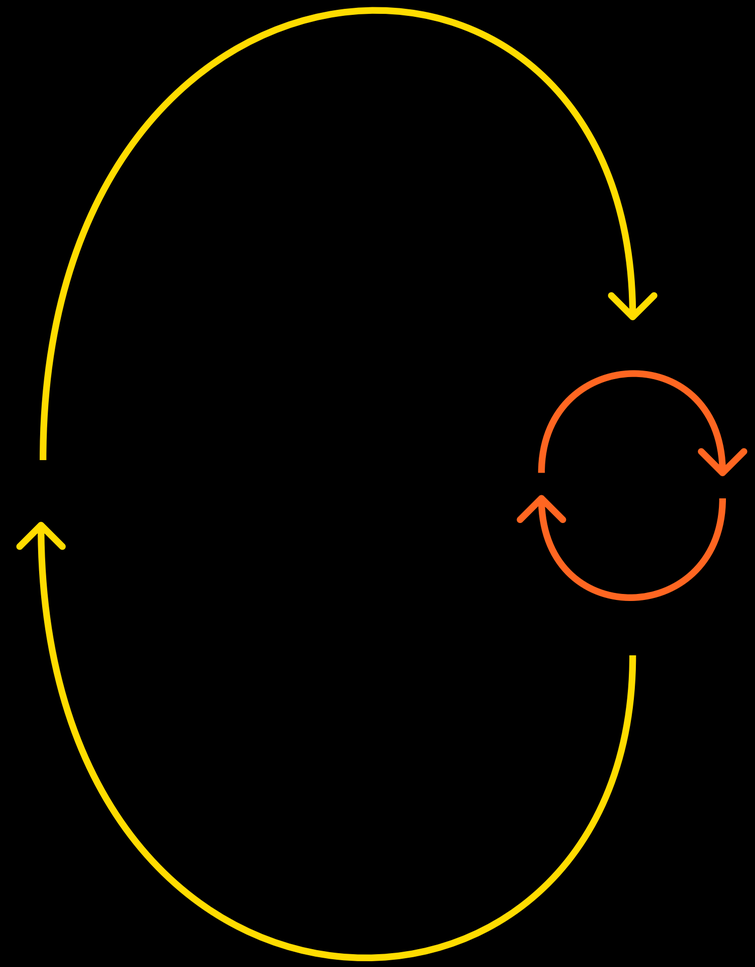


initialize condition increment

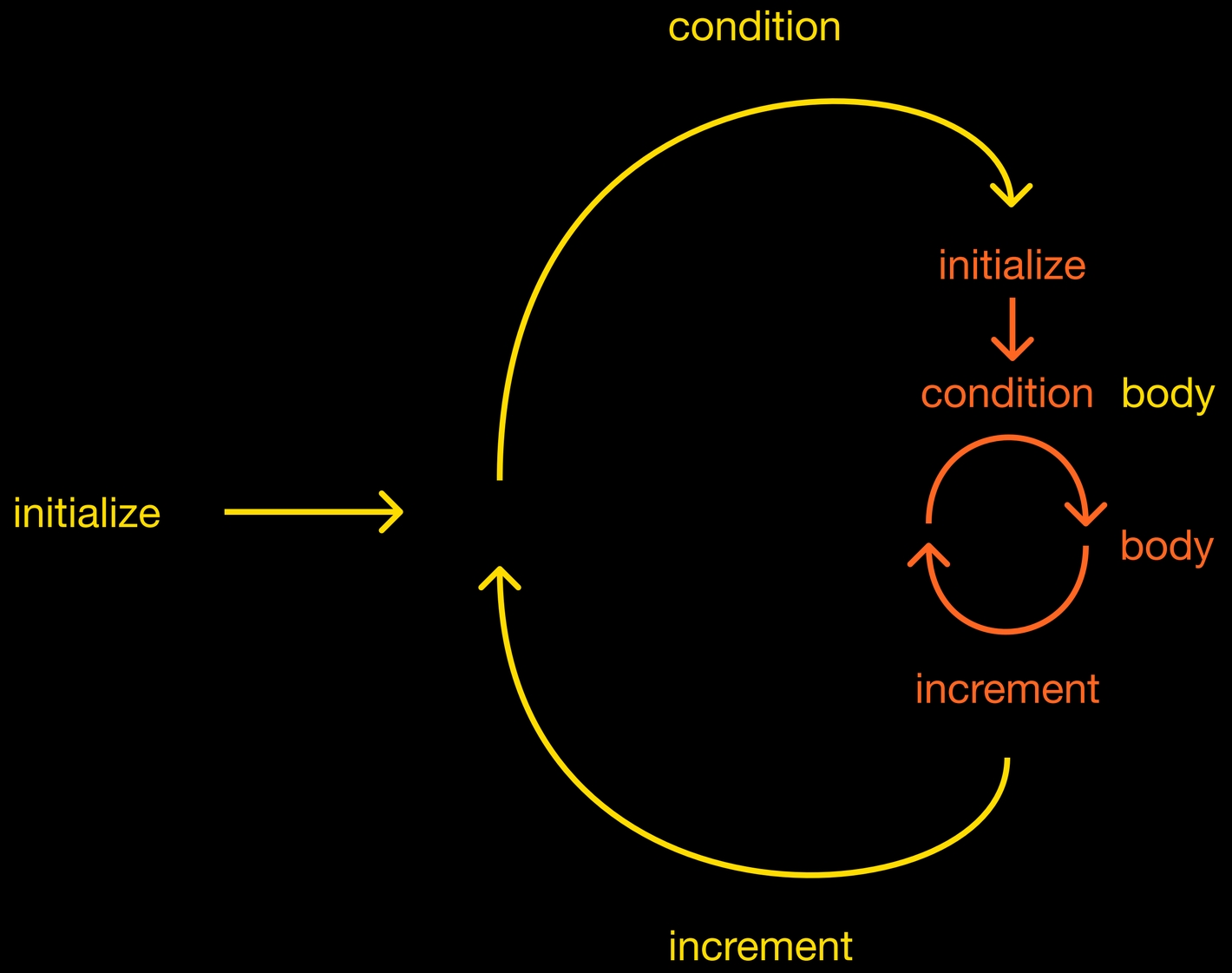
```
for (let i = 0;  i < 5;   i++) {

  doSomething();  body

}
```



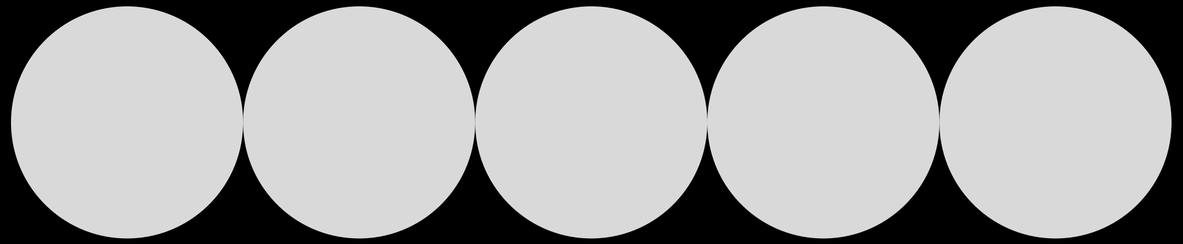
```
for (let i = 0;  i < 5;   i++) {

  for (let j = 0; j < 8;   j++) {

    doSomething();

  }

}
```



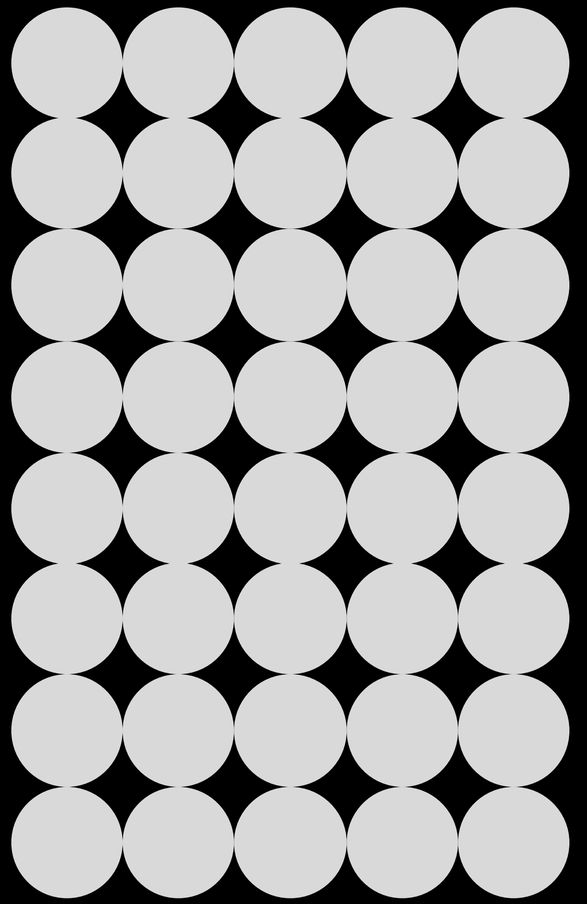
```
        initialize      condition   increment
for (let i = 0;   i < 5;    i++) {
         initialize     condition   increment
  for (let j = 0; j < 8;    j++) { body

    doSomething();   body

  }

}
```



```
for (let i = 0;  i < 5;   i++) {

    circle();

}
```



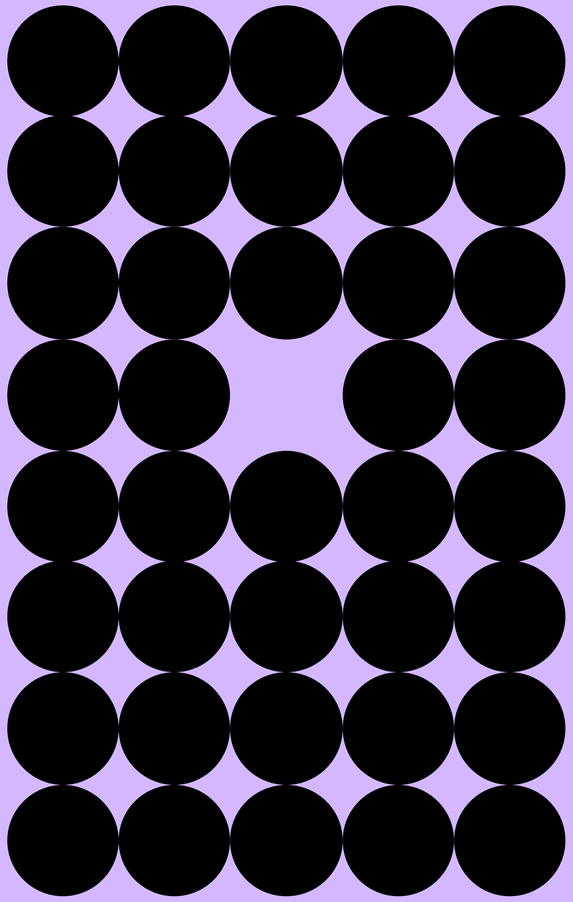
```
for (let i = 0;  i < 5;   i++) {

  for (let j = 0; j < 8;   j++) {

    circle();

  }

}
```



Comprehension Check

- use a nested loop
- use a conditional
- add some variation using your loop counters

optional: make it audio reactive!



```
let colors =
[‘red’, ‘blue’];
```

We’ve seen that arrays help us arrange multiple pieces of data under a single name. But just like nesting loops, we can put arrays inside other arrays as well.

And just like nested loops, this helps when we want to match the shape of our data to the shape of our array.

Think of it like a **list of lists**.

```
let flagColors =
[['red', 'blue', 'white'],
 ['white', 'red'],
 ['green', 'red', 'white'],
 ['yellow', 'red']];
```

```
let flagColors =
[['red', 'blue', 'white'],
 ['white', 'red'],
 ['green', 'red', 'white'],
 ['yellow', 'red']];
```

```
let coordinates =
[25, 68];
```

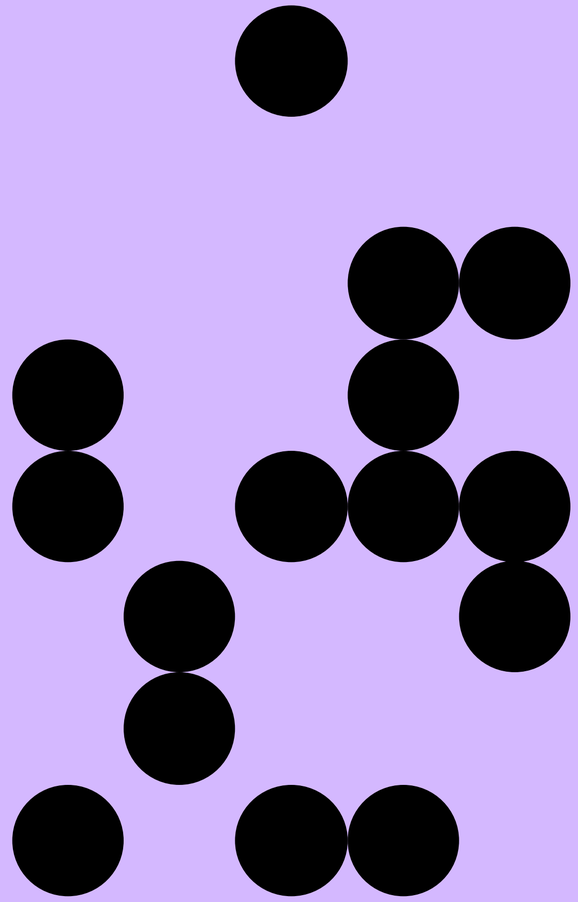
```
let coordinates =
[[25, 68],
 [14, 15],
 [10, 99]];
```

```
let people =
[["Kevin", 129, "🍱"],
 ["Giulio", 176, "🍜"],
 ["Irti", 99, "🌯"]];
```

Comprehension Check

draw some geometric primitives using a multi-dimensional array of coordinates

- now add a for loop. can you do all the drawing in one for loop?



Objects & JSON

```
let people =
[["Kevin", 129, "🍱"],
 ["Giulio", 176, "🍜"],
 ["Irti", 99, "🌯"]];
```

Sometimes, our data is structured in a way that it makes sense to give a name, instead of an index, to parts of it, just like how using a variable as a name for a piece of data helps us remember it.

Objects are a kind of data that can have named fields, called *properties*, inside them. Unlike an array, we can use these names to access data, instead of indices.

```
let people =
[{name: “Kevin”, id: 129, lunch: “🍱”},
 {name: “Giulio”, id: 176, lunch: “🍜”},
 {name: “Irti”, id: 99, lunch: “🌯”}];
```

`people[0][2]` becomes
`people[0].lunch`