TODO before this lecture

- Check Piazza
- Read (and follow) GSWP - Preface, Ch 1, Ch 2
- Watch required videos (last slide of last lecture)
- Read Assignment 1 (posted on the course website)

TODO after this lecture

Check the schedule to know the weekly assignments

- Read these lecture slides
- GSWP - Chapter 2
- Hello Processing (tutorial) (1 hour)
- You Should Learn to Code (TED video) (10 mins)
- Check your grade for ICA 1 on Gradescope
- Check Piazza
- Work on Assignment 1 (posted on the course website)
- Attend office hours or ask in Piazza when you need help

Any burning questions?

New class Q & A
(Questions and Answers)

On Live Piazza

CS 101
Intro to Processing

Chapter 2

Processing - What is it?

- ***Processing*** is a flexible software language for learning how to code within the context of the visual arts
- In other words, it is a programming language designed for easily creating images, graphics, and animations!
- We will do lots of this throughout the semester
- *The visual feedback is helpful when first learning to program*
- *Since processing is primarily used to generate graphics, it is excellent as a “first language”*

Processing - What is it?

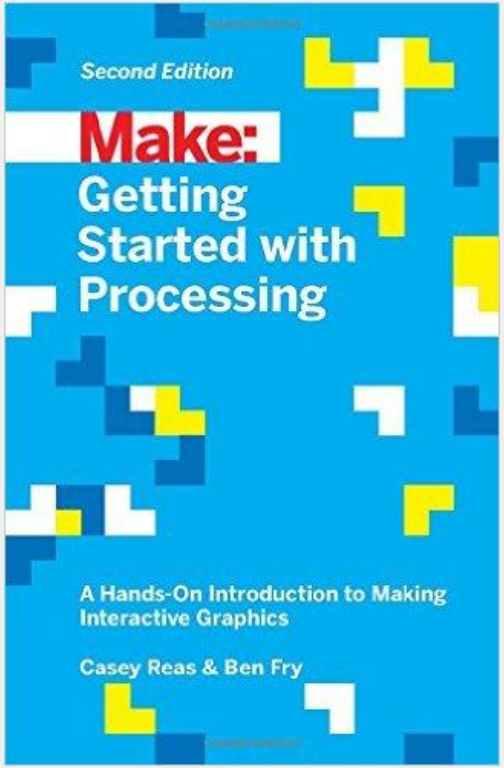
- ***Processing*** is just one of many computer programming languages
- You may have heard of some of the more popular languages such as ***C***, ***Java***, and ***Python***
- If you continue on in computer science, you will learn all of these languages, and perhaps more!
- For this class, we stick to ***Processing***

Processing - What is it?

- ***Processing*** is actually a ***dialect*** of Java
- This means the ***syntax*** (the way the code looks and is structured) is very similar to Java
  - You probably don’t know Java, but when you someday learn it, you’ll see the clear resemblance
  - Processing isn’t Java :)

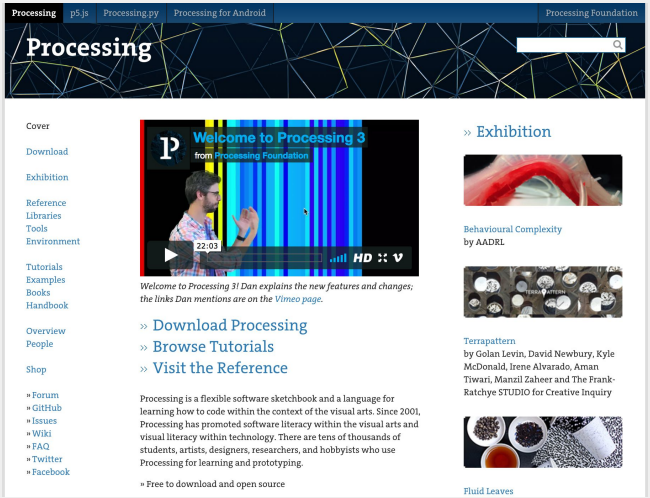
Processing - The Textbook

- This is our textbook
- We will be following the order and structure of this book closely while learning processing
- This book has great examples and visual aids to help solidify concepts, so make sure and do the readings!



Processing - processing.org

- Along with the textbook, processing.org will be a great resource for you to learn processing
  - This site has great learning resources, code examples, tutorials, and more
  - You can also download the processing IDE from here



Processing - processing.org

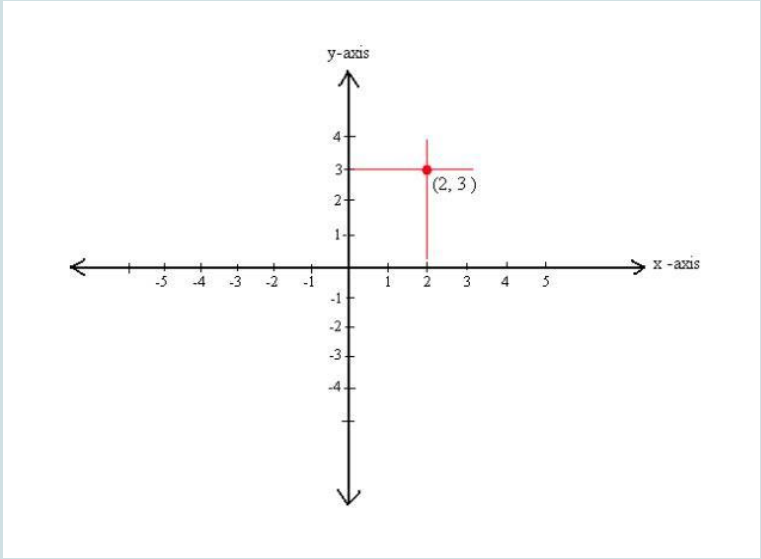
- The first step to get started with processing is to download and install the Processing IDE (Integrated Development Environment)
  - The Processing IDE is the program you will use to both ***write*** processing code and to ***run*** the programs you write
- Start at processing.org
  - https://processing.org/download/

Processing - How does it work?

- In Processing (as with most programming languages) we write a sequence of instructions, which get executed step-by-step, in order
  - Algorithms!
- In general, the instructions you give to a computer are executed from the top to the bottom
  - We will learn how to change this later!
  - Just as in the decision trees, we need to go in different directions depending on the decision

Drawing

- Do you remember drawing on graph paper?
- Let's look back at the cartesian coordinates
  - *Use your whiteboards*
  - *Make a circle with center at (0,0) with width 6 and height 6*



3 minutes

Processing

- Let’s follow the first example on Chapter 2

Open Gradescope

- Today we will use Gradescope!
- Wait until I ask you to do each problem
- First let me show you how to submit on Gradescope
- Now, please open Gradescope on your laptop

1 minute for individual (silent) work

Problem 1

- Open Processing
- Test this code

```
ellipse(50, 50, 80, 80);
```

- Take a screenshot of your entire computer after running the code
- Upload the screenshot to Gradescope
- Try changing the numbers in the code, and see what differences occur

1 minute for individual (silent) work

4 minutes for group work

```
ellipse(50, 50, 80, 80);
```

- Let’s return to this line of code
- What happens when we use different values for X, Y, width, and height?
- How does it change?

```
ellipse(50, 50, 80, 80);
```

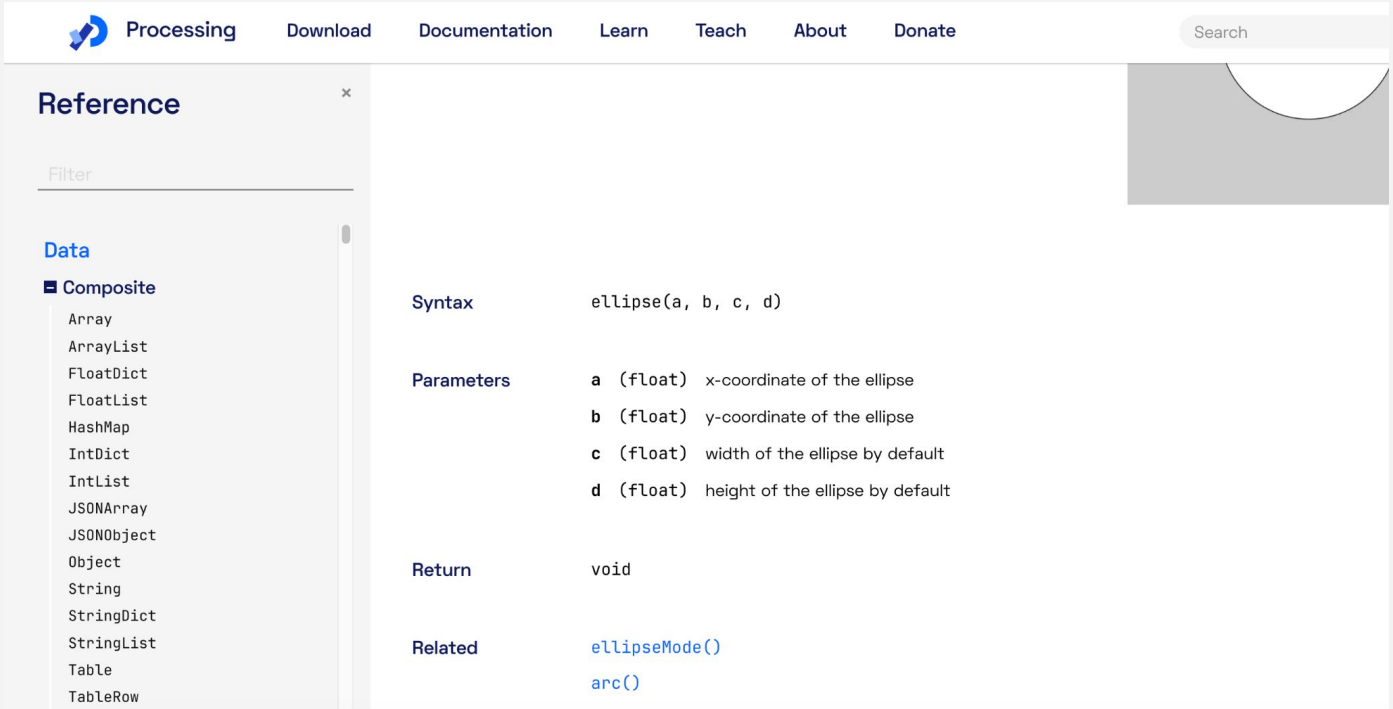
- We need to know how big the canvas is!
- The default size of the canvas is 100 X 100
- The code below

```
ellipse(50, 50, 80, 80);
```

  - draws an ellipse with the center at (50,50)
  - the width is 80
  - the height is 80
  - (a circle…)

Always read the reference!

In Processing go to **Help** -> **Reference**

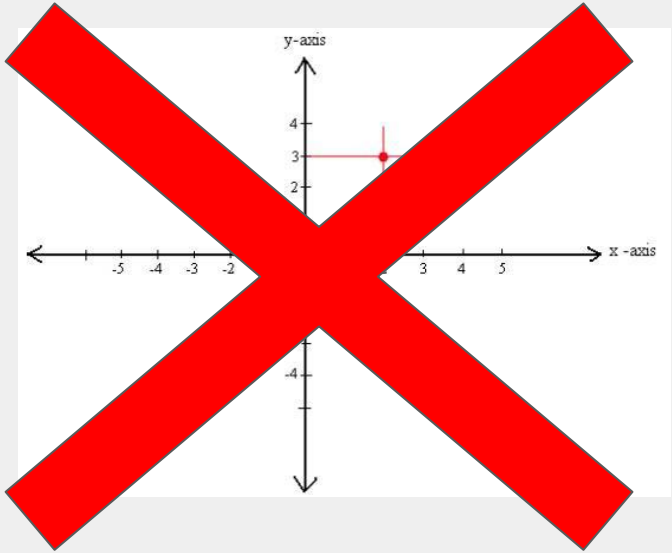


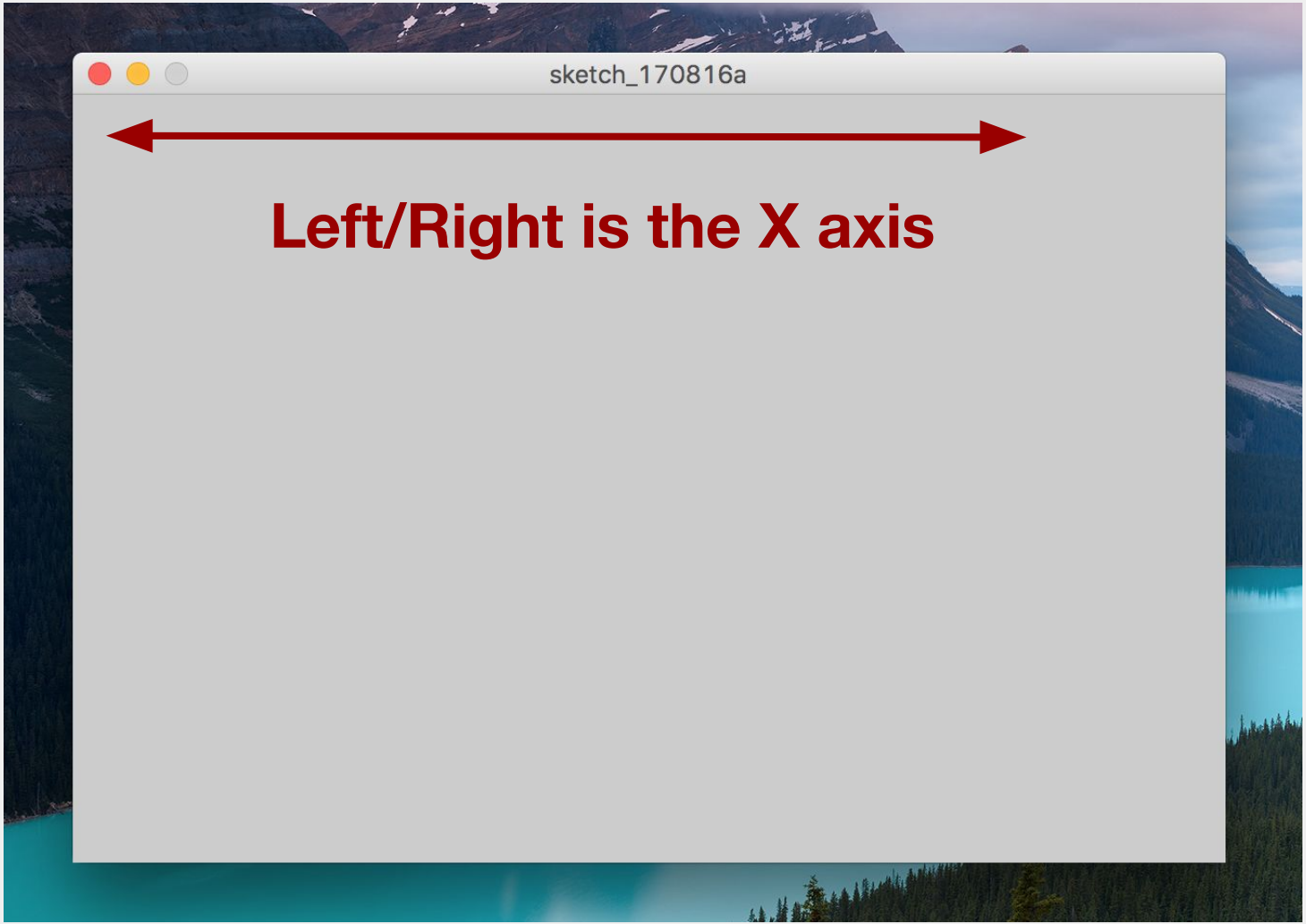
```
ellipse(50, 50, 80, 80);
```

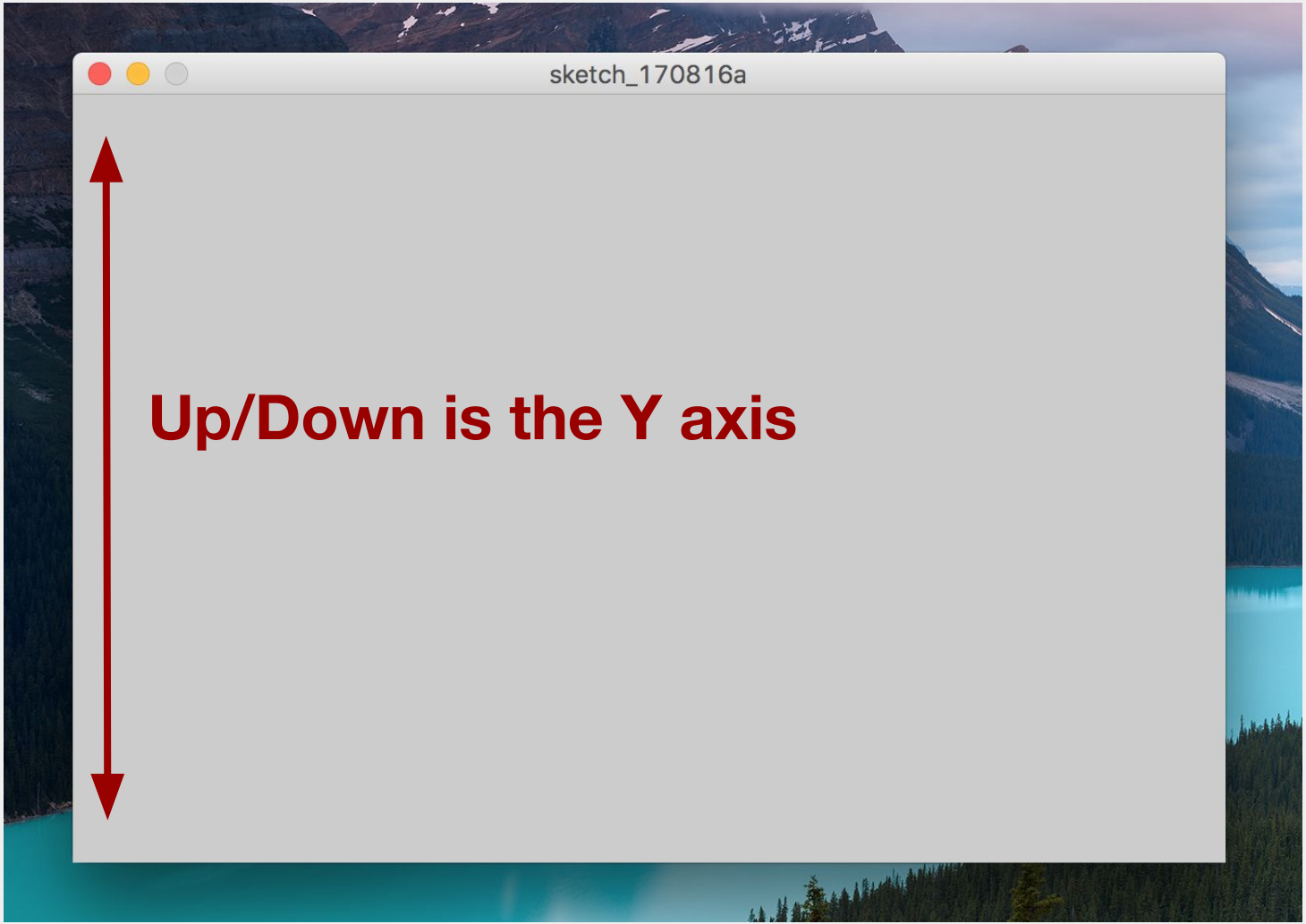
This line of code means “draw an ellipse, with the center `50` pixels over from the left and `50` pixels down from the top, with a width of `80` pixels and height of `80` pixels”

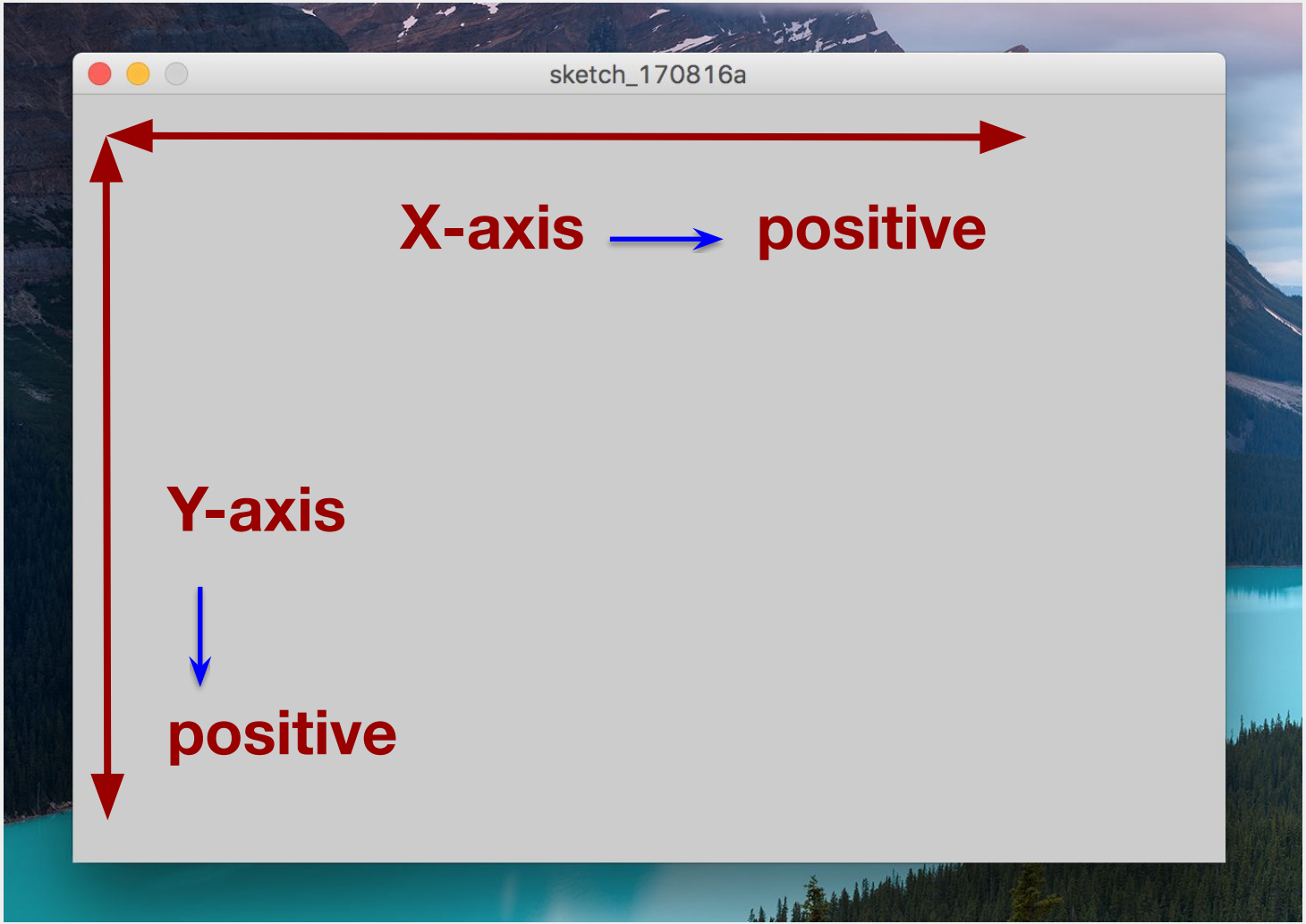
The Canvas

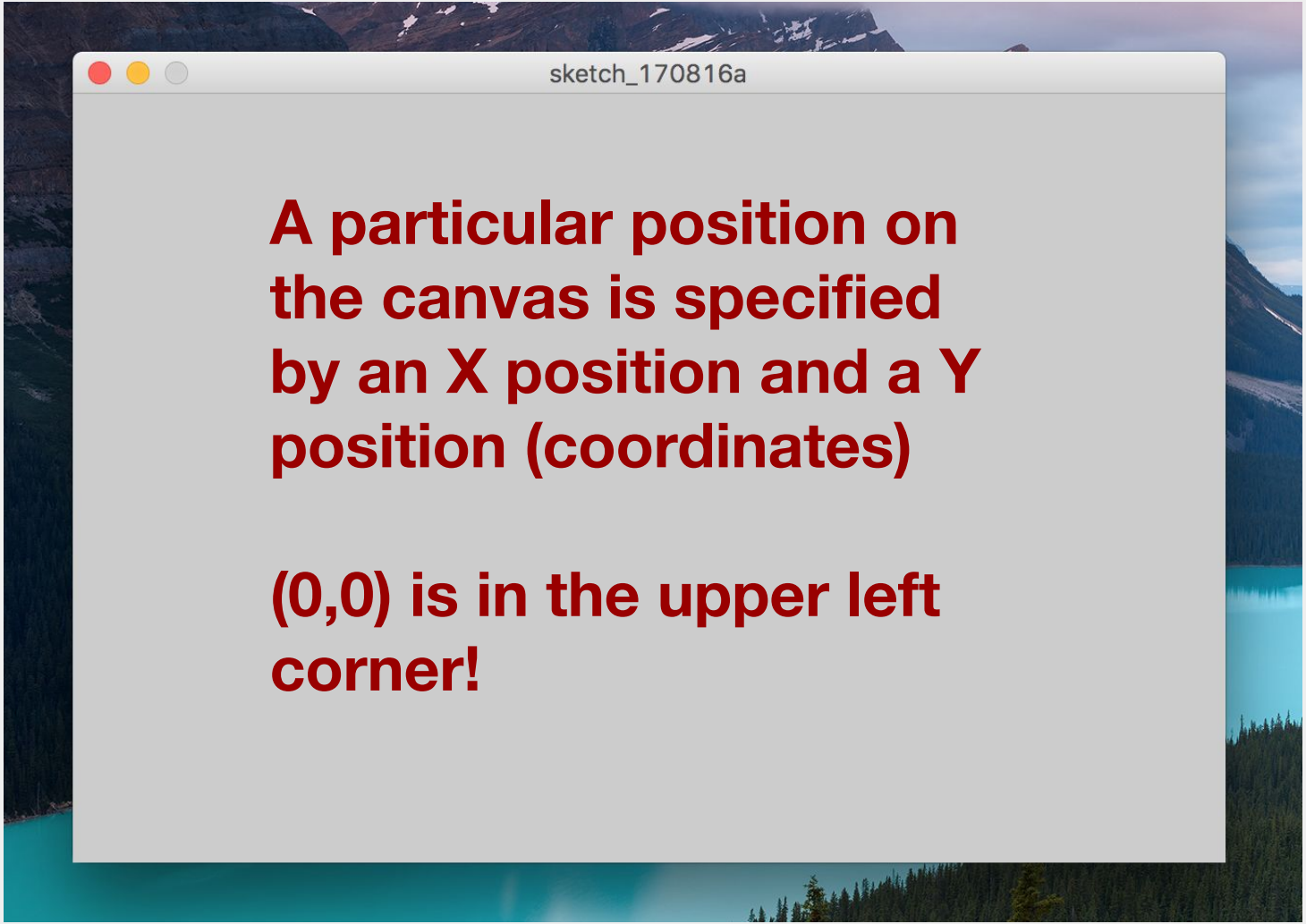
- As mentioned, processing programs are visual
- Graphics are drawn onto the ***canvas***
- The canvas is a grid of tiny pixels
  - Arranged in rows and columns
- We specify where we want things to be drawn on the processing canvas using pixel coordinates
- BUT, the coordinate system is oriented in a different way…

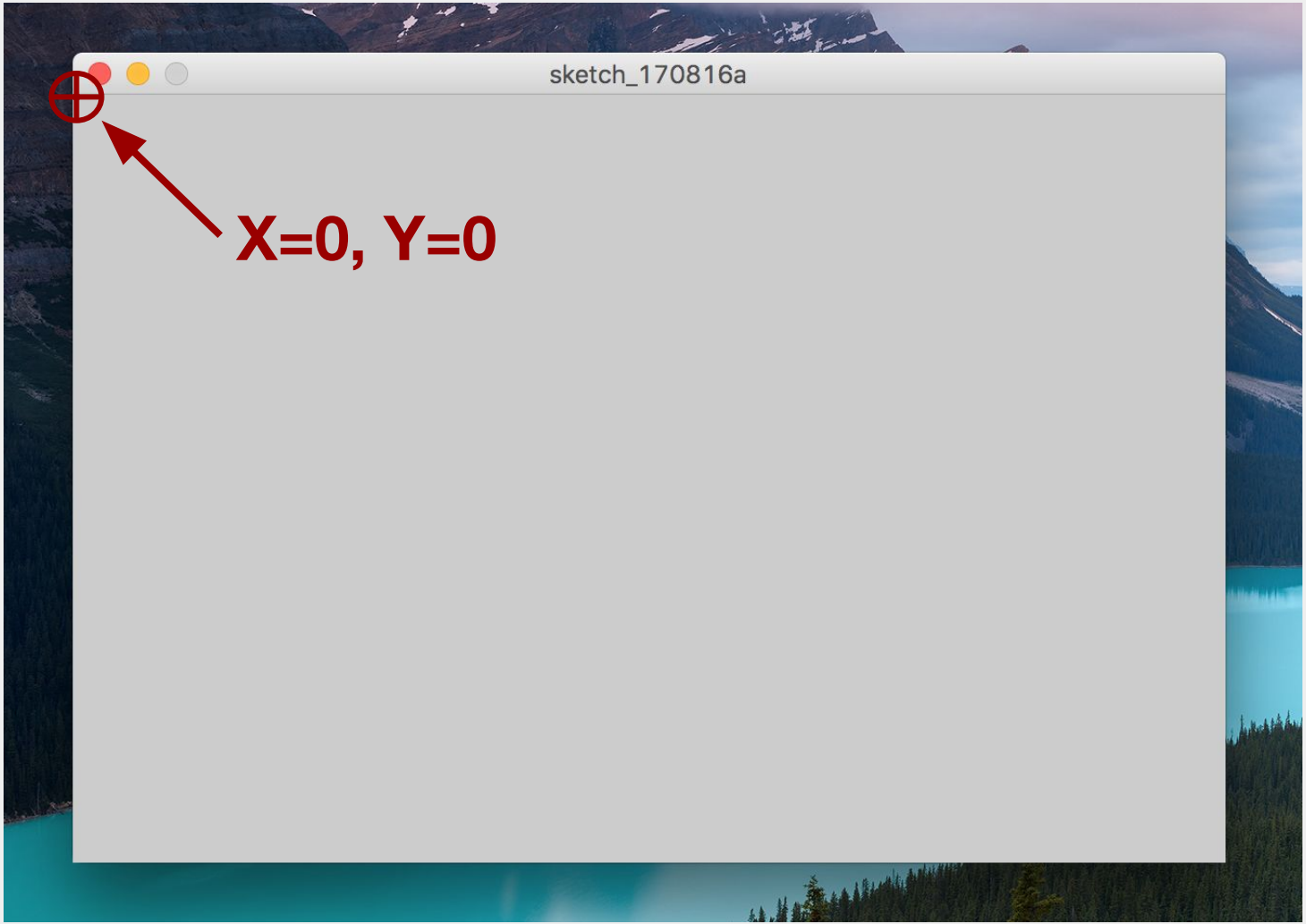


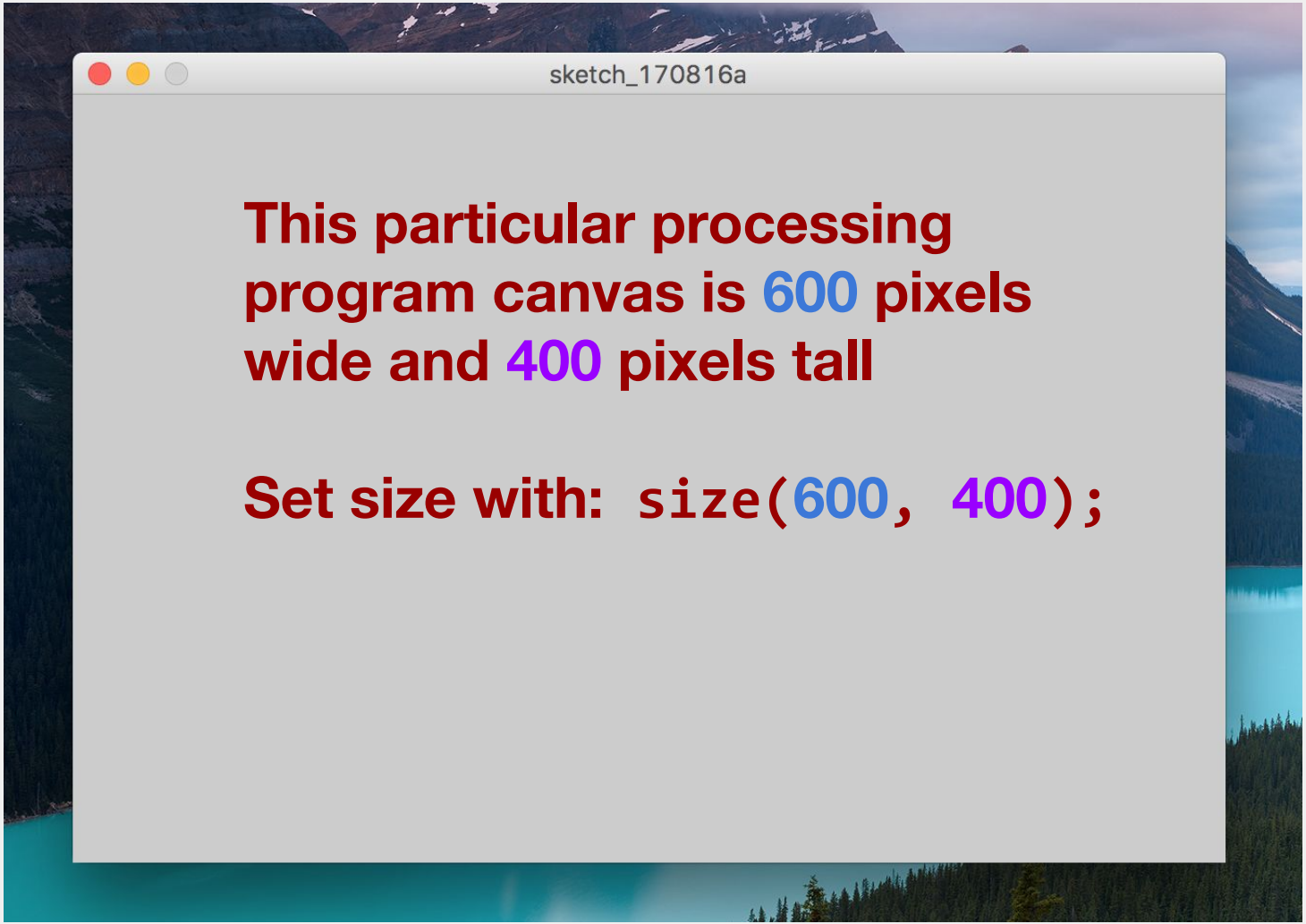

sketch_170816a
Left/Right is the X axis

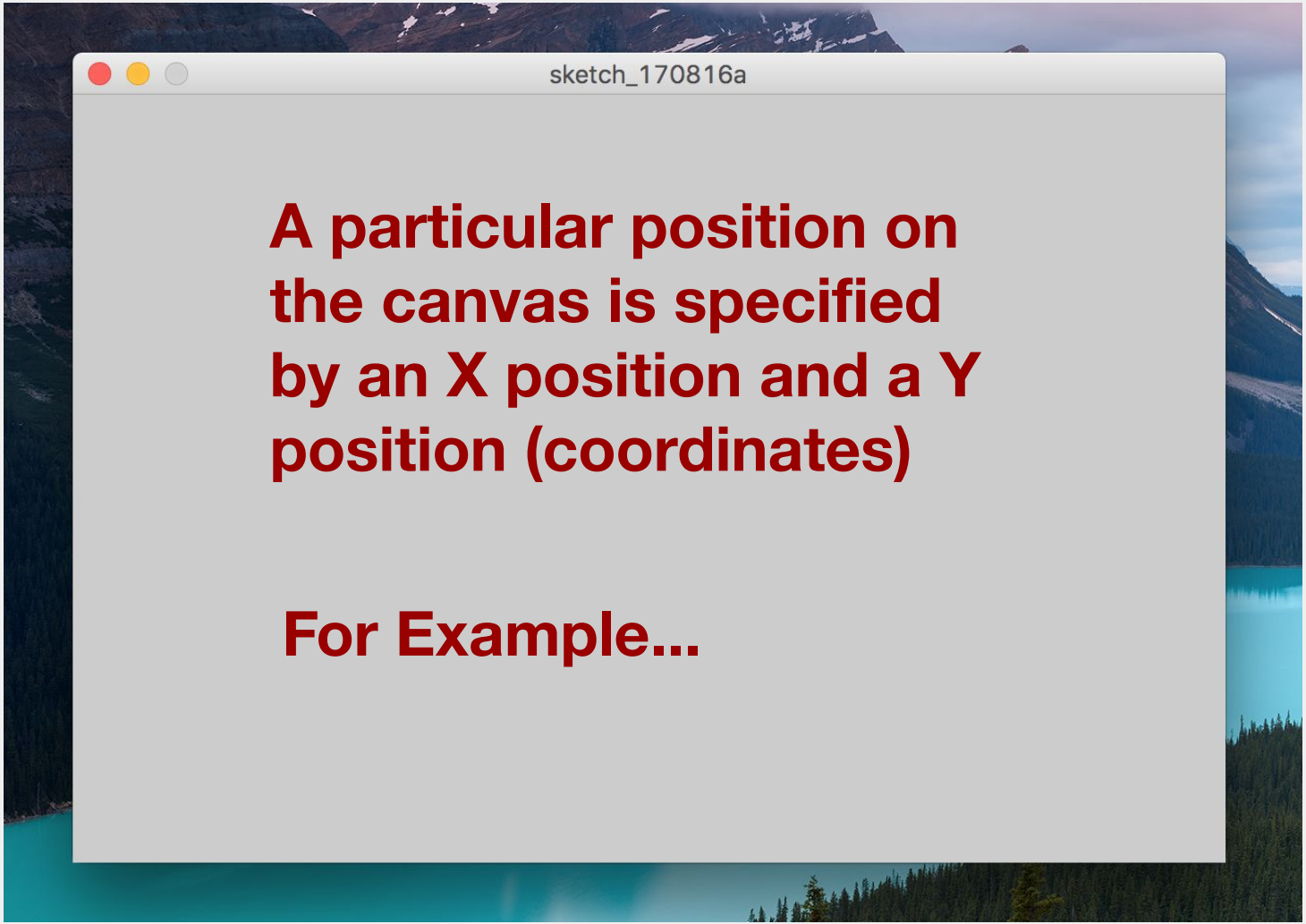

sketch_170816a
Up/Down is the Y axis

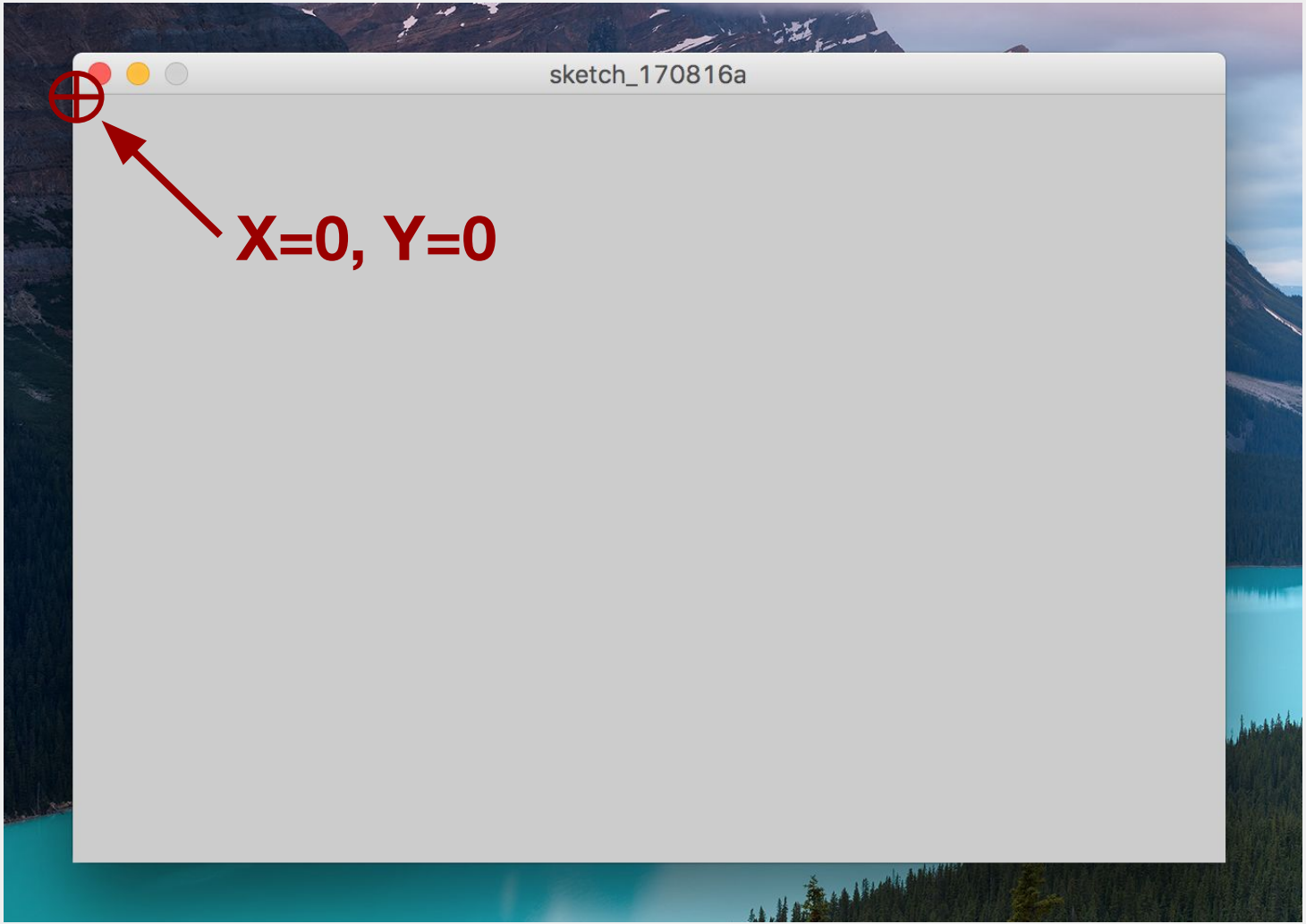

sketch_170816a
X-axis
positive
Y-axis
positive

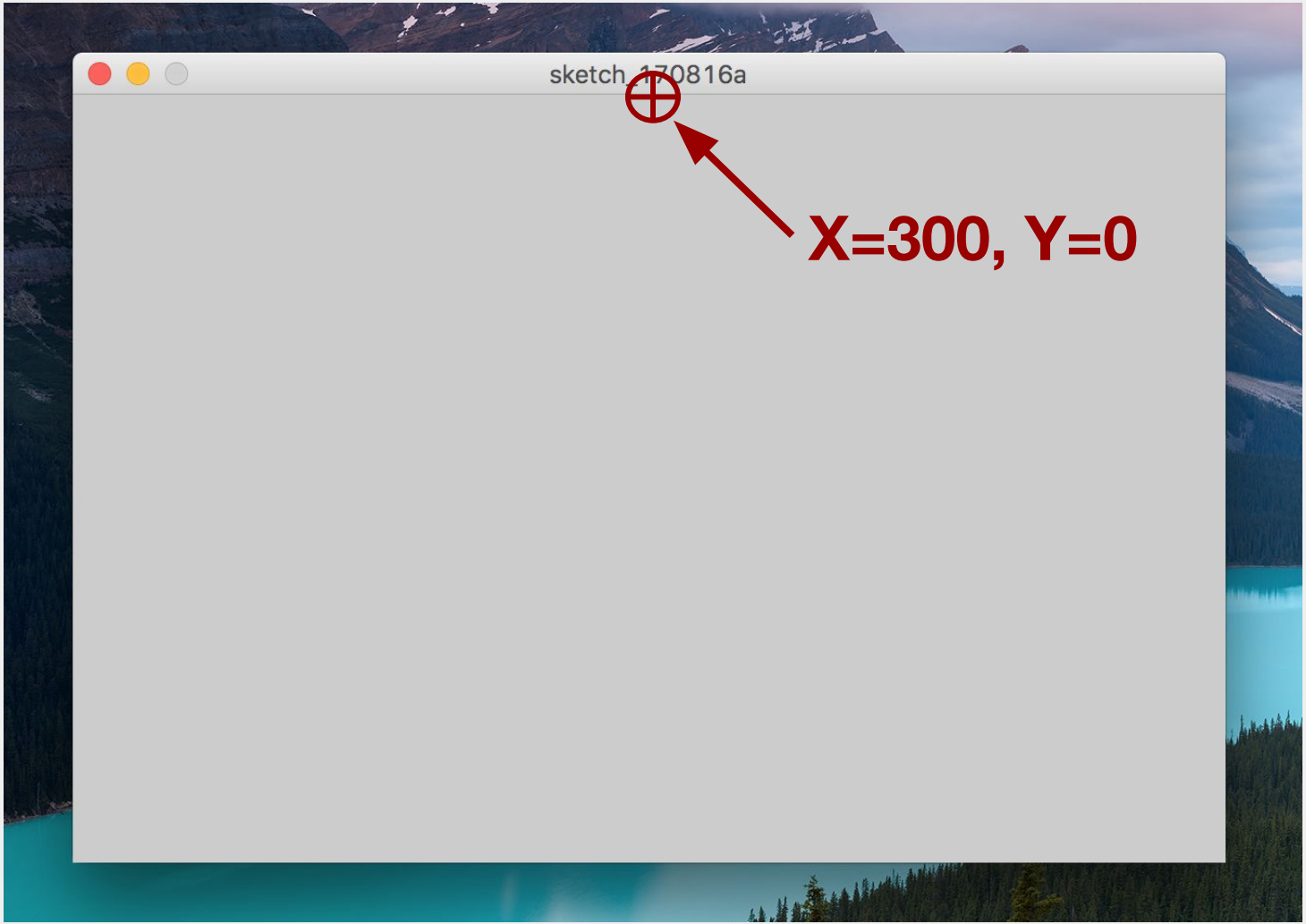

sketch_170816a
A particular position on the canvas is specified by an X position and a Y position (coordinates)
(0,0) is in the upper left corner!

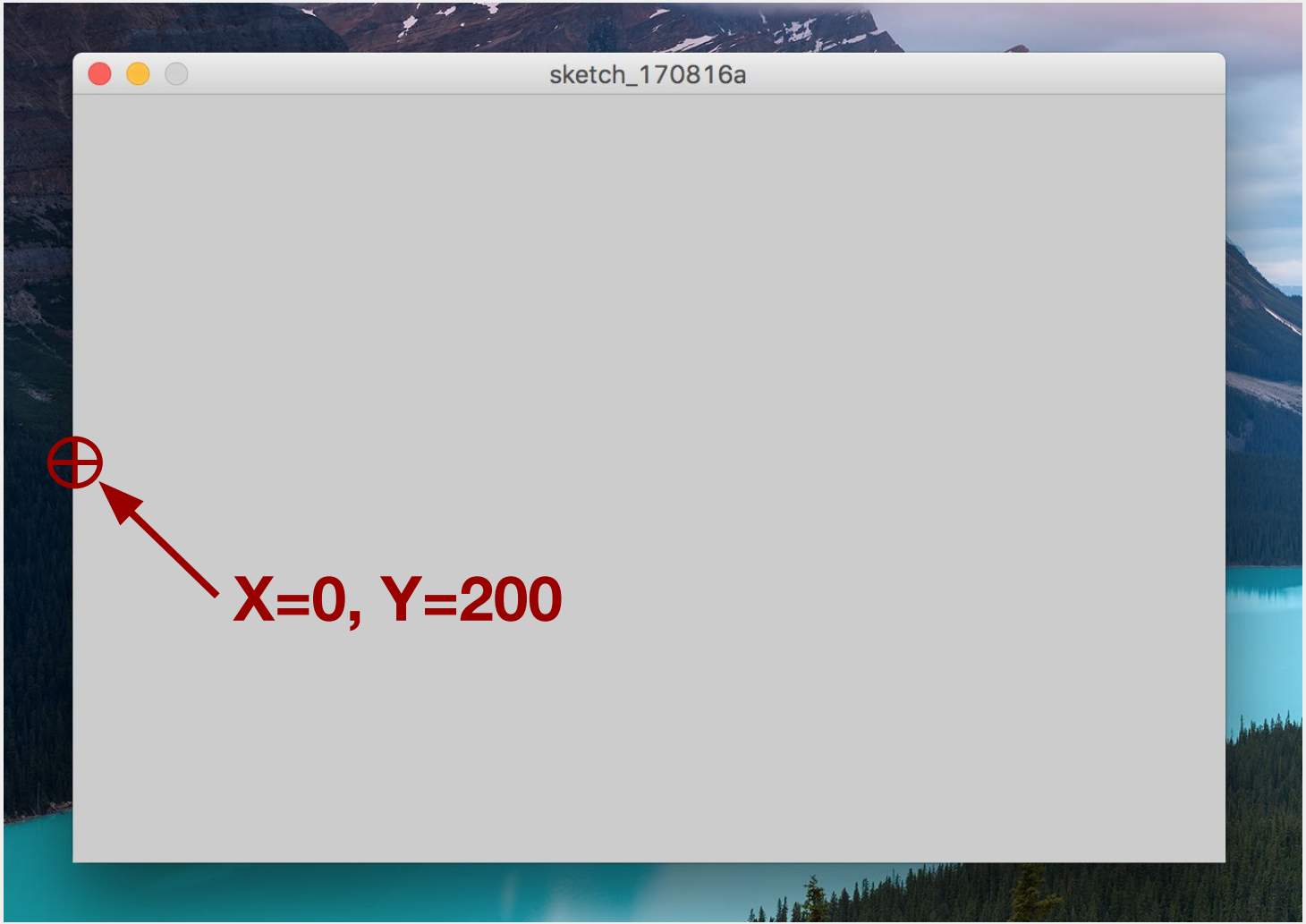

sketch_170816a
X=0, Y=0

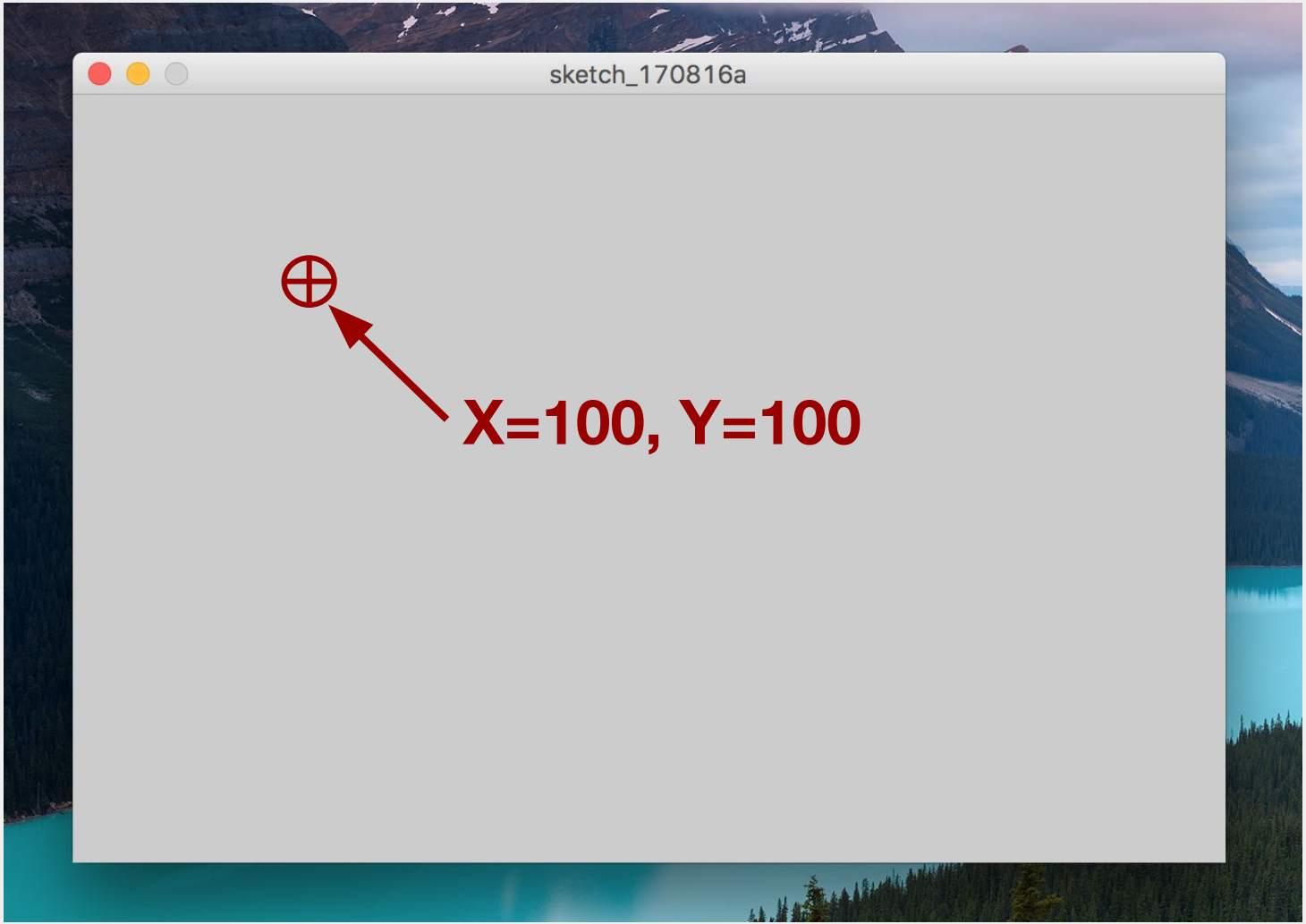

sketch_170816a
This particular processing program canvas is 600 pixels wide and 400 pixels tall
Set size with: size(600, 400);

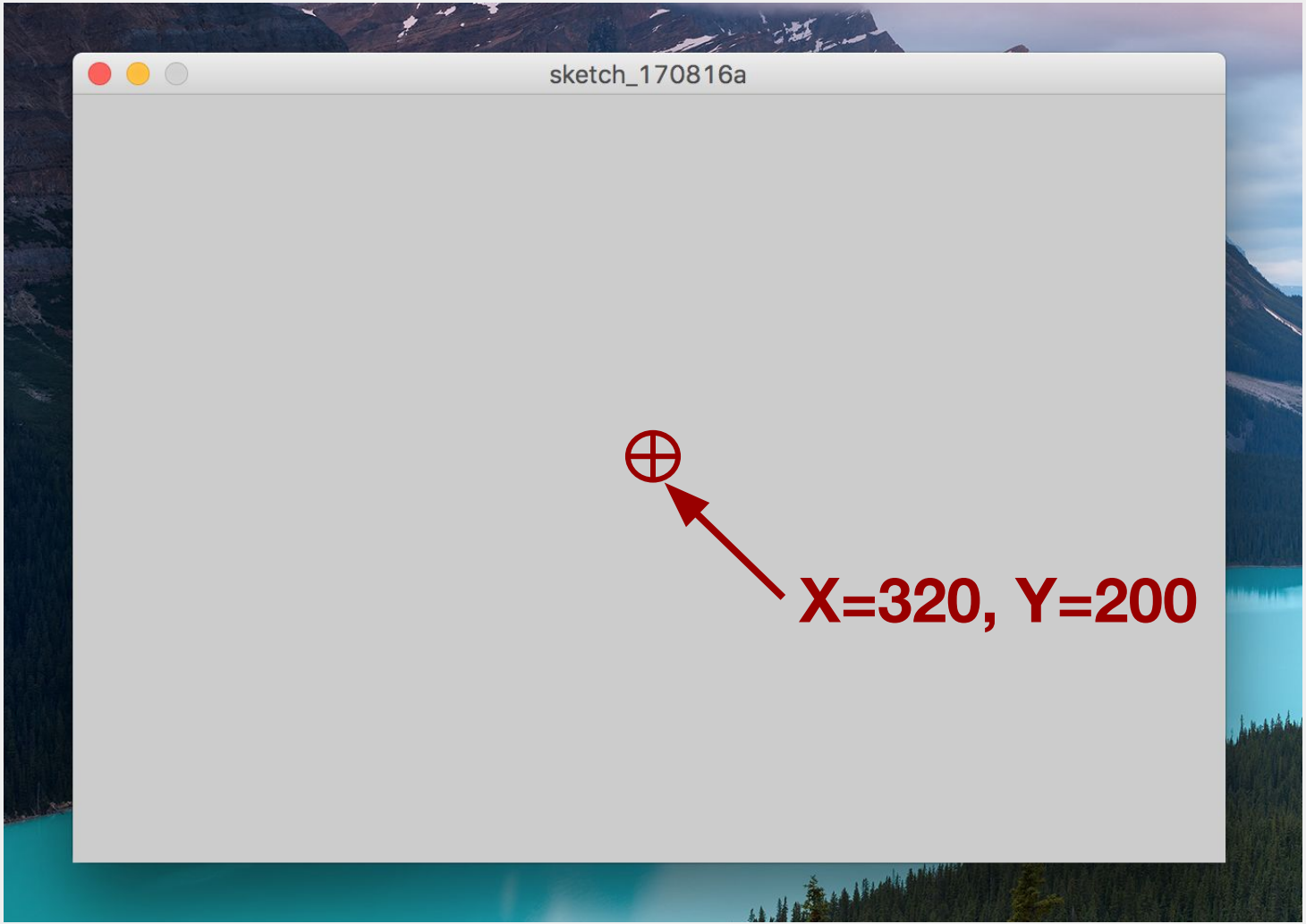

sketch_170816a
A particular position on the canvas is specified by an X position and a Y position (coordinates)
For Example...

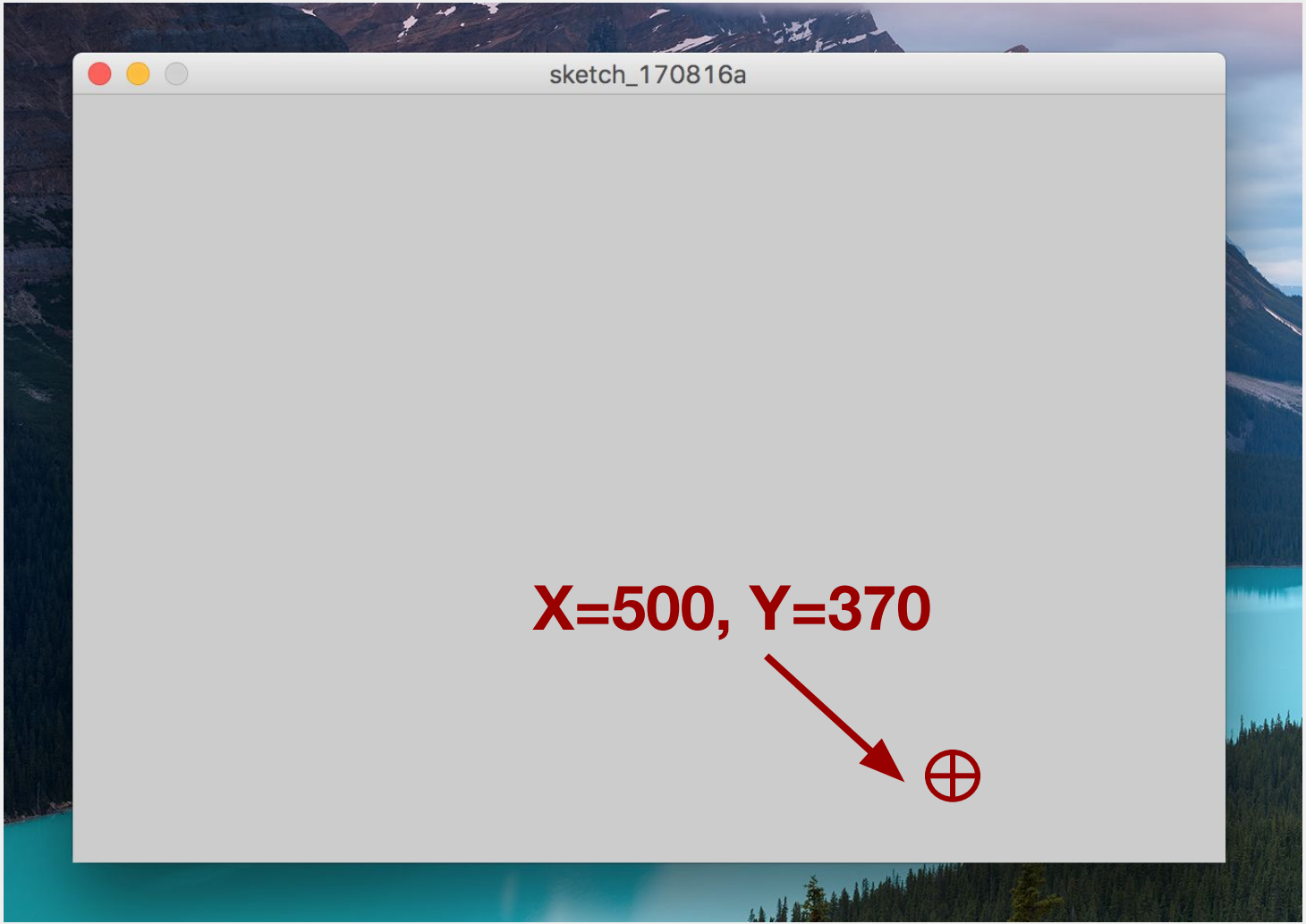

sketch_170816a
X=0, Y=0


sketch_170816a
X=300, Y=0


sketch_170816a
X=0, Y=200

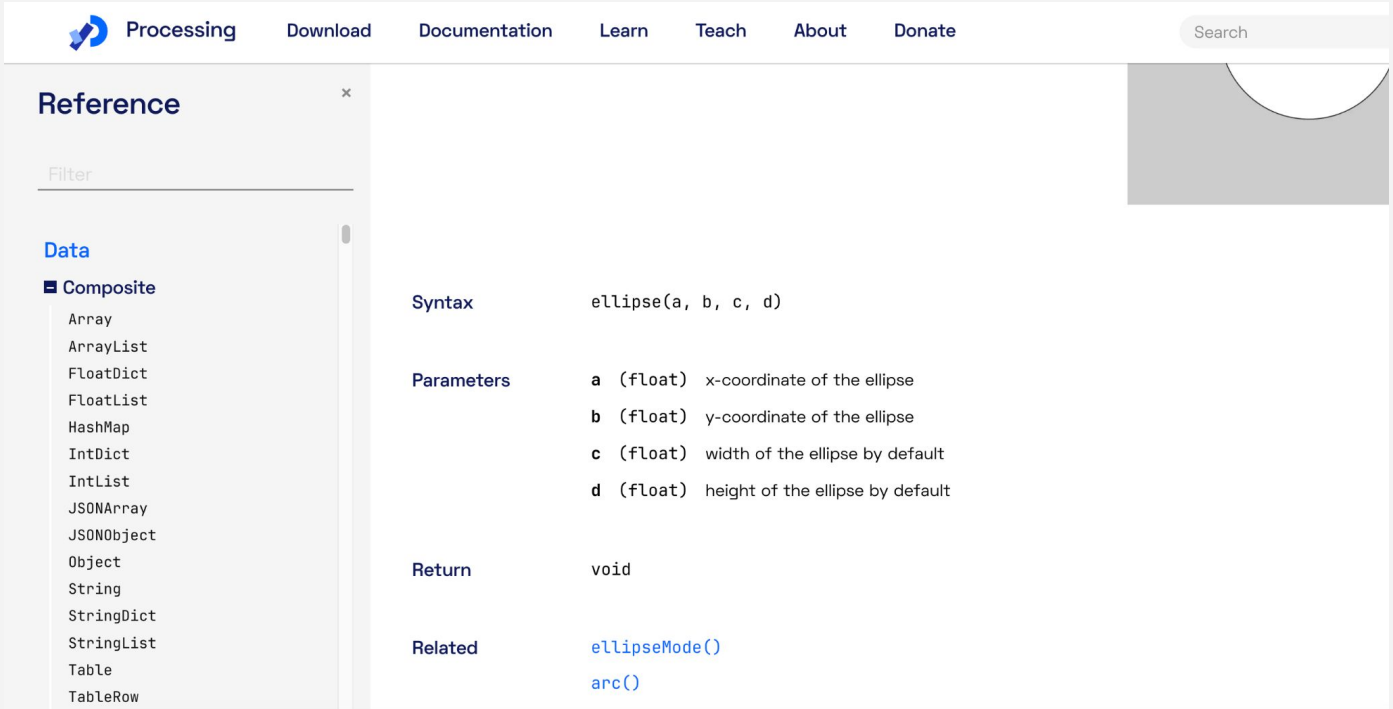

sketch_170816a
X=100, Y=100


sketch_170816a
X=320, Y=200


sketch_170816a
X=500, Y=370

Always read the reference!

In Processing go to **Help** -> **Reference**



Demo

- Using the reference, Let’s say I want to
  - Draw a point in the middle of the canvas
  - Draw a circle
  - Color the circle

Problem 2

- Using Processing
- Setup a 400 by 400 canvas
- Draw a 20 x 20 square in the lower right corner of the canvas
- Upload the screenshot to Gradescope

1 minute for individual (silent) work

4 minutes for group work

Functions

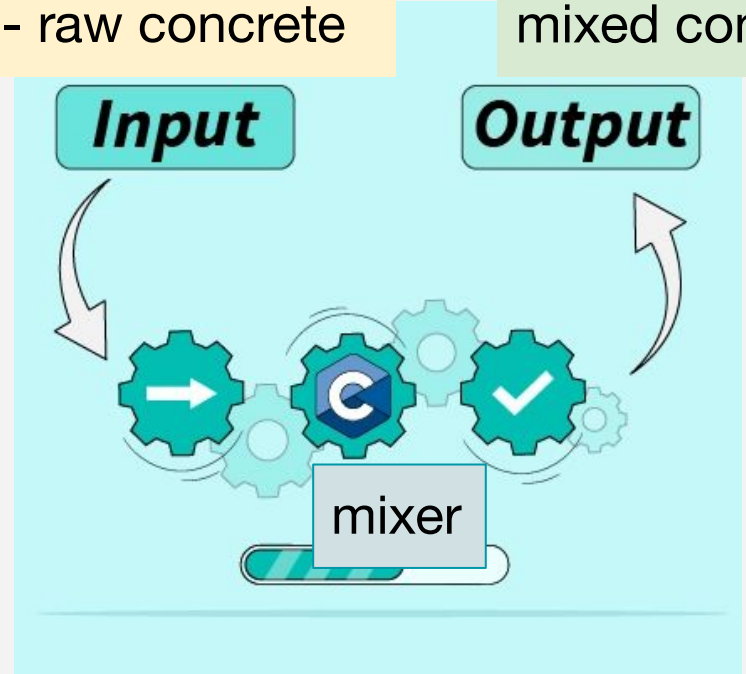
- As a programmer, you tell the Processing language what, where, and how to draw things by calling **functions**
- A **function** is a sequence of code that can be “called” or “invoked” by calling it
- In fact, we’ve already called a few functions

Functions

- When you call a function, you must give the function 0 or more arguments
  - A argument is a bit of information that you can give the function to control what it does
  - The order that you write argument in matters!
  - Each of the functions you've used take a few arguments

Function (method) = actions or verbs

- water
- raw concrete

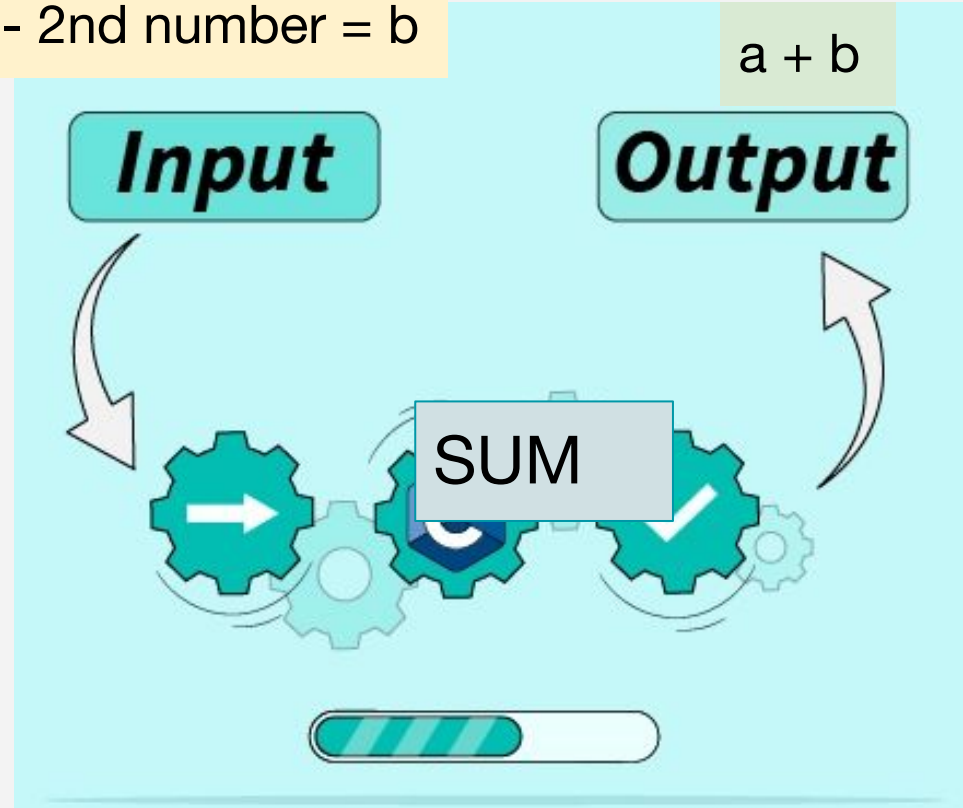


mixed concrete

1) Pour/spray 0.75 gallons of water into
2) Cut open 1 bag of concrete
3) Pour bag concrete into mixer
4) Turn on mixer
5) If too stiff
    a) Add more water
6) If too watery
    a) Add more concrete mix
7) Pour concrete out of mixer

Function (method) = actions or verbs

- 1st number = a
- 2nd number = b

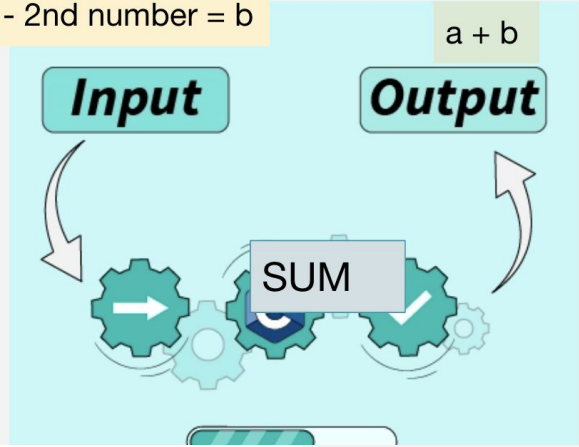


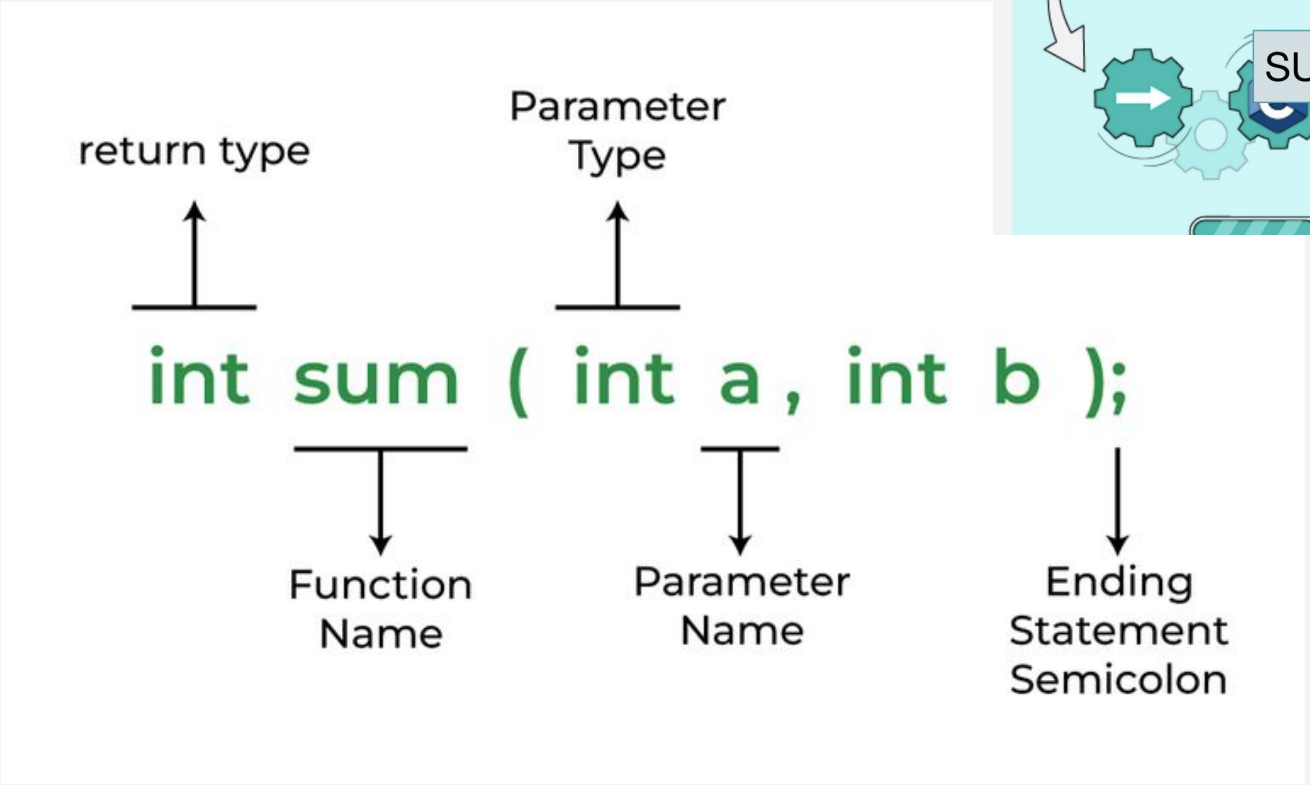
Sum two numbers

1) Turn on adder
2) Choose first number
3) Choose second number
4) Compute the sum

Functions

- 1st number = a
- 2nd number = b

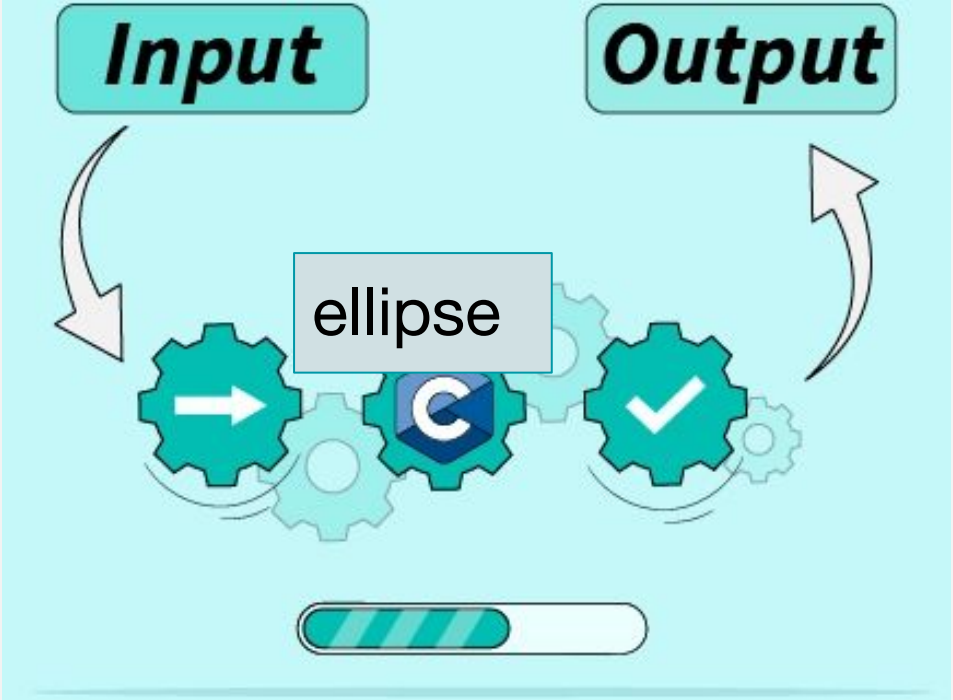




Function (method) = actions or verbs

X,y, width, height

the drawing of the ellipse



the “ellipse” function
draws an ellipse

Functions on Processing

- `ellipse(x, y, w, h)` - A call to a function that draws an ellipse at the x/y coordinate and width/height provided
- `size(w, h)` - A call to a function that sets the size of the processing drawing canvas
- `rect(x, y, w, h)` - A call to a function that draws a rectangle at the x/y coordinate and width/height provided
- . . . and more!

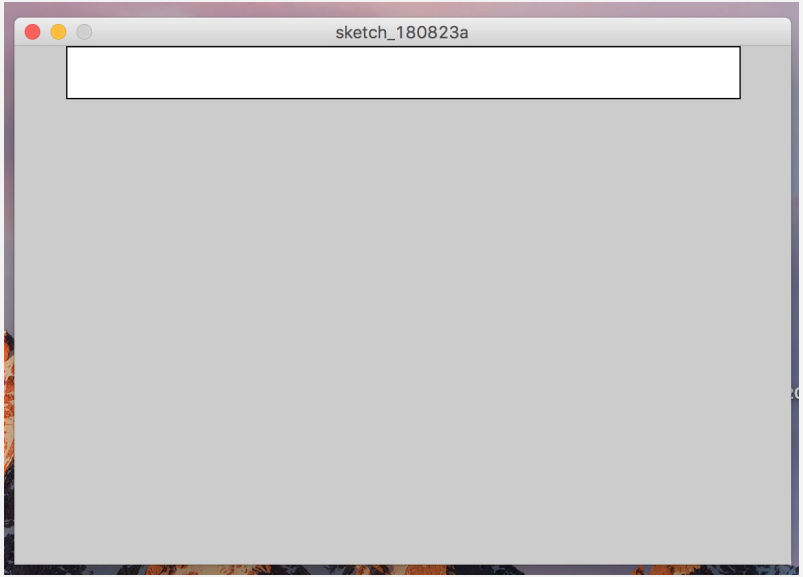
Processing

- We can draw other shapes too:

```
rect(x, y, w, h);

triangle(x1, y1, x2, y2, x3, y3);

line(x1, y1, x2, y2);

point(x, y);
```

Drawing a simple canvas

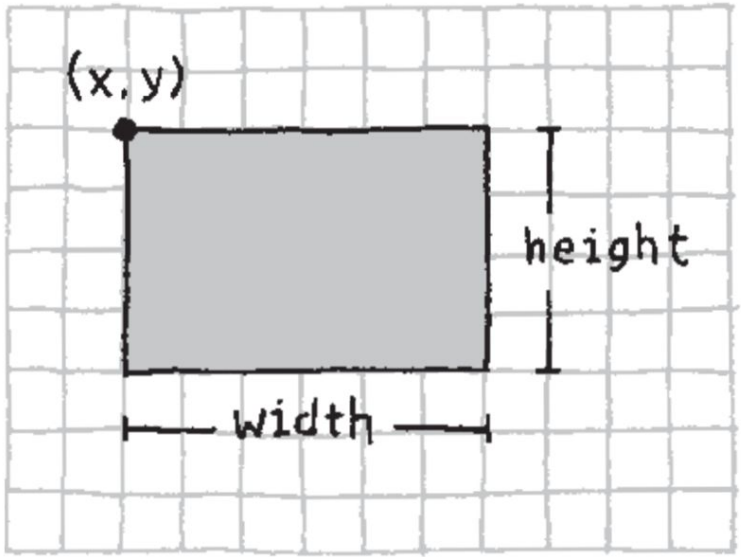
Let's say we want to draw a small Processing program that creates the following canvas



Remember:

- `size(width, height)`
- `rect(x, y, w, h);`
- But where does x,y start?

Functions



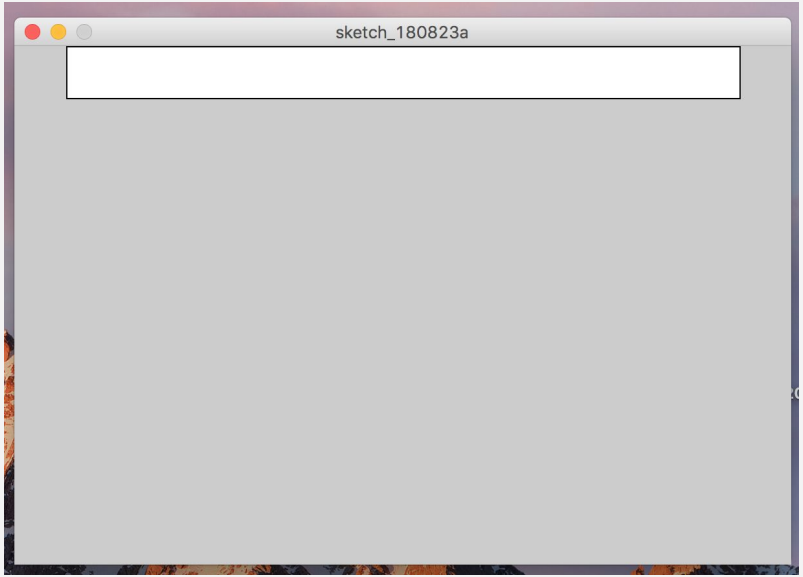

```
rect(x, y, width, height)
```

Drawing a simple canvas

Write a small processing program that creates the following canvas

Remember:

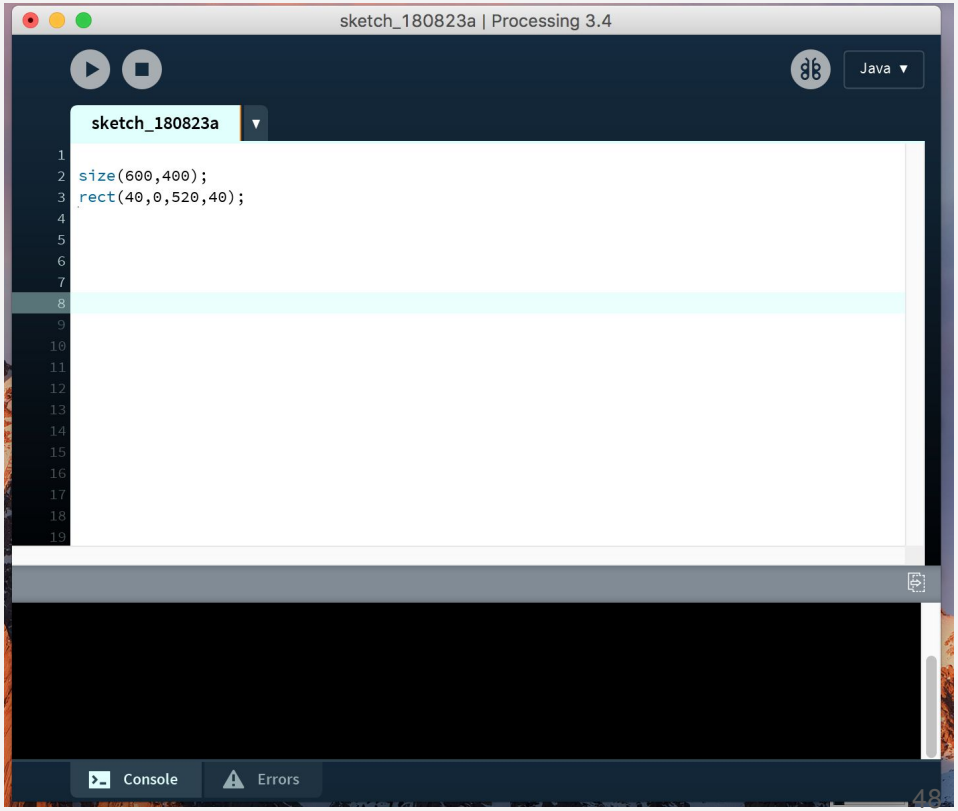
- `size(width, height)`
- `rect(x, y, w, h);`
- We'll use a 600 x 400 canvas



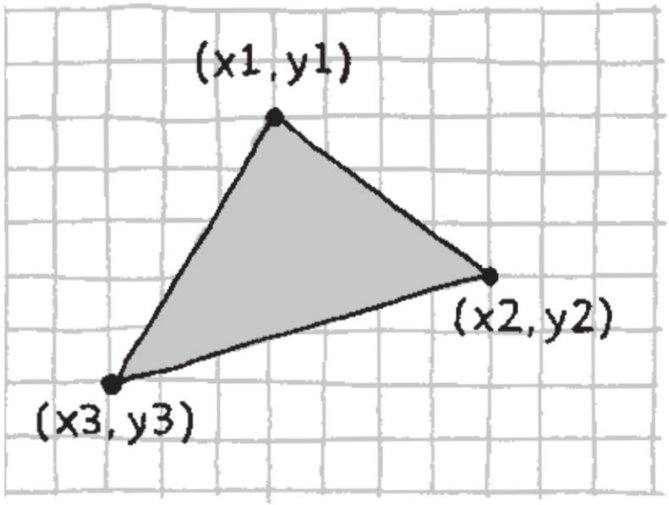
Drawing a simple canvas

Solution:

```
size(600,400);
rect(40,0,520,40);
```

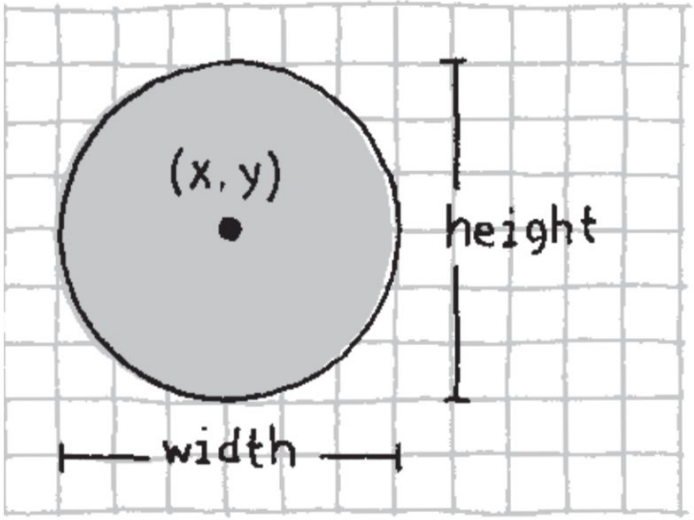


Functions




```
triangle(x1, y1, x2, y2, x3, y3)
```

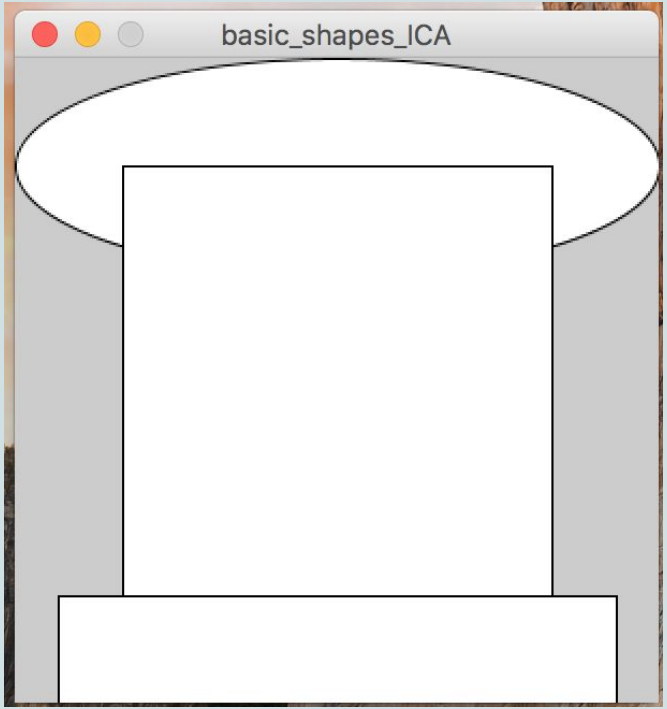
Functions




```
ellipse(x, y, width, height)
```

Problem 3

Write a Processing program that creates the following canvas

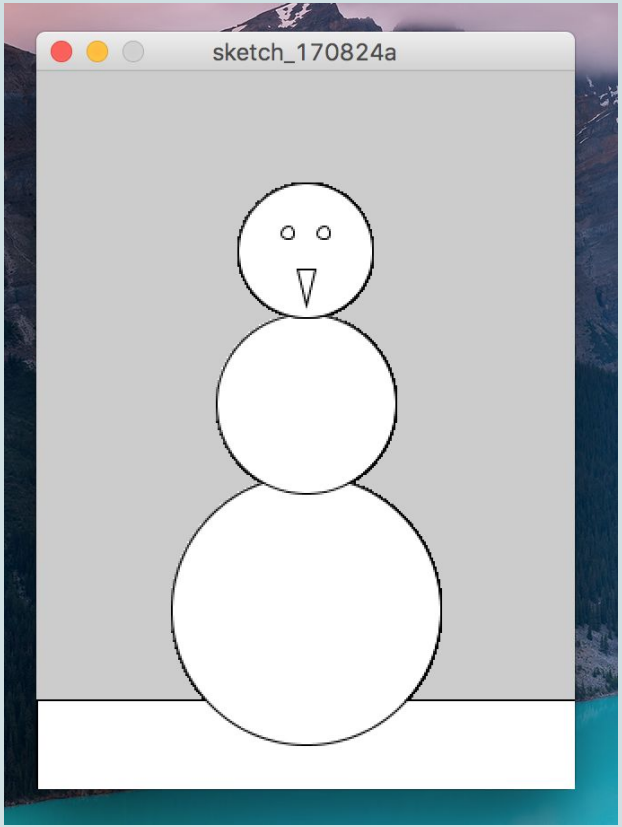


Remember:

- `size(width, height)`
- `ellipse(x, y, w, h);`
- `rect(x, y, w, h);`

Problem 4

- Goal: Draw an simple snowman like the one to the right
- How can this be done, using what we know about Processing so far?



Problem 4

- Goal: Draw an simple snowman like the one to the right
- How can this be done, using what we know about Processing so far?
- What can we add to it to make it look better?