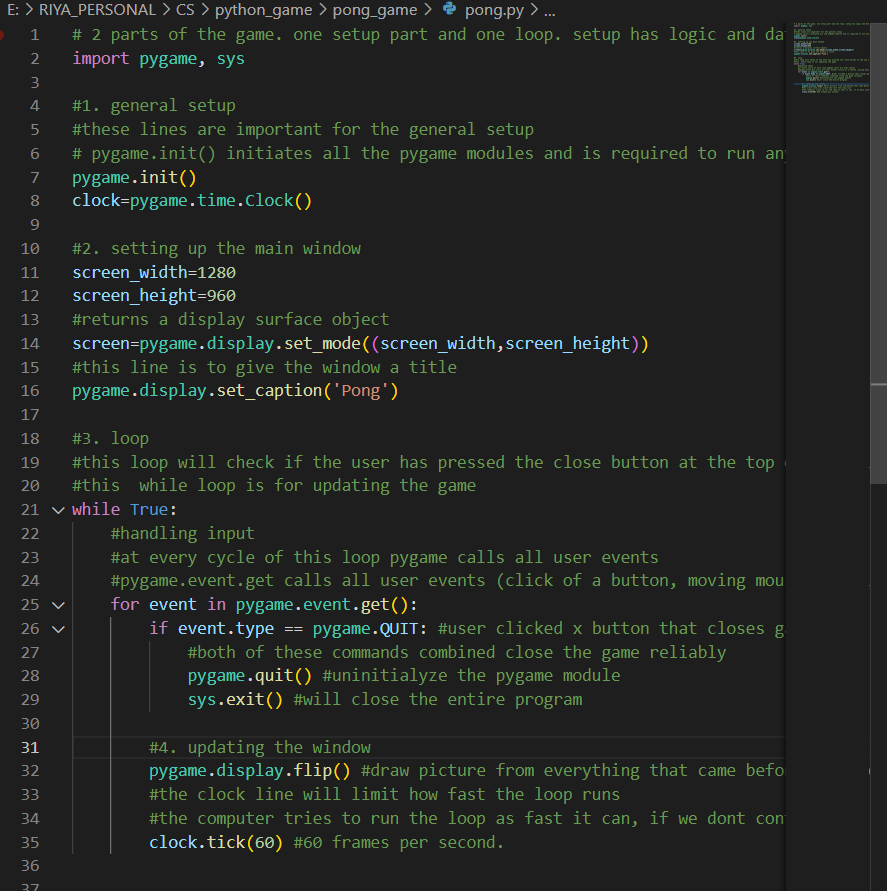
Pong game

1. Using the “pygame” library. It dosent come with python so first install it.
2. Pygame code has 2 parts one part has logic and the other part is the loop part that is needed to actually run the game.
3. Pygame has a lot of keywords to check user inputs/events called “locals”. They are always capitalized.
4. Creating a basic setup , a black screen:

This code will create a basic window for us where the window gets updated with 60 frames per second.



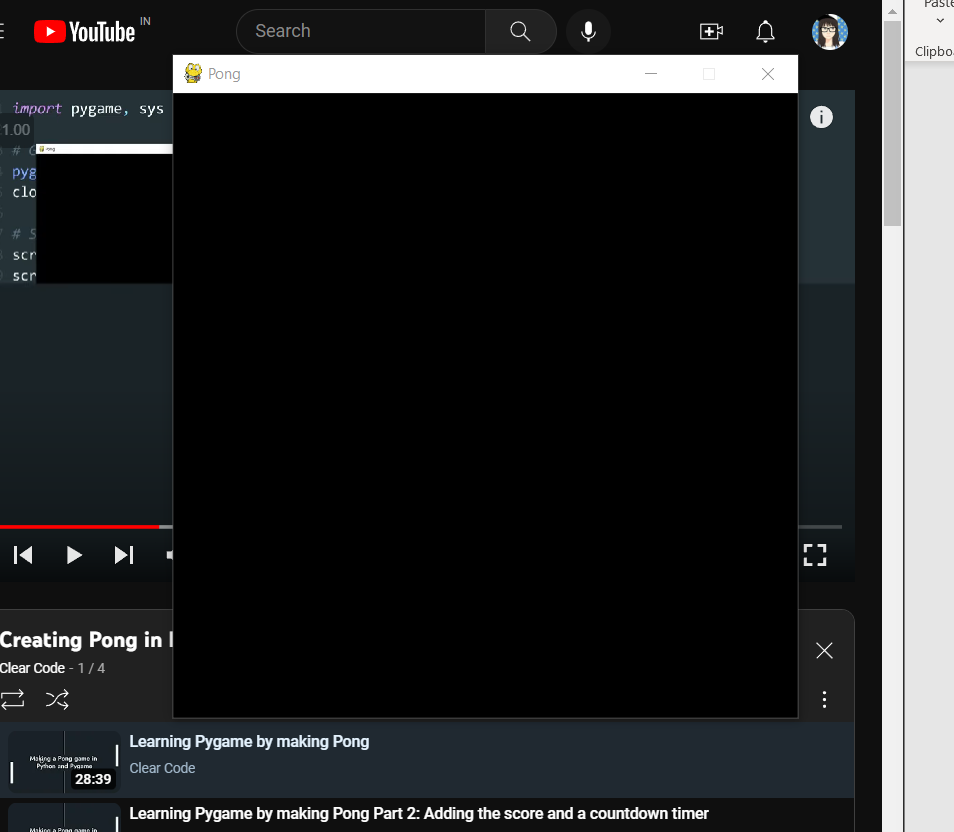


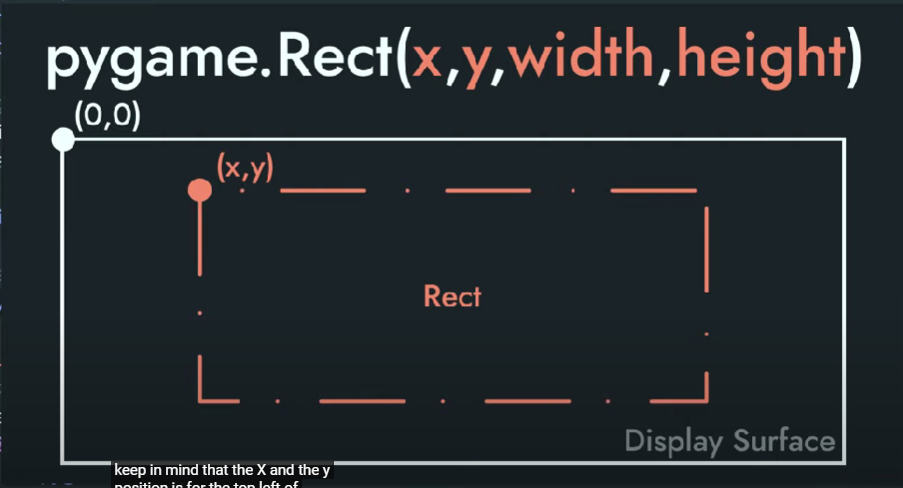
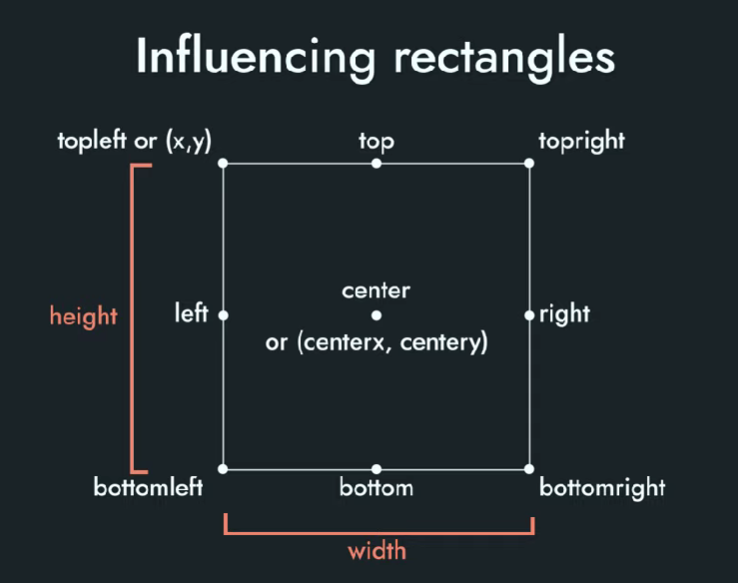
Figure 500x500 screen

Couldn’t fir the previous dimensions so changing the dimensions of the screen to be 640 x 480 pixels.

1. Drawing in pygame:

For this we gotta understand the hierarchy of 4 basic elements.

* Display surface: it is the most basic element. It is an object. Created it with the help of the function pygame.display.set\_mode() It is the main screen where we will draw all our shapes on. There can only be one Display Surface. All the shapes and images will only be visible if they are on the display surface.
* Pygame.draw: directly draws on the display surface. It is the module to draw all types of shapes. Its syntax is pygame.draw(surface,color,rect). These are the surface to draw on, the color and rectangle that we are drawing.
* Regular Surfaces: Extra Surface. It is a surface that sits on top of the display surface and has to be added explicitly. It is used to keep the game organized, we can add as many of them as we want on top of the display surface. We can put drawings or pictures on it, just like we can on the display surface.
* Rect: it’s a rectangle that can be made surrounding an element (shapes and regular surfaces). Helps accurately measure the dimensions of shapes and drawing.

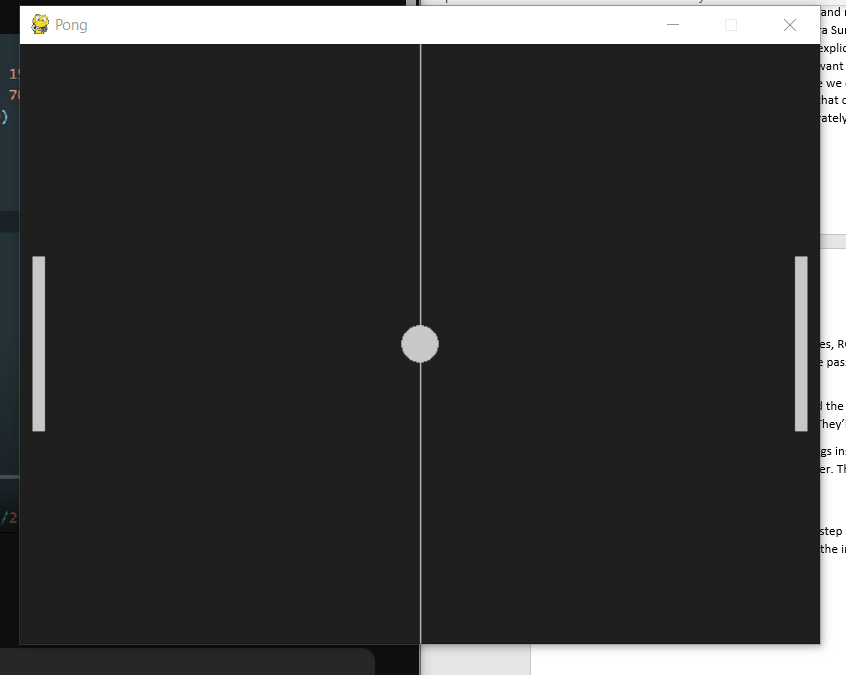
 

\*gotta increase y if u wanna go down

Colour can be set in 2 types, RGB or the colour object. Colour object can be used by pygame.color(‘name’). we pass the name of the color as a string an all the color strings can be found online.

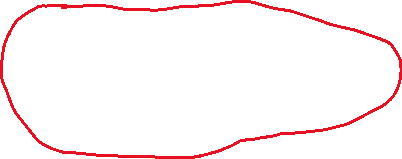
The background color and the line that separates the two sides of the game do not need a shape to be displayed. They’ll be drawn without a rect.

The order of drawing things inside the loop is very important. Successive things are drawn on top of each other. The first element called in the code will be at the bottom of the frame.



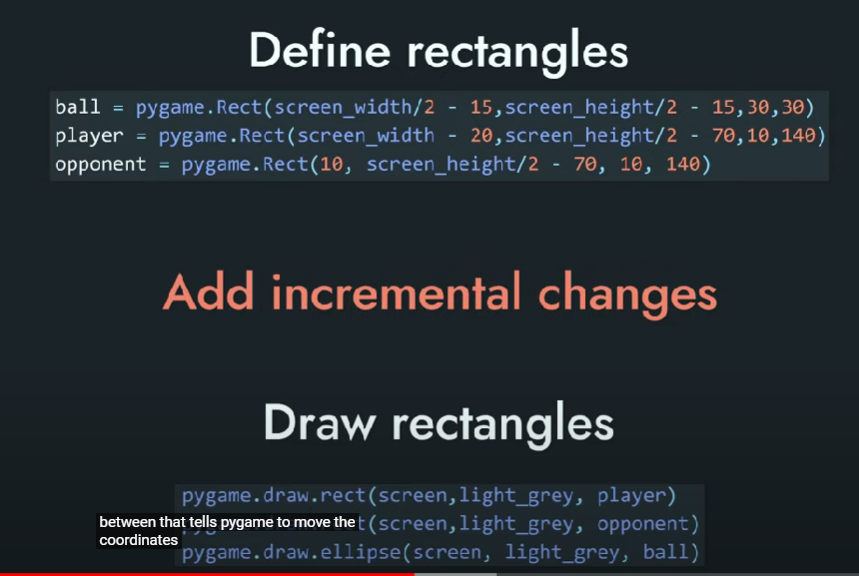
Text

Description automatically generated



1. Animation:

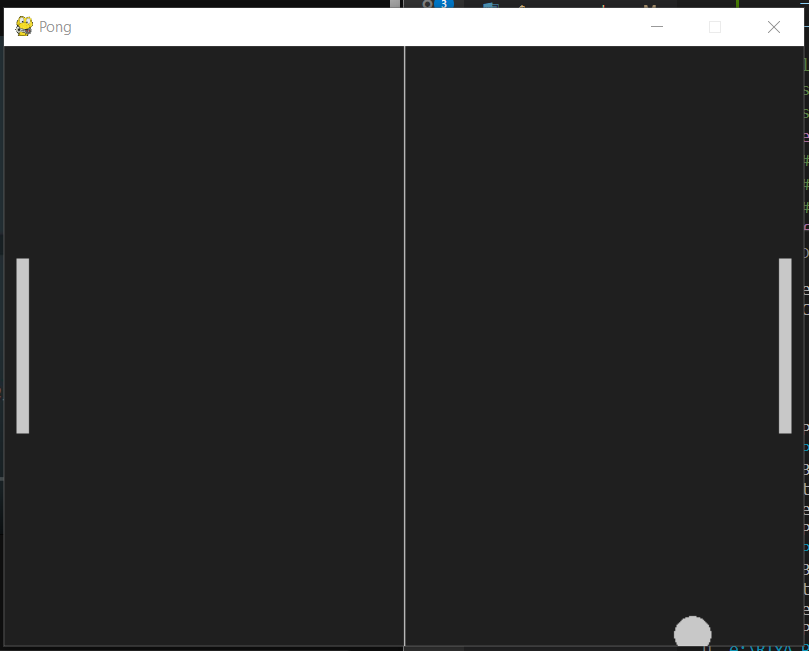
To make animation we add a step in between the defining a rectangle and drawing a rectangle step.



This step tells the increment in coordinate at every cycle of the loop.

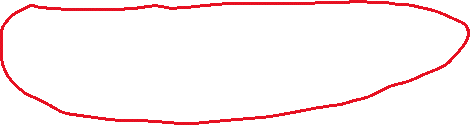
We can do x=x+5 y=y+5 this will move the x and y position by 5 in every loop.

Ball moving:



Text

Description automatically generated



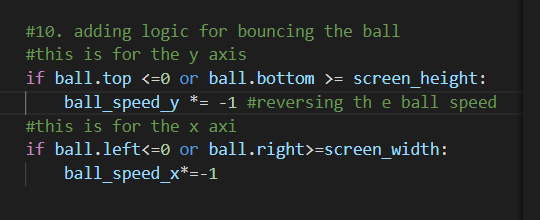
Right now, the ball will move but it will move out of the frame.

To stop the ball from moving out of the frame we need collisions.

Adding bounce when the ball hits any of the 4 sides of the display screen edges:

If the ball’s top is getting out of the screen display then we reverse the speed

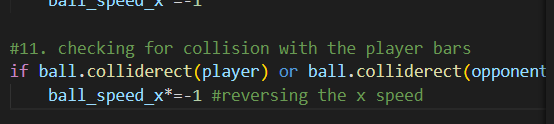
If the balls’ x axis gets out of the width of the speed then we are reversing the x speed of the ball.



<noticed an error here, had all the sentences below the for loop inside the for loop that is why the animation was not smooth. Specially, flip function needs to be called only once every while loop.

Checking if the 2 rectangles are colliding or not:

We also want the ball to collide with the 2 players. Player and opponent bars. To do this we can check if 2 rectangles are getting collided with each other. For this we will use the colliderect() function. Passing one rectangle and the other rectangle. If these 2 rectangles collide then it will return True and if there is no collision then it will return False.



1. Cleaning up the code:

It is better to keep all the animation logic and the loop separate. So, we’ll put all the ball animation code in a function.

Ball\_spped\_x and ball\_speed\_y needs to be global variables so we’ll do that by adding a global keyword before them and then declaring them inside the function.

1. Input:

We have one input command in the game already, it is that we can close the program by clocking at the x(quit) button.

Now we want to add input with which when we move the up and down arrows the bars will also move up and down.

Input for the bar requires us to check 2 things:

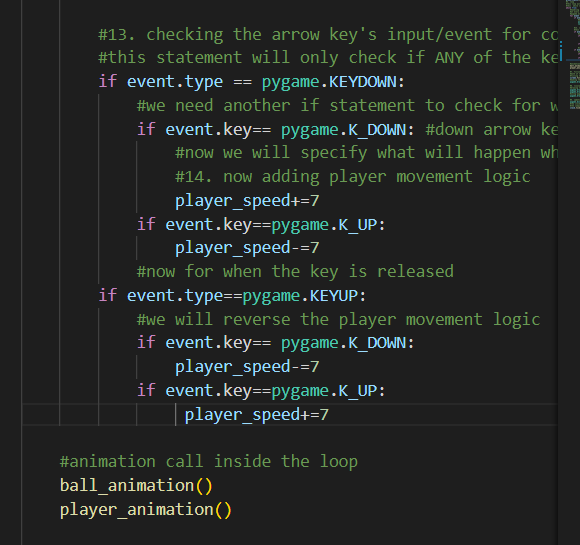
* The button was pressed, and the movement should start now.
* The button has been released and the bar should stop now.

So, we’ll be checking these events for key\_up and key\_down.

Pygame only checks if the button was changed from the stet of being not pushed to pushed so move an object we’d need to continusly press it for small movements. We don’t want this, so we’ll use another logic.

The logic is:

* Declare player speed variable.
* Add this speed to player on every frame.
* When no button is pressed then this speed is 0
* When button is pressed this speed becomes positive or negative.



We will also add logic to avoid the player from going outside the window and we will put all that logic into a function called player animation.