

PYTHON PROJECT

- PROJECT NAME : **Bouncing Ball Simulator**

- SUBMITTING TO :

1. Mr. Mayur Dev Sewak
General Manager , Operations
Eisystems Services
2. Ms. Mallika Srivastava
Trainer , Programming & Algorithms,
Eisystems Services

1. SUBMITTING BY:

ANUJ MISHRA

Content Table

Serial No.	Title	Page No.
1.	Cover page	1
2.	Content table	2
3.	List of figures	3
4.	Abstract of project	4
5.	Project summary	4
6.	Objectives of Project	5
7.	System Requirements used	5
8.	Text code/program	6-7
9.	Screenshots of program	8-9
10.	References	10

List of Figures

1. Screenshots shows the working of program:

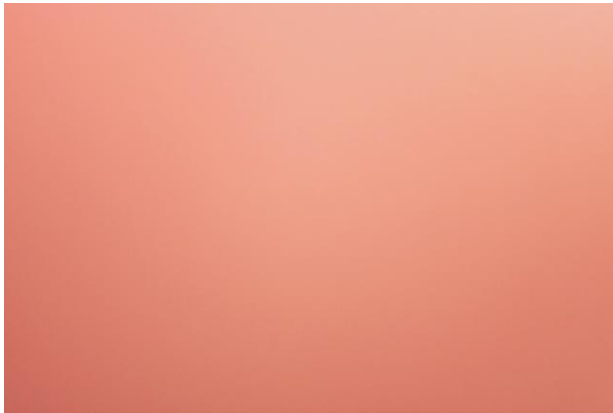
- Caption : Screenshot1, Screenshot2, Screenshot3, Screenshot4, Screenshot5
- Page No : 8-9

2. Images and Files were used in the Project

- Python File.py



- Ball.jpg



- Background.jpg

Abstract of Project

Project Title : Bouncing Ball Simulator

- It's most fun and most instructive to work on this simulation from *first principles* without using any "fancy" modules. The pygame module is a brilliant tool to create such physical simulations as it's relatively straightforward to use, and it gives you full control.
- Pygame is an open-source Python library for making multimedia applications like games built on top of the excellent SDL library.
- Using only relatively basic knowledge of the laws of motion and the effect of gravity on a falling ball, you've been able to create a reasonably realistic simulation of a bouncing ball in Python.
- Python is a multipurpose language and can be used in almost every field of development. Python can also be used to develop different type of game. Let's try to develop a simple Bouncing Ball Simulator.

Project Summary

Project Title: Bouncing Ball Simulator

- This is as simple as you can get for a bouncing animation. First, we see importing and initializing pygame is nothing noteworthy. The import pygame imports the package with all the available pygame modules. The call to pygame.init() initializes each of these modules.
- This simulation of Bouncing ball looks more attractive and with the help of pygame module another game can be created to play in endless loop.

Objectives Of Project

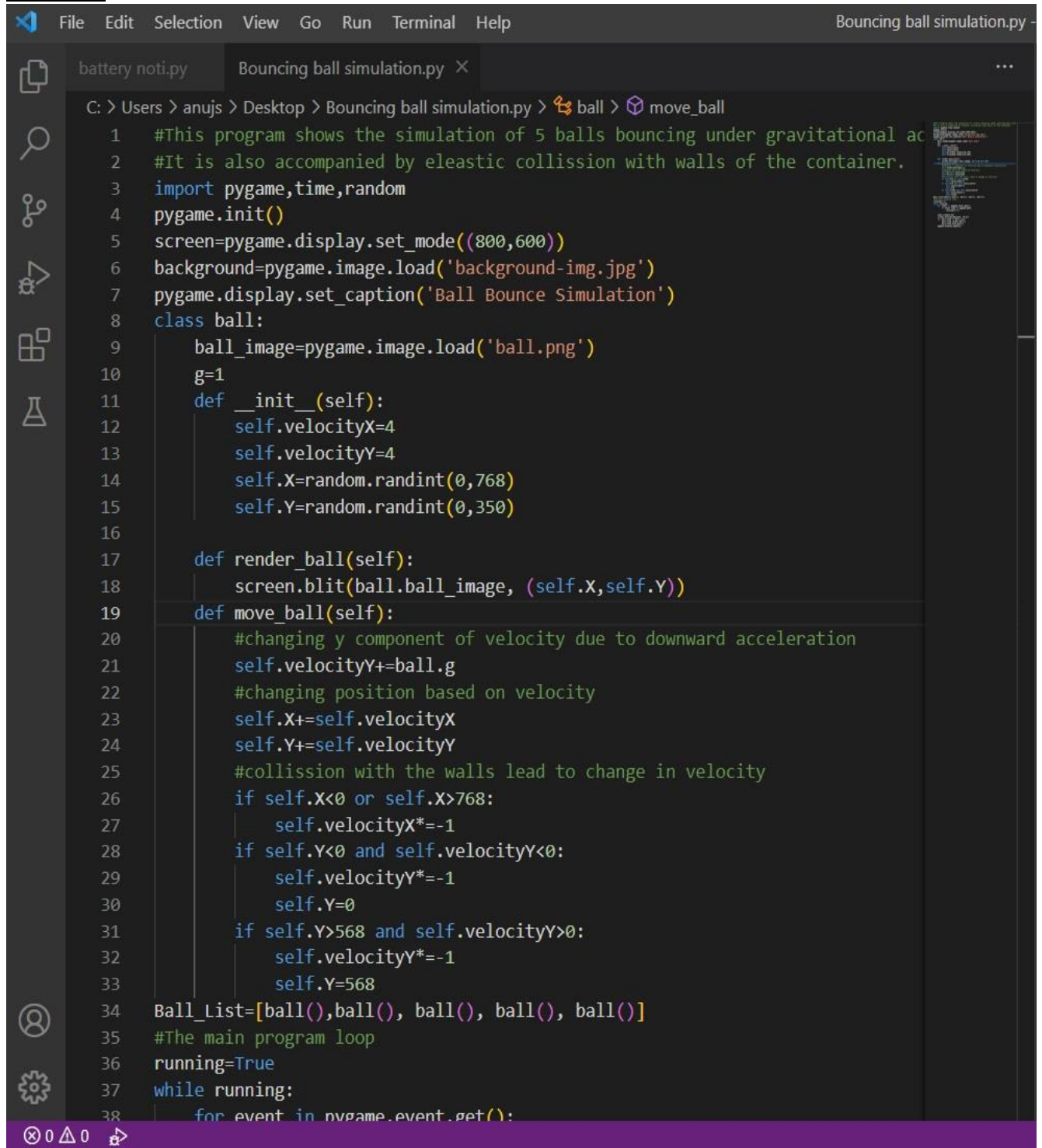
- To understand how to implement a bouncing algorithm, it provides you an essential knowledge to understand how the computer controls the trajectory of a sprite (e.g. ball) on the screen.
- The aim of the ball is to bounce randomly from one position to another.
- All five balls just move randomly from left to right and right to left and strike to the wall.

System Requirement Used

1. Windows 10 pro
2. Python 3.10.4
3. VS CODE
4. Command prompt for pip file installation

Text code/program

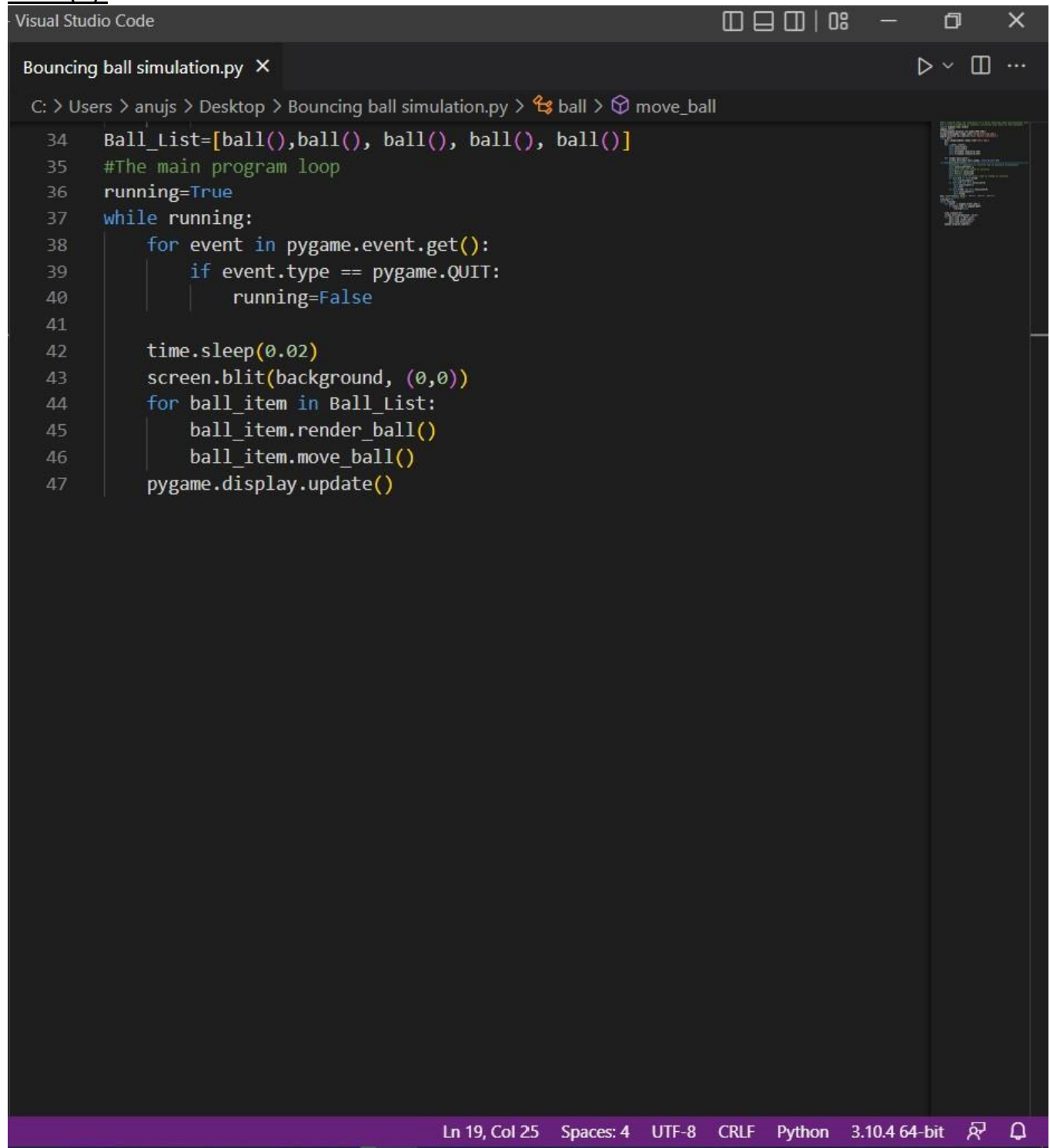
Code(a).



The screenshot shows a code editor with a dark theme. The menu bar at the top includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The title bar indicates the file is 'Bouncing ball simulation.py'. The editor has two tabs: 'battery noti.py' and 'Bouncing ball simulation.py'. The code is a Python program for a bouncing ball simulation. It imports pygame, time, and random. It initializes pygame and sets the screen mode to (800, 600). It loads a background image 'background-img.jpg' and sets the window caption to 'Ball Bounce Simulation'. A class 'ball' is defined with attributes 'ball_image' and 'g', and methods '__init__', 'render_ball', and 'move_ball'. The 'move_ball' method handles velocity changes and collisions with the walls. Five balls are created in a list, and a main loop runs until 'running' is False. The status bar at the bottom shows 0 errors, 0 warnings, and 0 suggestions.

```
C: > Users > anuj > Desktop > Bouncing ball simulation.py > ball > move_ball
1  #This program shows the simulation of 5 balls bouncing under gravitational ac
2  #It is also accompanied by eleastic collission with walls of the container.
3  import pygame,time,random
4  pygame.init()
5  screen=pygame.display.set_mode((800,600))
6  background=pygame.image.load('background-img.jpg')
7  pygame.display.set_caption('Ball Bounce Simulation')
8  class ball:
9      ball_image=pygame.image.load('ball.png')
10     g=1
11     def __init__(self):
12         self.velocityX=4
13         self.velocityY=4
14         self.X=random.randint(0,768)
15         self.Y=random.randint(0,350)
16
17     def render_ball(self):
18         screen.blit(ball.ball_image, (self.X,self.Y))
19     def move_ball(self):
20         #changing y component of velocity due to downward acceleration
21         self.velocityY+=ball.g
22         #changing position based on velocity
23         self.X+=self.velocityX
24         self.Y+=self.velocityY
25         #collission with the walls lead to change in velocity
26         if self.X<0 or self.X>768:
27             self.velocityX*=-1
28         if self.Y<0 and self.velocityY<0:
29             self.velocityY*=-1
30             self.Y=0
31         if self.Y>568 and self.velocityY>0:
32             self.velocityY*=-1
33             self.Y=568
34     Ball_List=[ball(),ball(), ball(), ball(), ball()]
35     #The main program loop
36     running=True
37     while running:
38         for event in pygame.event.get():
```

Code(b).

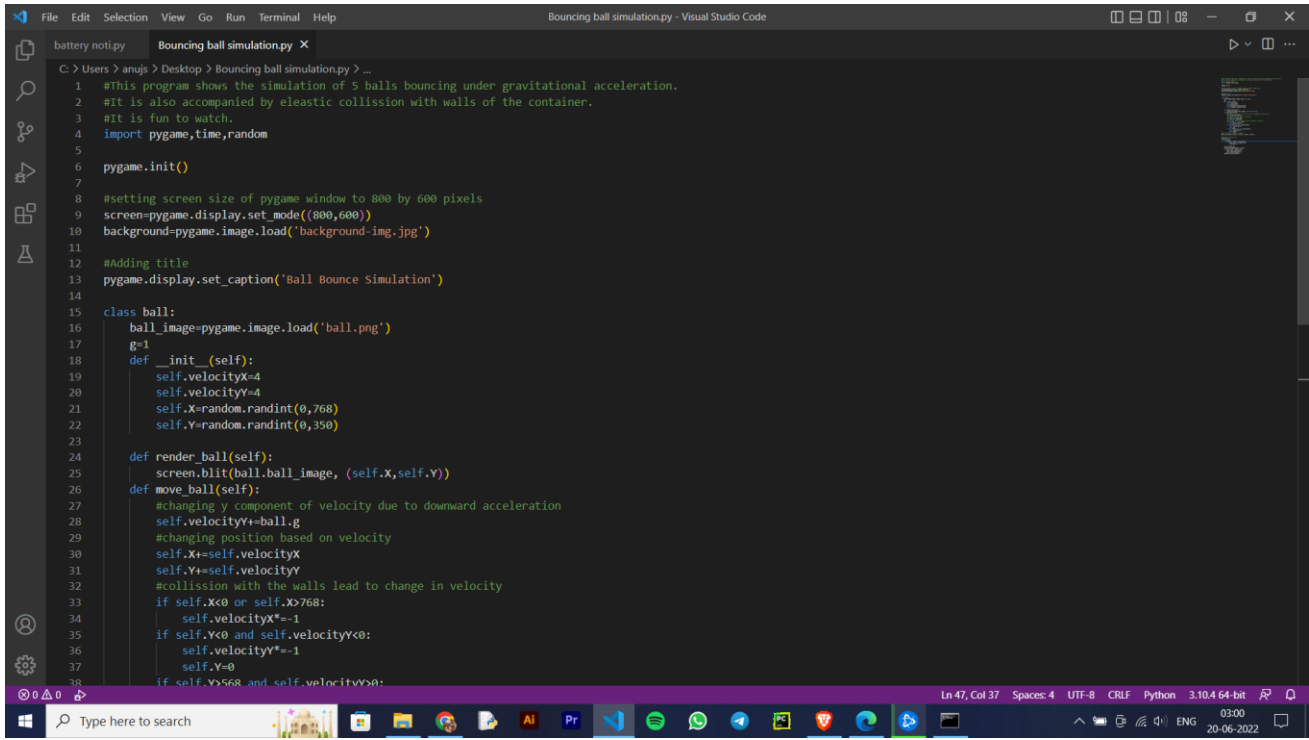


The image shows a screenshot of the Visual Studio Code editor interface. The title bar at the top reads "Visual Studio Code". Below it, the file explorer shows the path "C: > Users > anuj > Desktop > Bouncing ball simulation.py". The main editor window displays the following Python code:

```
34 Ball_List=[ball(),ball(), ball(), ball(), ball()]
35 #The main program loop
36 running=True
37 while running:
38     for event in pygame.event.get():
39         if event.type == pygame.QUIT:
40             running=False
41
42     time.sleep(0.02)
43     screen.blit(background, (0,0))
44     for ball_item in Ball_List:
45         ball_item.render_ball()
46         ball_item.move_ball()
47     pygame.display.update()
```

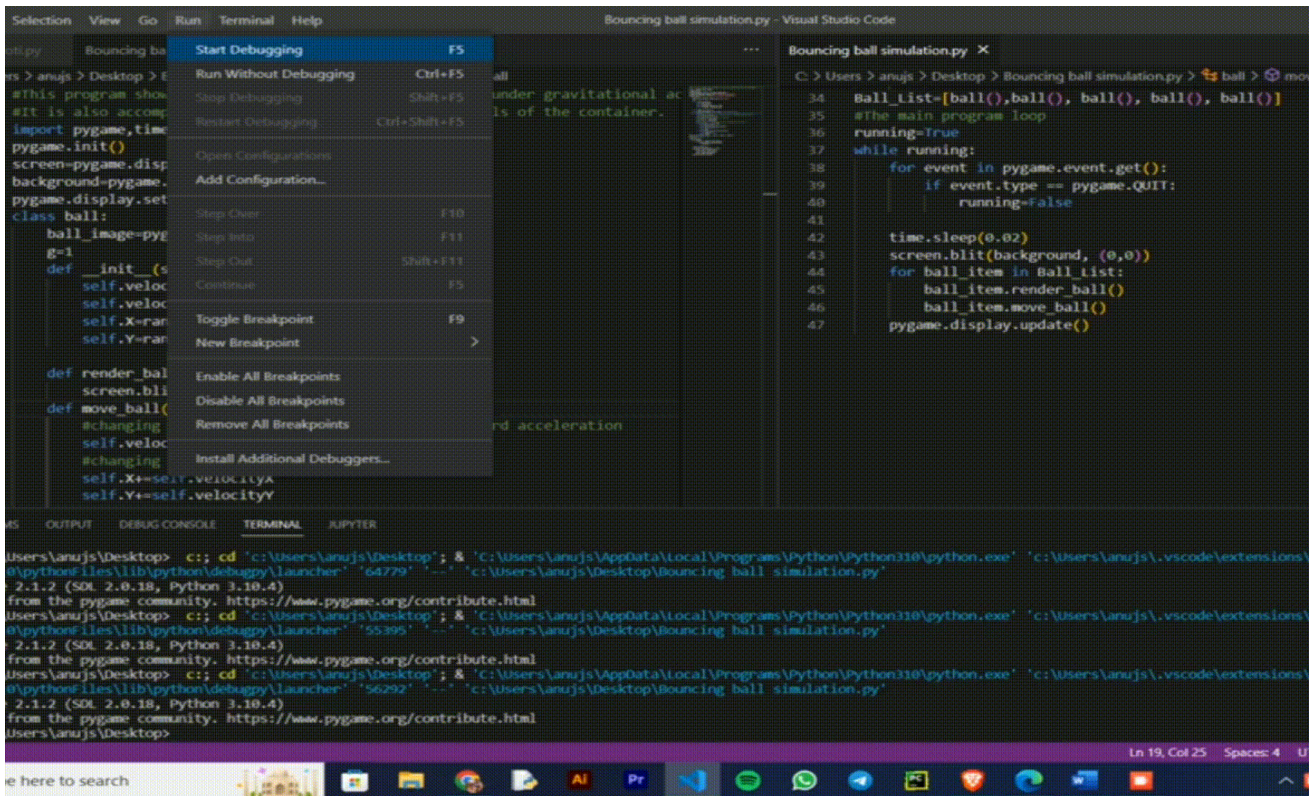
The status bar at the bottom indicates the current position is "Ln 19, Col 25", with "Spaces: 4", "UTF-8", "CRLF", "Python", and "3.10.4 64-bit" settings.

Screenshots of program



```
File Edit Selection View Go Run Terminal Help
Bouncing ball simulation.py - Visual Studio Code

battery not.py Bouncing ball simulation.py X
C:\Users\anju\Desktop> Bouncing ball simulation.py > ...
1 #This program shows the simulation of 5 balls bouncing under gravitational acceleration.
2 #It is also accompanied by elastic collision with walls of the container.
3 #It is fun to watch.
4 import pygame,time,random
5
6 pygame.init()
7
8 #setting screen size of pygame window to 800 by 600 pixels
9 screen=pygame.display.set_mode((800,600))
10 background=pygame.image.load("background-img.jpg")
11
12 #Adding title
13 pygame.display.set_caption("Ball Bounce Simulation")
14
15 class ball:
16     ball_image=pygame.image.load("ball.png")
17     g=1
18     def __init__(self):
19         self.velocityX=4
20         self.velocityY=4
21         self.X=random.randint(0,768)
22         self.Y=random.randint(0,350)
23
24     def render_ball(self):
25         screen.blit(ball_image, (self.X,self.Y))
26     def move_ball(self):
27         #changing y component of velocity due to downward acceleration
28         self.velocityY+=ball.g
29         #changing position based on velocity
30         self.X+=self.velocityX
31         self.Y+=self.velocityY
32         #collision with the walls lead to change in velocity
33         if self.X<0 or self.X>768:
34             self.velocityX*=-1
35         if self.Y<0 and self.velocityY<0:
36             self.velocityY*=-1
37             self.Y=0
38         if self.Y>568 and self.velocityY>0:
```

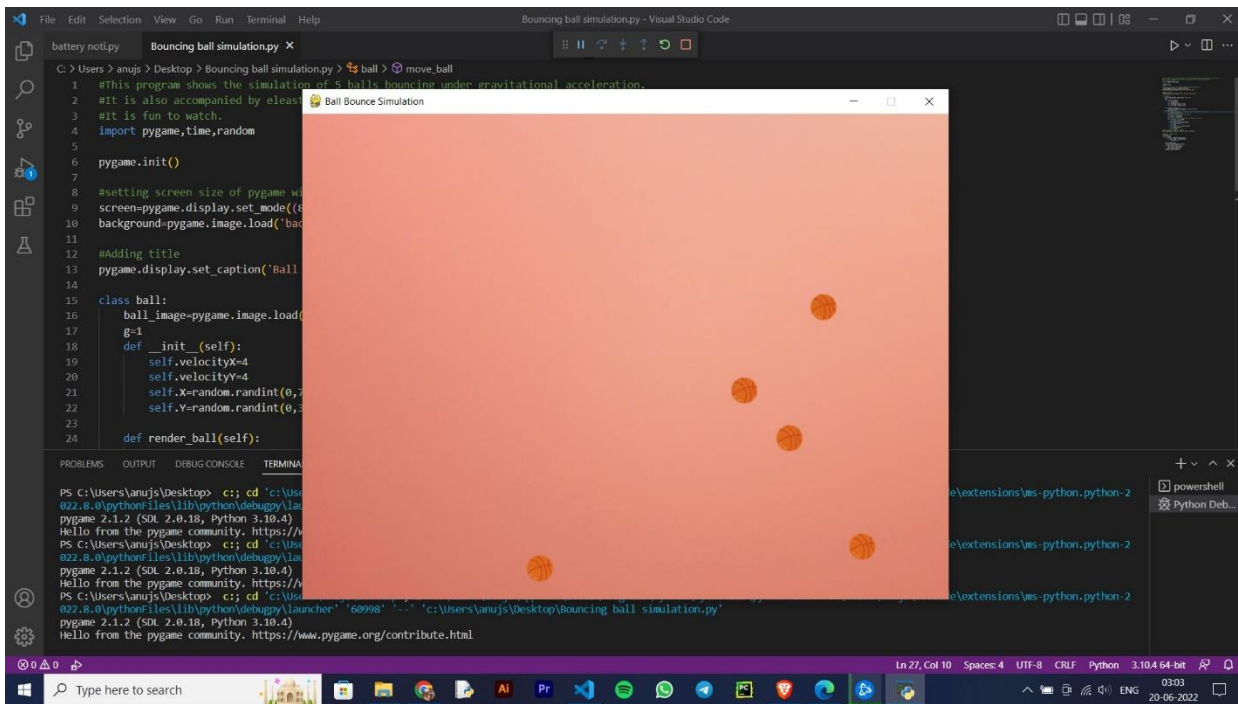
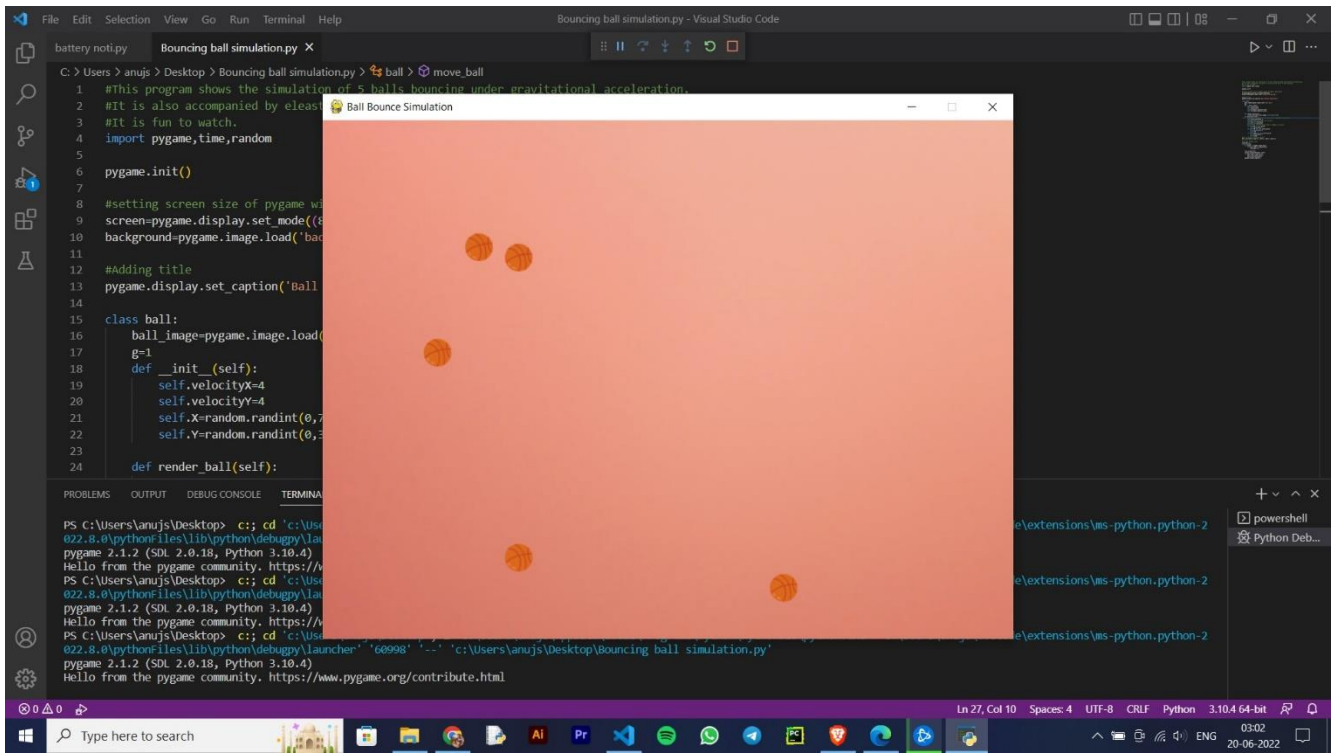


```
Selection View Go Run Terminal Help
Bouncing ball simulation.py - Visual Studio Code

Start Debugging Ctrl+F5
Run Without Debugging Ctrl+F5
Stop Debugging Shift+F5
Restart Debugging Ctrl+Shift+F5
Open Configurations
Add Configuration...
Step Over F10
Step Into F11
Step Out Shift+F11
Continue F5
Toggle Breakpoint F9
New Breakpoint >
Enable All Breakpoints
Disable All Breakpoints
Remove All Breakpoints
Install Additional Debuggers...

Bouncing ball simulation.py X
C:\Users\anju\Desktop> Bouncing ball simulation.py > ball > mo
34 Ball_list=[ball(),ball(), ball(), ball(), ball()]
35 #the main program loop
36 running=True
37 while running:
38     for event in pygame.event.get():
39         if event.type == pygame.QUIT:
40             running=False
41
42     time.sleep(0.02)
43     screen.blit(background, (0,0))
44     for ball_item in Ball_list:
45         ball_item.render_ball()
46         ball_item.move_ball()
47     pygame.display.update()
```


Screenshots of simulation..



References

- YouTube
- Wikipedia
- Brave 1.39.122
- And books

Thank you