

# User Interface Software Projects: Intro to Processing

Assoc. Professor Donald J. Patterson  
INF 134 Winter 2013



# Intro to Processing

<http://processing.org/>

# Intro to Processing



<http://processing.org/>

- What the heck is Processing?
  - A **programming language**
  - An **environment for running the programs**



<http://processing.org/>

- What the heck is Processing?
  - A **programming language**
  - An **environment for running the programs**
- What is it for?
  - It is for people who want to create
    - **images**
    - **animations**
    - **interactions**



<http://processing.org/>

- What the heck is Processing?
  - A **programming language**
  - An **environment for running the programs**
- What is it for?
  - It is for people who want to create
    - **images**
    - **animations**
    - **interactions**
- Who is it for?
  - **students**
  - **artists**
  - **designers**
  - **researchers**
  - **hobbyists**



<http://processing.org/>

# Intro to Processing



<http://processing.org/>



- Free to download

<http://processing.org/>





- Free to download
- Open source

<http://processing.org/>



- Free to download
- Open source
- Programs output in 2D, 3D or pdf

<http://processing.org/>



- Free to download
- Open source
- Programs output in 2D, 3D or pdf
- For Windows, Mac, Linux

<http://processing.org/>



- Free to download
- Open source
- Programs output in 2D, 3D or pdf
- For Windows, Mac, Linux
- Programs can be put in web pages

<http://processing.org/>



- Free to download
- Open source
- Programs output in 2D, 3D or pdf
- For Windows, Mac, Linux
- Programs can be put in web pages
- Programs can be run as applications

<http://processing.org/>



- Free to download
- Open source
- Programs output in 2D, 3D or pdf
- For Windows, Mac, Linux
- Programs can be put in web pages
- Programs can be run as applications
- Lots of documentation and books available

<http://processing.org/>

# Intro to Processing

<http://processing.org/>

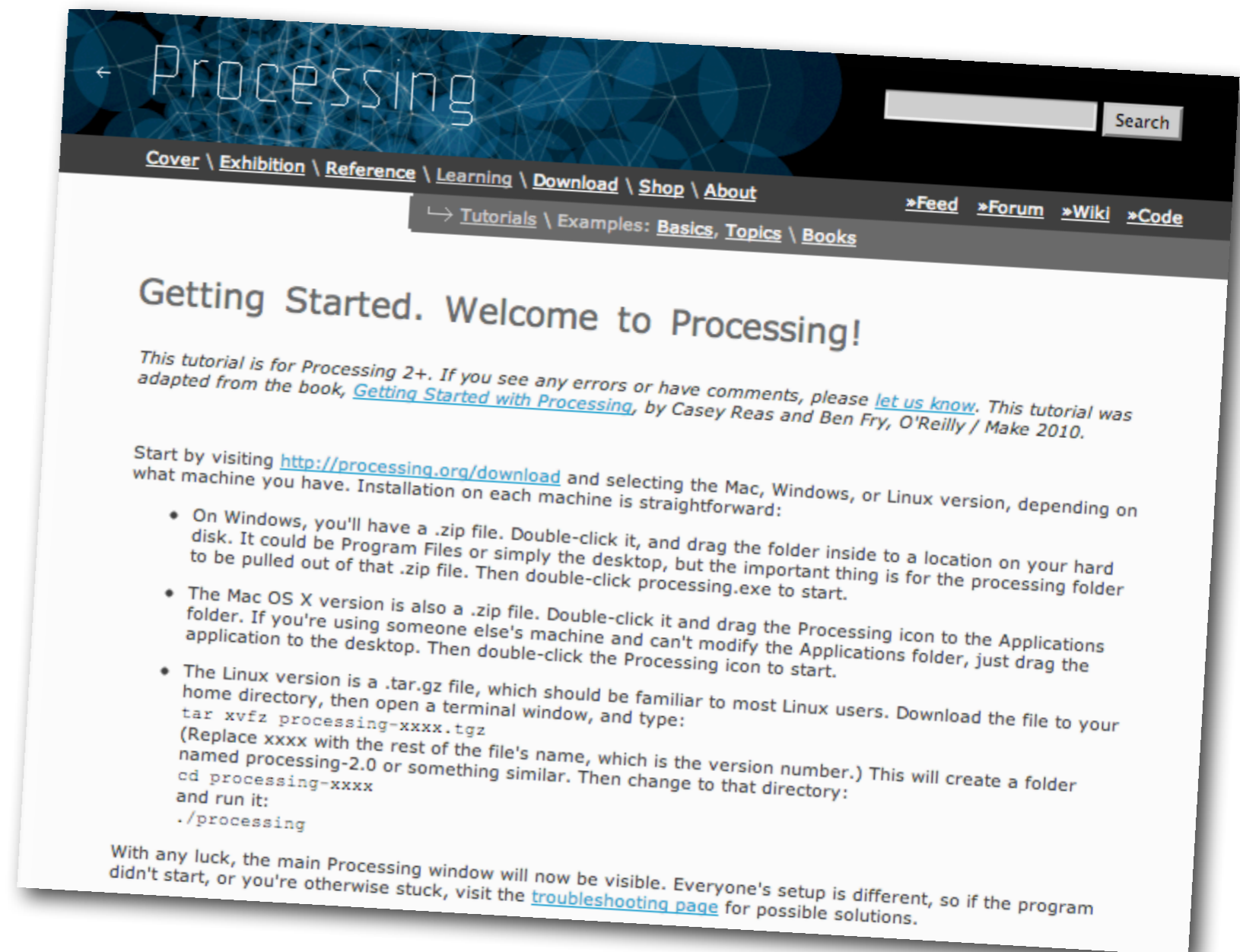
# Intro to Processing

- Optional Warm-up #1
  - Complete the lab:
    - “Getting Started. Welcome to Processing”

<http://processing.org/>



- Optional Warm-up #1
  - Complete the lab:
    - “Getting Started. Welcome to Processing”



<http://processing.org/>

# Intro to Processing

<http://processing.org/>

# Intro to Processing

- Download the software v 2.0 beta 6



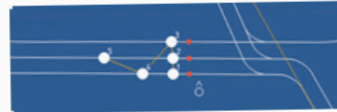

<http://processing.org/>

# Intro to Processing

- Download the software v 2.0 beta 6

The screenshot shows the Processing.org website homepage. At the top, the word "Processing" is displayed in a large, white, sans-serif font against a dark blue background with a network-like pattern. To the right of the logo is a search bar with the word "Search" inside. Below the logo, a navigation bar contains links: "Cover", "Exhibition", "Reference", "Learning", "Download", "Shop", "About", "»Feed", "»Forum", "»Wiki", and "»Code".

The main content area is divided into two columns. The left column features four exhibition entries, each with a small image and a title:

- » Exhibition**
  -   
[Digital Natives and Glitched Realities](#)  
by Matthew Plummer-Fernandez
  -   
[Stone Spray](#)  
by Petr Novikov, Inder Shergill and Anna Kulik
  -   
[City Symphonies](#)  
by Mark McKeague
  -   
[Silenc](#)  
by Manas Karambelkar, Momo Miyazaki and Kenneth A. Robertsen

The right column features a section titled "» Download Processing" with sub-links: "» Play with Examples" and "» Browse Tutorials". Below these links is a paragraph of text:

Processing is an open source programming language and environment for people who want to create images, animations, and interactions. Initially developed to serve as a software sketchbook and to teach fundamentals of computer programming within a visual context, Processing also has evolved into a tool for generating finished professional work. Today, there are tens of thousands of students, artists, designers, researchers, and hobbyists who use Processing for learning, prototyping, and production.

- » Free to download and open source
- » Interactive programs using 2D, 3D or PDF output
- » OpenGL integration for accelerated 3D
- » For GNU/Linux, Mac OS X, and Windows
- » Projects run online or as double-clickable applications
- » Over 100 libraries extend the software into sound, video, computer vision, and more...
- » Well [documented](#), with many [books](#) available

To see more of what people are doing with Processing, check out these sites:

- » [Processing Wiki](#)
- » [Processing Discussion Forum](#)
- » [OpenProcessing](#)
- » [Studio Sketchpad](#)
- » [CreativeApplications.Net](#)
- » [Vimeo](#)
- » [Flickr](#)

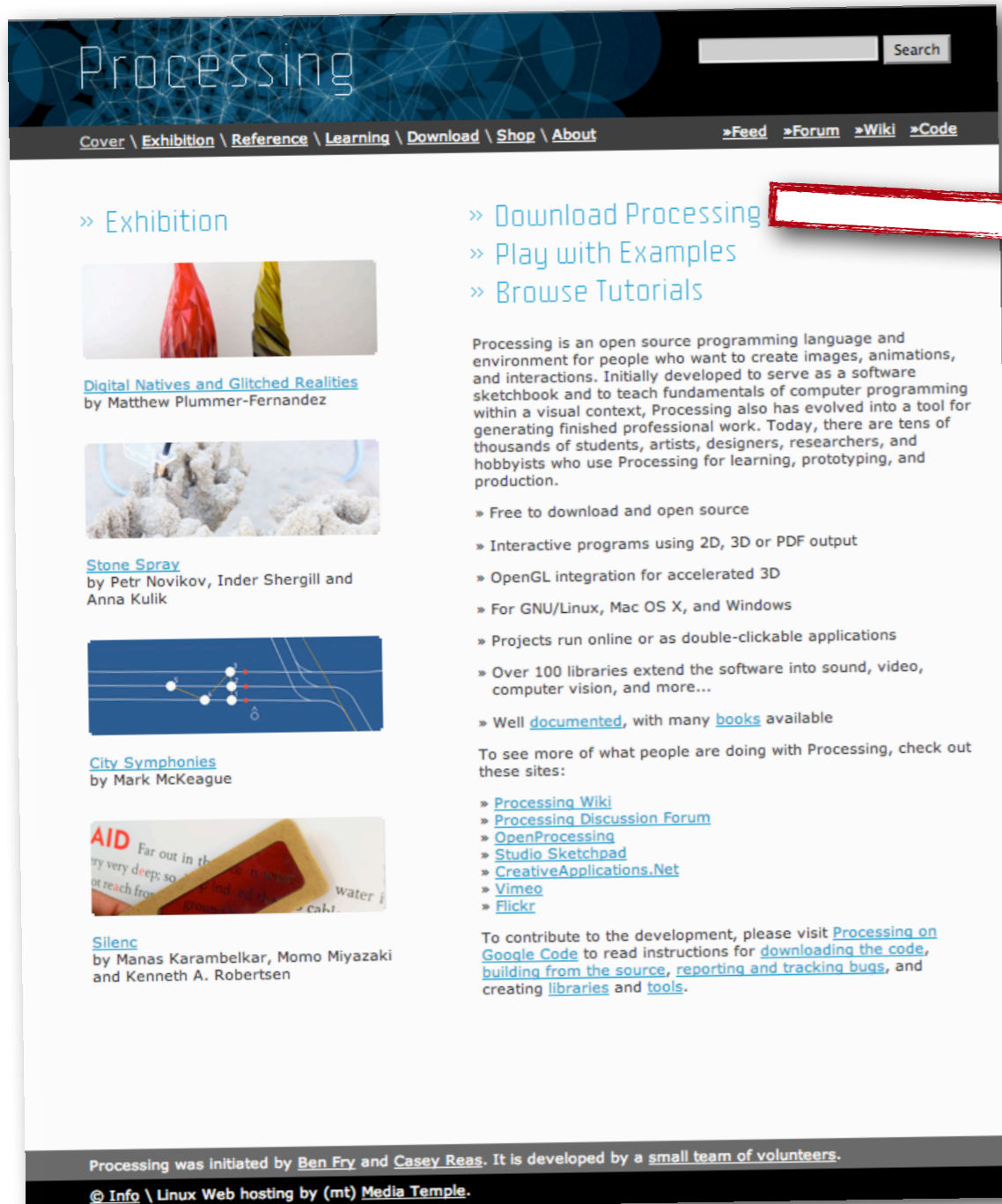
To contribute to the development, please visit [Processing on Google Code](#) to read instructions for [downloading the code](#), [building from the source](#), [reporting and tracking bugs](#), and creating [libraries](#) and [tools](#).

At the bottom of the page, a dark grey footer contains the text: "Processing was initiated by [Ben Fry](#) and [Casey Reas](#). It is developed by a [small team of volunteers](#)." Below this is a copyright notice: "© [Info](#) \ Linux Web hosting by (mt) [Media Temple](#)."

<http://processing.org/>

# Intro to Processing

- Download the software v 2.0 beta 6



Processing

Cover \ Exhibition \ Reference \ Learning \ Download \ Shop \ About

»Feed »Forum »Wiki »Code

» Exhibition

[Digital Natives and Glitched Realities](#)  
by Matthew Plummer-Fernandez

[Stone Spray](#)  
by Petr Novikov, Inder Shergill and Anna Kulik

[City Symphonies](#)  
by Mark McKeague

[Silenc](#)  
by Manas Karambelkar, Momo Miyazaki and Kenneth A. Robertsen

» Download Processing  
» Play with Examples  
» Browse Tutorials

Processing is an open source programming language and environment for people who want to create images, animations, and interactions. Initially developed to serve as a software sketchbook and to teach fundamentals of computer programming within a visual context, Processing also has evolved into a tool for generating finished professional work. Today, there are tens of thousands of students, artists, designers, researchers, and hobbyists who use Processing for learning, prototyping, and production.

- » Free to download and open source
- » Interactive programs using 2D, 3D or PDF output
- » OpenGL integration for accelerated 3D
- » For GNU/Linux, Mac OS X, and Windows
- » Projects run online or as double-clickable applications
- » Over 100 libraries extend the software into sound, video, computer vision, and more...
- » Well [documented](#), with many [books](#) available

To see more of what people are doing with Processing, check out these sites:

- » [Processing Wiki](#)
- » [Processing Discussion Forum](#)
- » [OpenProcessing](#)
- » [Studio Sketchpad](#)
- » [CreativeApplications.Net](#)
- » [Vimeo](#)
- » [Flickr](#)

To contribute to the development, please visit [Processing on Google Code](#) to read instructions for [downloading the code](#), [building from the source](#), [reporting and tracking bugs](#), and creating [libraries](#) and [tools](#).

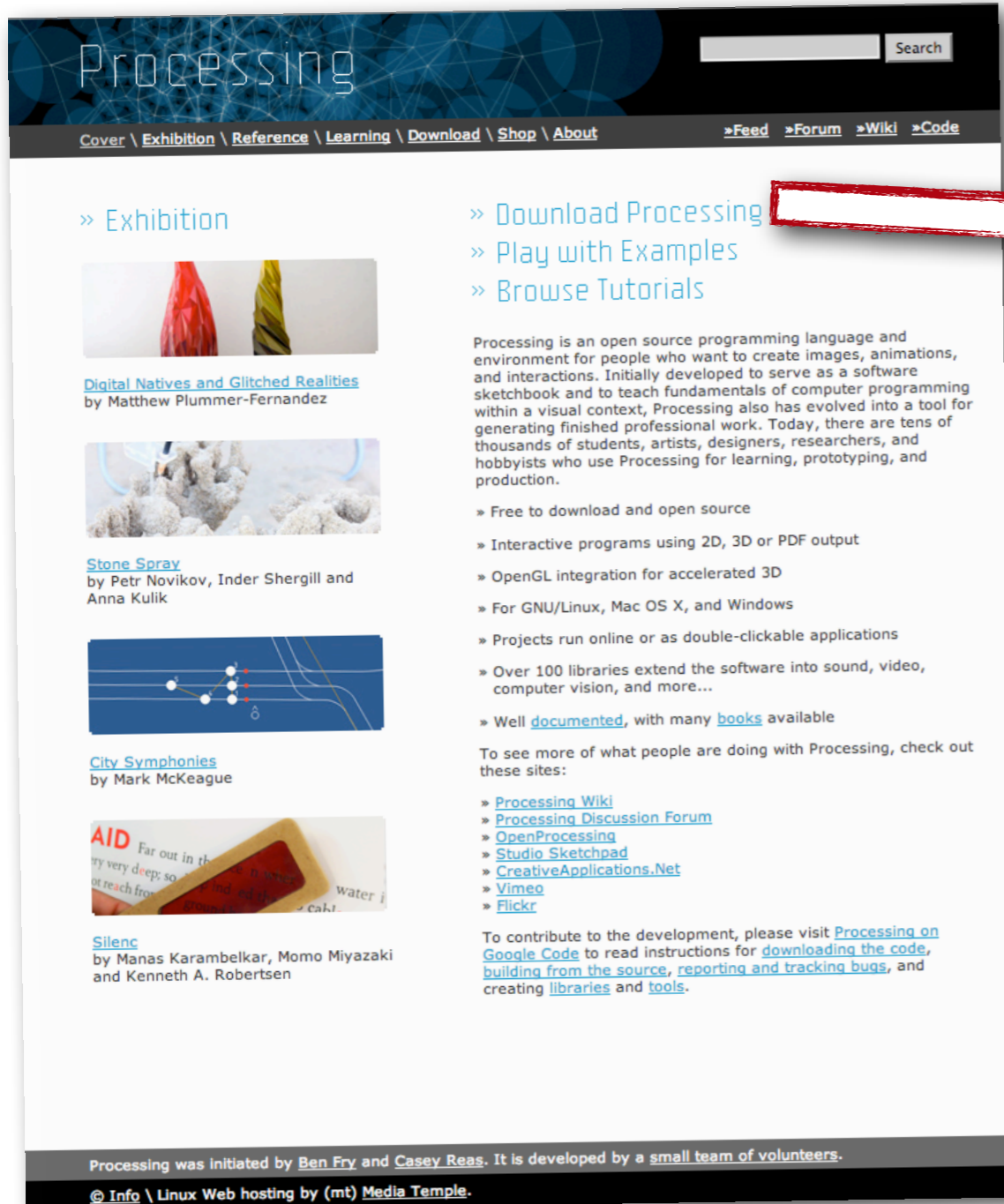
Processing was initiated by [Ben Fry](#) and [Casey Reas](#). It is developed by a [small team of volunteers](#).

© [Info](#) \ Linux Web hosting by (mt) [Media Temple](#).

<http://processing.org/>

# Intro to Processing

- Download the software v 2.0 beta 6



Processing

Cover \ Exhibition \ Reference \ Learning \ Download \ Shop \ About

» Feed » Forum » Wiki » Code

» Exhibition

[Digital Natives and Glitched Realities](#)  
by Matthew Plummer-Fernandez

[Stone Spray](#)  
by Petr Novikov, Inder Shergill and Anna Kulik

[City Symphonies](#)  
by Mark McKeague

[AID](#)  
by Manas Karambelkar, Momo Miyazaki and Kenneth A. Robertsen

» Download Processing  
» Play with Examples  
» Browse Tutorials

Processing is an open source programming language and environment for people who want to create images, animations, and interactions. Initially developed to serve as a software sketchbook and to teach fundamentals of computer programming within a visual context, Processing also has evolved into a tool for generating finished professional work. Today, there are tens of thousands of students, artists, designers, researchers, and hobbyists who use Processing for learning, prototyping, and production.

- » Free to download and open source
- » Interactive programs using 2D, 3D or PDF output
- » OpenGL integration for accelerated 3D
- » For GNU/Linux, Mac OS X, and Windows
- » Projects run online or as double-clickable applications
- » Over 100 libraries extend the software into sound, video, computer vision, and more...
- » Well [documented](#), with many [books](#) available

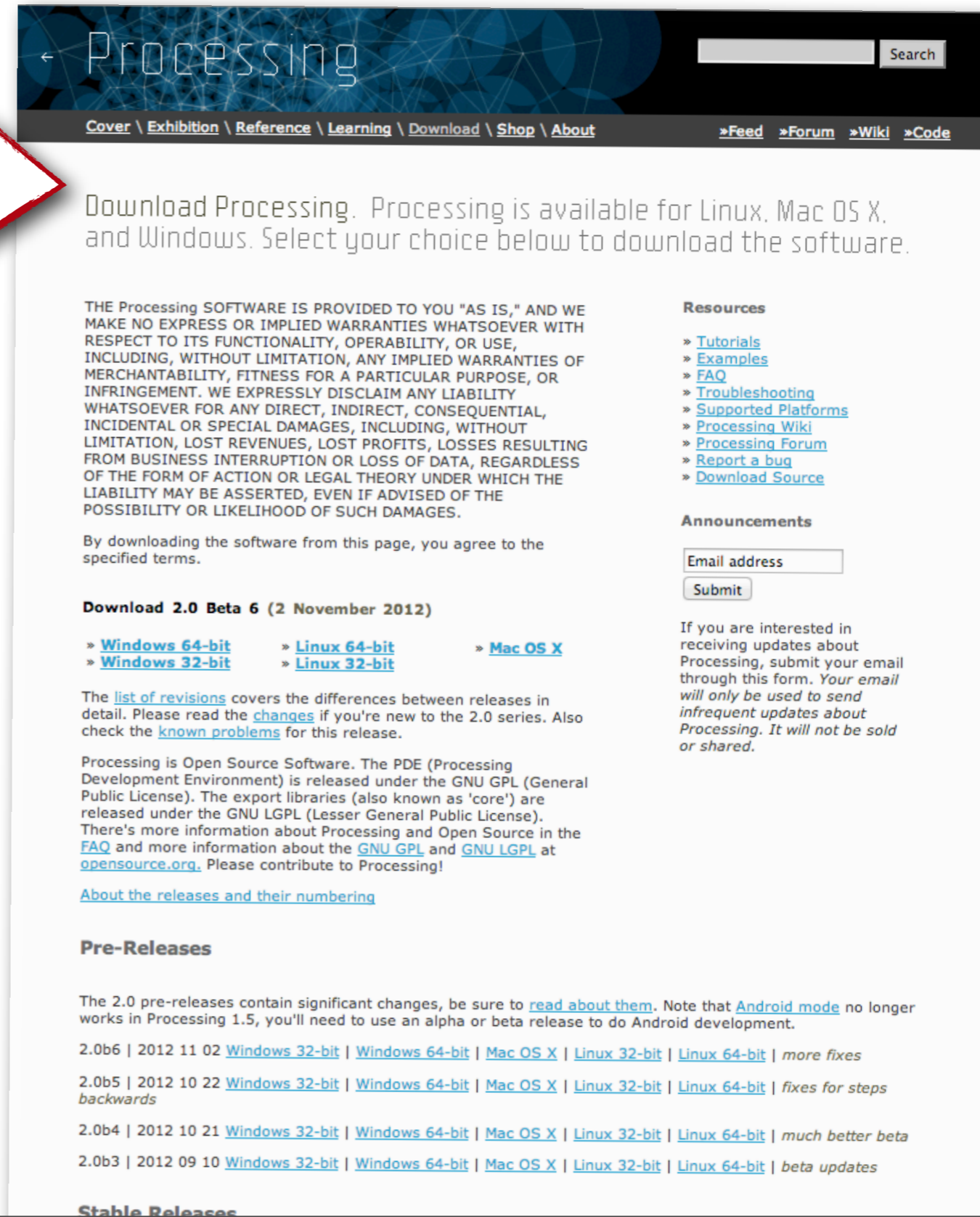
To see more of what people are doing with Processing, check out these sites:

- » [Processing Wiki](#)
- » [Processing Discussion Forum](#)
- » [OpenProcessing](#)
- » [Studio Sketchpad](#)
- » [CreativeApplications.Net](#)
- » [Vimeo](#)
- » [Flickr](#)

To contribute to the development, please visit [Processing on Google Code](#) to read instructions for [downloading the code](#), [building from the source](#), [reporting and tracking bugs](#), and creating [libraries](#) and [tools](#).

Processing was initiated by [Ben Fry](#) and [Casey Reas](#). It is developed by a [small team of volunteers](#).

© [Info](#) \ Linux Web hosting by (mt) [Media Temple](#).



Processing

Cover \ Exhibition \ Reference \ Learning \ Download \ Shop \ About

» Feed » Forum » Wiki » Code

Download Processing. Processing is available for Linux, Mac OS X, and Windows. Select your choice below to download the software.

THE Processing SOFTWARE IS PROVIDED TO YOU "AS IS," AND WE MAKE NO EXPRESS OR IMPLIED WARRANTIES WHATSOEVER WITH RESPECT TO ITS FUNCTIONALITY, OPERABILITY, OR USE, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR INFRINGEMENT. WE EXPRESSLY DISCLAIM ANY LIABILITY WHATSOEVER FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, INCIDENTAL OR SPECIAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST REVENUES, LOST PROFITS, LOSSES RESULTING FROM BUSINESS INTERRUPTION OR LOSS OF DATA, REGARDLESS OF THE FORM OF ACTION OR LEGAL THEORY UNDER WHICH THE LIABILITY MAY BE ASSERTED, EVEN IF ADVISED OF THE POSSIBILITY OR LIKELIHOOD OF SUCH DAMAGES.

By downloading the software from this page, you agree to the specified terms.

**Download 2.0 Beta 6 (2 November 2012)**

- » [Windows 64-bit](#)
- » [Linux 64-bit](#)
- » [Mac OS X](#)
- » [Windows 32-bit](#)
- » [Linux 32-bit](#)

The [list of revisions](#) covers the differences between releases in detail. Please read the [changes](#) if you're new to the 2.0 series. Also check the [known problems](#) for this release.

Processing is Open Source Software. The PDE (Processing Development Environment) is released under the GNU GPL (General Public License). The export libraries (also known as 'core') are released under the GNU LGPL (Lesser General Public License). There's more information about Processing and Open Source in the [FAQ](#) and more information about the [GNU GPL](#) and [GNU LGPL](#) at [opensource.org](#). Please contribute to Processing!

[About the releases and their numbering](#)

**Pre-Releases**

The 2.0 pre-releases contain significant changes, be sure to [read about them](#). Note that [Android mode](#) no longer works in Processing 1.5, you'll need to use an alpha or beta release to do Android development.

2.0b6 | 2012 11 02 [Windows 32-bit](#) | [Windows 64-bit](#) | [Mac OS X](#) | [Linux 32-bit](#) | [Linux 64-bit](#) | [more fixes](#)

2.0b5 | 2012 10 22 [Windows 32-bit](#) | [Windows 64-bit](#) | [Mac OS X](#) | [Linux 32-bit](#) | [Linux 64-bit](#) | [fixes for steps backwards](#)

2.0b4 | 2012 10 21 [Windows 32-bit](#) | [Windows 64-bit](#) | [Mac OS X](#) | [Linux 32-bit](#) | [Linux 64-bit](#) | [much better beta](#)

2.0b3 | 2012 09 10 [Windows 32-bit](#) | [Windows 64-bit](#) | [Mac OS X](#) | [Linux 32-bit](#) | [Linux 64-bit](#) | [beta updates](#)

**Stable Releases**

<http://processing.org/>

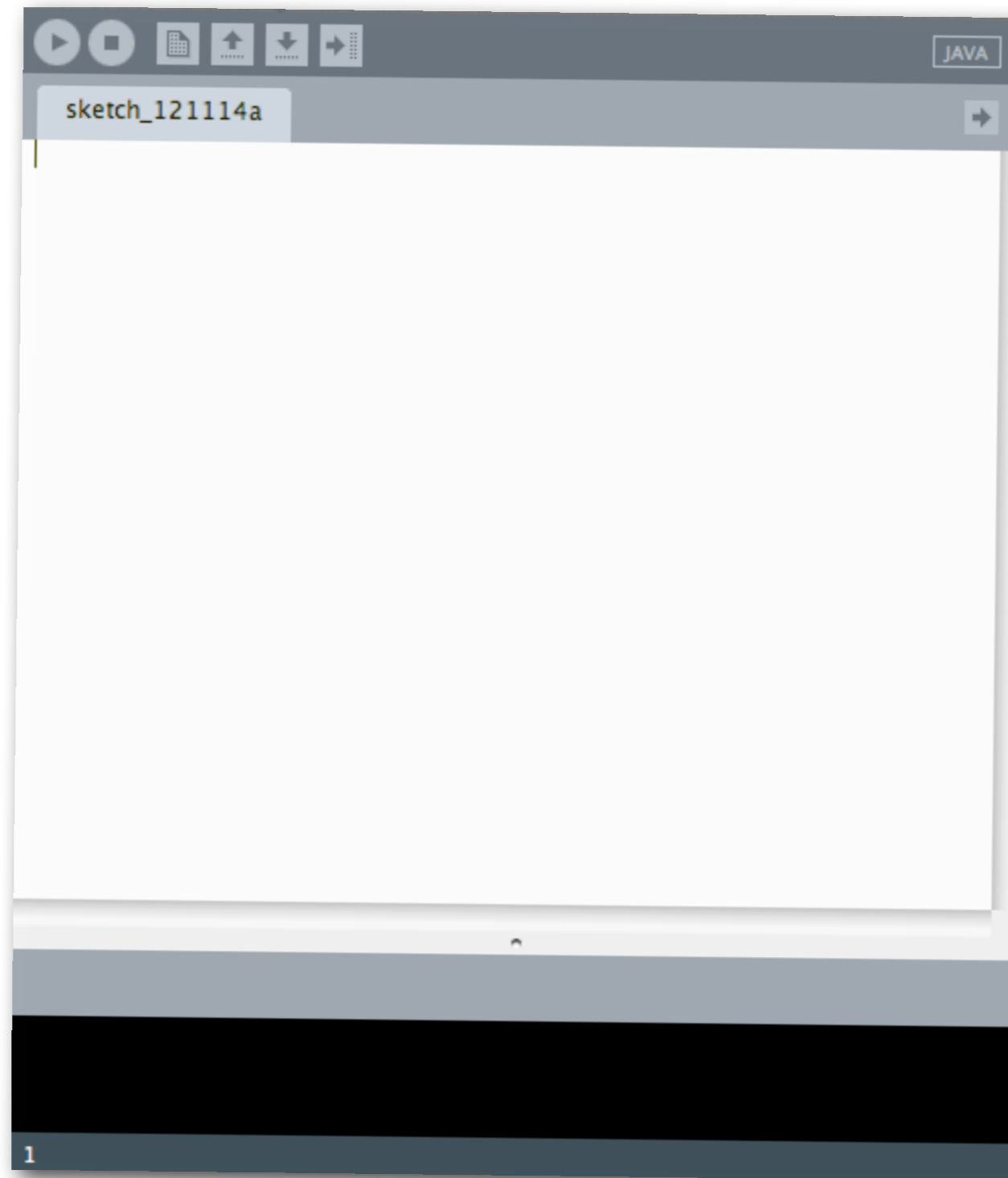
# Intro to Processing

- Run the software

<http://processing.org/>

# Intro to Processing

- Run the software

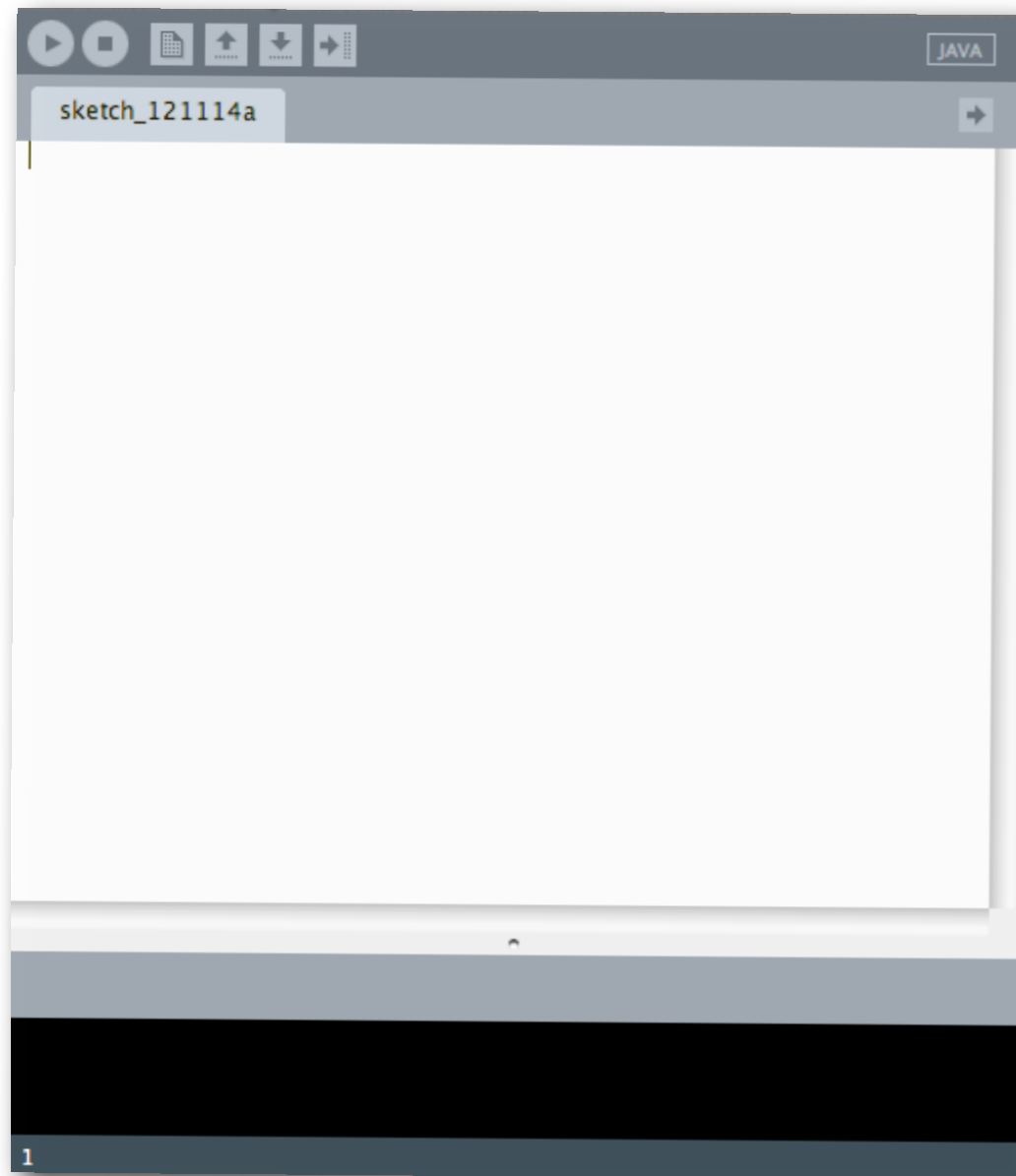


<http://processing.org/>



# Intro to Processing

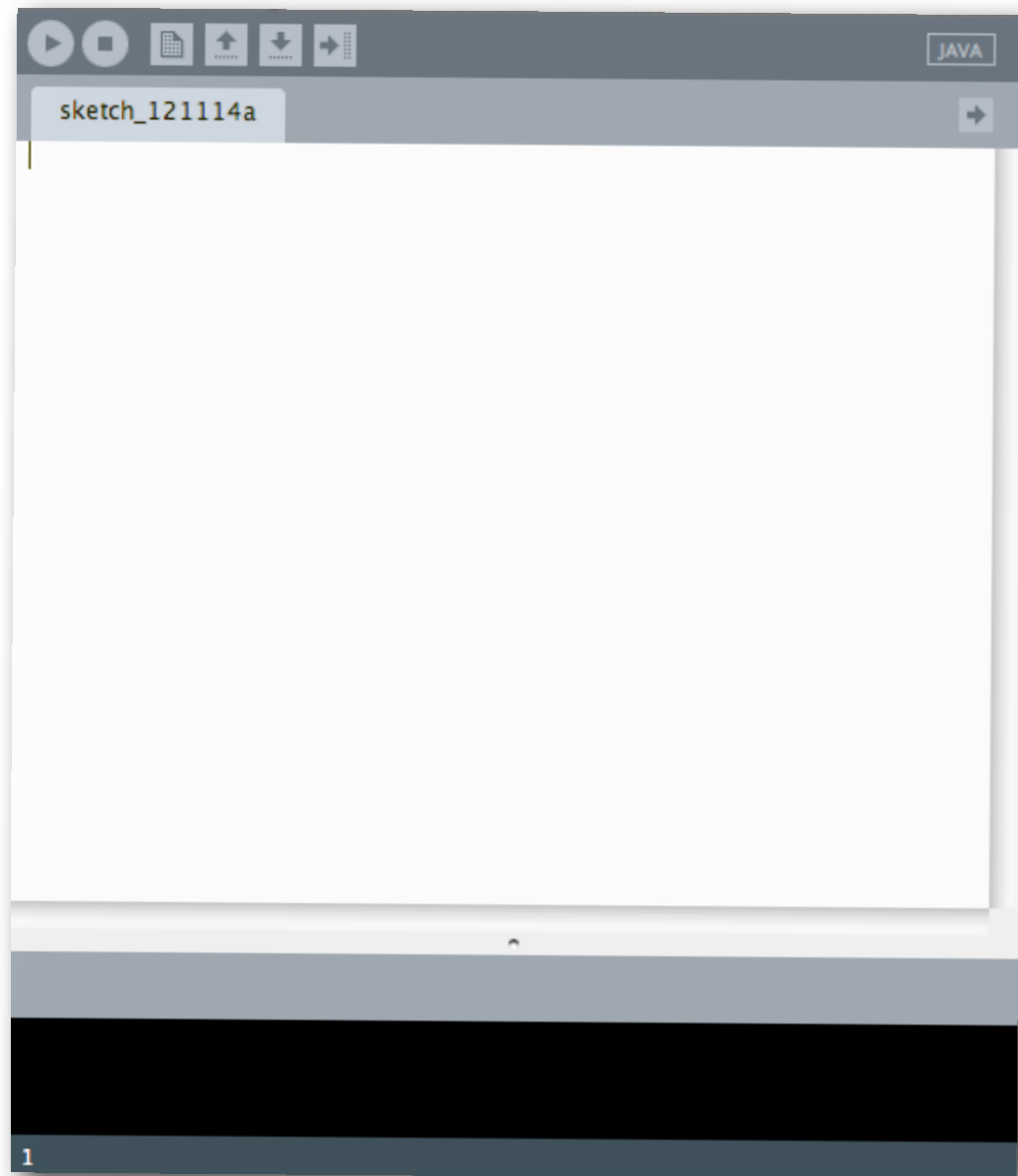
- Run the software



<http://processing.org/>

# Intro to Processing

- Run the software

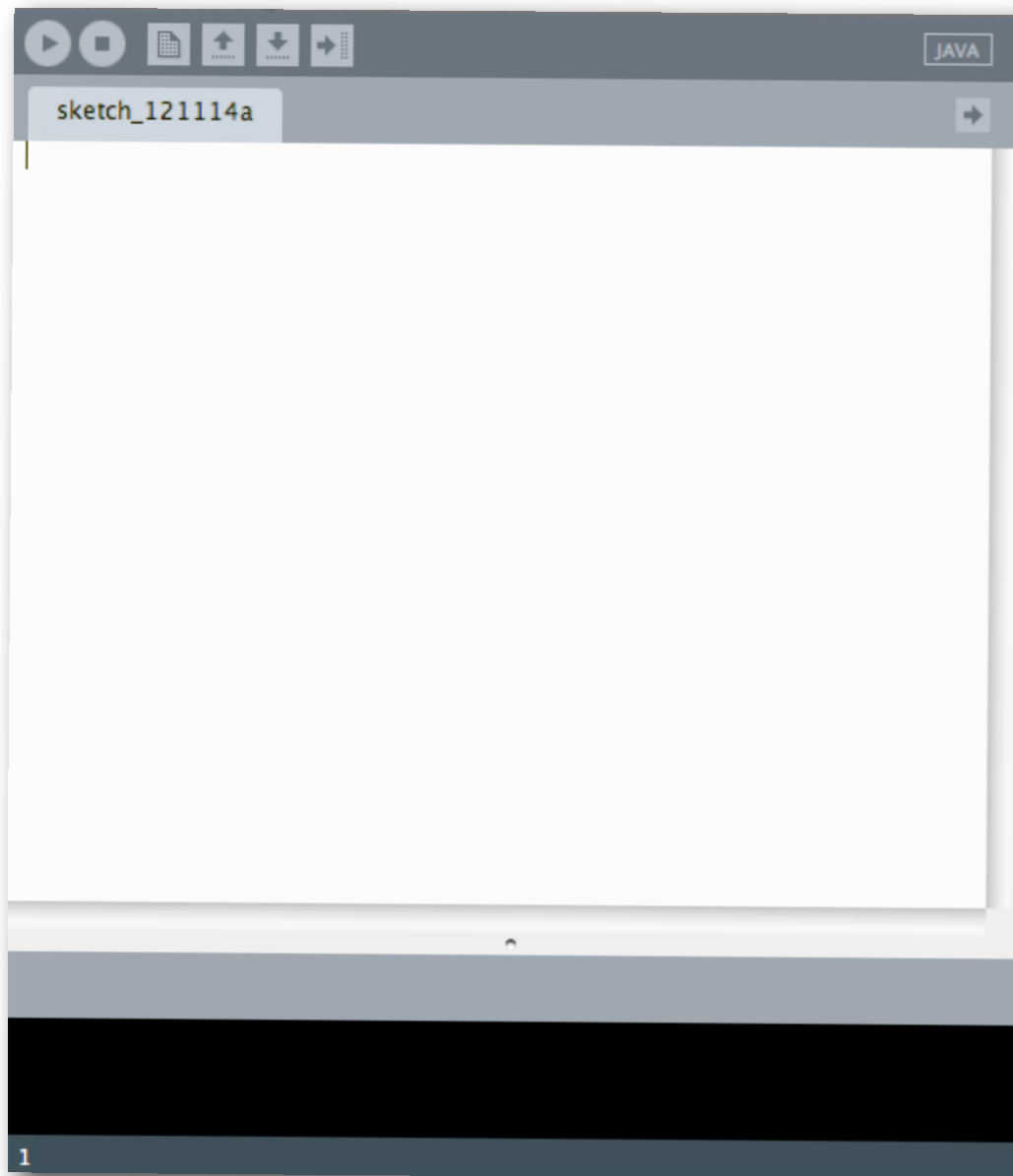


- This is the sketch window

<http://processing.org/>

# Intro to Processing

- Run the software

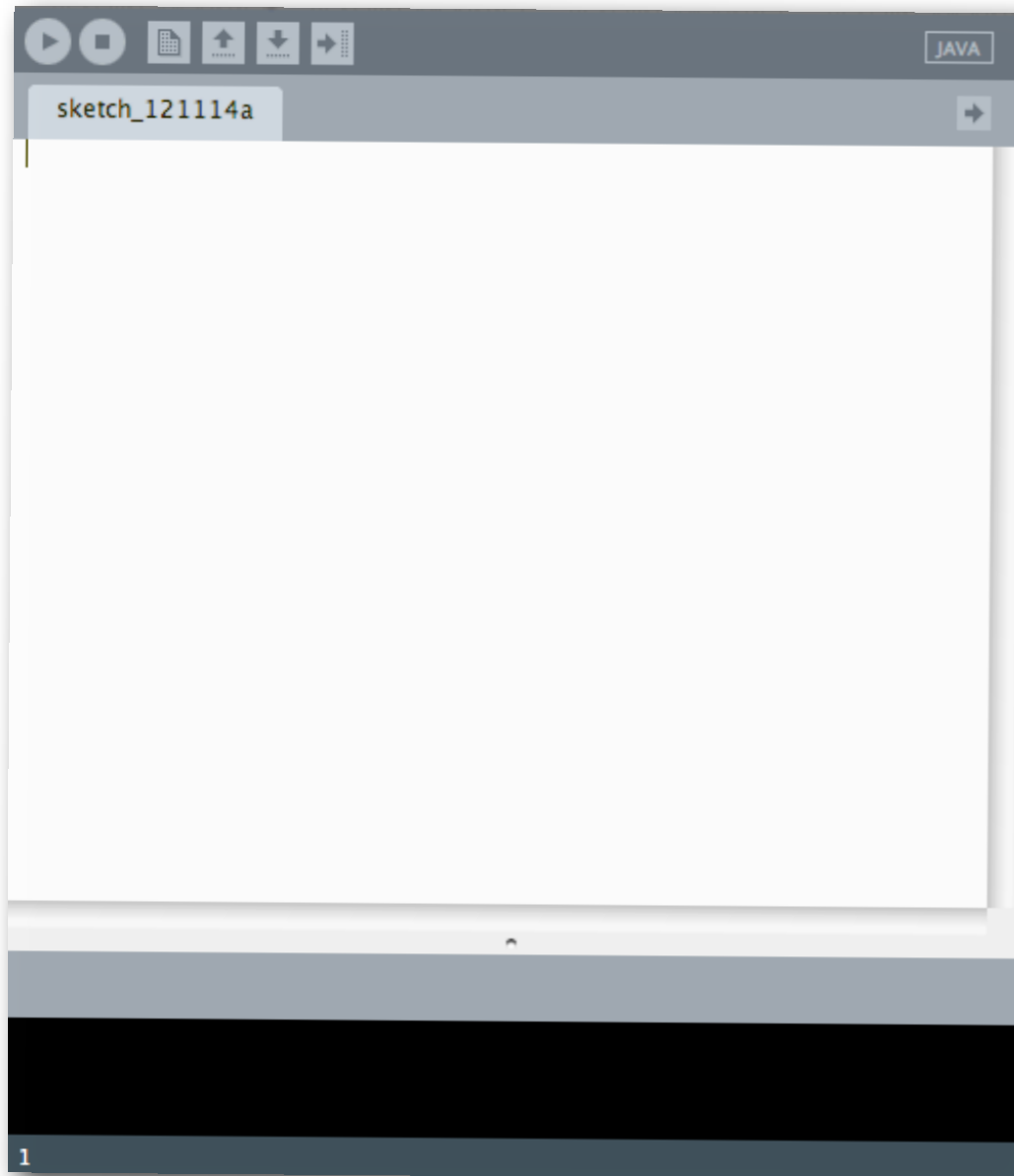


- This is the sketch window
  - It is part of the Processing Development Environment (PDE)

<http://processing.org/>

# Intro to Processing

- Run the software



- This is the sketch window
  - It is part of the Processing Development Environment (PDE)
  - This is where you put your program's instructions

<http://processing.org/>

# Intro to Processing

- Run the software



<http://processing.org/>

# Intro to Processing

- Run the software

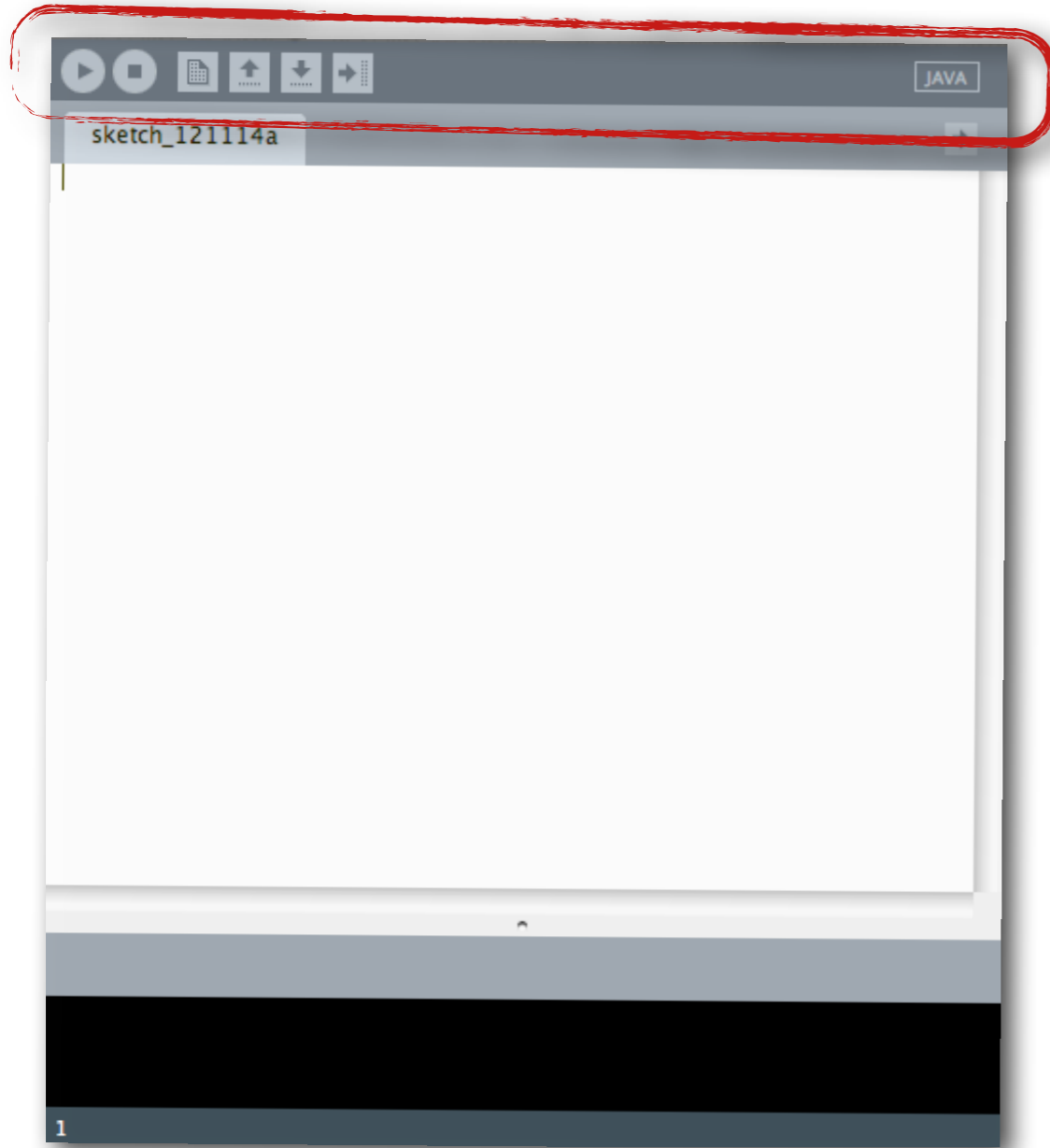


- This is the **Text Editor**

<http://processing.org/>

# Intro to Processing

- Run the software



<http://processing.org/>

# Intro to Processing

- Run the software



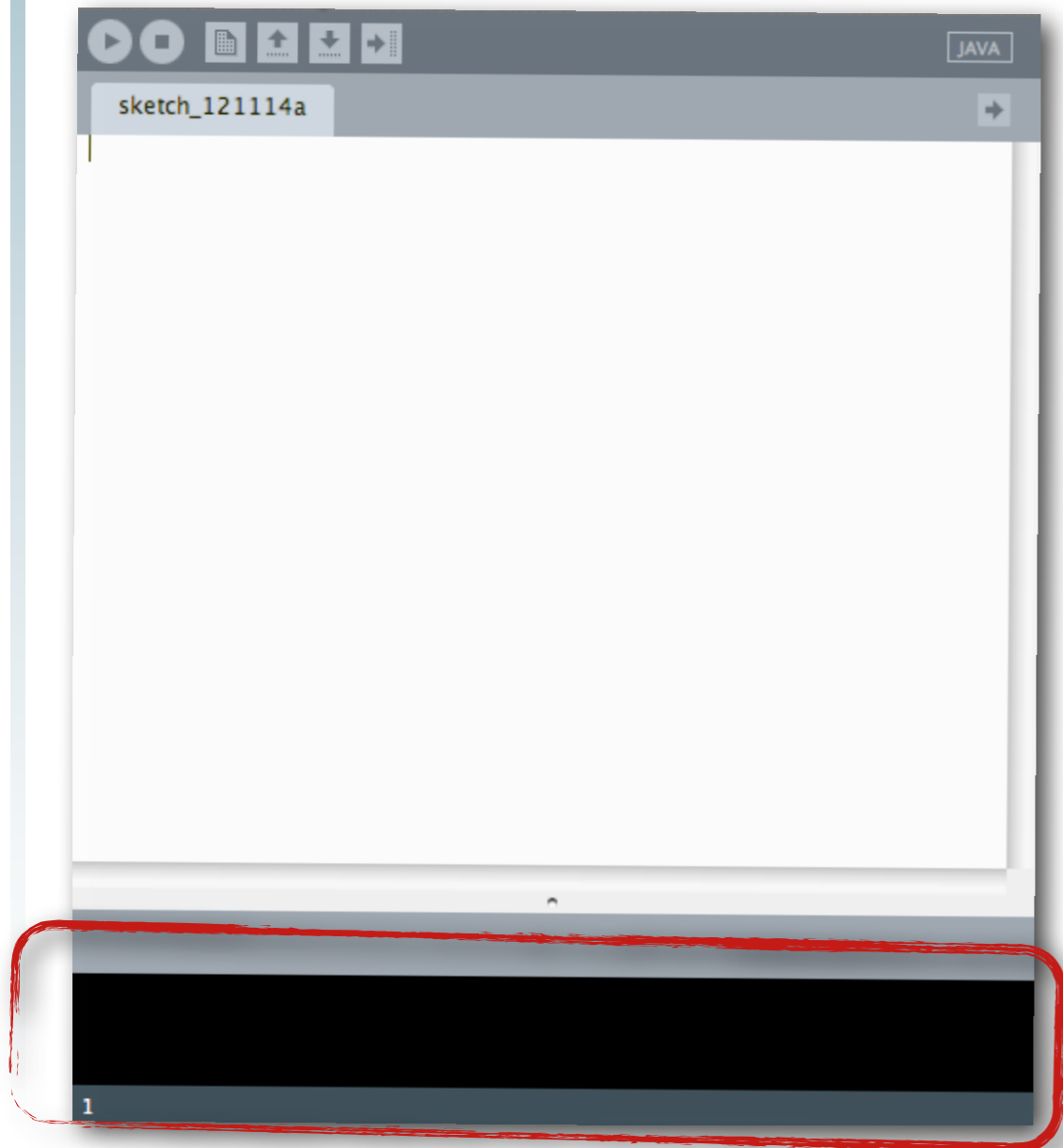
- This is the **toolbar**

<http://processing.org/>



# Intro to Processing

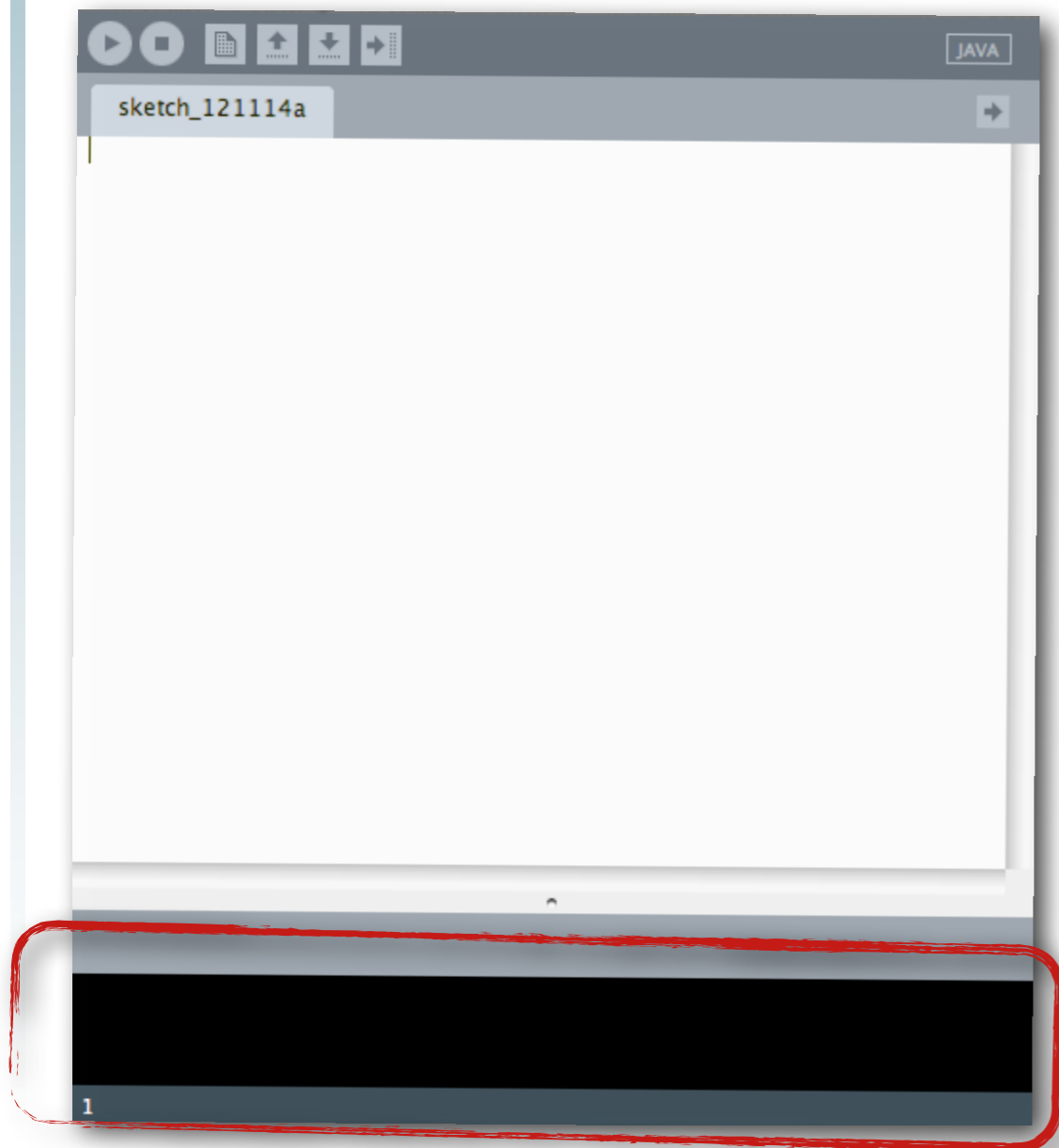
- Run the software



<http://processing.org/>

# Intro to Processing

- Run the software

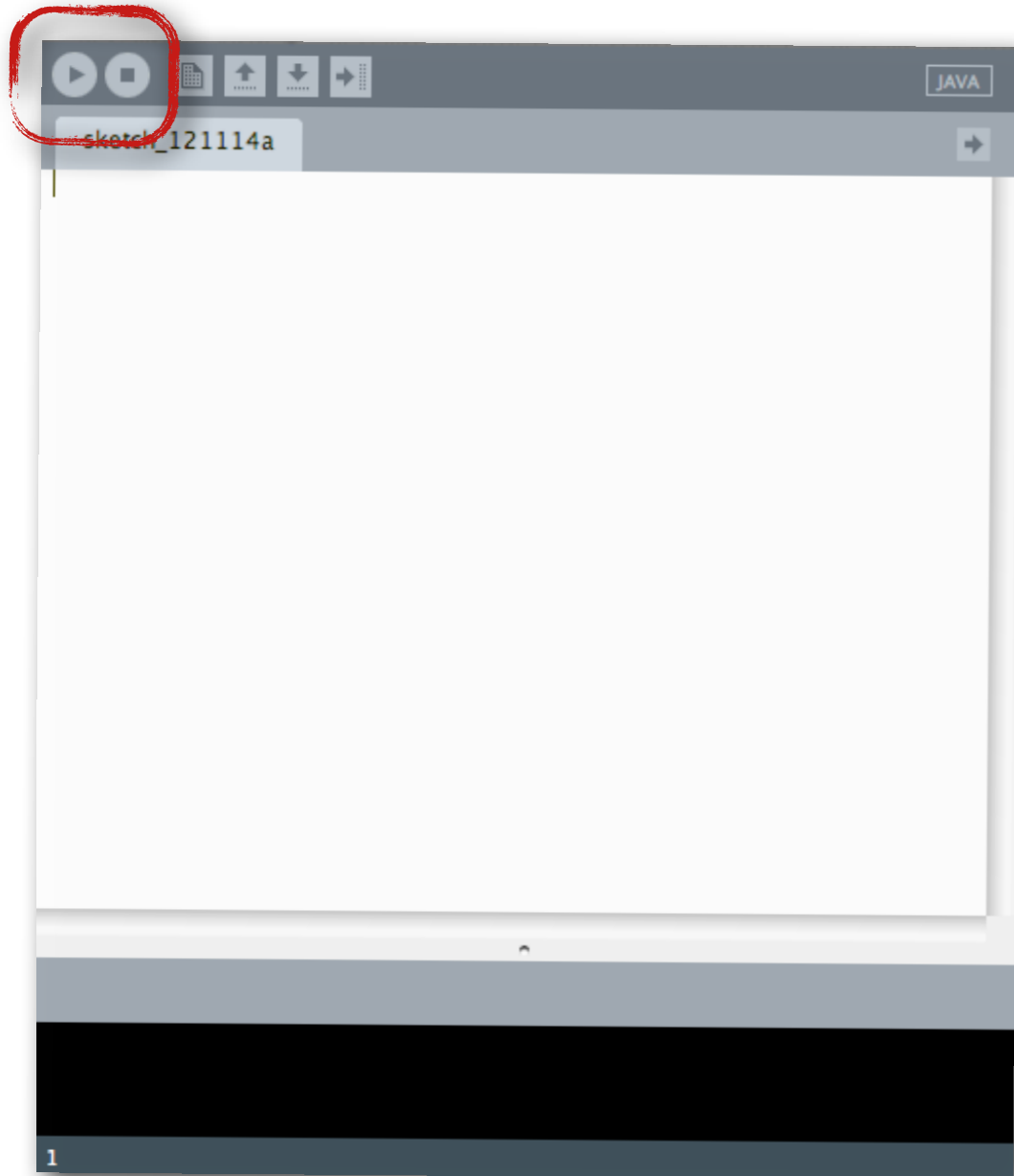


- This is the **message area**

<http://processing.org/>

# Intro to Processing

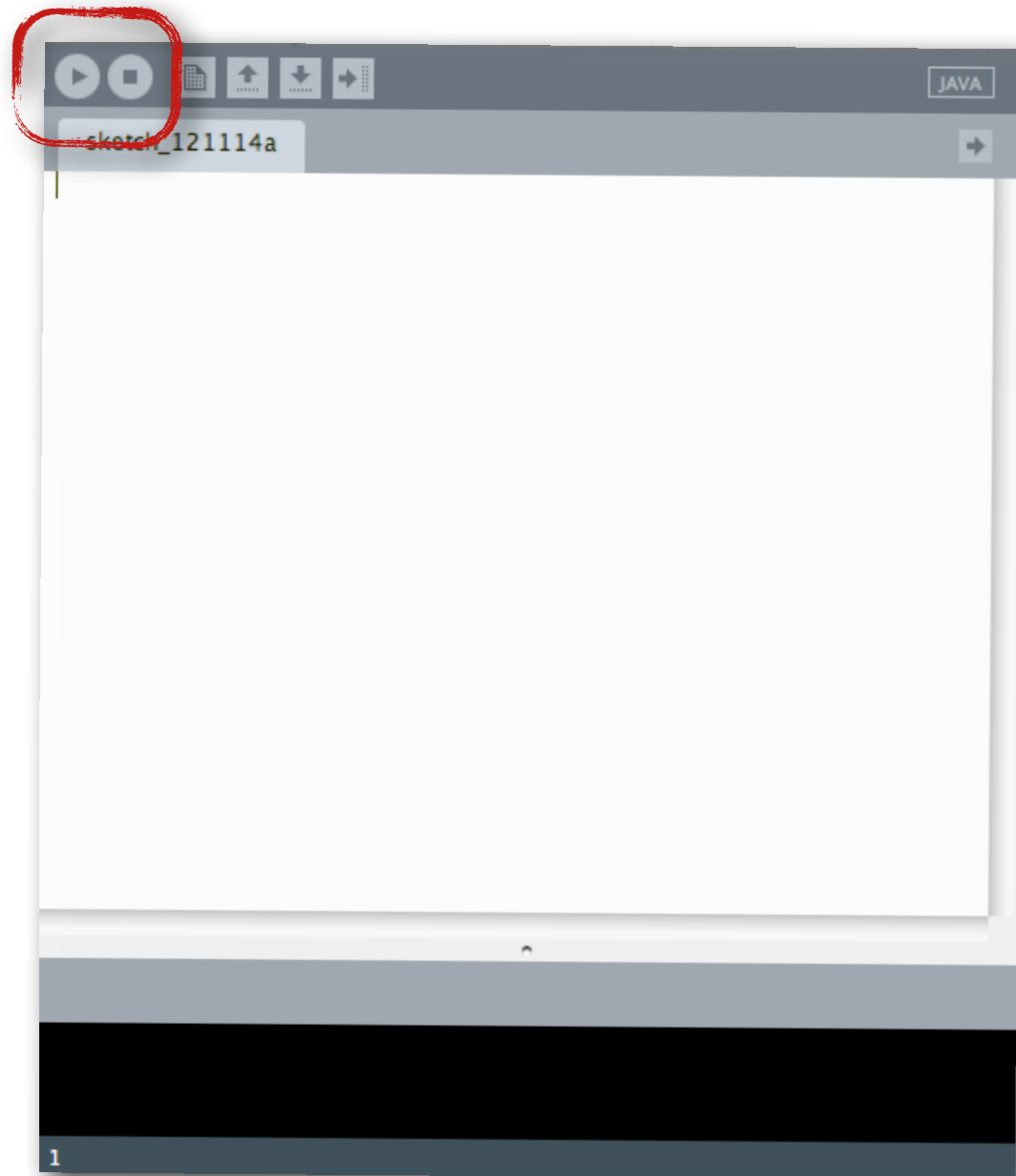
- Run the software



<http://processing.org/>

# Intro to Processing

- Run the software

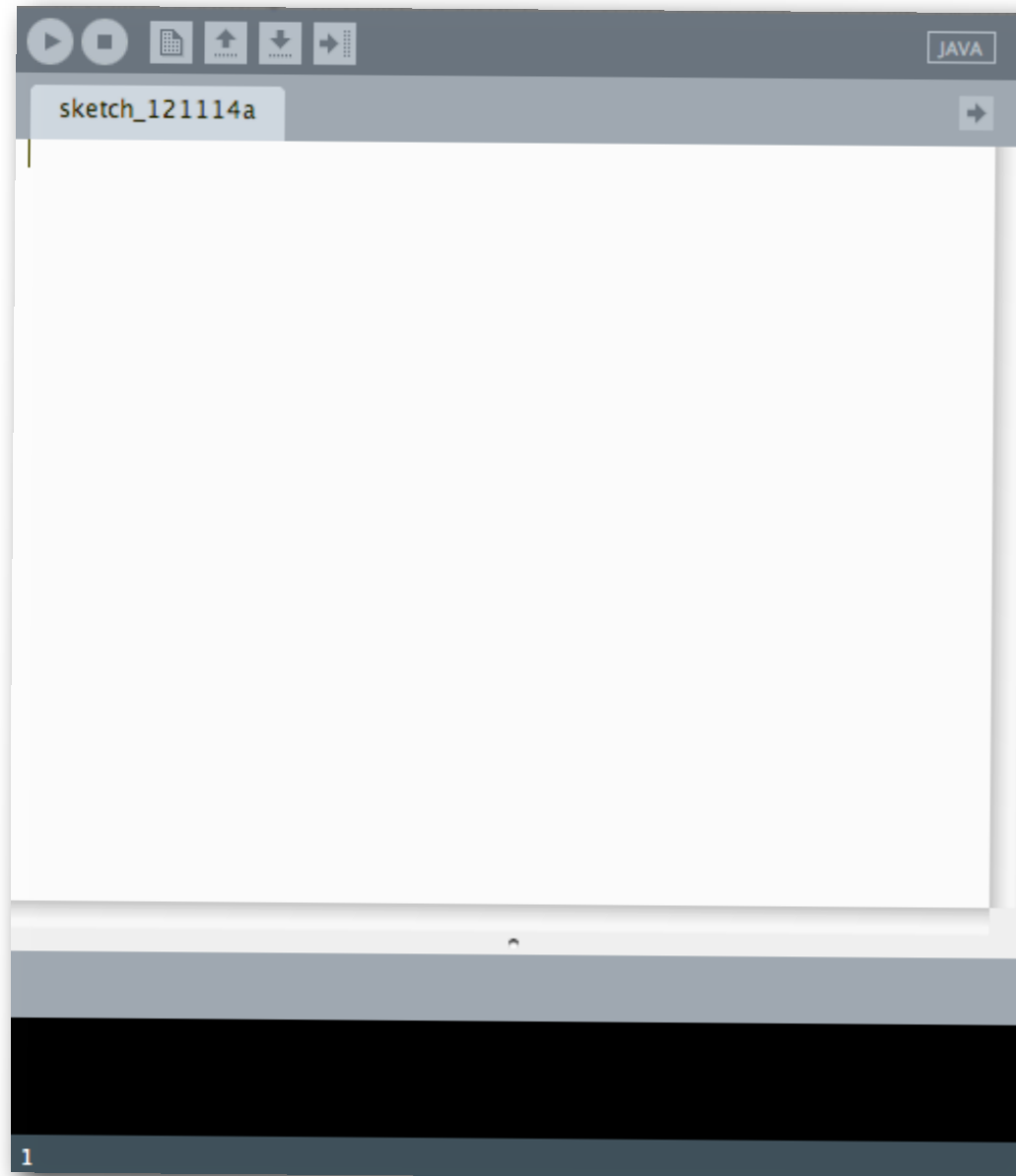


- This is the **run and stop buttons**

<http://processing.org/>

# Intro to Processing

- Run the software

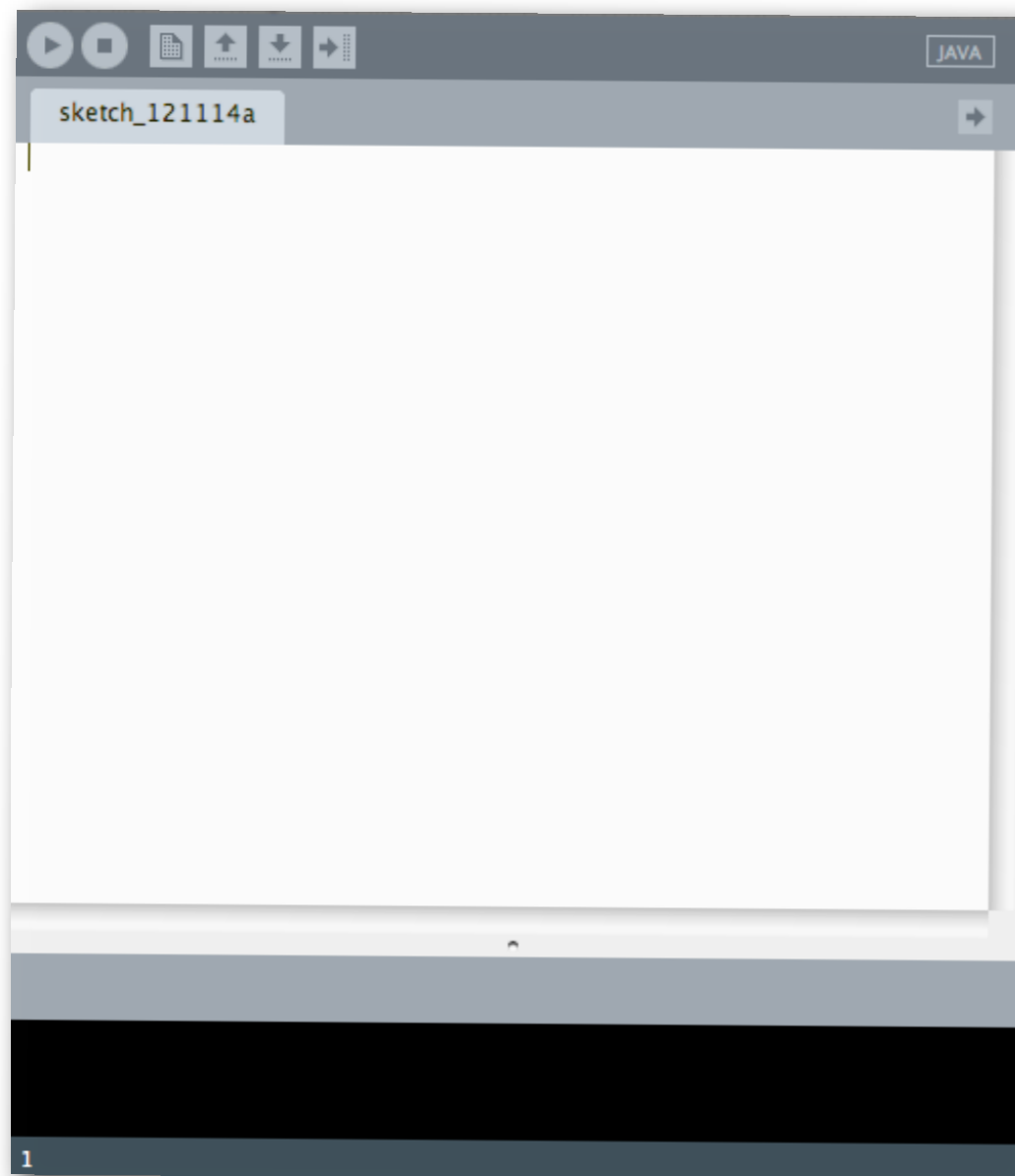


- To run a program
  - type in the program
  - hit run
  - look for the **display window**

<http://processing.org/>

# Intro to Processing

- Run the software

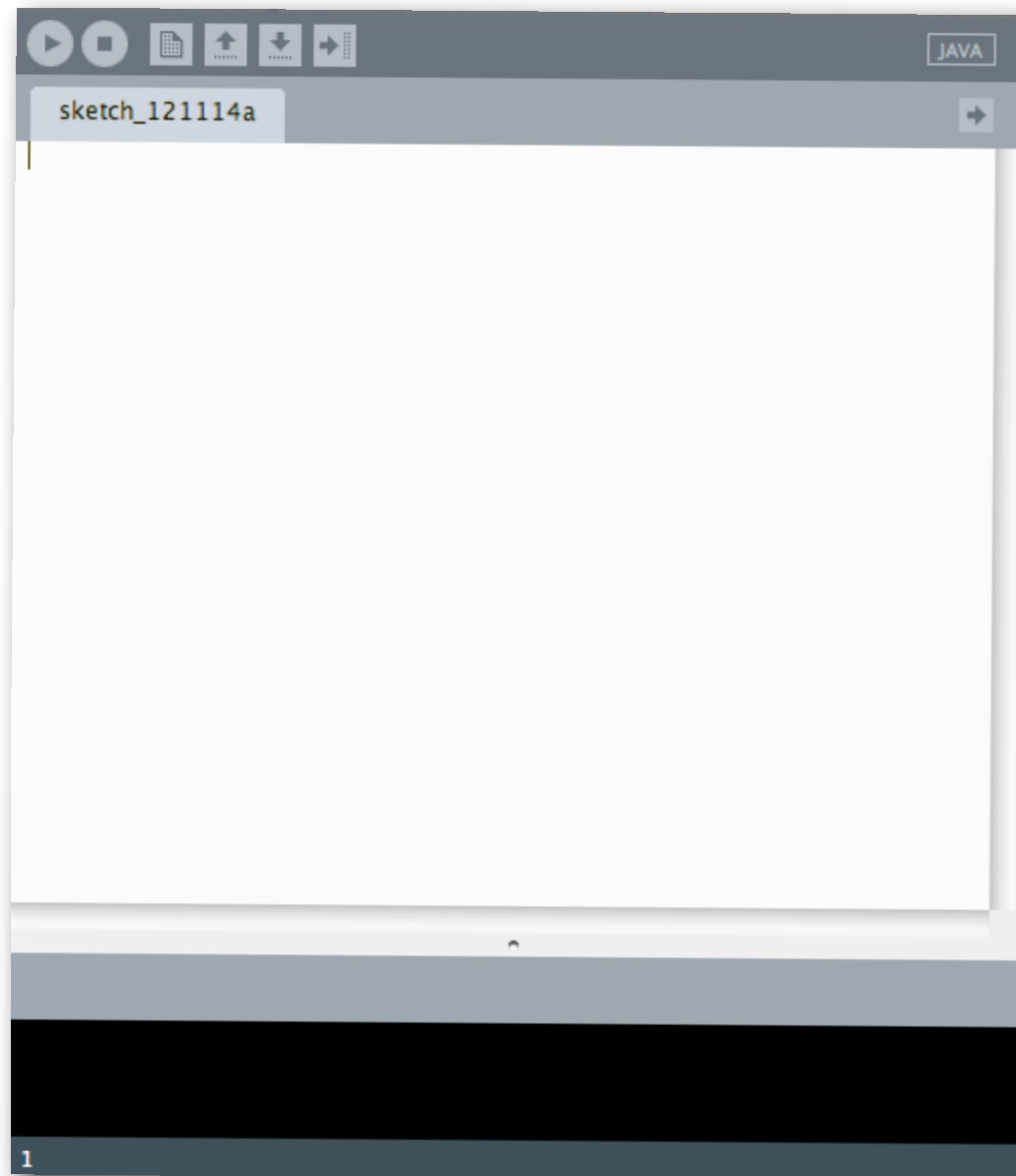



- To run a program
  - type in the program `ellipse(50,50,80,80);`
  - hit run
  - look for the **display window**

<http://processing.org/>

# Intro to Processing

- Run the software

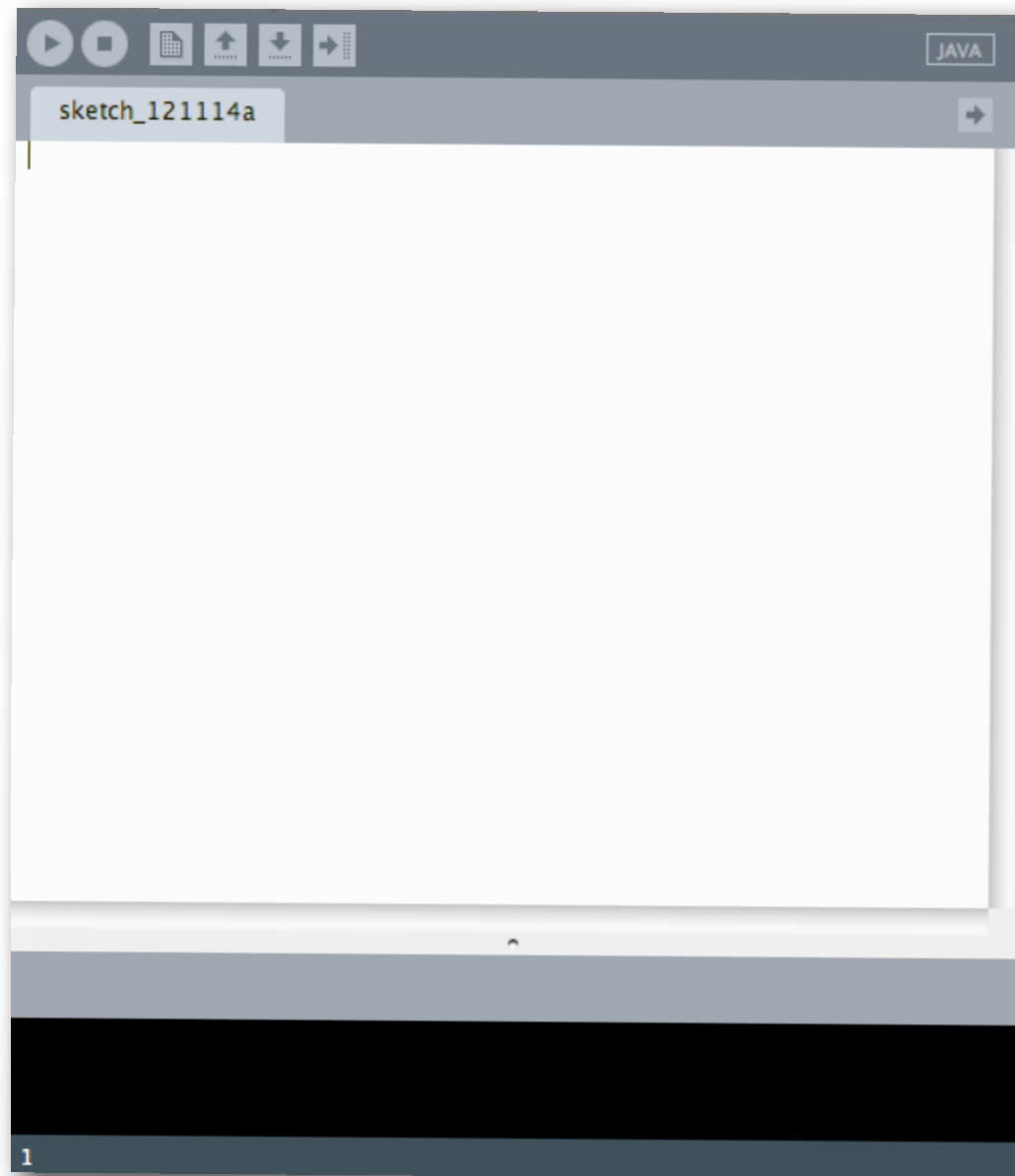


- To run a program
  - type in the program `ellipse(50,50,80,80);`
  - hit run 
  - look for the **display window**

<http://processing.org/>

# Intro to Processing

- Run the software



- To run a program
  - type in the program
  - hit run
  - look for the **display window**

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

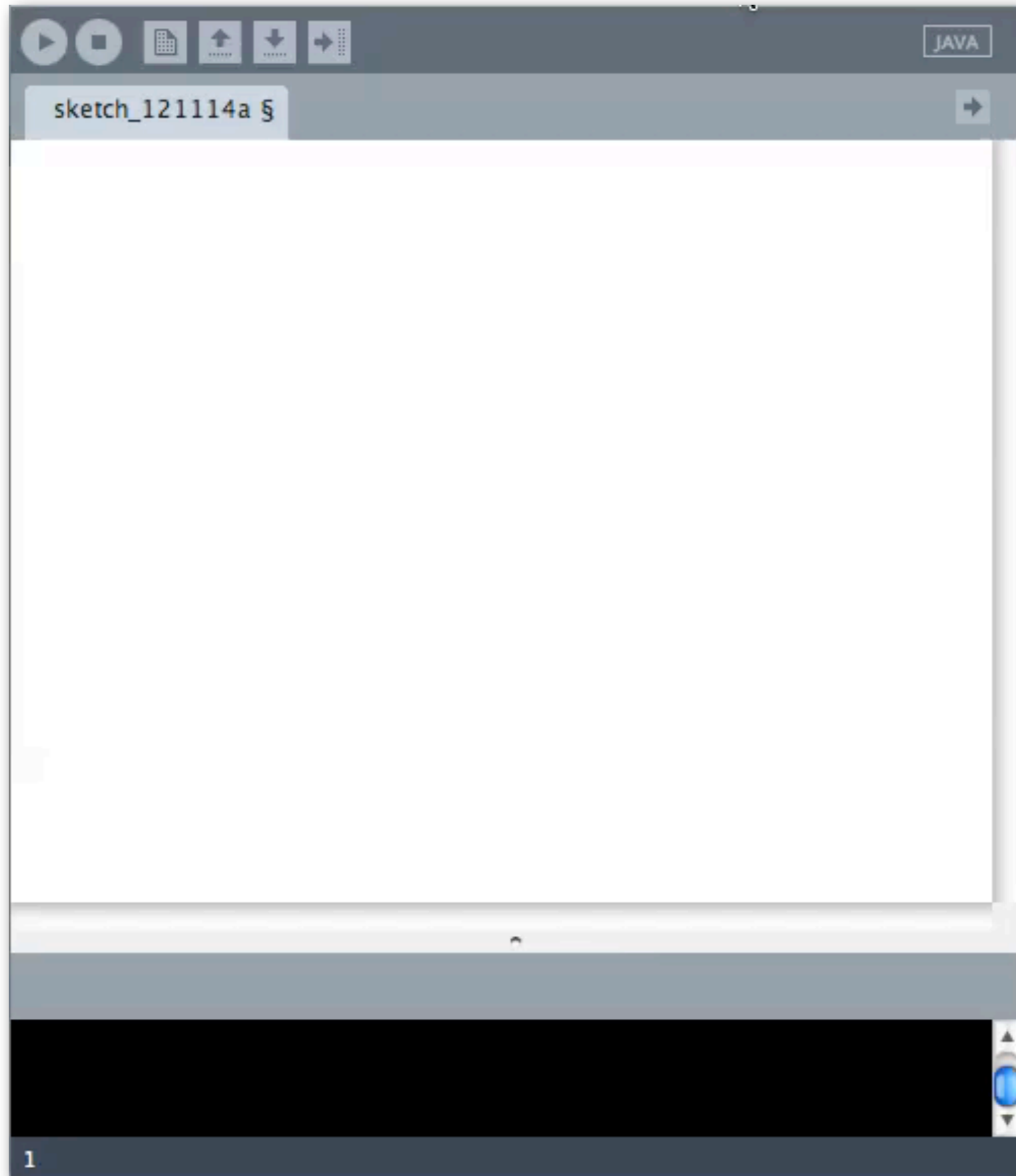


<http://processing.org/>

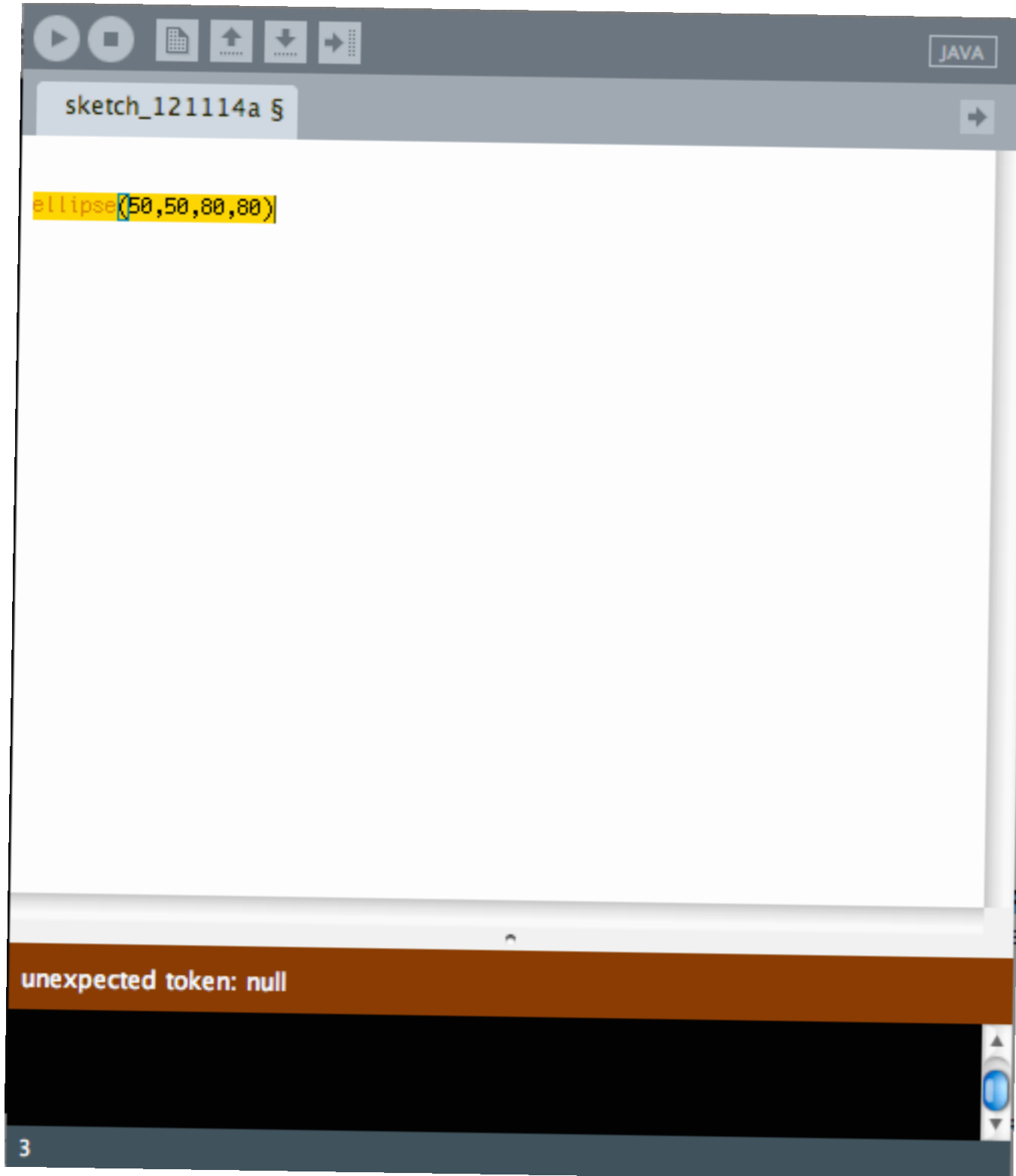


# Intro to Processing

# Intro to Processing

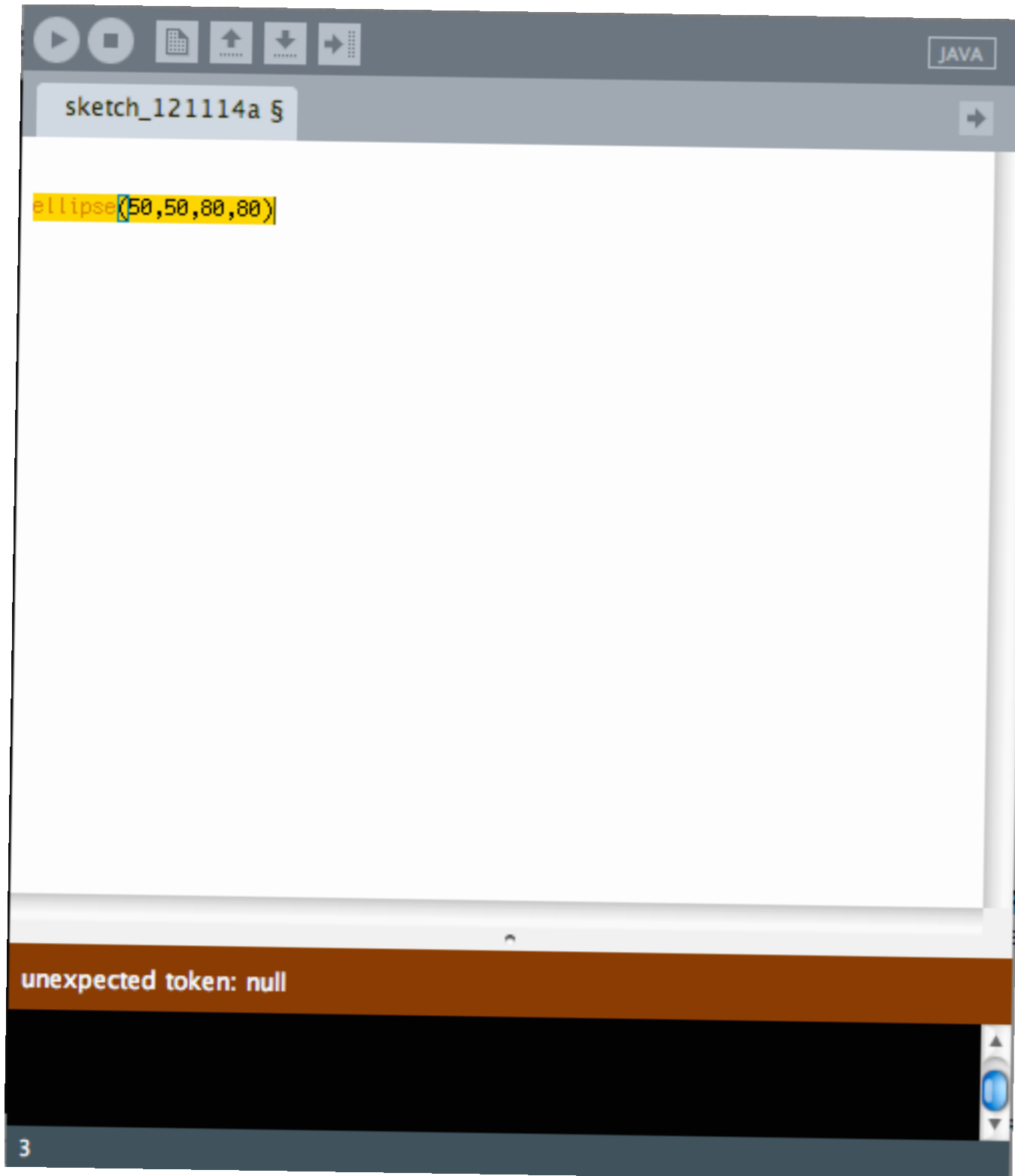


# Intro to Processing



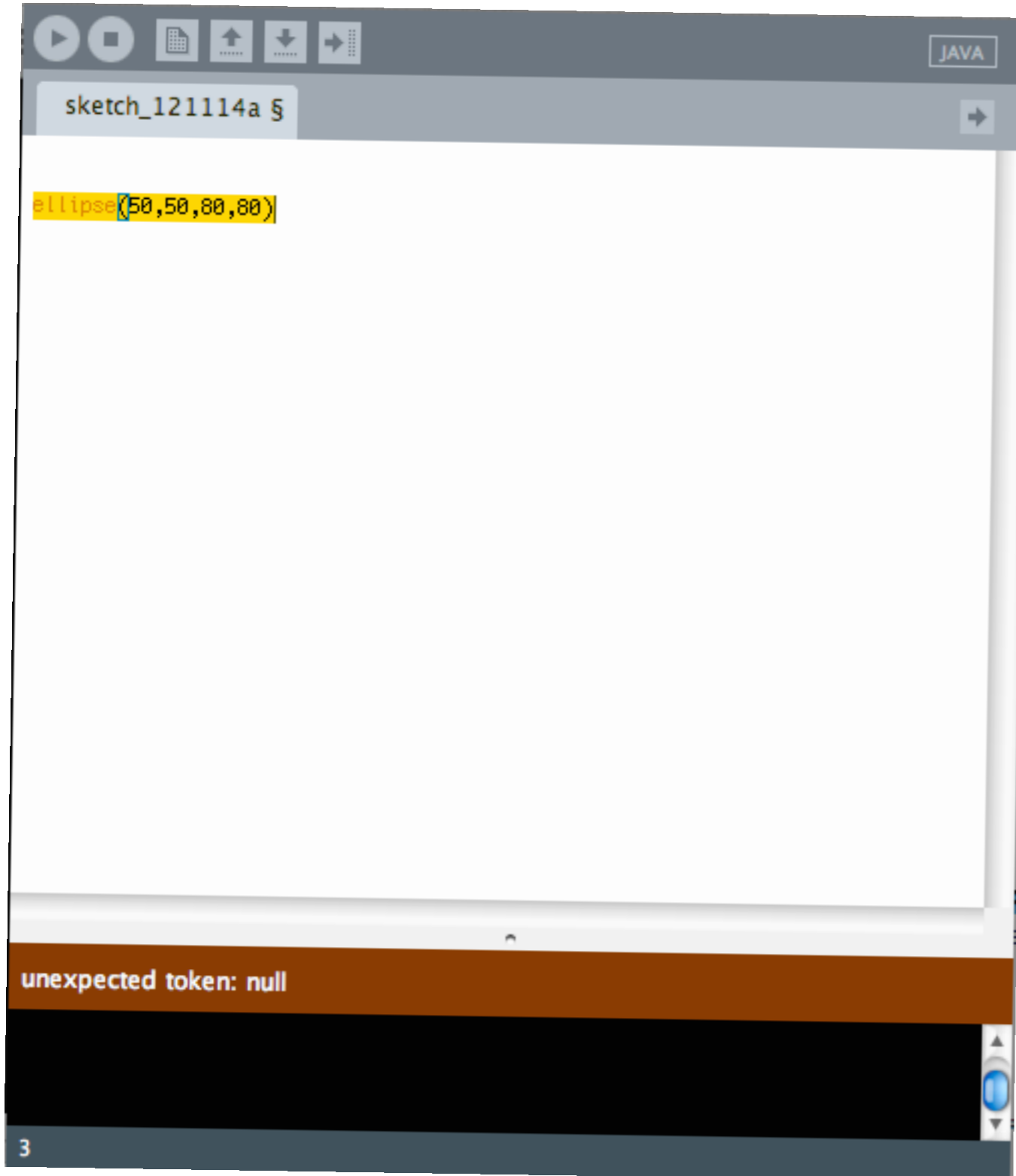
# Intro to Processing

- What if something goes wrong?



# Intro to Processing

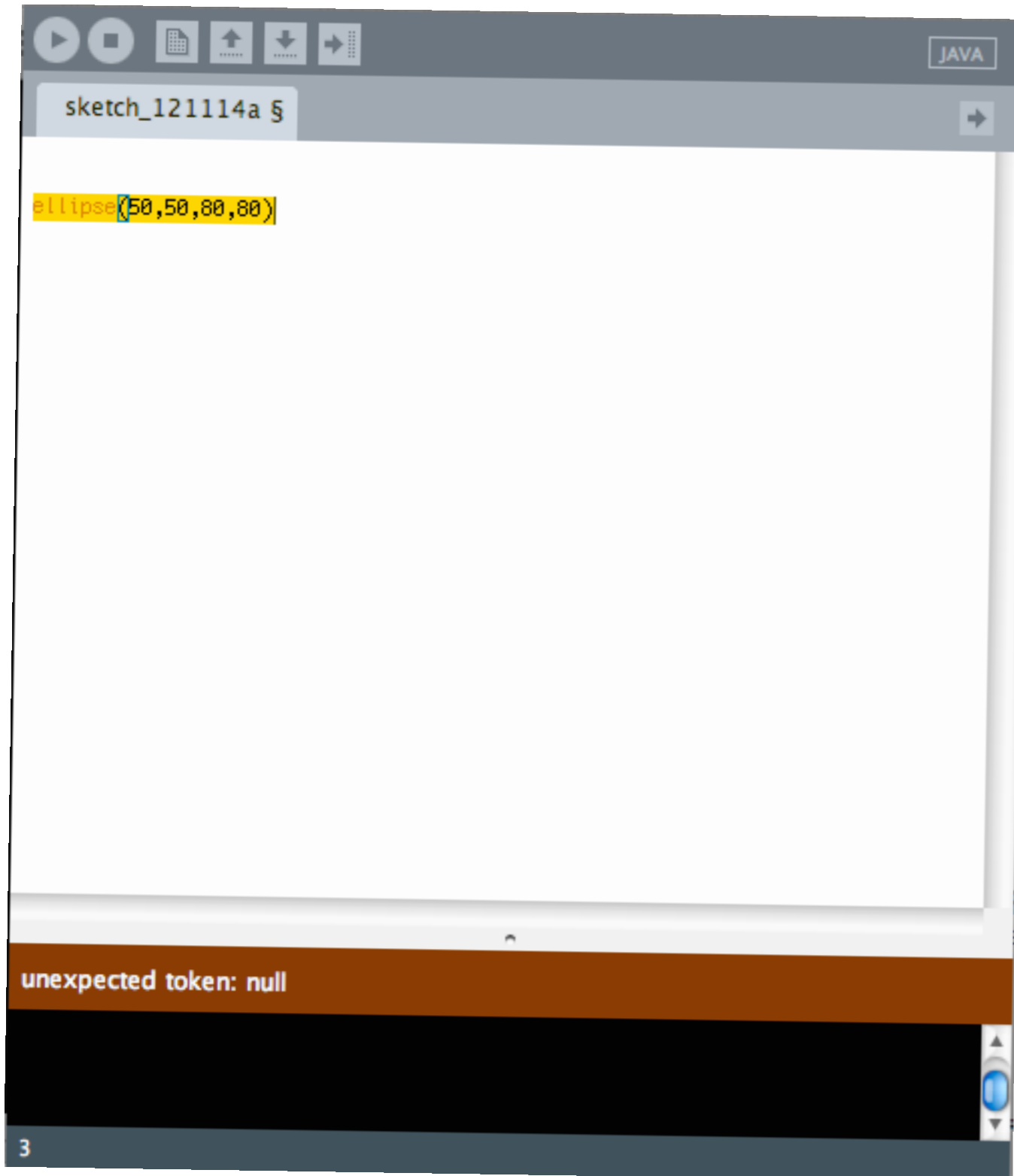
- What if something goes wrong?



- An error will show up in the message area

# Intro to Processing

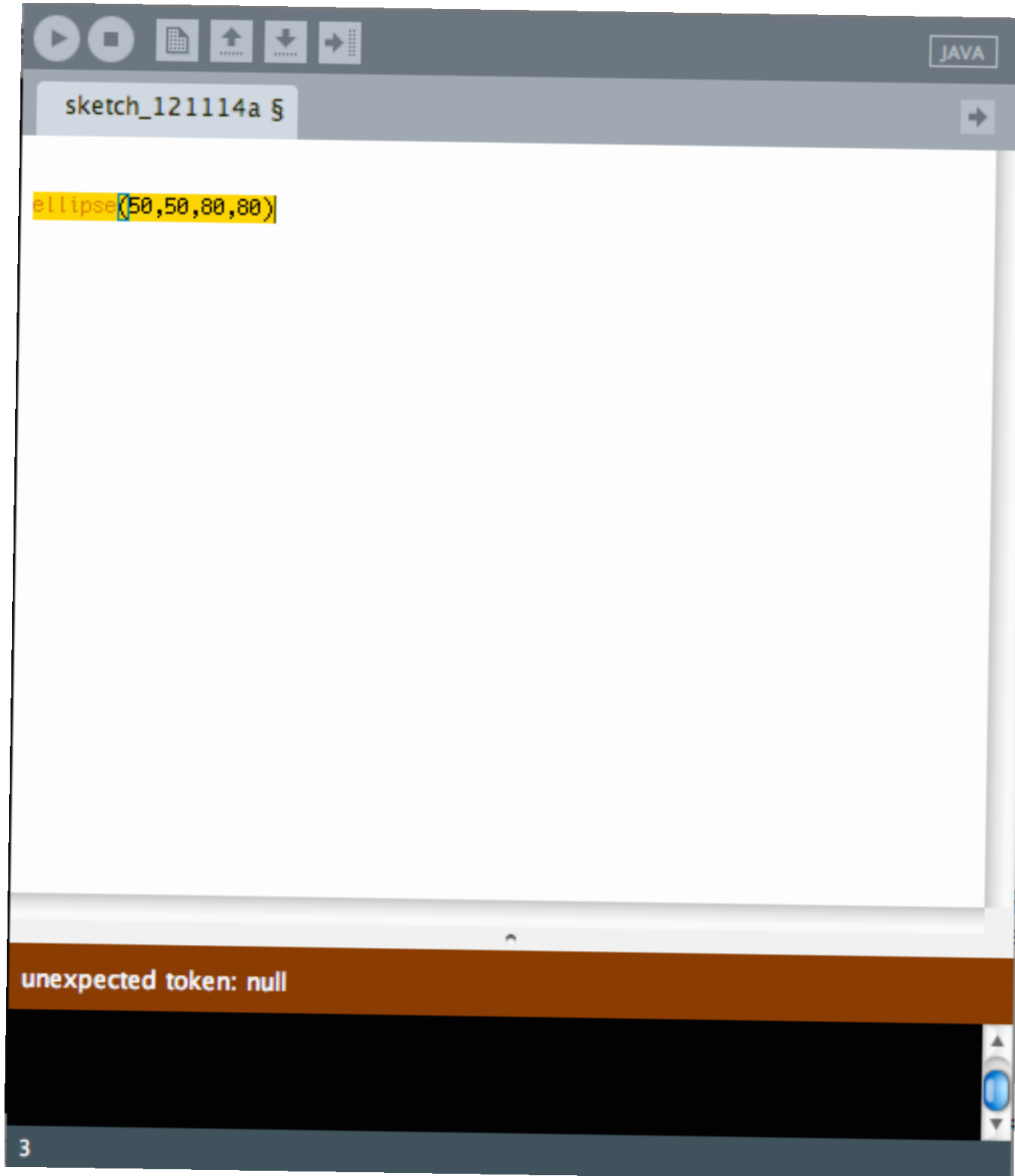
- What if something goes wrong?



- An error will show up in the message area
- Sometimes it will give you a clue about the problem

# Intro to Processing

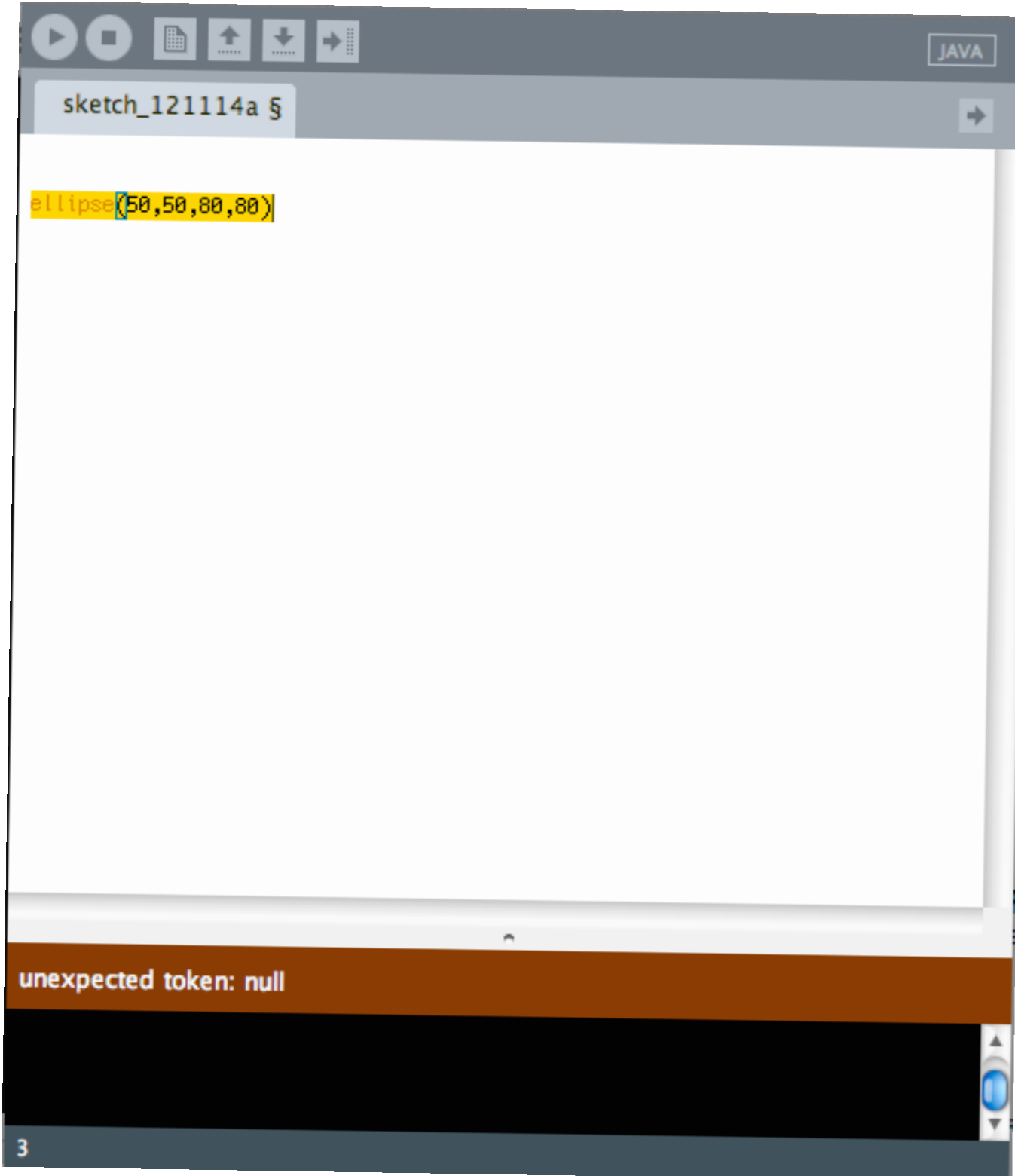
- What if something goes wrong?



- An error will show up in the message area
- Sometimes it will give you a clue about the problem
- Make sure you are using parentheses in pairs

# Intro to Processing

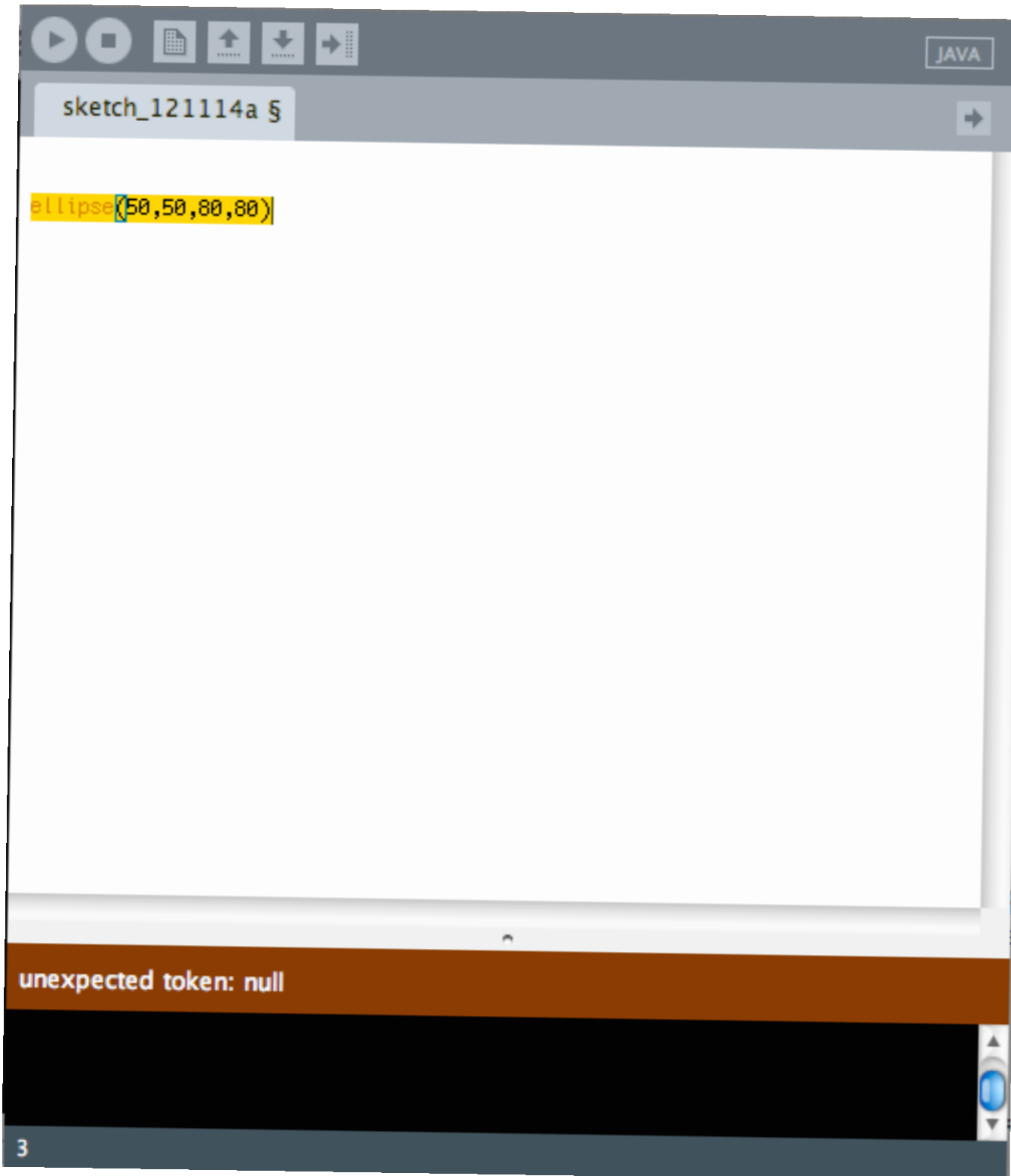
- What if something goes wrong?



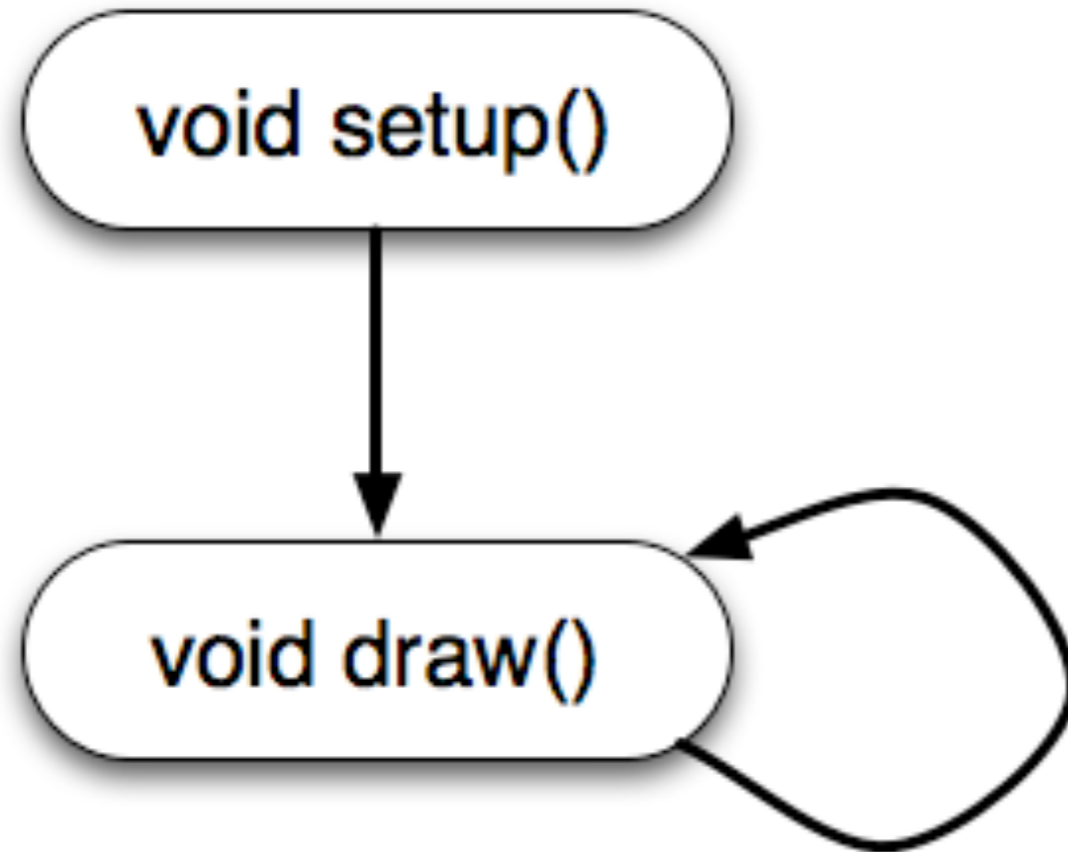
- An error will show up in the message area
- Sometimes it will give you a clue about the problem
- Make sure you are using parentheses in pairs
- Make sure you end a line with a semi-colon



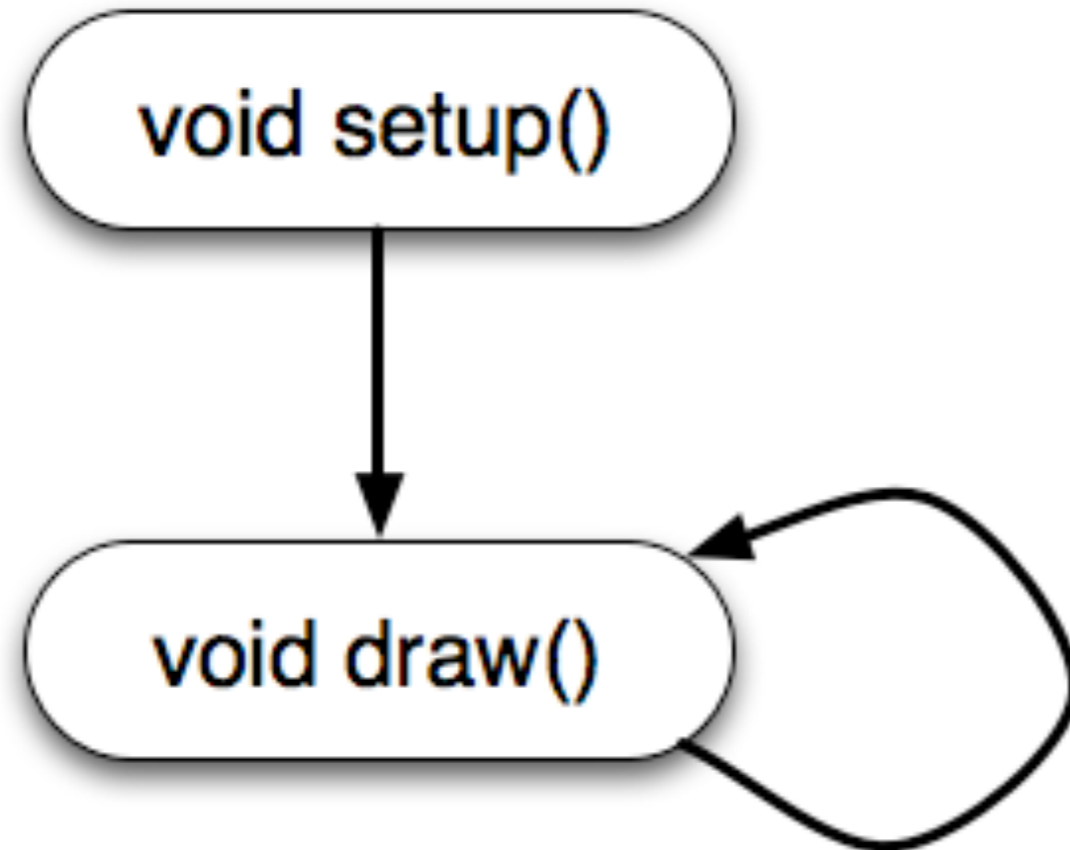
- What if something goes wrong?



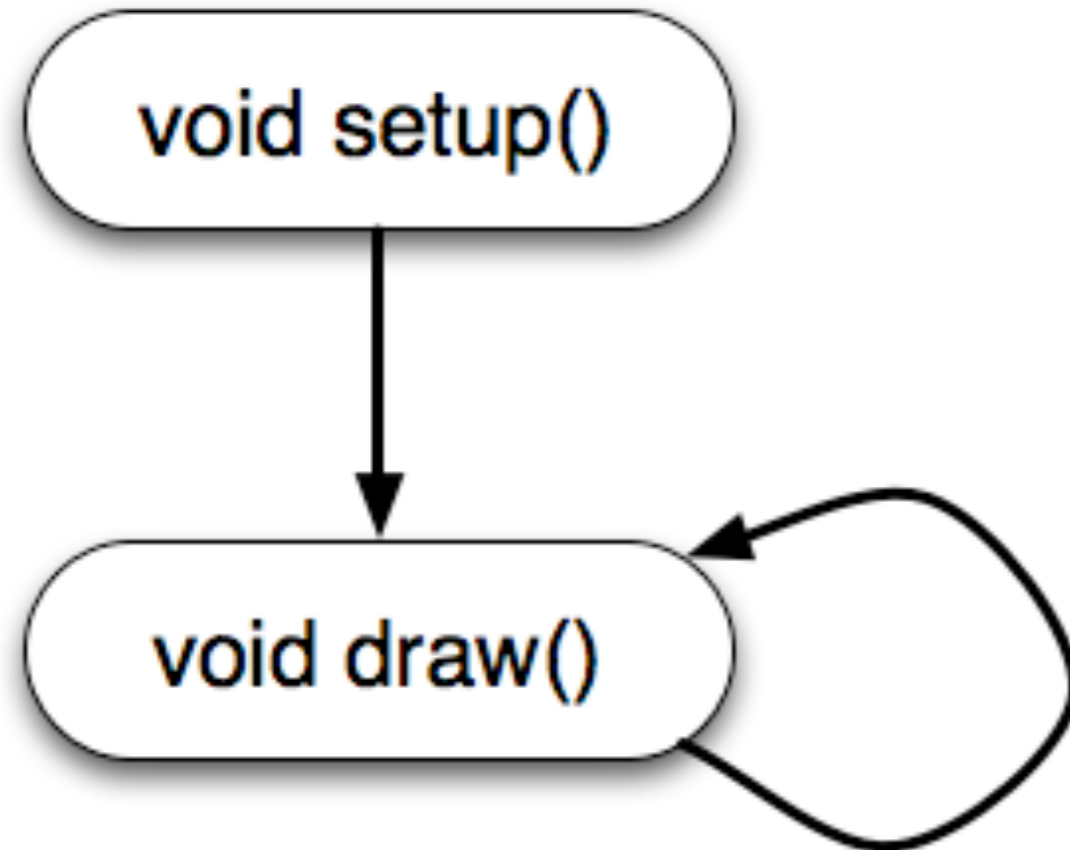
- An error will show up in the message area
- Sometimes it will give you a clue about the problem
- Make sure you are using parentheses in pairs
- Make sure you end a line with a semi-colon
- Make sure you have the right number of parameters for your function



- Program flow

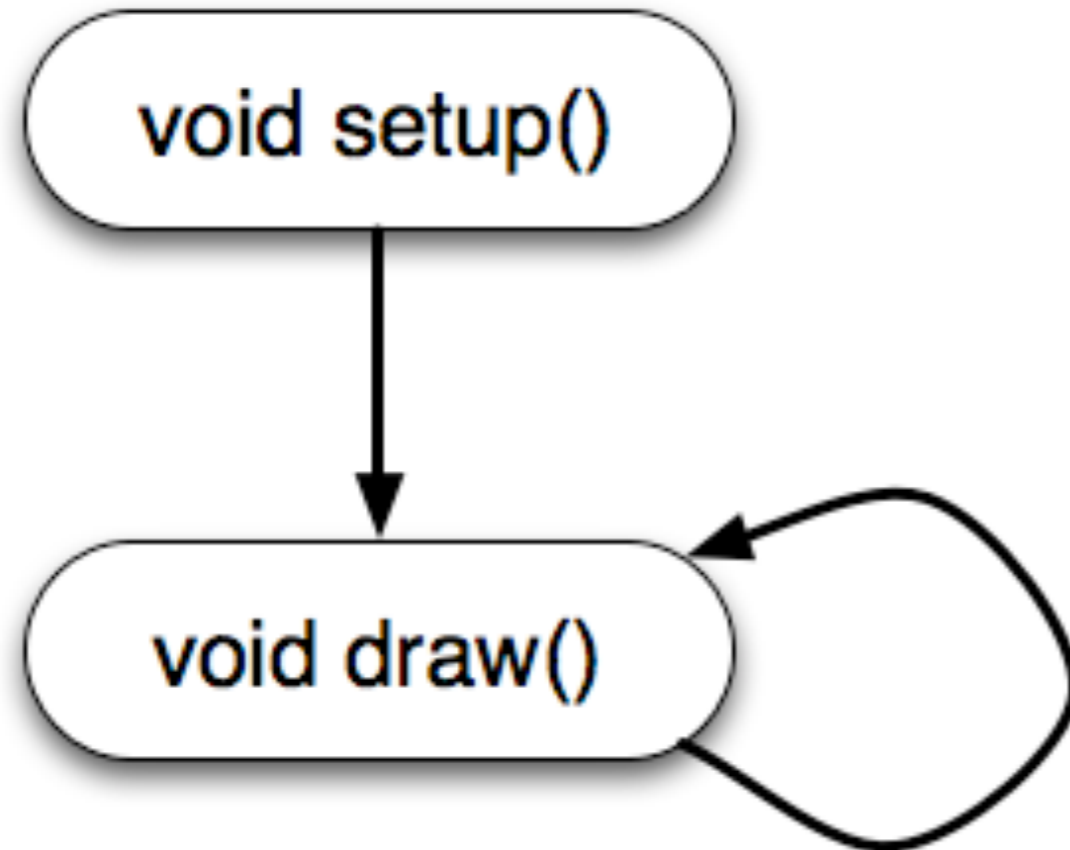


- Program flow



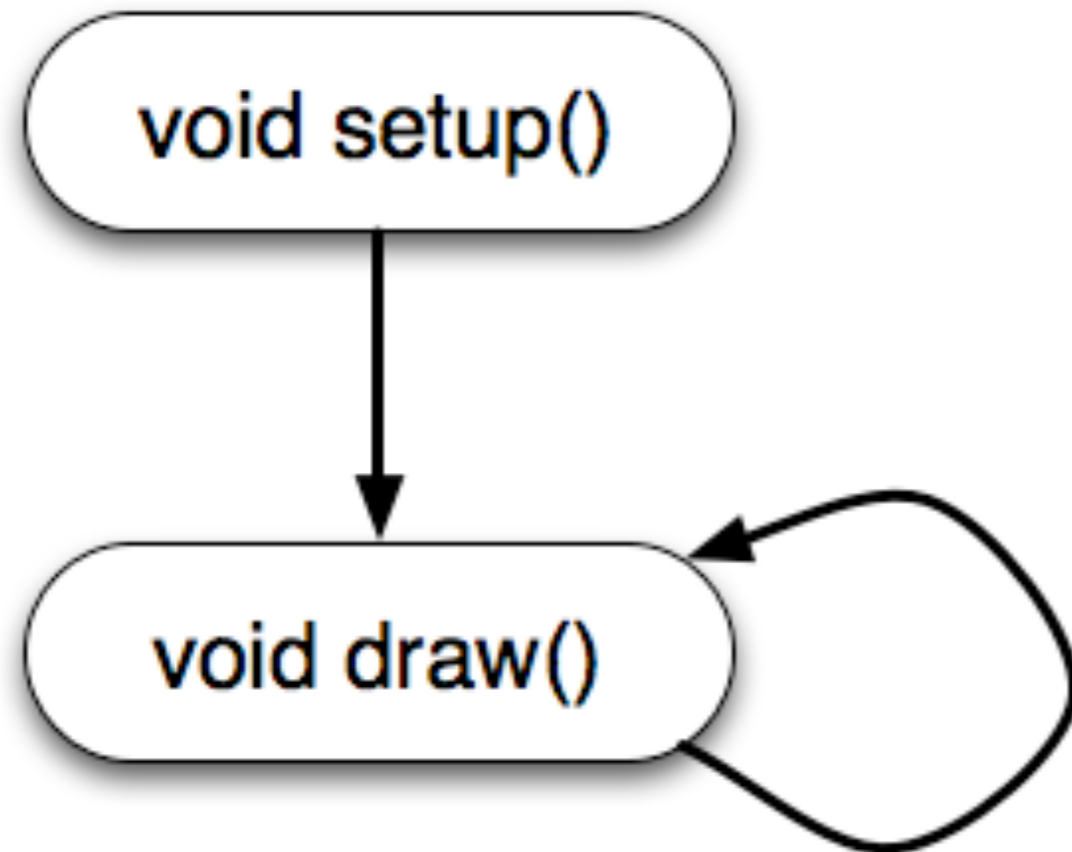
- You can write collections of commands that get run in particular ways by Processing

- Program flow



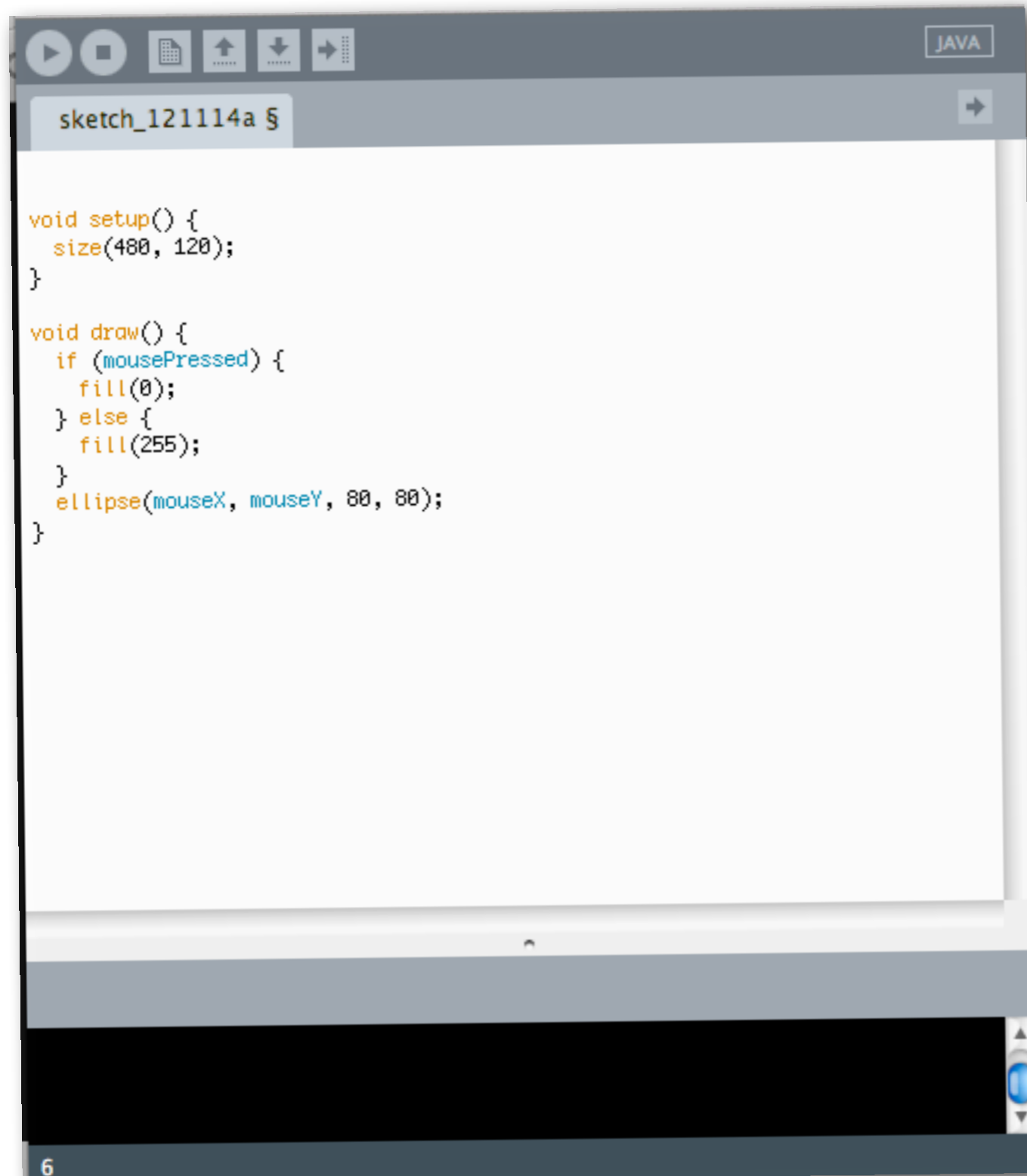
- You can write collections of commands that get run in particular ways by Processing
- the **setup** function is run once at the beginning

- Program flow



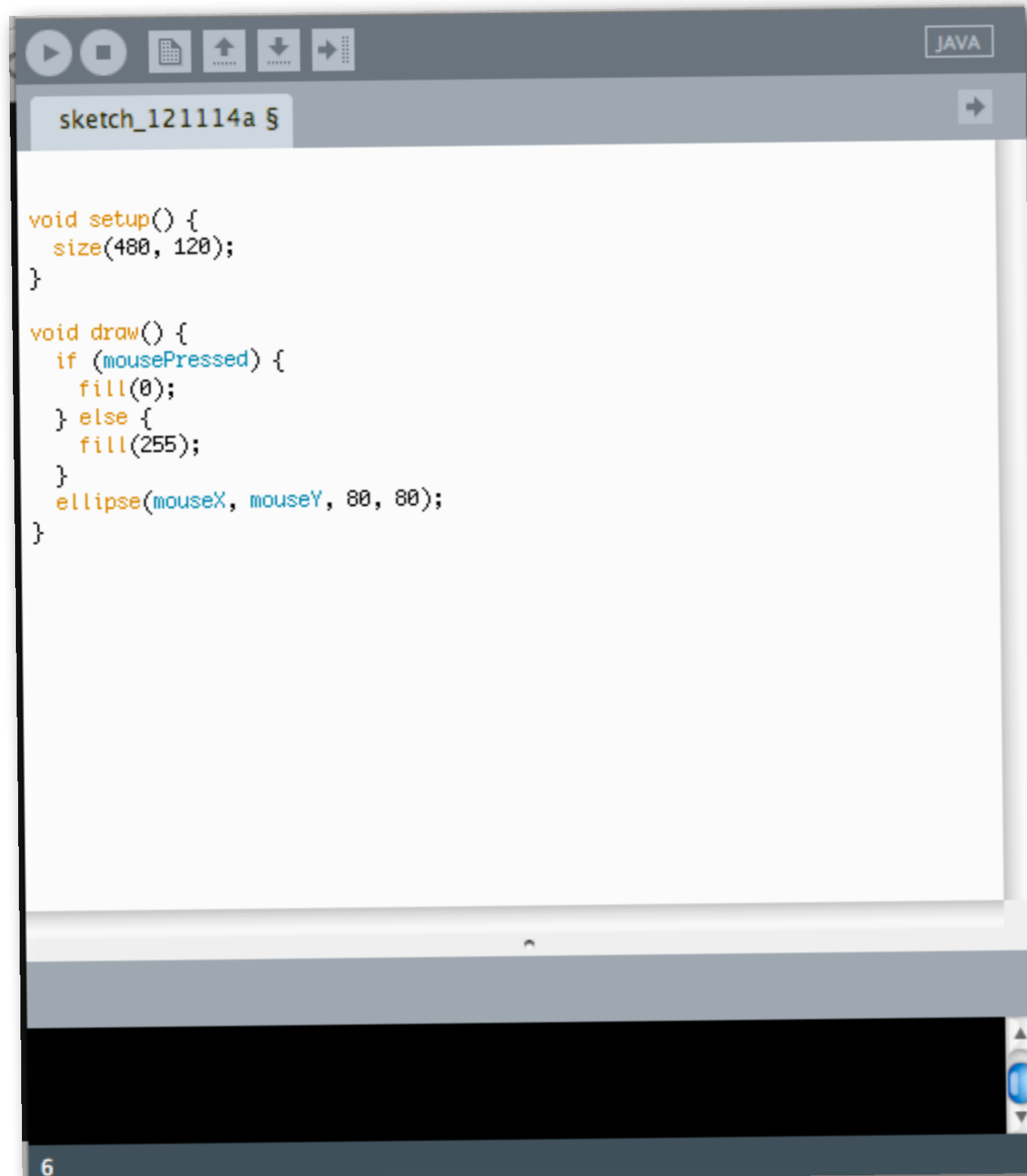
- You can write collections of commands that get run in particular ways by Processing
- the **setup** function is run once at the beginning
- the **draw** function is run repeatedly until the user hits stop

# Intro to Processing



```
void setup() {  
  size(480, 120);  
}  
  
void draw() {  
  if (mousePressed) {  
    fill(0);  
  } else {  
    fill(255);  
  }  
  ellipse(mouseX, mouseY, 80, 80);  
}
```

- Program Flow

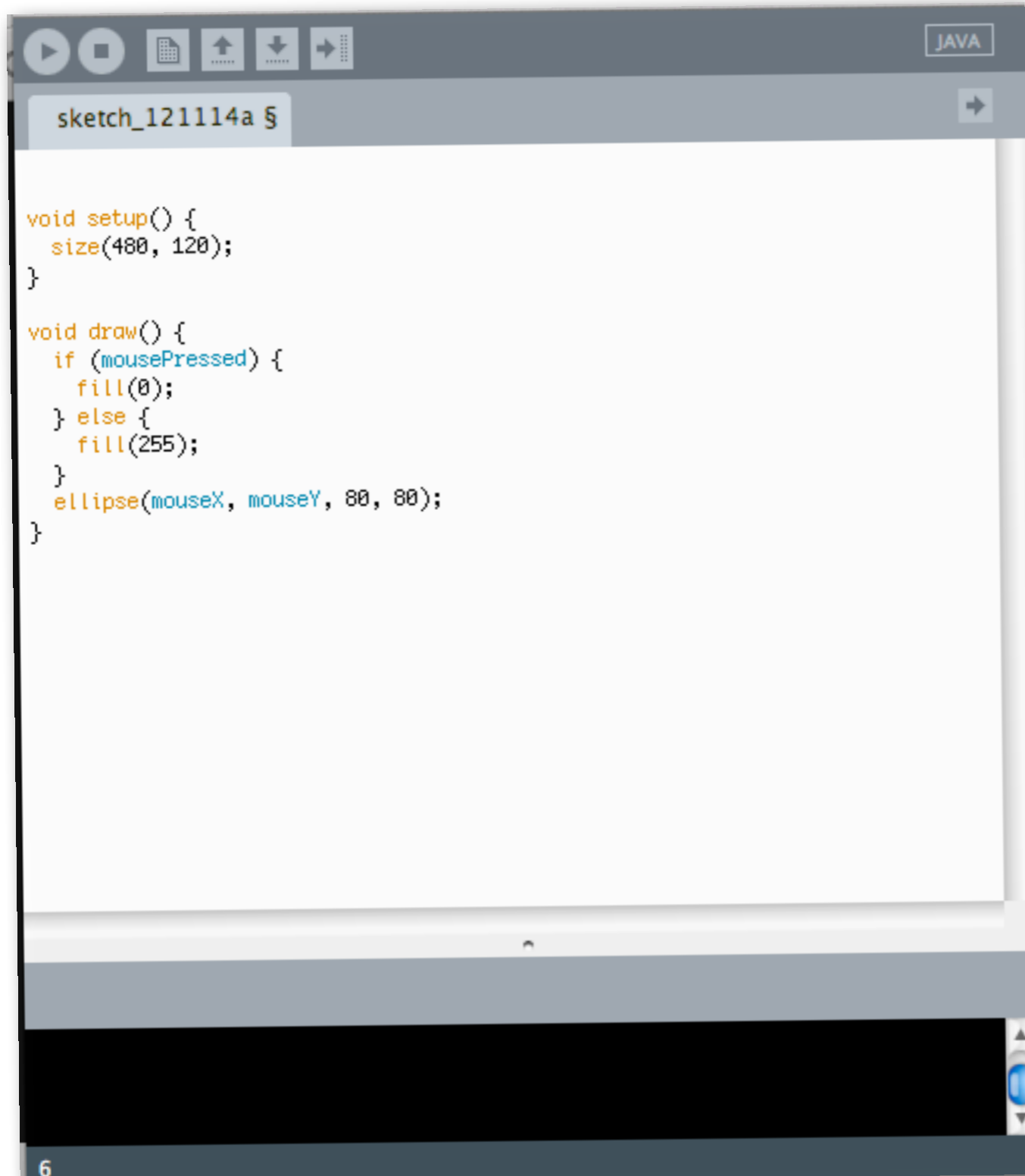


```
void setup() {  
  size(480, 120);  
}  
  
void draw() {  
  if (mousePressed) {  
    fill(0);  
  } else {  
    fill(255);  
  }  
  ellipse(mouseX, mouseY, 80, 80);  
}
```

The screenshot shows a Processing IDE window titled "sketch\_121114a §" with a "JAVA" label in the top right corner. The code is color-coded: keywords like "void", "if", "else", and "ellipse" are in orange, and variables like "mouseX", "mouseY", "mousePressed", and "fill" are in blue. The code defines a "setup" function to set the window size to 480x120 pixels, and a "draw" function that checks if the mouse is pressed. If pressed, the background is filled with black (0); otherwise, it is filled with white (255). An ellipse is drawn at the mouse cursor position with a width and height of 80 pixels.



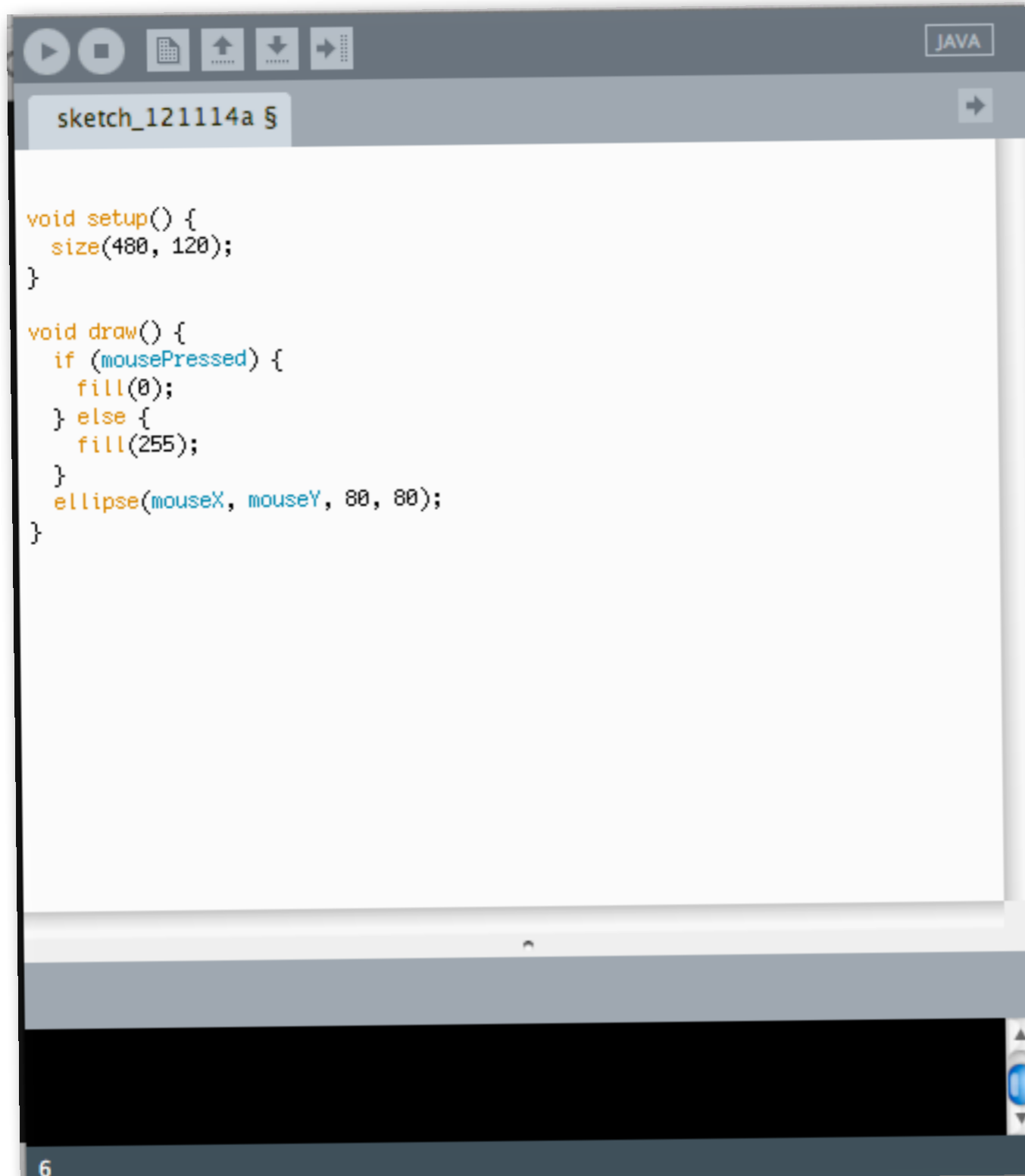
- Program Flow



```
void setup() {  
  size(480, 120);  
}  
  
void draw() {  
  if (mousePressed) {  
    fill(0);  
  } else {  
    fill(255);  
  }  
  ellipse(mouseX, mouseY, 80, 80);  
}
```

- functions use curly braces to hold all the commands

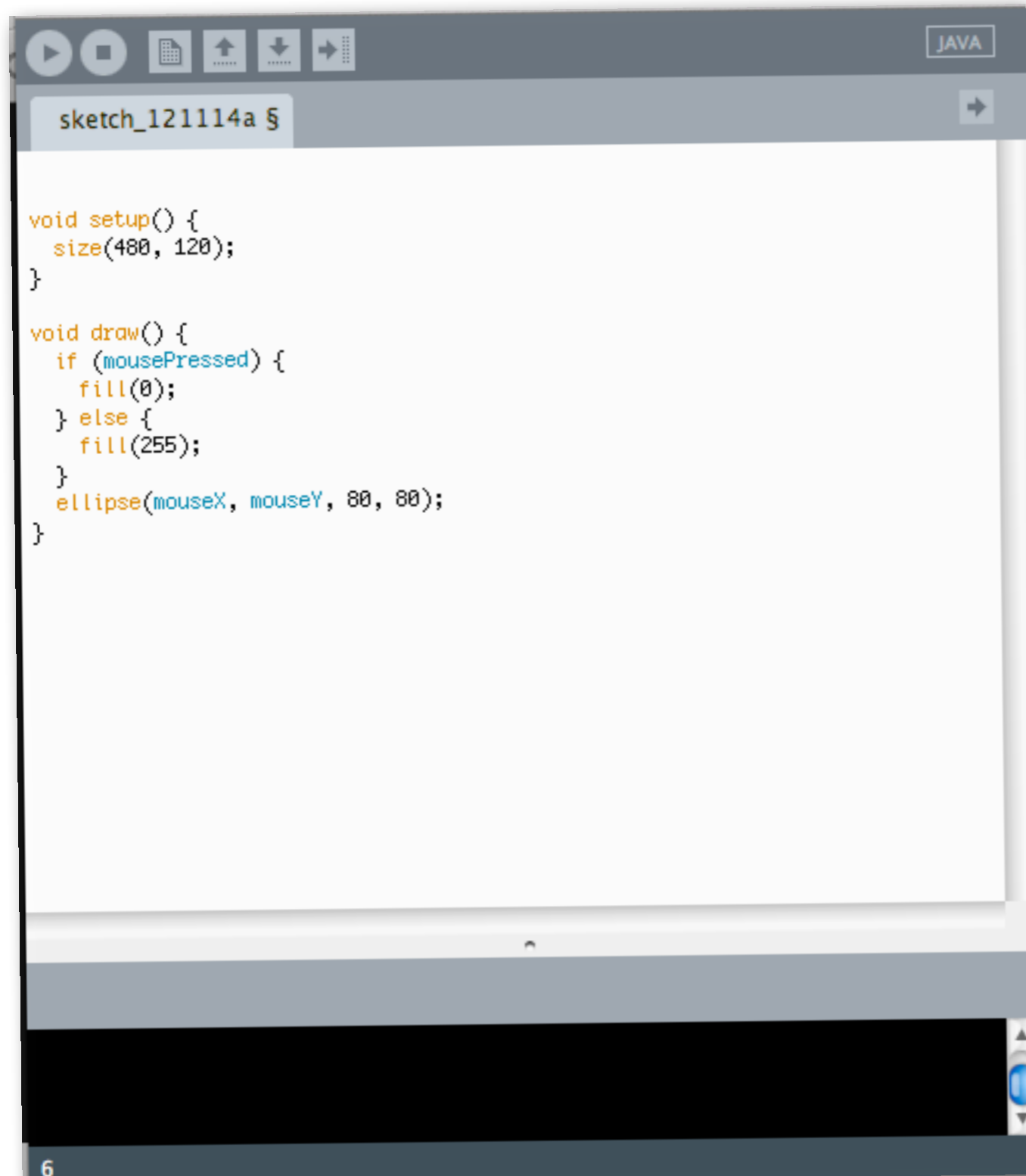
- Program Flow



```
void setup() {  
  size(480, 120);  
}  
  
void draw() {  
  if (mousePressed) {  
    fill(0);  
  } else {  
    fill(255);  
  }  
  ellipse(mouseX, mouseY, 80, 80);  
}
```

- functions use curly braces to hold all the commands
- **size()** changes the display window size

- Program Flow

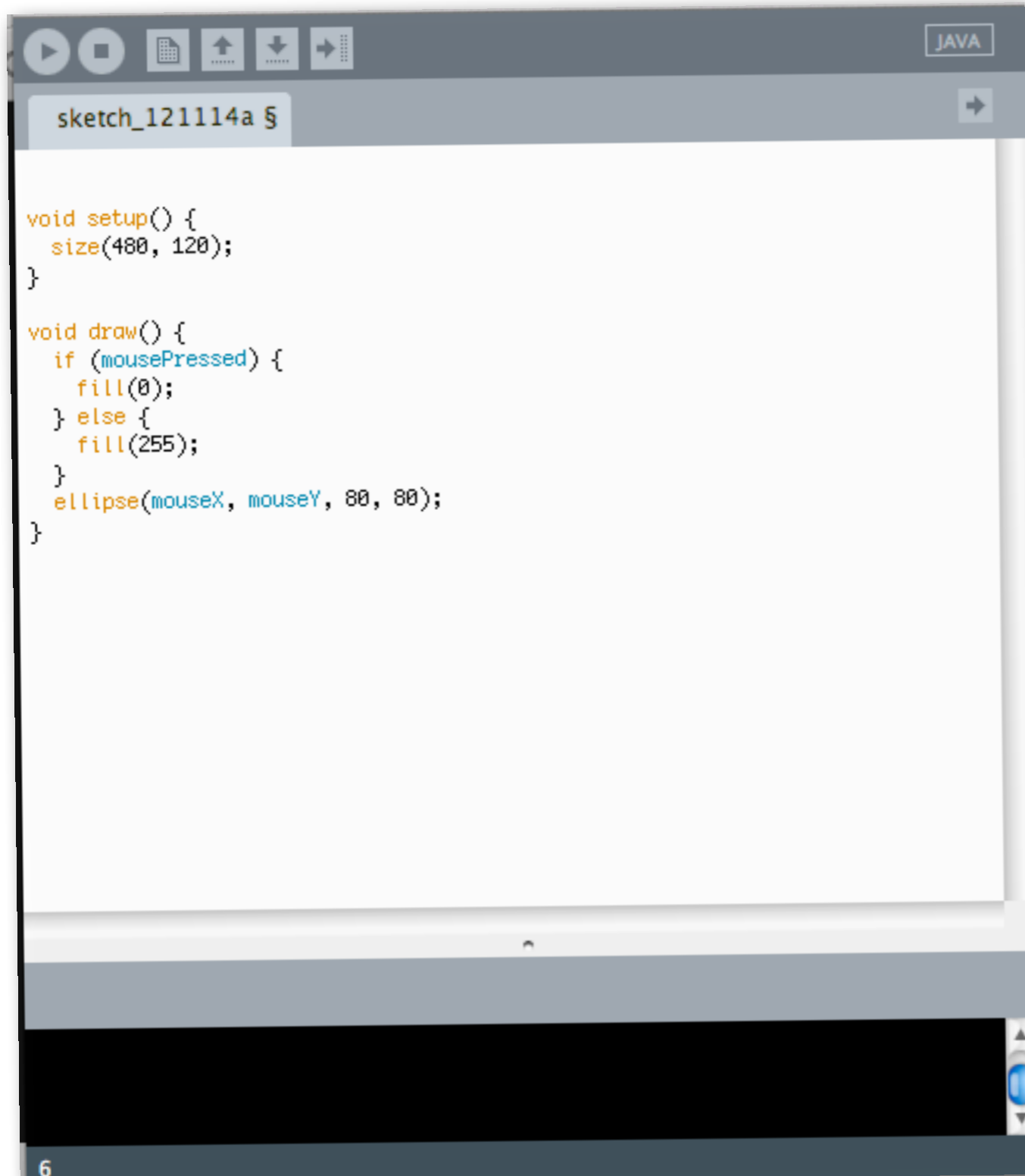


```
void setup() {
  size(480, 120);
}

void draw() {
  if (mousePressed) {
    fill(0);
  } else {
    fill(255);
  }
  ellipse(mouseX, mouseY, 80, 80);
}
```

- functions use curly braces to hold all the commands
- **size()** changes the display window size
- **mousePressed** is true if the user is pressing the mouse button

- Program Flow

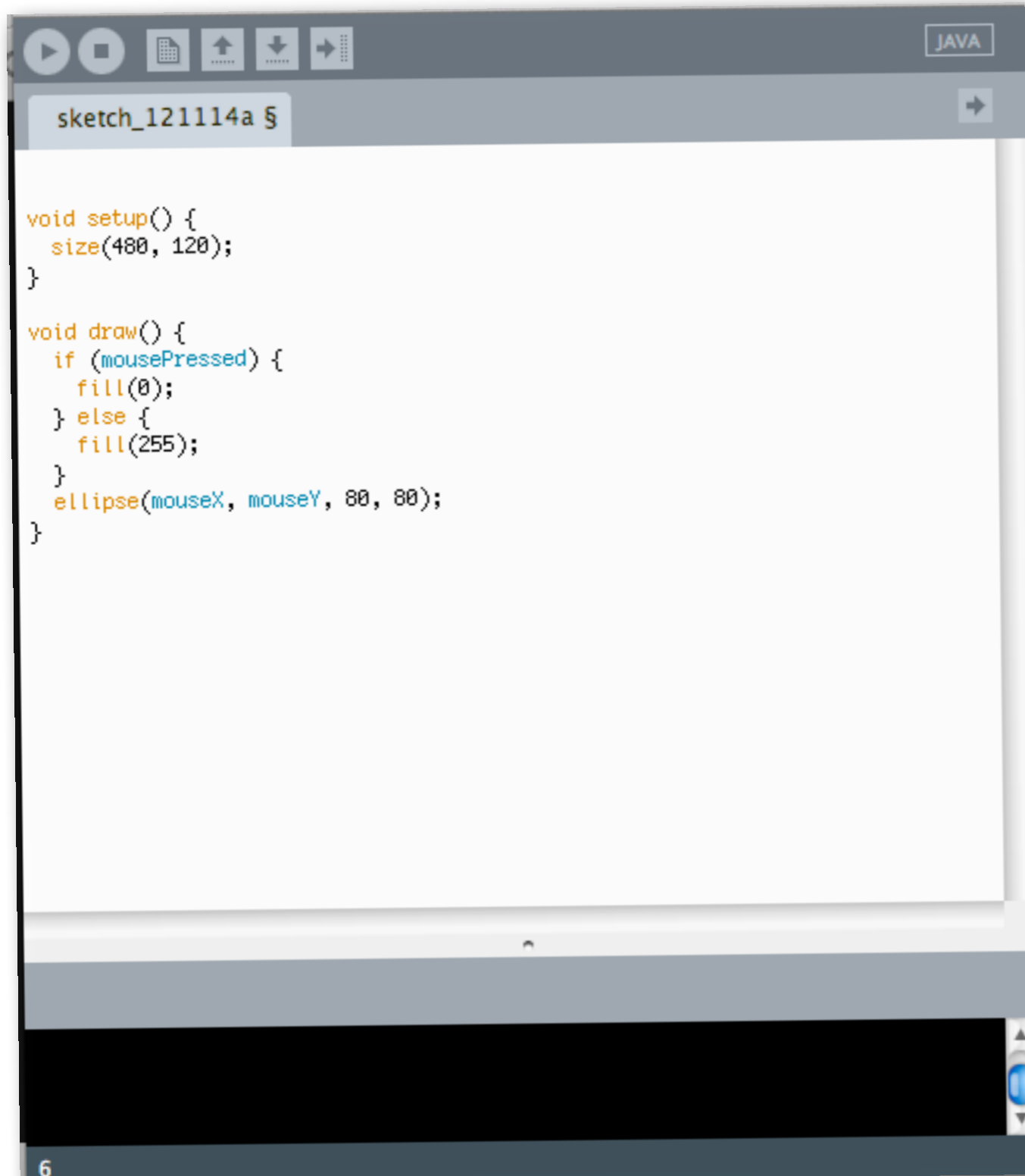


```
void setup() {
  size(480, 120);
}

void draw() {
  if (mousePressed) {
    fill(0);
  } else {
    fill(255);
  }
  ellipse(mouseX, mouseY, 80, 80);
}
```

- functions use curly braces to hold all the commands
- **size()** changes the display window size
- **mousePressed** is true if the user is pressing the mouse button
- **mouseX** and **mouseY** is the position of the mouse at the current time

- Program Flow

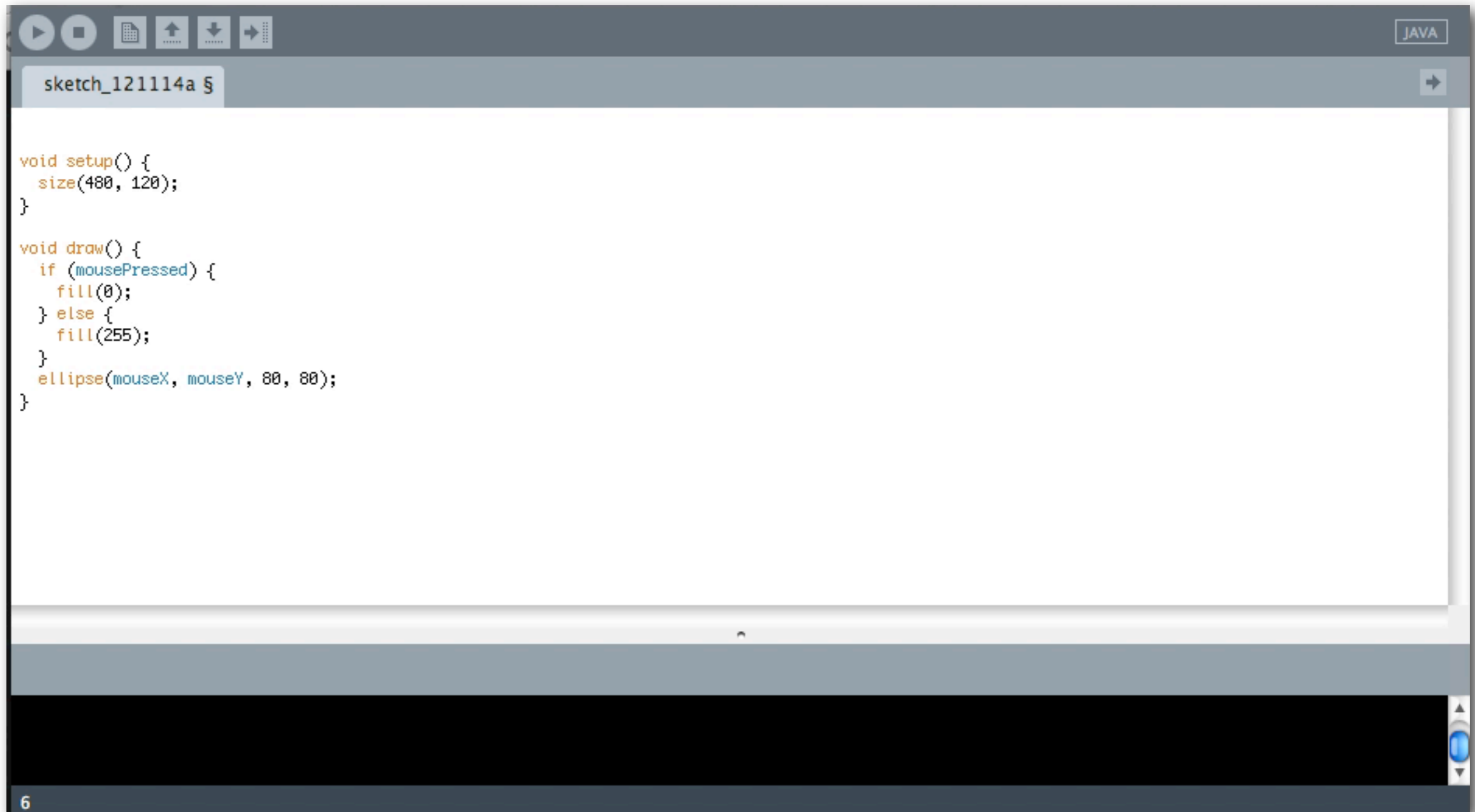


```
void setup() {
  size(480, 120);
}

void draw() {
  if (mousePressed) {
    fill(0);
  } else {
    fill(255);
  }
  ellipse(mouseX, mouseY, 80, 80);
}
```

- functions use curly braces to hold all the commands
- **size()** changes the display window size
- **mousePressed** is true if the user is pressing the mouse button
- **mouseX** and **mouseY** is the position of the mouse at the current time
- **fill()** changes the color inside the shape that gets drawn next

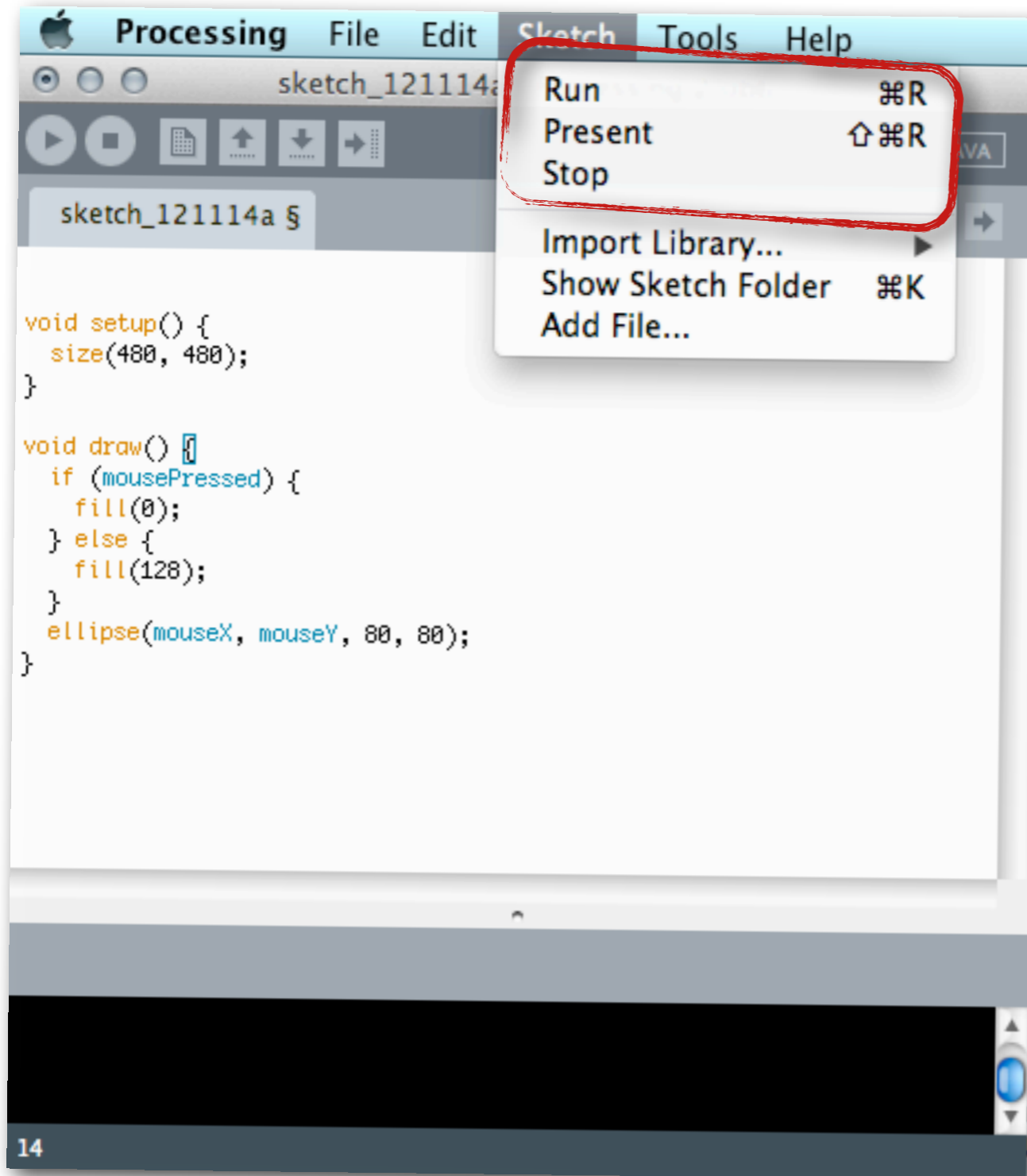
# Intro to Processing



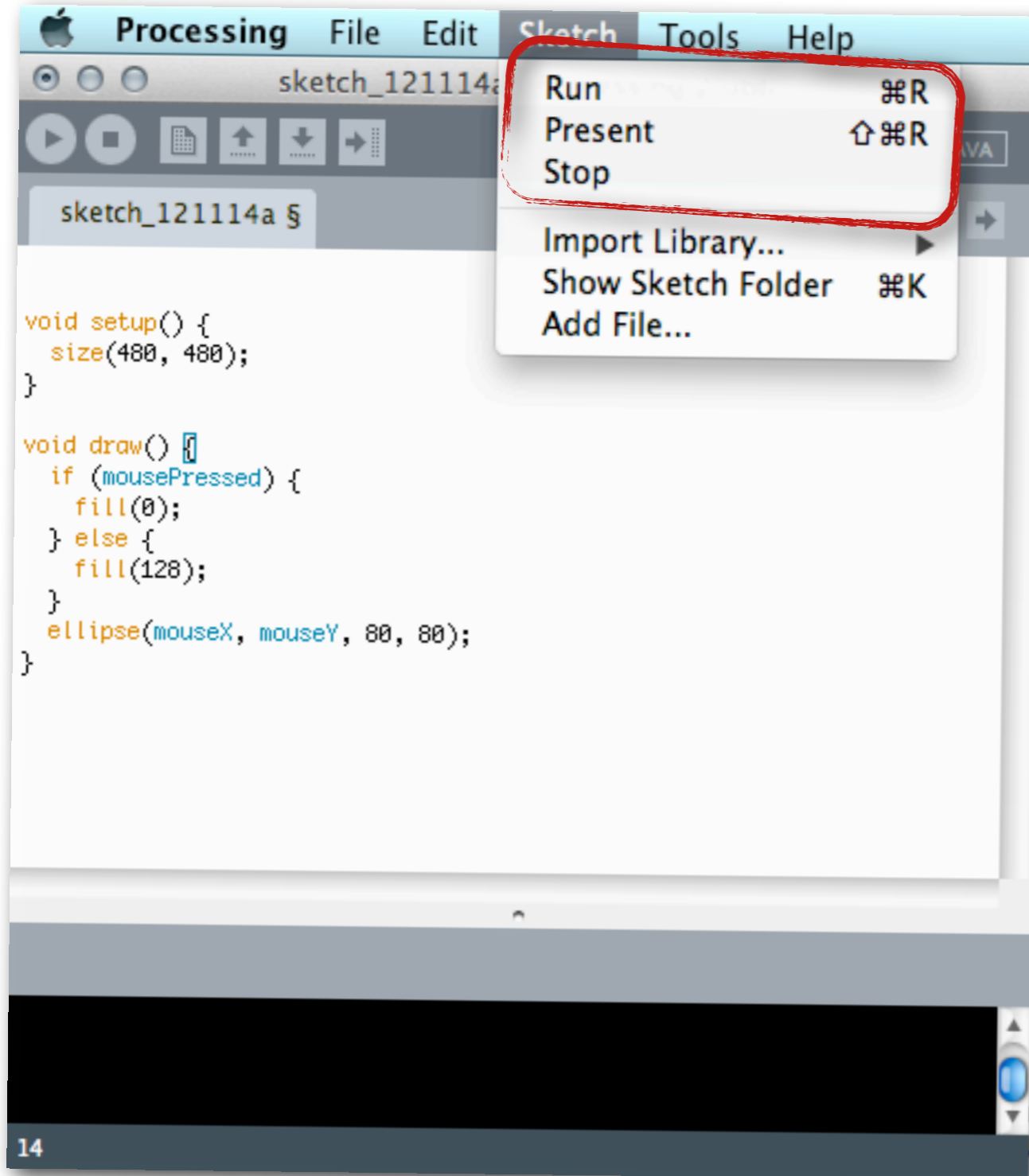
```
void setup() {  
  size(480, 120);  
}  
  
void draw() {  
  if (mousePressed) {  
    fill(0);  
  } else {  
    fill(255);  
  }  
  ellipse(mouseX, mouseY, 80, 80);  
}
```

6

# Intro to Processing

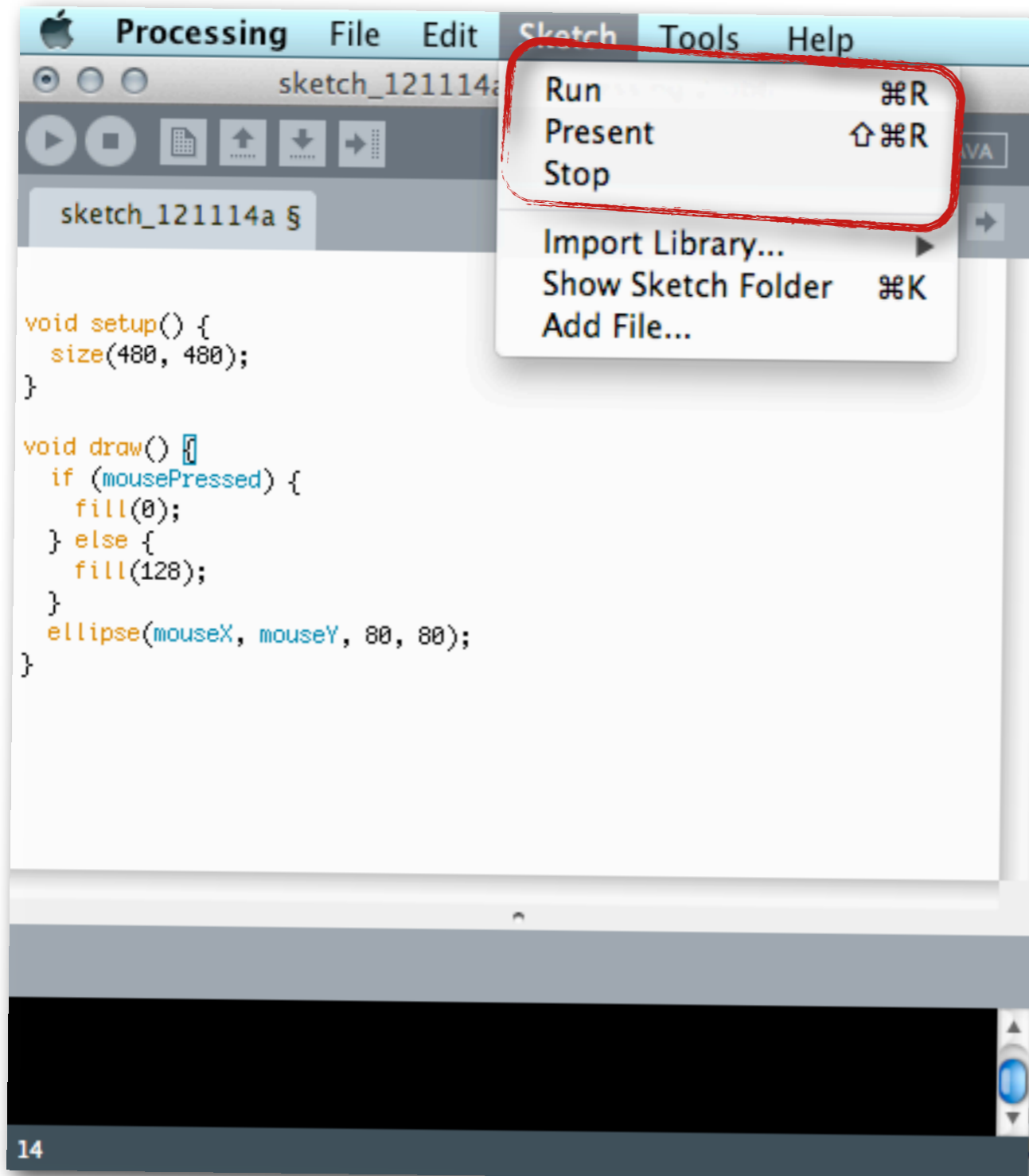


- Options



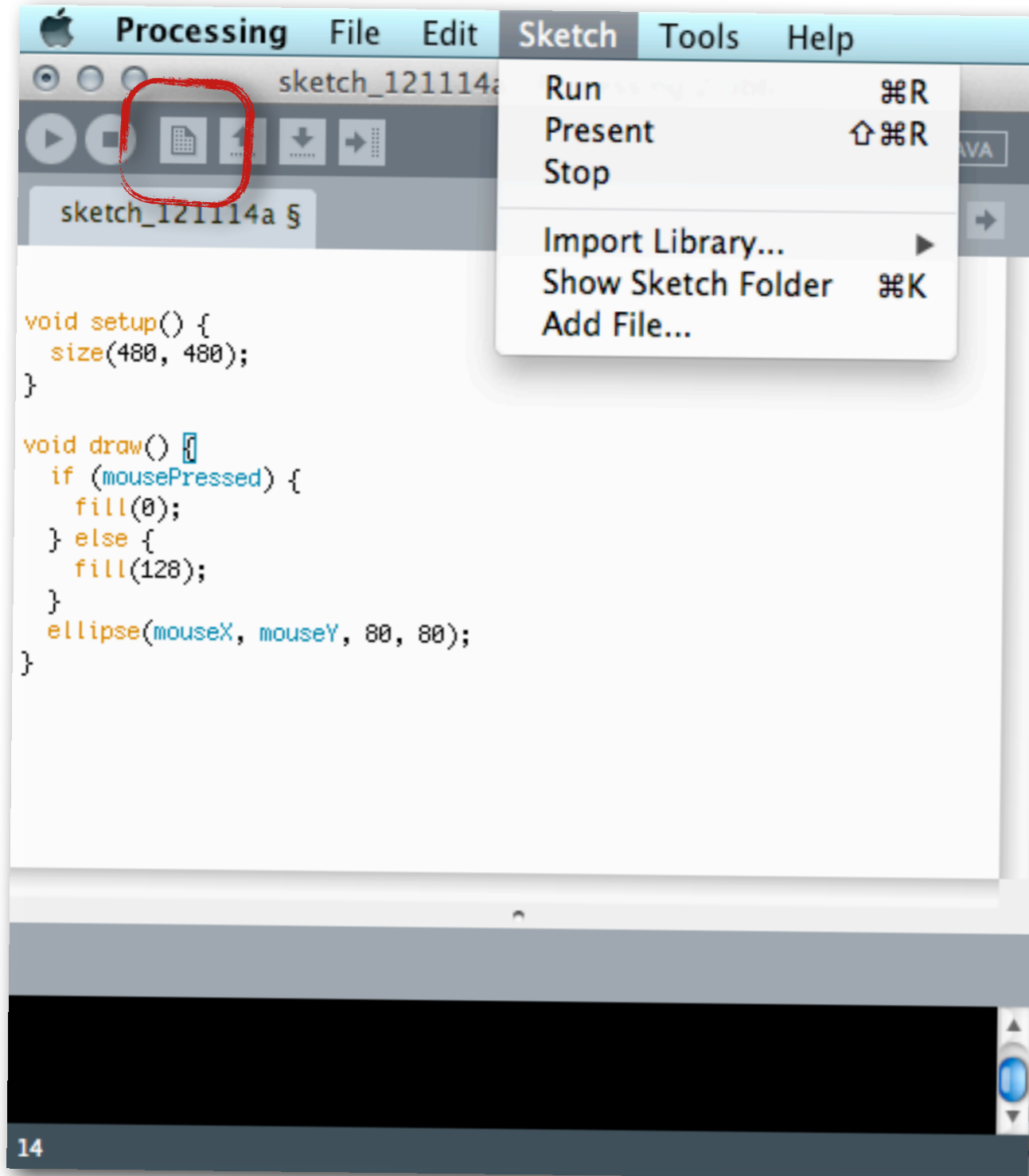


- Options

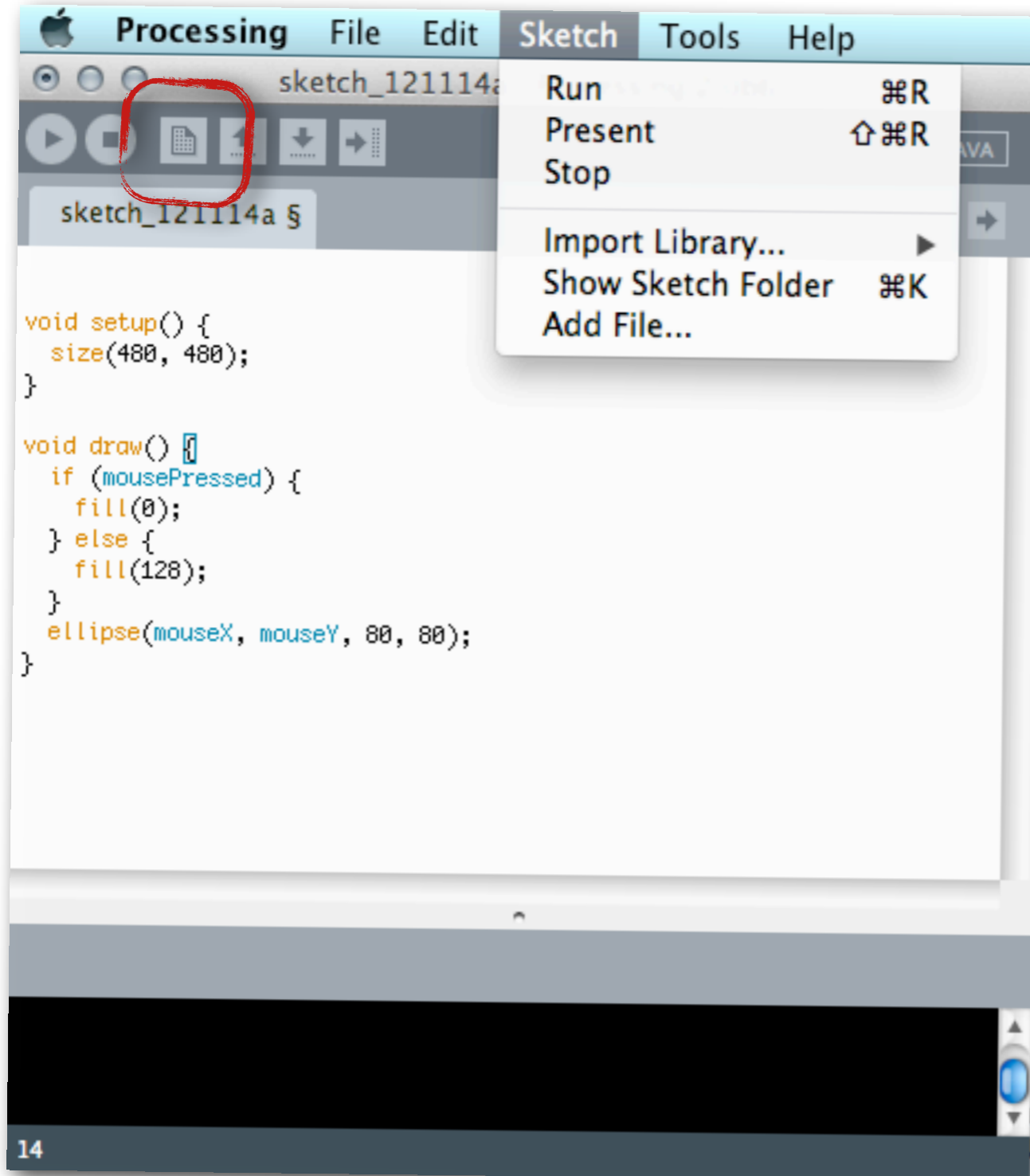


- Menu bar shows shortcuts instead of requiring the buttons to be used

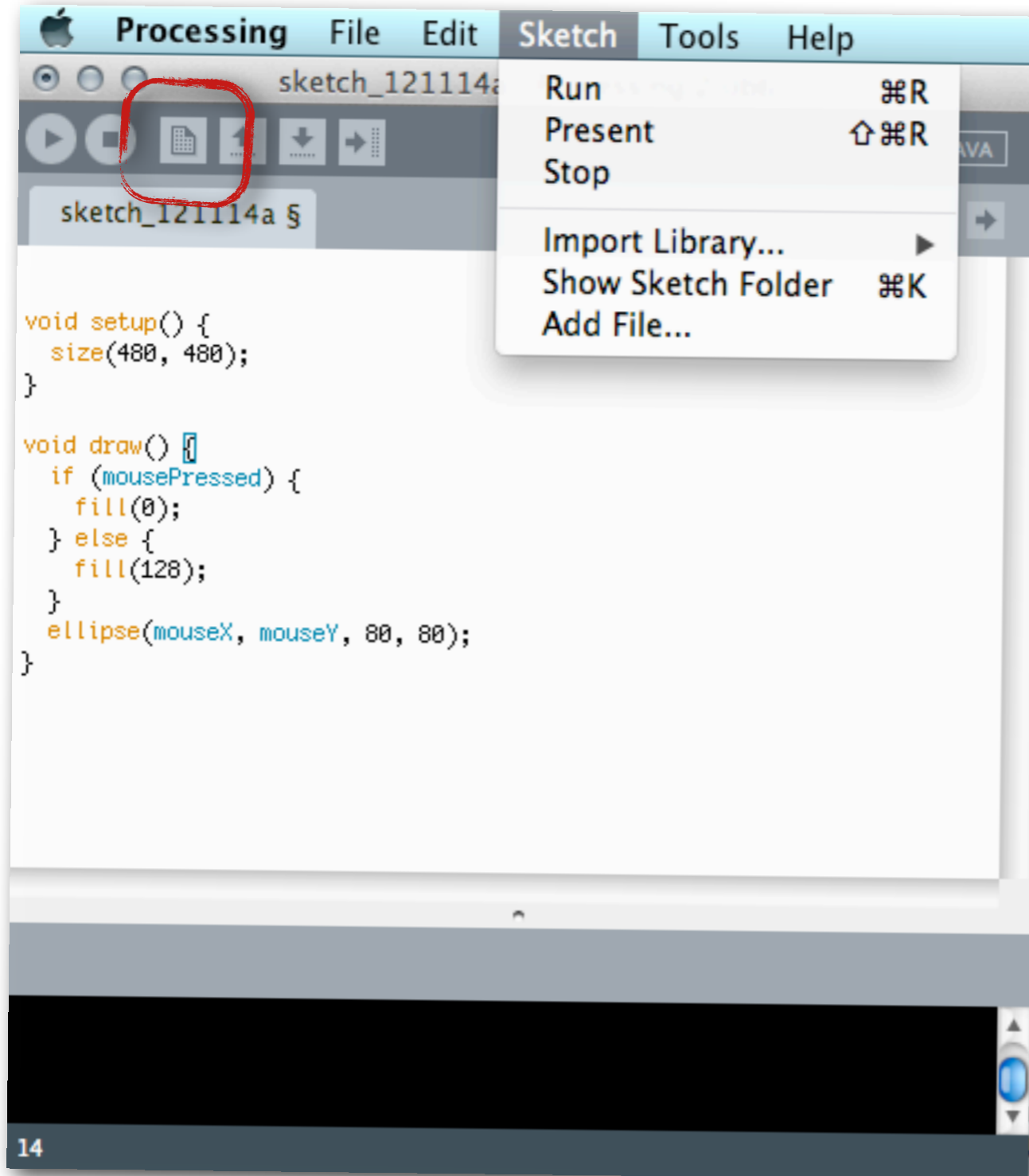
# Intro to Processing



- Options

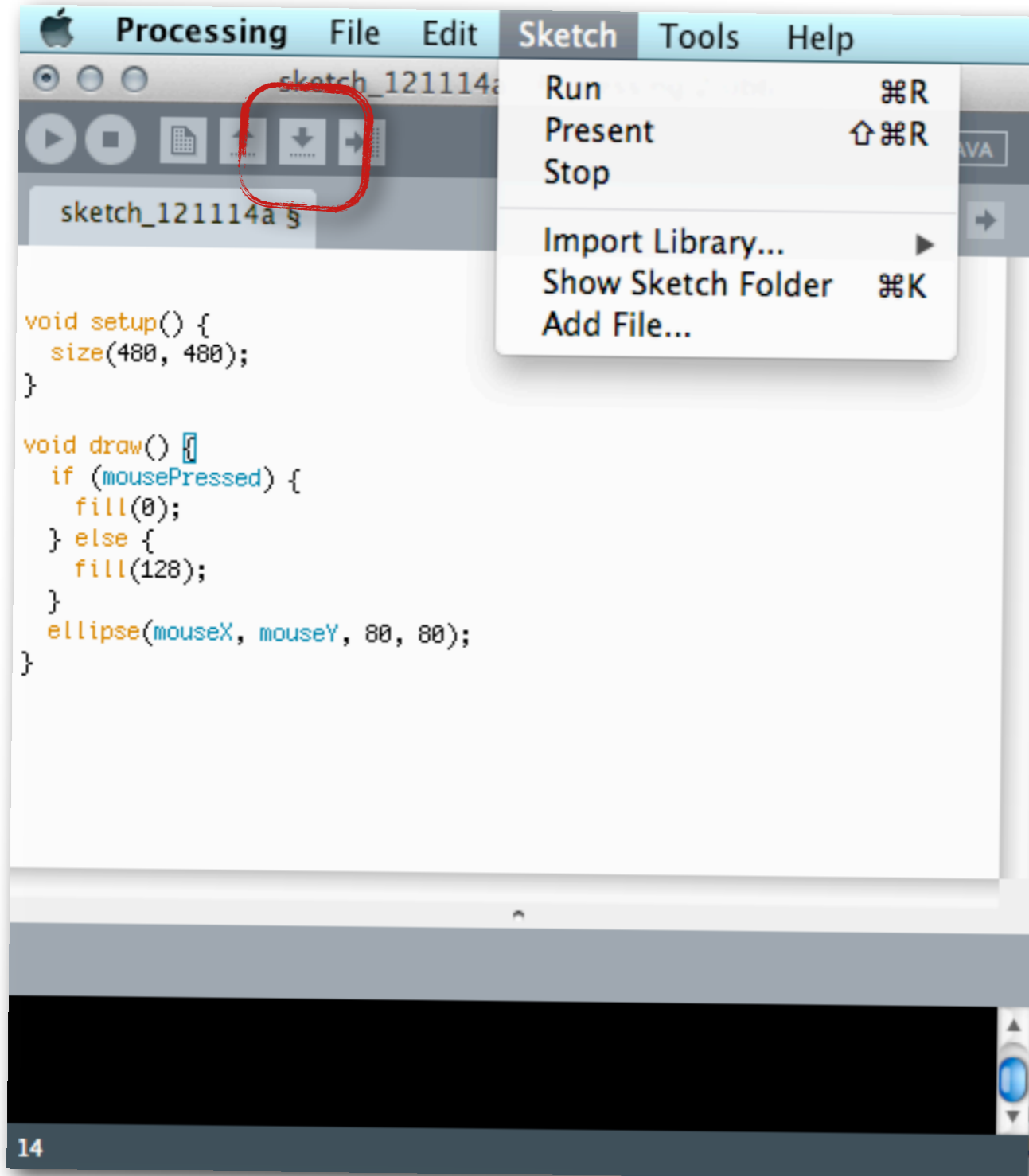


- Options

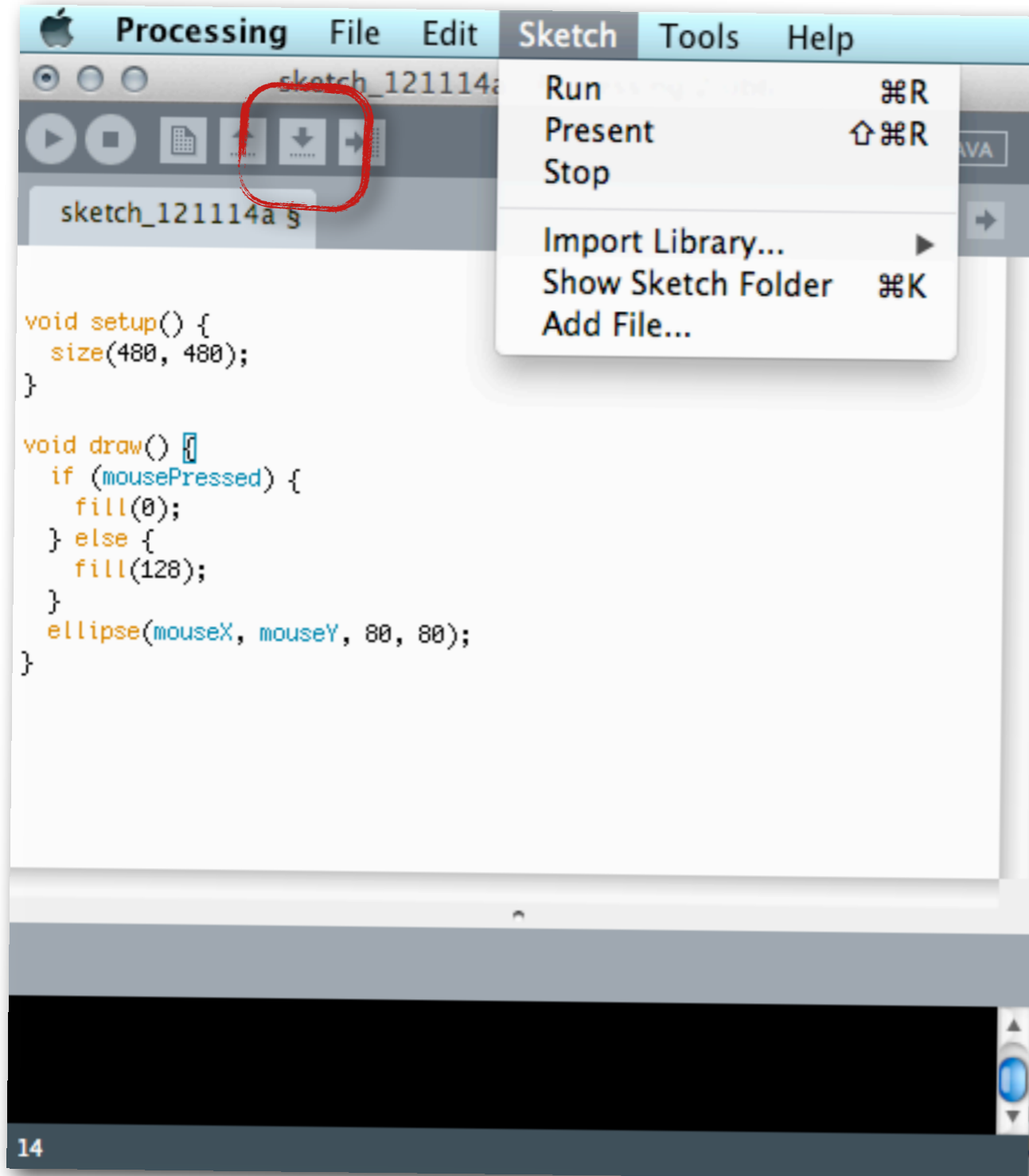


- Create a new sketch

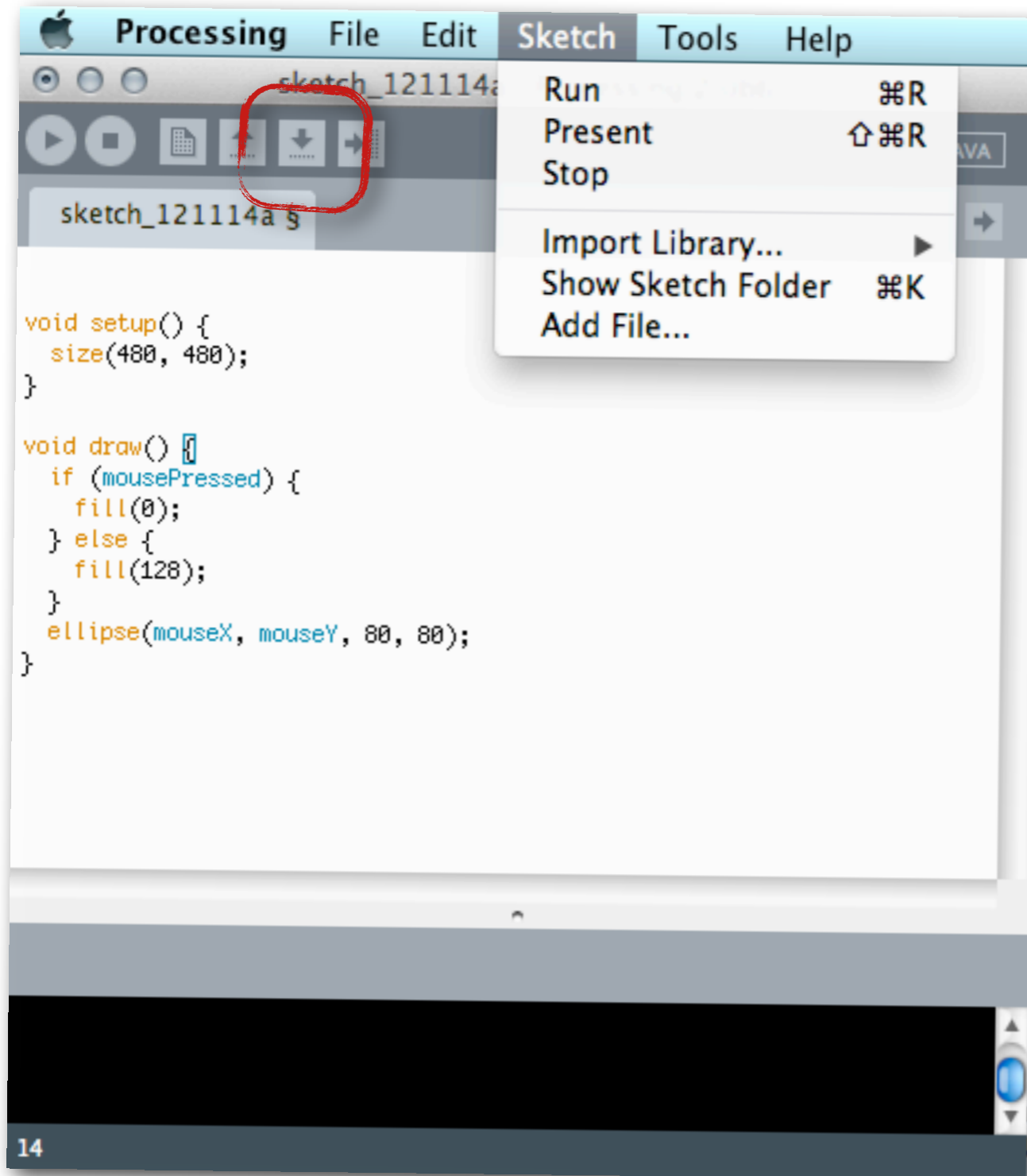
# Intro to Processing



- Options

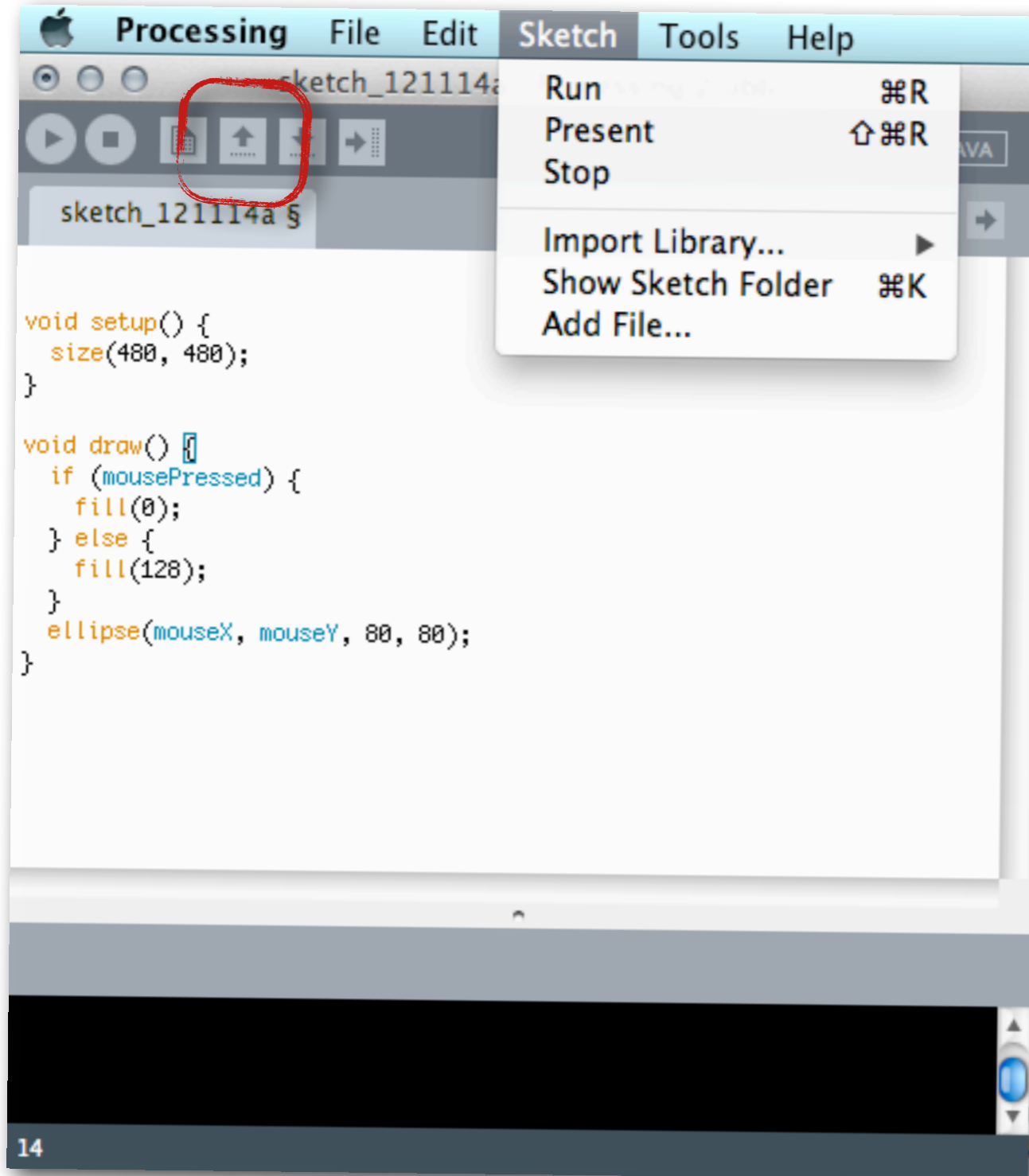


- Options



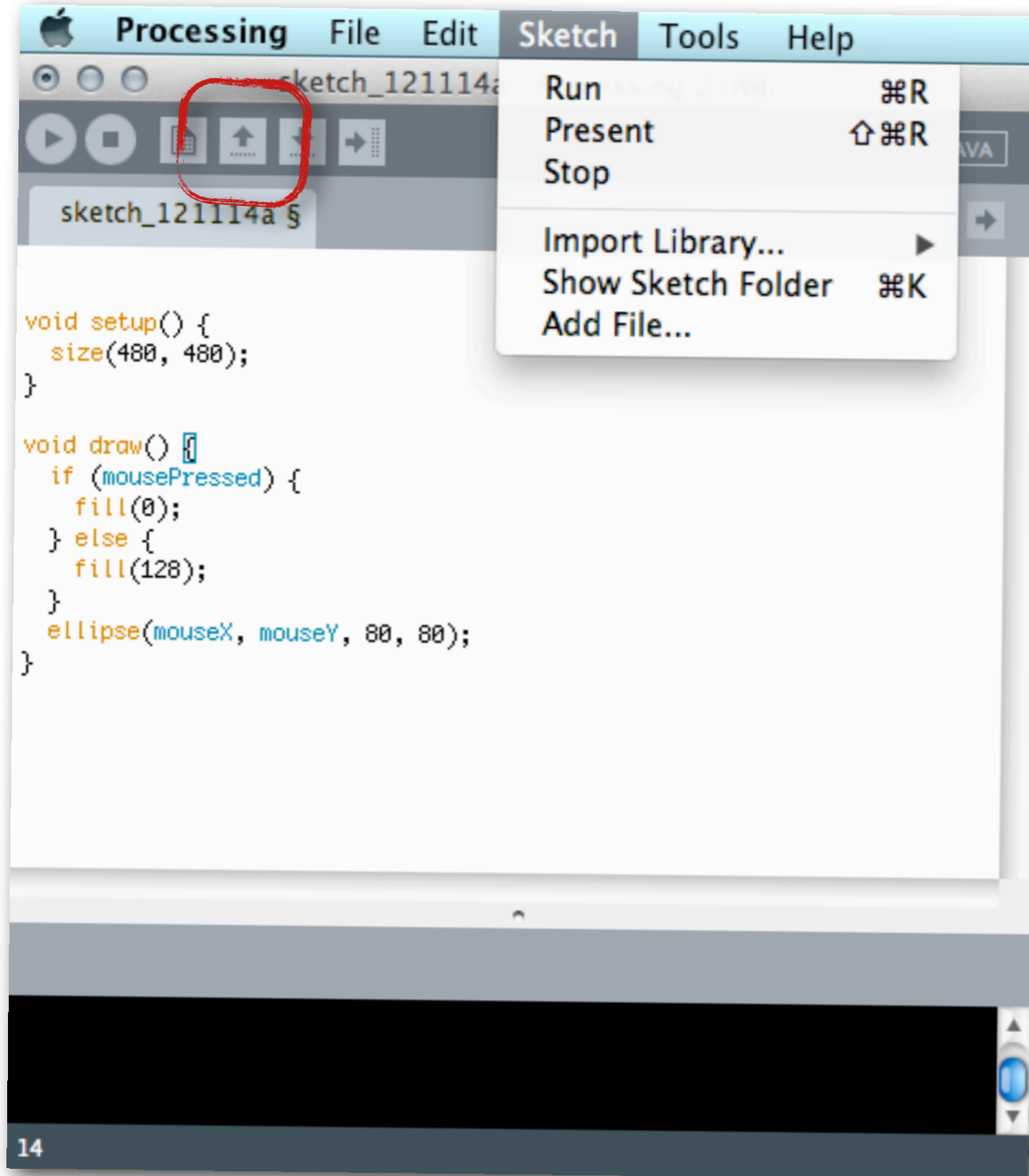
- Save the current sketch

# Intro to Processing

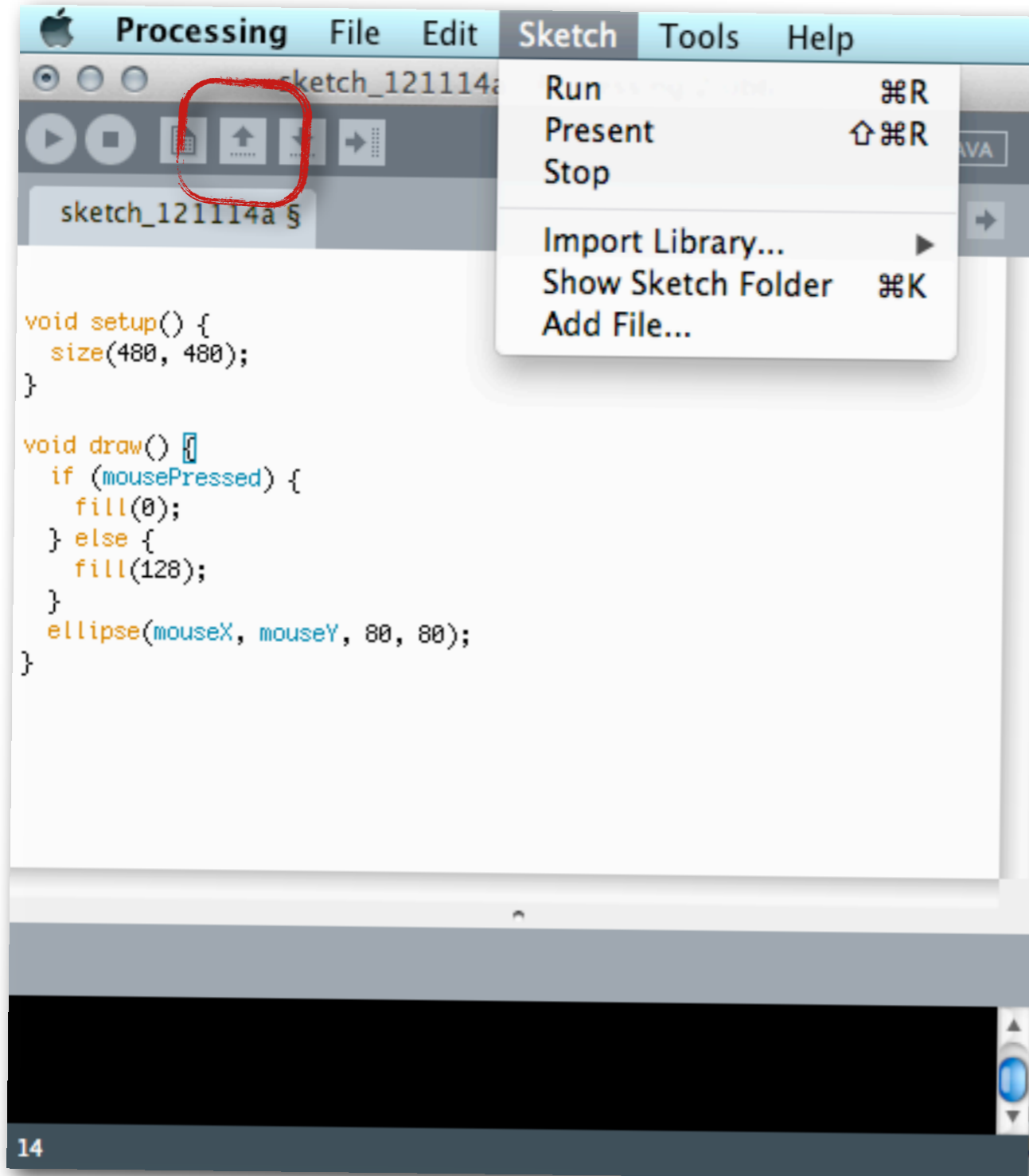




- Options

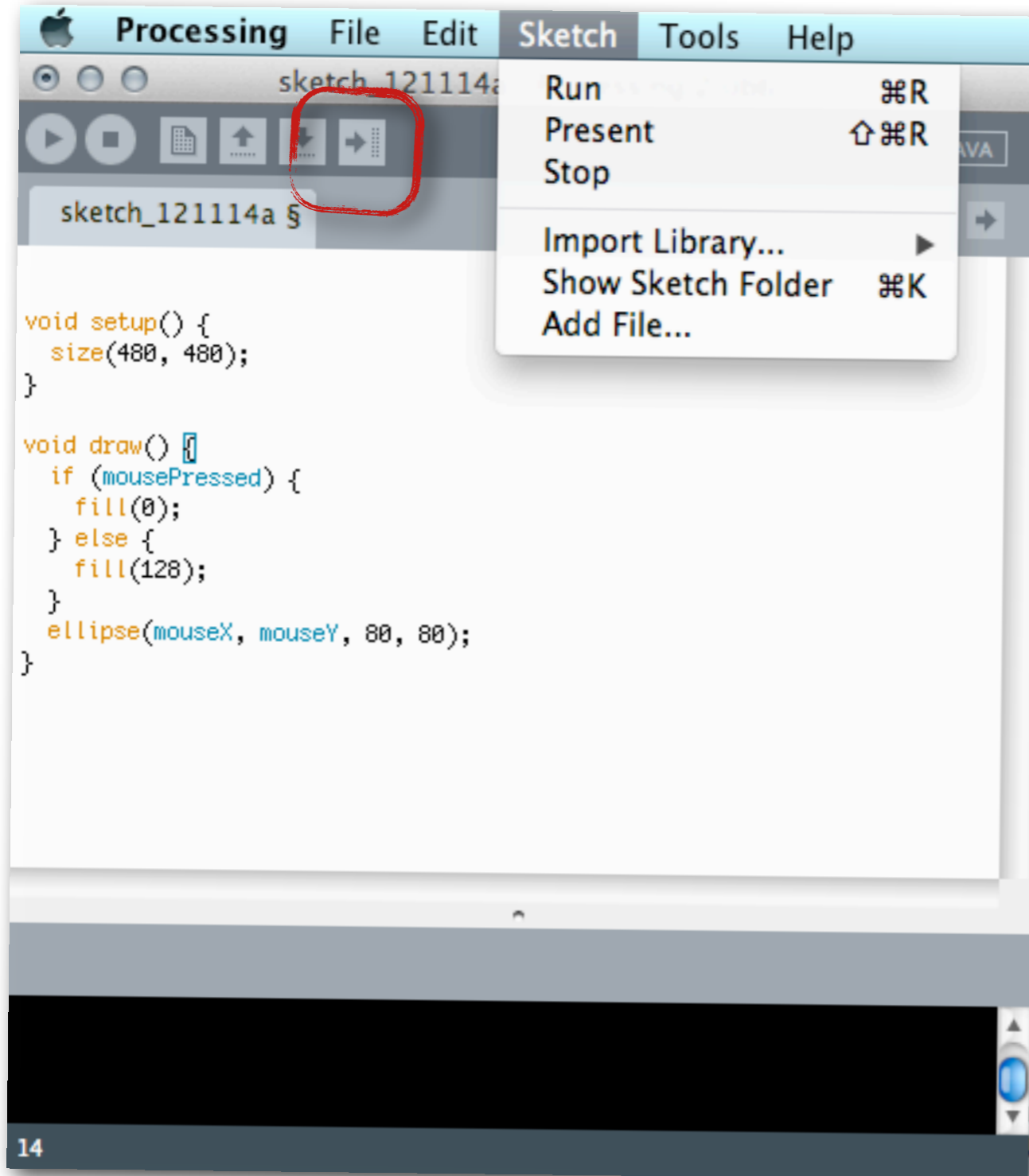


- Options

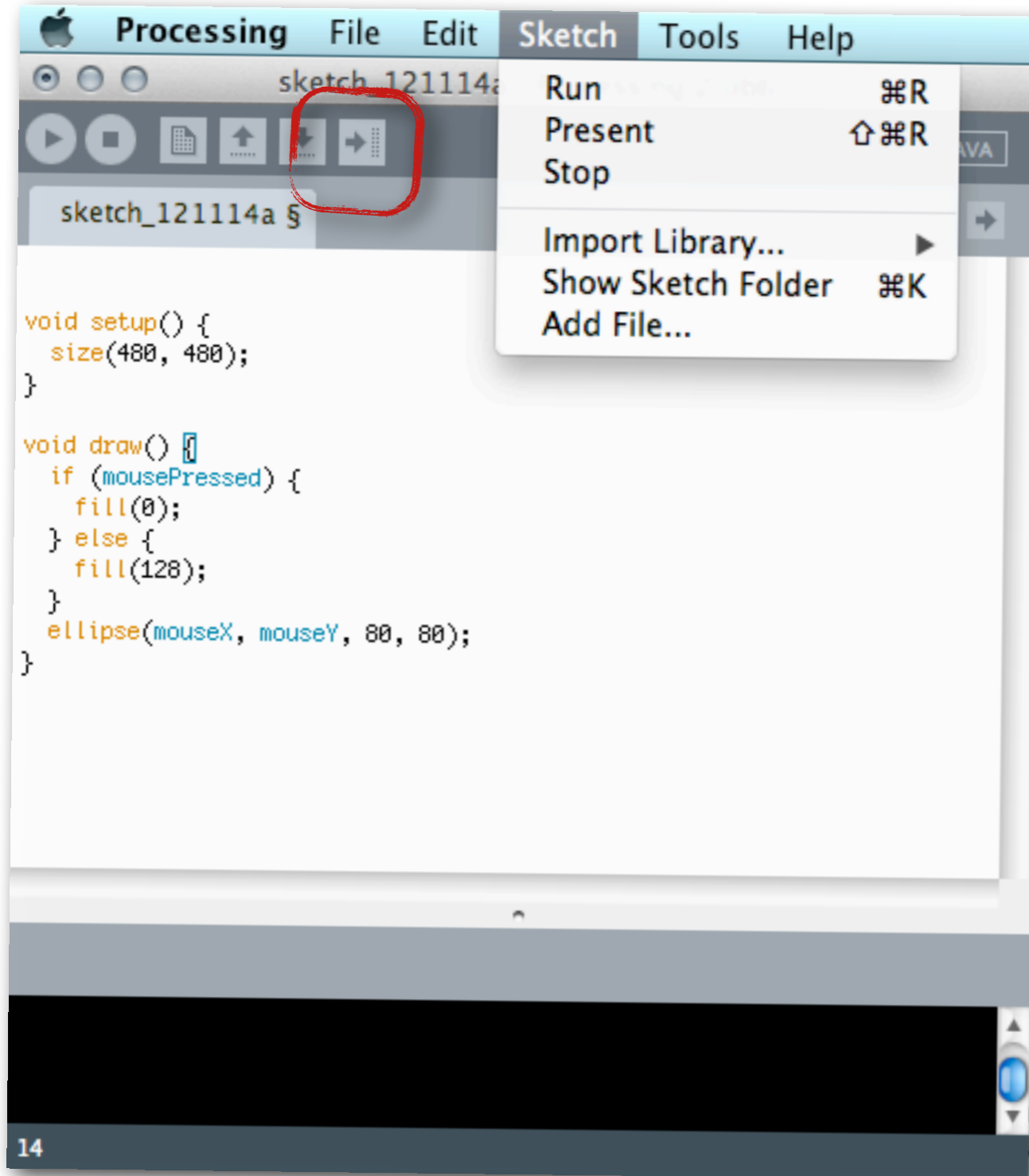


- Open a previously saved sketch

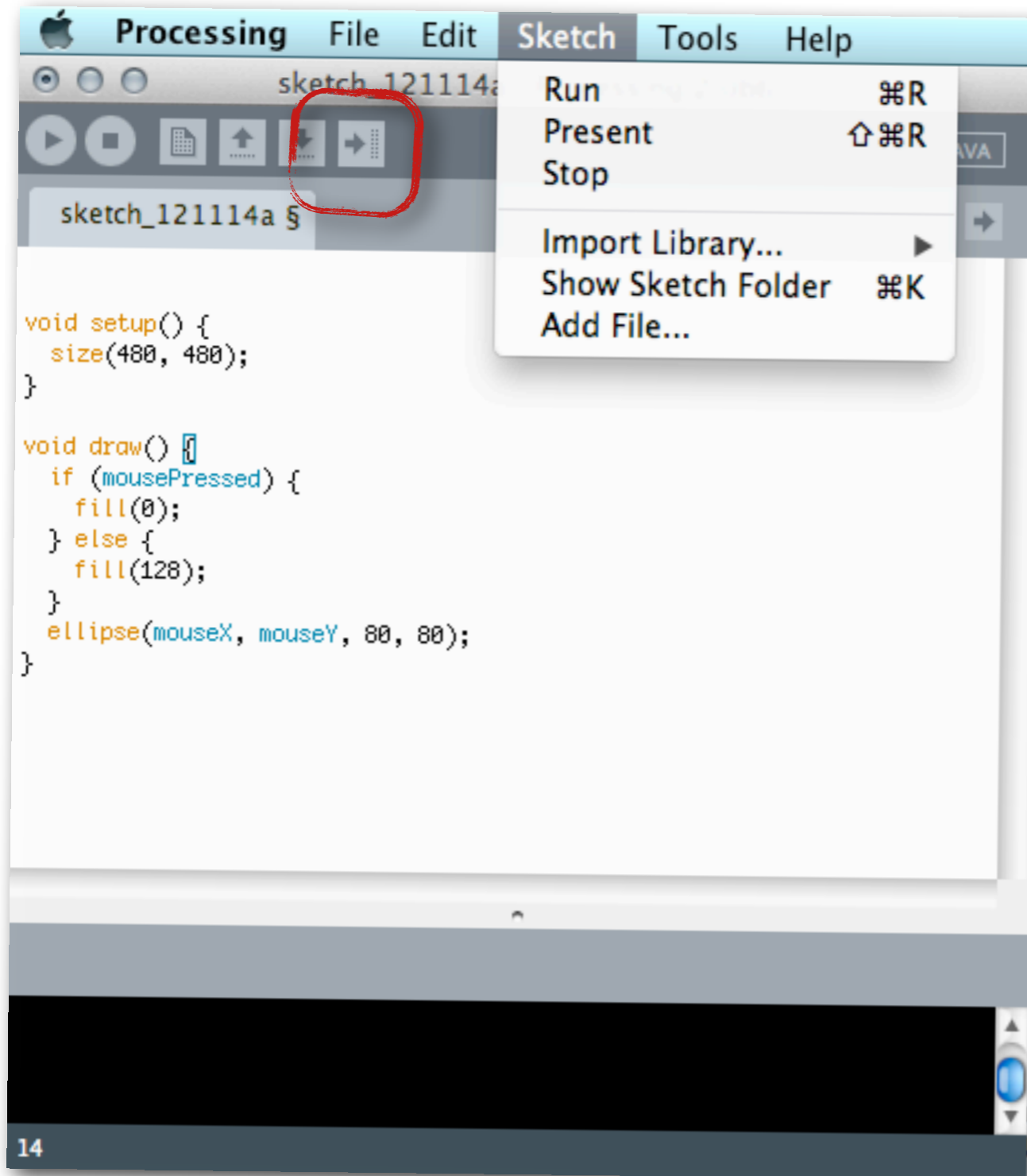
# Intro to Processing



- Options

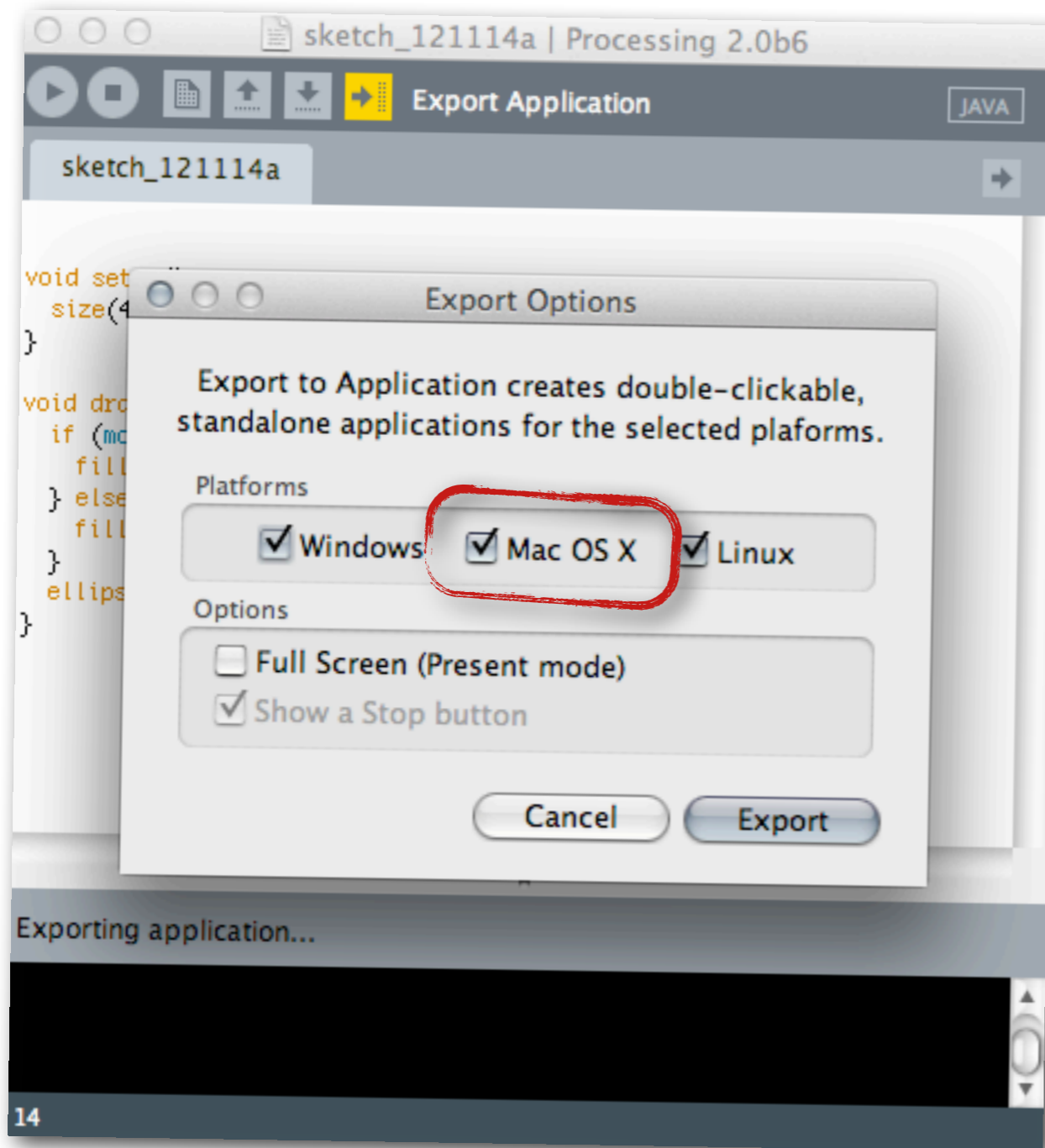


- Options

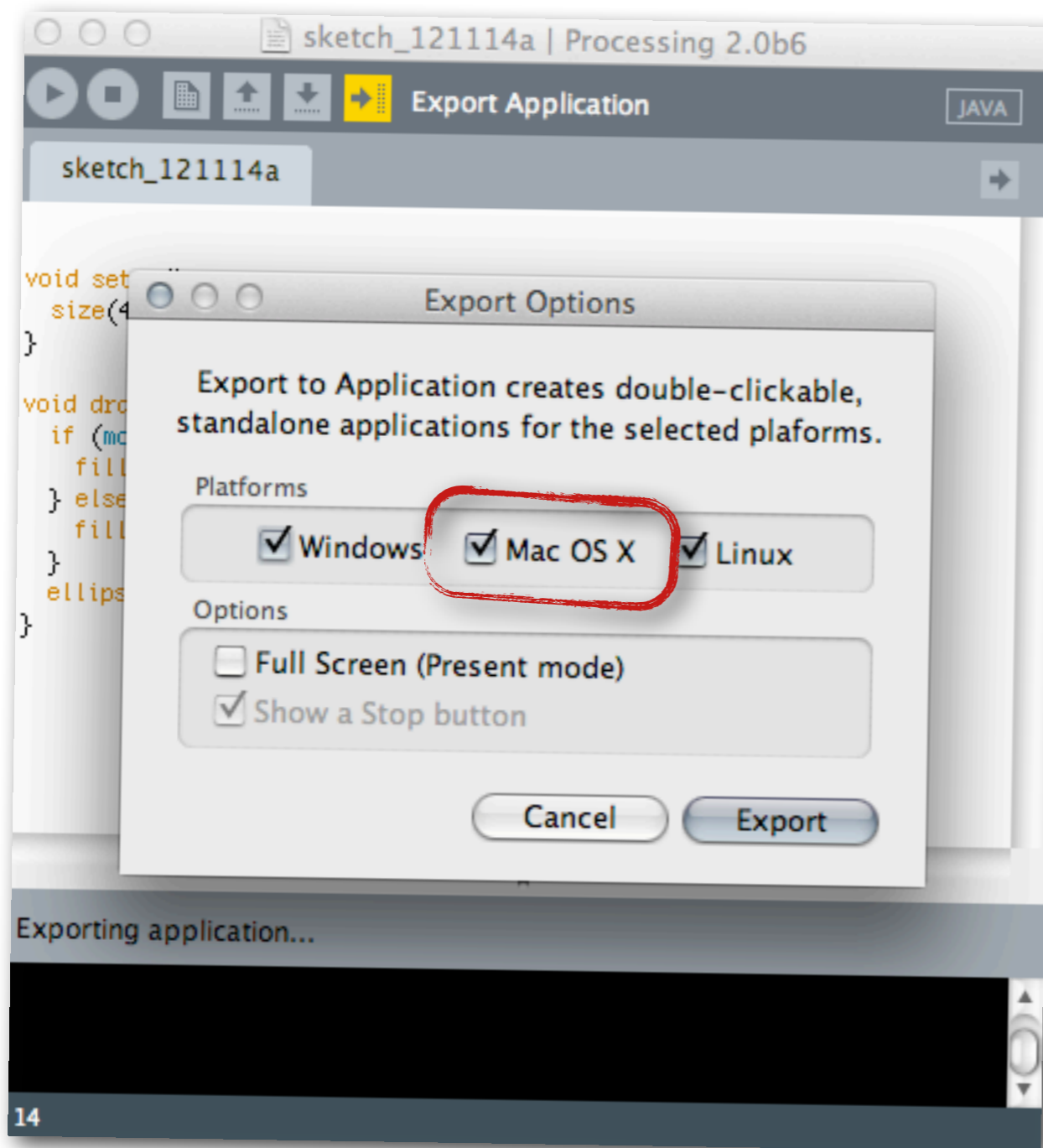


- Share a sketch as an application

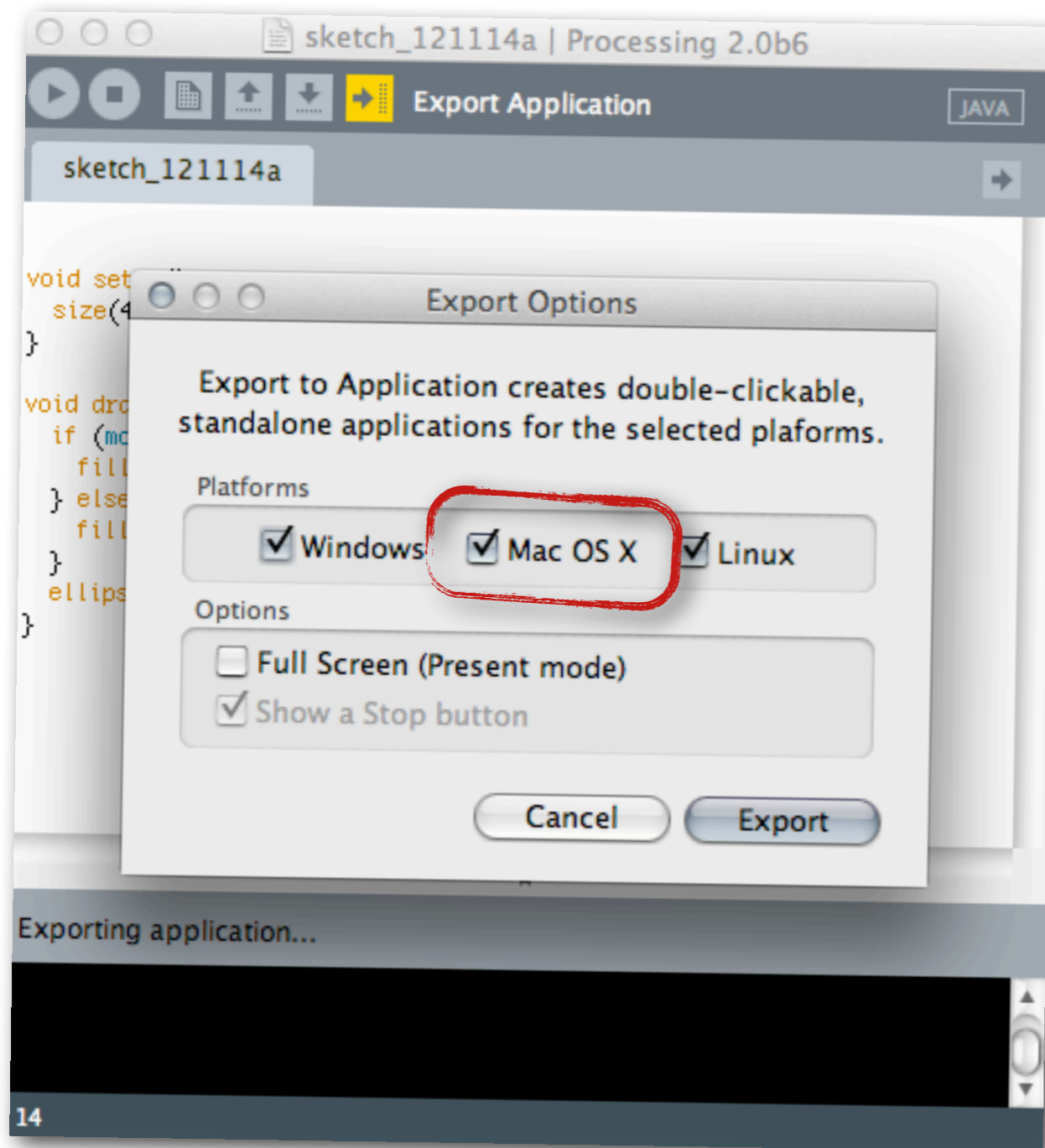
# Intro to Processing



- Options



- Options



- Creates an application that you can share (or turn in)



# Intro to Processing

← → ↻ [processing.org/reference/](http://processing.org/reference/)

← Processing Search

[Cover](#) \ [Exhibition](#) \ [Reference](#) \ [Learning](#) \ [Download](#) \ [Shop](#) \ [About](#) [»Feed](#) [»Forum](#) [»Wiki](#) [»Code](#)

↳ [Language \(A-Z\)](#) \ [Libraries](#) \ [Tools](#) \ [Environment](#)

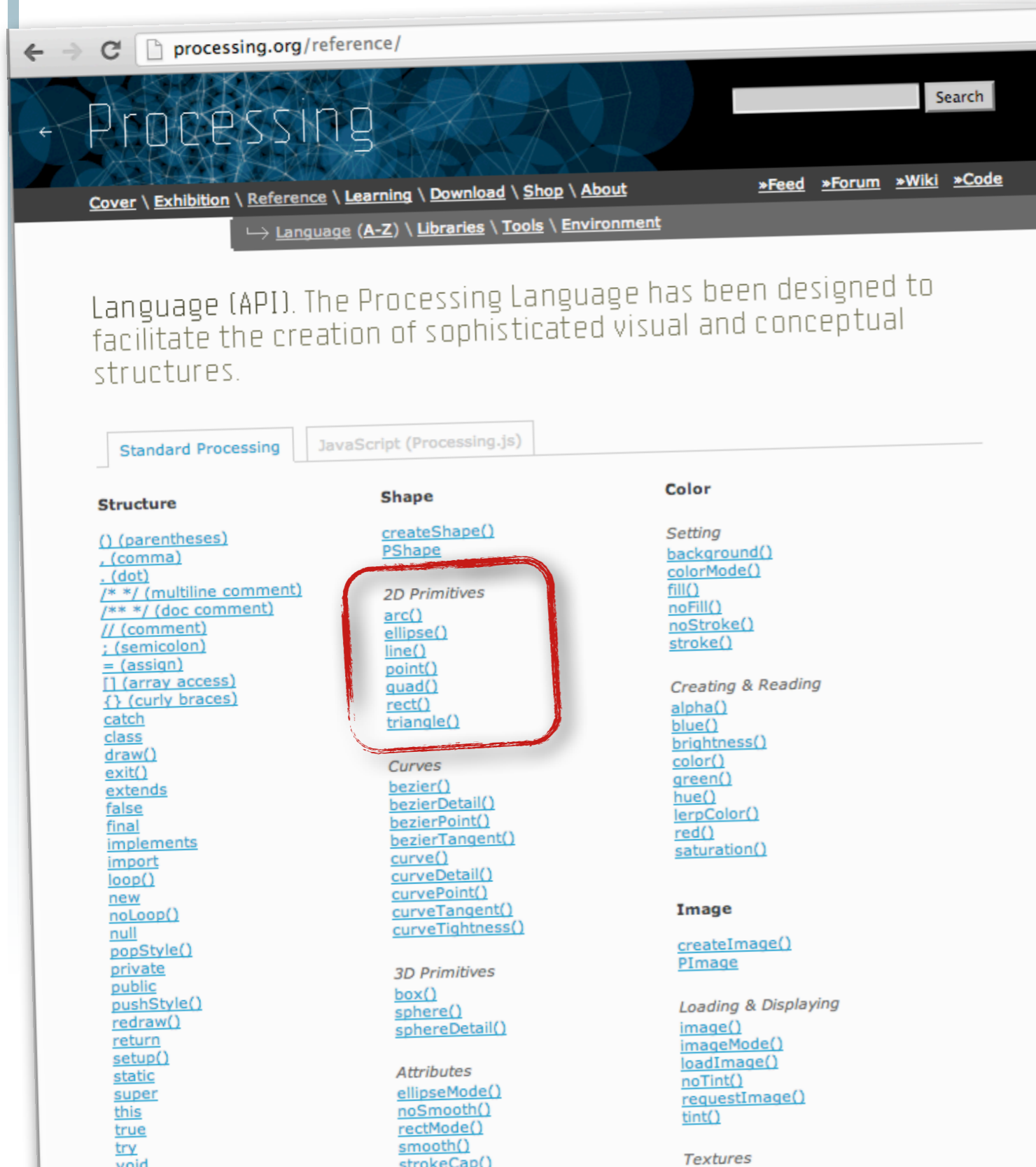
Language (API). The Processing Language has been designed to facilitate the creation of sophisticated visual and conceptual structures.

[Standard Processing](#) [JavaScript \(Processing.js\)](#)

Structure	Shape	Color
<a href="#">()</a> (parentheses) <a href="#">,</a> (comma) <a href="#">.</a> (dot) <a href="#">/* */</a> (multiline comment) <a href="#">/** */</a> (doc comment) <a href="#">//</a> (comment) <a href="#">;</a> (semicolon) <a href="#">=</a> (assign) <a href="#">[]</a> (array access) <a href="#">{ }</a> (curly braces) <a href="#">catch</a> <a href="#">class</a> <a href="#">draw()</a> <a href="#">exit()</a> <a href="#">extends</a> <a href="#">false</a> <a href="#">final</a> <a href="#">implements</a> <a href="#">import</a> <a href="#">loop()</a> <a href="#">new</a> <a href="#">noLoop()</a> <a href="#">null</a> <a href="#">popStyle()</a> <a href="#">private</a> <a href="#">public</a> <a href="#">pushStyle()</a> <a href="#">redraw()</a> <a href="#">return</a> <a href="#">setup()</a> <a href="#">static</a> <a href="#">super</a> <a href="#">this</a> <a href="#">true</a> <a href="#">try</a> <a href="#">void</a>	<a href="#">createShape()</a> <a href="#">PShape</a>  <b>2D Primitives</b> <a href="#">arc()</a> <a href="#">ellipse()</a> <a href="#">line()</a> <a href="#">point()</a> <a href="#">quad()</a> <a href="#">rect()</a> <a href="#">triangle()</a>  <b>Curves</b> <a href="#">bezier()</a> <a href="#">bezierDetail()</a> <a href="#">bezierPoint()</a> <a href="#">bezierTangent()</a> <a href="#">curve()</a> <a href="#">curveDetail()</a> <a href="#">curvePoint()</a> <a href="#">curveTangent()</a> <a href="#">curveTightness()</a>  <b>3D Primitives</b> <a href="#">box()</a> <a href="#">sphere()</a> <a href="#">sphereDetail()</a>  <b>Attributes</b> <a href="#">ellipseMode()</a> <a href="#">noSmooth()</a> <a href="#">rectMode()</a> <a href="#">smooth()</a> <a href="#">strokeCap()</a>	<b>Setting</b> <a href="#">background()</a> <a href="#">colorMode()</a> <a href="#">fill()</a> <a href="#">noFill()</a> <a href="#">noStroke()</a> <a href="#">stroke()</a>  <b>Creating &amp; Reading</b> <a href="#">alpha()</a> <a href="#">blue()</a> <a href="#">brightness()</a> <a href="#">color()</a> <a href="#">green()</a> <a href="#">hue()</a> <a href="#">lerpColor()</a> <a href="#">red()</a> <a href="#">saturation()</a>  <b>Image</b> <a href="#">createImage()</a> <a href="#">PImage</a>  <b>Loading &amp; Displaying</b> <a href="#">image()</a> <a href="#">imageMode()</a> <a href="#">loadImage()</a> <a href="#">noTint()</a> <a href="#">requestImage()</a> <a href="#">tint()</a>  <b>Textures</b>

# Intro to Processing

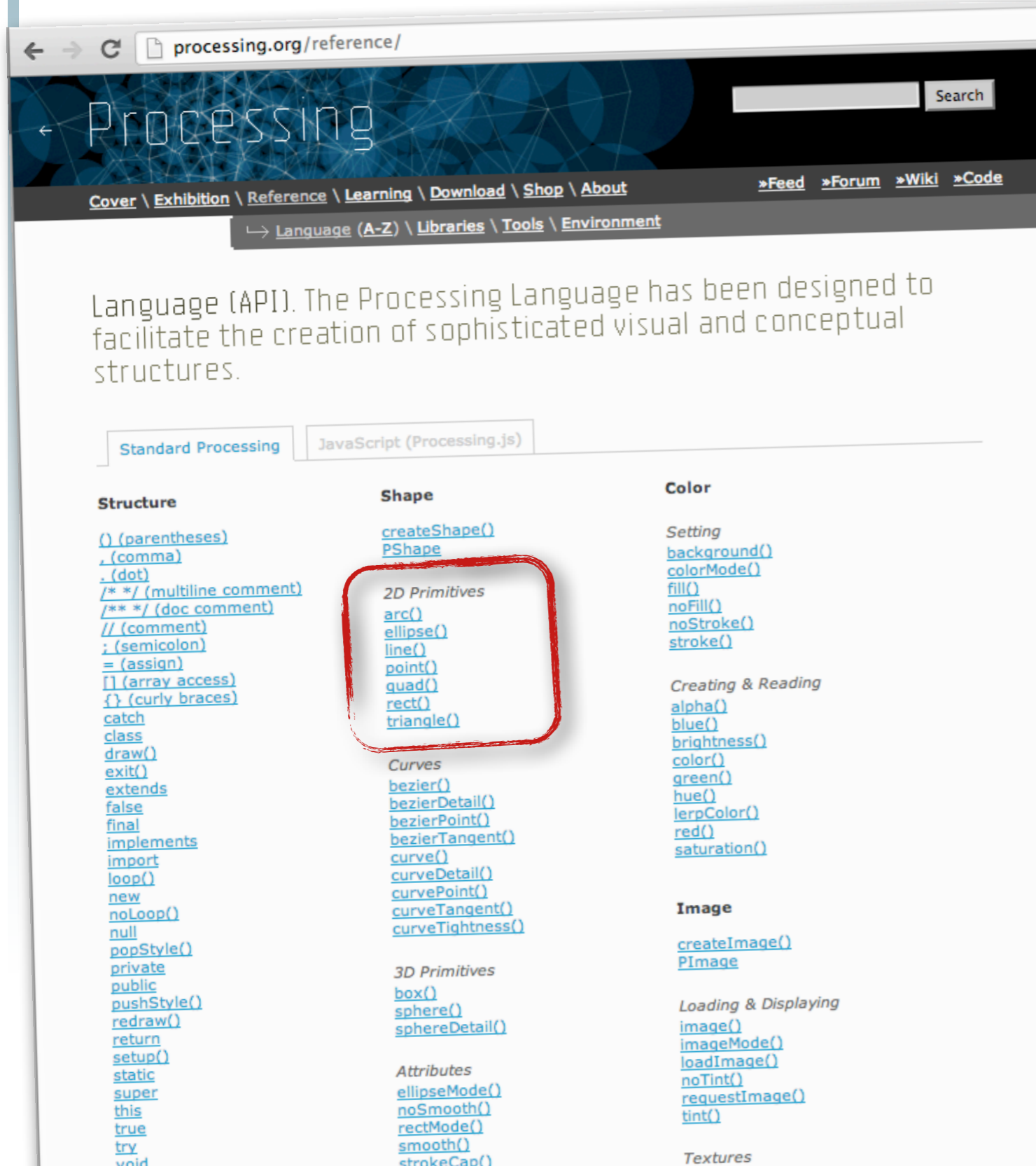
- What else can you draw?



The screenshot shows the Processing.org reference page. The browser address bar displays 'processing.org/reference/'. The page header includes the Processing logo and navigation links: 'Cover', 'Exhibition', 'Reference', 'Learning', 'Download', 'Shop', 'About', 'Feed', 'Forum', 'Wiki', and 'Code'. A breadcrumb trail shows 'Language (A-Z) > Libraries > Tools > Environment'. The main content area is titled 'Language (API). The Processing Language has been designed to facilitate the creation of sophisticated visual and conceptual structures.' Below this, there are two tabs: 'Standard Processing' (selected) and 'JavaScript (Processing.js)'. The page is organized into three columns: 'Structure', 'Shape', and 'Color'. The 'Shape' column is highlighted with a red rounded rectangle and contains the following functions: 'createShape()', 'PShape', '2D Primitives' (arc(), ellipse(), line(), point(), quad(), rect(), triangle()), 'Curves' (bezier(), bezierDetail(), bezierPoint(), bezierTangent(), curve(), curveDetail(), curvePoint(), curveTangent(), curveTightness()), '3D Primitives' (box(), sphere(), sphereDetail()), and 'Attributes' (ellipseMode(), noSmooth(), rectMode(), smooth(), strokeCap()). The 'Color' column lists 'Setting' functions (background(), colorMode(), fill(), noFill(), noStroke(), stroke()) and 'Creating & Reading' functions (alpha(), blue(), brightness(), color(), green(), hue(), lerpColor(), red(), saturation()). The 'Image' column lists 'Image' functions (createImage(), PImage) and 'Loading & Displaying' functions (image(), imageMode(), loadImage(), noTint(), requestImage(), tint()). The 'Textures' section is partially visible at the bottom.

# Intro to Processing

- What else can you draw?

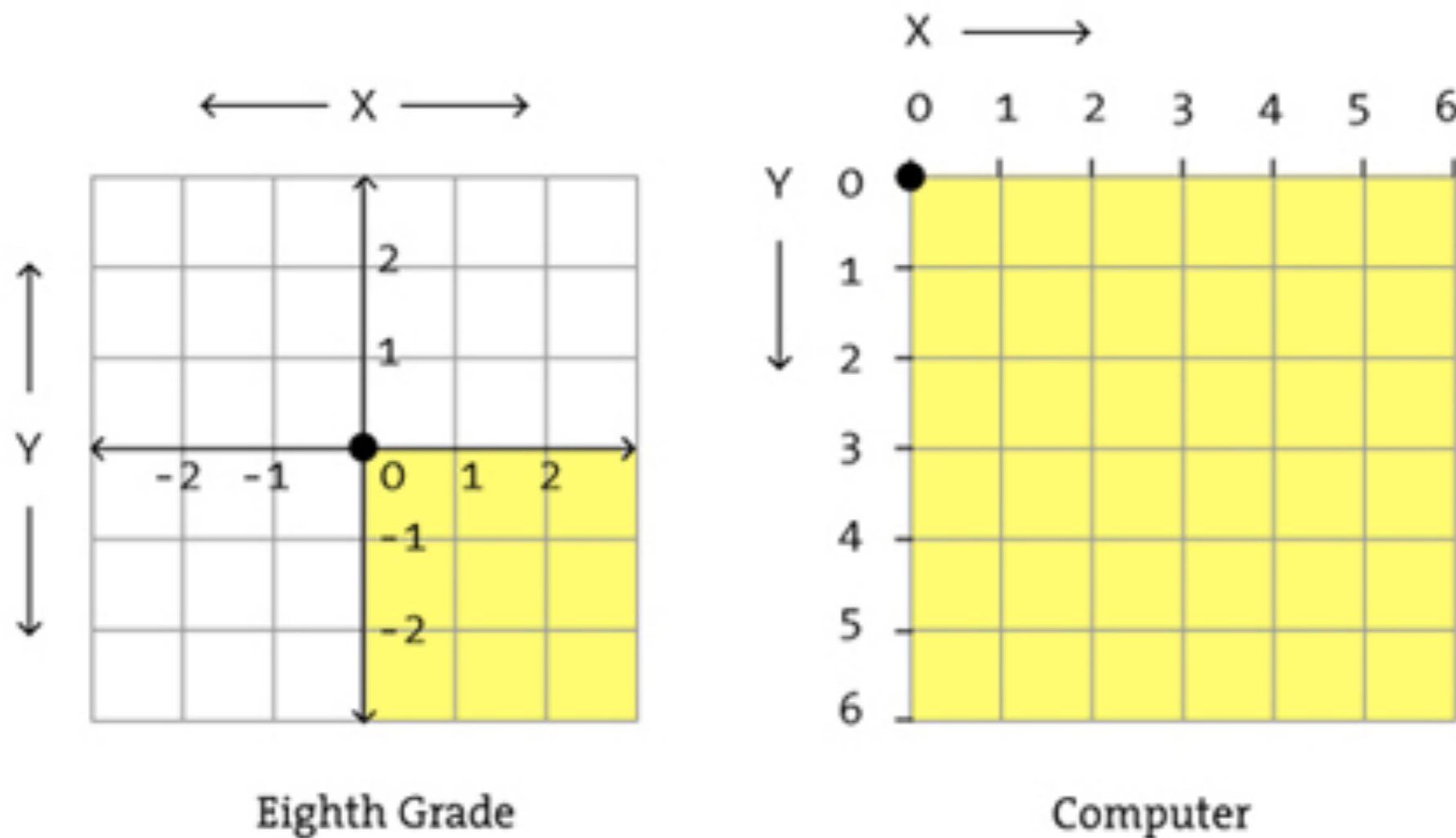


The screenshot shows the Processing.org reference page. The browser address bar displays 'processing.org/reference/'. The page has a dark blue header with the 'Processing' logo and a search bar. Below the header is a navigation menu with links for 'Cover', 'Exhibition', 'Reference', 'Learning', 'Download', 'Shop', and 'About'. A secondary menu includes 'Language (A-Z)', 'Libraries', 'Tools', and 'Environment'. The main content area features a paragraph about the Processing Language (API) and two tabs: 'Standard Processing' (selected) and 'JavaScript (Processing.js)'. The 'Standard Processing' tab is active, showing a list of functions categorized into 'Structure', 'Shape', 'Color', and 'Image'. The 'Shape' category is further divided into '2D Primitives', 'Curves', and '3D Primitives'. A red hand-drawn box highlights the '2D Primitives' list, which includes: [arc\(\)](#), [ellipse\(\)](#), [line\(\)](#), [point\(\)](#), [quad\(\)](#), [rect\(\)](#), and [triangle\(\)](#).

- Click on the link to get an example

# Intro to Processing

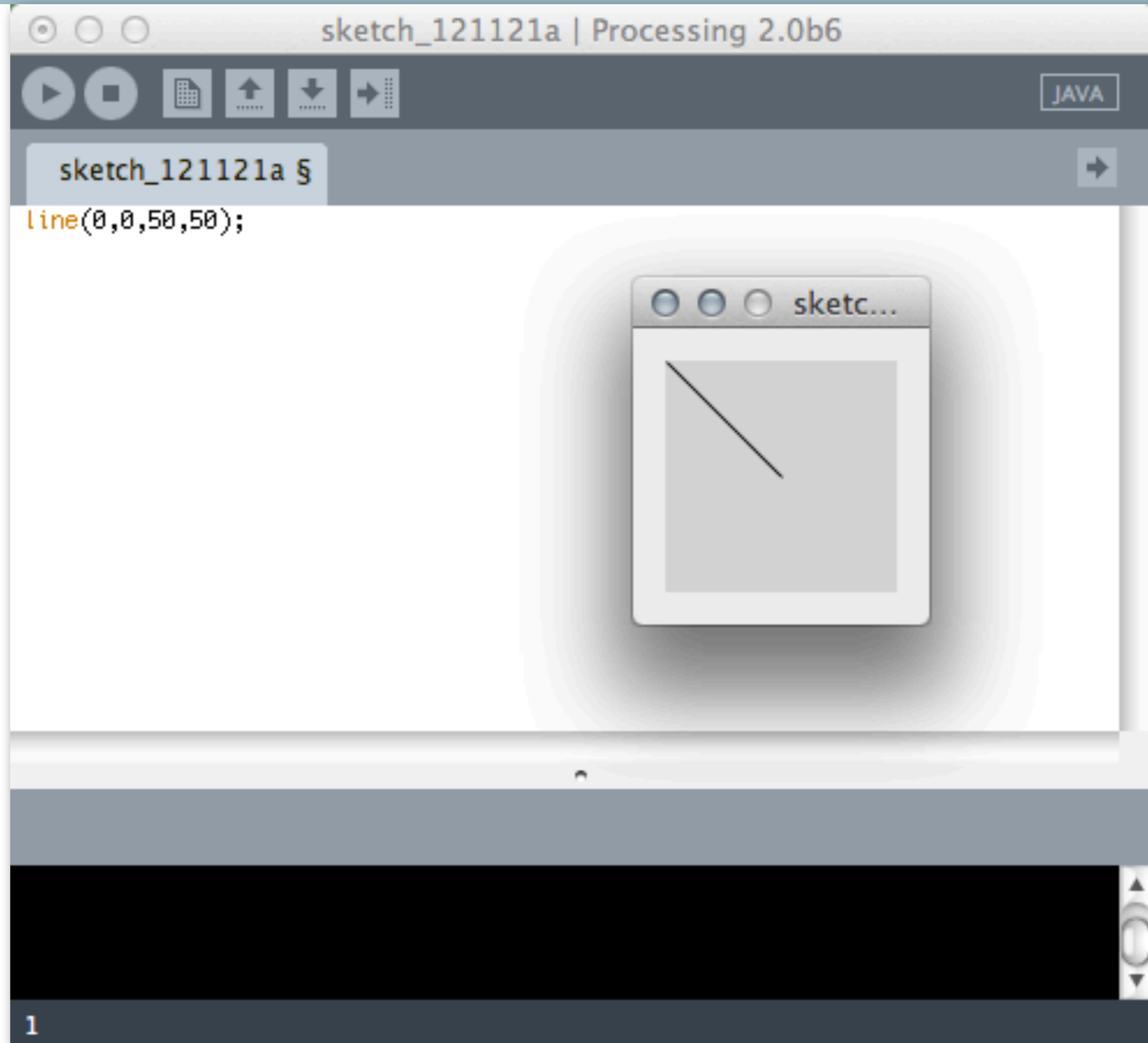
- The computer screen is like a big sheet of graph paper



- The cross-points refer to **pixels**.

<http://processing.org/>

# Intro to Processing



<http://processing.org/>

- How do simple shapes get drawn on the computer?

<http://processing.org/>

- How do simple shapes get drawn on the computer?



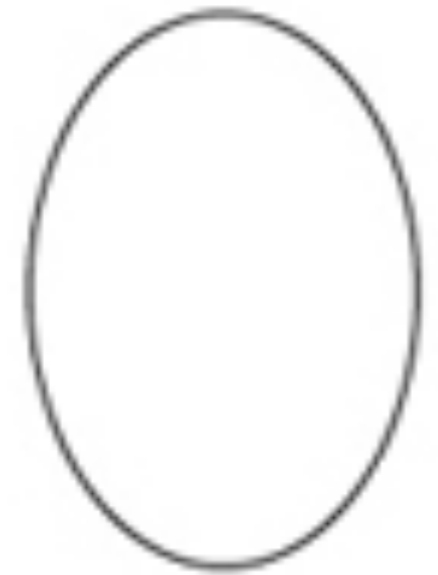
Point



Line



Rectangle



Ellipse

<http://processing.org/>

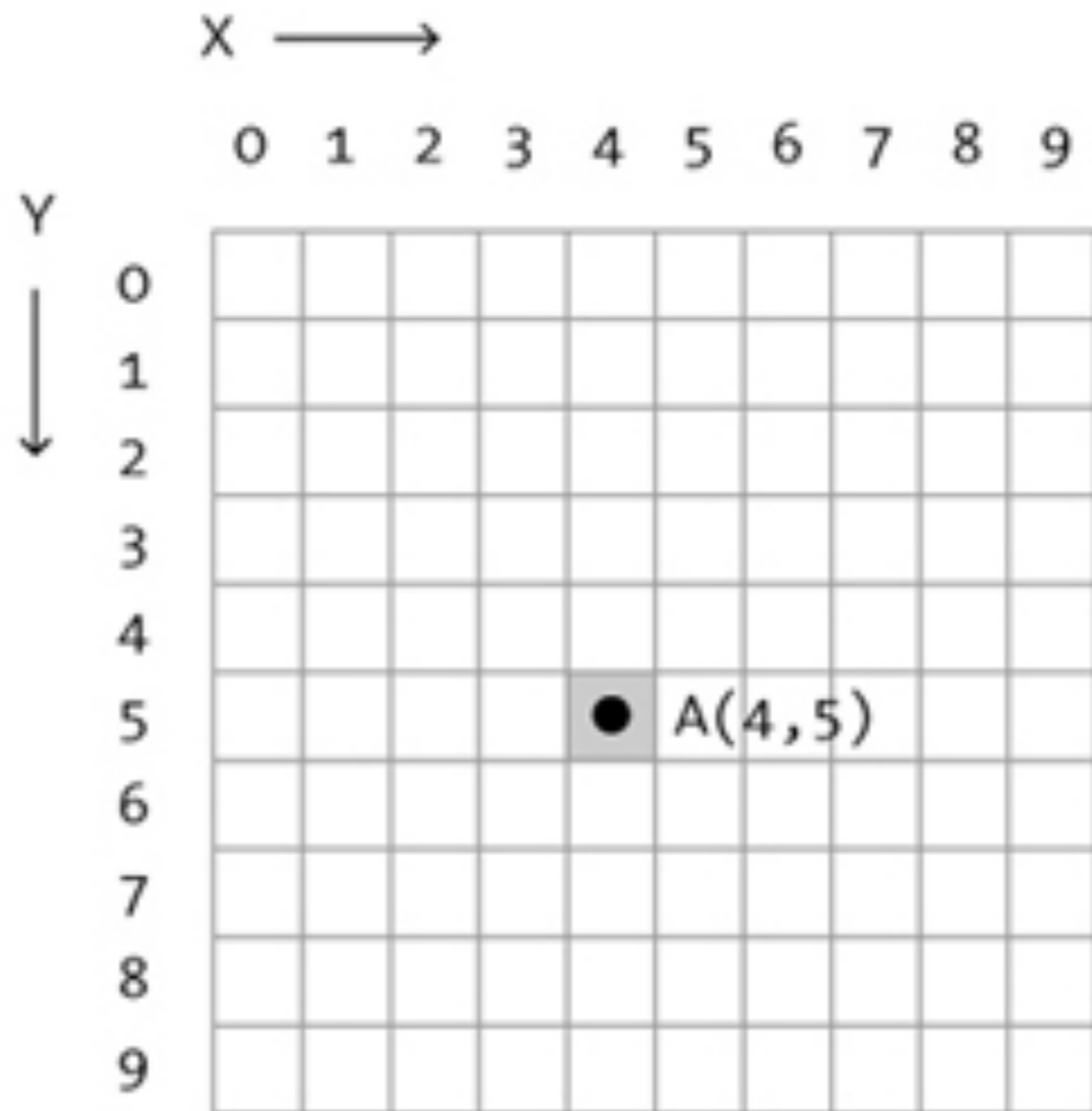
- a **point**
  - `point (4,5)`

<http://processing.org/>



# Intro to Processing

- a **point**
  - point (4,5)

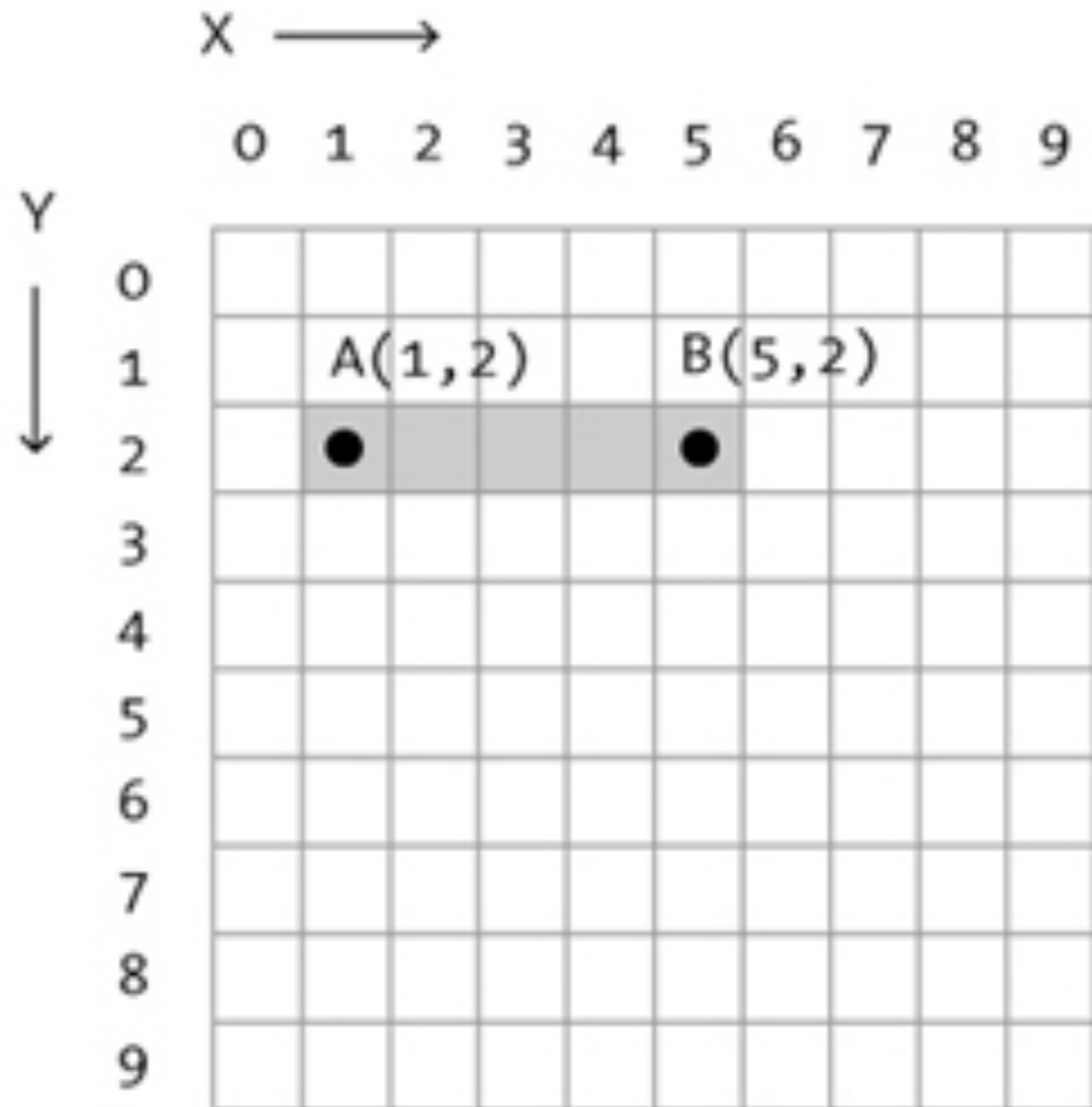


<http://processing.org/>

- a **line**
  - `line(1,2,5,2)`

<http://processing.org/>

- a **line**
  - line (1,2,5,2)



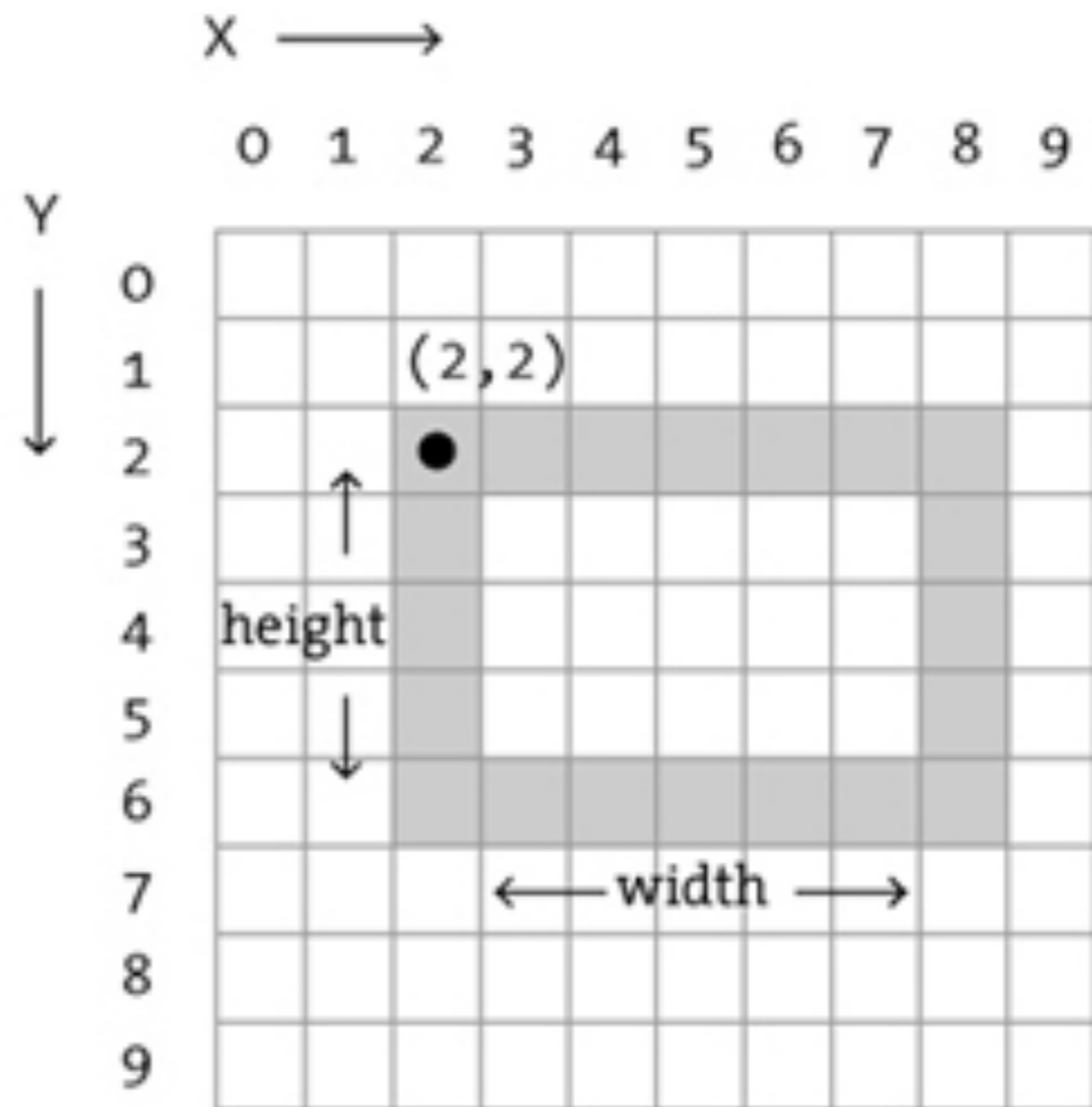
<http://processing.org/>

- a **rectangle**
  - `rect (2,2,7,5)`

<http://processing.org/>

# Intro to Processing

- a **rectangle**
  - `rect (2,2,7,5)`

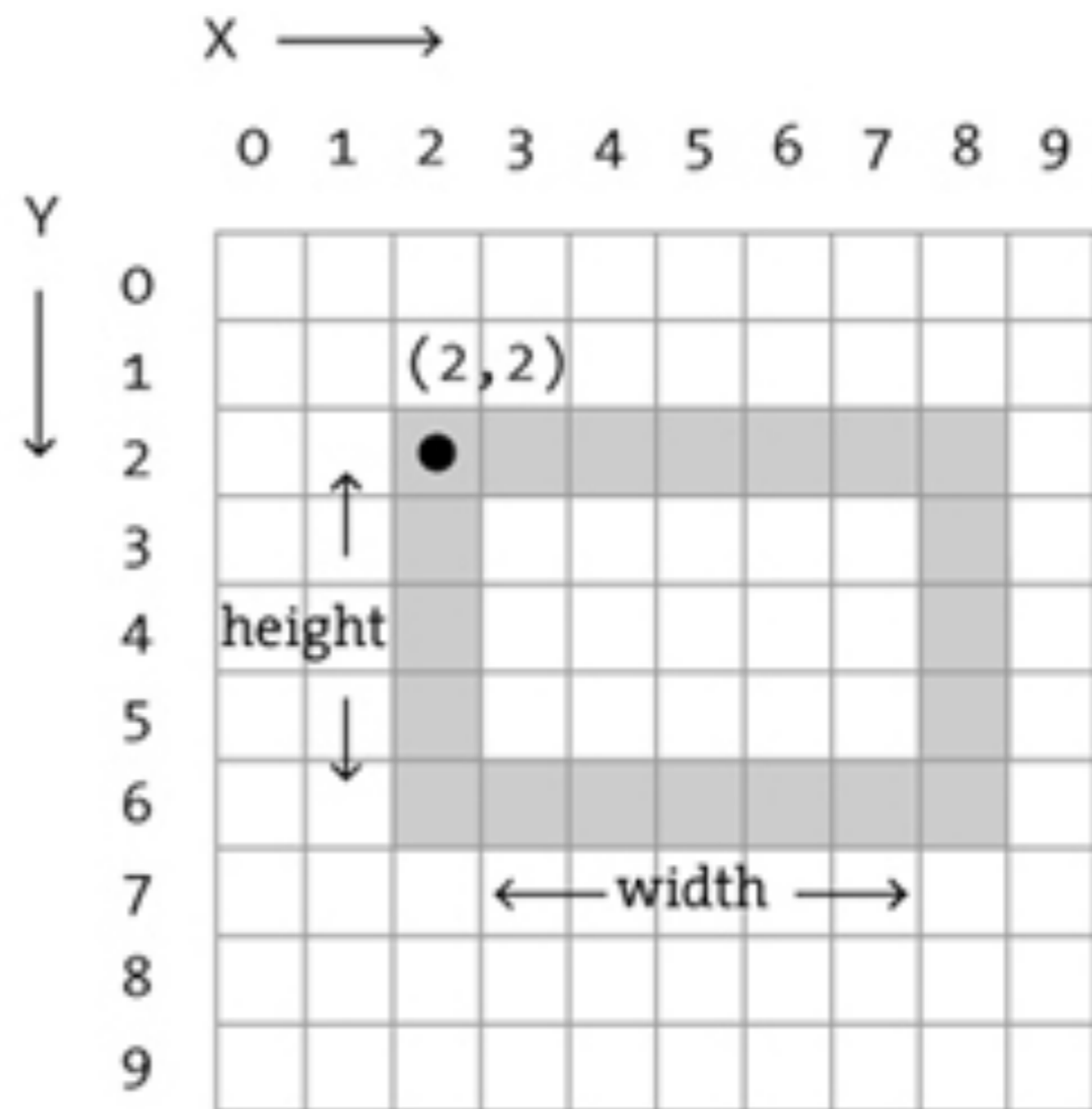


<http://processing.org/>

# Intro to Processing

- a **rectangle**
  - `rect (2,2,7,5)`

Watch Out! 7  
is width and 5  
is height!



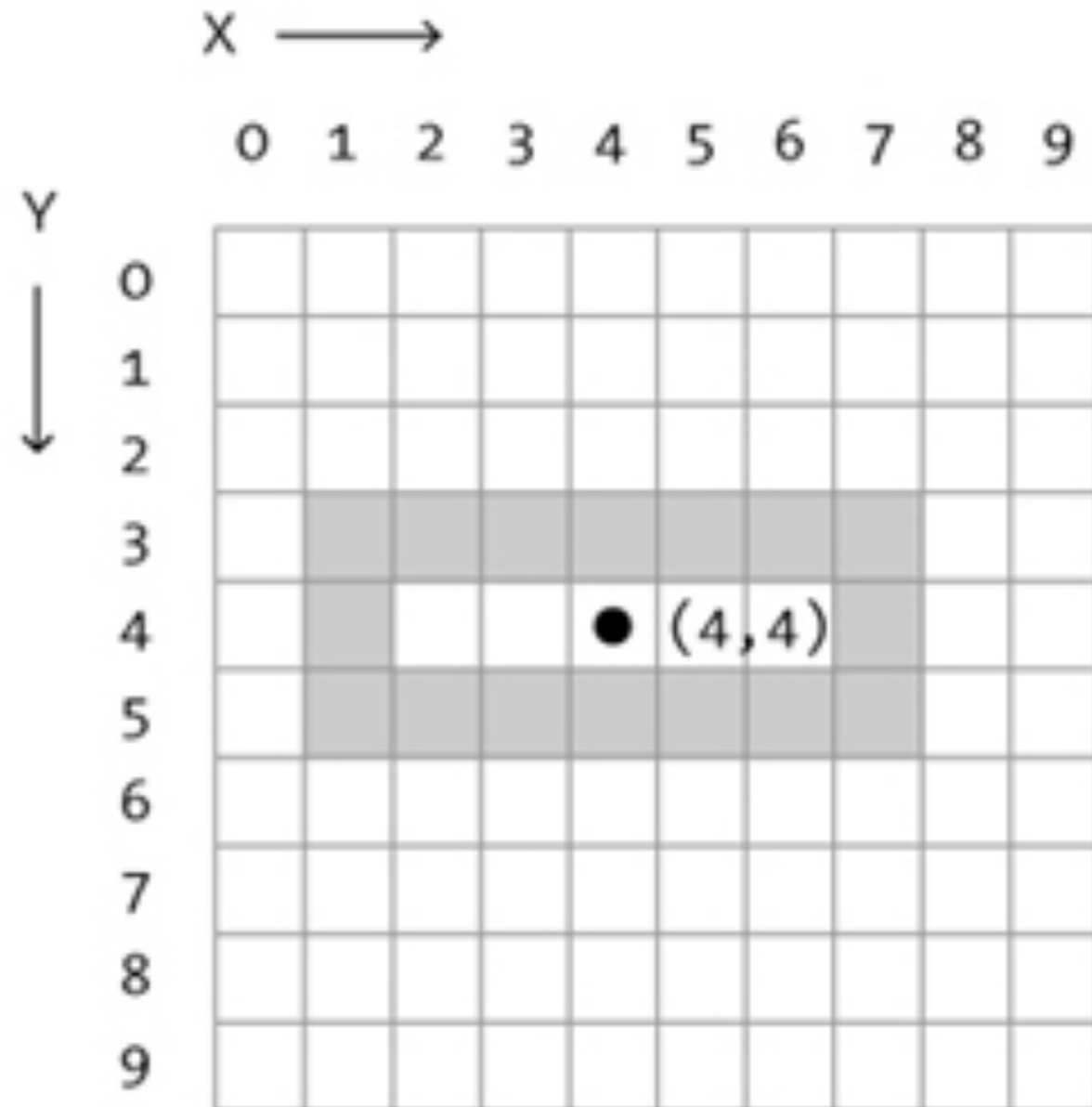
<http://processing.org/>

- a **rectangle**
  - `rectMode(CENTER)`
  - `rect(4,4,7,3)`

<http://processing.org/>

# Intro to Processing

- a **rectangle**
  - `rectMode(CENTER)`
  - `rect(4,4,7,3)`

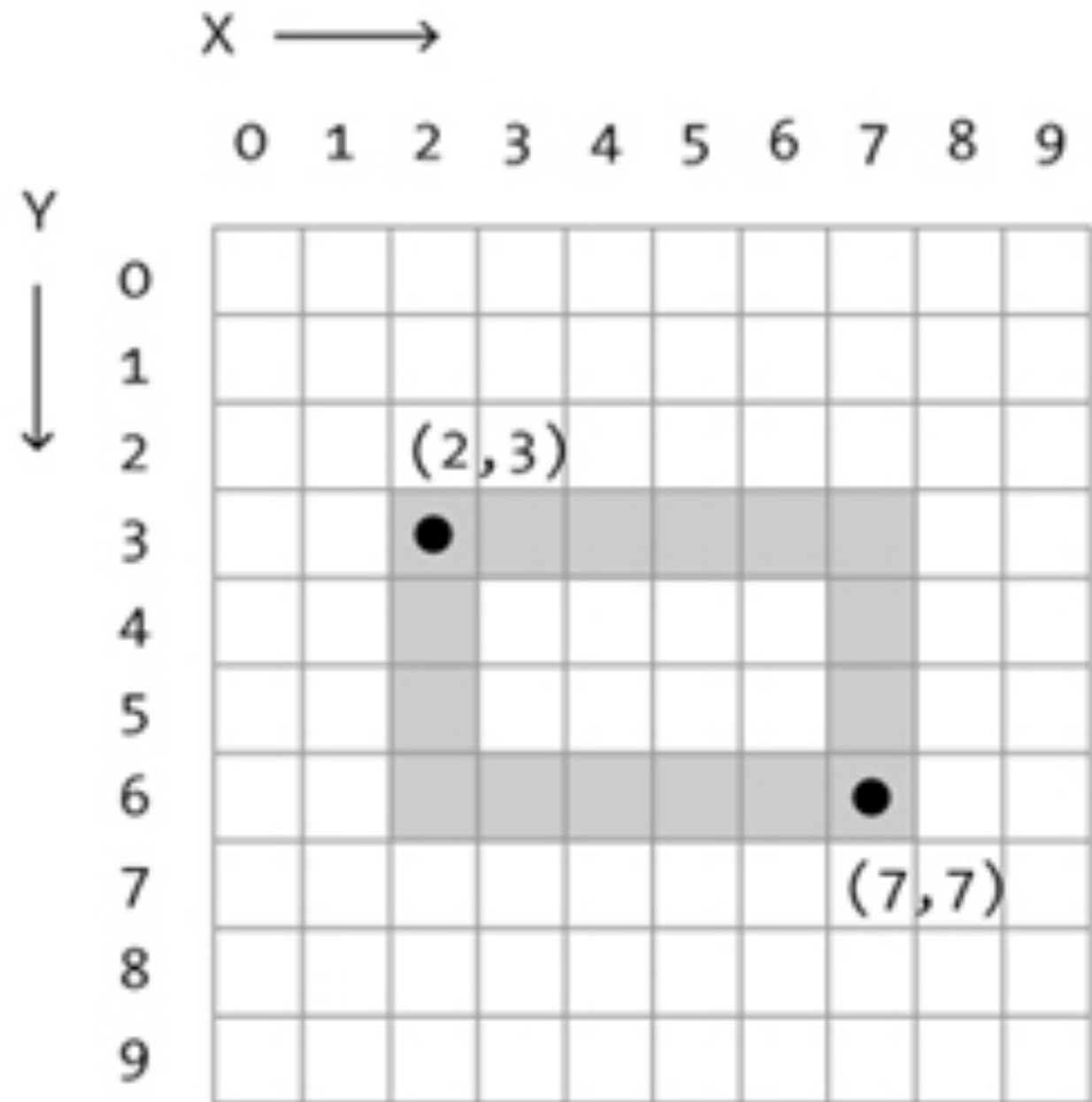


<http://processing.org/>



# Intro to Processing

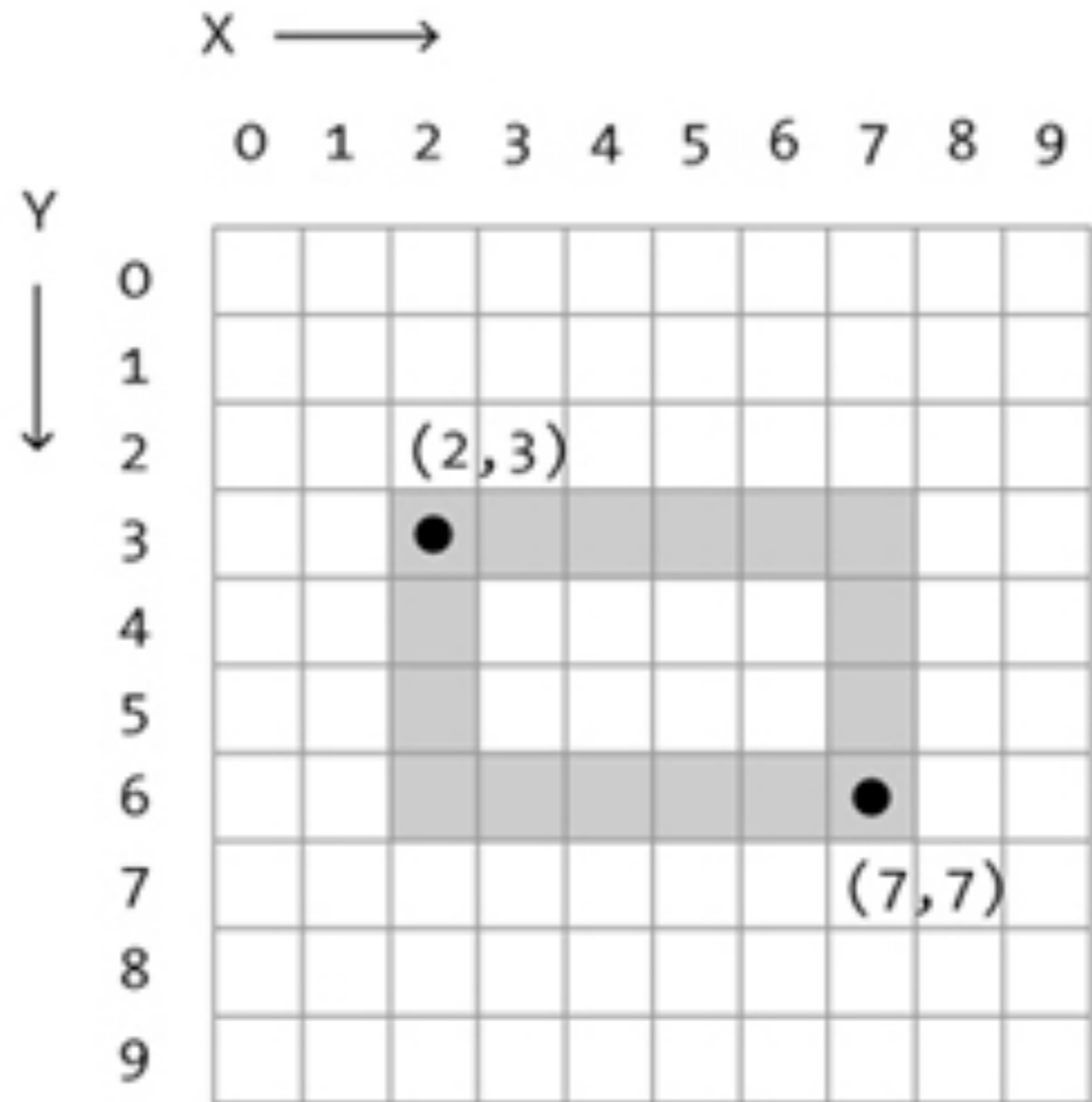
- a **rectangle**
  - `rectMode(CORNER)`
  - `rect(2,3,7,7)`



<http://processing.org/>

# Intro to Processing

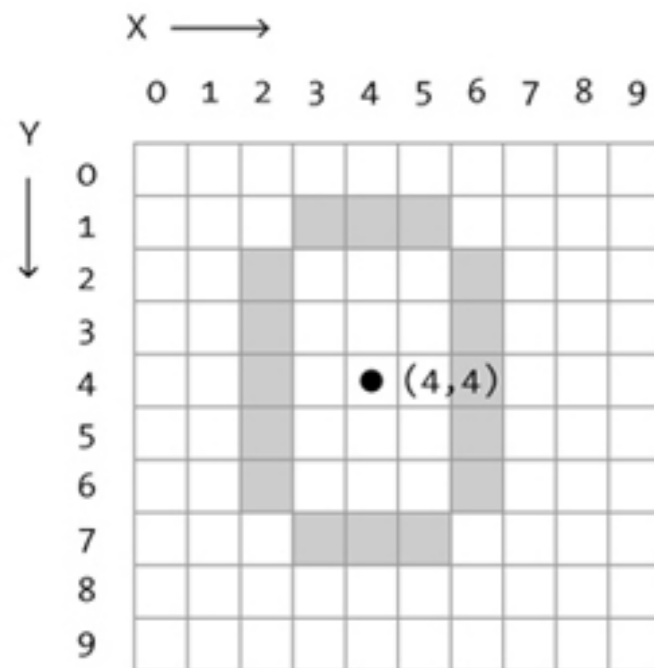
- a **rectangle**
  - `rectMode(CORNER)`
  - `rect(2,3,7,7)`



<http://processing.org/>

# Intro to Processing

- an **ellipse**
  - `ellipseMode(CENTER)`
  - `ellipse(4,4,5,7)`



<http://processing.org/>

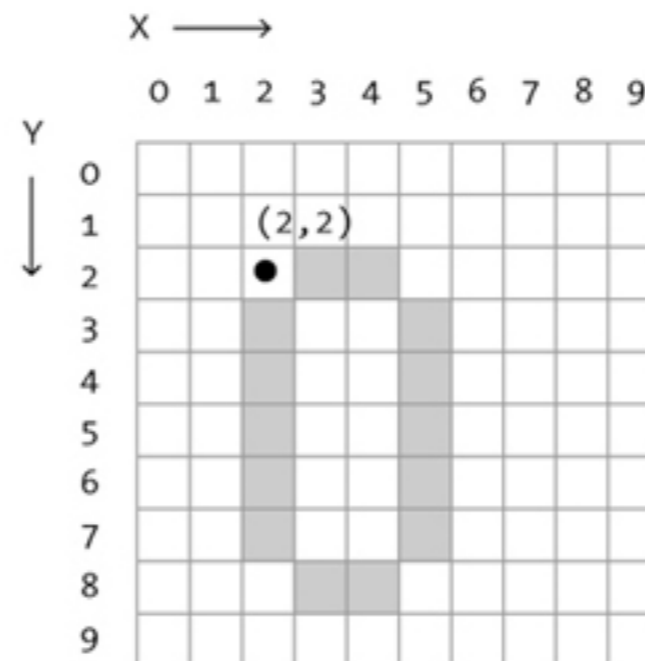
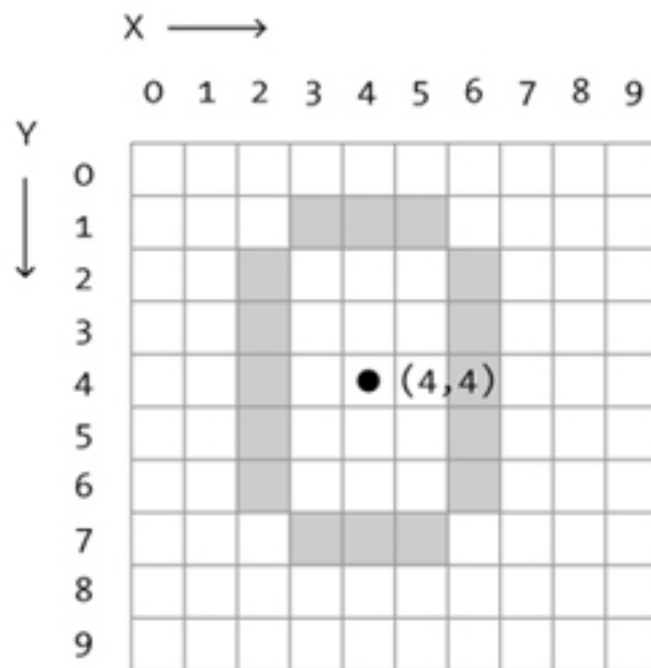
# Intro to Processing

- an ellipse

- ellipseMode(CENTER)
- ellipse(4,4,5,7)

- an ellipse

- ellipseMode(CORNER)
- ellipse(2,2,4,7)



<http://processing.org/>

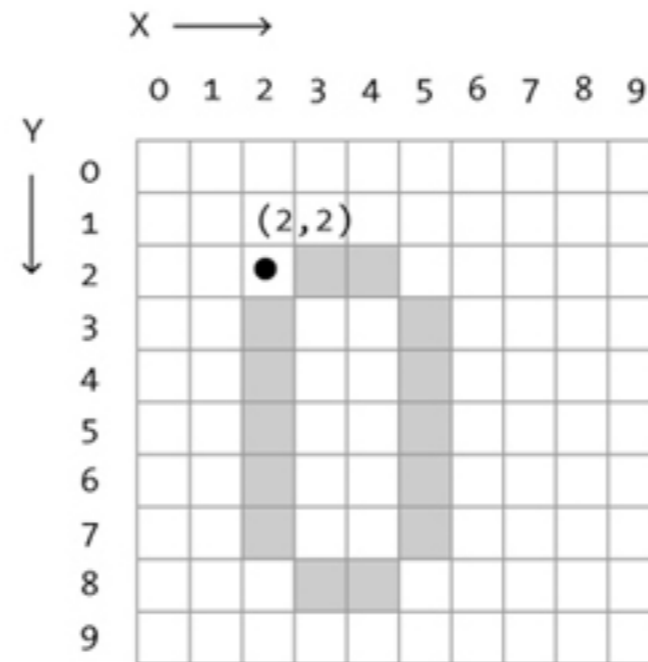
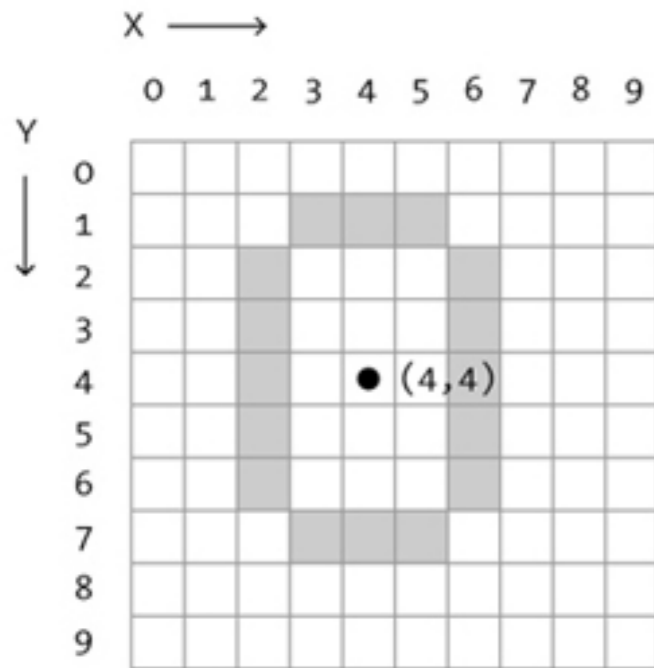
# Intro to Processing

- an ellipse

- ellipseMode(CENTER)
- ellipse(4,4,5,7)

- an ellipse

- ellipseMode(CORNER)
- ellipse(2,2,4,7)



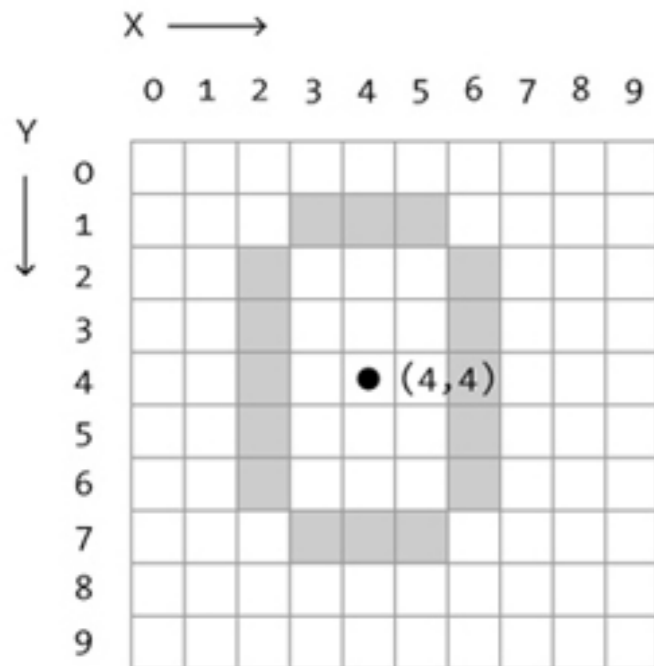
Watch Out! 4  
is width and 7  
is height!

[http](http://)

# Intro to Processing

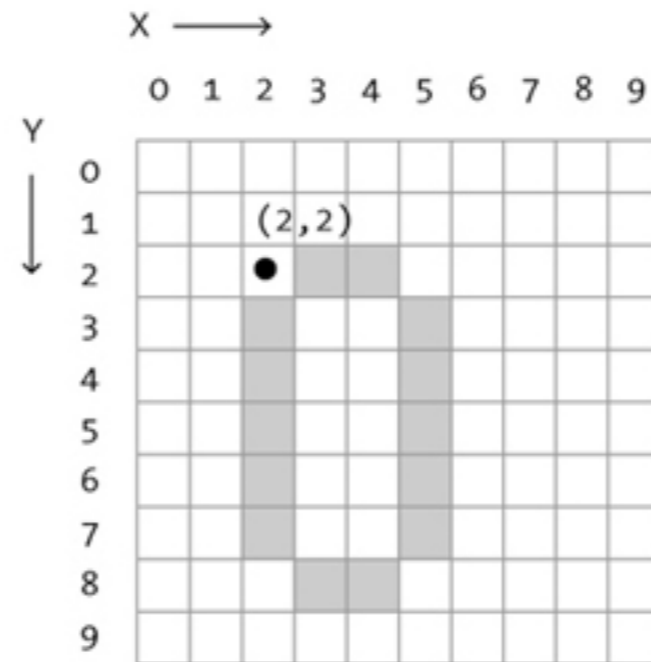
- an ellipse

- ellipseMode(CENTER)
- ellipse(4,4,5,7)



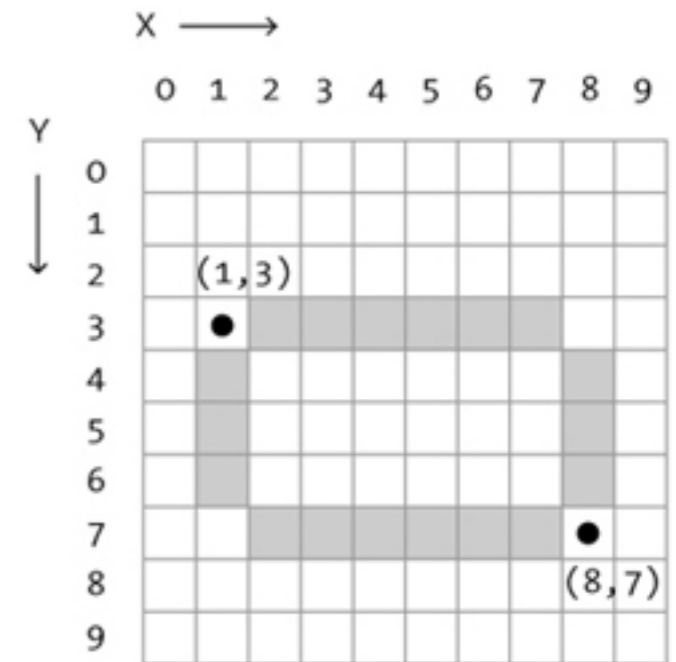
- an ellipse

- ellipseMode(CORNER)
- ellipse(2,2,4,7)



- an ellipse

- ellipseMode(CORNERS)
- ellipse(1,3,8,7)

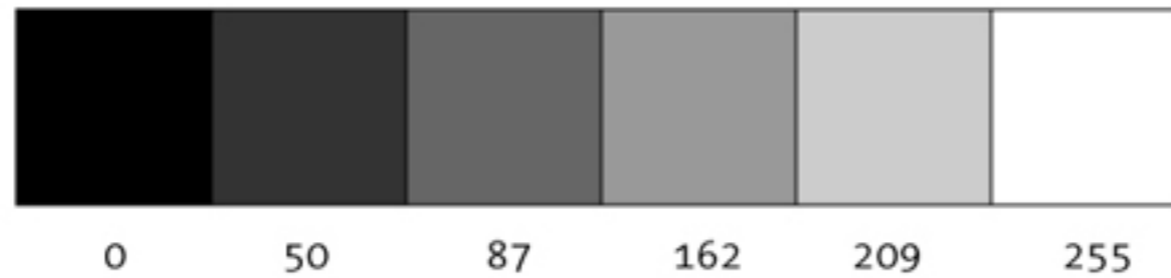


Watch Out! 4  
is width and 7  
is height!

[http](http://)

# Intro to Processing

- when describing color to a computer you must be precise
  - Grayscale color



<http://processing.org/>

- Three examples of commands that use color
  - Set the background color
    - `background(<color>);`
  - Pick the pen color that you are going to draw with
    - `stroke(<color>);`
  - Pick the fill color that you are going to draw with
    - `fill(<color>);`

<http://processing.org/>



# Intro to Processing

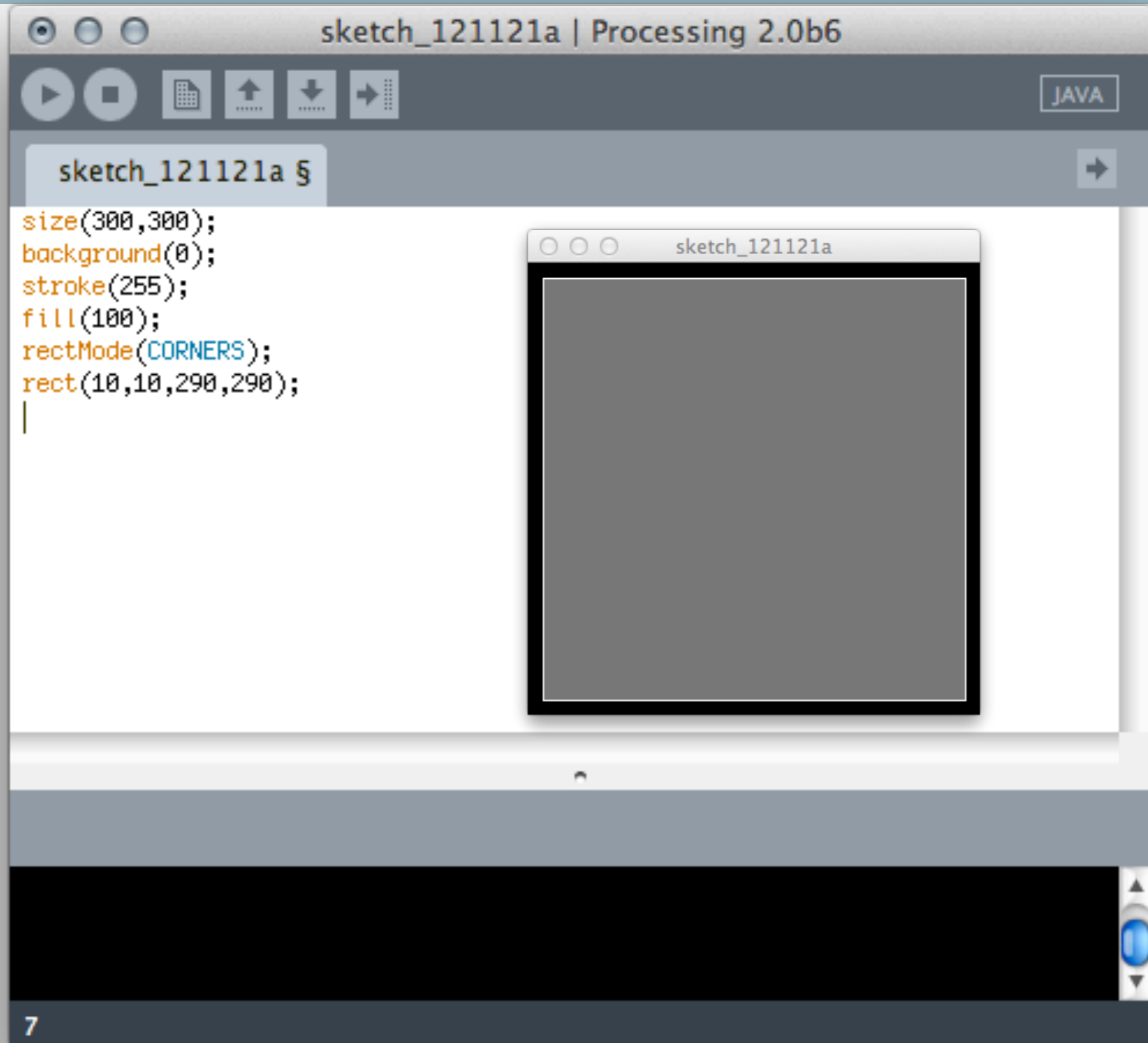


The image shows a screenshot of the Processing IDE. The window title is "sketch\_121121a | Processing 2.0b6". The interface includes a toolbar with icons for play, stop, new, save, and zoom. A "JAVA" button is visible in the top right. Below the toolbar is a tab labeled "sketch\_121121a §" with a right-pointing arrow. The main area contains the following code:

```
size(300,300);  
background(0);  
stroke(255);  
fill(100);  
rectMode(CORNERS);  
rect(10,10,290,290);  
|
```

The bottom of the window shows a dark preview area and a status bar with the number "7".

# Intro to Processing



# Intro to Processing



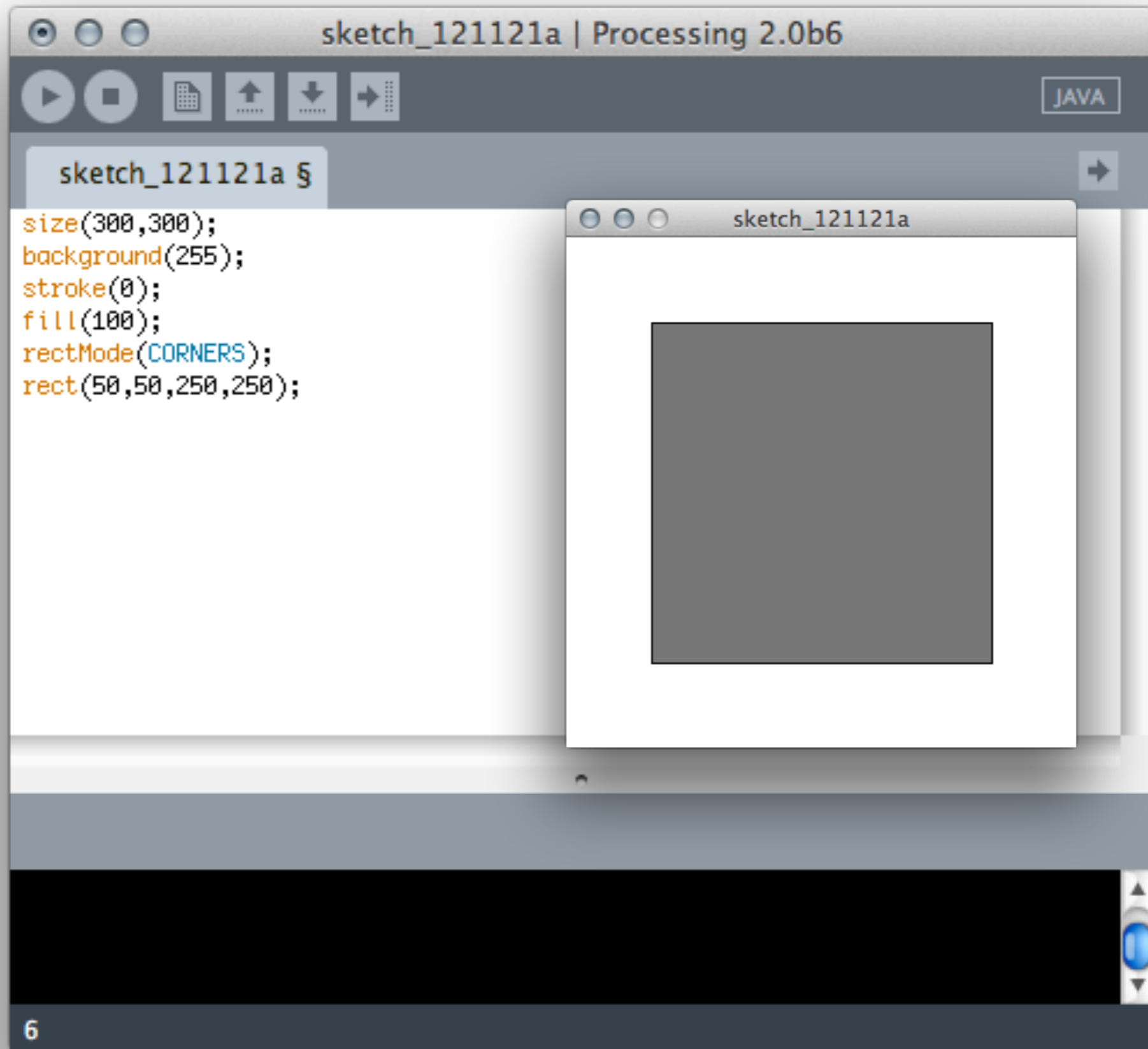
The image shows a screenshot of the Processing IDE interface. The window title is "sketch\_121121a | Processing 2.0b6". The interface includes a toolbar with icons for play, stop, save, copy, paste, and zoom. A "JAVA" button is visible in the top right. Below the toolbar is a command line with "sketch\_121121a §" and a right-pointing arrow. The main area contains the following code:

```
size(300,300);  
background(255);  
stroke(0);  
fill(100);  
rectMode(CORNERS);  
rect(50,50,250,250);
```

At the bottom of the window, there is a status bar with the number "6" on the left and a vertical scrollbar on the right.

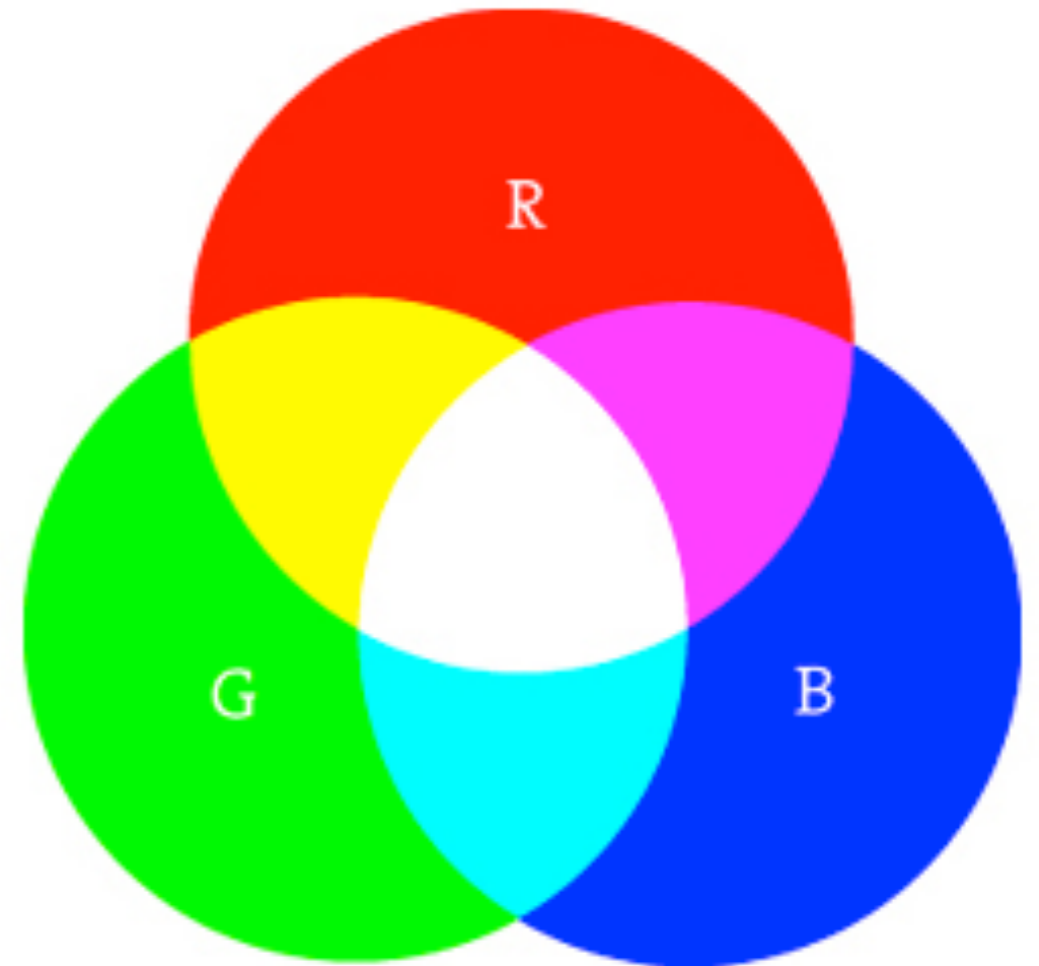
<http://processing.org/>

# Intro to Processing



<http://processing.org/>

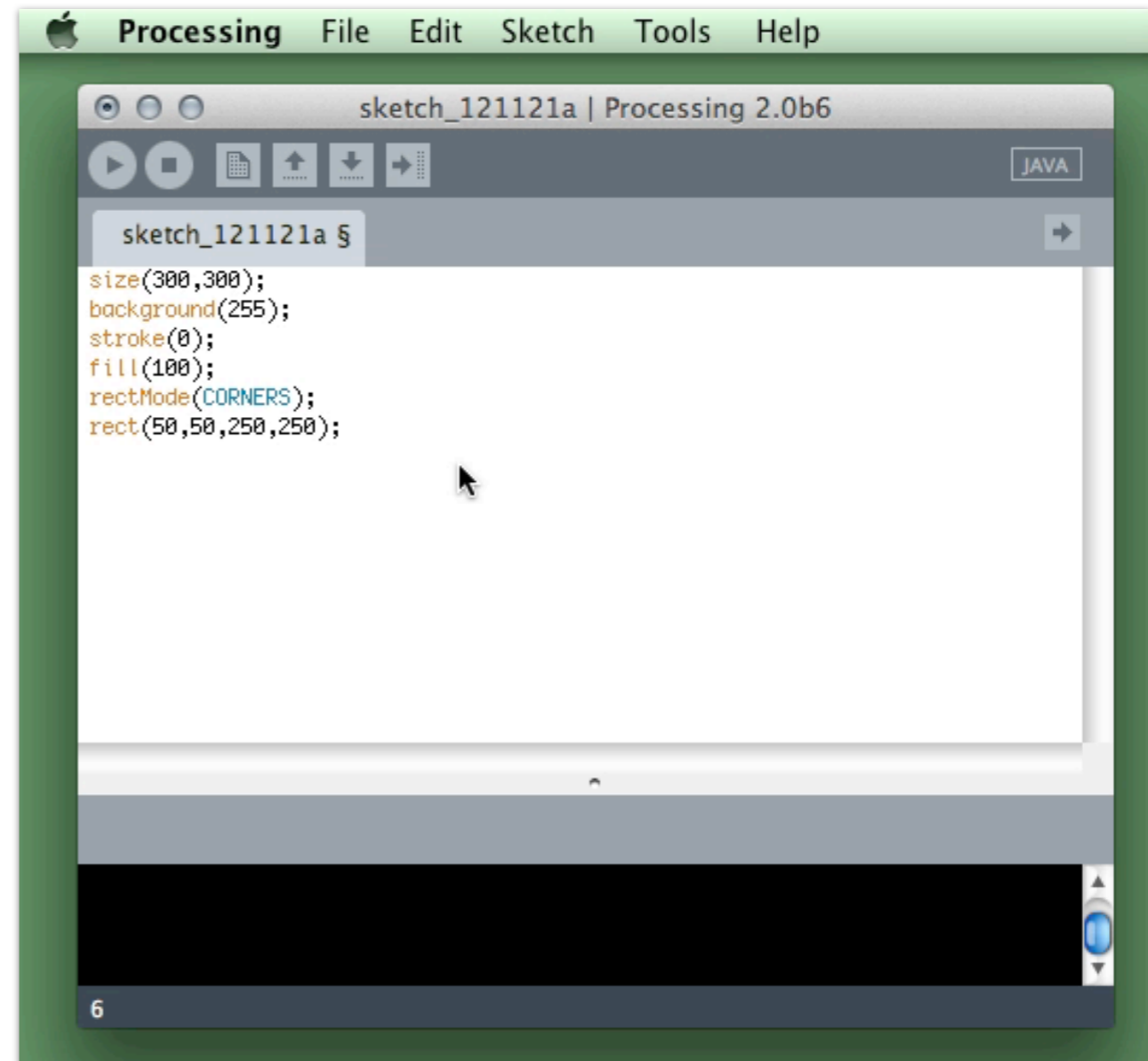
- when describing color to a computer you must be precise
  - **RGB color**
    - Red
    - Green
    - Blue
  - Red + Green = Yellow
  - Green + Blue = Cyan
  - Red + Blue = Magenta
  - Red + Green + Blue = White
  - no color = Black



<http://processing.org/>

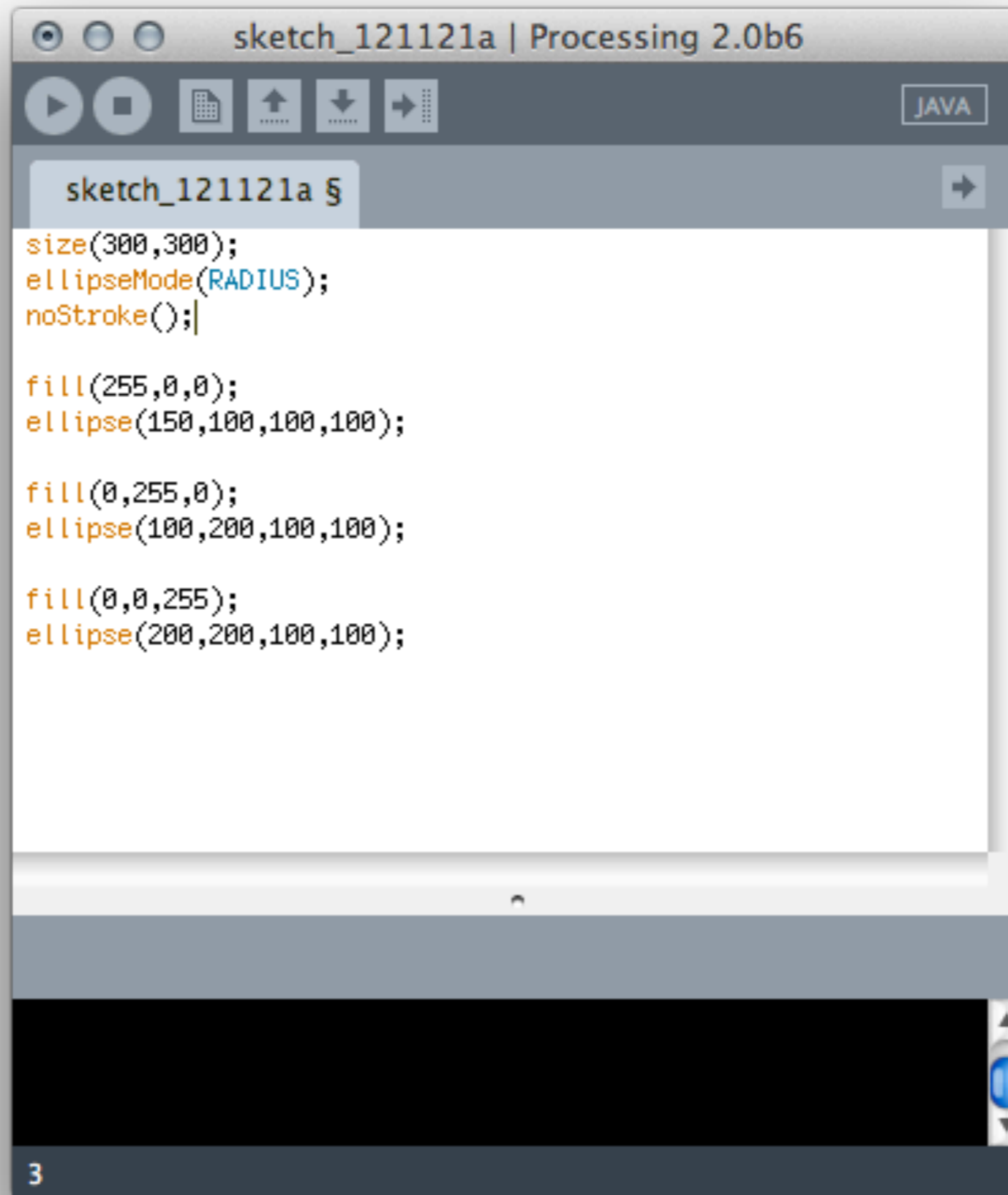
# Intro to Processing

- when describing color to a computer you must be precise
  - **RGB color**
    - Red
    - Green
    - Blue
  - Red + Green = Yellow
  - Green + Blue = Cyan
  - Red + Blue = Magenta
  - Red + Green + Blue = White
  - no color = Black



<http://processing.org/>

# Intro to Processing



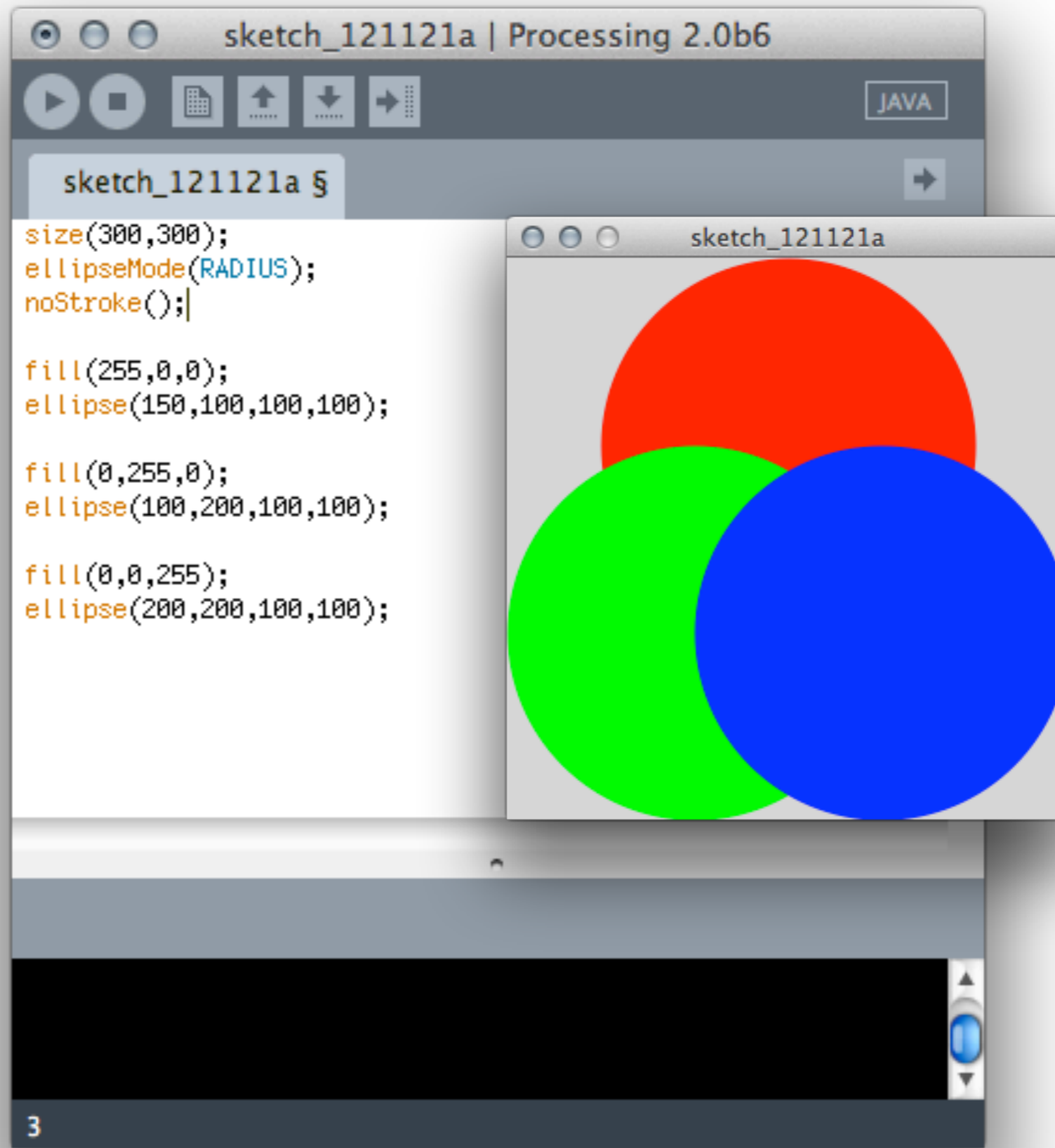
The image shows a screenshot of the Processing IDE window titled "sketch\_121121a | Processing 2.0b6". The window contains a code editor with the following Java code:

```
size(300,300);  
ellipseMode(RADIUS);  
noStroke();  
  
fill(255,0,0);  
ellipse(150,100,100,100);  
  
fill(0,255,0);  
ellipse(100,200,100,100);  
  
fill(0,0,255);  
ellipse(200,200,100,100);
```

The code defines a 300x300 canvas and uses the RADIUS ellipse mode. It draws three ellipses: a red one at (150, 100), a green one at (100, 200), and a blue one at (200, 200). The IDE interface includes a toolbar with icons for play, stop, save, and other functions, and a status bar at the bottom showing the line number 3.

<http://processing.org/>

# Intro to Processing



<http://processing.org/>




# Intro to Processing

- when describing color to a computer you must be precise
  - **alpha transparency**
    - allows for colors to blend when on top of each other
    - **0** is completely transparent
    - **255** is completely opaque

<http://processing.org/>

# Intro to Processing

- when describing color to a computer you must be precise
  - **alpha transparency**
    - allows for colors to blend when on top of each other
    - **0** is completely transparent
    - **255** is completely opaque



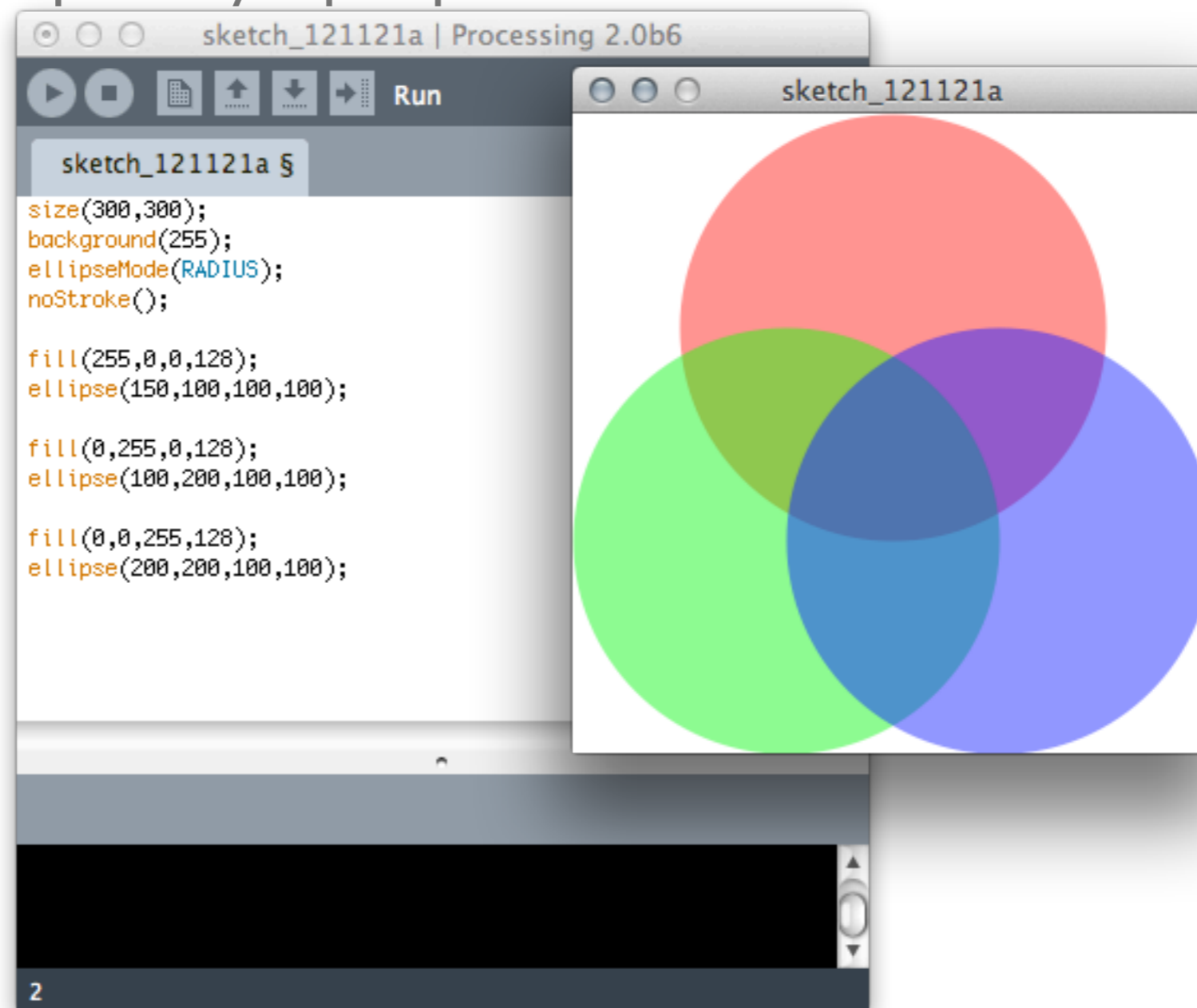
The screenshot shows a Processing IDE window titled "sketch\_121121a | Processing 2.0b6". The code editor contains the following code:

```
sketch_121121a $  
size(300,300);  
background(255);  
ellipseMode(RADIUS);  
noStroke();  
  
fill(255,0,0,128);  
ellipse(150,100,100,100);  
  
fill(0,255,0,128);  
ellipse(100,200,100,100);  
  
fill(0,0,255,128);  
ellipse(200,200,100,100);
```

The IDE interface includes a toolbar with icons for play, stop, refresh, and run, along with a "Run" button and a "JAVA" button. The sketch area at the bottom is currently black, indicating that the code has not yet been executed.

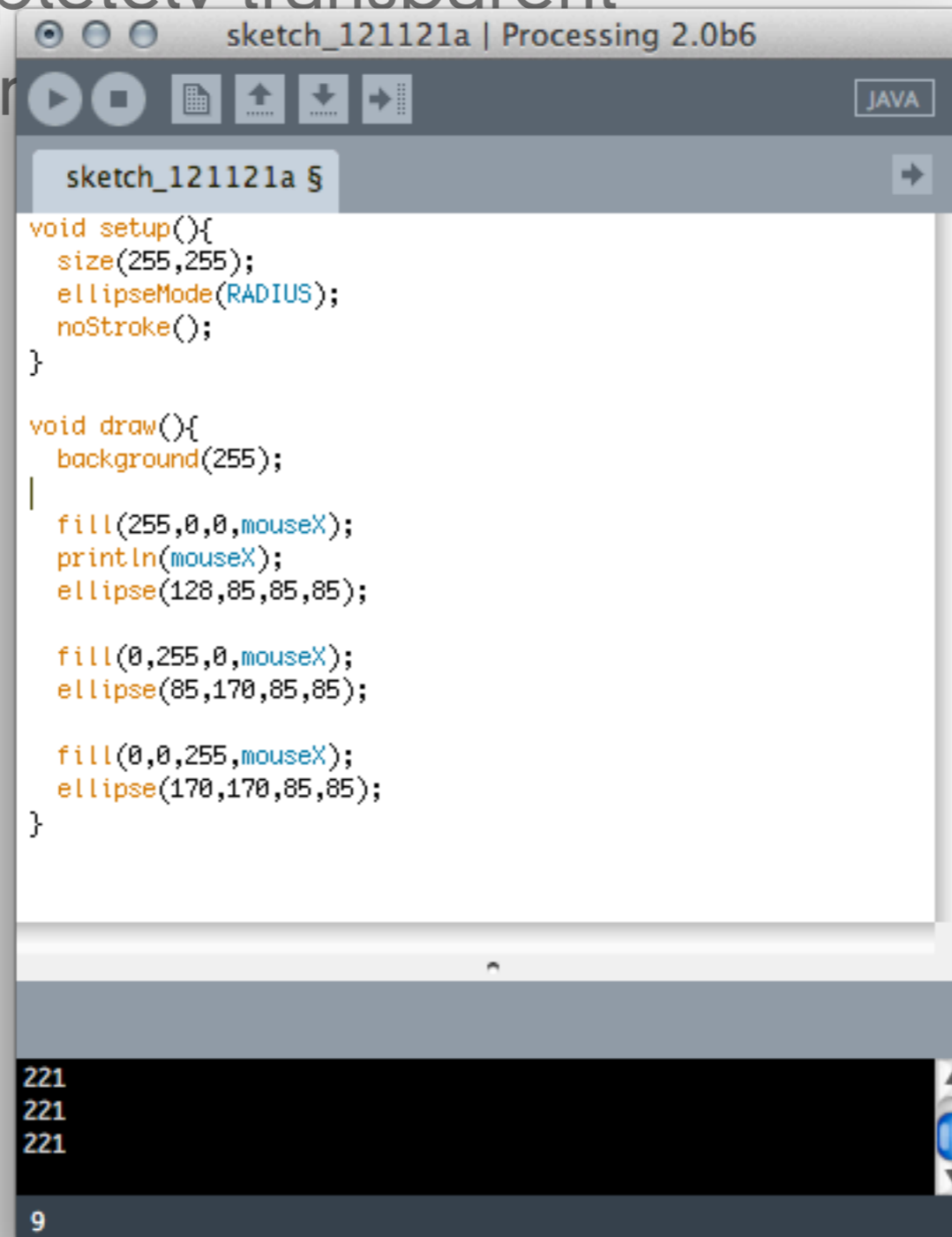
# Intro to Processing

- when describing color to a computer you must be precise
  - **alpha transparency**
    - allows for colors to blend when on top of each other
    - **0** is completely transparent
    - **255** is completely opaque



# Intro to Processing

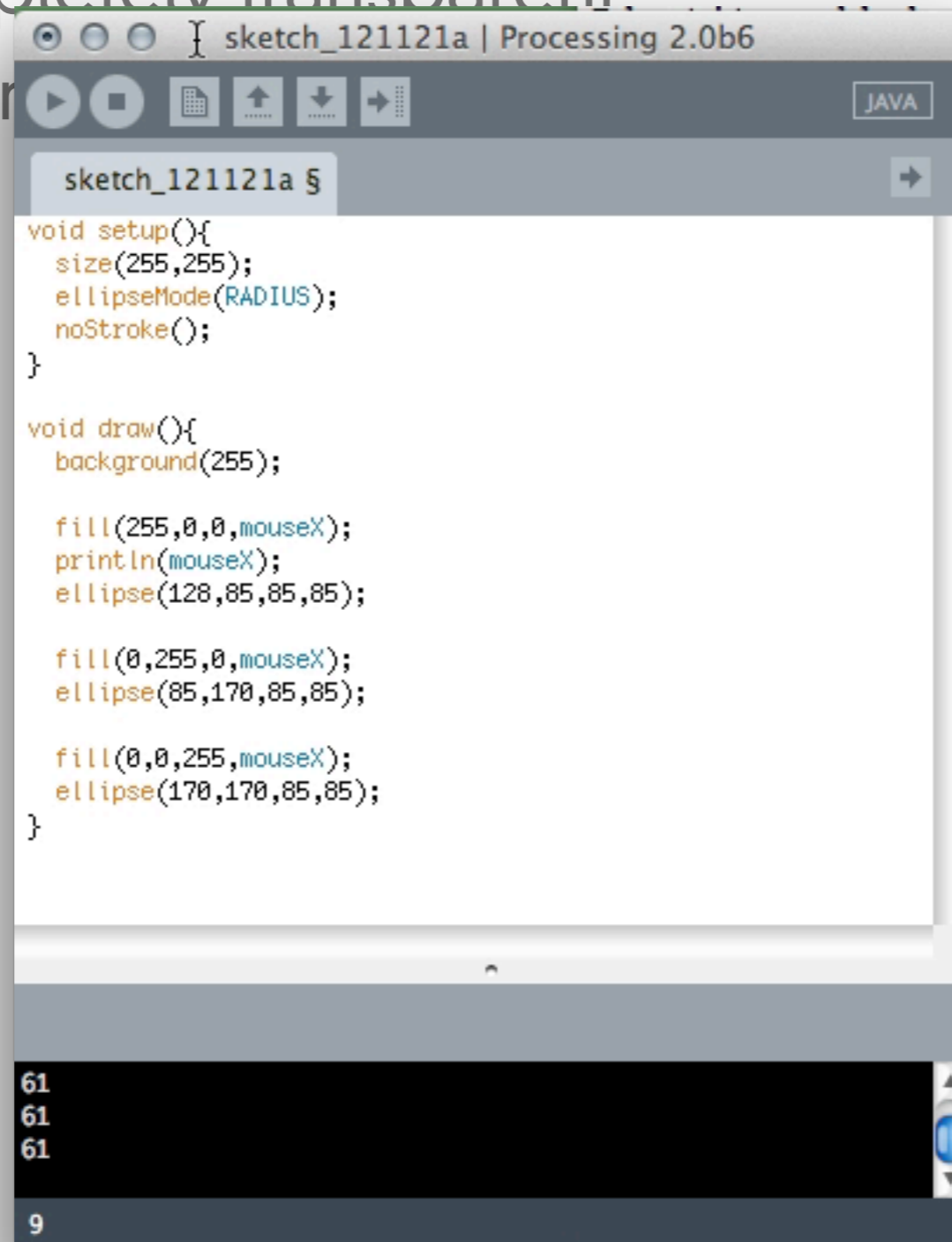
- when describing color to a computer you must be precise
  - **alpha transparency**
    - allows for colors to blend when on top of each other
    - **0** is completely transparent
    - **255** is completely opaque



```
sketch_121121a | Processing 2.0b6  
sketch_121121a §  
void setup(){  
  size(255,255);  
  ellipseMode(RADIUS);  
  noStroke();  
}  
  
void draw(){  
  background(255);  
  |  
  fill(255,0,0,mouseX);  
  println(mouseX);  
  ellipse(128,85,85,85);  
  
  fill(0,255,0,mouseX);  
  ellipse(85,170,85,85);  
  
  fill(0,0,255,mouseX);  
  ellipse(170,170,85,85);  
}  
  
221  
221  
221  
9
```

# Intro to Processing

- when describing color to a computer you must be precise
  - **alpha transparency**
    - allows for colors to blend when on top of each other
    - **0** is completely transparent
    - **255** is completely opaque

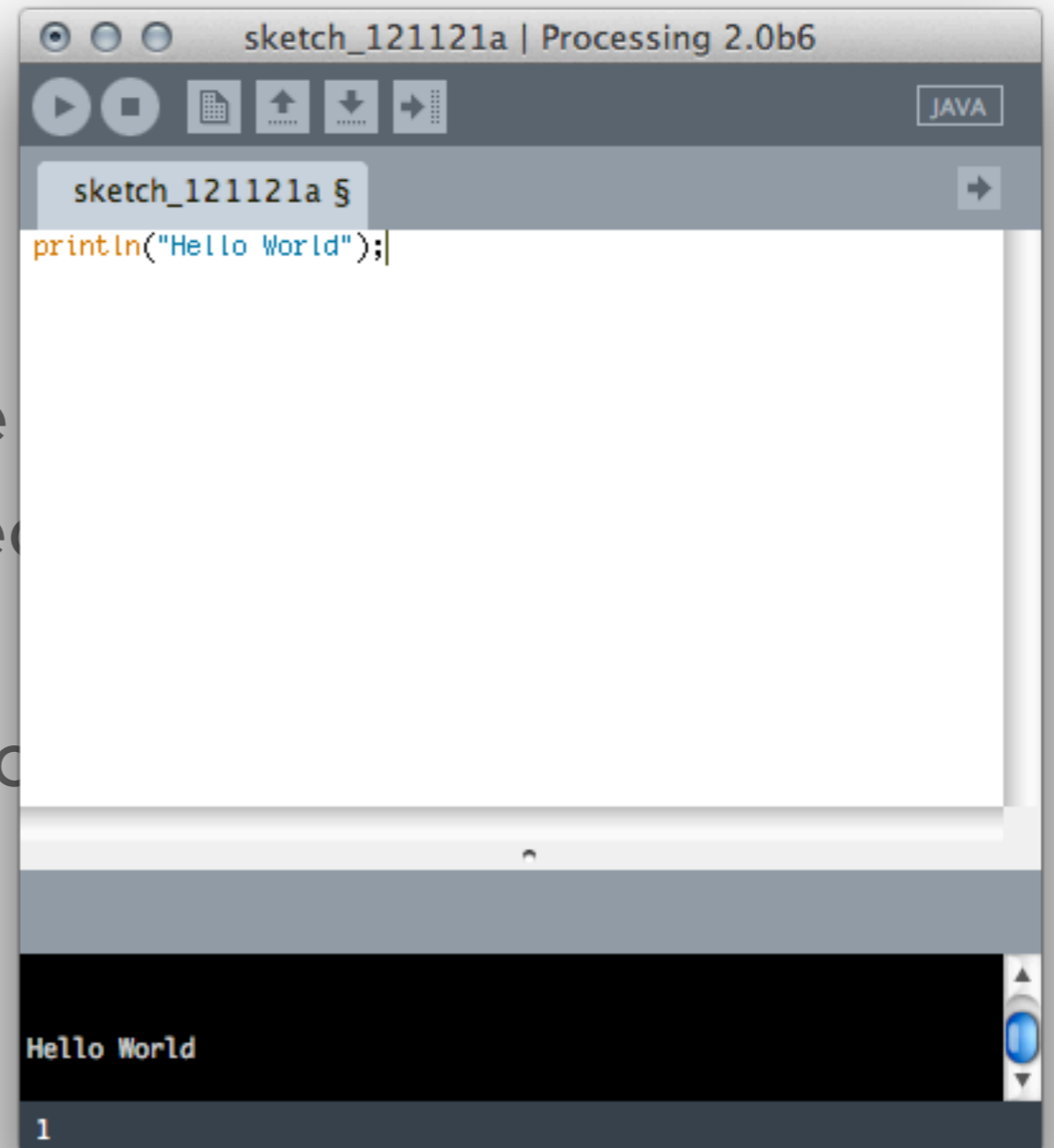


```
sketch_121121a | Processing 2.0b6  
sketch_121121a §  
void setup(){  
  size(255,255);  
  ellipseMode(RADIUS);  
  noStroke();  
}  
  
void draw(){  
  background(255);  
  
  fill(255,0,0,mouseX);  
  println(mouseX);  
  ellipse(128,85,85,85);  
  
  fill(0,255,0,mouseX);  
  ellipse(85,170,85,85);  
  
  fill(0,0,255,mouseX);  
  ellipse(170,170,85,85);  
}  
  
61  
61  
61  
9
```

- Text
  - **String**
    - A collection of letters that are put between quotes
    - They get treated as one object
      - "Hello World" is a String
  - Displaying text in the message console
    - `println("Hello World")`

<http://processing.org/>

- Text
  - **String**
    - A collection of letters that are
    - They get treated as one object
      - "Hello World" is a String
  - Displaying text in the message console
    - `println("Hello World")`



- Displaying text in a sketch
  - Find out what fonts are available to you
    - `PFont.list()`

<http://processing.org/>



# Intro to Processing

- Displaying text in a sketch
  - Find out what fonts are available to you
    - `PFont.list()`

A screenshot of a Processing IDE window titled "sketch\_121121a | Processing 2.0b6". The window has a toolbar with icons for play, stop, refresh, and zoom. Below the toolbar is a code editor with the text `println(PFont.list());`. The output window below the code editor displays a list of 14 font names in a white monospace font on a black background. The list is indexed from [0] to [13].

```
[0] "Serif"
[1] "SansSerif"
[2] "Monospaced"
[3] "Dialog"
[4] "DialogInput"
[5] "ACaslonPro-Bold"
[6] "ACaslonPro-BoldItalic"
[7] "ACaslonPro-Italic"
[8] "ACaslonPro-Regular"
[9] "ACaslonPro-Semibold"
[10] "ACaslonPro-SemiboldItalic"
[11] "AGaramondPro-Bold"
[12] "AGaramondPro-BoldItalic"
[13] "AGaramondPro-Italic"
1
```

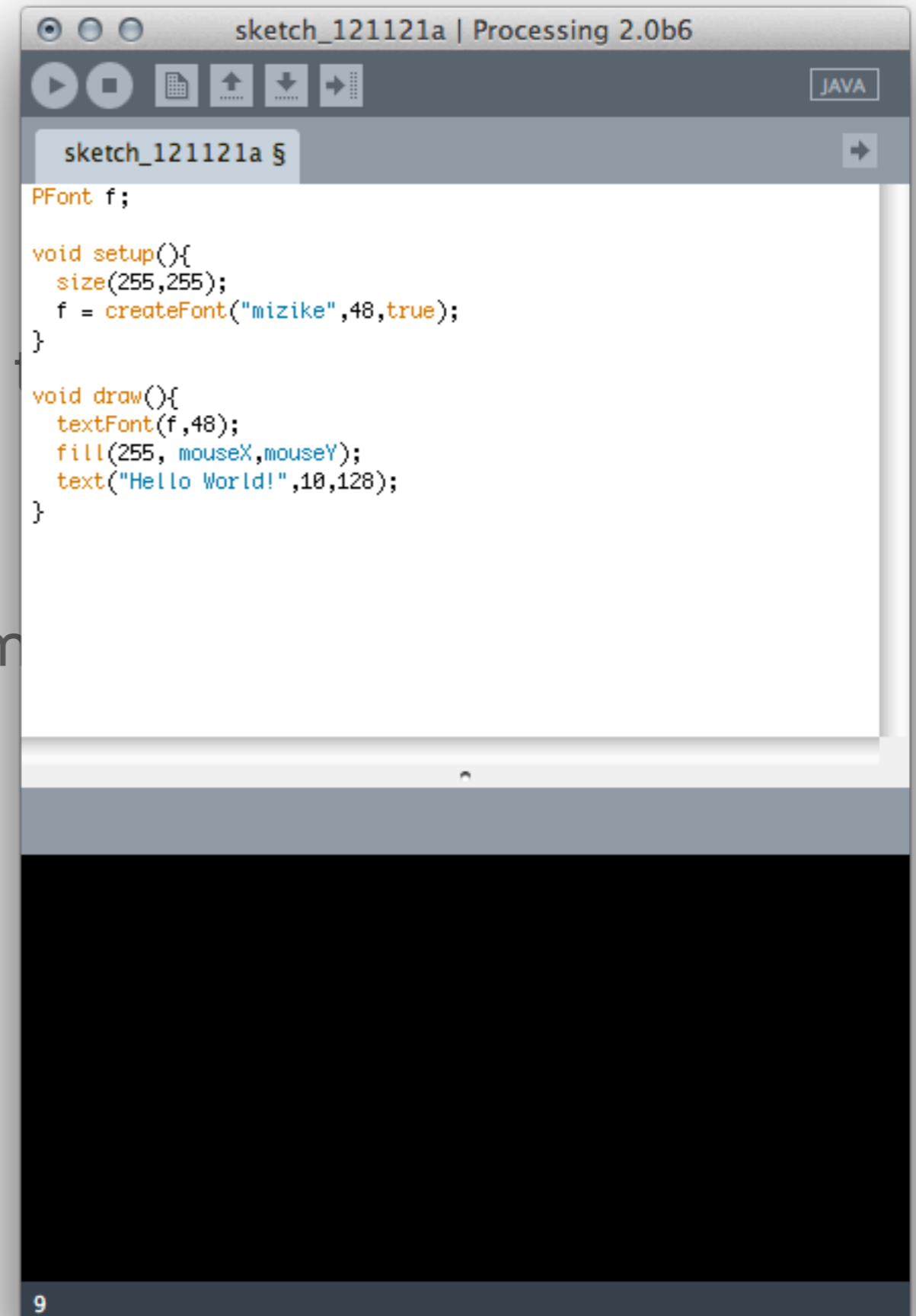
<http://processing.org/>

- Create a font to use in `setup()`
  - `createFont()`
- Specify the way to write the font to in `draw()`
  - `textFont()`
  - `fill()`
- Specify the string and the placement in `draw()`
  - `text("Hello World!", 50, 50)`

<http://processing.org/>

# Intro to Processing

- Create a font to use in `setup()`
  - `createFont()`
- Specify the way to write the font
  - `textFont()`
  - `fill()`
- Specify the string and the placement
  - `text("Hello World!", 50, 50)`



```
sketch_121121a | Processing 2.0b6
sketch_121121a §
PFont f;

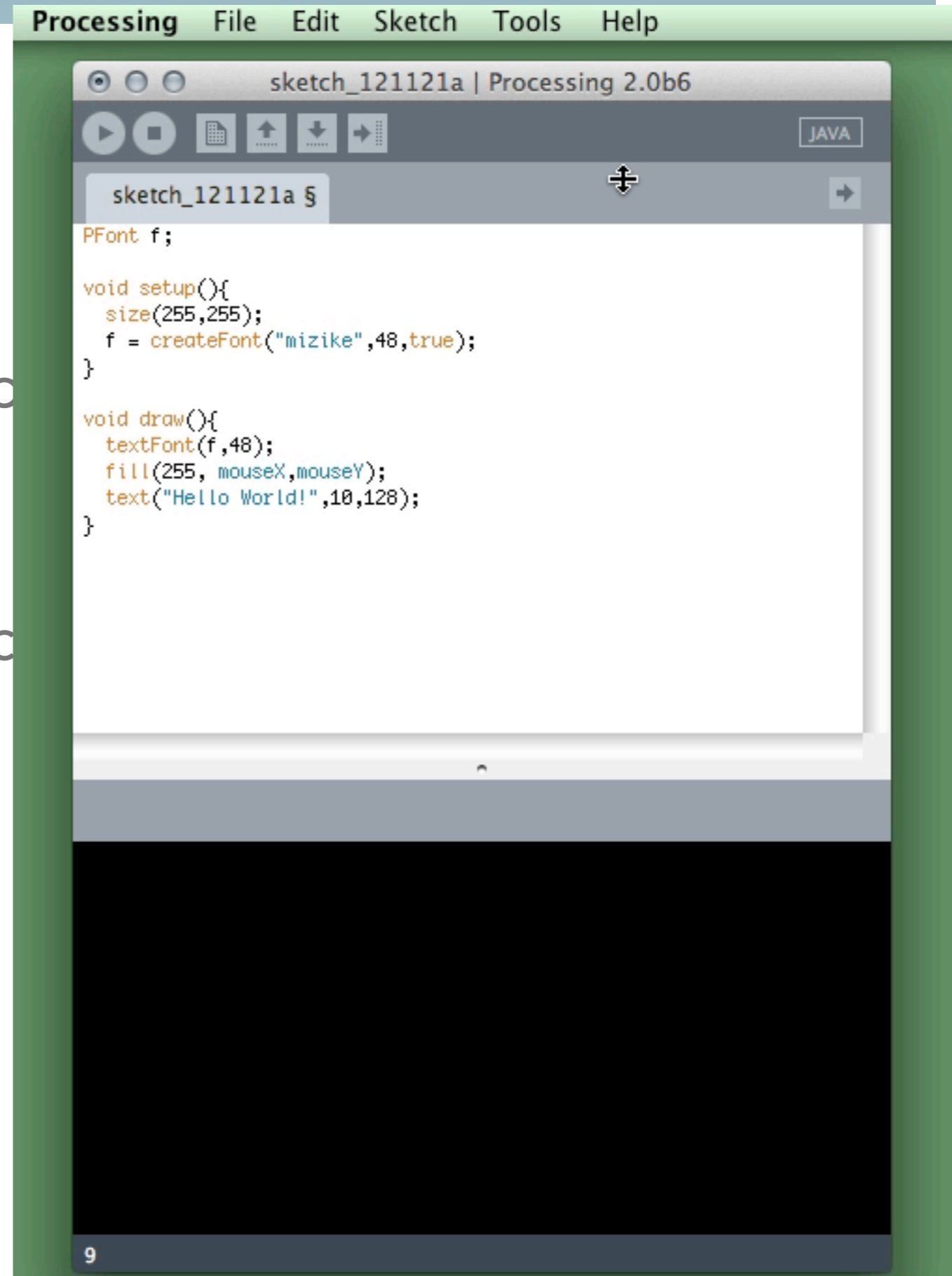
void setup(){
  size(255,255);
  f = createFont("mizike",48,true);
}

void draw(){
  textFont(f,48);
  fill(255, mouseX,mouseY);
  text("Hello World!",10,128);
}
```

<http://processing.org/>

# Intro to Processing

- Create a font to use in `setup()`
  - `createFont()`
- Specify the way to write the font
  - `textFont()`
  - `fill()`
- Specify the string and the place
  - `text("Hello World!", 50, 50)`



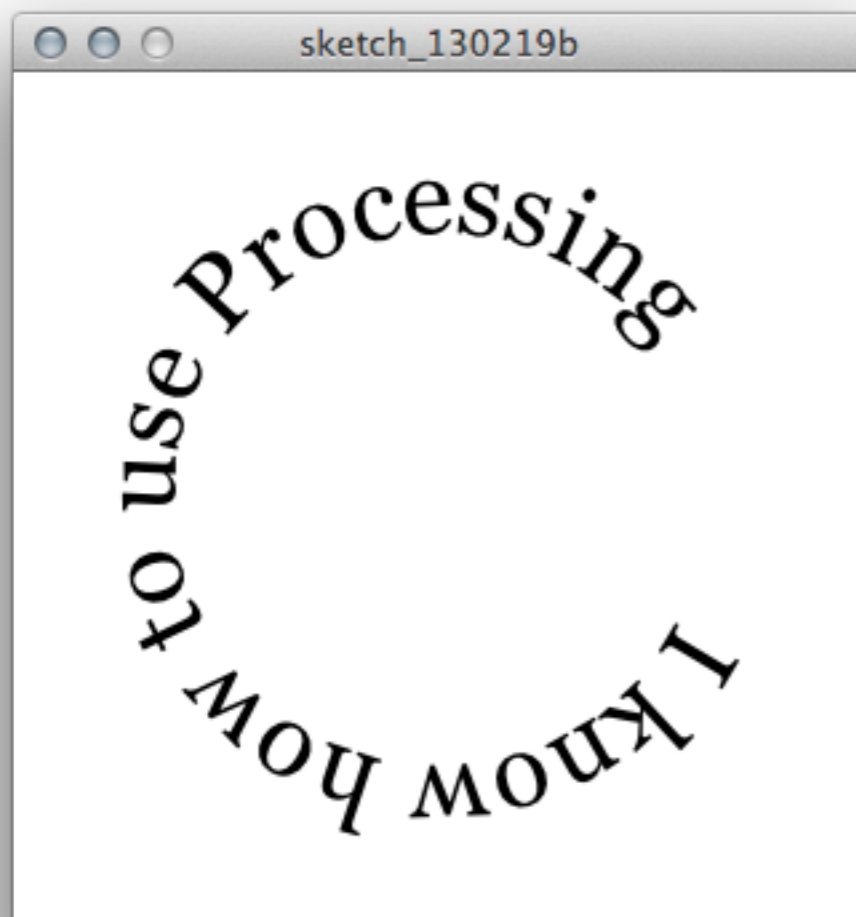
The screenshot shows the Processing IDE interface. The menu bar includes Processing, File, Edit, Sketch, Tools, and Help. The window title is "sketch\_121121a | Processing 2.0b6". The toolbar contains icons for running, stopping, saving, and other actions, along with a "JAVA" button. The code editor shows the following code:

```
PFont f;  
  
void setup(){  
  size(255,255);  
  f = createFont("mizike",48,true);  
}  
  
void draw(){  
  textFont(f,48);  
  fill(255, mouseX,mouseY);  
  text("Hello World!",10,128);  
}
```

<http://processing.org/>

# Intro to Processing

- Going nuts!



```
void setup() {
  size(320, 320);
  f = createFont("Georgia", 40, true);
  textFont(f);
  // The text must be centered!
  textAlign(CENTER);
  smooth();
}

void draw() {
  background(255);

  // Start in the center and draw the circle
  translate(width / 2, height / 2);
  noFill();
  stroke(0);

  // We must keep track of our position along the curve
  float arclength = 2*mouseX;

  // For every box
  for (int i = 0; i < message.length(); i++)
  {
    // Instead of a constant width, we check the width of each character.
    char currentChar = message.charAt(i);
    float w = textWidth(currentChar);

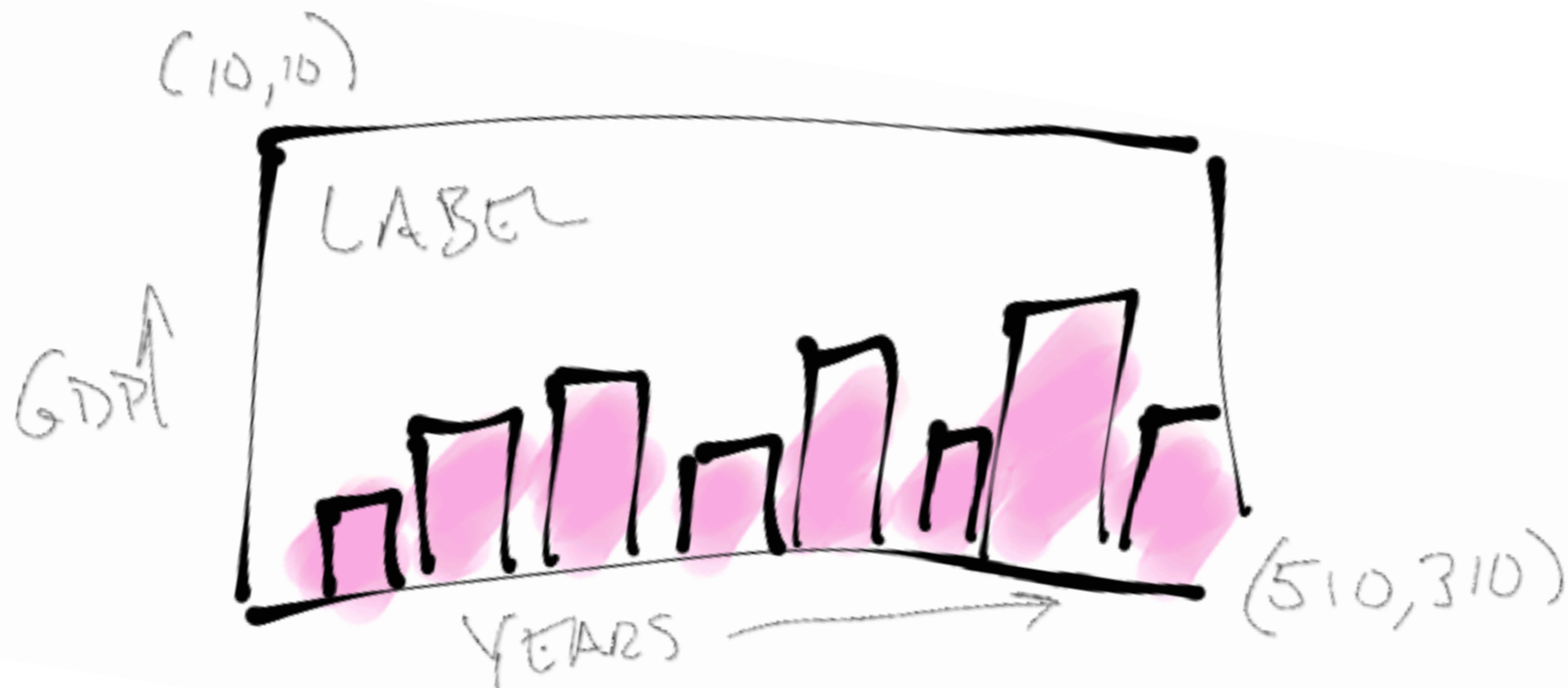
    // Each box is centered so we move half the width
    arclength += w/2;
    // Angle in radians is the arclength divided by the radius
    // Starting on the left side of the circle by adding PI
    float theta = PI + arclength / r;

    pushMatrix();
    // Polar to cartesian coordinate conversion
    translate(r*cos(theta), r*sin(theta));
    // Rotate the box
    rotate(theta+PI/2); // rotation is offset by 90 degrees
    // Display the character
    fill(0);
    text(currentChar, 0, 0);
    popMatrix();
    // Move halfway again
    arclength += w/2;
  }
}
```

<http://processing.org/>

# Intro to Processing

- Let's walk through how to make a graph in Processing
  - Plan out your graph
  - What data do you want to graph?
  - How is it going to be laid out?
  - What is it going to look like?

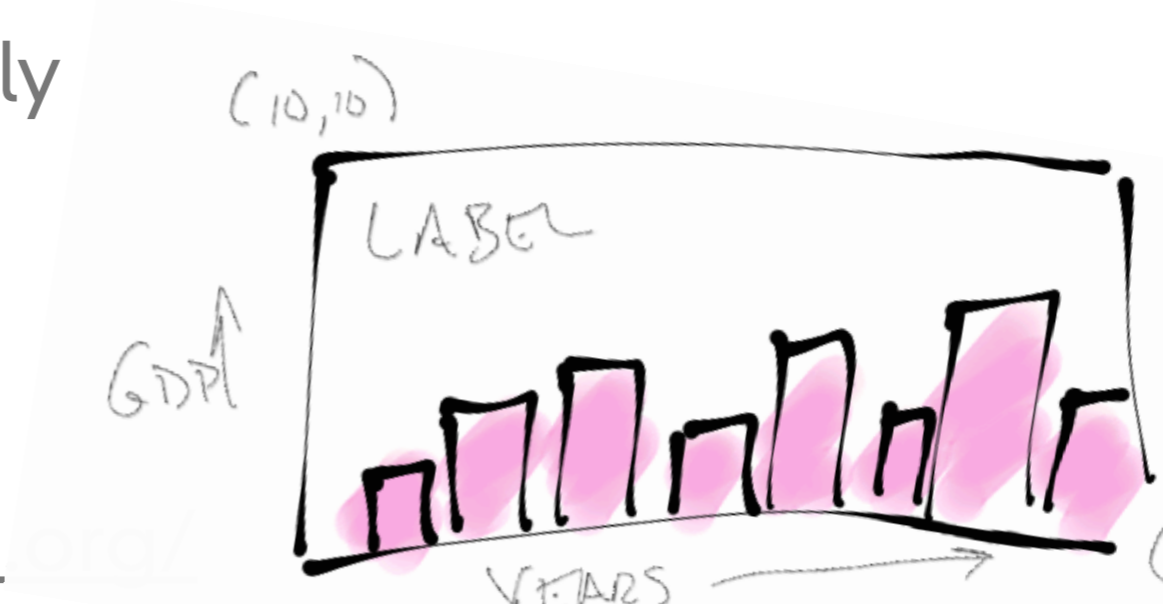


<http://processing.org/>

# Intro to Processing

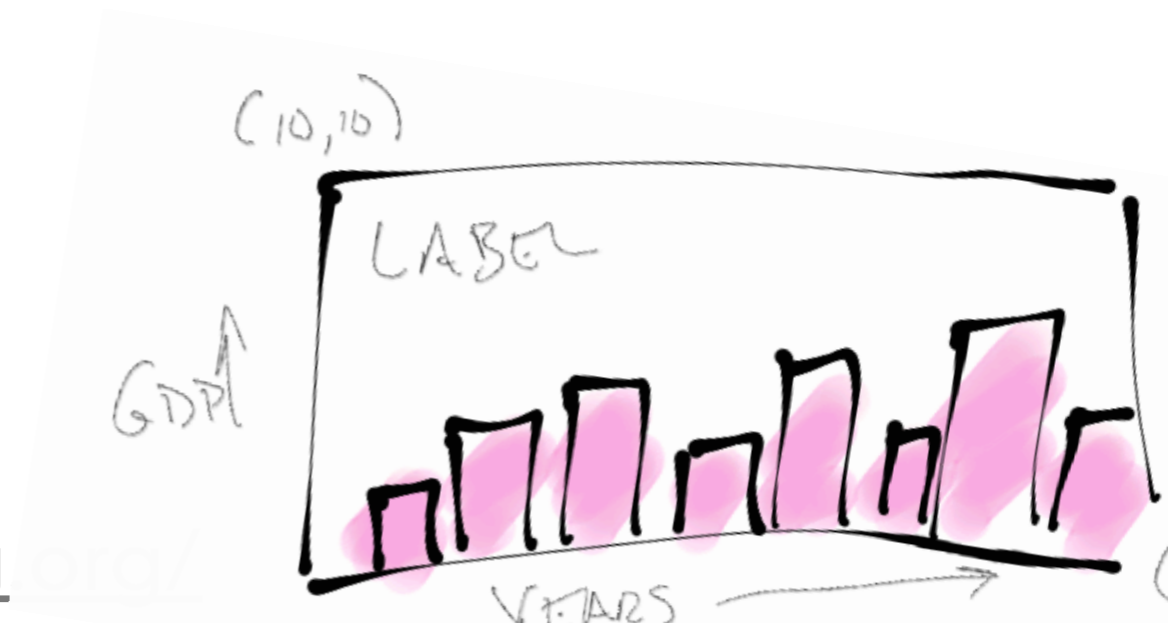
- Let's walk through how to make a graph in Processing
  - Option 1:
    - Draw each square on a bar chart individually
      - Calculate the corners of the graph
        - Draw a black line for the border
      - Calculate how many bars you have
      - Figure out how wide to make them
      - Figure out how tall to make them
        - Convert the data to pixel distances
      - Figure out where the corners are
      - Draw each shape individually

<http://processing.org/>



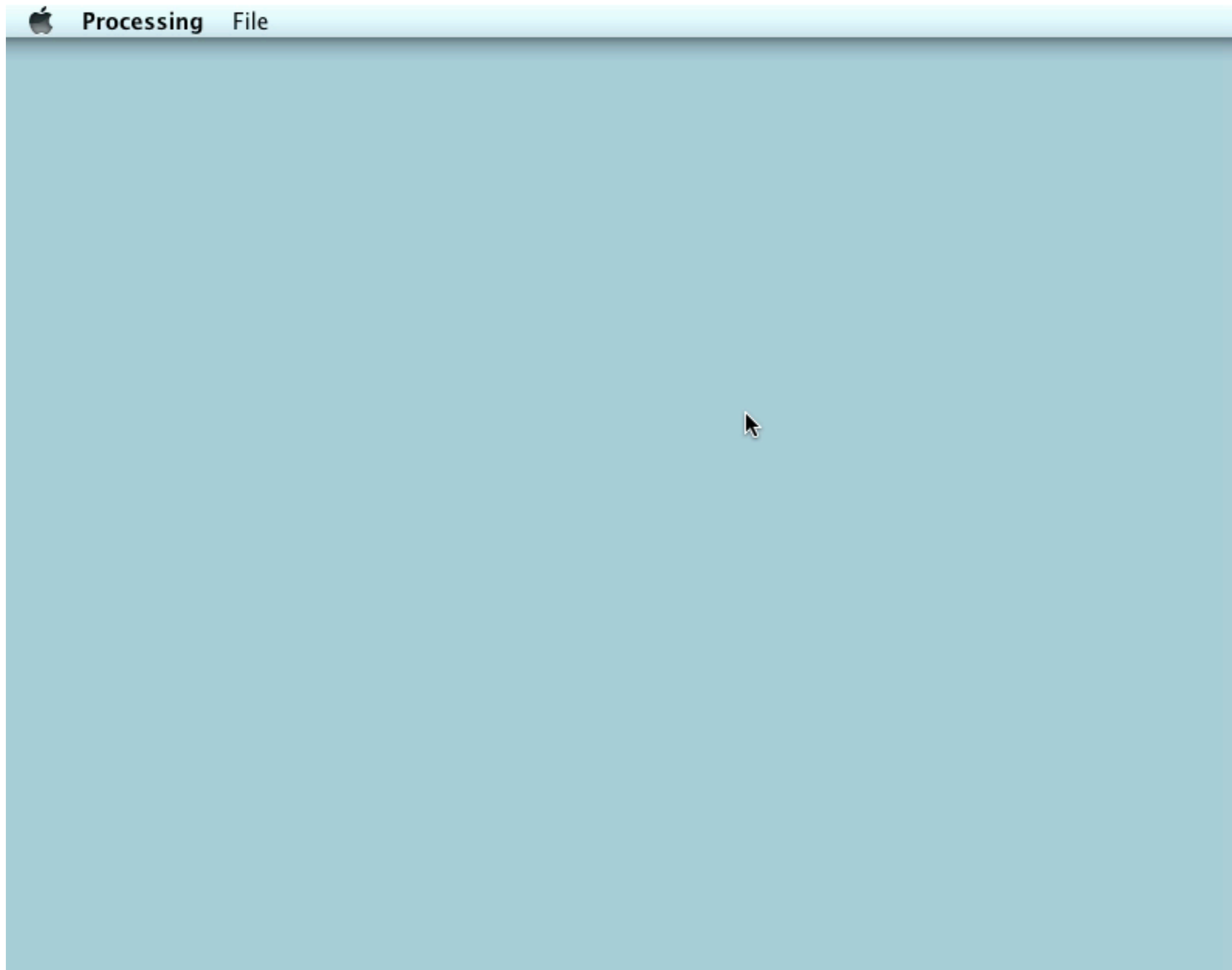
- Let's walk through how to make a graph in Processing
  - Option 2:
    - Use a computer code that someone else has written to do all that and focus on the design.
    - This is called using a **library**
    - You must "install" the library so that you can use it

<http://processing.org/>





# Intro to Processing



<http://processing.org/>

- How do you use the library?
  - Include the extra code that is going to do the hard work for you
    - At the top of the sketch, tell Processing you are going to use a library

---

```
import org.gicentre.utils.stat.*;           // For chart classes.
```

<http://processing.org/>

- How do you use the library?
  - Create some variables to represent the chart.

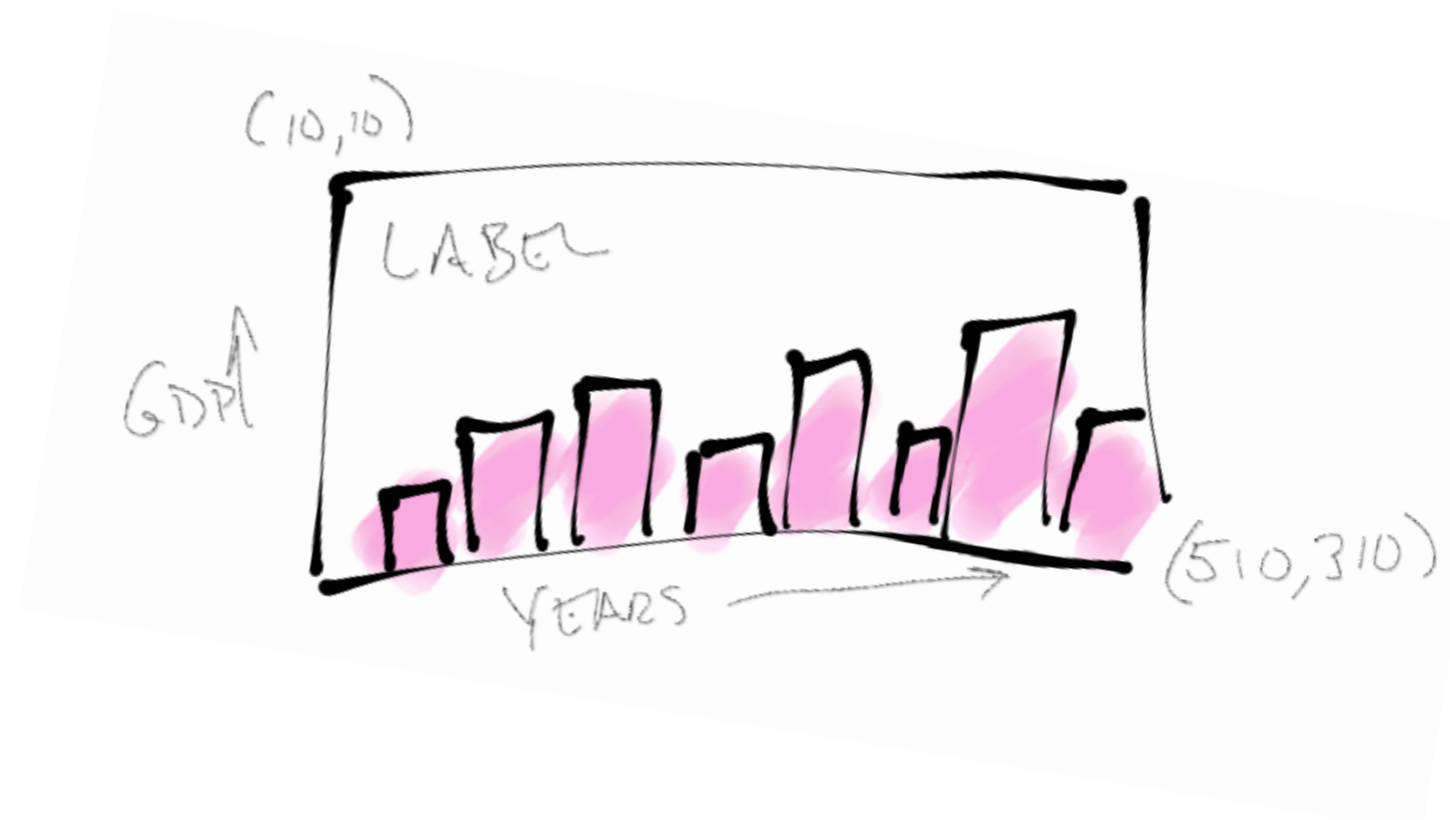
```
BarChart barChart;  
PFont titleFont, smallFont;
```

<http://processing.org/>

# Intro to Processing

- How do you use the library?
  - Inside `setup()`
    - Do some basic setup:

```
size(520, 320);  
smooth();  
noLoop();
```

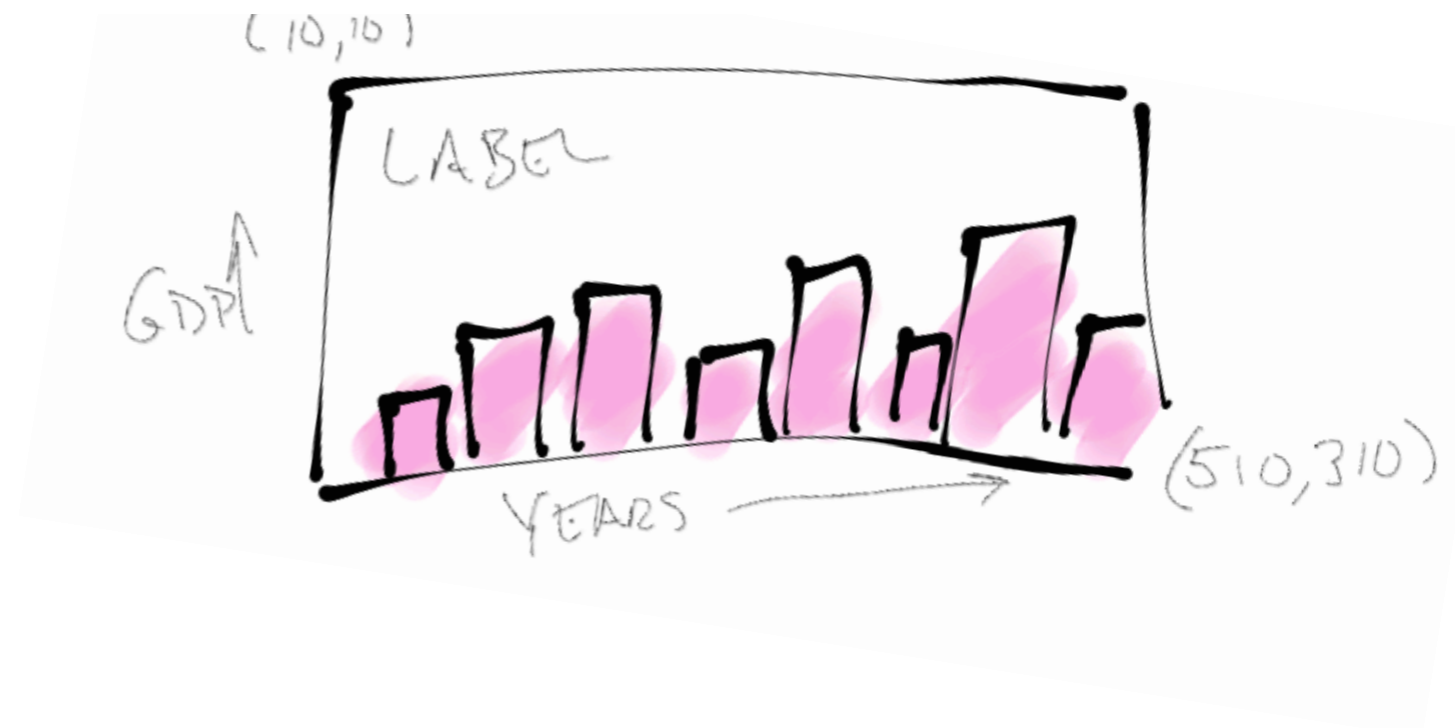


<http://processing.org/>

# Intro to Processing

- How do you use the library?
  - Inside `setup()`
    - Pick the fonts for the graph:

```
titleFont = createFont("Helvetica",22);  
smallFont = createFont("Helvetica",10);  
textFont(smallFont);
```



<http://processing.org/>

- How do you use the library?
  - Inside `setup()`
    - Build the bar chart from the data

```
barChart = new BarChart(this);
barChart.setData(new float[] {2462,2801,3280,3983, 4490, 4894, 5642, 6322, 6489,
                              6401,7657,9649,9767,12167,15154,18200,23124,28645});
barChart.setBarLabels(new String[] {"1830", "1840", "1850", "1860", "1870", "1880", "1890",
                                     "1900", "1910", "1920", "1930", "1940", "1950", "1960",
                                     "1970", "1980", "1990", "2000"});
barChart.setBarColour(color(200,80,80,100));
barChart.setBarGap(2);
barChart.setValueFormat("$###,###");
barChart.showValueAxis(true);
barChart.showCategoryAxis(true);
```

<http://processing.org/>

- How do you use the library?
  - Inside `draw()`
    - Display the bar chart

```
void draw()
{
  background(255);

  barChart.draw(10,10,width-10,height-10);
  fill(120);
  textFont(titleFont);
  text("Income per person, United Kingdom", 70,30);
  float textHeight = textAscent();
  textFont(smallFont);
  text("Gross domestic product measured in inflation-corrected $US", 70,30+textHeight);
}
```



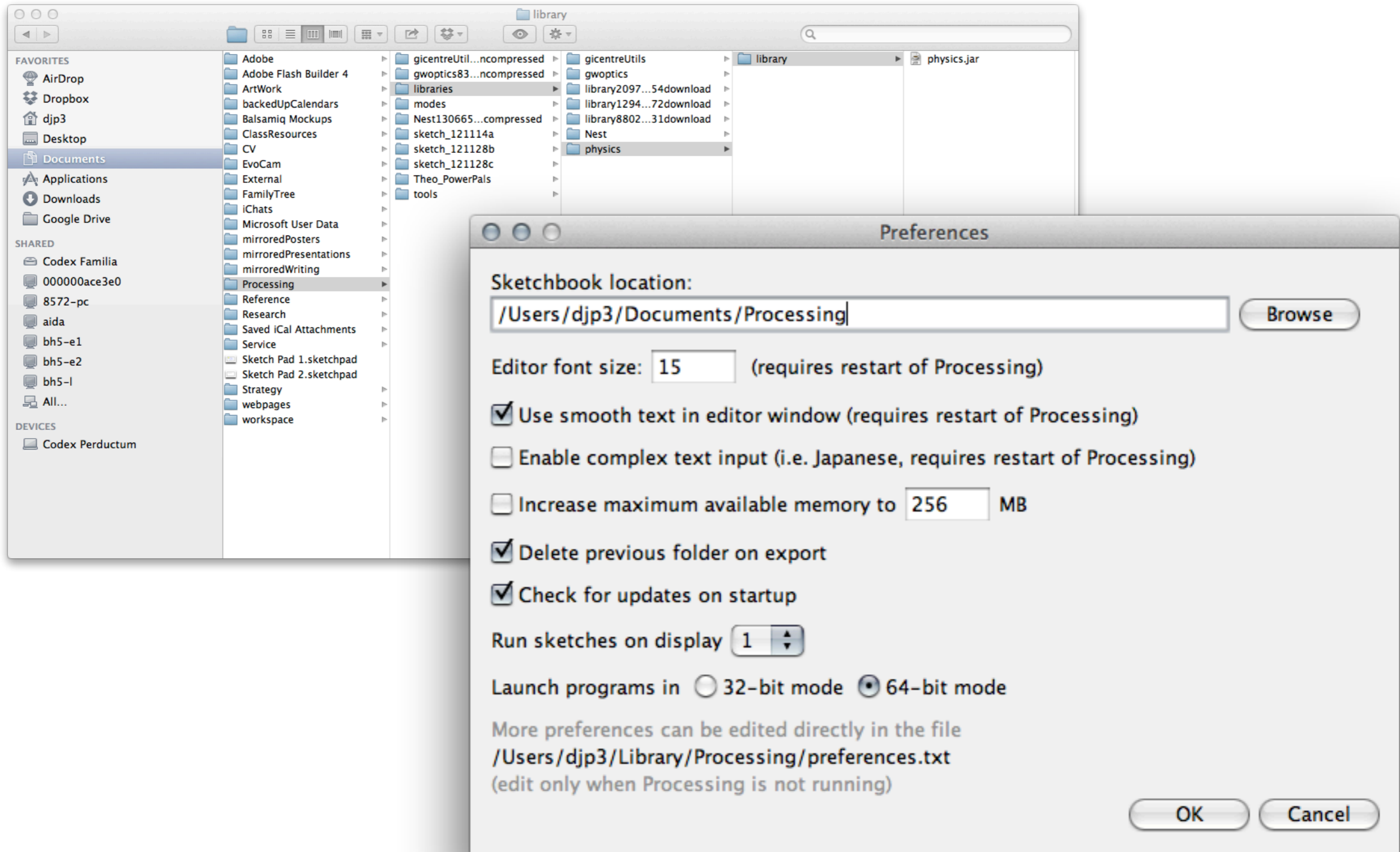
- Where is the information on how to use the library?

<http://processing.org/>



# Intro to Processing

- Manual method of installing a library



<http://processing.org/>



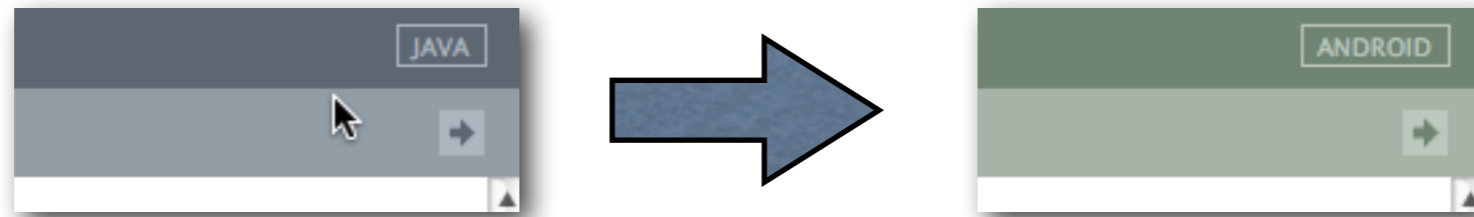
Goldfrapp: Solar

## Making the phone work

- Turn on developer mode
  - “home”->“menu”->“settings”->“applications” -> “Development”
    - “USB debugging” on
    - “Stay awake” on
    - “Allow mock locations” on
  - Dial **\*###CHECKIN##\***
    - to update phone software

<http://www.google.com/support/android/bin/topic.py?hl=en&topic=28930>

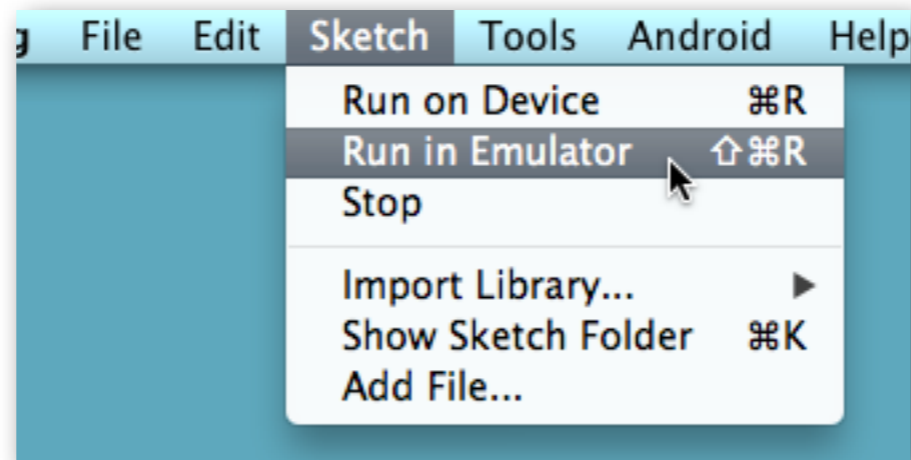
# Switch to Android Mode



- You will be asked where you put the SDK on your hard drive



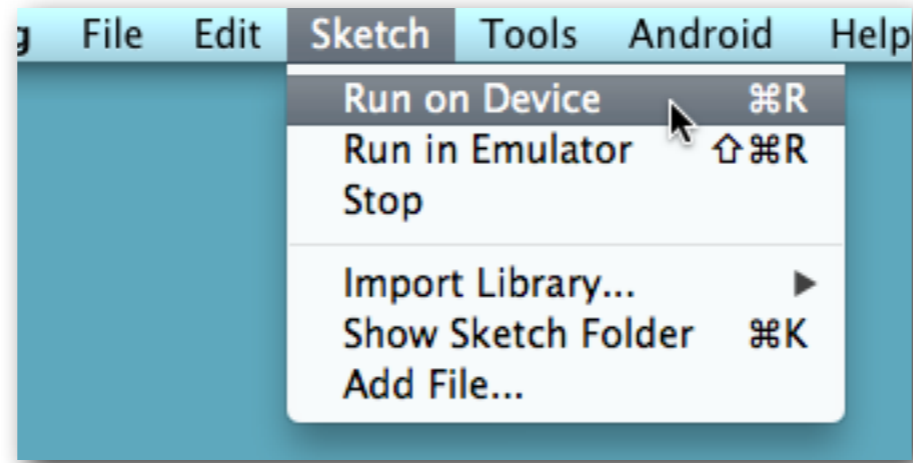
# Run in the emulator



- What is the emulator?
- Make sure you set your sketch to the right resolution
  - 480x800?
- Demo



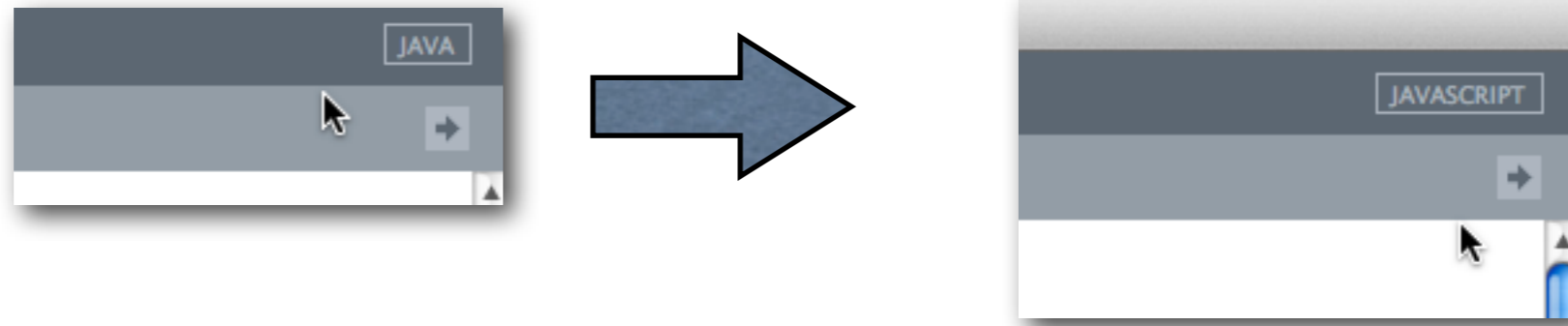
# Run on a real phone



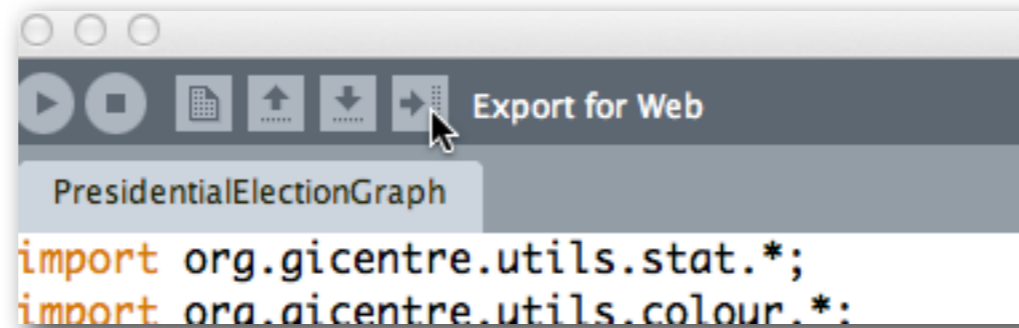
- What is the emulator?
- Make sure you set your sketch to the right resolution
  - 480x800?
- Demo



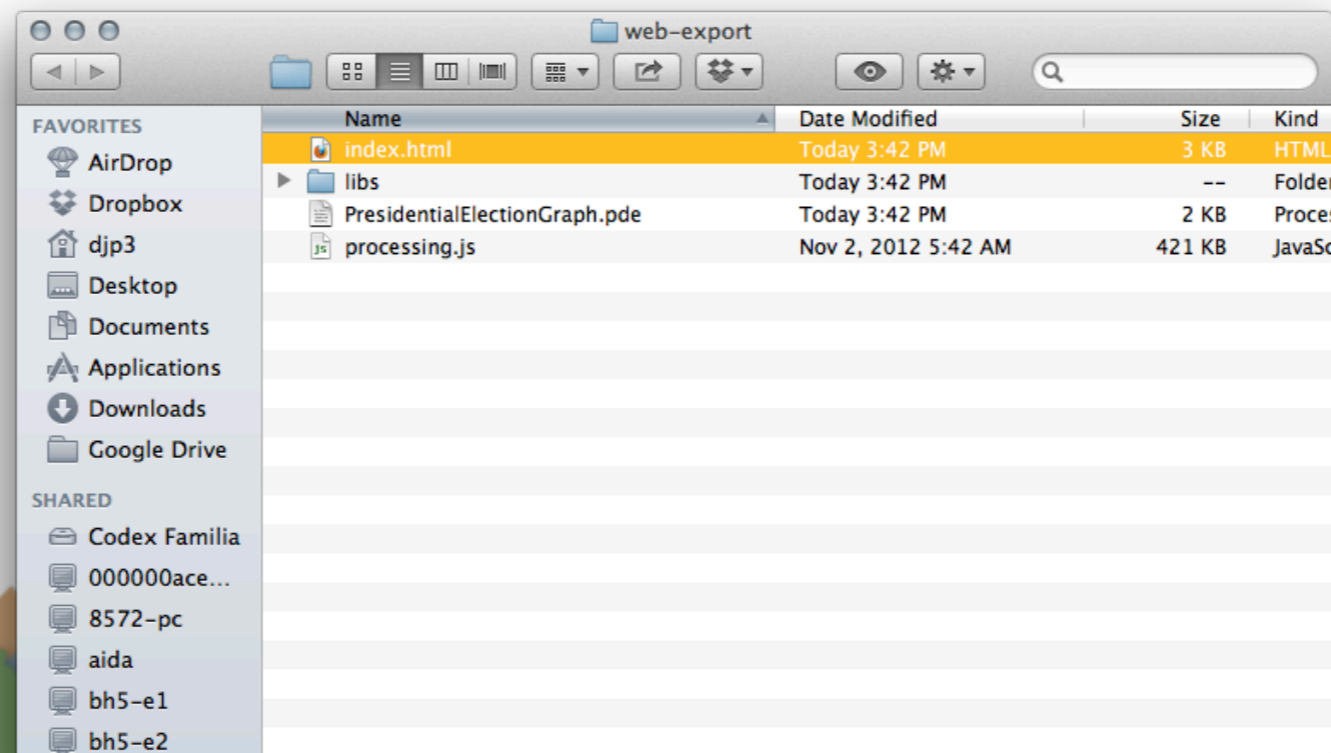
# Run on a website



# Run on a web site

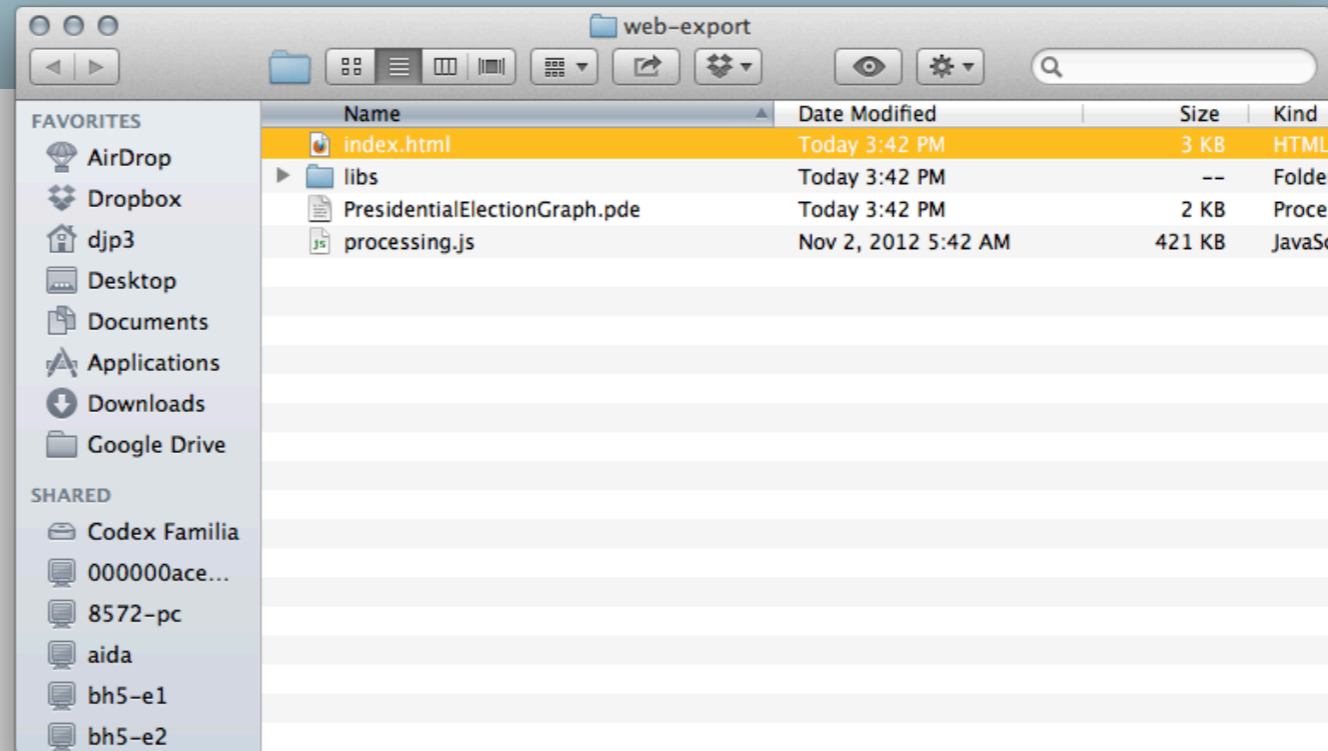


- “Export for Web”
- A file dialog will open showing you where the website is





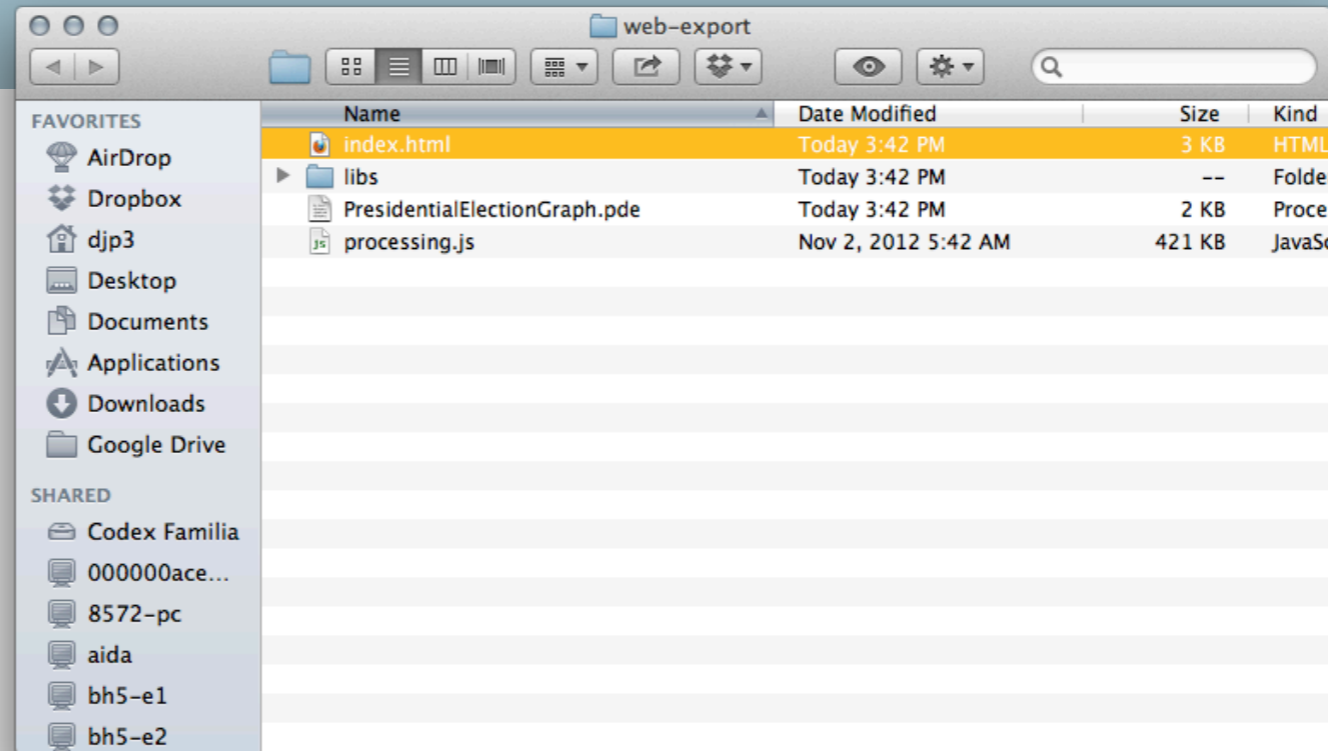
# Run on a web site



- You can open “index.html” in your browser from your hard drive.
- You might need to find the right browser because it uses Java (Chrome doesn't work for me, but Firefox does)



# Run on a web site



- To run it from a web site you need:
  - A server to put the files on
  - Move the files there
  - Then point your browser to the server
  - Demo



# To embed

- To put the sketch on your own website, just pull out the pieces you need from the exported version
- “View source” -> HTML (don't forget the external files)

```
    </style>
    <!--[if lt IE 9]>
      <script type="text/javascript">alert("Your browser does not support the canvas tag.");</script>
    </endif-->
    <script src="processing.js" type="text/javascript"></script>
    <script type="text/javascript">
// convenience function to get the id attribute of generated sketch html element
function getProcessingSketchId () { return 'thanksgiving'; }
</script>

</head>
<body>
  <div id="content">
    <div>
      <canvas id="thanksgiving" data-processing-sources="thanksgiving.pde"
        width="750" height="500">
        <p>Your browser does not support the canvas tag.</p>
        <!-- Note: you can put any alternative content here. -->
      </canvas>
      <noscript>
        <p>JavaScript is required to view the contents of this page.</p>
      </noscript>
    </div>
    <h1>thanksgiving</h1>
    <p id="description"></p>
    <p id="sources">Source code: <a href="thanksgiving.pde">thanksgiving</a> </p>
    <p>
      Built with <a href="http://processing.org" title="Processing">Processing</a>
      and <a href="http://processingjs.org" title="Processing.js">Processing.js</a>
    </p>
  </div>
</body>
</html>
```

## To embed (alternate)

- Link the libraries and put the code in your HTML file

```
<script src="http://processing.org/javascript/MM_functions.js" type="text/javascript"></script>
<script src="http://processing.org/javascript/processing.js" type="text/javascript"></script>
<script src="http://processing.org/javascript/jquery-1.2.6.hardware.js" type="text/javascript"></script>
<script src="http://processing.org/javascript/slideshow.js" type="text/javascript"></script>
```

```
                                <div class="example"><script type="application/processing">
// The message to be displayed
String message = "How to Lie with Infographics";

PFont f;
// The radius of a circle
float r = 100;

void setup() {
  size(320, 320);
  f = createFont("Georgia",40,true);
  textFont(f);
  // The text must be centered!
  textAlign(CENTER);
  smooth();
}

void draw() {
  background(255);

  // Start in the center and draw the circle
  translate(width / 2, height / 2);
  noFill();
  stroke(0);

  // We must keep track of our position along the curve
  float arclength = 2*mouseX;

  // For every box
  for (int i = 0; i < message.length(); i++)
  {
```

To em

- Link

```
<script>  
<script>  
<script>  
<script>
```

```
centerOf(1);  
// The text must be centered!  
textAlign(CENTER);  
smooth();  
}  
  
void draw() {  
  background(255);  
  
  // Start in the center and draw the circle  
  translate(width / 2, height / 2);  
  noFill();  
  stroke(0);  
  
  // We must keep track of our position along the curve  
  float arclength = 2*mouseX;  
  
  // For every box  
  for (int i = 0; i < message.length(); i++)  
  {  
    // Instead of a constant width, we check the width of each character.  
    char currentChar = message.charAt(i);  
    float w = textWidth(currentChar);  
  
    // Each box is centered so we move half the width  
    arclength += w/2;  
    // Angle in radians is the arclength divided by the radius  
    // Starting on the left side of the circle by adding PI  
    float theta = PI + arclength / r;  
  
    pushMatrix();  
    // Polar to cartesian coordinate conversion  
    translate(r*cos(theta), r*sin(theta));  
    // Rotate the box  
    rotate(theta+PI/2); // rotation is offset by 90 degrees  
    // Display the character  
    fill(0);  
    text(currentChar,0,0);  
    popMatrix();  
    // Move halfway again  
    arclength += w/2;  
  }  
}
```

```
</script>
```

```
!important; "></canvas>
```

```
<canvas width="640" height="360" tabindex="0" id="__processing
```

```
<p><strong>Inspiration for Exercise 2</strong></p>
```

```
</div>
```

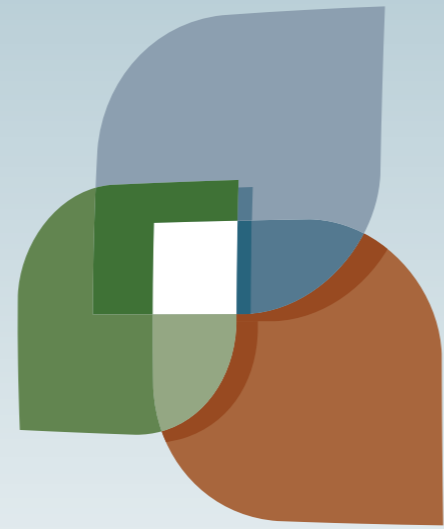
# Assignment 05

- Where is the information on how to use the library?

<http://processing.org/>



Weird Fishes: Radiohead



L U C I

