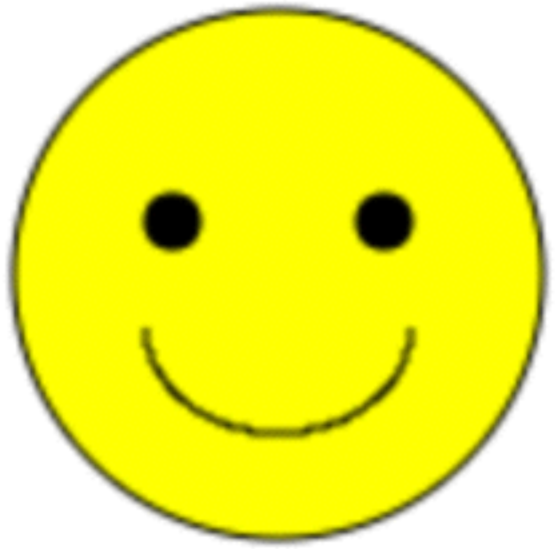
Variables, values, expressions

warm-up: smile

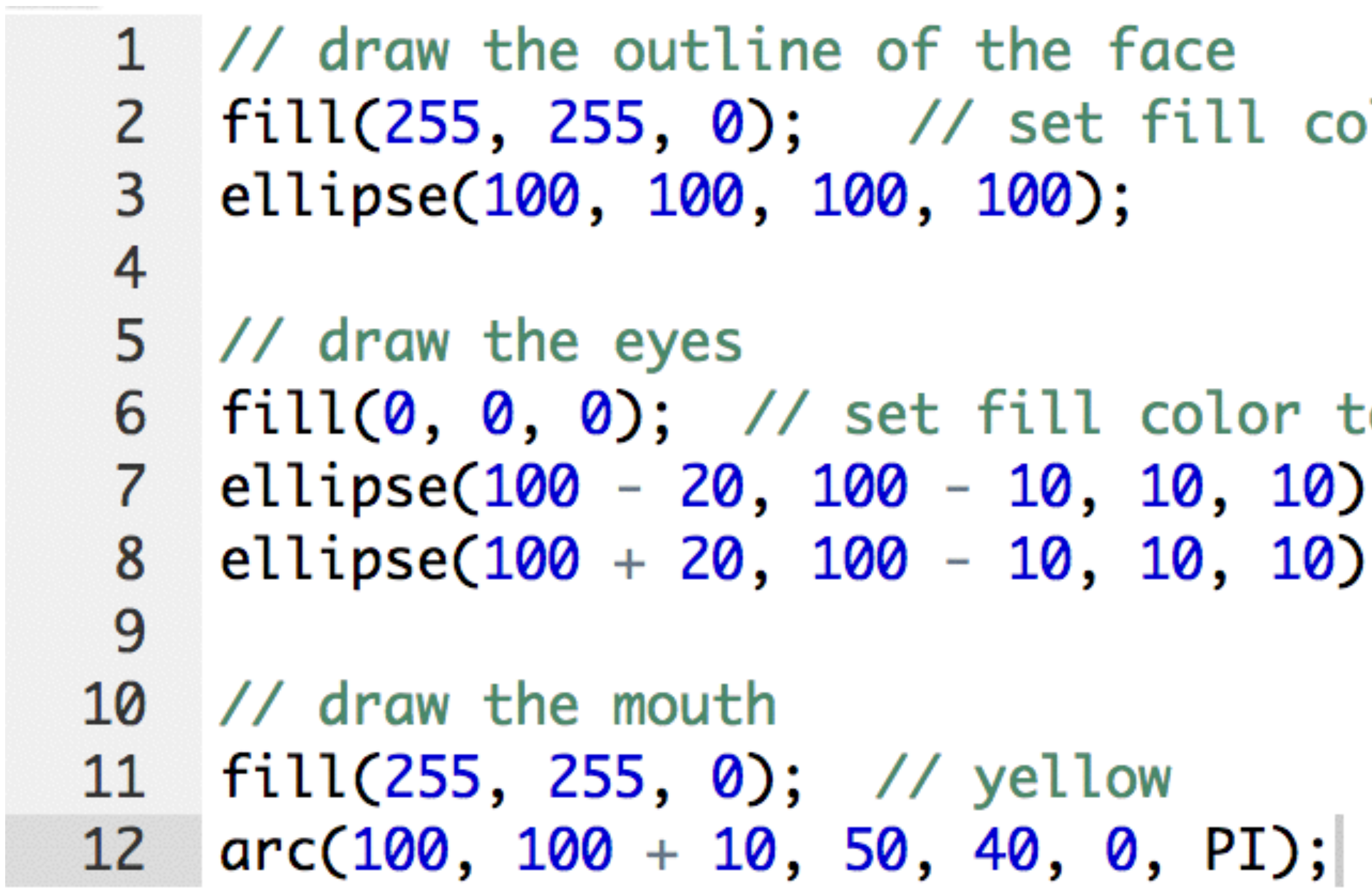


Grab a partner and do exercise: smile (lecture 1 notes)

http://processingjs.org/reference/

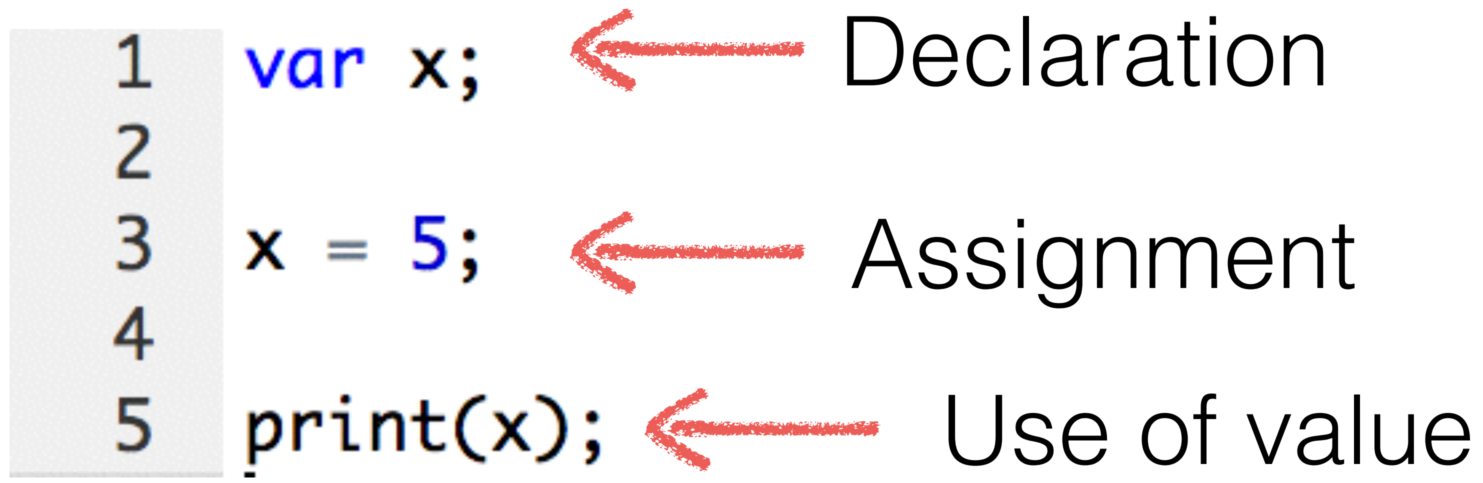
ellipse(), rect(), fill(), stroke()

style: comments



```
// draw the outline of the face
fill(255, 255, 0);   // set fill color to yellow
ellipse(100, 100, 100, 100);

// draw the eyes
fill(0, 0, 0);  // set fill color to black
ellipse(100 - 20, 100 - 10, 10, 10);
ellipse(100 + 20, 100 - 10, 10, 10);

// draw the mouth
fill(255, 255, 0);  // yellow
arc(100, 100 + 10, 50, 40, 0, PI);
```

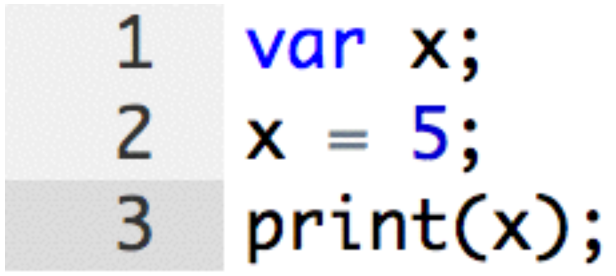
variables



```
var x;

x = 5;

print(x);
```

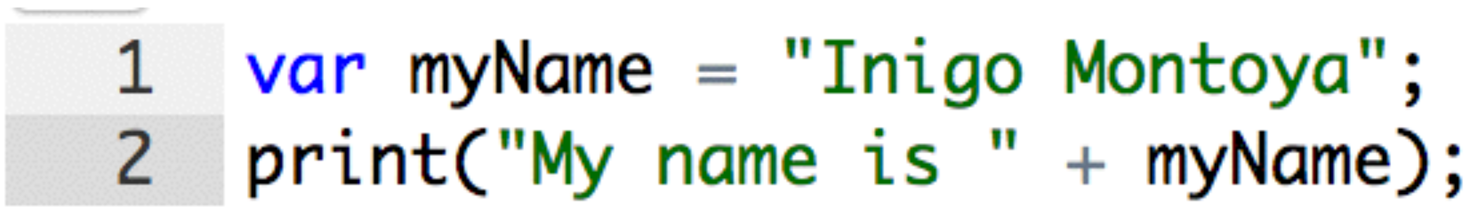
1. **Declare** the variable with **var**: reserve space in memory to store a value, and give it a name
2. **Assign a value** to the variable with **=,** copying a pattern of 0s and 1s to that location
3. **Use** the variable anywhere a value is needed.

values have types



```
var x;
x = 5;
print(x);
```

5 is a **number**.
x holds a **number**

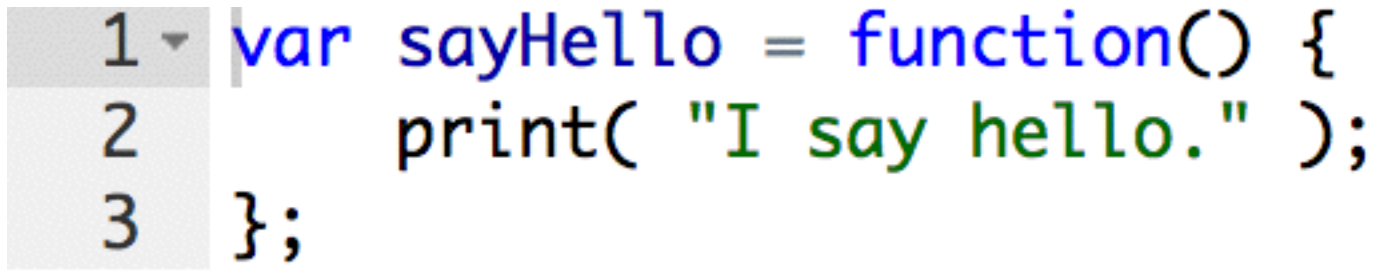


```
var myName = "Inigo Montoya";
print("My name is " + myName);
```

“Inigo Montoya” is a **string**.
myName holds a **string.**

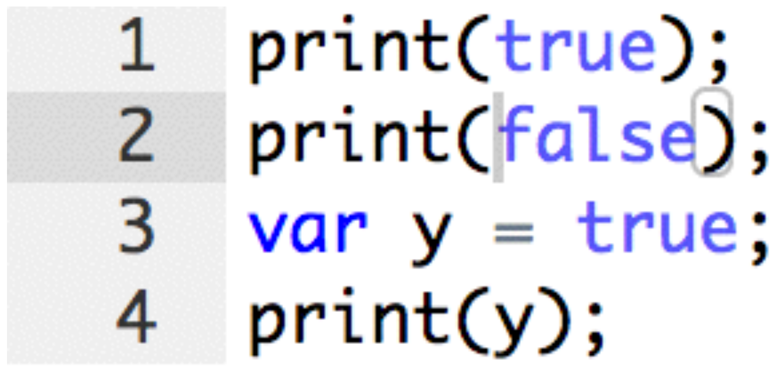
Declaration and **assignment** can be combined.

values have types



```
var sayHello = function() {
    print( "I say hello." );
};
```

sayHello is a **function**



```
print(true);
print(false);
var y = true;
print(y);
```

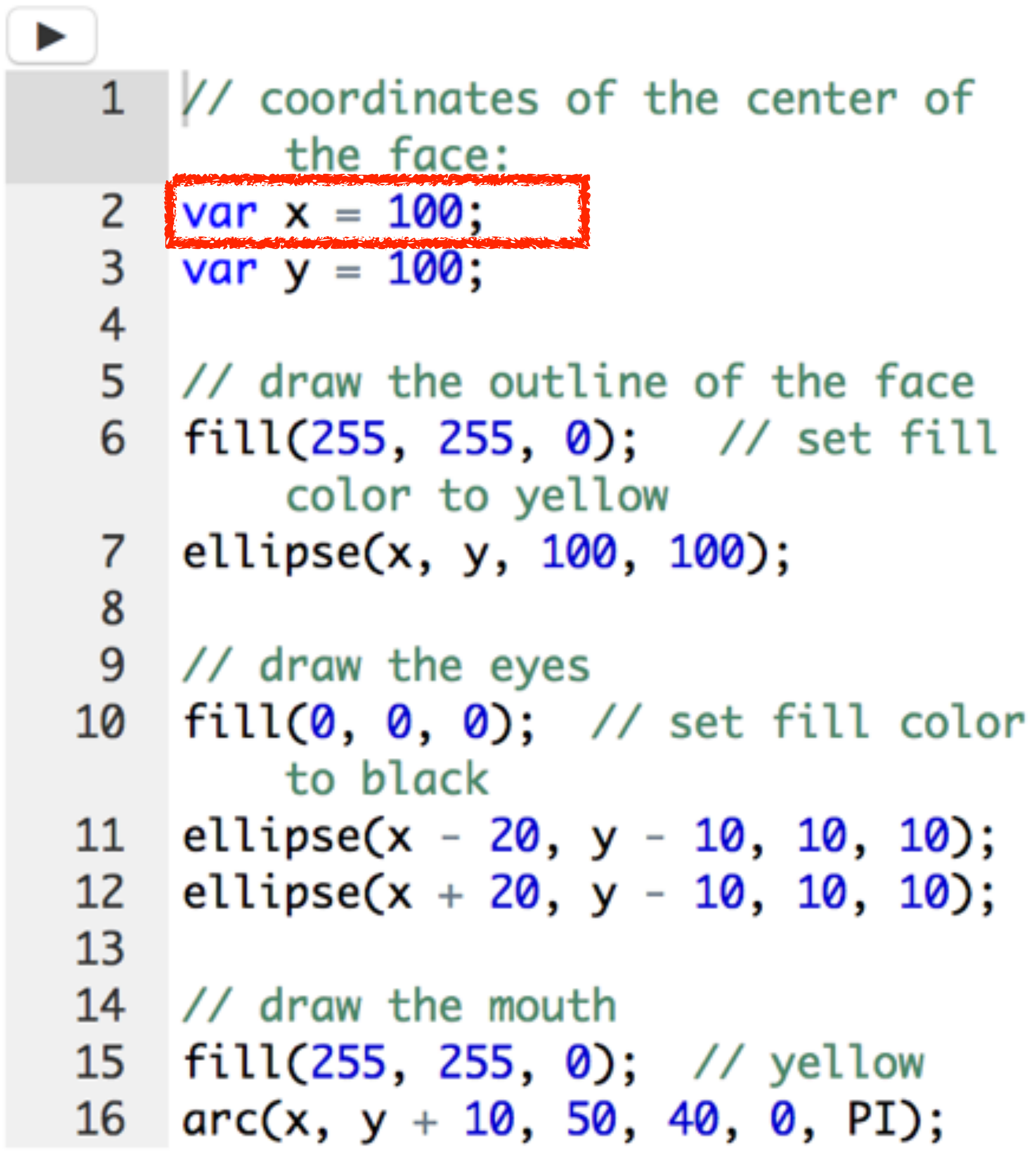
true, **false**, and **y** are **boolean values**

values have types

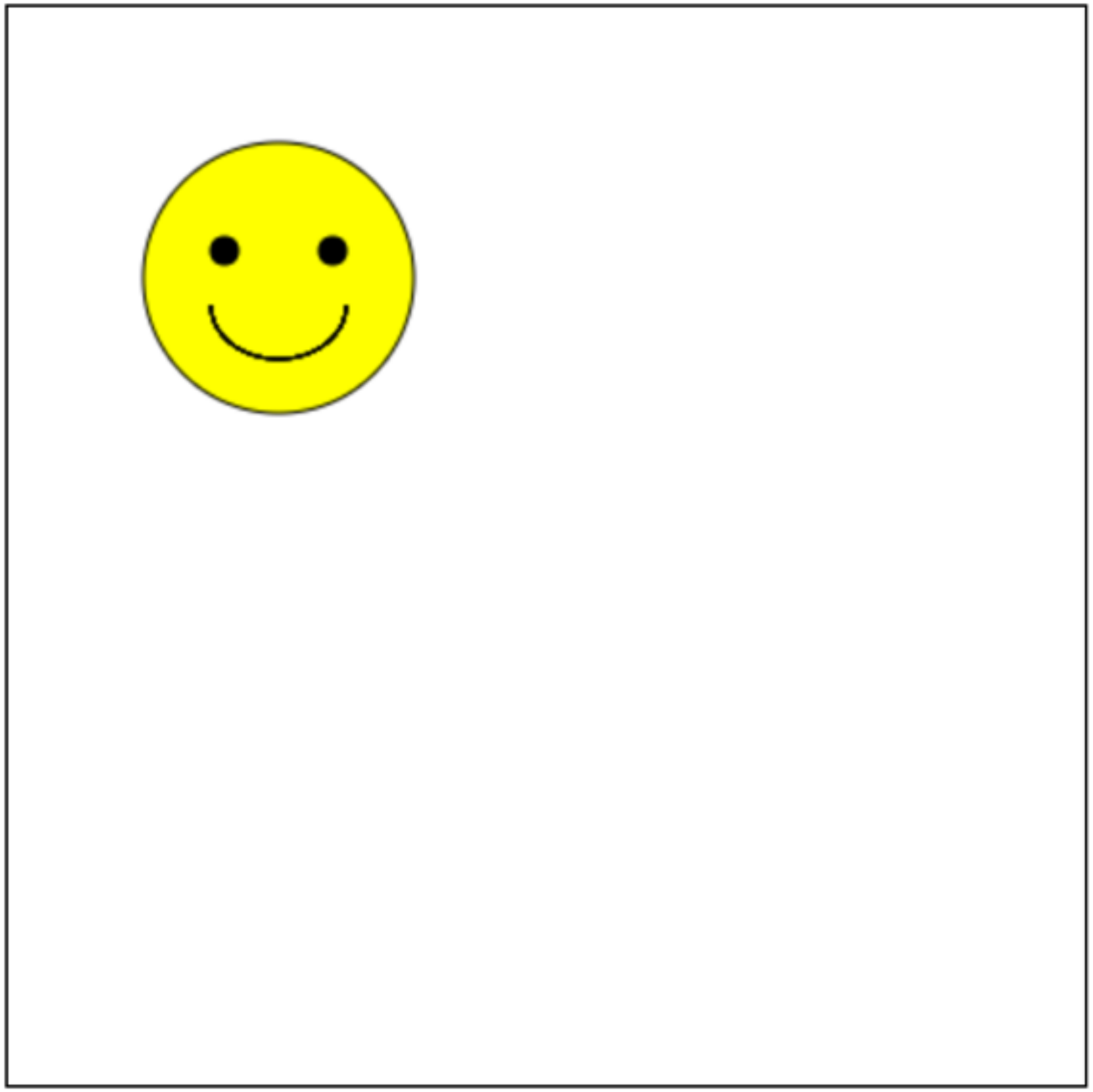
1. numbers (3.141592654, 7)
2. strings (“frog”)
3. booleans (true, false)
4. functions (function() {…})

Always know what **type** of value every one of your variables holds.

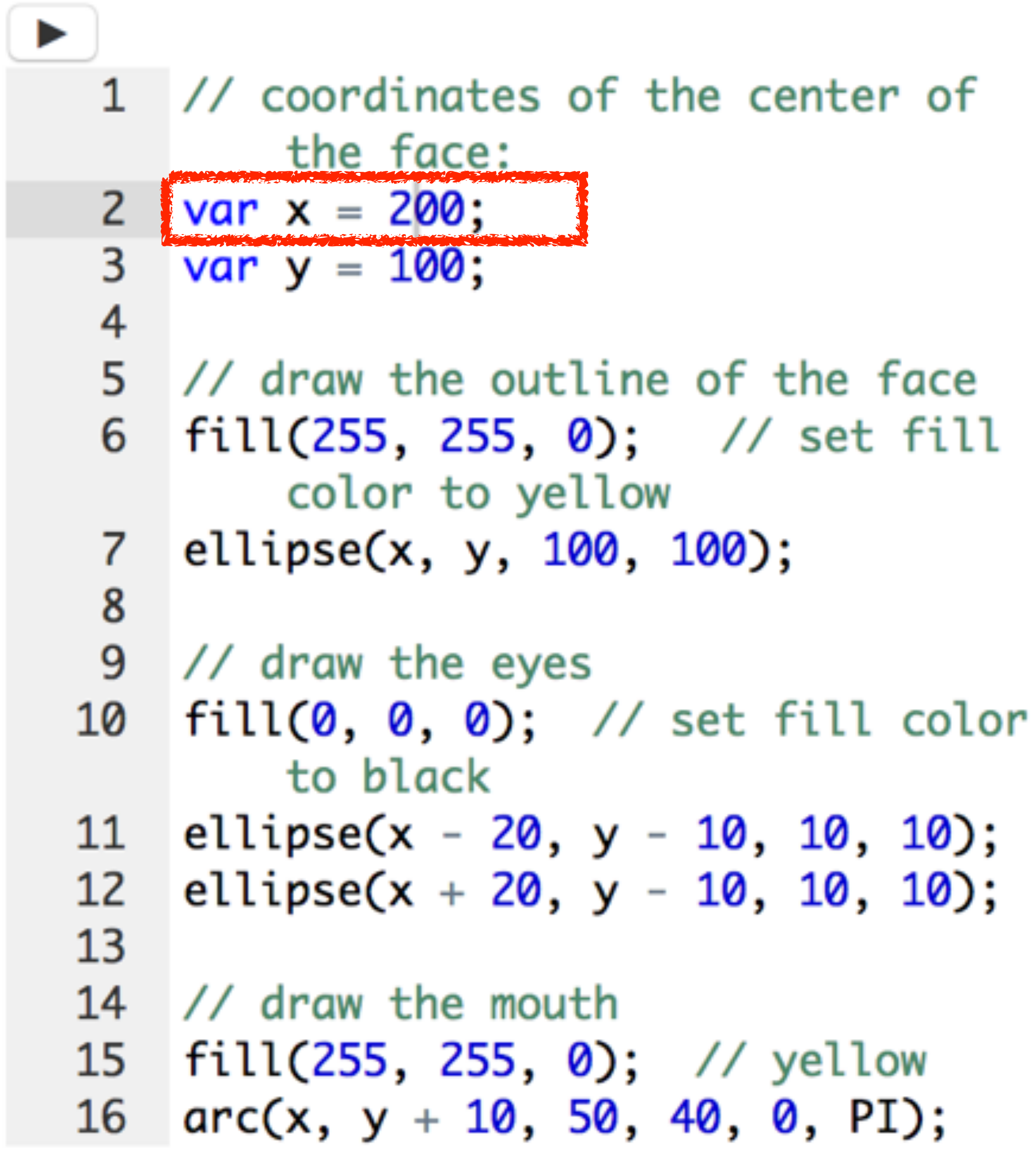
variables change behaviors



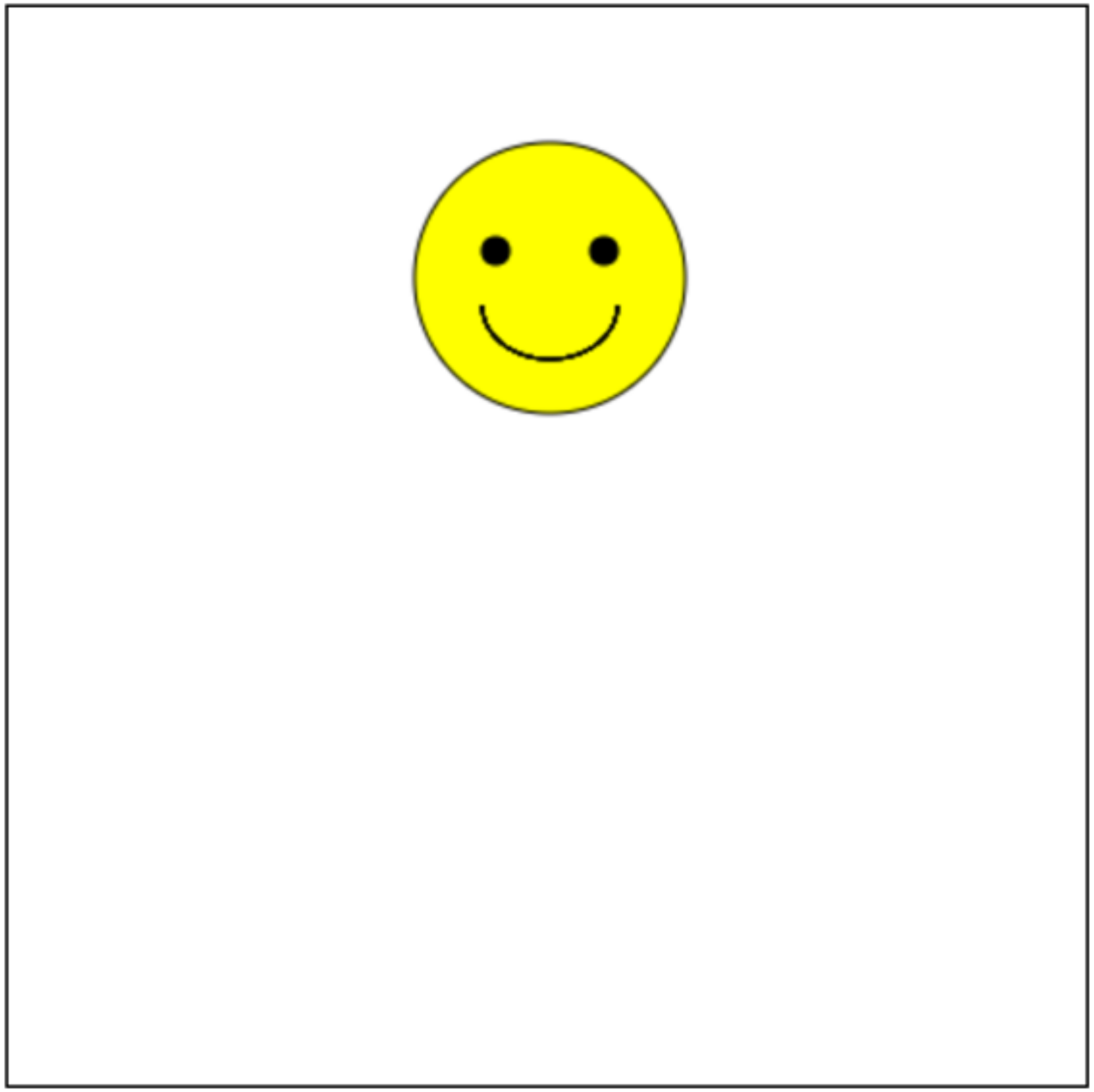
```
// coordinates of the center of the face:
var x = 100;
var y = 100;

// draw the outline of the face
fill(255, 255, 0);   // set fill color to yellow
ellipse(x, y, 100, 100);

// draw the eyes
fill(0, 0, 0);  // set fill color to black
ellipse(x - 20, y - 10, 10, 10);
ellipse(x + 20, y - 10, 10, 10);

// draw the mouth
fill(255, 255, 0);  // yellow
arc(x, y + 10, 50, 40, 0, PI);
```



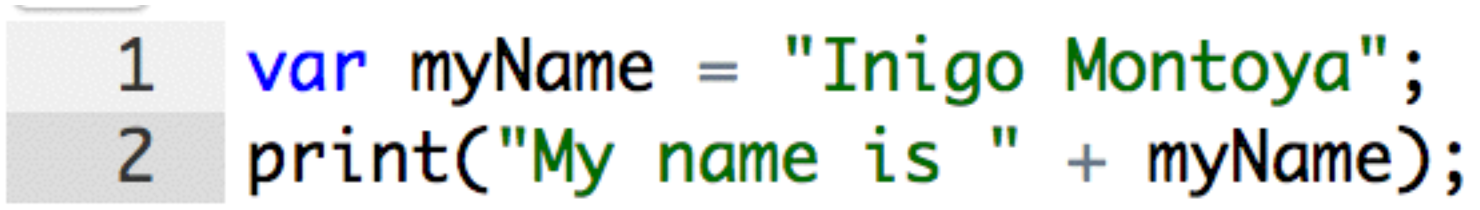
variables change behaviors



```
// coordinates of the center of
    the face:
var x = 200;
var y = 100;

// draw the outline of the face
fill(255, 255, 0);    // set fill
    color to yellow
ellipse(x, y, 100, 100);

// draw the eyes
fill(0, 0, 0);   // set fill color
    to black
ellipse(x - 20, y - 10, 10, 10);
ellipse(x + 20, y - 10, 10, 10);

// draw the mouth
fill(255, 255, 0);   // yellow
arc(x, y + 10, 50, 40, 0, PI);
```

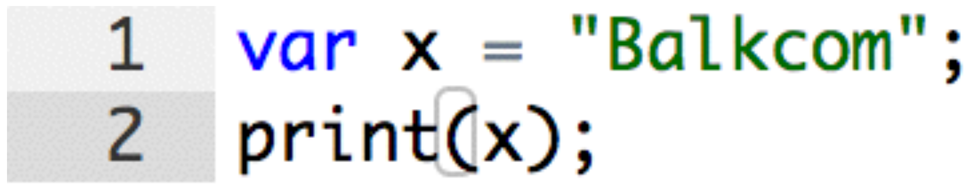


style: good variable names



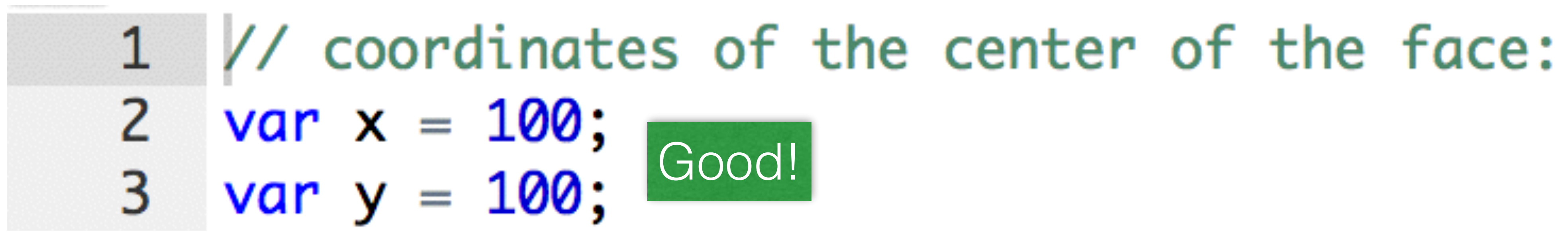
```
var myName = "Inigo Montoya";
print("My name is " + myName);
```

Good!



```
var x = "Balkcom";
print(x);
```

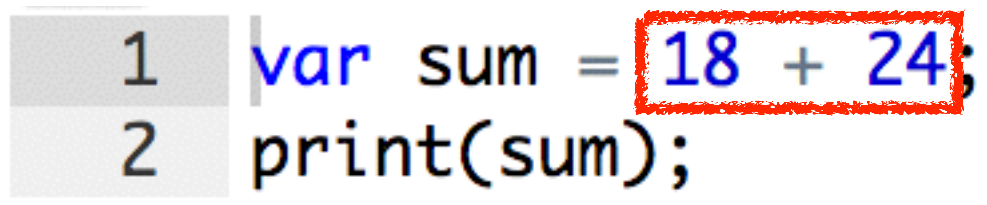
Bad!



```
// coordinates of the center of the face:
var x = 100;
var y = 100;
```

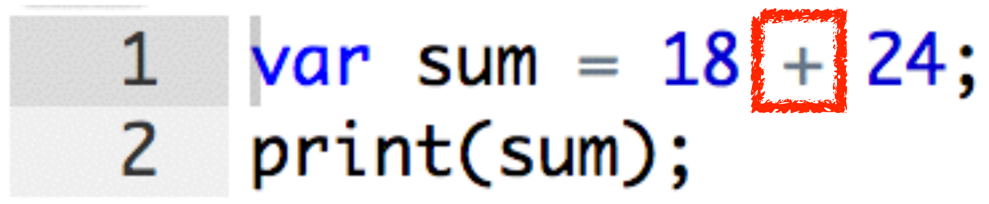
Good!

Expressions and operators



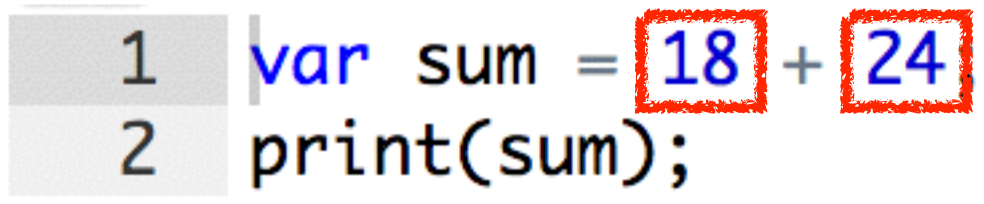
```
var sum = 18 + 24;
print(sum);
```

Expression



```
var sum = 18 + 24;
print(sum);
```

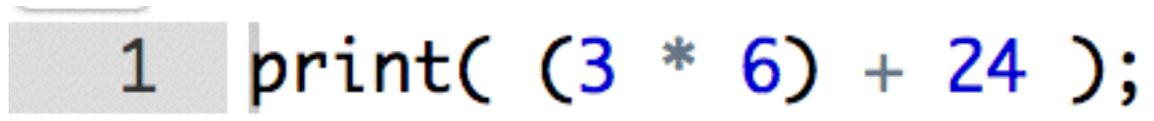
Operator



```
var sum = 18 + 24;
print(sum);
```

Operands

You can use anywhere a value is required:



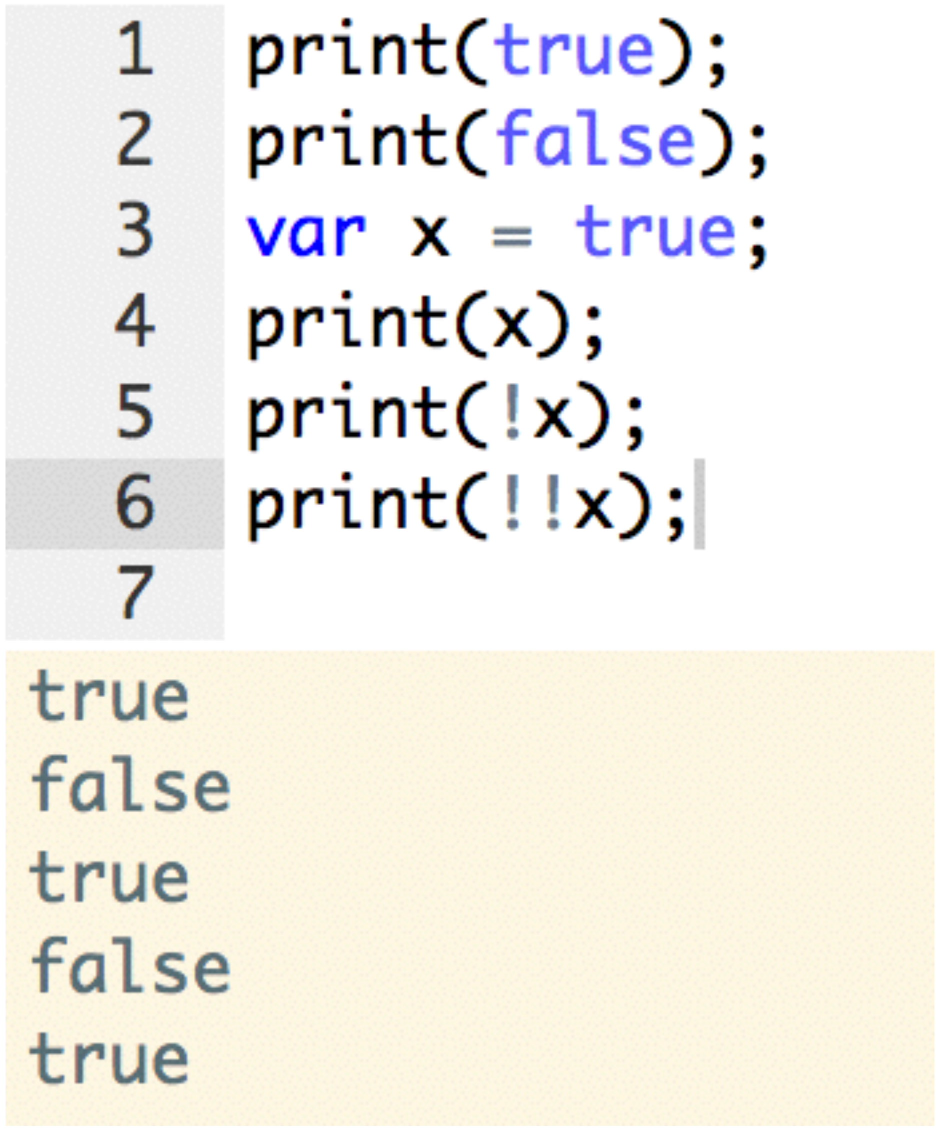
```
print( (3 * 6) + 24 );
```

Number operators

+	addition	5 + 4	9
-	subtraction	5 - 19	-14
*	multiplication	3 * 6.2	18.6
/	division	19 / 5	3.8
%	modulus (remainder)	19 % 5	4

Style note: **always** surround operators with spaces.

Boolean operator: !



```
print(true);
print(false);
var x = true;
print(x);
print(!x);
print(!!x);
```

```
true
false
true
false
true
```

negation operator: takes one boolean operand, and flips it.

The assignment operator

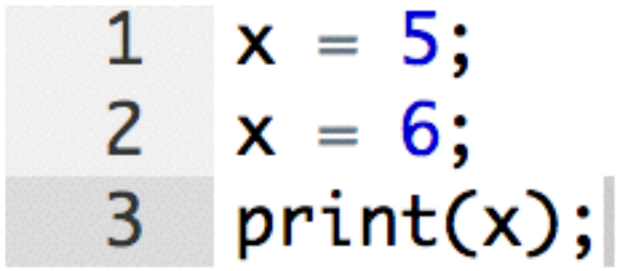
Mathematics, equality sign: equation

$x = 5$ means “x is 5, forever”

$$\begin{array}{r} x = 5 \\ -\ x = 6 \\ \hline 0 = 1 \end{array}$$

Bogus!

Code, = is the **assignment operator**

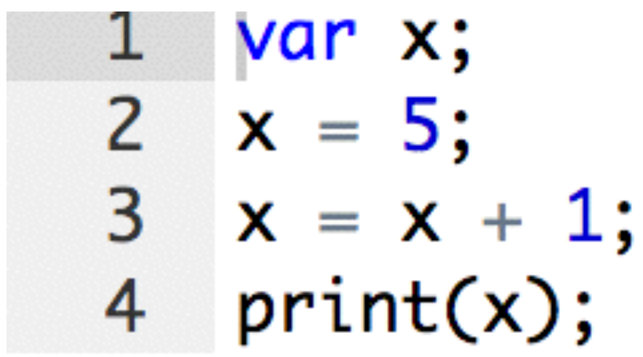


```
x = 5;
x = 6;
print(x);
```

Copy value 5 into x

Copy value 6 into x

The assignment operator



```
1  var x;
2  x = 5;
3  x = x + 1;
4  print(x);
```

Does line 3 mean
“x is the number 1 greater than itself”?

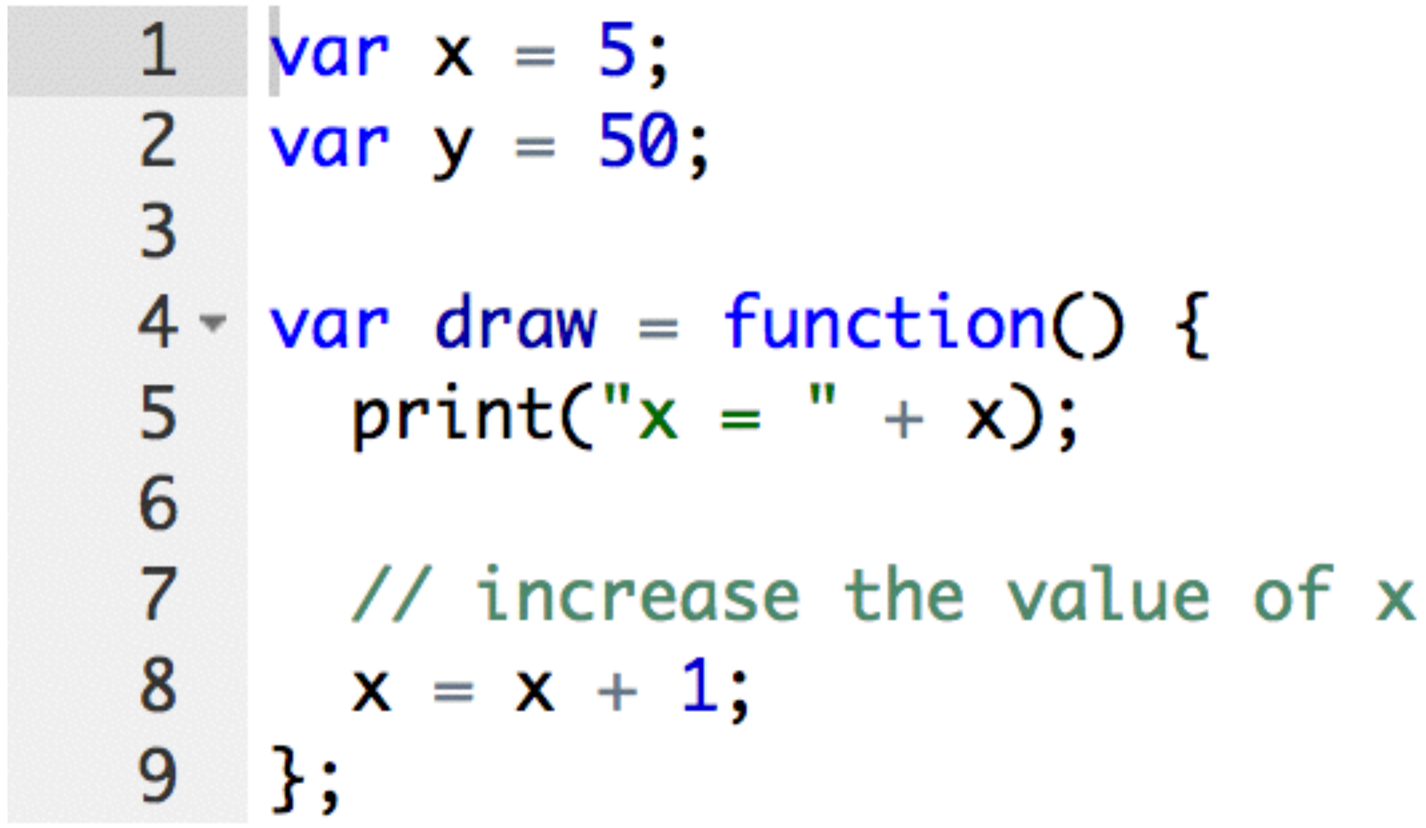
Animation

Animation: algorithm

1. Create variables with initial values (state)
2. Clear the screen
3. Draw picture using variables
4. Change variable values
5. Go back to step two.

You know how to do steps 1… 4

Animation: draw function

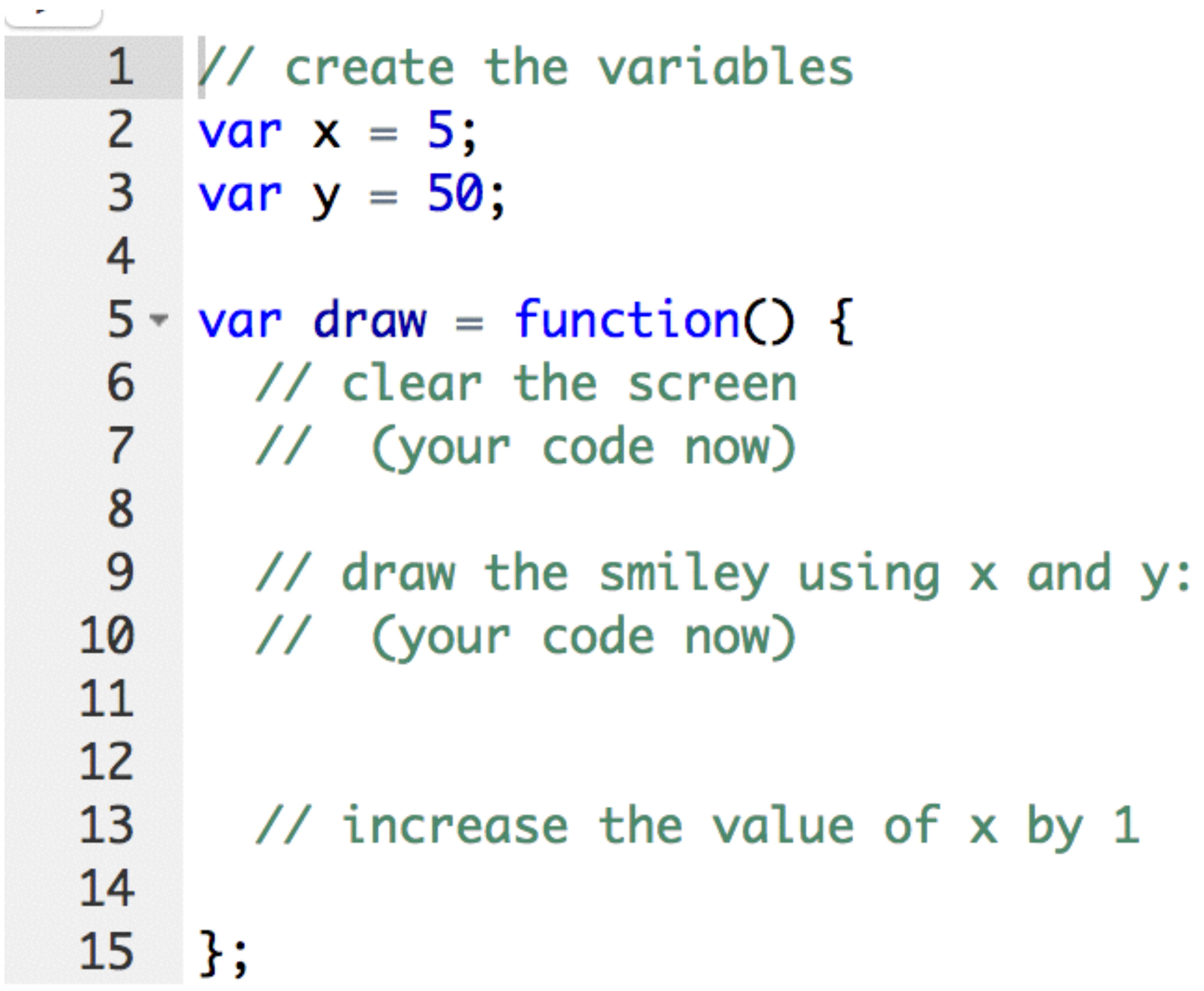


```
var x = 5;
var y = 50;

var draw = function() {
  print("x = " + x);

  // increase the value of x
  x = x + 1;
};
```

This code should have no effect.

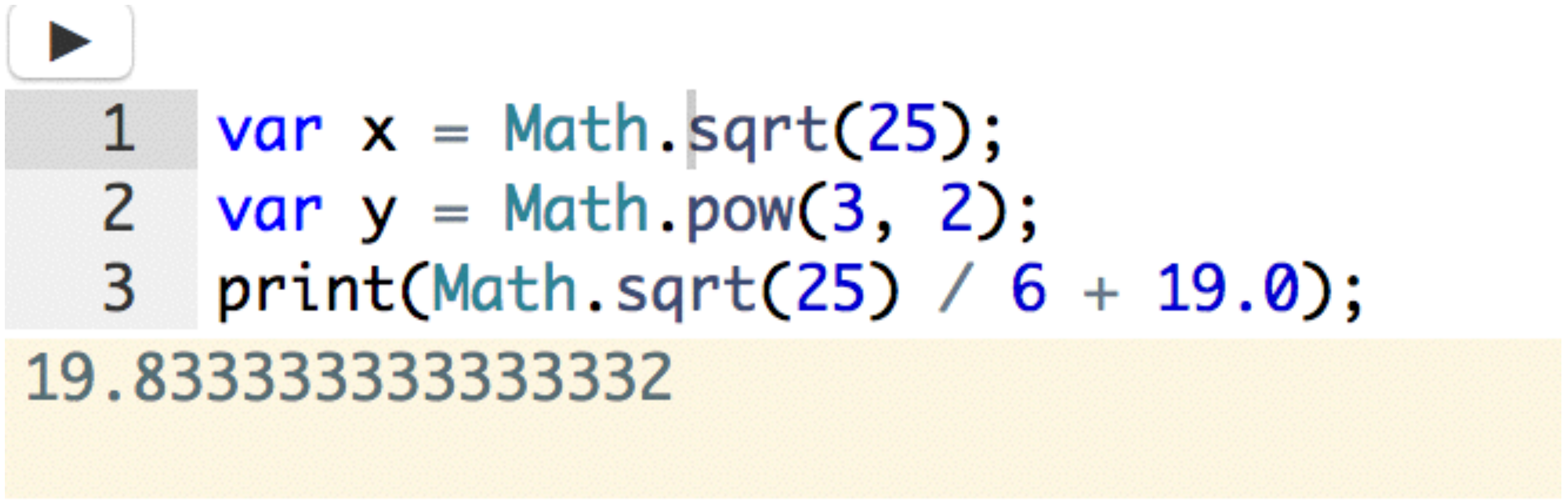
draw is a special function. The browser calls draw repeatedly. (Use it only for animation.)
Go run it in lecture 2 notes.

Exercise: moving smile



```
// create the variables
var x = 5;
var y = 50;

var draw = function() {
  // clear the screen
  //  (your code now)

  // draw the smiley using x and y:
  //  (your code now)


  // increase the value of x by 1

};
```

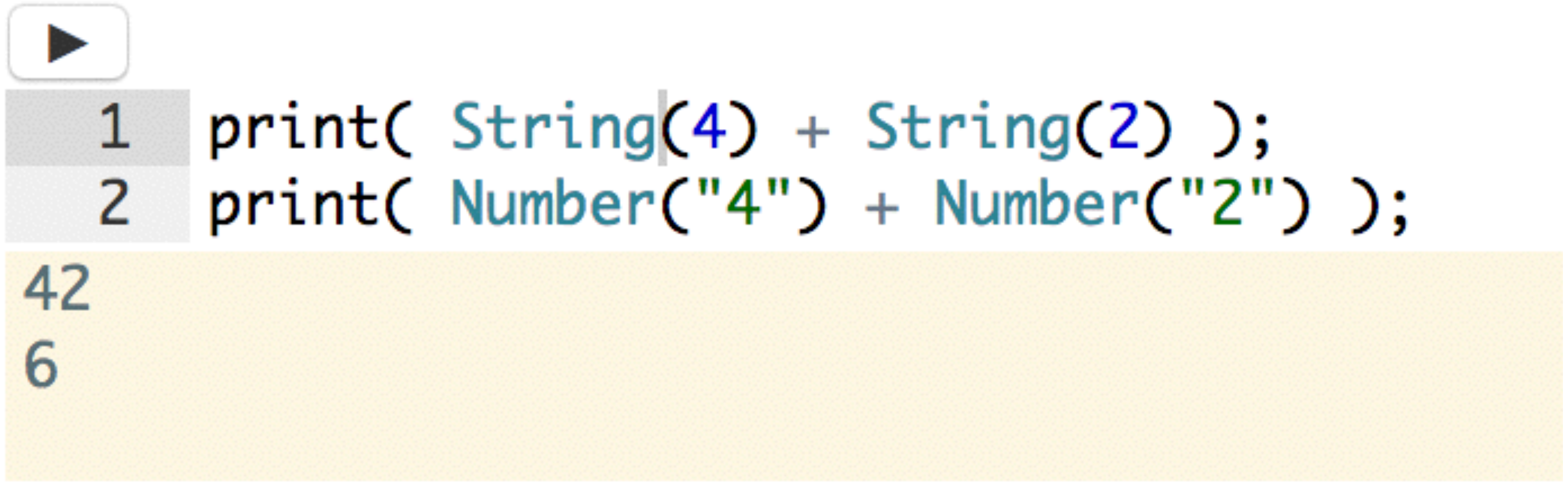
Function calls are expressions



```
var x = Math.sqrt(25);
var y = Math.pow(3, 2);
print(Math.sqrt(25) / 6 + 19.0);
```

```
19.833333333333332
```

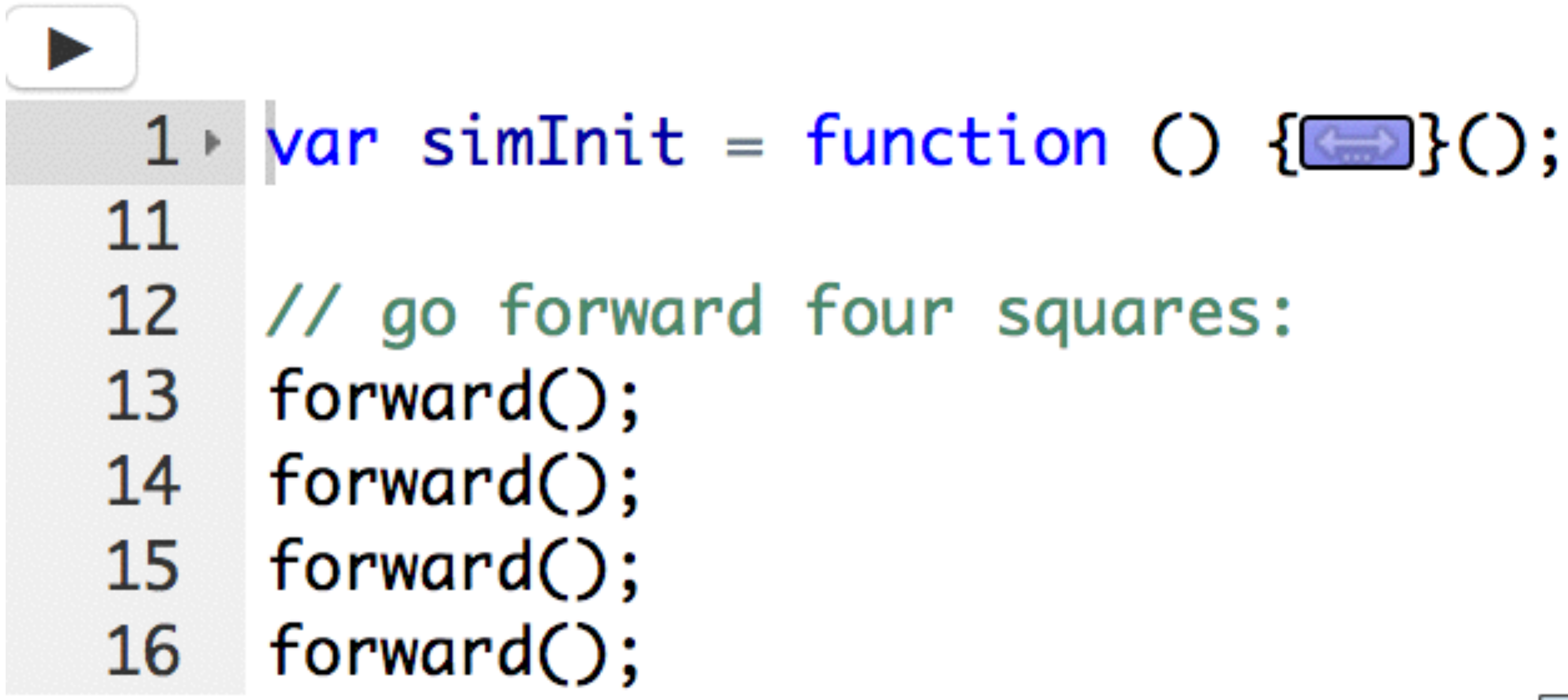
Converting between types



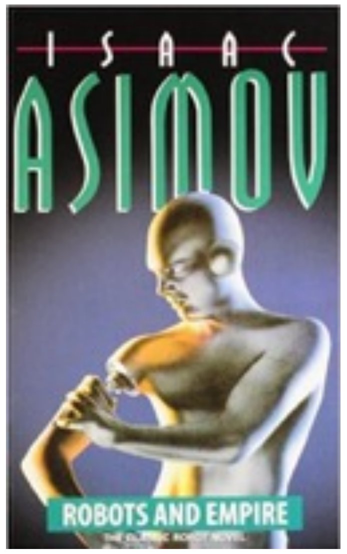
```
print( String(4) + String(2) );
print( Number("4") + Number("2") );
```

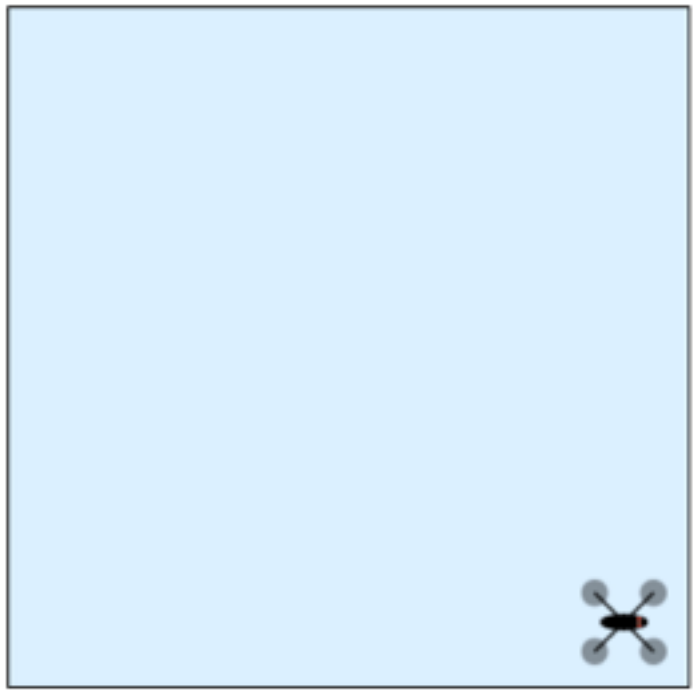
```
42
6
```

Robots and code



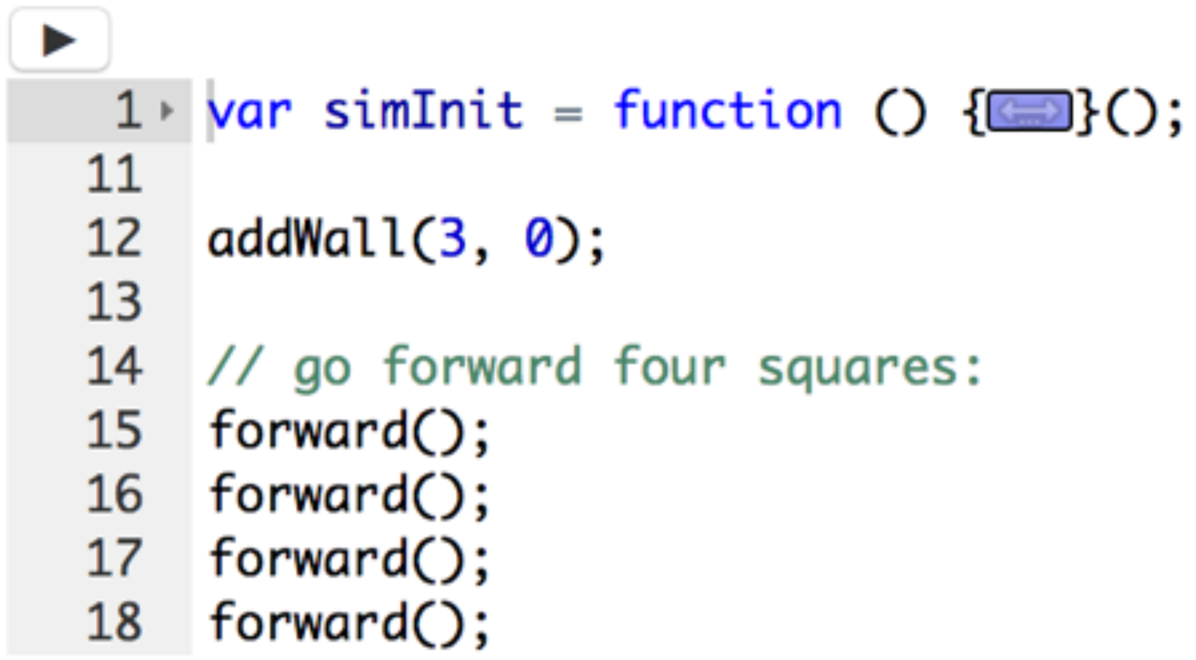
```
var simInit = function () {}();

// go forward four squares:
forward();
forward();
forward();
forward();
```



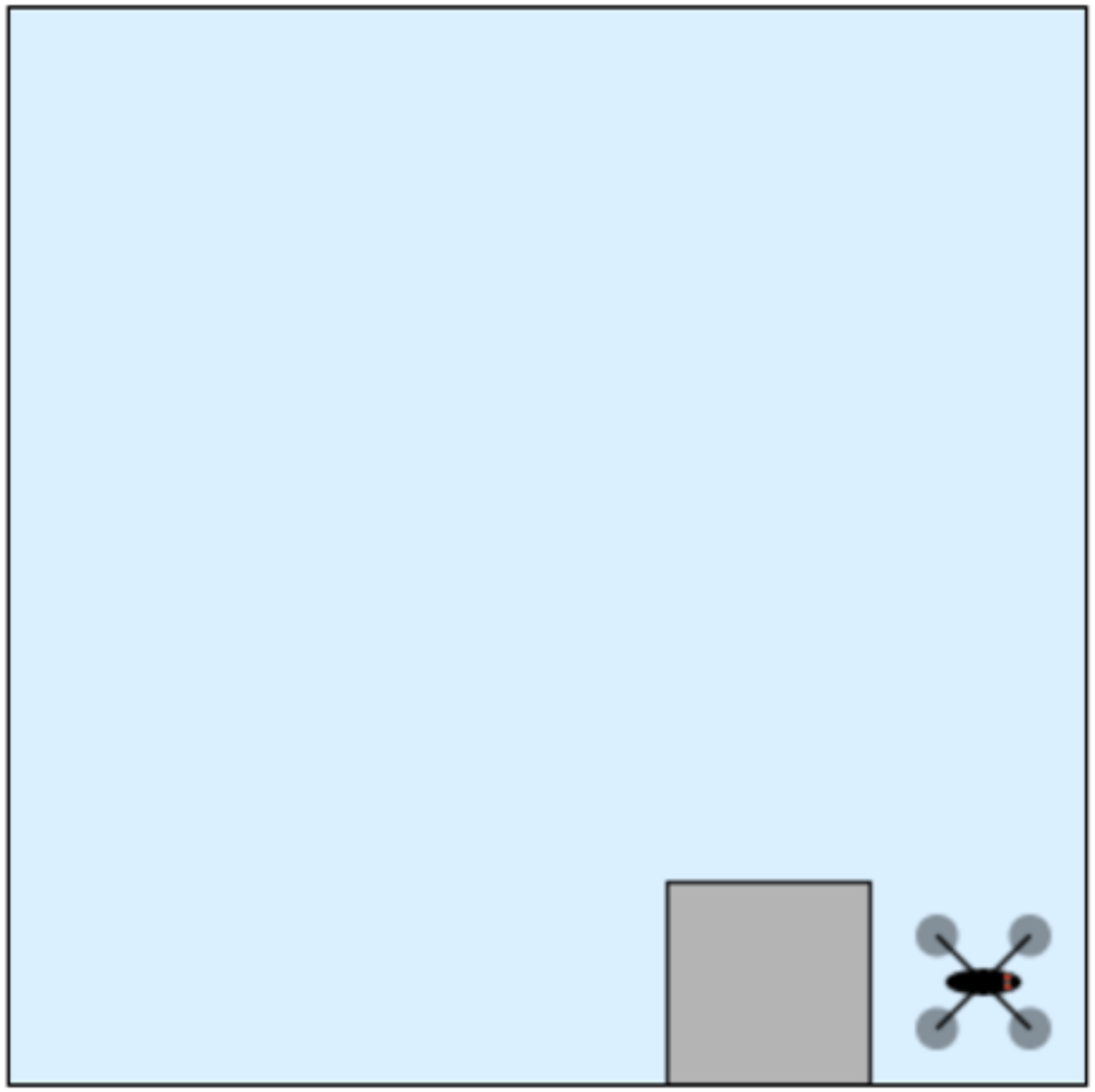


Run it now!

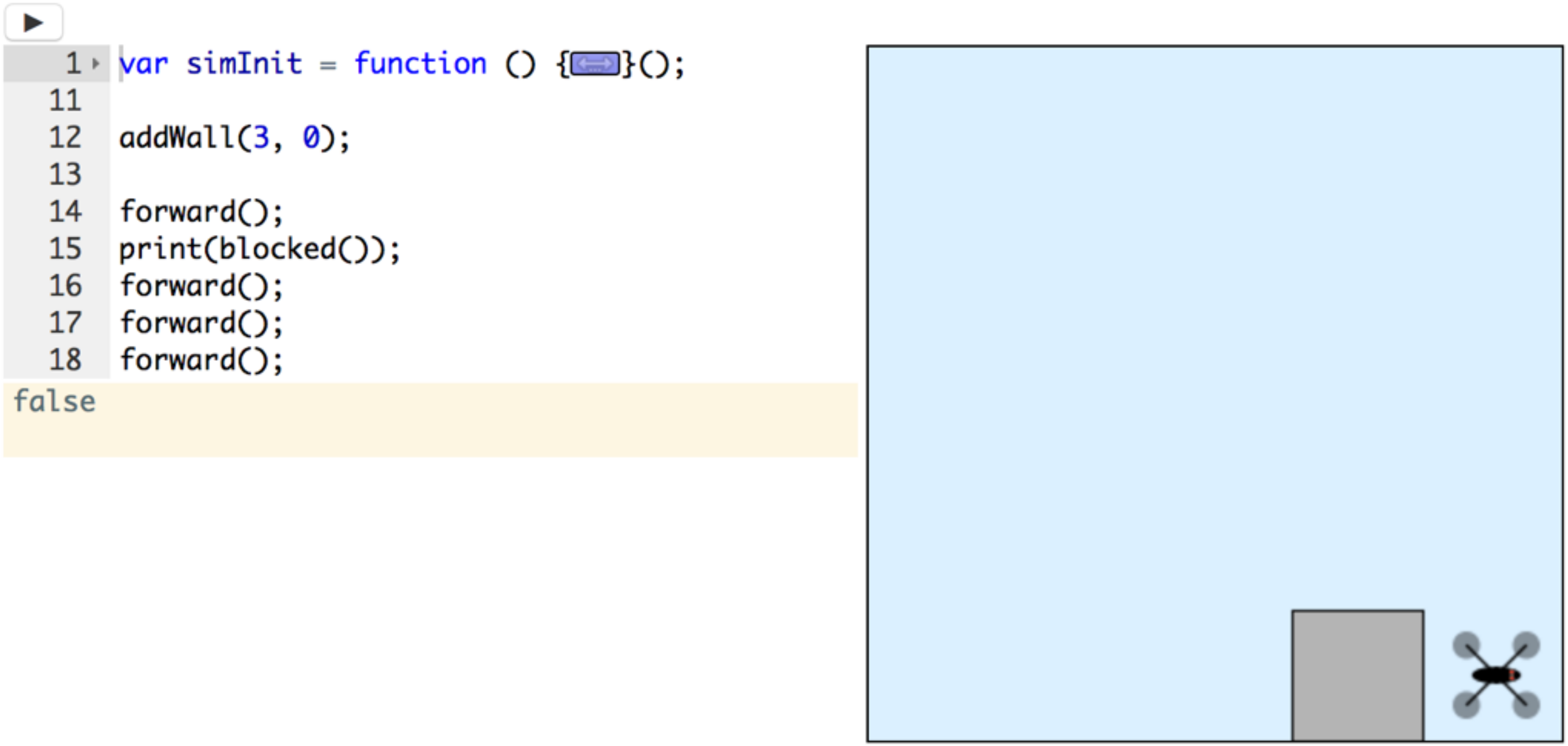
Robots and wall



```
var simInit = function () {⟺}();

addWall(3, 0);

// go forward four squares:
forward();
forward();
forward();
forward();
```



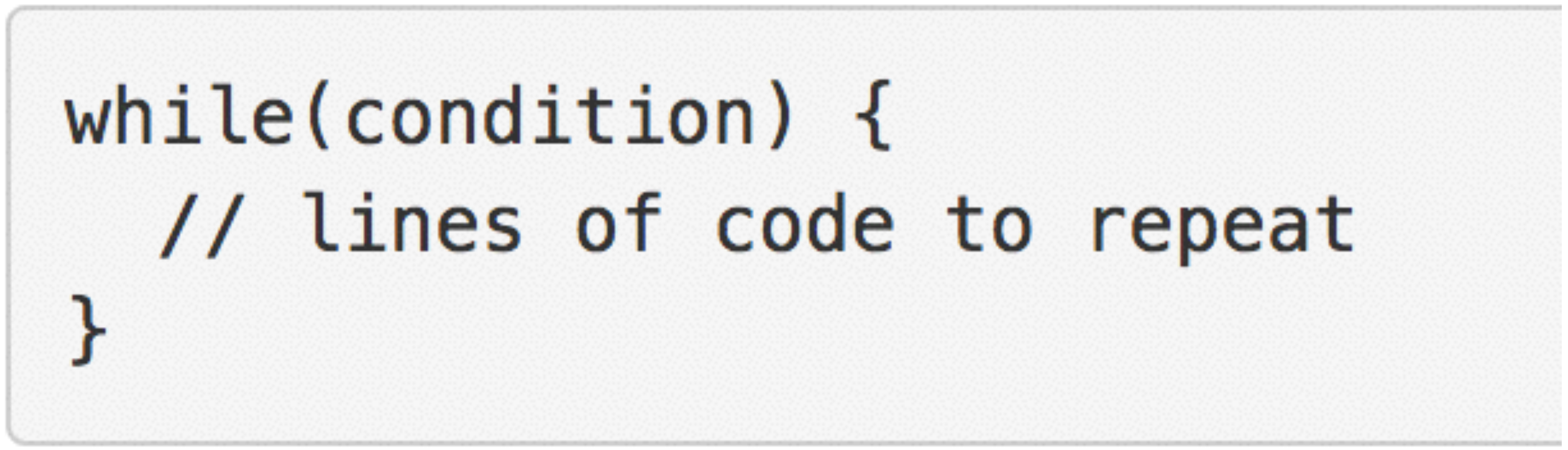
Exercise: robot sensor



```
var simInit = function () {}();

addWall(3, 0);

forward();
print(blocked());
forward();
forward();
forward();
```

```
false
```

Intro to while-loops

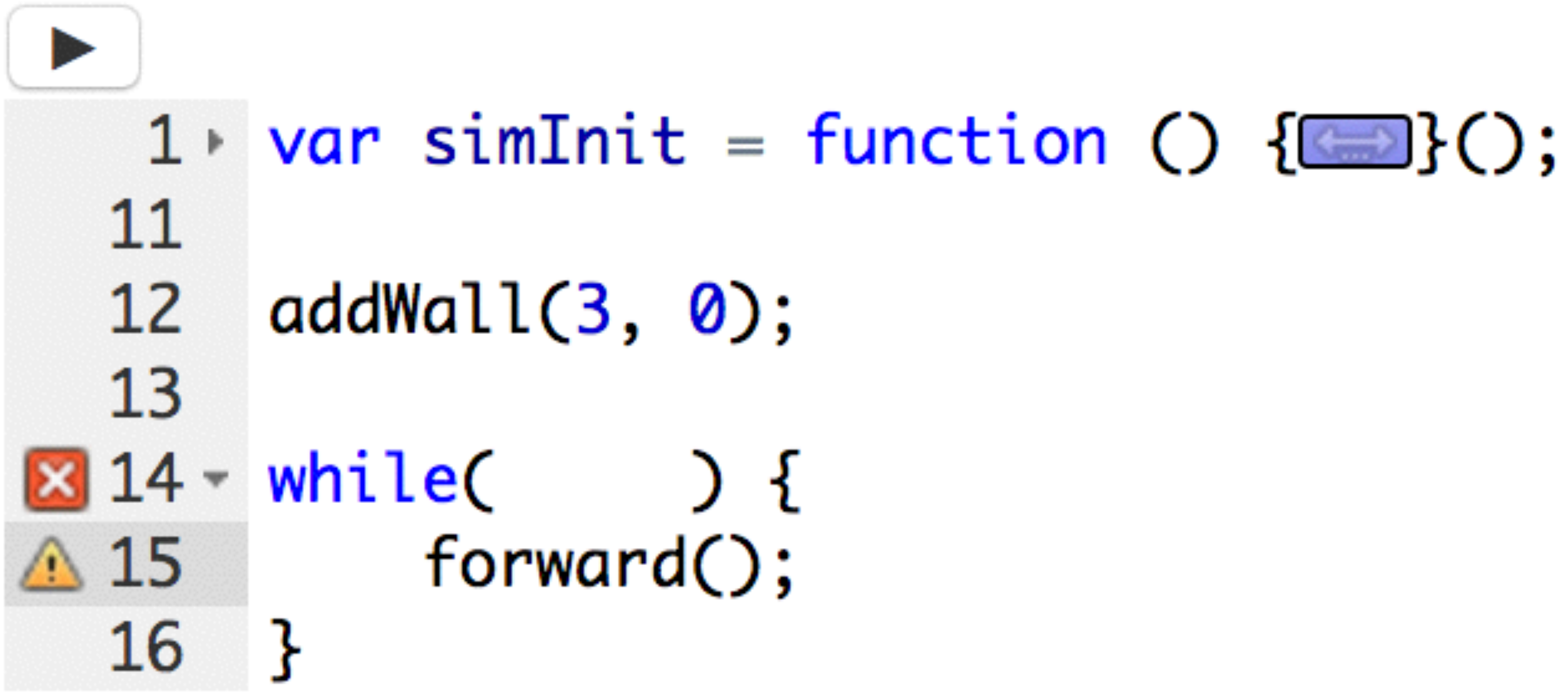


```
while(condition) {
  // lines of code to repeat
}
```

- **condition** is a **boolean** value
- if **condition** is **true**, execute body and test condition again
- if **condition** is **false**, skip body

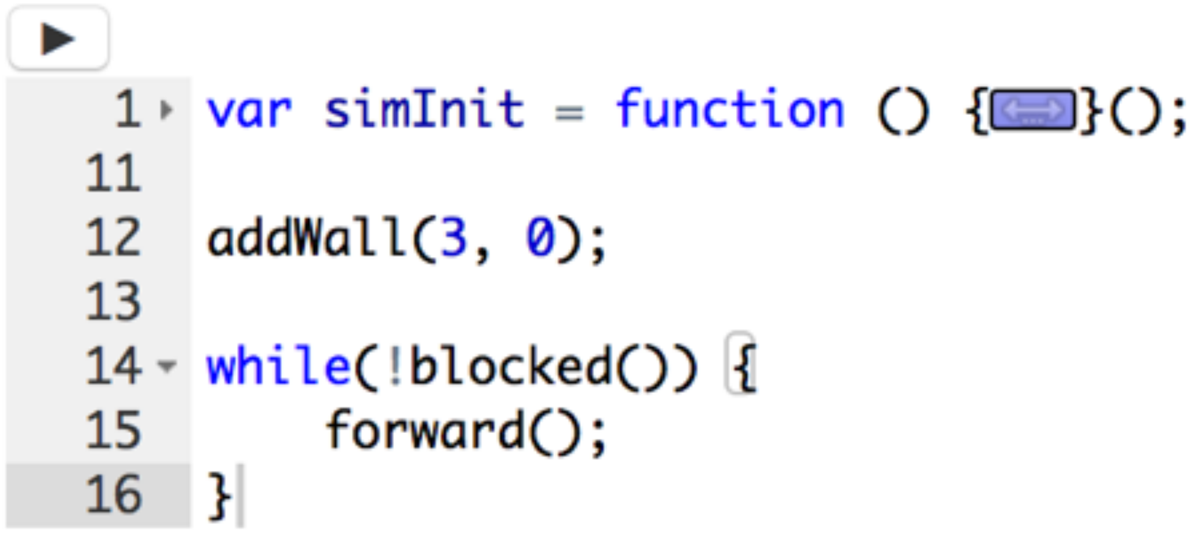
Exercise: robot bonk

Exercise: robot bonk

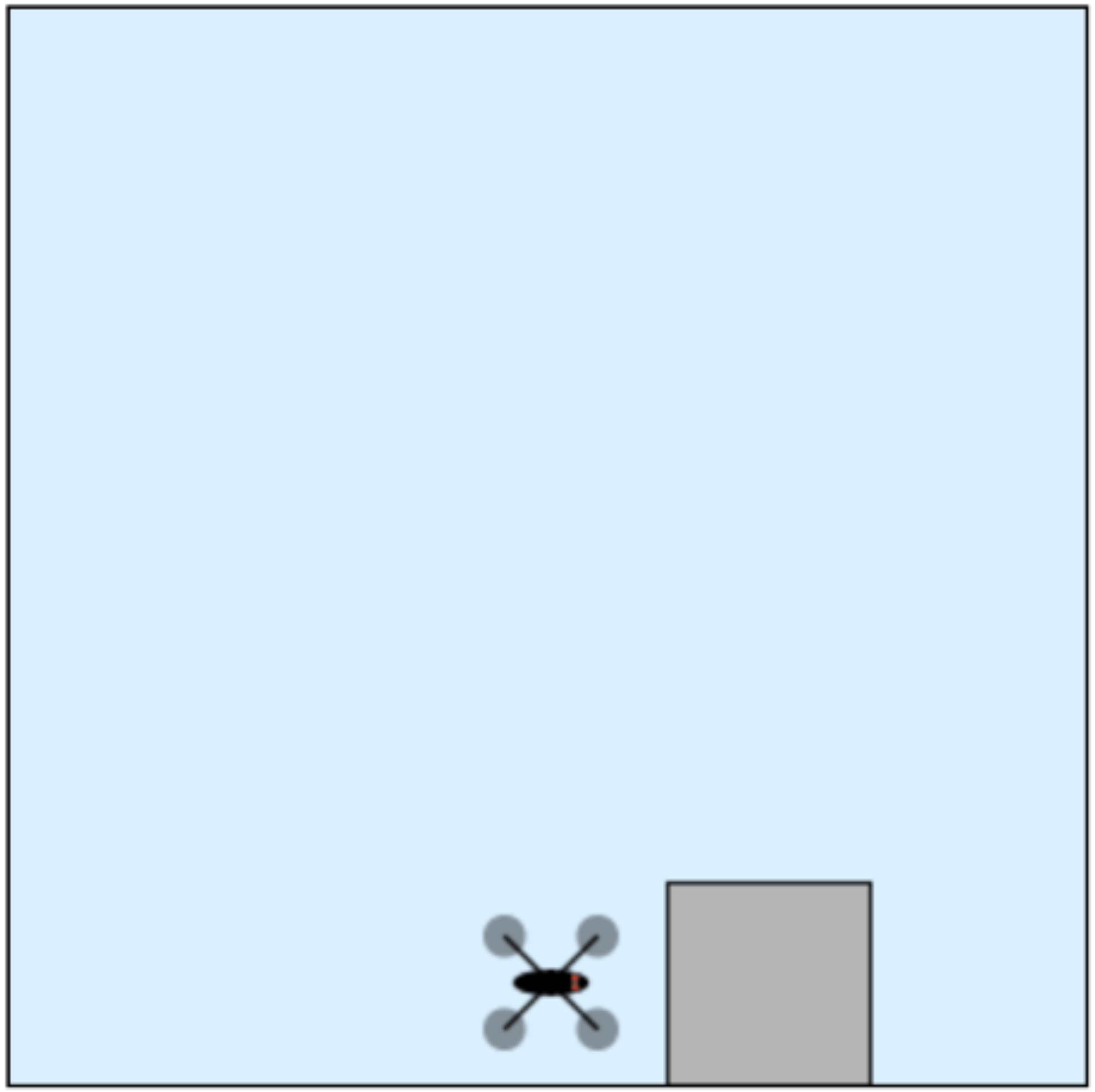


```
var simInit = function () {⟺}();

addWall(3, 0);

while(      ) {
    forward();
}
```

Exercise: robot bonk



```
var simInit = function () {⟷}();

addWall(3, 0);

while(!blocked()) {
    forward();
}
```



Exercise: driveToWall()

Factor, factor, factor!