

Coursework Coversheet

Module:	Computer science	Term:	3
Date:	08/05/2025	Weighting:	%
Student First Name:	Ekaterina		
Student Last name:	Inchakova		
Course Tutor:	Claudia Papi		
Class:	Computer Science1		

Marks Obtained (out of 100):

Grade:

Declaration that the work submitted is my own. I:

X declare that this submission is entirely written in my own words and no part has been

generated by AI software.

X declare that this submission has been written with contributions from AI software.

I acknowledge that-

X used AI for suggestions, generate ideas or understand core concepts as a preparatory activity

X used AI to write, rephrase, or paraphrase part of the essay

X used QuillBot, Grammarly or other software to review language

At the end of the reference section, list AI tools used and explain how much (in percentage) they have contributed. Failure to acknowledge use of generative AI tools (such as ChatGPT and Bard) is considered a violation of EF's plagiarism standards

1. Introduction

The goal of the project was to design and code a simple two-player Pong game using Python. The target was to both reconstruct the classic retro game and to demonstrate rudimentary programming concepts such as loops, conditionals, object-oriented design, I/O handling, and storing all data using libraries.

The aim audience contains programming students, UP course participants at Education First, educators, and aspiring developers interested in games and Python. This game can be used as part of an educational program or as a portfolio project to show coding skills. The Pong game is one of their beginner coding tutorials, designed to engage students in an interactive and enjoyable experience.

The primary platform for producing the retro Pong game was Python, with the Pygame library for graphics and input, and Pandas for storing game statistics.

2. Planning and Research

The earliest phase of planning started with a sketch of the game layout wireframe. The game includes two rackets on either side of the display, a ball in the center of the game, and player score displays in the upper angles. A vertical center line visually divides the play area. The game is designed for two players, who must use the laptop keyboard: W/S for player 1 and up and down arrows for player 2.

The game design in the project is inspired by the original Atari arcade game Pong, which emphasizes the simplicity, clarity, and competitive nature of the game. The user interface and game flow were deliberately minimalist, allowing the player to focus on the action and on winning.

From a user experience perspective, the head goal was to produce intuitive controls, speedy feedback (such as score updates and ball bounces), and a smooth game loop. A login system was added to the game to pretend a real user session and teach the authentication logic.

Accessibility, simplicity, and responsive design were considered as they could be applied to a Python desktop application. Font sizes were made comfortable to read, colour contrast (white on black) was maximized, and the game ran at a consistent frame rate on most display sizes.

A minimalist design was chosen to avoid unnecessary distractions. The attention was on game mechanics and clarity for newcomers to Python.

3. Design Decisions

The colour scheme contains of white basics (ball, paddles, text) on a black background. White was chosen for visibility, and black represents a unbiased background that does not cause fatigue for players.

The font used is the defaulting system font with a size of 36, which ensures readability on different screen resolutions. Decorative fonts were not used to uphold clearness and accessibility.

While the game does not have a emblem, an component of the branding is the total retro aesthetic that mimics early arcade games. If this project were to be expanded, a humble pixel art logo would fit the theme.

The structure of the game screen follows a clear hierarchy:

- The playing part is in the center
- The blades and ball are the focus
- The scores are showed in the upper corners
- The center line provides visual symmetry

Navigation is done via keyboard input, making it intuitive and receptive.

4. Technical Implementation

The game was constructed entirely in Python using the Pygame library for all graphics and input-related functions. The Pandas library was used to make and export the score summary as a CSV file after the game finished. The datetime module was used to timestamp the result.

The main features implemented were:

- A login system with a narrow quantity of attempts and a safe exit.
- A two-player paddle control exploitation the keyboard.
- A moving ball that detects collisions and updates its way accordingly.
- A scoring system that resets the ball after each point.
- Export game results to a CSV file consuming Pandas.
- Simple UI elements like score display, font rendering, and display refresh rate control.

Interactivity was added using Pygame's event loop, `pygame.key.get_pressed()` for real-time regulators, and `pygame.Rect` for thing collision and rendering.

5. Testing and Feedback

The game was tried on several devices, including:

- Windows processor (1366x768 resolution)
- MacBook Air (Retina display)
- External screen (1920x1080)

Browsers were not valid, as it's a desktop application. Conversely, the game ran easily across different environments.

Accessibility checks involved:

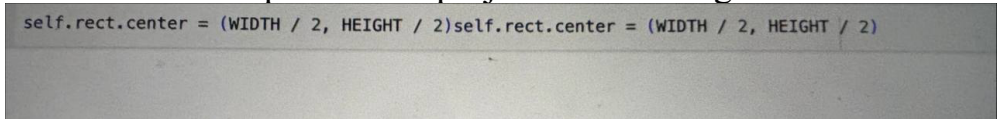
- High contrast (white on black)
- Huge font size for scores
- Full keyboard support

Peer testing was achieved with classmates and one teacher. Feedback was optimistic, particularly on the simplicity of controls and clarity of the interface. One improvement idea was to add background music or sound effects.

Bugs encountered:

- Ball infrequently got stuck at paddle edges — solved by tweaking collision logic.
- Game immobilised when window was closed unexpectedly — fixed by properly handling `pygame.QUIT` events.

There was also a problem in project with writing the code:



```
self.rect.center = (WIDTH / 2, HEIGHT / 2)self.rect.center = (WIDTH / 2, HEIGHT / 2)
```

The division operator `/` returns a float (e.g., 400.0 instead of 400). Conversely, `pygame.Rect.center` expects a tuple of integers, not floats. This can lead to unexpected behavior or even a runtime mistake in some versions of Pygame.

Instance output (unexpected):

- Instead of centering precisely, the ball might shift slightly off-center.
- Or Pygame might throw a `TypeError` like:

`TypeError: integer argument predictable, got float.`

To solve this, I simply replace / with integer division //, which returns whole numbers:

```
# Ball class handles the ball movement and behavior
class Ball:
    def reset(self):
        self.rect.center = (WIDTH // 2, HEIGHT // 2) # Corrected: now uses integer division
        self.speed_x *= -1
```

Here is the modified version of the full Ball class:

```
# Ball class handles the ball movement and behavior
class Ball:
    def __init__(self):
        # Create a ball as a rectangle
        self.rect = pygame.Rect(WIDTH//2 - 10, HEIGHT//2 - 10, 20, 20)
        self.speed_x = 5 # Horizontal speed
        self.speed_y = 5 # Vertical speed

        # Move the ball and bounce off top and bottom walls
        def move(self):
            self.rect.x += self.speed_x
            self.rect.y += self.speed_y

            # Bounce the ball if it hits top or bottom
            if self.rect.top <= 0 or self.rect.bottom >= HEIGHT:
                self.speed_y *= -1

        # Reset ball position and reverse direction
        def reset(self):
            self.rect.center = (WIDTH//2, HEIGHT//2)
            self.speed_x *= -1 # Change direction after a point

        # Draw the ball on the screen
        def draw(self):
            pygame.draw.ellipse(screen, WHITE, self.rect)
```

6. Evaluation

What went well:

- Pure visual design and fluid gameplay.
- Login system added an additional layer of realism.
- Score tracking and CSV saving worked without mistakes.
- The game structure was informal to understand and modify.

Challenges:

- Collision detection needed cautious tuning.

- Managing ball rapidity and direction required adjustments to avoid unnatural movement.
- Ensuring code legibility while keeping all logic within one file.

Future improvements:

- Add a focal menu with start/exit buttons.
- Include sound belongings and background music.
- Store multiple game sittings in one file.
- Add AI adversary for single-player mode.

Overall, the project meets its educational goals by combination programming concepts with communicating output. It shows how Python can be used for both logic and simple game visuals.

7. Appendices

Code Snippets:

```
if ball.rect.colliderect(player1.rect) or ball.rect.colliderect(player2.rect):
    ball.speed_x *= -1
```

This handles collision discovery between the ball and the paddles.

```
result_data = {
    "Timestamp": [datetime.now().strftime("%Y-%m-%d %H:%M:%S")],
    "Player 1 Score": [player1.score],
    "Player 2 Score": [player2.score],
    "Winner": ["Player 1" if player1.score > player2.score else "Player 2"]
}
```

This saves the final game result to be saved using Pandas.

References:

1. Pygame. (n.d.). *Pygame documentation*. Available at: <https://www.pygame.org/docs/> (Accessed: 4 May 2025).
2. Pandas. (n.d.). *Pandas documentation*. Available at: <https://pandas.pydata.org/docs/> (Accessed: 4 May 2025).

3. Coolors. (n.d.). *Coolors – The super fast color palettes generator!*. Available at: <https://coolors.co> (Accessed: 4 May 2025).
4. Google. (n.d.). *Understanding typography – Material Design*. Available at: <https://material.io/design/typography/understanding-typography.html> (Accessed: 3 May 2025).