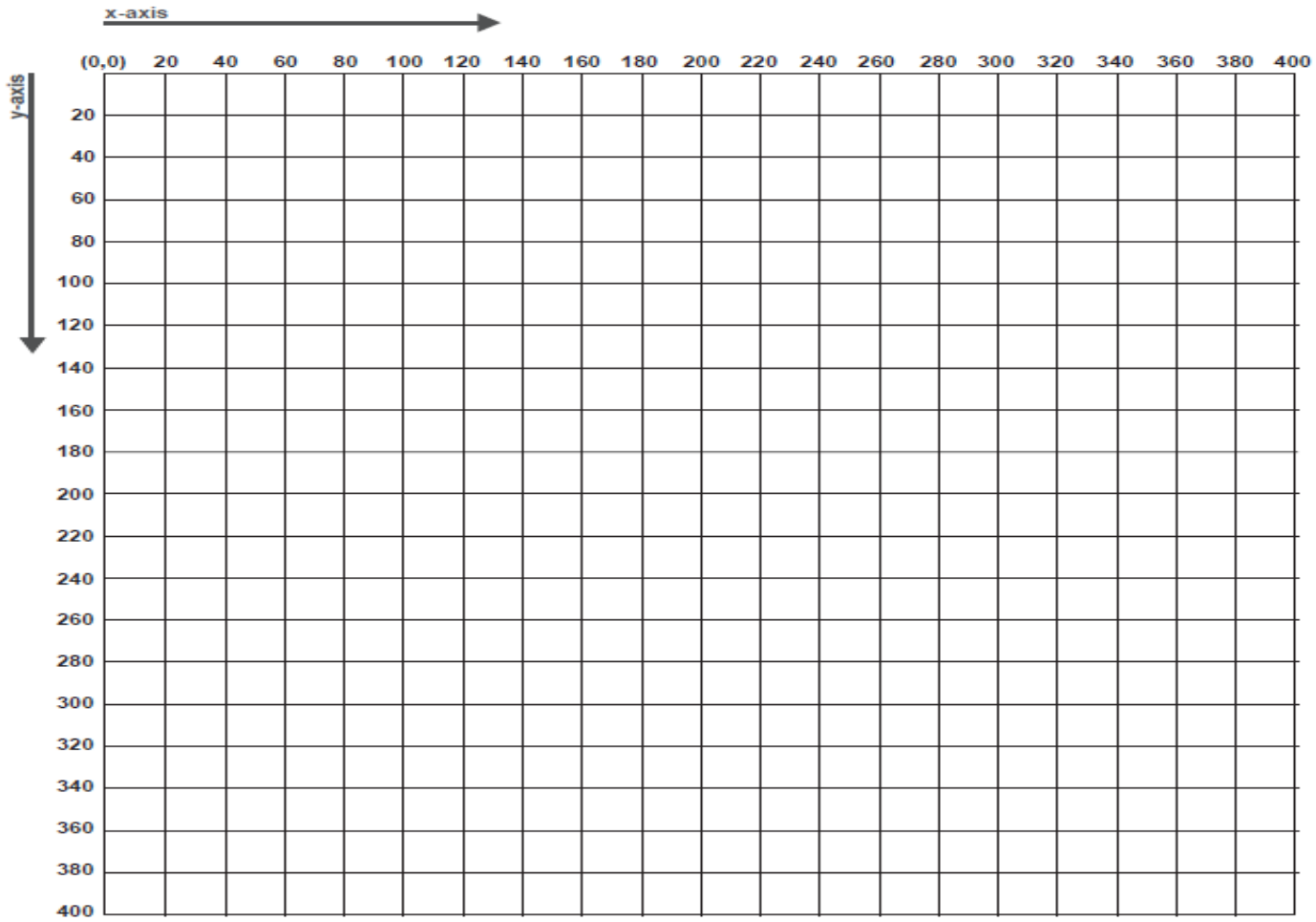
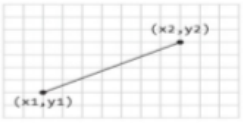
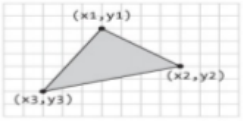
Patti Elfers-Wygand, Saranii Muller, Jerusha Theobald	
Unit of Study: P5 Unit One	**Lesson 2 - Creating With Shapes in the P5 Editor**
Topic: Creating With Shapes	**CSDFS: Algorithms and Programming:** **7-8.CT.10 Document the iterative design process of developing a computational artifact that incorporates user feedback and preferences.** **CCLS: RST 6-8:4 - Determine the meaning of symbols, key terms, and other domain specific words and phrases as they are used in a specific scientific or technical context relevant to grades 6-8 texts and topics.** **Blueprint for the Arts: Digital Media**
Skill: Plotting shapes in the P5 editor	**Academic Vocabulary:** **Function** **parameter** **argument** **canvas** **JavaScript** **Pixels** **Debug**
Warm Up: Think/Write/Pair/Share: **Warm Up:** What do you think JavaScript can do to help your web pages be more expressive in combination with HTML/CSS? - first jot your answer down on paper or in a doc, and then discuss with your partner. Once you have come to a consensus, then answer on the pear deck here in the space provided.	
Connection: (Review with Class) - We have learned how the P5 graph looks and how to plot shapes on the graph. Today we will try to plot some of those shapes on the P5 editor.	
Mini Lesson: How can we break down a drawing into basic shapes in P5.js? We will be looking at a graph that helps us pre-plan our sketches in P5 to learn the layout of the canvas/background. Unlike what you are used to in Scratch where the center coordinates for x and y are: 0, 0 are located in the upper left hand corner of the canvas on P5. We will practice on the handouts first to see how to identify points for lines and then plot a rectangle and an ellipse and finally will try in the P5 editor. Included in the lesson is also a helpful handout to show you how to draw basic shapes with their necessary parameters. We will do a **code along** for these shapes to help you transfer your shapes to the P5 program.	

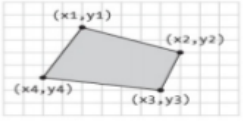




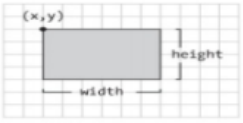
line(x1, y1, x2, y2)



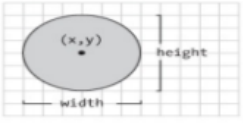
triangle(x1, y1, x2, y2, x3, y3)



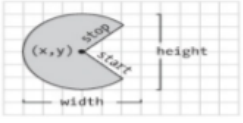
quad(x1, y1, x2, y2, x3, y3, x4, y4)



rect(x, y, width, height)



ellipse(x, y, width, height)

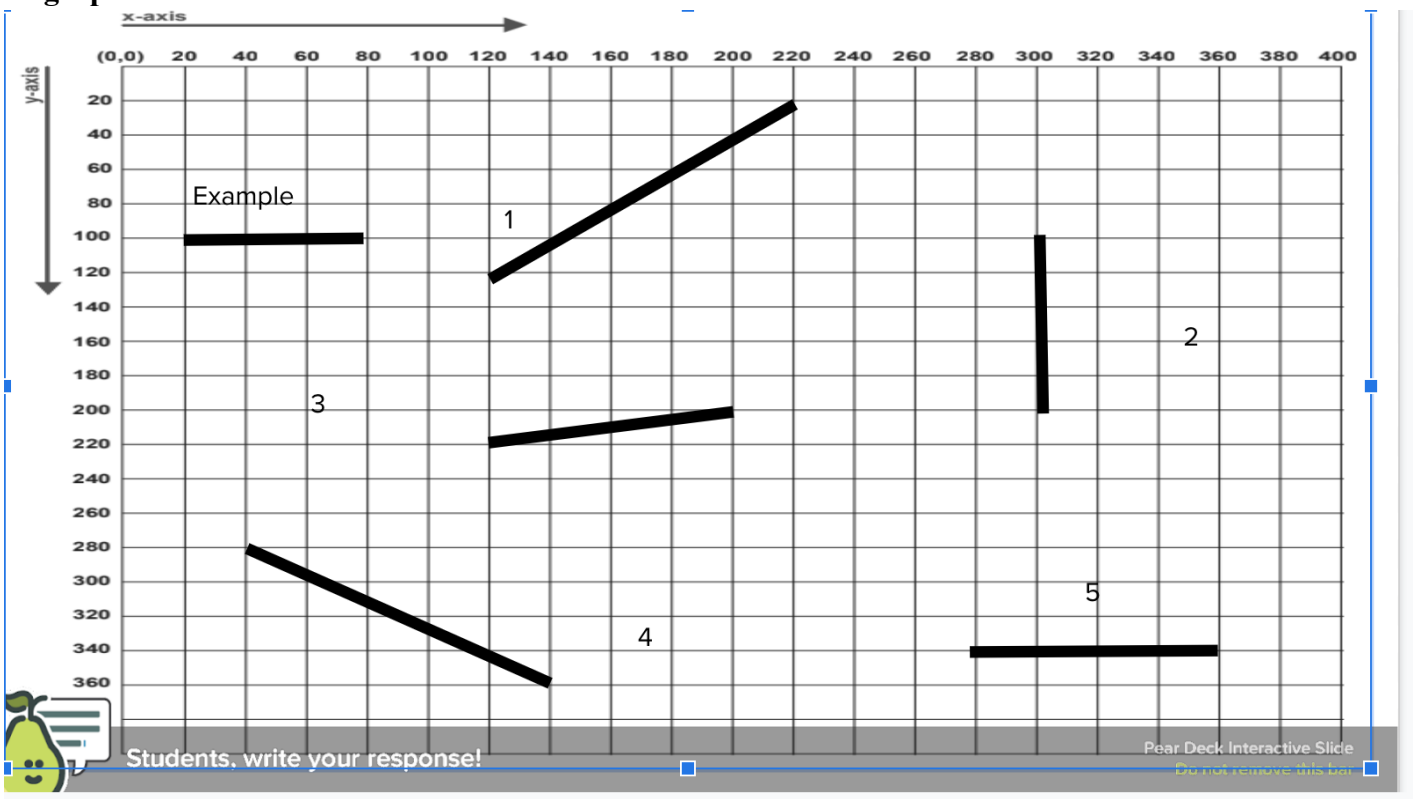


arc(x, y, width, height, start, stop)

Figure 3-1. Shapes and their coordinates

Quick Check: What do you think the center points: x and y are in a canvas that measures 400 px by 400 px? - Answer on the pear deck on the slide presentation.

Work period:
Task One: Students will indicate the coordinates on the handout to indicate their understanding of the P5 graph.



They will also receive code for a rectangle and ellipse on the slide deck in order for them to practice plotting the points and then trying to draw their shapes in the P5 Editor.

Use the code below and the p5 graph to draw several rectangles and ellipses labeling their coordinates. Once you think you have them where you want them, then you may open up the p5 program, sign in with Google and create a sketch that you have drawn on your graph. Use the guidelines on the reference sheet to create ellipses and rectangles.

Rectangle:
rect(x, y, w, h)
Parameters

X - Number: x-coordinate of the rectangle

Y - Number: y-coordinate of the rectangle

W - Number: Width of the rectangle

H - Number: height of the rectangle

Ellipse:

Ellipse (x, y, w, h)

Parameters

X - Number: x-coordinate of the ellipse

Y - Number: y-coordinate of the ellipse

W - Number: Width of the ellipse

H - Number: height of the ellipse

Notes on how the p5 program runs:

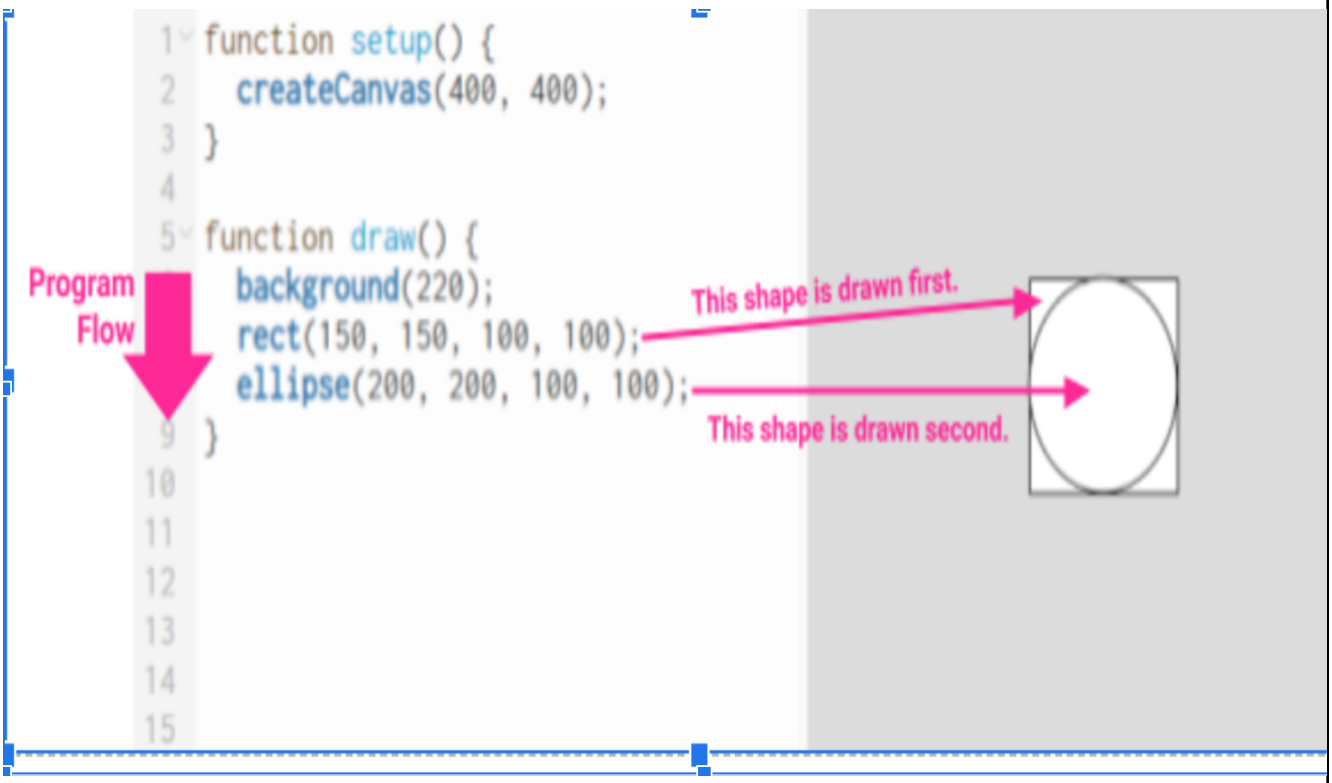
the `setup()` and `draw()` functions:

- These functions are special because p5 calls them automatically when a sketch is run. They don't have parameters, so the parentheses are empty.
- We use the `setup()` function to *set up* a sketch. In order to draw anything in p5, we need to make a canvas and give it a size (like 400 x 400) by calling `createCanvas()` **inside** `setup()`. When you hit play, anything that is inside the `setup()` function will run **one time**.
- In this unit, all of our shape functions will be called inside `draw()`.
- The code inside the `draw()` function actually runs in a **loop**. Every function we use inside draw is being called over and over again until the program is stopped. The loop happens so fast that the preview looks like one image, but in reality, the shapes are constantly being drawn on top of each other. Note: This loop will become relevant when students add color to their shapes, and later when they learn to animate shapes.

```
function setup() {
  createCanvas(400, 400);
}

function draw() {
  background(220);
  rect(150, 150, 100, 100);
  ellipse(200, 200, 100, 100);
}
```

- ellipse is on top of the rectangle, and it's because the program draws the rectangle first then ellipse. Everything in the `draw()` loop (and the p5 sketch as a whole) will run from top to bottom.



In this image, you will notice that the rectangle is drawn first and the ellipse is drawn on top of the rectangle. What would happen if we drew the ellipse first?

Assessments/Questions: In this image, you will notice that the rectangle is drawn first and the ellipse is drawn on top of the rectangle. What would happen if we drew the ellipse first? How does the P5 program run?

Share/Discuss: What shapes were you able to make?
Closing/Exit Ticket: ○ Share one new thing that you learned. ○ What was challenging? Why? ○ What elements would you add to your drawing if you had more time?
Note on grouping: **Students are seated next to a partner with differing ability so the more experienced student can work with the less experienced student. ELL students have similar language partners for additional translation help (if available)**
Materials and Scaffolds used: P5 graph, Shapes Reference Sheet, Slide Deck for Lesson, P5 Editor **Coding Train: https://www.youtube.com/watch?v=yPWkPOfnGsw**

Additional details used for ELL's and SWD students

Modifications -English Language Learners	Modifications-Special Education/Support Group
• Working with partners • Using visuals/gesture • Total physical response • Rep of modeling • Vocabulary dictionary in the program	• Working with partners • Using visuals/gesture • Total physical response • One/one modeling when needed • Vocabulary dictionary in the program