PyGame Pong Tutorial - Part 7

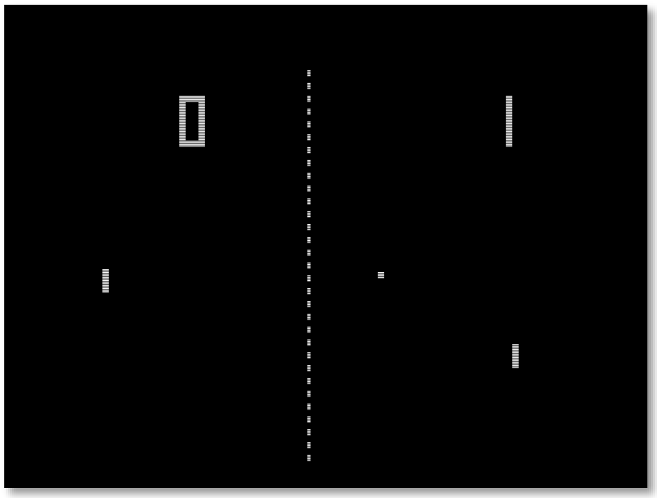
Contents

Time required: 30 minutes

Preview of the Game

Atari. - the year: 1973 - the date: - November 29th - The game is Pong.

Pong Demo Video



Sounds

You can use the sounds in the asset file, or create your own.

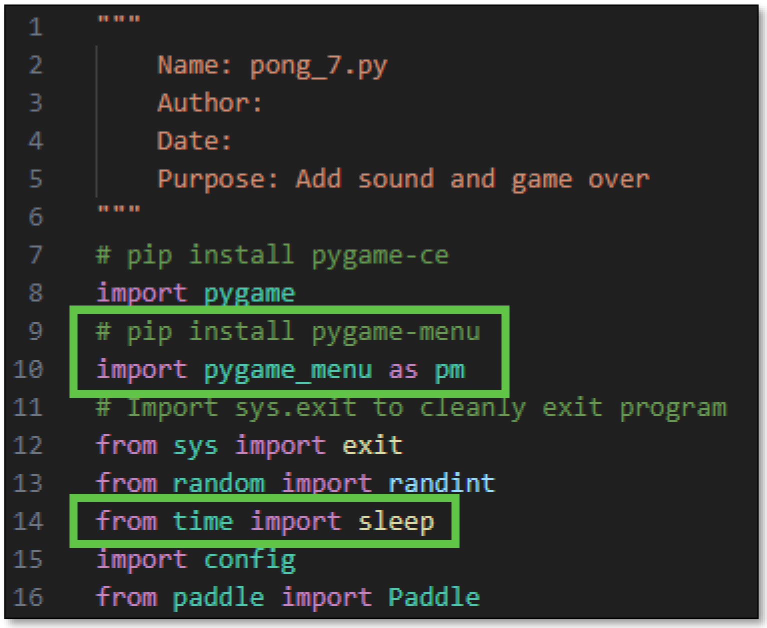
- https://www.beepbox.co (Create 8 bit songs.)

- https://sfxr.me/ (Create sound effects.)
- https://elevenlabs.io/sound-effects
- https://www.leshylabs.com/apps/sfMaker

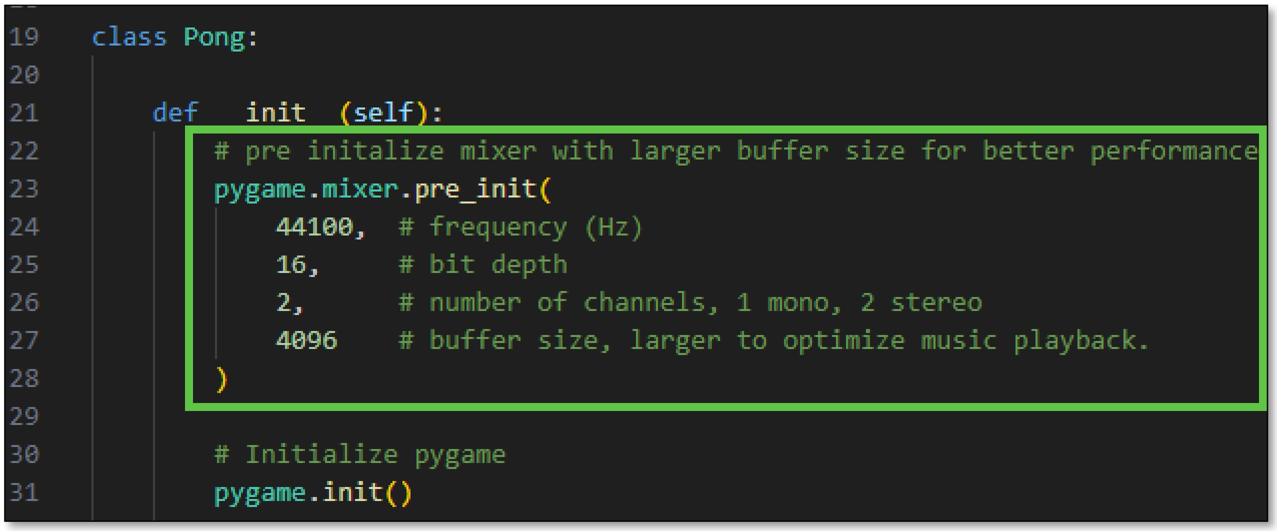
Time for music, sound effects, a game over menu, and a real ping pong game.

pong_assets.zip is attached to this assignment. Unzip it into a folder underneath your game folder called **assets**

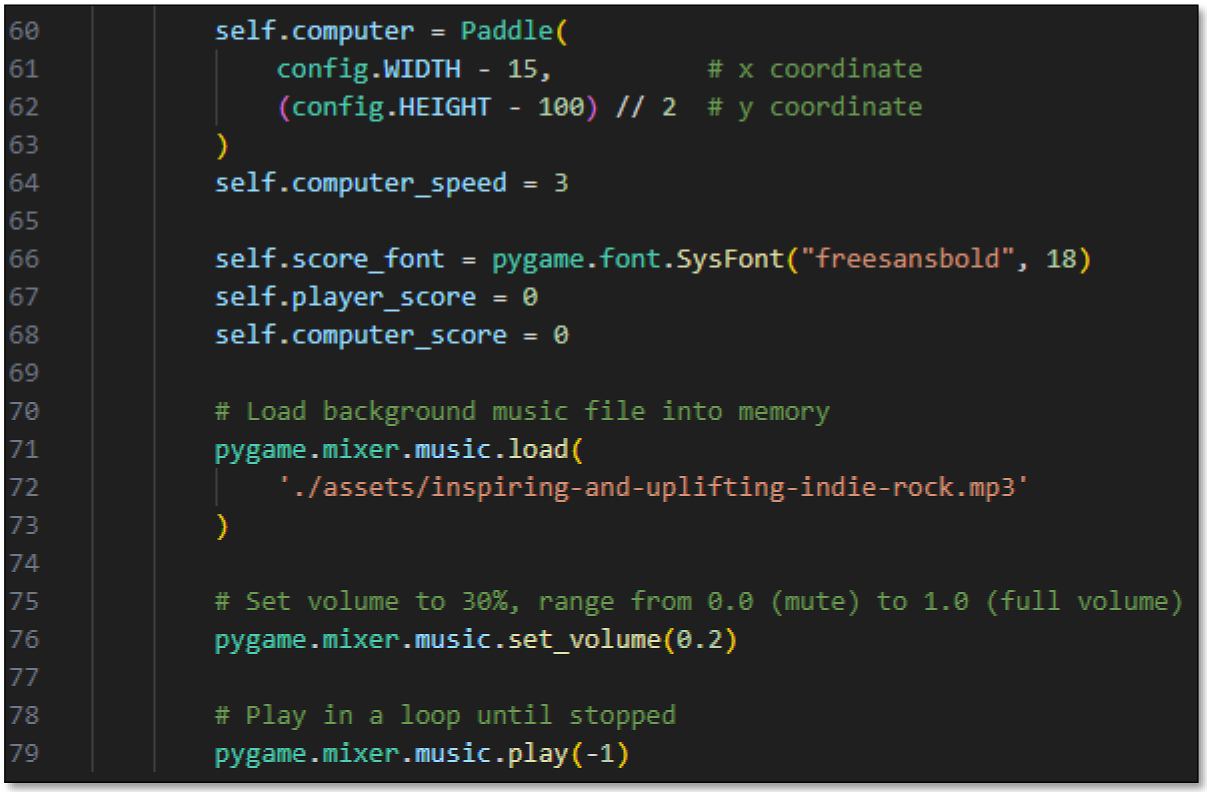
1. Save **pong_6.py** as **pong_7.py**
2. Add the following code.



```python
"""
    Name: pong_7.py
    Author:
    Date:
    Purpose: Add sound and game over
"""
# pip install pygame-ce
import pygame
# pip install pygame-menu
import pygame_menu as pm
# Import sys.exit to cleanly exit program
from sys import exit
from random import randint
from time import sleep
import config
from paddle import Paddle
```



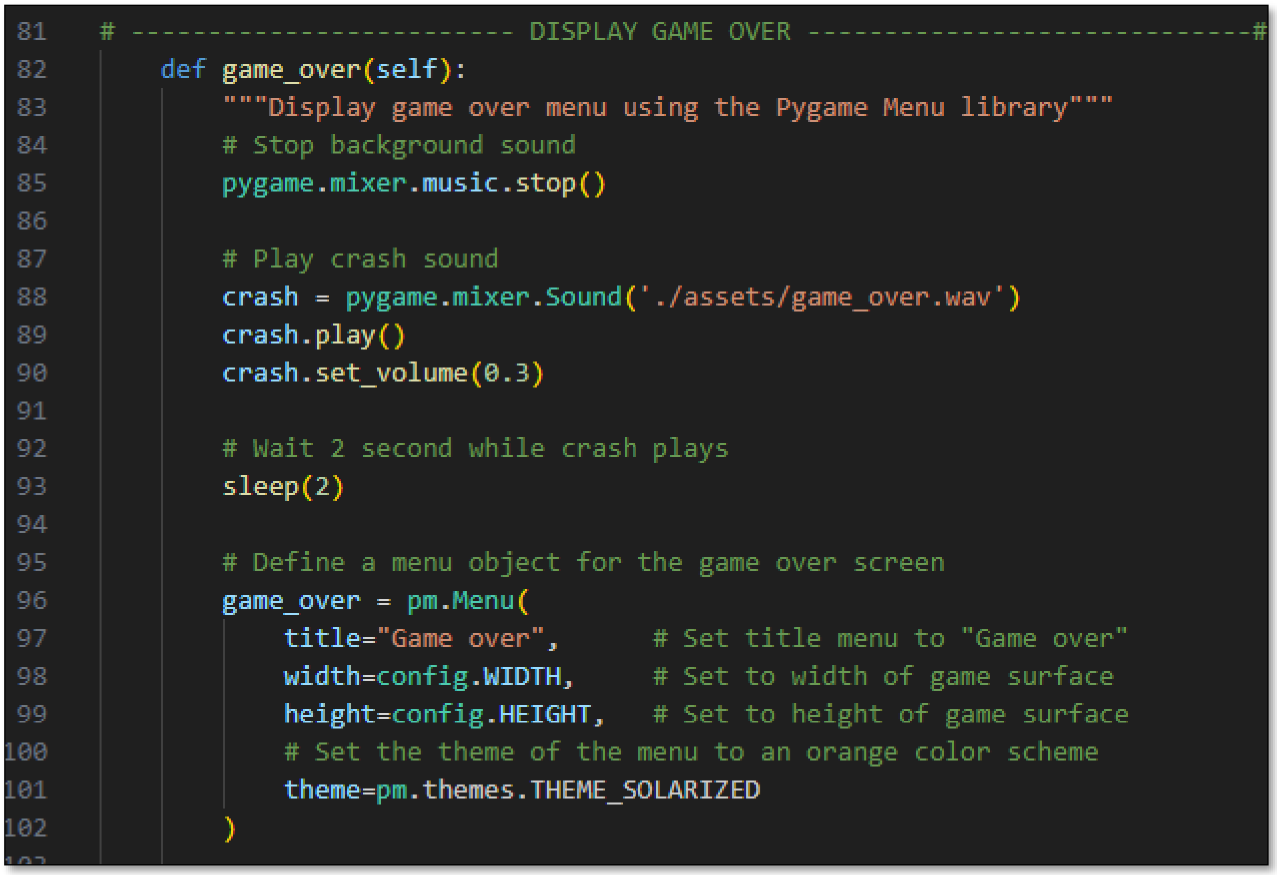
```python
class Pong:

    def __init__(self):
        # pre initalize mixer with larger buffer size for better performance
        pygame.mixer.pre_init(
            44100,  # frequency (Hz)
            16,     # bit depth
            2,      # number of channels, 1 mono, 2 stereo
            4096    # buffer size, larger to optimize music playback.
        )

        # Initialize pygame
        pygame.init()
```



```python
        self.computer = Paddle(
            config.WIDTH - 15,           # x coordinate
            (config.HEIGHT - 100) // 2  # y coordinate
        )
        self.computer_speed = 3

        self.score_font = pygame.font.SysFont("freesansbold", 18)
        self.player_score = 0
        self.computer_score = 0

        # Load background music file into memory
        pygame.mixer.music.load(
            './assets/inspiring-and-uplifting-indie-rock.mp3'
        )

        # Set volume to 30%, range from 0.0 (mute) to 1.0 (full volume)
        pygame.mixer.music.set_volume(0.2)

        # Play in a loop until stopped
        pygame.mixer.music.play(-1)
```

Game Over

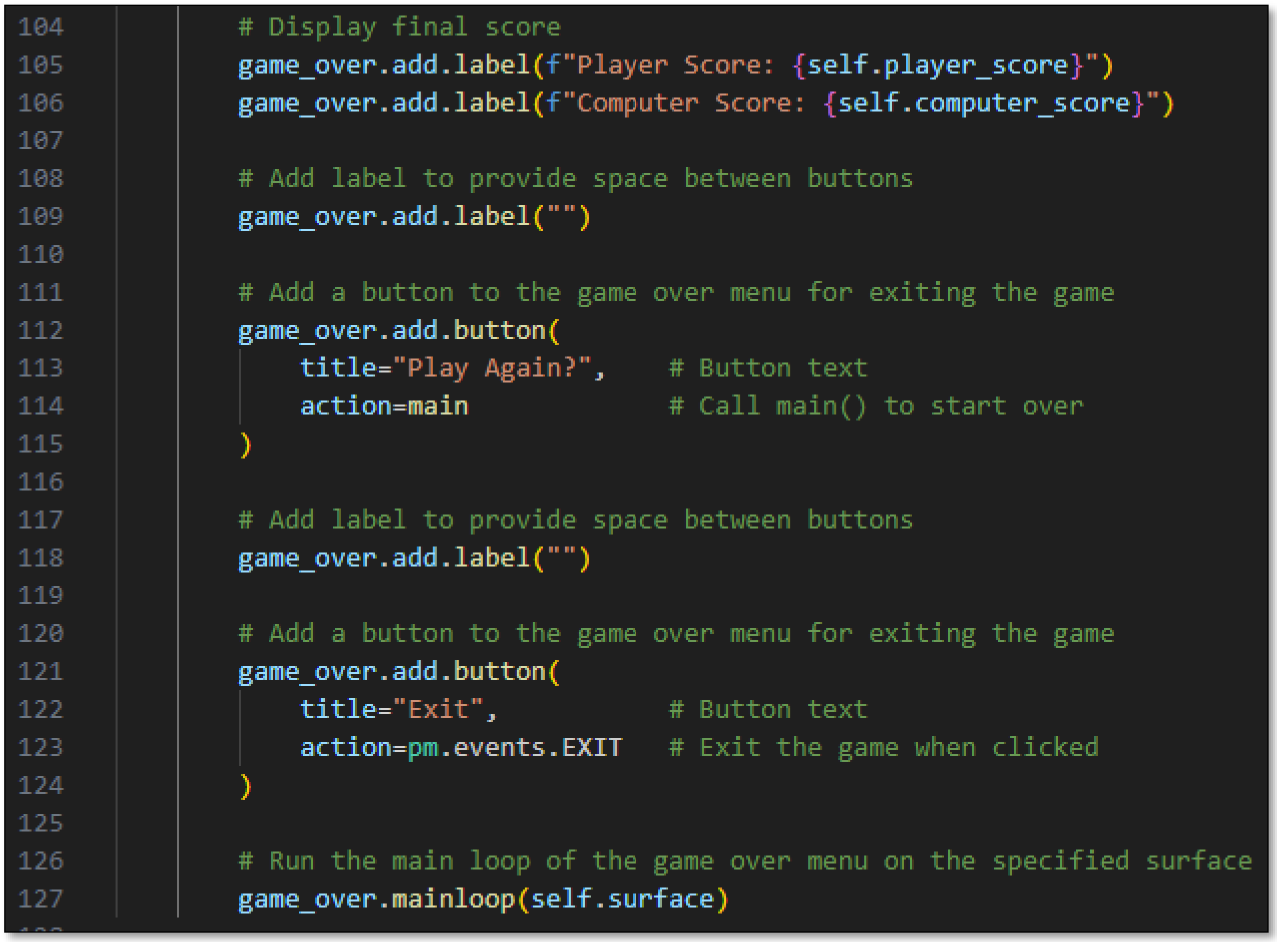
Add the following game_over method.



```python
# ------------------------ DISPLAY GAME OVER ----------------------------#
    def game_over(self):
        """Display game over menu using the Pygame Menu library"""
        # Stop background sound
        pygame.mixer.music.stop()

        # Play crash sound
        crash = pygame.mixer.Sound('./assets/game_over.wav')
        crash.play()
        crash.set_volume(0.3)

        # Wait 2 second while crash plays
        sleep(2)

        # Define a menu object for the game over screen
        game_over = pm.Menu(
            title="Game over",      # Set title menu to "Game over"
            width=config.WIDTH,     # Set to width of game surface
            height=config.HEIGHT,   # Set to height of game surface
            # Set the theme of the menu to an orange color scheme
            theme=pm.themes.THEME_SOLARIZED
        )
```

There are different themes you can choose for the game_over object. This example uses THEME_SOLARIZED. You can use any of the following to customize your menu.

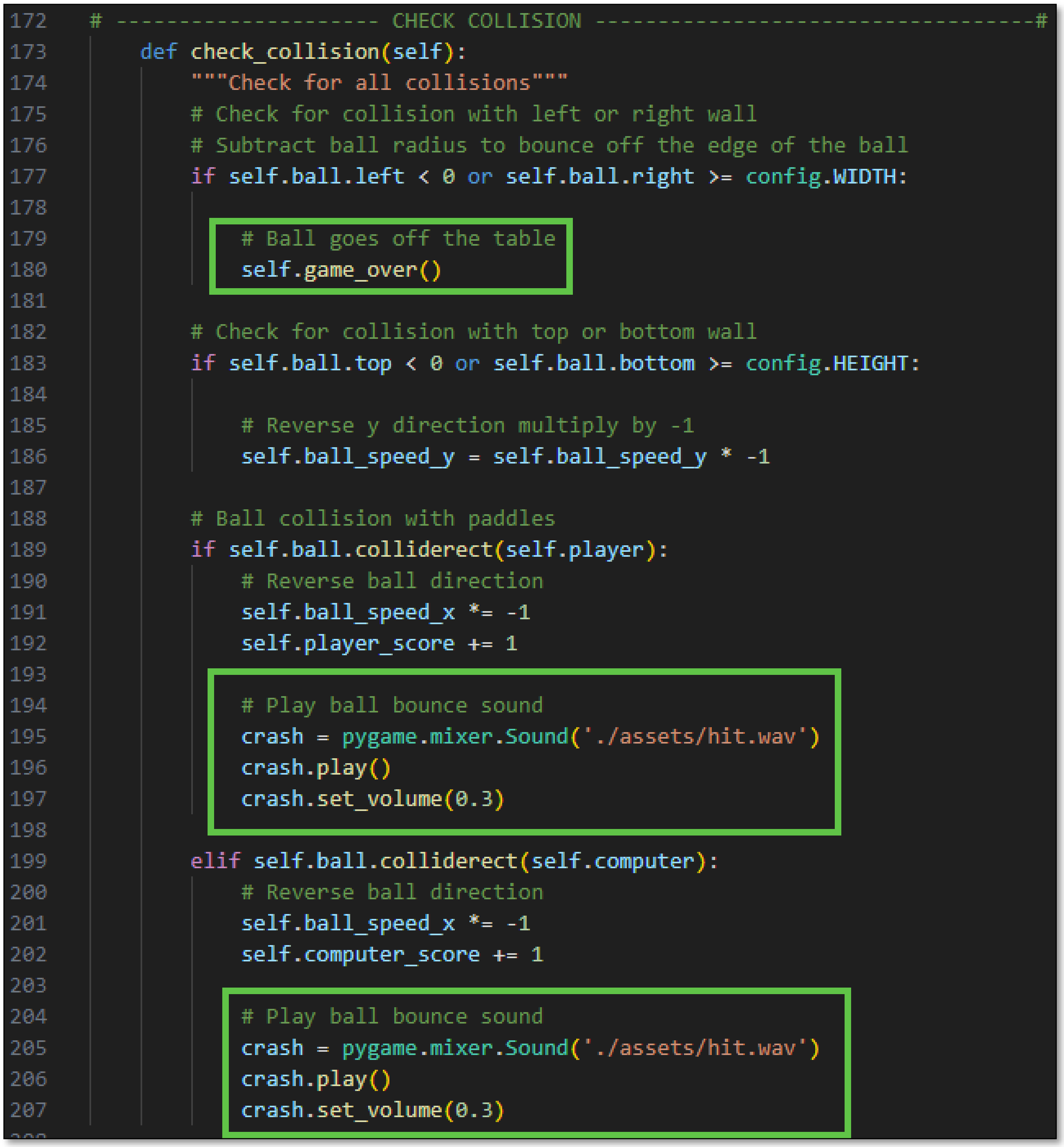
```
THEME_BLUE
THEME_DARK
THEME_DEFAULT
THEME_GREEN
THEME_ORANGE
THEME_SOLARIZED
```

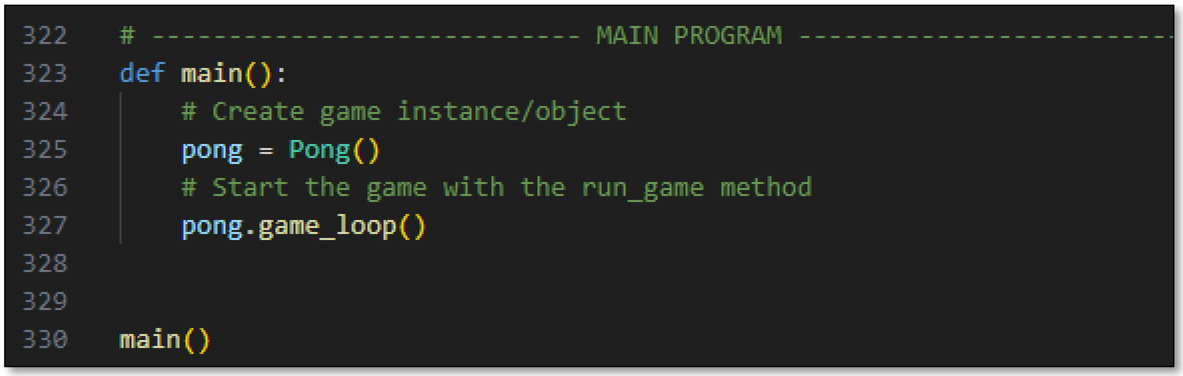


```
        # Display final score
        game_over.add.label(f"Player Score: {self.player_score}")
        game_over.add.label(f"Computer Score: {self.computer_score}")

        # Add label to provide space between buttons
        game_over.add.label("")

        # Add a button to the game over menu for exiting the game
        game_over.add.button(
            title="Play Again?",     # Button text
            action=main              # Call main() to start over
        )

        # Add label to provide space between buttons
        game_over.add.label("")

        # Add a button to the game over menu for exiting the game
        game_over.add.button(
            title="Exit",            # Button text
            action=pm.events.EXIT    # Exit the game when clicked
        )

        # Run the main loop of the game over menu on the specified surface
        game_over.mainloop(self.surface)
```

Check Collision

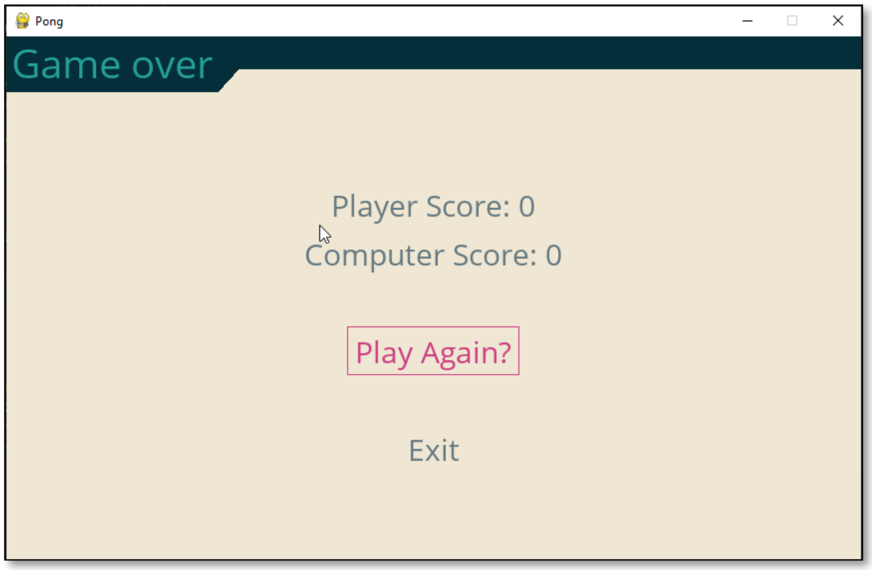
Modify the check collision method.


```python
    # -------------------- CHECK COLLISION --------------------------------#
    def check_collision(self):
        """Check for all collisions"""
        # Check for collision with left or right wall
        # Subtract ball radius to bounce off the edge of the ball
        if self.ball.left < 0 or self.ball.right >= config.WIDTH:

            # Ball goes off the table
            self.game_over()

        # Check for collision with top or bottom wall
        if self.ball.top < 0 or self.ball.bottom >= config.HEIGHT:

            # Reverse y direction multiply by -1
            self.ball_speed_y = self.ball_speed_y * -1

        # Ball collision with paddles
        if self.ball.colliderect(self.player):
            # Reverse ball direction
            self.ball_speed_x *= -1
            self.player_score += 1

            # Play ball bounce sound
            crash = pygame.mixer.Sound('./assets/hit.wav')
            crash.play()
            crash.set_volume(0.3)

        elif self.ball.colliderect(self.computer):
            # Reverse ball direction
            self.ball_speed_x *= -1
            self.computer_score += 1

            # Play ball bounce sound
            crash = pygame.mixer.Sound('./assets/hit.wav')
            crash.play()
            crash.set_volume(0.3)
```



```python
# ---------------------------- MAIN PROGRAM ------------------------
def main():
    # Create game instance/object
    pong = Pong()
    # Start the game with the run_game method
    pong.game_loop()


main()
```

Example run:



Tada, a real game!

There is always room for improvement.

What's Next?

- Change the colors to different RGB color.
- Change the size or shape of the ball or paddles.
- Add more difficulty levels.
- Keep track of the highest score between games.
- Add more music, change the music
- Change the game to make it your own.

Assignment Submission

Zip up the program files folder and submit in Blackboard.