Demo Project – Flappy Bird

Now, we will do class pipes.

CODE:

```
__init__.py entities    colors.py    pipe.py    scoreboard.py    bird.py    ground.py ×
entities > ground.py > Ground > display
from entities.sprite import Sprite
import pathlib
import os
from util.constants import Constants
import pygame


class Ground(Sprite):
    def __init__(self):
        self.width = Constants.ground_width
        self.height = Constants.ground_height

        current_path = pathlib.Path(__file__).parent.parent.absolute()
        path =  os.path.join(current_path, "assets", "images", "ground.png")

        self.platforms_required = 2 * (Constants.width // self.width) + 2
        self.images = [pygame.image.load(path) for _ in range(self.platforms_required)]
        self.positions = list()
        self.draw_platform()

    def draw_platform(self):
        dx = 0
        for _ in range(self.platforms_required):
            x = self.width/2 + dx
            y = Constants.height - self.height / 2
            self.positions.append((x, y))
            dx += self.width
```

We can optimize this code as we having lot of image being displayed. We will load the image once and then will use the rectangle of that image into an array and will run the image according to the positions that we have already set.

Now we can set the animation part of the bird. We have a bird.png file with size of 276 X 64. So, for each bird, the width would be 276/3. In Constants.py file in util folder, we can set the width and height of the bird image as 92 and 64 respectively.

We are not using the jump function, so we can remove it. Now let’s google how to crop an image in python.

We will use the cropped.blit(image, (position), (cropped dimensions)).

```
screen.blit(self.image, (100, 100), (0, 0, Constants.bird_width,
Constants.bird_height)
```

This will show the first bird with its wings down.

```
screen.blit(self.image, (100, 100), (Constants.bird_width, 0,
Constants.bird_width, Constants.bird_height)
```

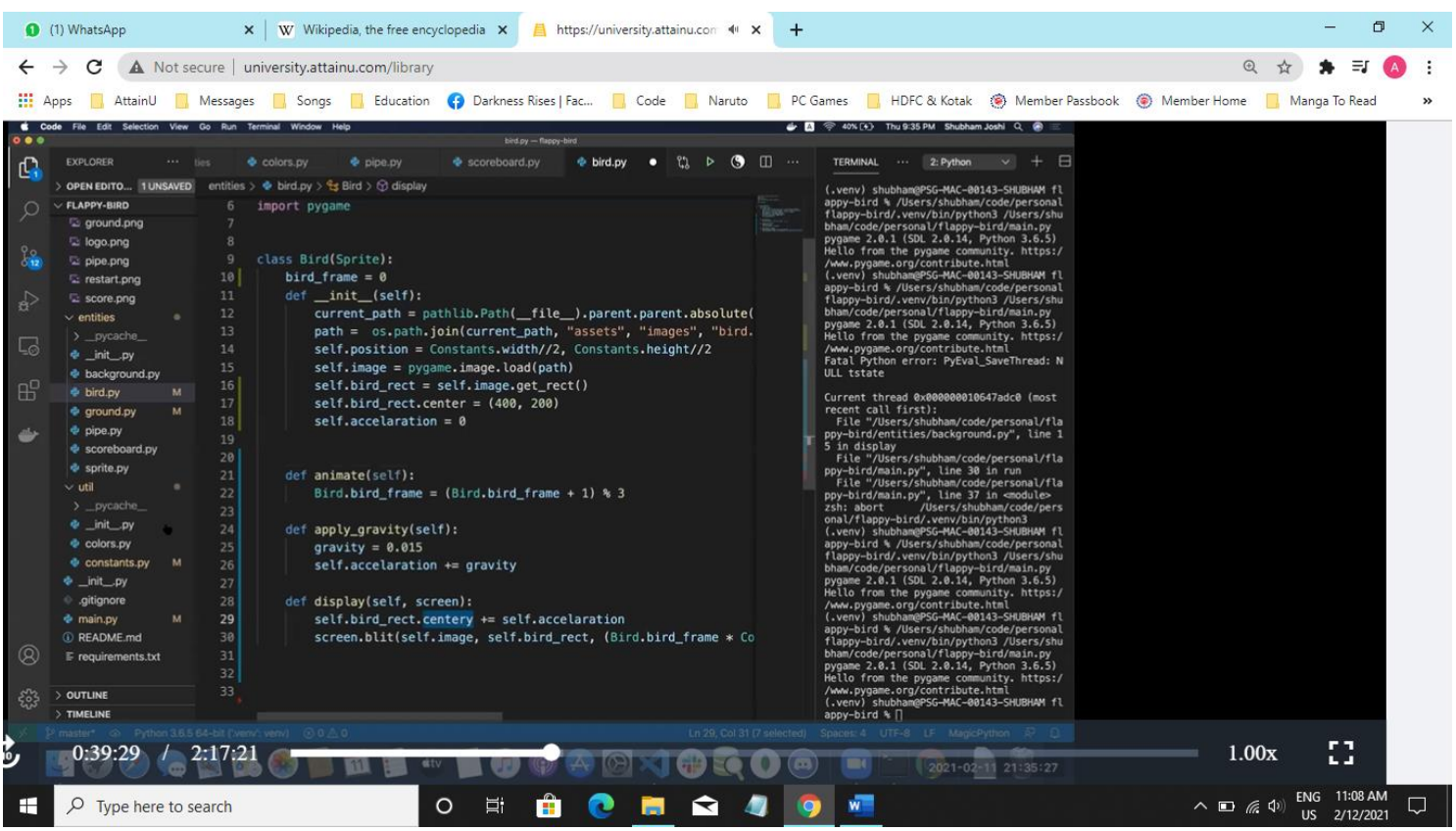
This will show the second bird in the image with its wings horizontal

```
screen.blit(self.image, (100, 100), (2 * Constants.bird_width, 0,
Constants.bird_width, Constants.bird_height)
```

This will show the third bird in the image with its wings up

In the bird.py, we have the display function. We can change the code in this by adding the

```
def display(self, screen):
    self.bird_rect.centery += self.accelaration
    screen.blit(self.image, self.bird_rect, (Bird.bird_frame *
Constants.bird_width, 0, Constants.bird_width, Constants.bird_h
eight))
```

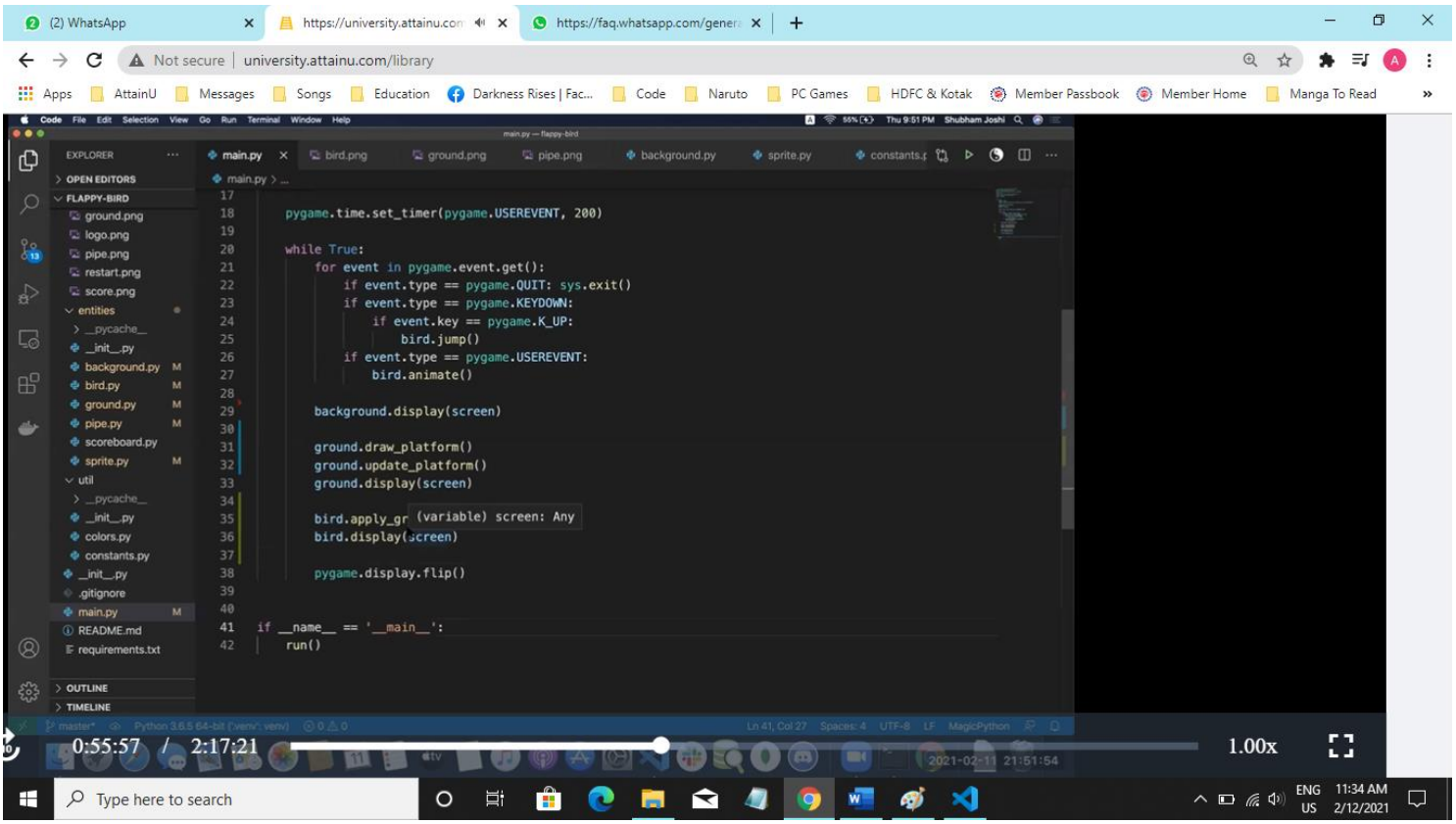


We have added the animate and apply_gravity functions in our code.

Even for pipes, we need a class pipe and a constructor as well. We can copy the can duplicate the code from bird.py file into the pipe.py file. Then make changes to the code.

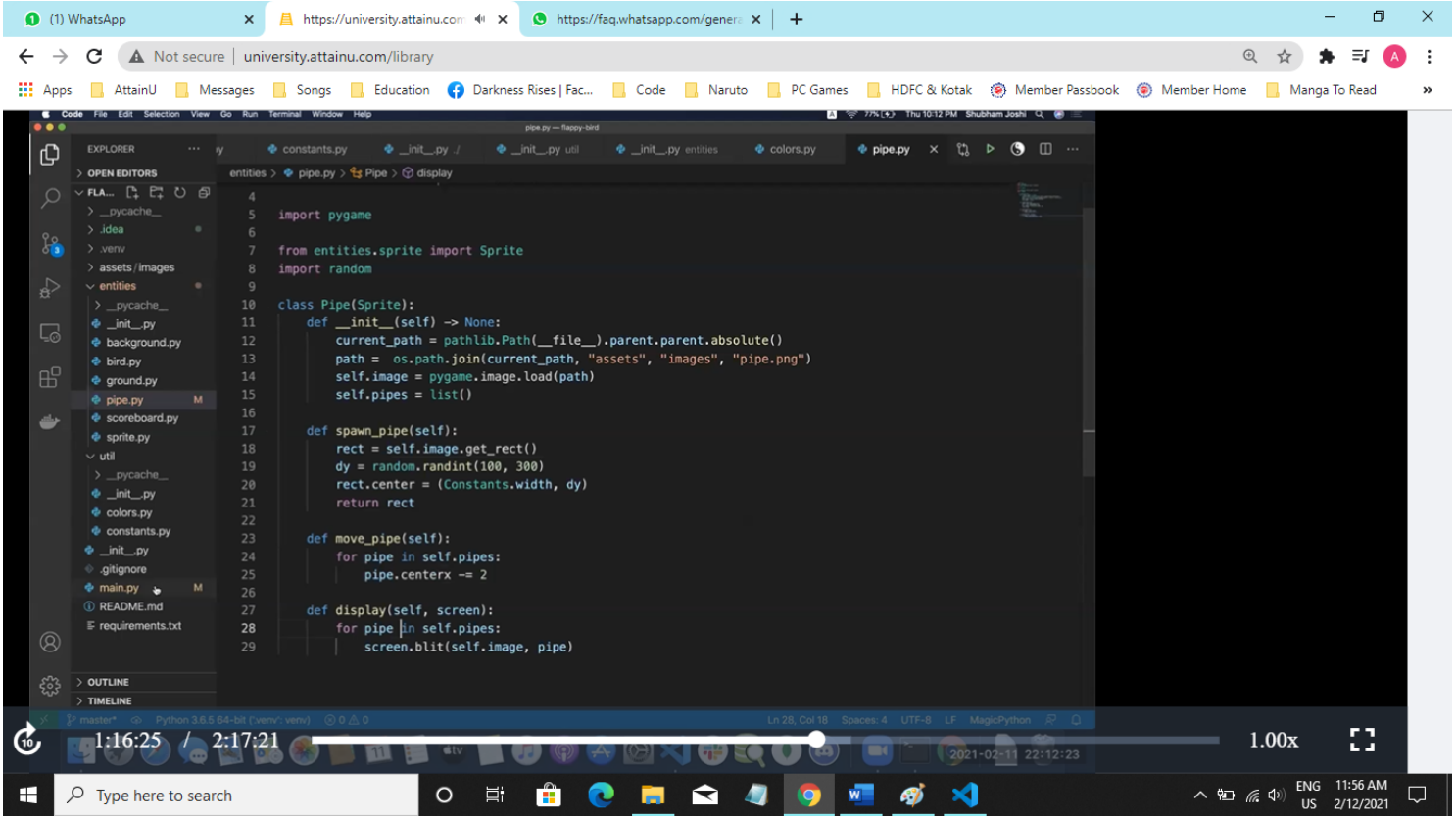
In the sprite.py file, all the common code in the python files, can be saved in sprite so that it will run the code and we will be able to run the code. Every sprite has a ‘rect’ also, so we can change it to it. The (x, y) for the bird would be (400, 200).

The work of display function is only to display. But in the ground.py, display is also used for updating the platform. We can move the update code into the main.py file.

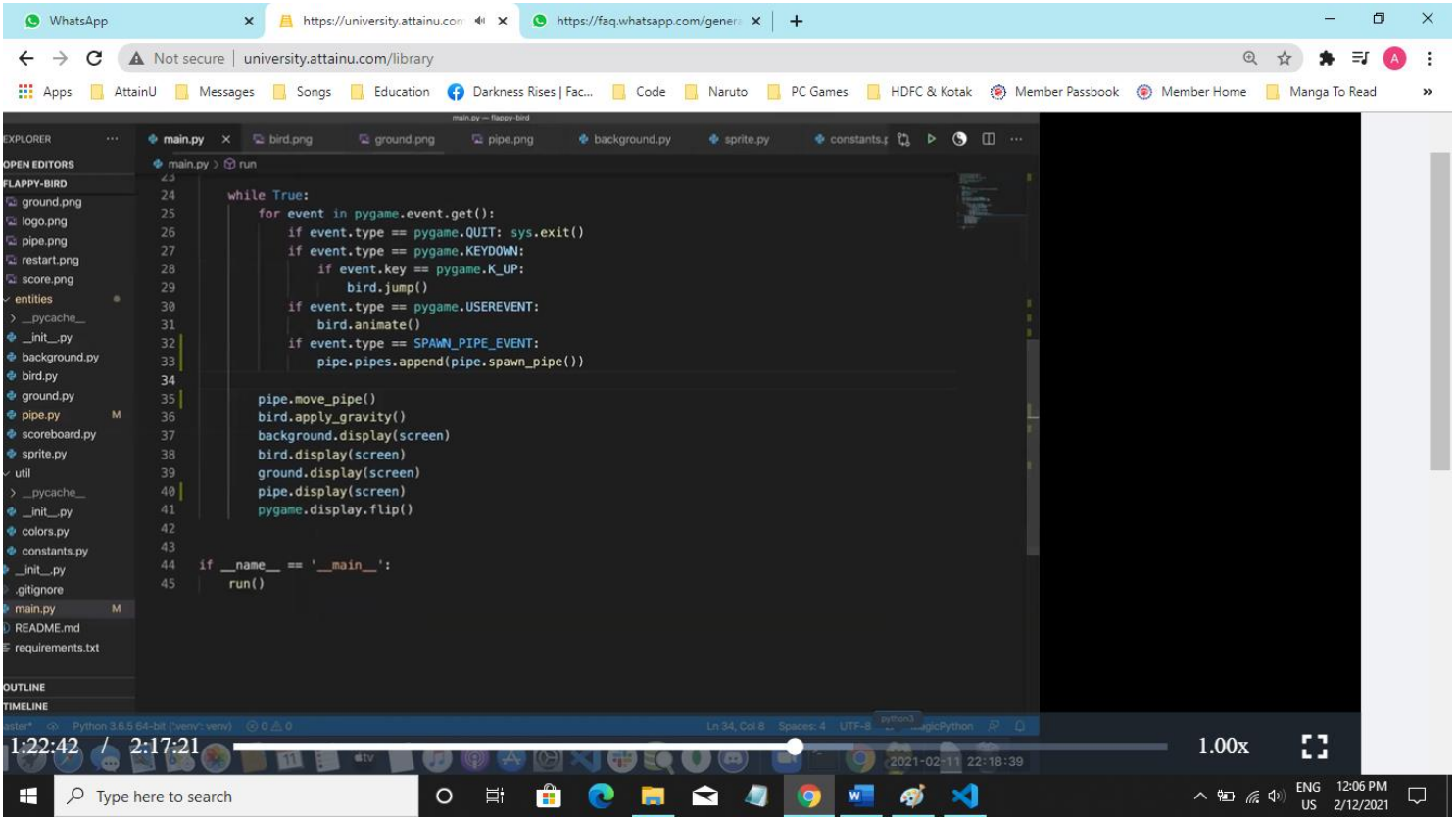


Back to pipe.py file, we need to generate the pipes at random places. In the main file, we need to create SPAWN_PIPE_EVENT = pygame.USEREVENT + 1. Using if condition, if event is true for SPAWN_PIPE_EVENT, then program will run **`spawn_pipe()`** function, it will run after every 1.5 second or 1500 milli second.

We will have multiple pipes, so it can be a list. So, in **`spawn_pipe()`** should return get_rect(). In the main file, under the event, we will append the pipes. Now we need to draw the pipe, so we will use the display function.

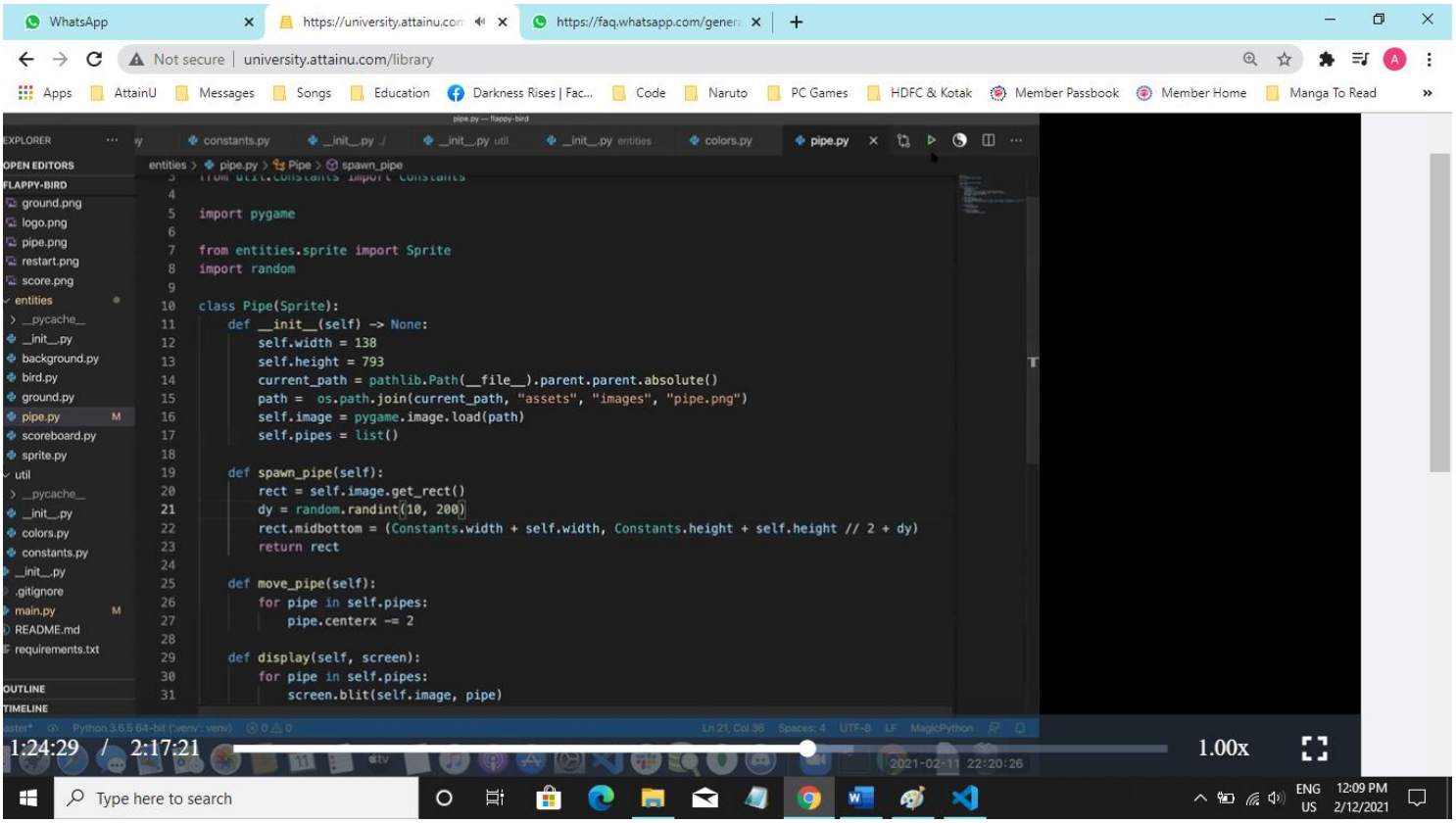


We will fix the position of pipe. The size of the pipe is 138 X 793. We can say, mid-bottom would be 793 / 2, which will put the pipe at the bottom of the page.

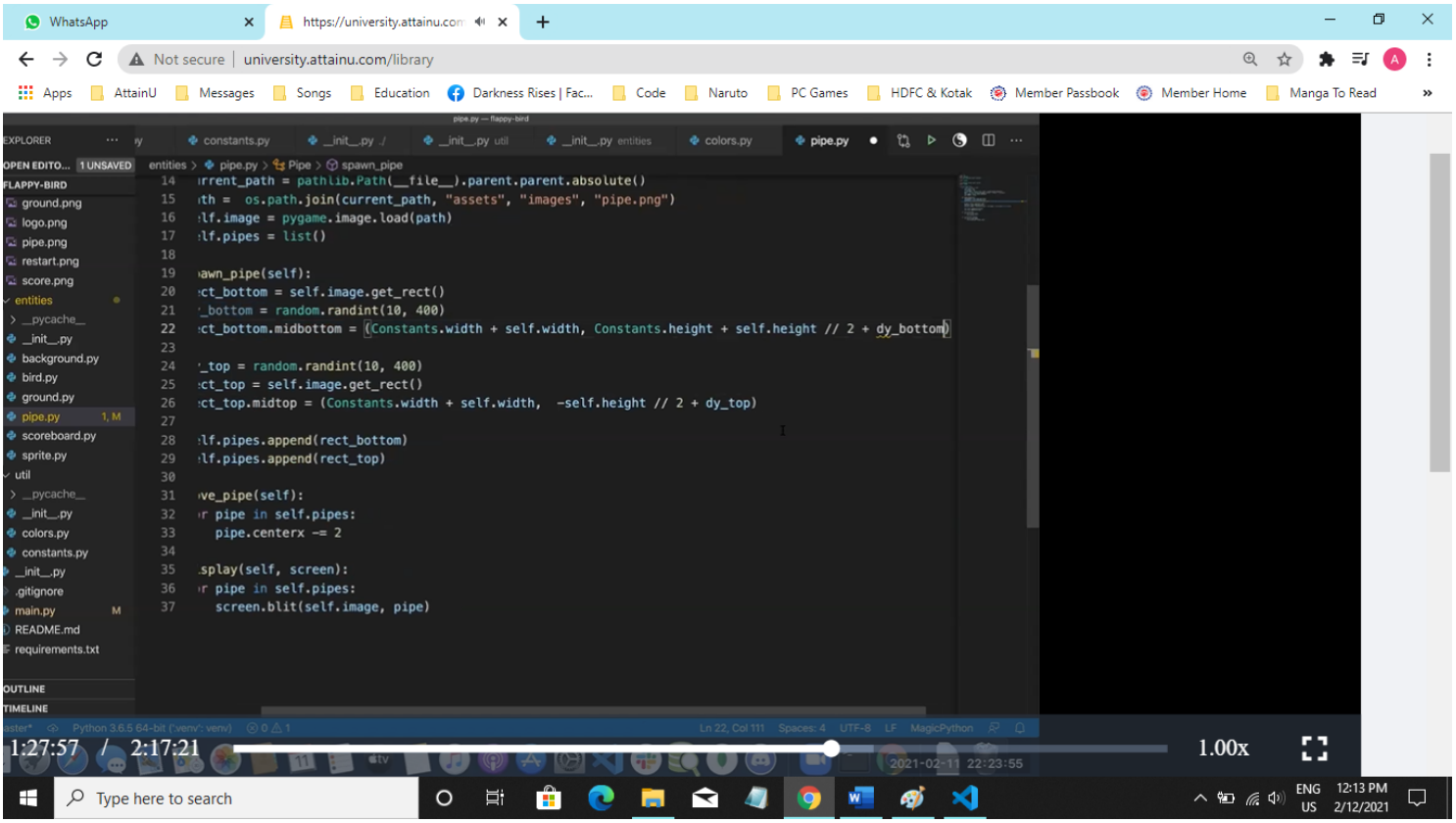


Pipe should be displayed behind the ground.png. So, we will put the ground function on the top and pipe function below it.

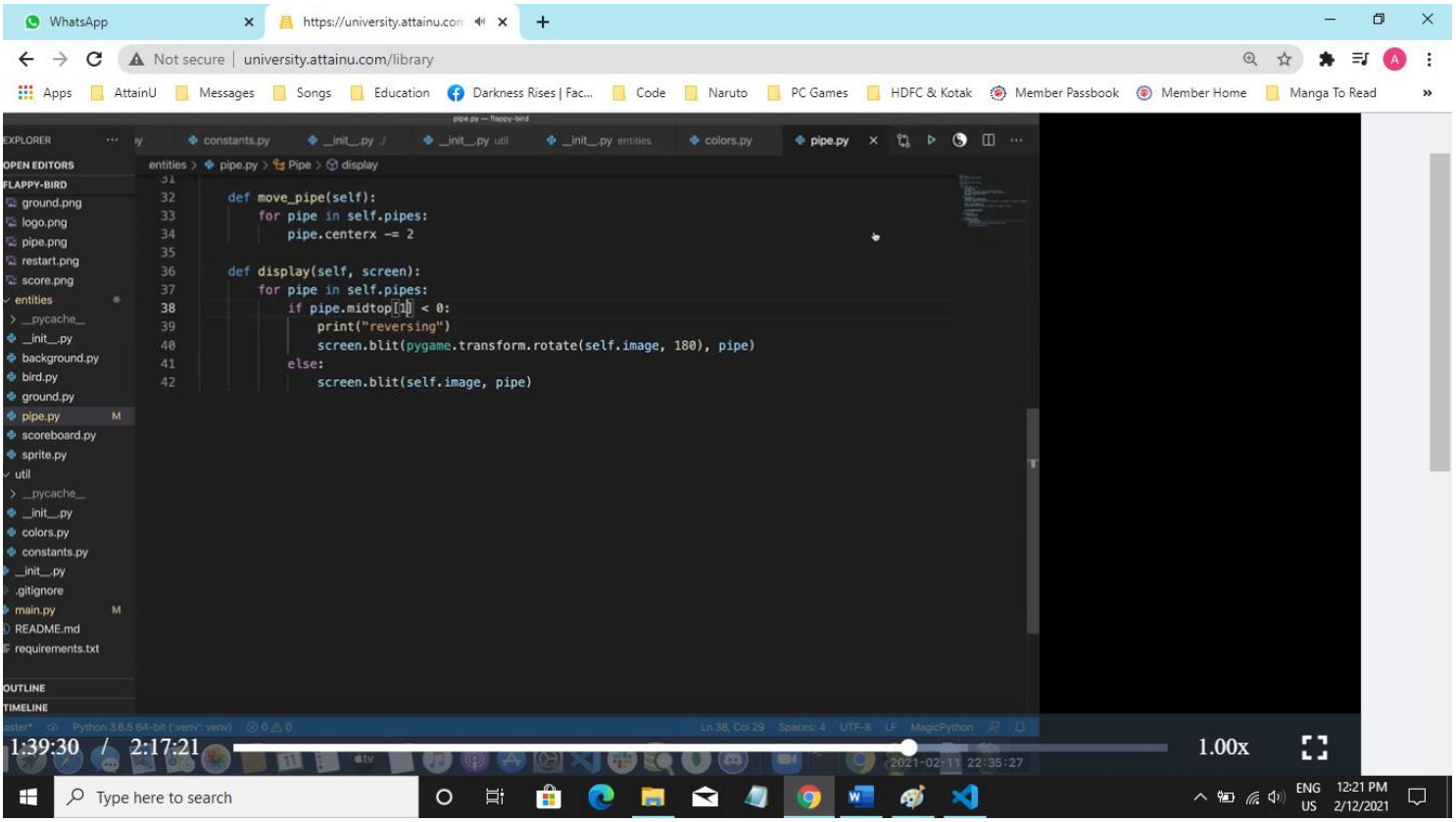
The height of the pipe will keep changing.



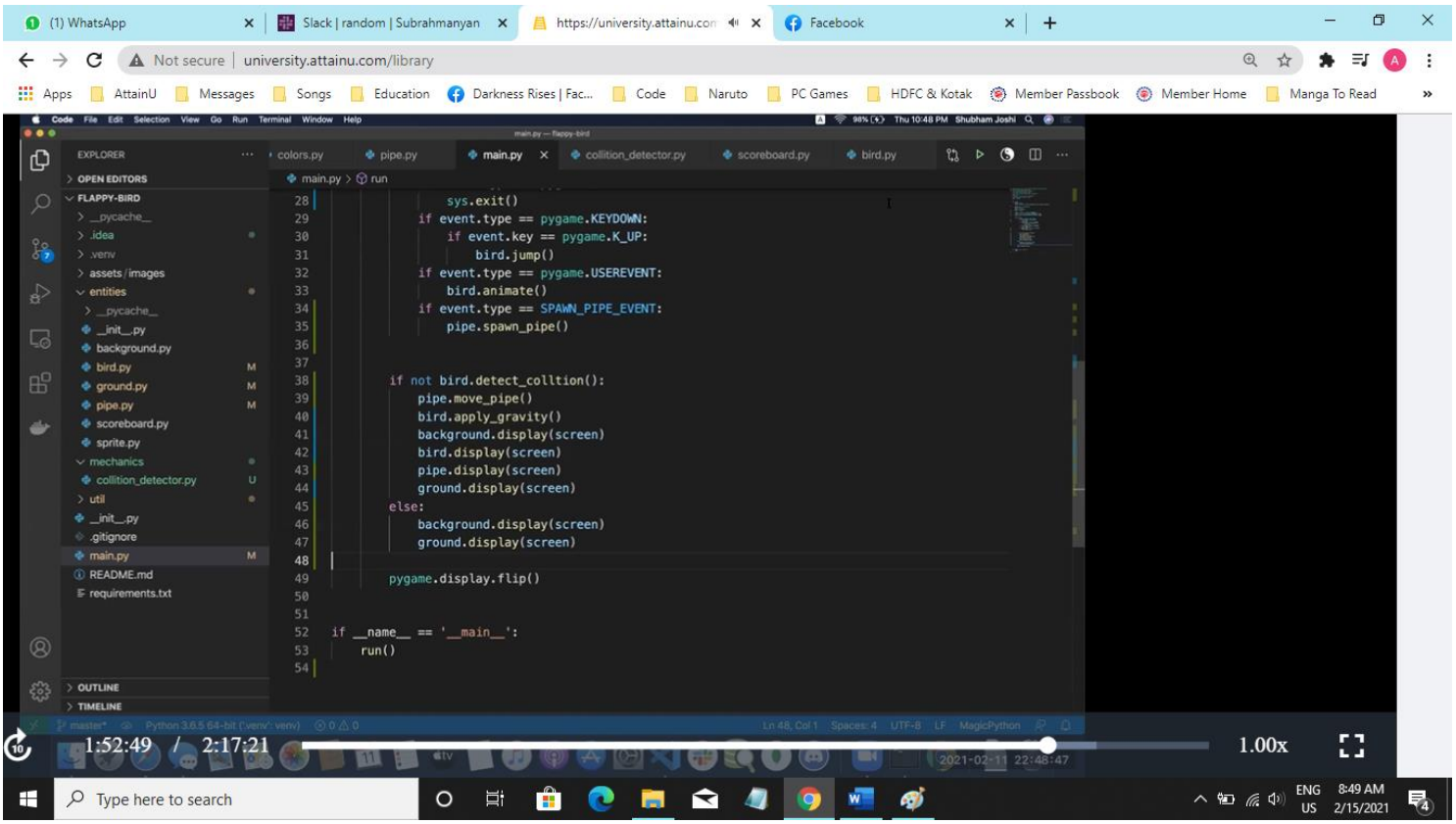
We need to add pipes at the top as well. We will



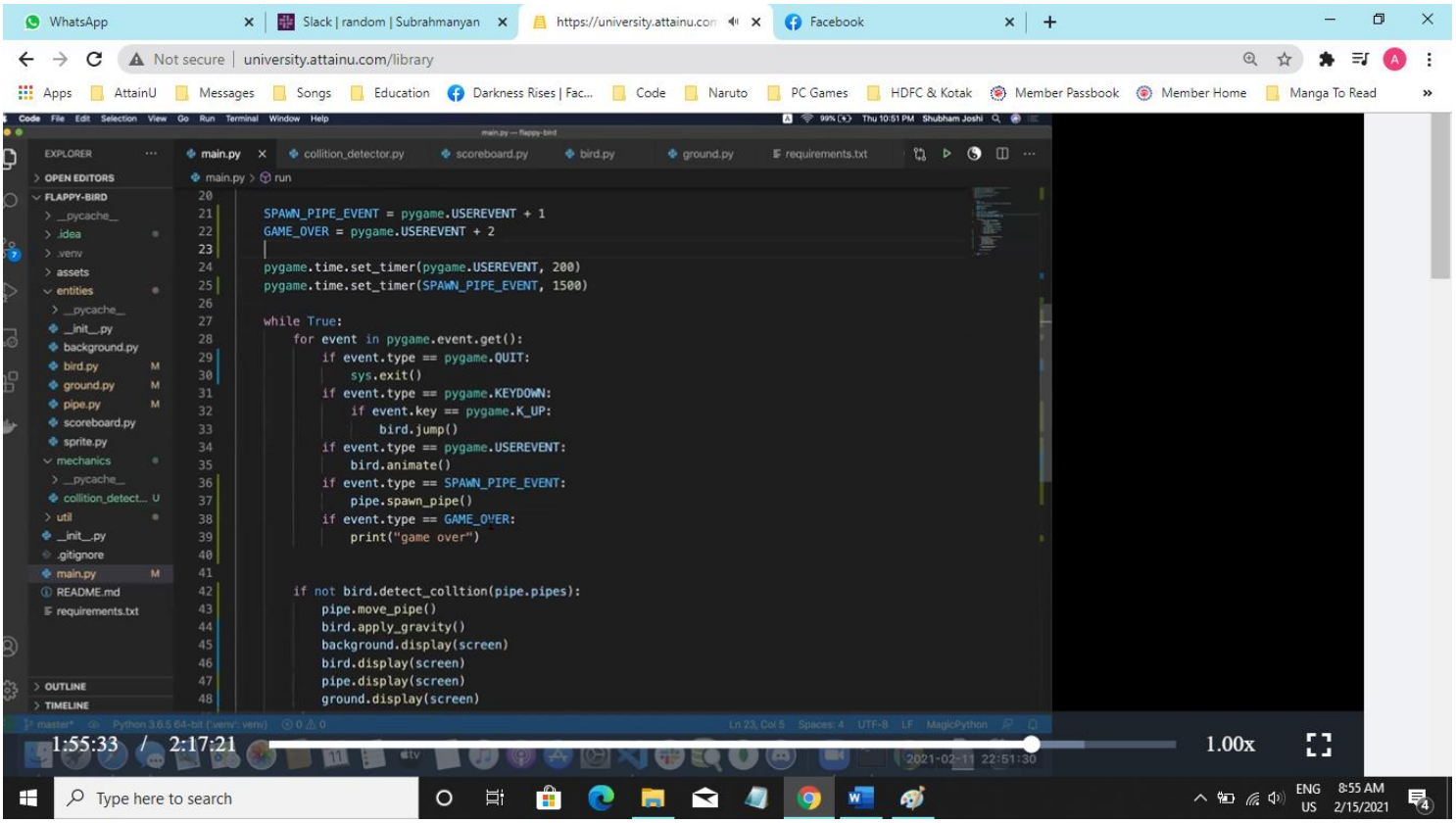
In the main we can call the function **`spwan_pipe()`.** We are getting the pipes, but not as expected.

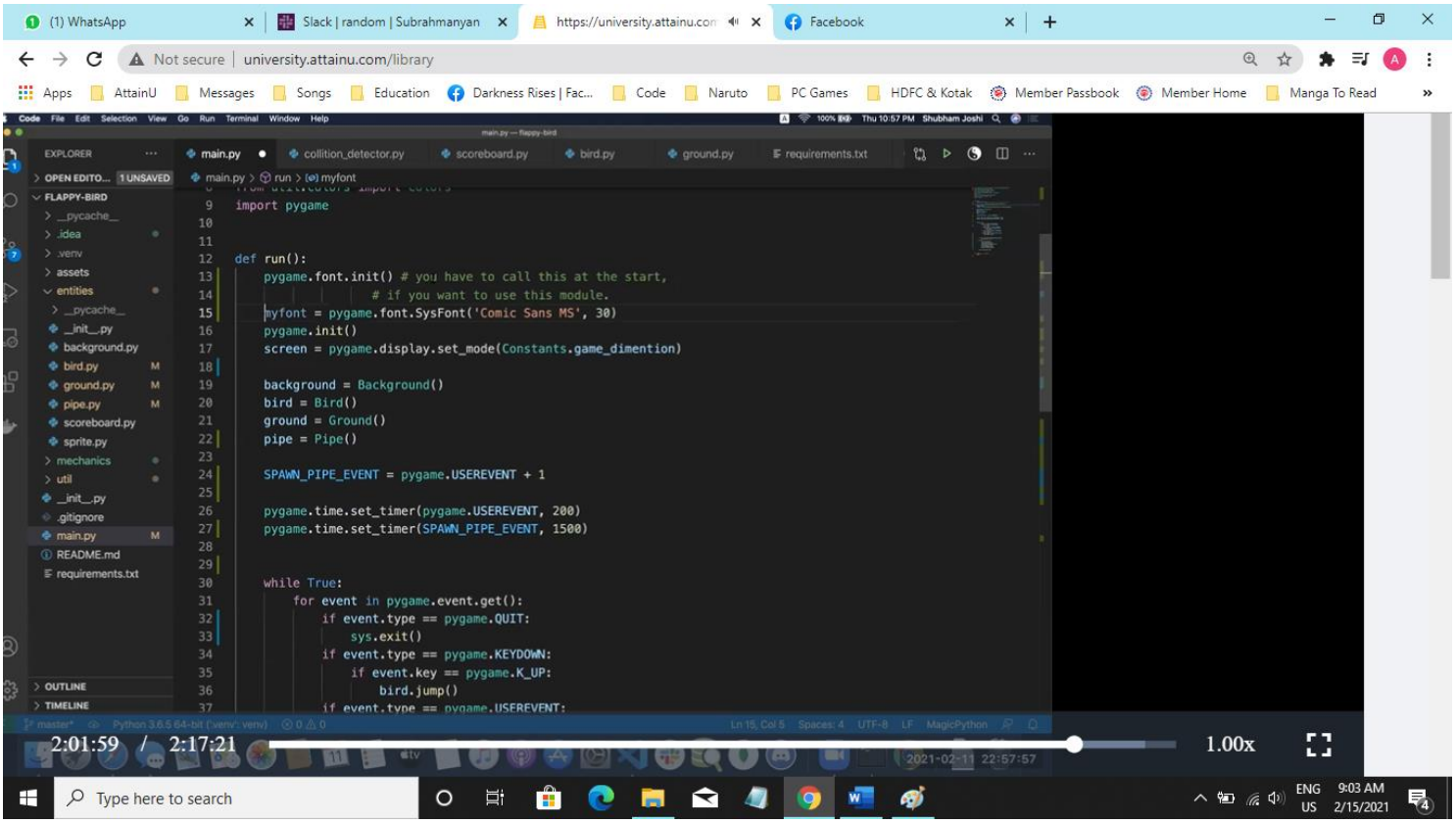


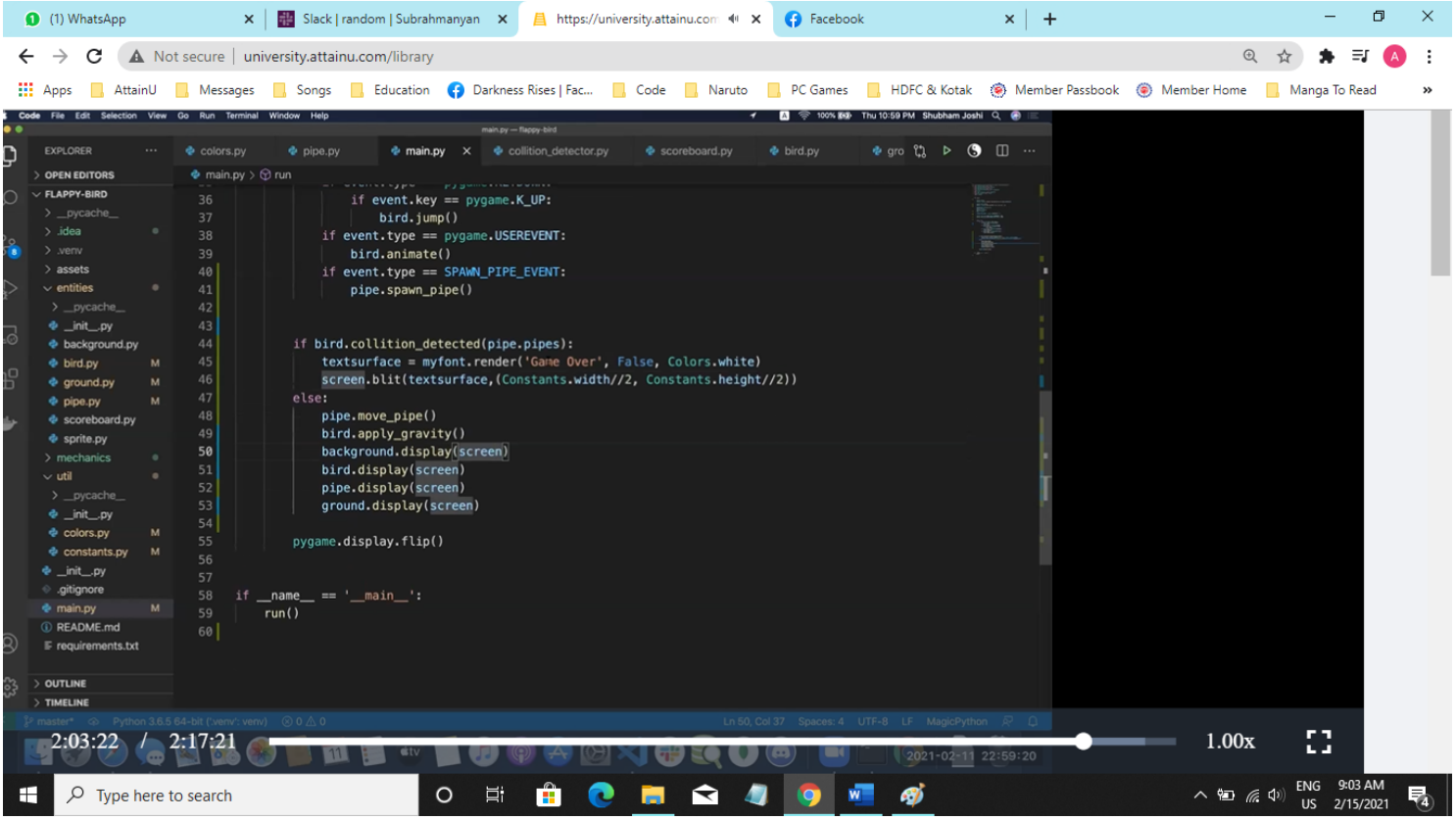
We can optimize this code. Now, we need to write a program to detect collision. We will create a file name collition_detector.py in a mechanics folder.



In this code, even before collision is done, it is stopping the game. Let’s change,







https://github.com/joshi95/flappy-bird