Week 12 Lab Assignment Goals

- Use object-oriented programming and the model-view-controller pattern with Pygame
- Draw a scatter plot!

Step 0 : Create a GitHub repository

- Go to https://classroom.github.com/assignment-invitations/fd02f18509dba2126f3f12f124e8909c
- Accept the assignment invite and clone the repository onto your machine
- Open a Terminal window or command prompt and 'cd' to the cloned directory

Step 1 : Understand the Model-View-Controller pattern

- The MVC pattern separates applications into three parts:
    - The *model* contains functionality to load, store, and modify data
    - The *view* handles the user interface, and
    - The *controller* takes care of program control, often calling on the model and view to assist with data management and user interaction, respectively.
- In keeping with this pattern, your folder contains three files:
    - model.py, view.py, and controller.py
- We will modify model.py and view.py to build our own ScatterPlot widget. We will restrict our graph to data points with x > 0 and y > 0.

Step 2 : model.py

Our *model* will interact with comma-separated files such as data.csv

- **Edit the get_data() function** to read the CSV file and return a list of tuples.
    - Each tuple contains the comma-separated values of one line of the file.
- **get_data('data.csv') should return the following list:**

  ```
  [('10', '20', 'red'), ('5', '10', 'green'), ('30', '40', 'blue')]
  ```
- This file contains three data points in our scatter plot:
    - a red point at (10, 20)
    - a green point at (5, 10), and
    - a blue data point at (30, 40).

Step 3 : controller.py

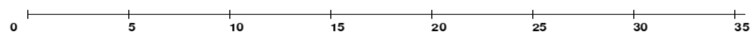
- Read and understand controller.py. You will not need to modify it, but it is important that you understand how the controller calls the *model* module to load data from the CSV file; and the *view* module to draw a scatter plot based on this data.

Step 4 : view.py

Our *view* will create and render a scatter plot based on the data contained in data.csv

- The class **Point** represents a single point on the plot
- The class **ScatterPlot** represents a scatter plot that is made up of a list of Points and meta data.

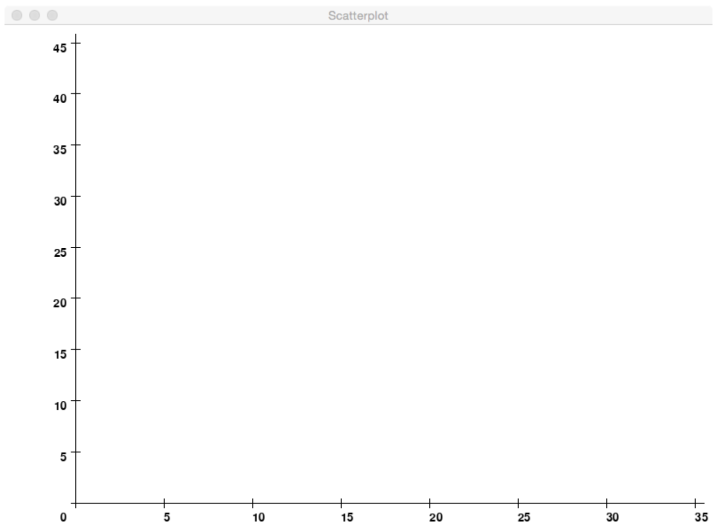
Run ‘python controller.py’ on the command line. You should see a horizontal X axis as follows:



Step 5 : Draw the Y-axis

- Drawing an axis is hard work! Note that there are two coordinate systems at play:
  - **Pygame’s coordinate system**, where (0, 0) is the top-left of the window.
  - **The scatter plot’s coordinate system**, where the origin is displayed towards the bottom-left of the screen.
- Look at the ScatterPlot.**draw_axes()** to see how we drew the X axis
  - How does draw_axes() transform between the two coordinate systems?
- **Write code to draw the Y axis**
  - First a vertical line that represents the Y axis
  - Then add ticks and labels to mark values along the Y axis

Your output should look like this:



Step 6 : Display data points

- Edit the Point**.draw()** method to draw the point object using its own X and Y coordinates, and the arguments supplied.
- **Ensure that all three points are displayed correctly with respect to the X and Y axes**.

Step 7 : Add more points to your plot!

- Play around with data.csv. Add more points.
- **Ensure that the scales, ticks and labels** on your X and Y axes **adjust** automatically according to the maximum X and Y values.

Step 8 : Commit code to GitHub

- Commit and push everything to GitHub

Bonus Step 9 : Handle negative values

- Modify ScatterPlot and Point to work with negative X and Y values.