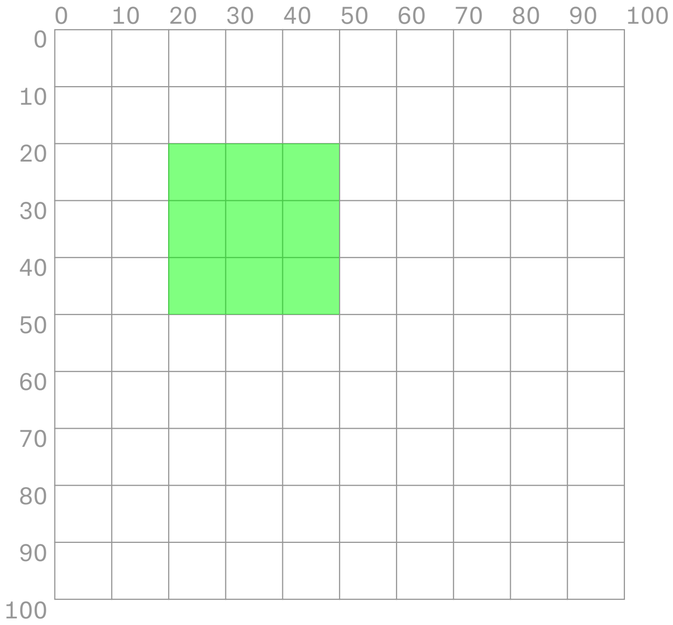
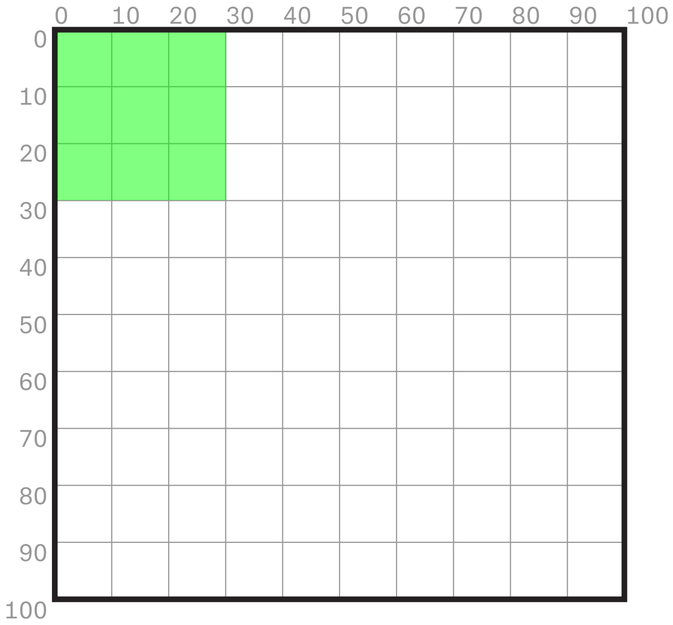
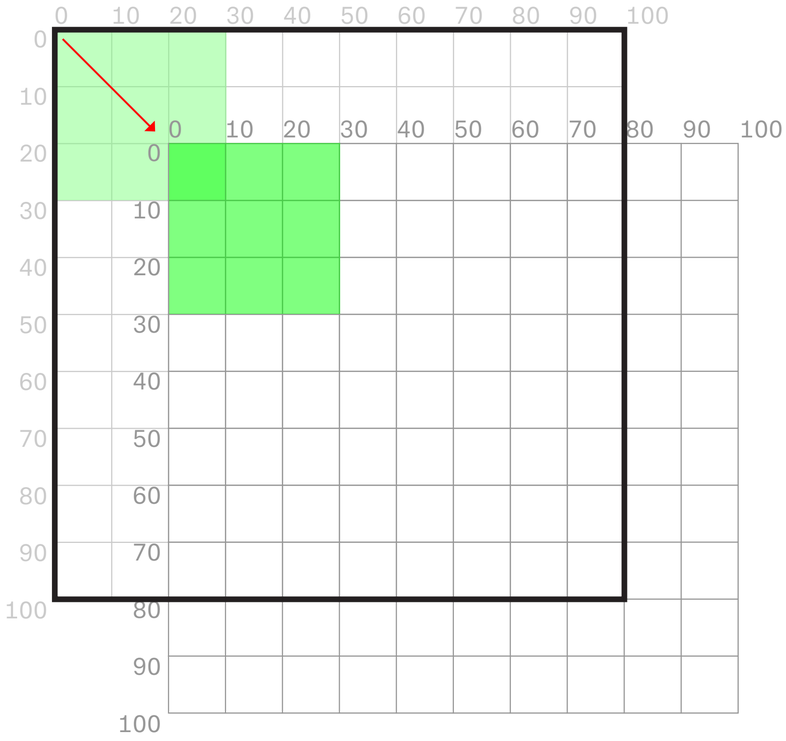
2D Transformations



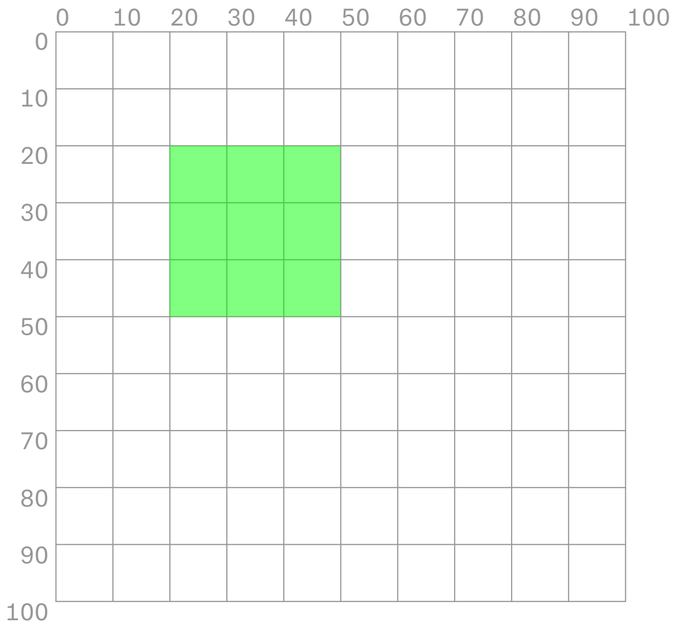

```
rect(20, 20, 30, 30);
```



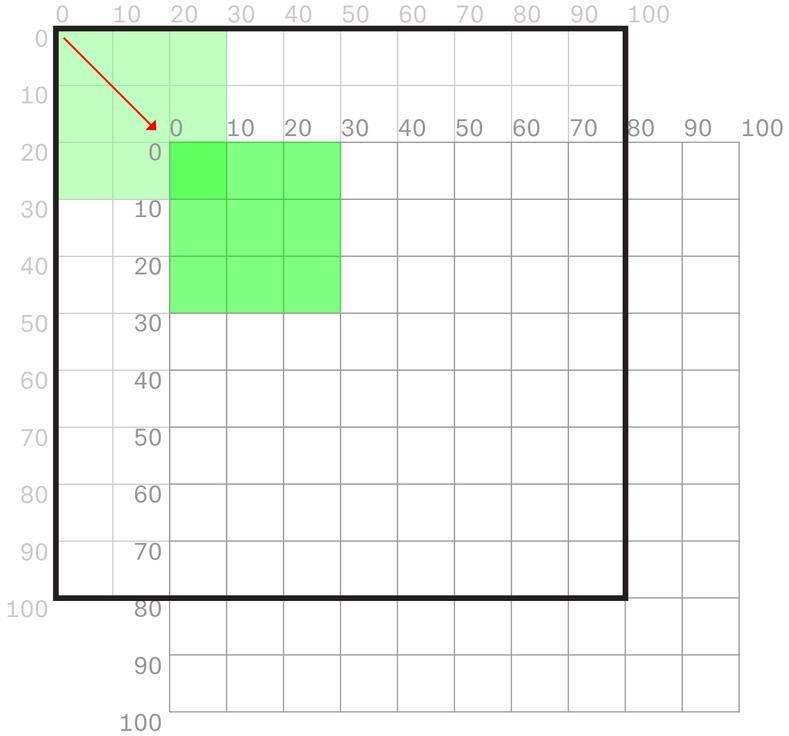

```
rect(0, 0, 30, 30);
```



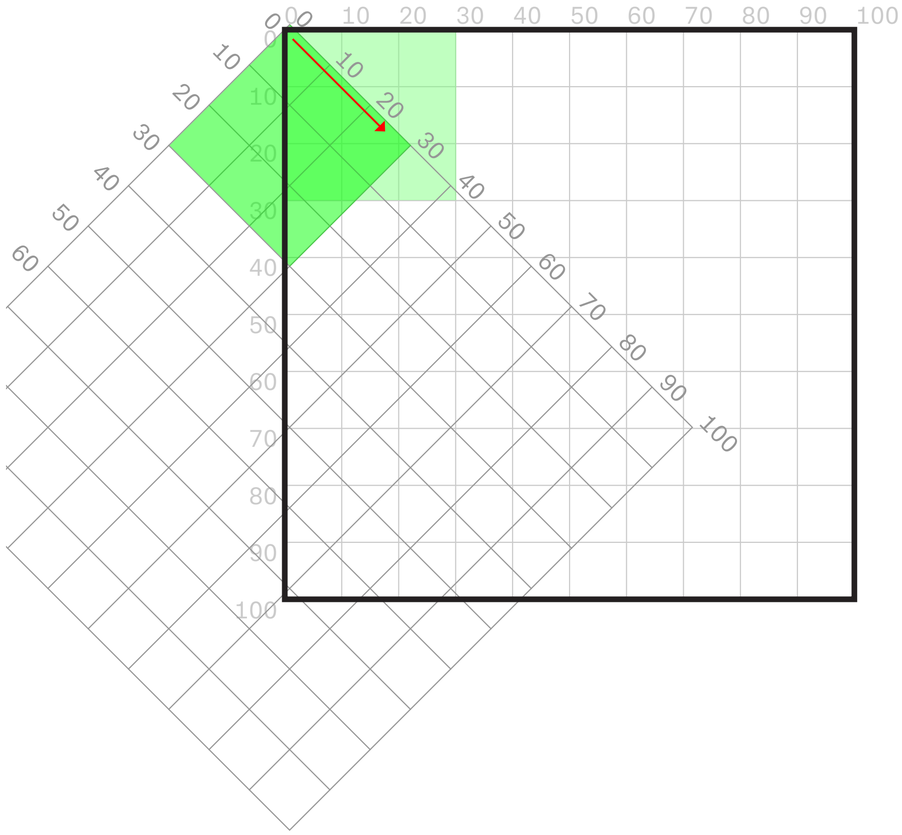

```
translate(20, 20);
rect(0, 0, 30, 30);
```




```
rect(20, 20, 30, 30);
```




```
translate(20, 20);
rect(0, 0, 30, 30);
```




```
rotate(rotate(45));
rect(0, 0, 30, 30);
```

Transformations_1_A | Processing 3.2.1

Java

Transformations_1_A

```
void setup() {
  size(594, 841);
  background(255);
}

void draw() {
  background(255);

  noFill();
  rect(0, 0, width, height);

  fill(255, 0, 0);
  rect(0, 0, 100, 100);
}
```

Console Errors

Transformations_1_B | Processing 3.2.1

Transformations_1_B

```
void setup() {
  size(594, 841);
  background(255);
}

void draw() {
  background(255);

  rotate(.1);

  noFill();
  rect(0, 0, width, height);

  fill(255, 0, 0);
  rect(0, 0, 100, 100);
}
```

Console Errors

Add a rotation

Transformations_1_B | Processing 3.2.1

Transformations_1_B

```
float myRotation = 0;

void setup() {
  size(594, 841);
  background(255);
}

void draw() {
  background(255);

  rotate(.1);

  noFill();
  rect(0, 0, width, height);

  fill(255, 0, 0);
  rect(0, 0, 100, 100);
}
```

Console Errors

To have the rectangle rotate, use a variable

Transformations_1_B | Processing 3.2.1

Java

Transformations_1_B

```
float myRotation = 0;

void setup() {
  size(594, 841);
  background(255);
}

void draw() {
  background(255);

  rotate(.1);

  noFill();
  rect(0, 0, width, height);

  fill(255, 0, 0);
  rect(0, 0, 100, 100);

  myRotation += 0.1;
  println(myRotation);
}
```

Console Errors

Transformations_1_B | Processing 3.2.1

Java

Transformations_1_B

```
float myRotation = 0;

void setup() {
  size(594, 841);
  background(255);
}

void draw() {
  background(255);

  rotate(myRotation);

  noFill();
  rect(0, 0, width, height);

  fill(255, 0, 0);
  rect(0, 0, 100, 100);

  myRotation += 0.1;
  println(myRotation);
}
```

```
25.30006
25.40006
25.500061
```

Console Errors

Transformations_1_C | Processing 3.2.1

Transformations_1_C

```
float myRotation = 0;

void setup() {
  size(594, 841);
  background(255);
}

void draw() {
  background(255);

  rotate(myRotation);

  noFill();
  rect(mouseX, mouseY, width, height);

  fill(255, 0, 0);
  rect(0, 0, 100, 100);

  myRotation += 0.1;
  println(myRotation);
}
```

```
20.400042
20.500042
20.600042
```

Draw the rectangle at the mouse position to see how rotation affects the entire matrix of pixels

Transformations_1_C | Processing 3.2.1

Java

Transformations_1_C

```
float myRotation = 0;

void setup() {
  size(594, 841);
  background(255);
}

void draw() {
  background(255);

  rotate(myRotation);

  noFill();
  rect(mouseX, mouseY, width, height);

  fill(255, 0, 0);
  rect(mouseX, mouseY, 100, 100);

  myRotation += 0.1;
  println(myRotation);
}
```

```
20.400042
20.500042
20.600042
```

Console Errors

Transformations_1_D | Processing 3.2.1

Transformations_1_D

```
float myRotation = 0;

void setup() {
  size(594, 841);
  background(255);
}

void draw() {
  background(255);

  rotate(myRotation);

  noFill();
  rect(0, 0, width, height);

  fill(255, 0, 0);
  rect(mouseX, mouseY, 100, 100);

  myRotation += 0.1;
  println(myRotation);
}
```

```
46.09987
46.199867
46.299866
```

Console | Errors

To get the rectangle to rotate around the mouse, rather than around the top-left corner, we need to use a transformation.

First, set the coordinates back to 0,0

Transformations_1_D | Processing 3.2.1

Java

Transformations_1_D

```
float myRotation = 0;

void setup() {
  size(594, 841);
  background(255);
}

void draw() {
  background(255);

  rotate(myRotation);

  noFill();
  rect(0, 0, width, height);

  fill(255, 0, 0);
  rect(0, 0, 100, 100);

  myRotation += 0.1;
  println(myRotation);
}
```

```
151.70006
151.80006
151.90007
```

Console Errors

Transformations_1_D | Processing 3.2.1

Java

Transformations_1_D

```
float myRotation = 0;

void setup() {
  size(594, 841);
  background(255);
}

void draw() {
  background(255);

  rotate(myRotation);

  noFill();
  rect(0, 0, width, height);

  fill(255, 0, 0);
  rect(0, 0, 100, 100);

  myRotation += 0.1;
  println(myRotation);
}
```

```
177.40163
177.50163
177.60164
```

Console Errors

Transformations_1_D | Processing 3.2.1

Transformations_1_D

```
float myRotation = 0;

void setup() {
  size(594, 841);
  background(255);
}

void draw() {
  background(255);

  translate(mouseX, mouseY);
  rotate(myRotation);

  noFill();
  rect(0, 0, width, height);

  fill(255, 0, 0);
  rect(0, 0, 100, 100);

  myRotation += 0.1;
  println(myRotation);
}
```

```
17.60003
17.700031
17.800032
```

Translate to the mouse position instead. The rectangle should rotate around the mouse.

Transformations_1_E | Processing 3.2.1

Transformations_1_E

```
float myRotation = 0;

void setup() {
  size(594, 841);
  background(255);
}

void draw() {
  background(255);

  translate(mouseX, mouseY);
  rotate(myRotation);

  noFill();
  rect(0, 0, width, height);

  fill(255, 0, 0);
  rect(-50, -50, 100, 100);

  myRotation += 0.1;
  println(myRotation);
}
```

```
6.0999966
6.1999965
6.2999964
```

Draw the rectangle at negative 1/2 of its width and height to have it ‘spin’ around the mouse position.

Transformations_1_F | Processing 3.2.1

Transformations_1_F

```
float myRotation = 0;

void setup() {
  size(594, 841);
  background(255);
}

void draw() {
  background(255);

  translate(mouseX, mouseY);
  rotate(myRotation);

  //noFill();
  //rect(0, 0, width, height);

  fill(255, 0, 0);
  rect(-50, -50, 100, 100);

  myRotation += 0.1;
  println(myRotation);
}
```

```
13.700016
13.800016
13.900017
```

Console | Errors

Remove the ‘guide’ rectangle

Transformations_1_G | Processing 3.2.1

Transformations_1_G

```
float myRotation = 0;

void setup() {
  size(594, 841);
  background(255);
}

void draw() {
  background(255);

  translate(mouseX, mouseY);
  rotate(myRotation);

  //noFill();
  //rect(0, 0, width, height);

  fill(255, 0, 0);
  rect(-50, -50, 100, 100);

  fill(0, 0, 255);
  rect(0, 0, 100, 100);

  myRotation += 0.1;
  println(myRotation);
}
```

```
8.299995
8.399996
8.499996
```

What if we want to have a second, blue, rectangle?

Just like stroke, fill, etc., it will also be affected by the transformations.

But what if we want it to be still in the top-left corner?

Transformations_1_H | Processing 3.2.1

Transformations_1_H

```
float myRotation = 0;

void setup() {
  size(594, 841);
  background(255);
}

void draw() {
  background(255);

  pushMatrix();
  translate(mouseX, mouseY);
  rotate(myRotation);

  //noFill();
  //rect(0, 0, width, height);

  fill(255, 0, 0);
  rect(-50, -50, 100, 100);

  fill(0, 0, 255);
  rect(0, 0, 100, 100);

  myRotation += 0.1;
  println(myRotation);
}
```

By adding pushMatrix()

Transformations_1_H | Processing 3.2.1

Transformations_1_H

```
float myRotation = 0;

void setup() {
  size(594, 841);
  background(255);
}

void draw() {
  background(255);

  pushMatrix();
  translate(mouseX, mouseY);
  rotate(myRotation);

  //noFill();
  //rect(0, 0, width, height);

  fill(255, 0, 0);
  rect(-50, -50, 100, 100);
  popMatrix();

  fill(0, 0, 255);
  rect(0, 0, 100, 100);

  myRotation += 0.1;
  println(myRotation);
}
```

```
1.9000003
2.0000002
2.1000001
```

And popMatrrix() around the elements to be transformed, we isolate the transformation.

Only the elements created by code ‘in-between’ pushMatrix() and popMatrix() will be transformed.

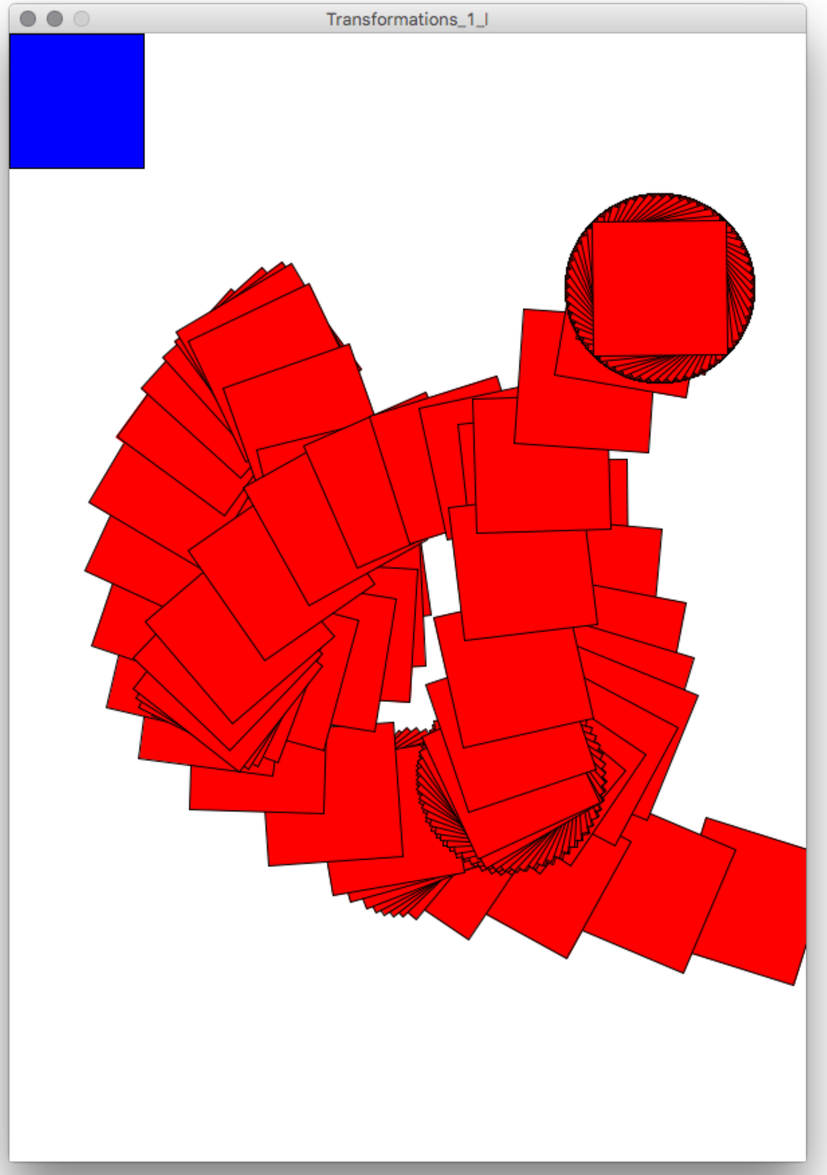
Transformations_1_I | Processing 3.2.1

Transformations_1_I

```
float myRotation = 0;

void setup() {
  size(594, 841);
  background(255);
}

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

  pushMatrix();
  translate(mouseX, mouseY);
  rotate(myRotation);

  //noFill();
  //rect(0, 0, width, height);

  fill(255, 0, 0);
  rect(-50, -50, 100, 100);
  popMatrix();

  fill(0, 0, 255);
  rect(0, 0, 100, 100);

  myRotation += 0.1;
  println(myRotation);
}
```

```
9.6
9.700001
9.800001
```

Console | Errors



Without the background...

Transformations_1_J | Processing 3.2.1

Transformations_1_J

```
float myRotation = 0;

void setup() {
  size(594, 841);
  background(255);
}

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

  pushMatrix();
  translate(mouseX, mouseY);
  scale(random(1.5));
  rotate(myRotation);

  //noFill();
  //rect(0, 0, width, height);

  fill(255, 0, 0);
  rect(-50, -50, 100, 100);
  popMatrix();

  fill(0, 0, 255);
  rect(0, 0, 100, 100);

  myRotation += 0.1;
  println(myRotation);
}
```

```
24.500057
24.600058
24.700058
```

Console | Errors

scale() works in the same way as rotation.

Poster

RandomPoster_02_B | Processing 3.2.1

RandomPoster_02_B

```
import processing.pdf.*;

String myFilepathPDF = "data/posters/pdf/poster-" + year() + "-" + month() + "-" + day() + "_" +
String myFilepathPNG = "data/posters/png/poster-" + year() + "-" + month() + "-" + day() + "_" +

// define minimum and maximum possible X positions
float myMinX = -100;
float myMaxX = 694; // width + 100

// define minimum and maximum possible Y positions
float myMinY = -100;
float myMaxY = 941; // height + 100

// define minimum and maximum possible scale
float myMinScale = 0.2; // 10% of original size
float myMaxScale = 0.8; // 300% of original size

// define minimum and maximum possible stroke weights
float myMinStrokeWeight = 1;
float myMaxSgtrokeWeight = 5;

// definite a minimum and maximum rotation
float myMinRotation = 0;
float myMaxRotation = 360;

void setup() {
  size(594, 841);
  background(255);
  beginRecord(PDF, myFilepathPDF);
}
```

To implement random rotation in our poster example:

create some variables for the minimum and maximum rotation.

RandomPoster_02_B | Processing 3.2.1

Java

RandomPoster_02_B

```
  float myXPosition = random(myMinX, myMaxX);
  float myYPosition = random(myMinY, myMaxY);

  // determione a random scale
  float myScale = random(myMinScale, myMaxScale);
  float myShapeWidth = myShape.width * myScale;
  float myShapeHeight = myShape.height * myScale;

  // determine a random stroke weight
  float myStrokeWeight = random(myMinStrokeWeight, myMaxSgtrokeWeight);

  // determine a random rotation
  float myRotation = random(myMinRotation, myMaxRotation);

  // disable the shape's stroke and fill
  myShape.disableStyle();

  // set stroke and fill
  stroke(255);
  strokeWeight(myStrokeWeight);
  fill(random(255), random(255), random(255));

  shape(myShape, myXPosition, myYPosition, myShapeWidth, myShapeHeight);
}

void keyPressed() {
  endRecord();
  save(myFilepathPNG);
  exit();
}
```

The value of the local variable "myRotation" is not used

in draw() create a variable for the rotation...a random number between the minimum and maximum that we set above.

RandomPoster_02_B | Processing 3.2.1

RandomPoster_02_B

```
  float myXPosition = random(myMinX, myMaxX);
  float myYPosition = random(myMinY, myMaxY);

  // determione a random scale
  float myScale = random(myMinScale, myMaxScale);
  float myShapeWidth = myShape.width * myScale;
  float myShapeHeight = myShape.height * myScale;

  // determine a random stroke weight
  float myStrokeWeight = random(myMinStrokeWeight, myMaxSgtrokeWeight);

  // determine a random rotation
  float myRotation = random(myMinRotation, myMaxRotation);

  // disable the shape's stroke and fill
  myShape.disableStyle();

  // set stroke and fill
  stroke(255);
  strokeWeight(myStrokeWeight);
  fill(random(255), random(255), random(255));

  // instead of drawing the shape at this coordinate, use a transformation
  translate(myXPosition, myYPosition);

  shape(myShape, myXPosition, myYPosition, myShapeWidth, myShapeHeight);
}

void keyPressed() {
  endRecord();
  save(myFilepathPNG);
```

Because we will use rotation, we need to position the shape using translation, rather than the coordinates in the shape() function.

RandomPoster_02_B | Processing 3.2.1

Java

RandomPoster_02_B

```
  float myXPosition = random(myMinX, myMaxX);
  float myYPosition = random(myMinY, myMaxY);

  // determione a random scale
  float myScale = random(myMinScale, myMaxScale);
  float myShapeWidth = myShape.width * myScale;
  float myShapeHeight = myShape.height * myScale;

  // determine a random stroke weight
  float myStrokeWeight = random(myMinStrokeWeight, myMaxSgtrokeWeight);

  // determine a random rotation
  float myRotation = random(myMinRotation, myMaxRotation);

  // disable the shape's stroke and fill
  myShape.disableStyle();

  // set stroke and fill
  stroke(255);
  strokeWeight(myStrokeWeight);
  fill(random(255), random(255), random(255));

  // instead of drawing the shape at this coordinate, use a transformation
  translate(myXPosition, myYPosition);

  shape(myShape, 0, 0, myShapeWidth, myShapeHeight);
}

void keyPressed() {
  endRecord();
  save(myFilepathPNG);
```

RandomPoster_02_B | Processing 3.2.1

Java

RandomPoster_02_B

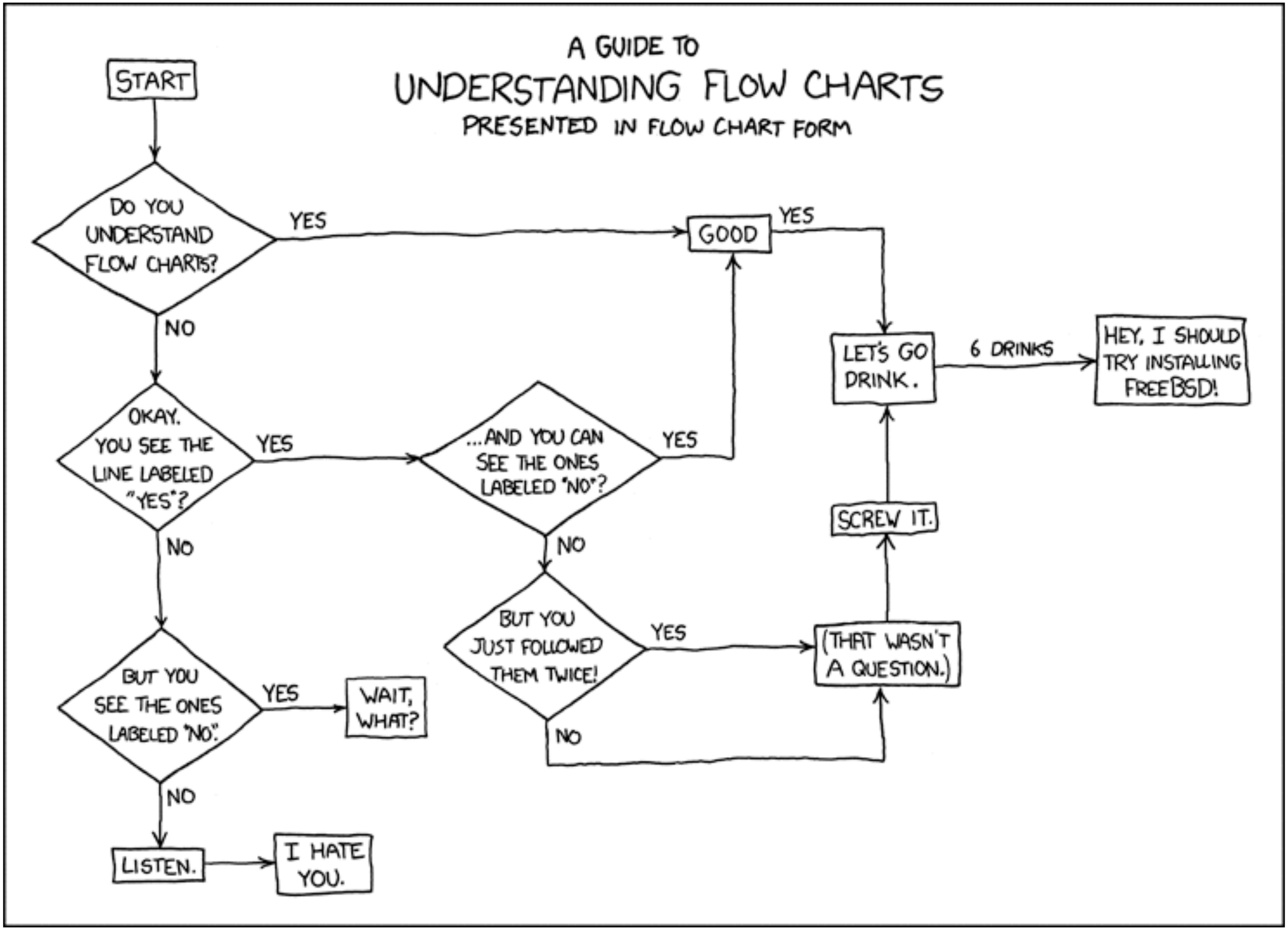
```
  float myShapeHeight = myShape.height * myScale;

  // determine a random stroke weight
  float myStrokeWeight = random(myMinStrokeWeight, myMaxSgtrokeWeight);

  // determine a random rotation
  float myRotation = random(myMinRotation, myMaxRotation);

  // disable the shape's stroke and fill
  myShape.disableStyle();

  // set stroke and fill
  stroke(255);
  strokeWeight(myStrokeWeight);
  fill(random(255), random(255), random(255));

  // instead of drawing the shape at this coordinate, use a transformation
  translate(myXPosition, myYPosition);

  // rotate the shape
  rotate(myRotation);

  shape(myShape, 0, 0, myShapeWidth, myShapeHeight);
}

void keyPressed() {
  endRecord();
  save(myFilepathPNG);
  exit();
}
```

And the rotation.

Conditionals

Conditional Statments

```
if

else

else if
```


A GUIDE TO
UNDERSTANDING FLOW CHARTS
PRESENTED IN FLOW CHART FORM
START
DO YOU UNDERSTAND FLOW CHARTS?
YES
GOOD
YES
NO
OKAY. YOU SEE THE LINE LABELED "YES"?
YES
...AND YOU CAN SEE THE ONES LABELED "NO"?
YES
LET'S GO DRINK.
6 DRINKS
HEY, I SHOULD TRY INSTALLING FREEBSD!
NO
NO
SCREW IT.
BUT YOU JUST FOLLOWED THEM TWICE!
YES
(THAT WASN'T A QUESTION.)
BUT YOU SEE THE ONES LABELED "NO".
YES
WAIT, WHAT?
NO
NO
LISTEN.
I HATE YOU.

```
if a person is over 18
    they can vote
else
    they cannot vote
```

```
if (person >= 18) {
    // they can vote
} else {
    // they cannot vote
}
```

Relational Operators

`>`	Greater than
`>=`	Greater than or equal to
`<`	Less than
`<=`	Less than or equal to
`==`	Equal to
`!=`	Not equal to

```
if (this expression is true) {
    // run this code
} else {
    // run this code
}
```

```
int age = 68;
```

```
if (age >= 65) {
    println("Retire!");
} else {
    println("Get to work!");
}
```

```
int age = 68;
```

true

```
if (age >= 65) {
    println("Retire!");
} else {
    println("Get to work!");
}
```

```
int age = 68;
```

```
if (age >= 65) {
    println("Retire!");
} else {
    println("Get to work!");
}
```

```
int age = 22;
```

```
if (age >= 65) {
    println("Retire!");
} else {
    println("Get to work!");
}
```

false

```
int age = 22;
```

```
if (age >= 65) {
    println("Retire!");
} else {
    println("Get to work!");
}
```

```
int age = 22;
```

```
if (age >= 65) {
    println("Retire!");
} else {
    println("Get to work!");
}
```

Logical Operators

&&	AND
||	OR
!	NOT

```
float temp = 28.6;
boolean sunshine = true;
```

```
if (temp > 25 && sunshine == true) {
     println("Go to the beach");
} else {
     println("Go to the movies");
}
```

```
float temp = 28.6;
boolean sunshine = true;


        true                    true

if (temp > 25 && sunshine == true) {
     println("Go to the beach");
} else {
     println("Go to the movies");
}
```

```
float temp = 28.6;
boolean sunshine = true;


                true
                  |
if (temp > 25 && sunshine == true) {
     println("Go to the beach");
} else {
     println("Go to the movies");
}
```

```
float temp = 28.6;
boolean sunshine = true;
```

```
if (temp > 25 && sunshine == true) {
    println("Go to the beach");
} else {
    println("Go to the movies");
}
```

```
float temp = 16.2;
boolean sunshine = true;
```

```
if (temp > 25 && sunshine == true) {
    println("Go to the beach");
} else {
    println("Go to the movies");
}
```

```
float temp = 16.2;
boolean sunshine = true;
```

```
if (temp > 25 && sunshine == true) {
    println("Go to the beach");
} else {
    println("Go to the movies");
}
```

```
float temp = 16.2;
boolean sunshine = true;
```

```
if (temp > 25 && sunshine == true) {
     println("Go to the beach");
} else {
     println("Go to the movies");
}
```

```
float temp = 16.2;
boolean sunshine = true;
```

```
if (temp > 25 && sunshine == true) {
    println("Go to the beach");
} else {
    println("Go to the movies");
}
```

```
float temp = 16.2;
boolean sunshine = true;
```

```
if (temp > 25 && sunshine == true) {
    println("Go to the beach");
} else {
    println("Go to the movies");
}
```

```
float temp = 16.2;
boolean sunshine = true;
```

```
if (temp > 25 || sunshine == true) {
     println("Go to the beach");
} else {
     println("Go to the movies");
}
```

```
float temp = 16.2;
boolean sunshine = true;
```

```
if (temp > 25 || sunshine == true) {
    println("Go to the beach");
} else {
    println("Go to the movies");
}
```

```
float temp = 16.2;
boolean sunshine = true;
```

```
if (temp > 25 || sunshine == true) {
    println("Go to the beach");
} else {
    println("Go to the movies");
}
```

```
float temp = 16.2;
boolean sunshine = true;
```

```
if (temp > 25 || sunshine == true) {
    println("Go to the beach");
} else {
    println("Go to the movies");
}
```

Poster

RandomPoster_02_C | Processing 3.2.1

Java

RandomPoster_02_C

```
import processing.pdf.*;

String myFilepathPDF = "data/posters/pdf/poster-" + year() + "-" + month() + "-" + day
String myFilepathPNG = "data/posters/png/poster-" + year() + "-" + month() + "-" + day

// define minimum and maximum possible X positions
float myMinX = -100;
float myMaxX = 694; // width + 100

// define minimum and maximum possible Y positions
float myMinY = -100;
float myMaxY = 941; // height + 100

// define minimum and maximum possible scale
float myMinScale = 0.2; // 10% of original size
float myMaxScale = 0.8; // 300% of original size

// define minimum and maximum possible stroke weights
float myMinStrokeWeight = 1;
float myMaxSgtrokeWeight = 5;

// define a minimum and maximum rotation
float myMinRotation = 0;
float myMaxRotation = 360;

void setup() {
  size(594, 841);
  background(255);
  beginRecord(PDF, myFilepathPDF);
}

void draw() {
```

Done saving.

RandomPoster_02_D | Processing 3.2.1

Java

RandomPoster_02_D

```
import processing.pdf.*;

String myFilepathPDF = "data/posters/pdf/poster-" + year() + "-" + month() + "-" + day
String myFilepathPNG = "data/posters/png/poster-" + year() + "-" + month() + "-" + day

// define minimum and maximum possible X positions
float myMinX = -100;
float myMaxX = 694; // width + 100

// define minimum and maximum possible Y positions
float myMinY = -100;
float myMaxY = 941; // height + 100

// define minimum and maximum possible scale
float myMinScale = 0.2; // 10% of original size
float myMaxScale = 0.8; // 300% of original size

// define minimum and maximum possible stroke weights
float myMinStrokeWeight = 1;
float myMaxSgtrokeWeight = 5;

// define a minimum and maximum rotation
float myMinRotation = 0;
float myMaxRotation = 360;

// define two boolean variables for fill and stroke
boolean myUseFill = true;
boolean myUseStroke = false;

void setup() {
  size(594, 841);
  background(255);
```

Create two boolean variables.
One for whether or not to use fill.
One for whether or not to use stroke.

RandomPoster_02_E | Processing 3.2.1

Java

RandomPoster_02_E

```
46  float myScale = random(myMinScale, myMaxScale);
47  float myShapeWidth = myShape.width * myScale;
48  float myShapeHeight = myShape.height * myScale;
49
50  // determine a random stroke weight
51  float myStrokeWeight = random(myMinStrokeWeight, myMaxSgtrokeWeight);
52
53  // determine a random rotation
54  float myRotation = random(myMinRotation, myMaxRotation);
55
56  // disable the shape's stroke and fill
57  myShape.disableStyle();
58
59  // set fill and stroke
60  fill(random(255), random(255), random(255));
61  stroke(random(255), random(255), random(255));
62  strokeWeight(myStrokeWeight);
63
64  // instead of drawing the shape at this coordinate, use a transformation
65  translate(myXPosition, myYPosition);
66
67  // rotate the shape
68  rotate(myRotation);
69
70  shape(myShape, 0, 0, myShapeWidth, myShapeHeight);
71 }
72
73 void keyPressed() {
74  endRecord();
75  save(myFilepathPNG);
76  exit();
77 }
```

Previously, we had a random stroke color and random fill color.

RandomPoster_02_E | Processing 3.2.1

Java

RandomPoster_02_E

```
  float myShapeHeight = myShape.height * myScale;

  // determine a random stroke weight
  float myStrokeWeight = random(myMinStrokeWeight, myMaxSgtrokeWeight);

  // determine a random rotation
  float myRotation = random(myMinRotation, myMaxRotation);

  // disable the shape's stroke and fill
  myShape.disableStyle();

  // set fill and stroke
  if (myUseFill == true) {
    fill(random(255), random(255), random(255));
  }

  stroke(random(255), random(255), random(255));
  strokeWeight(myStrokeWeight);

  // instead of drawing the shape at this coordinate, use a transformation
  translate(myXPosition, myYPosition);

  // rotate the shape
  rotate(myRotation);

  shape(myShape, 0, 0, myShapeWidth, myShapeHeight);
}

void keyPressed() {
  endRecord();
  save(myFilepathPNG);
  exit();
```

Auto Format finished.

Create a conditional to check if myUseFill is true.
If it is, set it to a random color.

RandomPoster_02_E | Processing 3.2.1

Java

RandomPoster_02_E

```
  float myShapeHeight = myShape.height * myScale;

  // determine a random stroke weight
  float myStrokeWeight = random(myMinStrokeWeight, myMaxSgtrokeWeight);

  // determine a random rotation
  float myRotation = random(myMinRotation, myMaxRotation);

  // disable the shape's stroke and fill
  myShape.disableStyle();

  // set fill and stroke
  if (myUseFill == true) {
    fill(random(255), random(255), random(255));
  } else {
    noFill();
  }

  stroke(random(255), random(255), random(255));
  strokeWeight(myStrokeWeight);

  // instead of drawing the shape at this coordinate, use a transformation
  translate(myXPosition, myYPosition);

  // rotate the shape
  rotate(myRotation);

  shape(myShape, 0, 0, myShapeWidth, myShapeHeight);
}

void keyPressed() {
  endRecord();
```

Auto Format finished.

Otherwise (else), disable fill.

RandomPoster_02_F | Processing 3.2.1

Java

RandomPoster_02_F

```
51  float myStrokeWeight = random(myMinStrokeWeight, myMaxSgtrokeWeight);
52
53  // determine a random rotation
54  float myRotation = random(myMinRotation, myMaxRotation);
55
56  // disable the shape's stroke and fill
57  myShape.disableStyle();
58
59  // set fill and stroke
60  if (myUseFill == true) {
61    fill(random(255), random(255), random(255));
62  } else {
63    noFill();
64  }
65
66  if (myUseStroke == true) {
67    stroke(random(255), random(255), random(255));
68    strokeWeight(myStrokeWeight);
69  }
70
71  // instead of drawing the shape at this coordinate, use a transformation
72  translate(myXPosition, myYPosition);
73
74  // rotate the shape
75  rotate(myRotation);
76
77  shape(myShape, 0, 0, myShapeWidth, myShapeHeight);
78 }
79
80 void keyPressed() {
81  endRecord();
82  save(myFilepathPNG);
```

Auto Format finished.

Similarly, if myUseStroke is true, set a random stroke and stroke width.

RandomPoster_02_F | Processing 3.2.1

Java

RandomPoster_02_F

```
  float myStrokeWeight = random(myMinStrokeWeight, myMaxSgtrokeWeight);

  // determine a random rotation
  float myRotation = random(myMinRotation, myMaxRotation);

  // disable the shape's stroke and fill
  myShape.disableStyle();

  // set fill and stroke
  if (myUseFill == true) {
    fill(random(255), random(255), random(255));
  } else {
    noFill();
  }

  if (myUseStroke == true) {
    stroke(random(255), random(255), random(255));
    strokeWeight(myStrokeWeight);
  } else {
    noStroke();
  }

  // instead of drawing the shape at this coordinate, use a transformation
  translate(myXPosition, myYPosition);

  // rotate the shape
  rotate(myRotation);

  shape(myShape, 0, 0, myShapeWidth, myShapeHeight);
}

void keyPressed() {
```

Auto Format finished.

Otherwise (else), disable it.

Map

Map_1_B | Processing 3.2.1

Map_1_B

```
void setup() {
  size(594, 841);
  background(255);

  float x = map(10, 0, 20, 0, 40);
  println(x);
}

void draw() {

}
```

Console Errors

The map() function allows you to map a value from one range to another.

10 sits 50% of the way between 0 and 20

Map will ‘map’ 10 to a new range.

50% of the way between 0 and 40 is: 20

So in this case x is 20.

Map_1_C | Processing 3.2.1

Java

Map_1_C | Map_1_B

```
void setup() {
  size(594, 841);
  background(255);

  float x = map(10, 0, 20, 0, 40);
  println(x);
}

void draw() {
  ellipse(width/2, height/2, 500, 500);
}
```

Done saving.

Console | Errors

Draw an ellipse.

Map_1_D | Processing 3.2.1

Map_1_D

```
void setup() {
  size(594, 841);
  background(255);

  float x = map(10, 0, 20, 0, 40);
  println(x);
}

void draw() {
  float myValue = map(mouseX, 0, width, 0, 255);

  ellipse(width/2, height/2, 500, 500);
}
```

Console Errors

If we map mouseX from it's normal range: 0 - width
To a new range: 0 - 255

Then as we move the mouse from left to right:

mouseX increases from 0 to 500
and
myValue will increase from 0 to 255

Map_1_D | Processing 3.2.1

Map_1_D

```
void setup() {
  size(594, 841);
  background(255);

  float x = map(10, 0, 20, 0, 40);
  println(x);
}

void draw() {
  float myValue = map(mouseX, 0, width, 0, 255);

  fill(myValue);
  ellipse(width/2, height/2, 500, 500);
}
```

```
20.0
```

If we then fill with myValue, then the ellipse with smoothly transition from black to white as the mouse moves from left to right.

Map_1_D | Processing 3.2.1

Java

Map_1_D

```
void setup() {
  size(594, 841);
  background(255);

  float x = map(10, 0, 20, 0, 40);
  println(x);
}

void draw() {
  float myValue = map(mouseX, 0, width, 0, 255);
  println(mouseX, myValue);
  fill(myValue);
  ellipse(width/2, height/2, 500, 500);
}
```

```
112.045456 261
112.045456 261
112.045456 261
```

Console Errors