Questions about Assignment 2?

Have you heard of the **20-80** Rule?

In high school, 80% - 20%

In college, Professors are responsible for providing ***20%*** *of content*

Students responsible for 80% *of the material.*

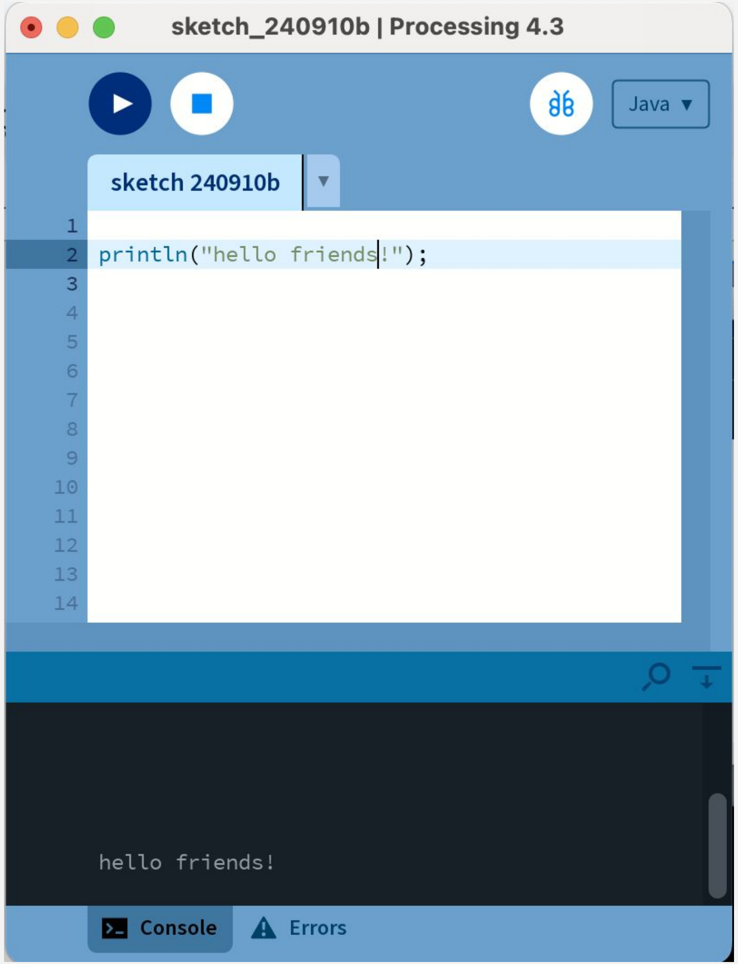
Students are required to prepare, review, and study outside of the classroom.

https://success.rice.edu/sites/g/files/bxs2961/files/inline-files/Lecture%20Participation.pdf

CS 101
The println function and the console

println()

The println() function writes to the console area, the black rectangle at the bottom of the Processing environment.



```
println("hello friends!");
```

```
hello friends!
```

https://processing.org/reference/println_.html

ICA

Answer Part 2, test the println function

1. Write the code to show the message “CSC101 is the best class!!” in the console of Processing
2. Run the following code and observe what happens
   ```
   println(1+1);
   ```
3. Run the following code and observe what happens
   ```
   println("the sum of 1+1 is ", 1+1);
   ```
4. Write the code to show the message “145698 / 2 = ” and computes this division in the console of Processing

CS 101
Variables in Processing

Mark the correct answer

a) A variable stores a value in the computer's memory so that it can be used later in the program.

b) A single variable can be used many times.

c) The value that is stored in a variable can change over time.

d) All of the above.

Computer Science

The process of finding solutions to difficult or complex issues by ***defining a set of steps or instructions*** *to be run by a computer for accomplishing a particular task*

Sometimes we need “to keep track” or “to remember” values

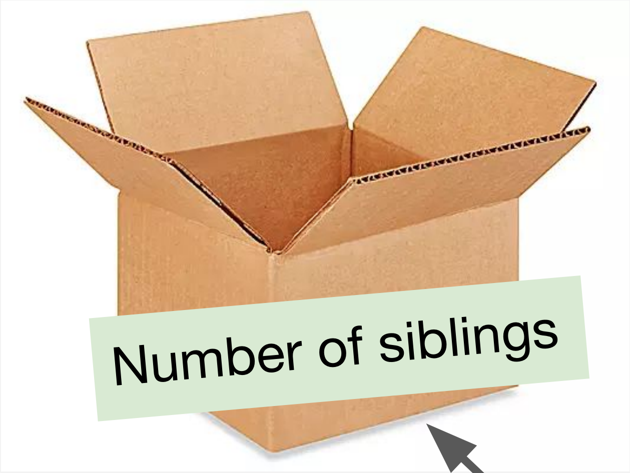
Examples of problems that require memory:

- Count the number of students in the class
- Find the student with the largest number of siblings
- Compute the average height of the students taking CSC101

Answer Part 3, Question 1 on the paper

We will store values in **Variables**

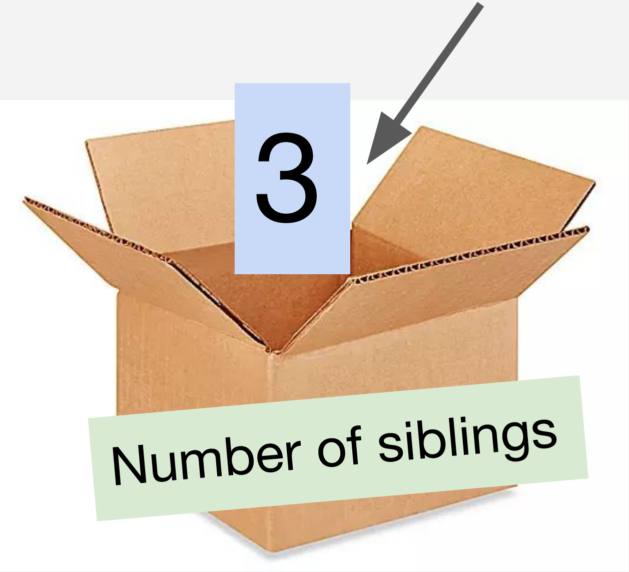
We will store values in **Variables**



Variable <u>Name</u>

We will store values in **Variables**

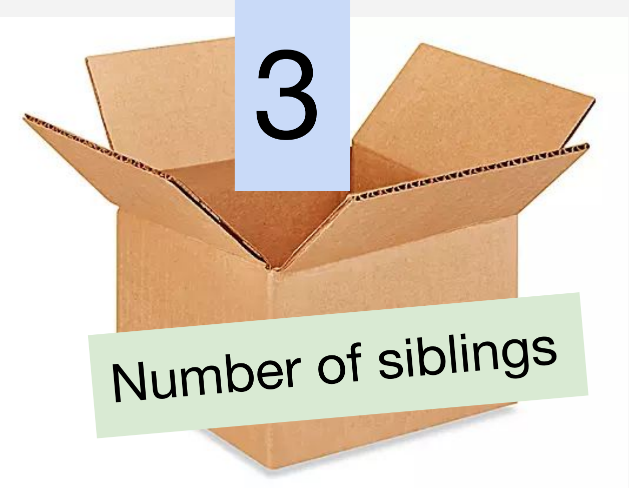
Variable <u>Value</u>



3

Number of siblings

We will store values in **Variables**

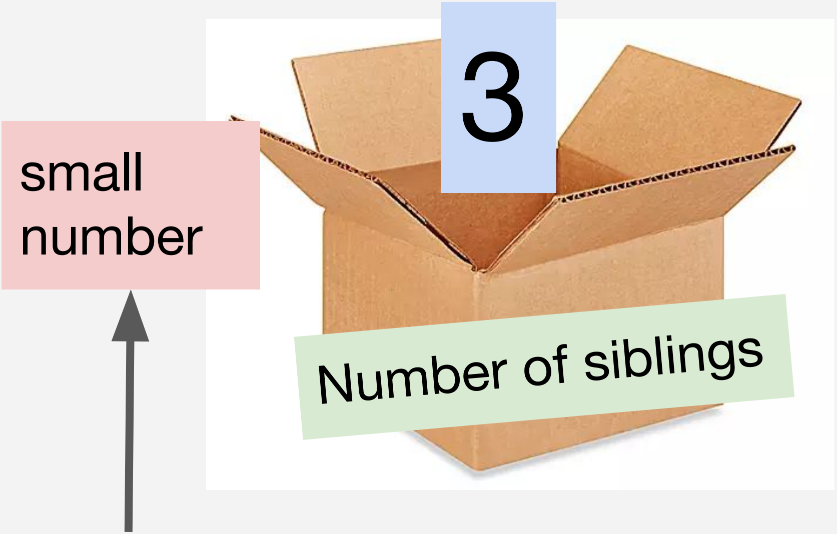


3

Number of siblings

123812312312 ← **Variable Address**

We will store values in **Variables**



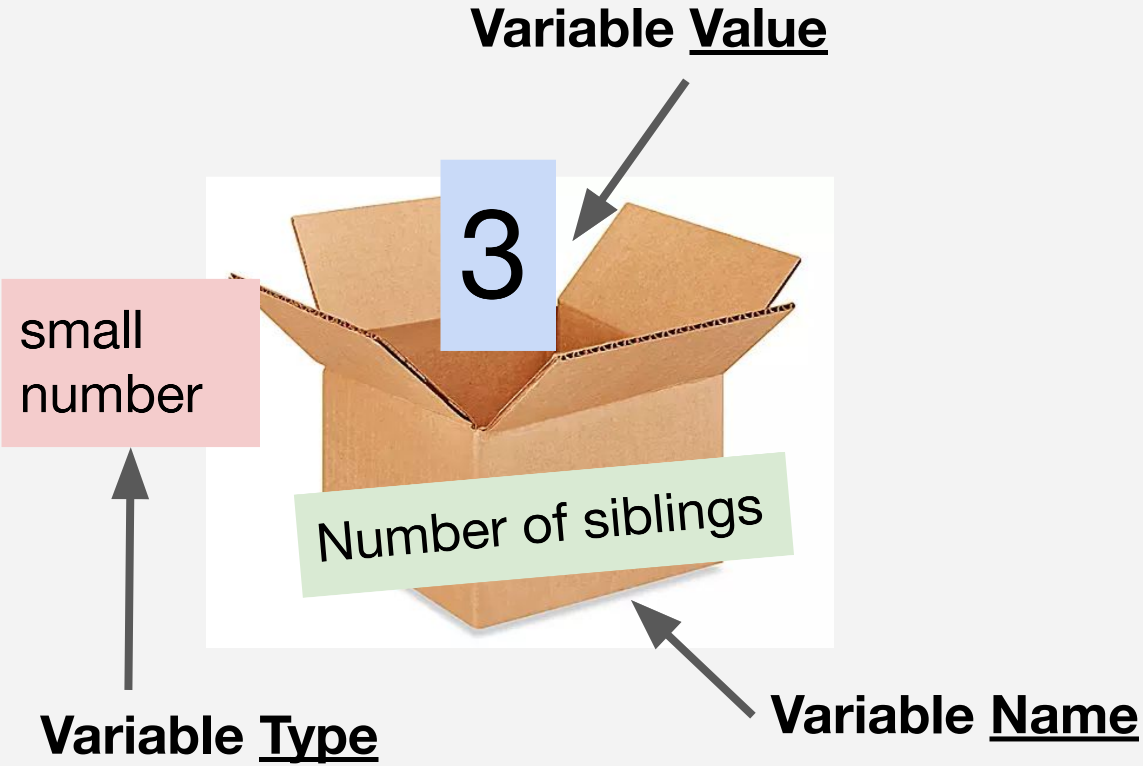
3

small number

Number of siblings

Variable <u>Type</u>

We will store values in **Variables**



Variable Value

3

small number

Number of siblings

Variable Type

Variable Name

Storing Numbers

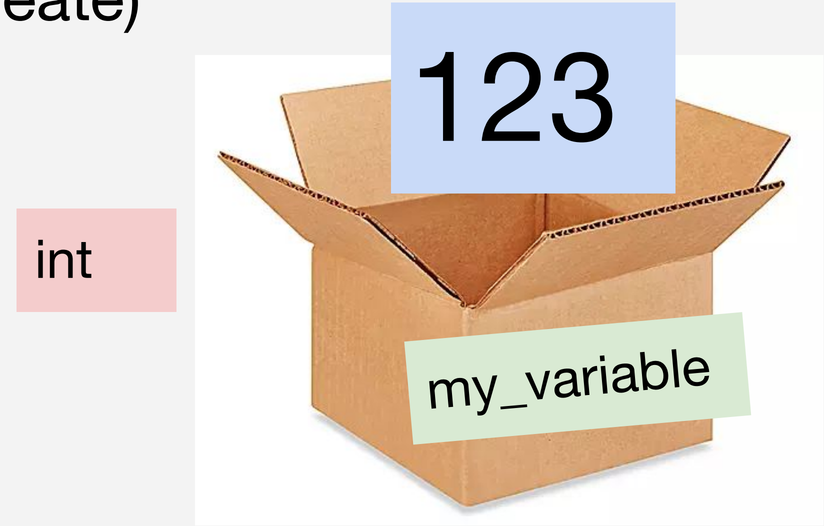
- A **variable** can store many types of values
- For the time being, we are only going to use them for storing whole numbers
  - These are called **integers**!

Variables

- What does a variable look like in Processing?
- How does one “create” a variable?

How to use Variables:

1. Read a variable
2. Write a variable
3. Declare a variable (or create)



1. Read a variable

- Print the value with println
- Then we use the variable as a parameter of a function
- When the variable is to the right of the equal (=) sign

We have been using variables already!

Built-In Variables

- Processing has a number of variables that are automatically declared an modified/updated behind-the-scenes
- You can use these variables too!
- A few that we’ll be using are:
  - `width` The width of the canvas
  - `height` The height of the canvas

Built-In Variables

- Suppose that you set the canvas size as follows:

```
size(600, 400);
```

- The size() function sets width and height variables:
  `width` is 600
  `height` is 400
- Now you can use the built-in variables **`width`** and **`height`** in your code

Using Built-In Variables

- Example

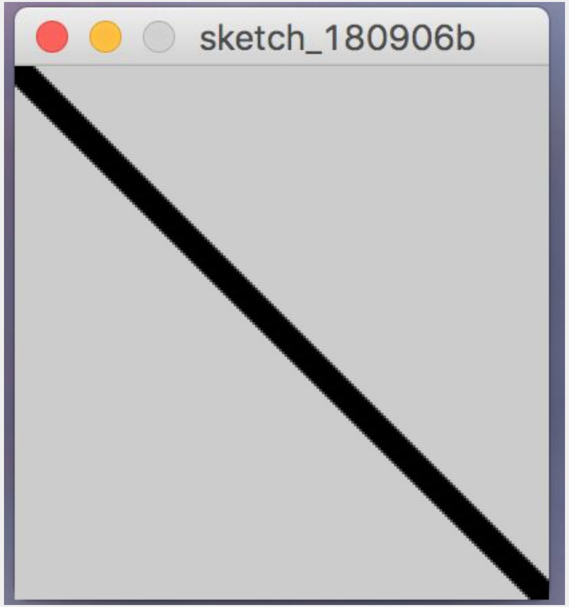
```
size(200, 200);         // sets width and height variables
strokeWeight(10);
line(0, 0, width, height);
//line(0, 0, 200, 200);  just the same as doing this
```

Reading the value in variable `width`

Using Built-In Variables

- Example

```
size(200, 200);
strokeWeight(10);
line(0, 0, width, height);
```



Let’s read the Built-in Variables

1. Run an empty processing program. What do you observe?
2. Run a program with the following line of code

```
println(width);
```

What do you observe?

Reading the value in variable `width`

Answer parte, question 2

Write the code to print the value of variable height.

width and height

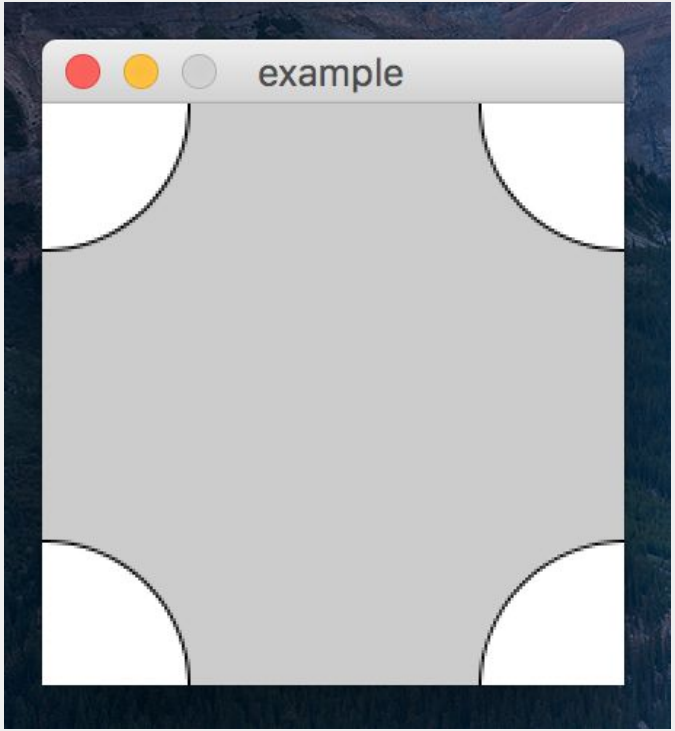
- What will this program produce?

```
size(200, 200);
int d = 100;
ellipse(0, 0, d, d);
ellipse(width, 0, d, d);
ellipse(0, height, d, d);
ellipse(width, height, d, d);
```

width and height

- What will this program produce?

```
size(200, 200);
int d = 100;
ellipse(0, 0, d, d);
ellipse(width, 0, d, d);
ellipse(0, height, d, d);
ellipse(width, height, d, d);
```



width and height

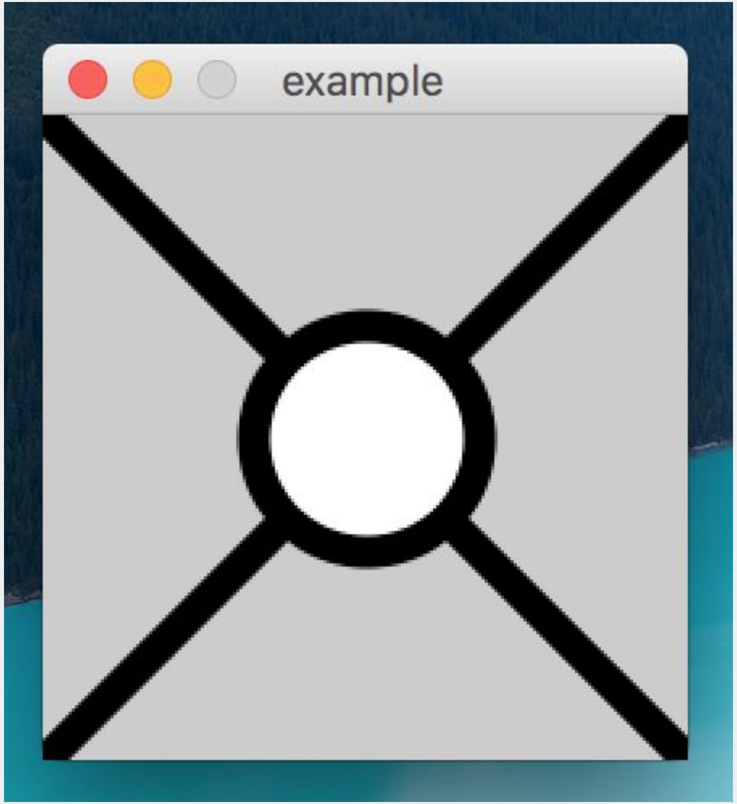
- What will this program produce?

```
size(200, 200);
strokeWeight(10);
line(0, 0, width, height);
line(width, 0, 0, height);
ellipse(100, 100, 70, 70);
```

width and height

- What will this program produce?

```
size(200, 200);
strokeWeight(10);
line(0, 0, width, height);
line(width, 0, 0, height);
ellipse(100, 100, 70, 70);
```



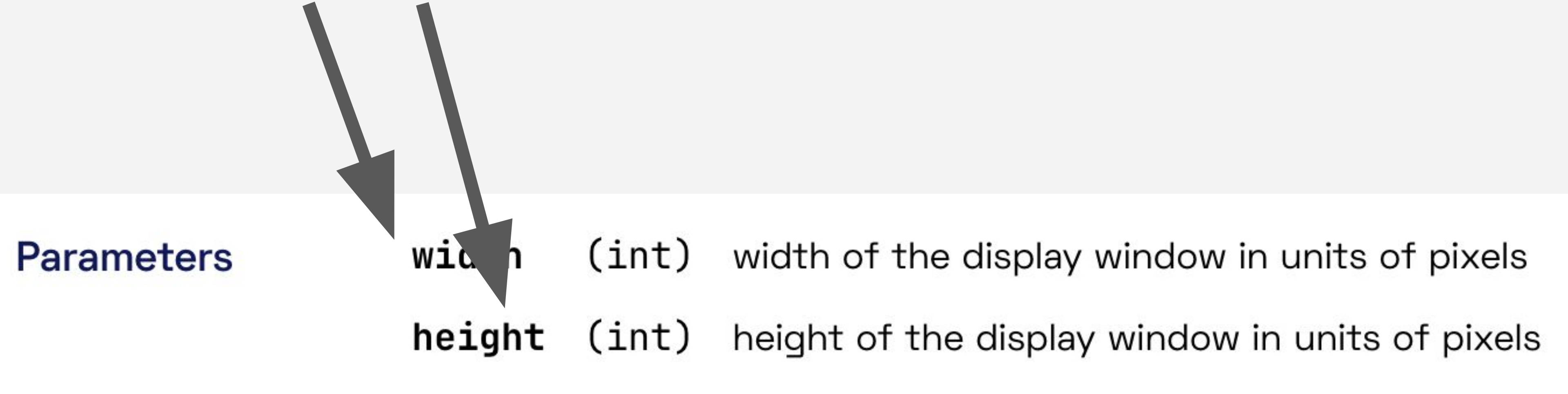
2. Write a variable

- Function parameters are also considered variables. So we write their value when we call the function

- When the variable is to the left of the equal (=) sign we write the variable

Writing parameters when calling a function

```
size(200, 200);
```



Parameters	width	(int)	width of the display window in units of pixels
	height	(int)	height of the display window in units of pixels

What does this program print in console?

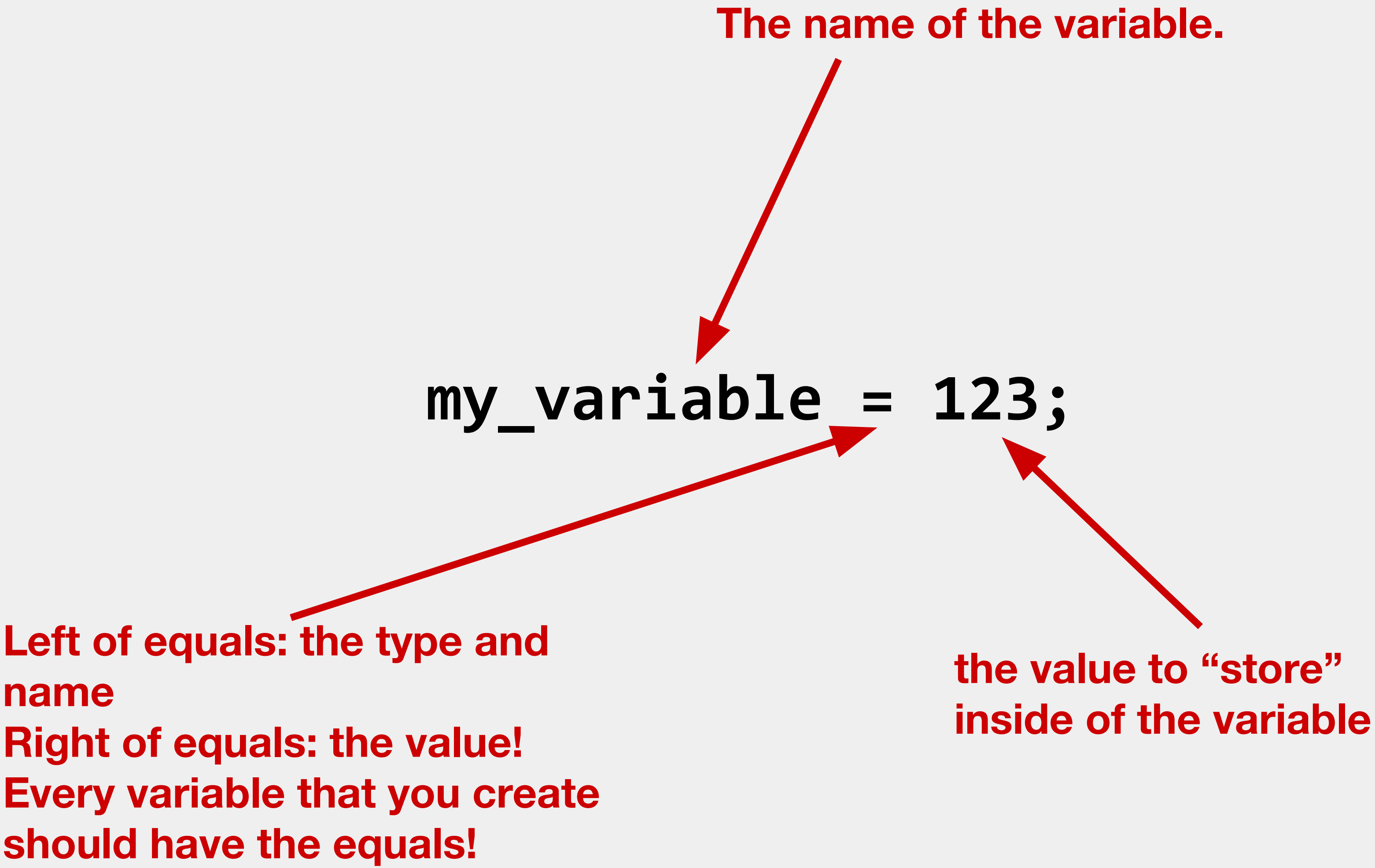
```
size(200, 200);
println(width);
```

The equal sign in programming

In math: x = y means y = x

In programing x = y means:

1. Read the value of y
2. Make a copy of such value
3. Write the value in x



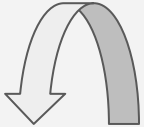
The name of the variable.

```
my_variable = 123;
```

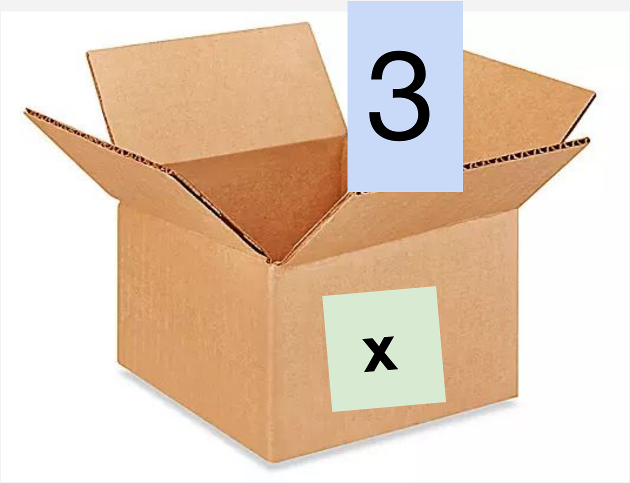
Left of equals: the type and name
Right of equals: the value!
Every variable that you create should have the equals!

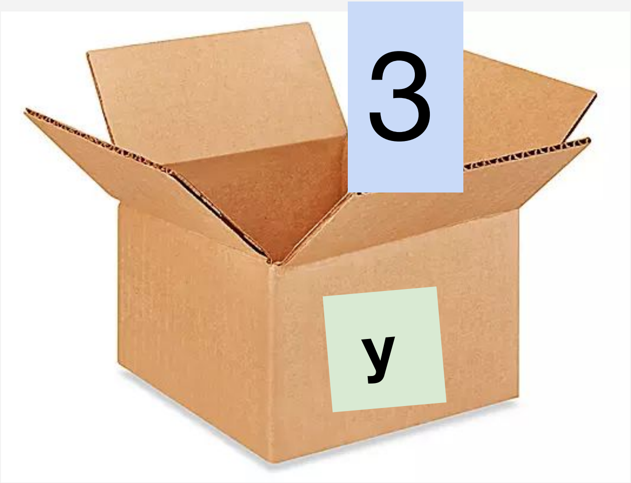
the value to “store” inside of the variable

```
y = 3;
```



```
x = y;
```



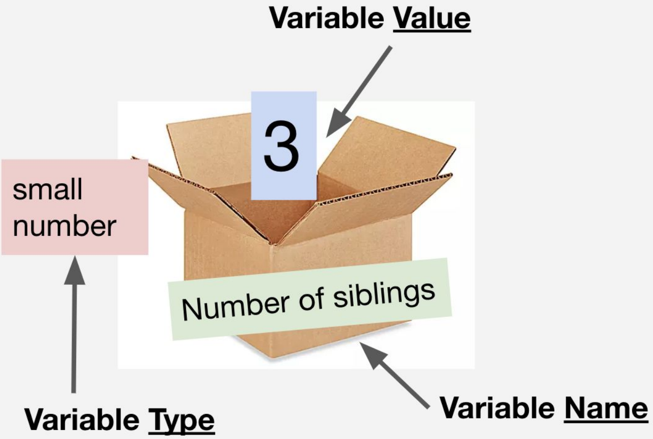


3. Declare a variable

- Write declaration at the beginning of the code

Syntax:

```
variable_type  variable_name = value;
```

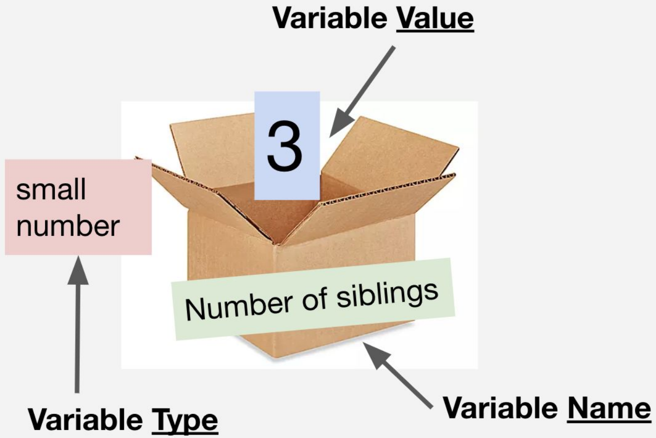


3. Declare a variable

- Write declaration at the beginning of the code

Syntax:

small number Number of siblings = 3 ;



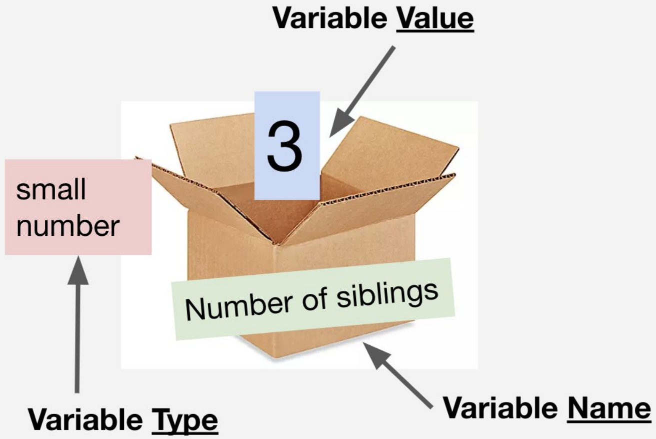
3. Declare a variable

- Write declaration at the beginning of the code

Syntax:

int Number of siblings **= 3 ;**

int <= integer <= to store numbers



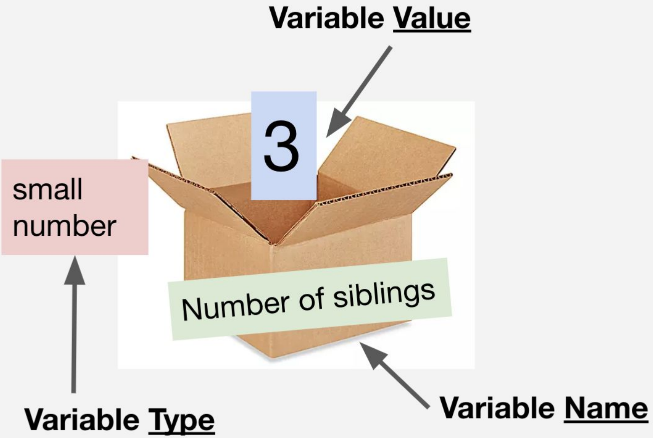
3. Declare a variable

- Write declaration at the beginning of the code

Syntax:

```
int     number_of_siblings     =  3  ;
```

Variable name ***cannot*** *have spaces*



Variable Value

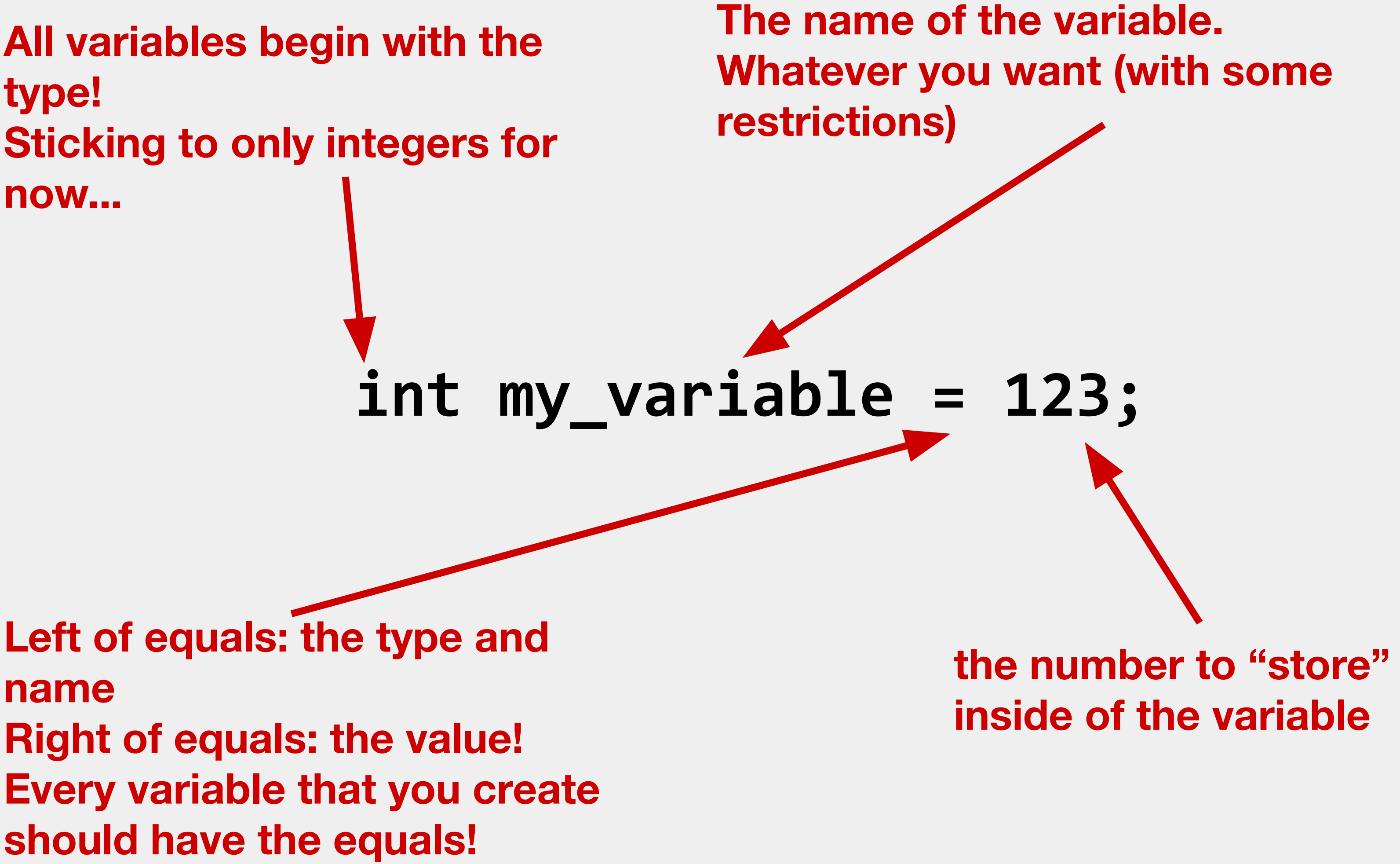
3

small number

Number of siblings

Variable Type

Variable Name



All variables begin with the type!
Sticking to only integers for now...

The name of the variable.
Whatever you want (with some restrictions)

```
int my_variable = 123;
```

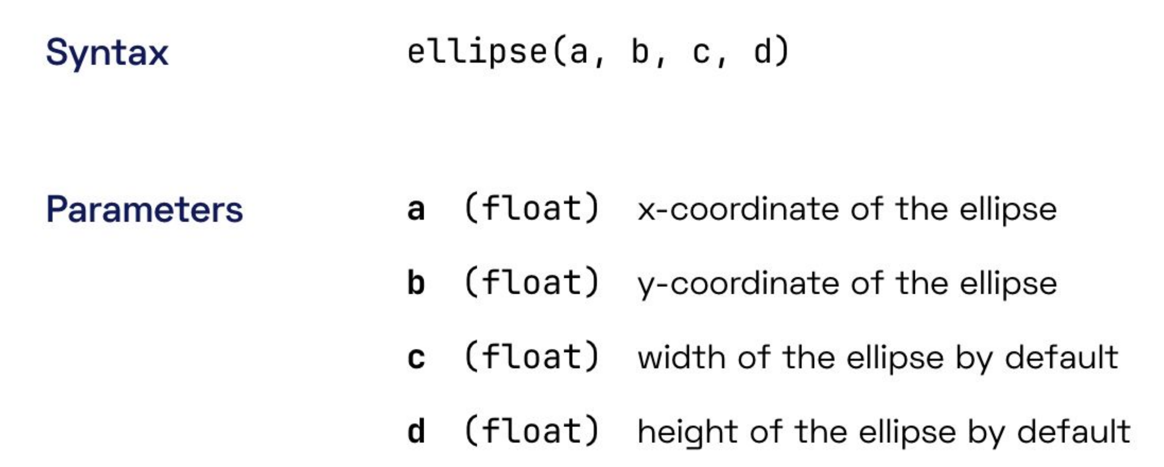
Left of equals: the type and name
Right of equals: the value!
Every variable that you create should have the equals!

the number to “store” inside of the variable

```
int name = 123;
```

- In programming, this is referred to as ***declaring a variable***
- The ***name*** can be anything, with a few restrictions
    - Can have upper/lower case letters, numbers, and a few special symbols such as underscore and dashes.
    - May ***not*** begin with a number

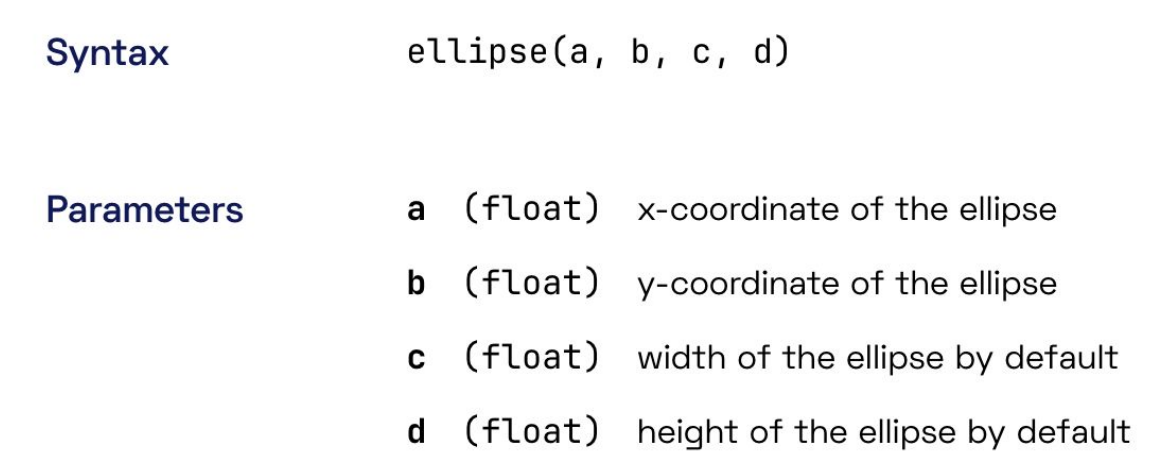
Using Variables as Parameters



Syntax	`ellipse(a, b, c, d)`		
Parameters	**a**	(float)	x-coordinate of the ellipse
	b	(float)	y-coordinate of the ellipse
	c	(float)	width of the ellipse by default
	d	(float)	height of the ellipse by default

```
ellipse(57, 92, 100, 200);
```

Using Variables as Parameters



Syntax

```
ellipse(a, b, c, d)
```

Parameters

a	(float)	x-coordinate of the ellipse
b	(float)	y-coordinate of the ellipse
c	(float)	width of the ellipse by default
d	(float)	height of the ellipse by default

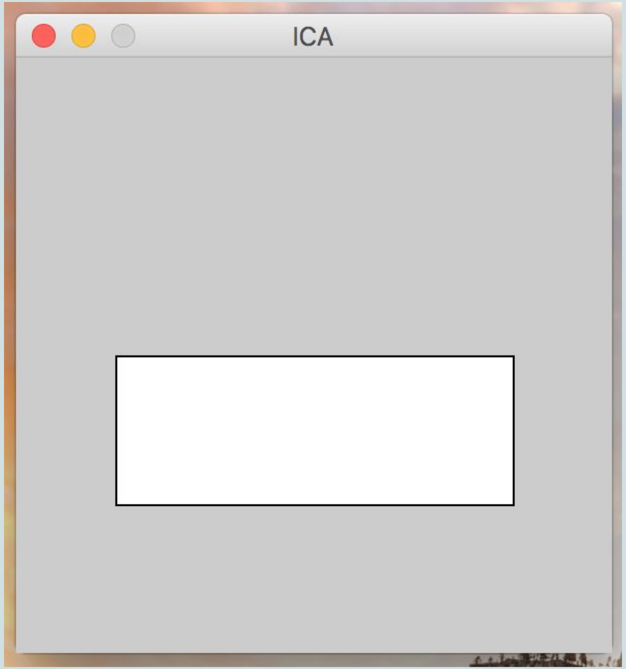
```
int xCoord = 57;
int yCoord = 92;
ellipse(xCoord, yCoord, 100, 200);
```

3. What values are needed?

```
//rect(50, 150, 200, 75);

size(300, 300);
int xCoord = ? ;
int yCoord = ? ;
int wVal = ? ;
int hVal = ? ;

rect(xCoord, yCoord, wVal, hVal);
```



Resize these circles using variables

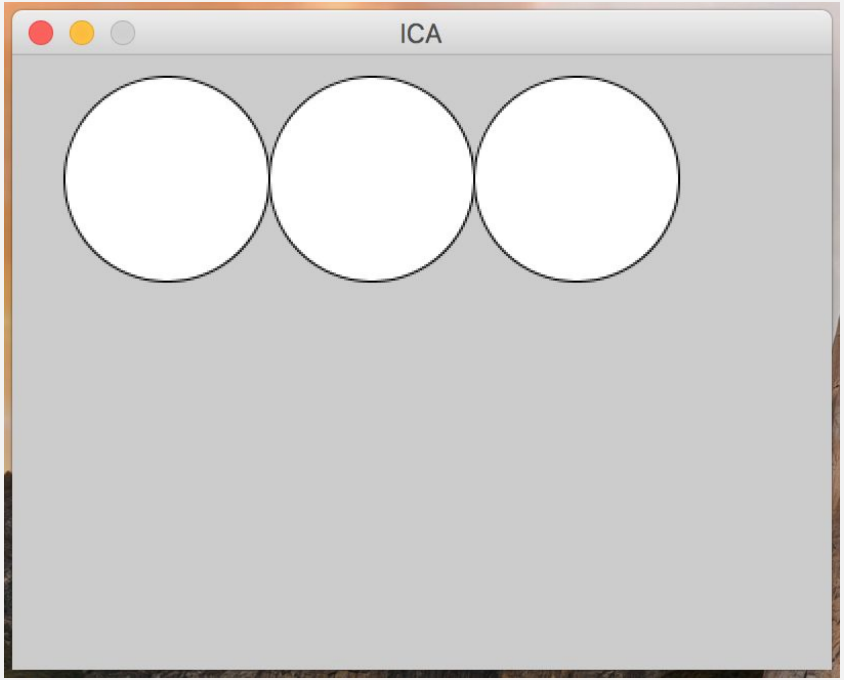
```
size(400, 300);
ellipse(75, 60, 80, 80);
ellipse(175, 60, 80, 80);
ellipse(275, 60, 80, 80);
```

```
size(400, 300);
ellipse(75, 60, 100, 100);
ellipse(175, 60, 100, 100);
ellipse(275, 60, 100, 100);
```

Resize the circles

```
size(400, 300);
ellipse(75, 60, 100, 100);
ellipse(175, 60, 100, 100);
ellipse(275, 60, 100, 100);
```

How many arguments did you have to change to make it look like this?



Resize the circles without variables

```
size(400, 300);
ellipse(75, 60, 80, 80);
ellipse(175, 60, 80, 80);
ellipse(275, 60, 80, 80);
```

```
size(400, 300);
ellipse(75, 60, 100, 100);
ellipse(175, 60, 100, 100);
ellipse(275, 60, 100, 100);
```

Resize the circles

```
int diameter = 80;
size(400, 300);
ellipse(75, 60, diameter, diameter);
ellipse(175, 60, diameter, diameter);
ellipse(275, 60, diameter, diameter);
```

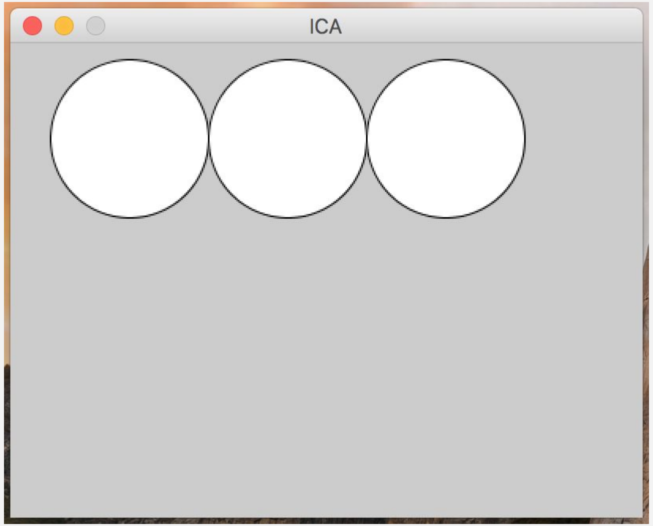
Resize the circles

```
int diameter = 80;
size(400, 300);
ellipse(75, 60, diameter, diameter);
ellipse(175, 60, diameter, diameter);
ellipse(275, 60, diameter, diameter);
```

```
int diameter = 100;
size(400, 300);
ellipse(75, 60, diameter, diameter);
ellipse(175, 60, diameter, diameter);
ellipse(275, 60, diameter, diameter);
```

Other variables ?

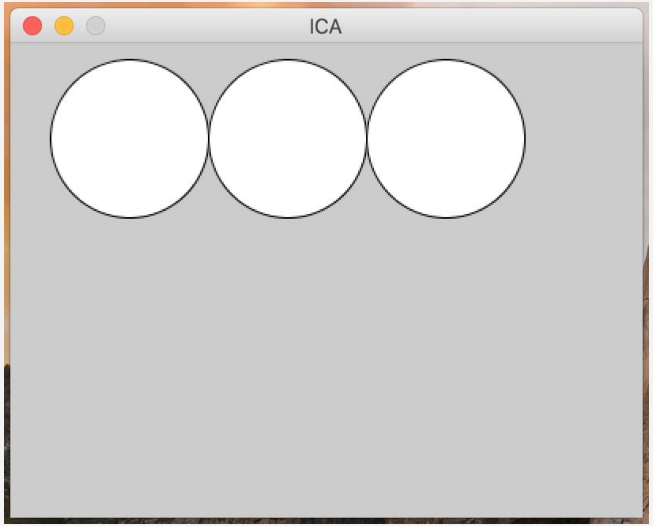
```
int diameter = 100;
size(400, 300);
ellipse(75, 60, diameter, diameter);
ellipse(175, 60, diameter, diameter);
ellipse(275, 60, diameter, diameter);
```



How else should we use variables in this program?

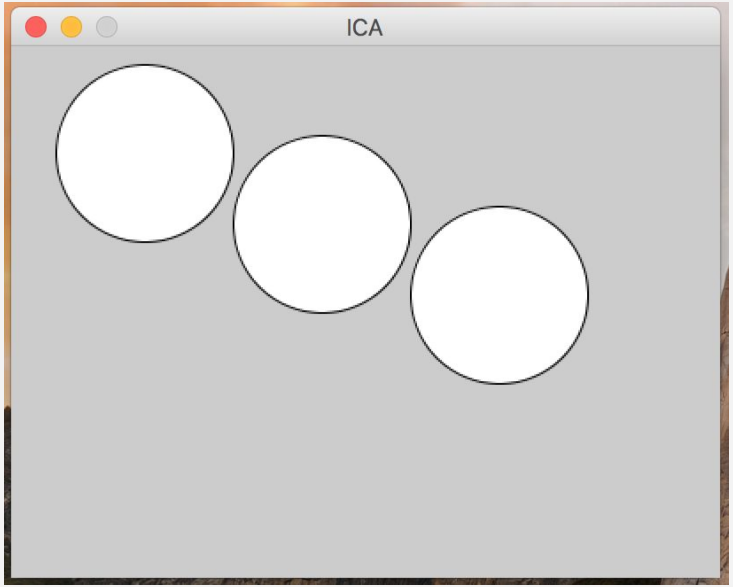
Other variables ?

```
int diameter = 100;
int y = 60;
size(400, 300);
ellipse(75, y, diameter, diameter);
ellipse(175, y, diameter, diameter);
ellipse(275, y, diameter, diameter);
```



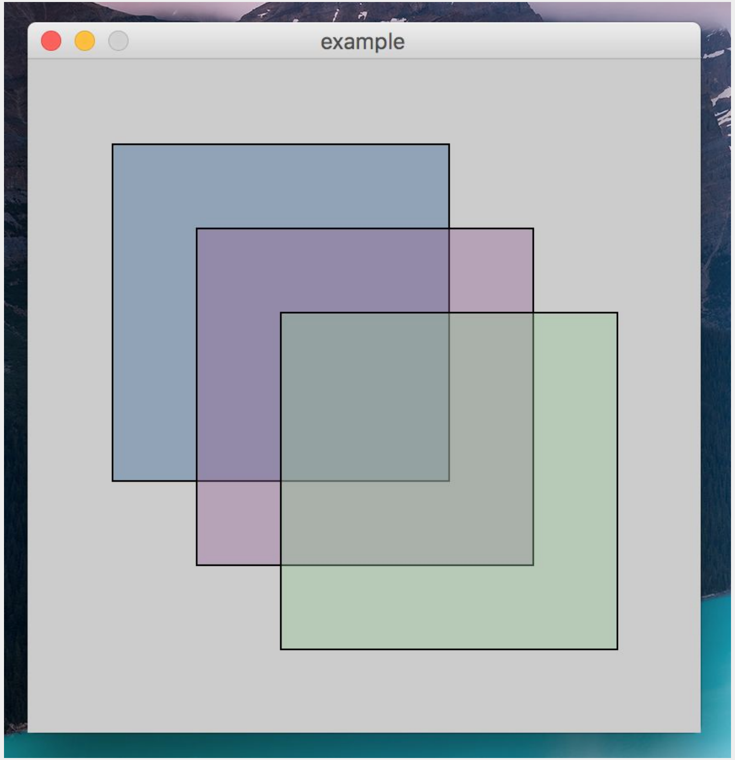
Other variables ?

```
int diameter = 100;
int y = 60;
size(400, 300);
ellipse(75, y, diameter, diameter);
y = 100;
ellipse(175, y, diameter, diameter);
y = 140;
ellipse(275, y, diameter, diameter);
```



Color can also be numeric variables

```
size(400, 400);
int r = 50;
int g = 100;
int b = 150;
fill(r, g, b, 100);
rect(50, 50, 200, 200);
r = r + 100;
fill(r, g, b, 100);
rect(100, 100, 200, 200);
g = g + 100;
fill(r, g, b, 100);
rect(150, 150, 200, 200);
```



4. Draw a picture with pen/pencil. Do not use Processing

```
size(26, 26);
int valueA = 6;
rect(4, 4, valueA, valueA);
ellipse(12, 12, valueA - 3, valueA + 2);
```