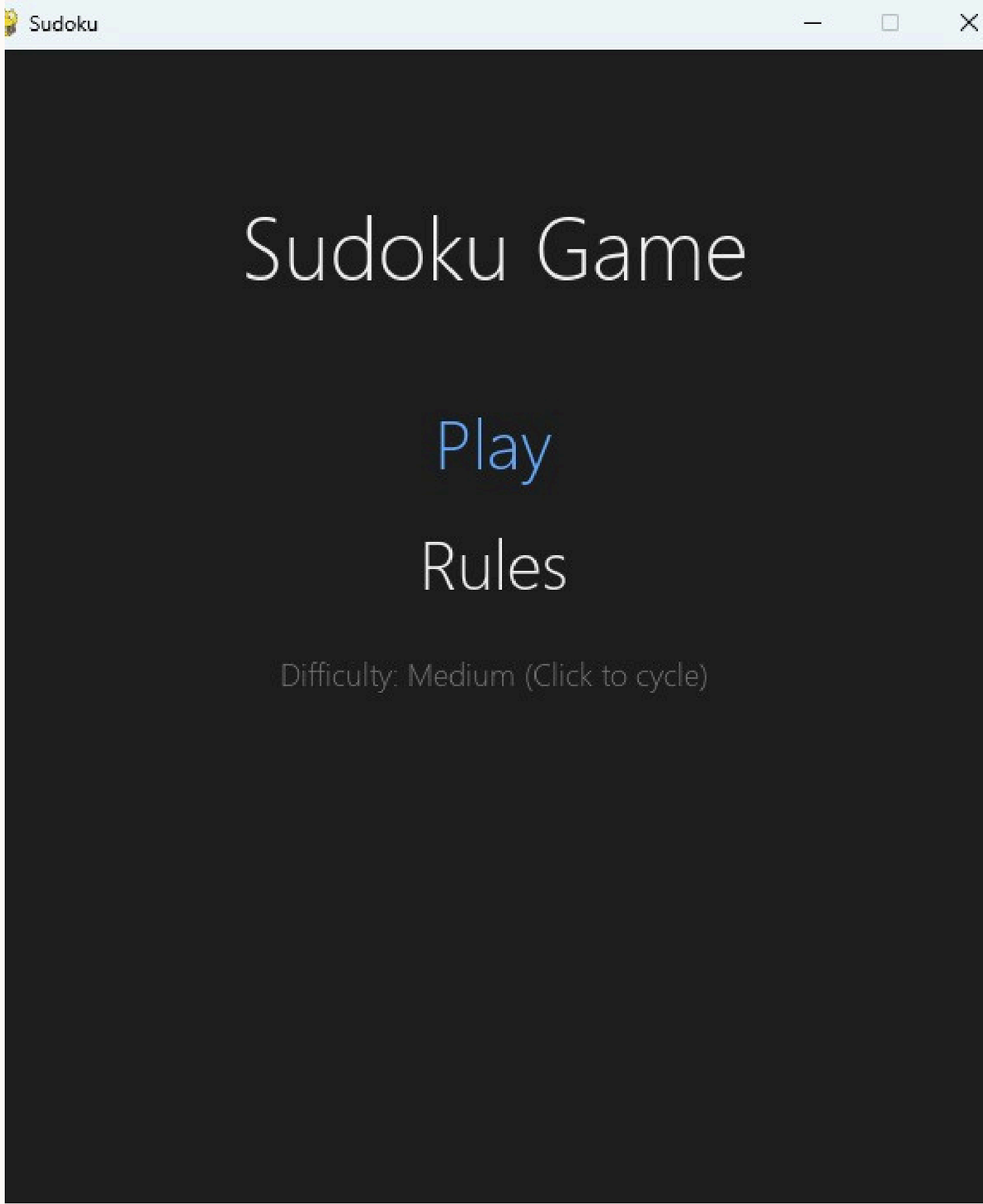
Sudoku Solver: A Python Game

Submitted By:

Sneha Jain	**MABSPG24008**
Aditi Verma	**MABSPG24018**
Mehak Gupta	**MABSPG24037**
Adarsh Nair	**MABSPG24041**
Chirag Thakkar	**MABSPG24051**



What is Sudoku?

Brief History

Sudoku is a logic-based number placement puzzle. The modern Sudoku started in 1986. It gained international popularity in 2005.

Rules

Fill a 9x9 grid with digits 1-9 so that each column, each row, and each of the nine 3x3 subgrids contains all of the digits from 1 to 9.

Strategies

The simplest solving strategy is scanning and marking. More advanced techniques include elimination and what-if analysis.

Project Overview: Python Implementation

Project Goal

The goal was to create a functional and efficient Sudoku game using Python.

- Efficient core logic
- Board generation
- Solving Algorithm
- User Interface

Core Libraries

The project uses the following libraries:

- pygame: For GUI and input handling
- random, copy: For puzzle generation and state management

Features of the Game

Difficulty Levels

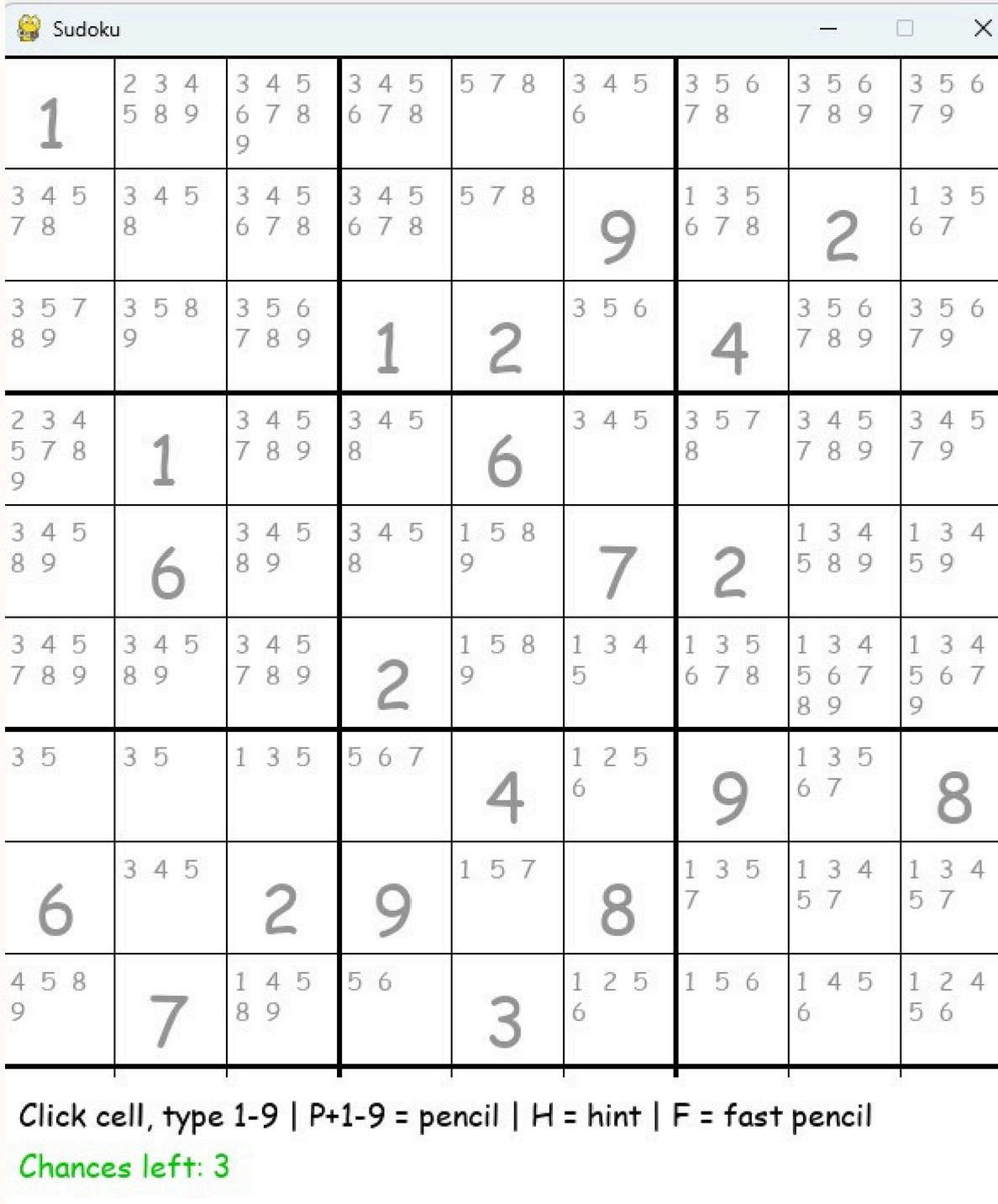
The game offers Easy, Medium, and Hard difficulty levels. This allows players to challenge themselves appropriately.

Pencil Marks

Pencil marks can be added using ‘P + number’, aiding complex solving.

Hint System

The 'H' key gives hints. Autofill pencil marks with 'F' key.



Sudoku Logic (Code Overview)

1. **is_valid()**
   Checks if a number can be legally placed in a cell.

2. **solve_board()**
   Uses backtracking to solve the Sudoku board.

3. **generate_full_board()**
   Generates a complete, solved Sudoku board.

4. **generate_puzzle()**
   Removes numbers from a solved board to create a puzzle.

5. **is_complete()**
   Validates if the game is correctly completed.

```python
def generate_full_board():
    board = [[0]*9 for _ in range(9)]
    solve_board(board)
    return board

def generate_puzzle(board, holes):
    puzzle = copy.deepcopy(board)
    count = 0
    while count < holes:
        row = random.randint(0, 8)
        col = random.randint(0, 8)
        if puzzle[row][col] != 0:
            puzzle[row][col] = 0
            count += 1
    return puzzle

def is_complete(board):
    for row in range(9):
        for col in range(9):
            if board[row][col] == 0 or not is_valid(boar
                return False
    return True
```

```python
# Sudoku Logic
def is_valid(board, row, col, num):
    for i in range(9):
        if board[row][i] == num or board[i][col] == num:
            return False
    start_row, start_col = 3 * (row // 3), 3 * (col // 3)
    for i in range(3):
        for j in range(3):
            if board[start_row + i][start_col + j] == num:
                return False
    return True

def solve_board(board):
    for row in range(9):
        for col in range(9):
            if board[row][col] == 0:
                for num in range(1, 10):
                    if is_valid(board, row, col, num):
                        board[row][col] = num
                        if solve_board(board):
                            return True
                        board[row][col] = 0
                return False
    return True
```

Game Design (GUI)

Grid Display

The Sudoku grid is displayed using pygame.

Cell Selection

Users click to select a cell.

Value Entry

Keyboard input is used to enter values.

Pencil Marks

Pencil marks are visible in corners of cells.

Main Game Loop

User Events

Handles keyboard and mouse events.

Input Tracking

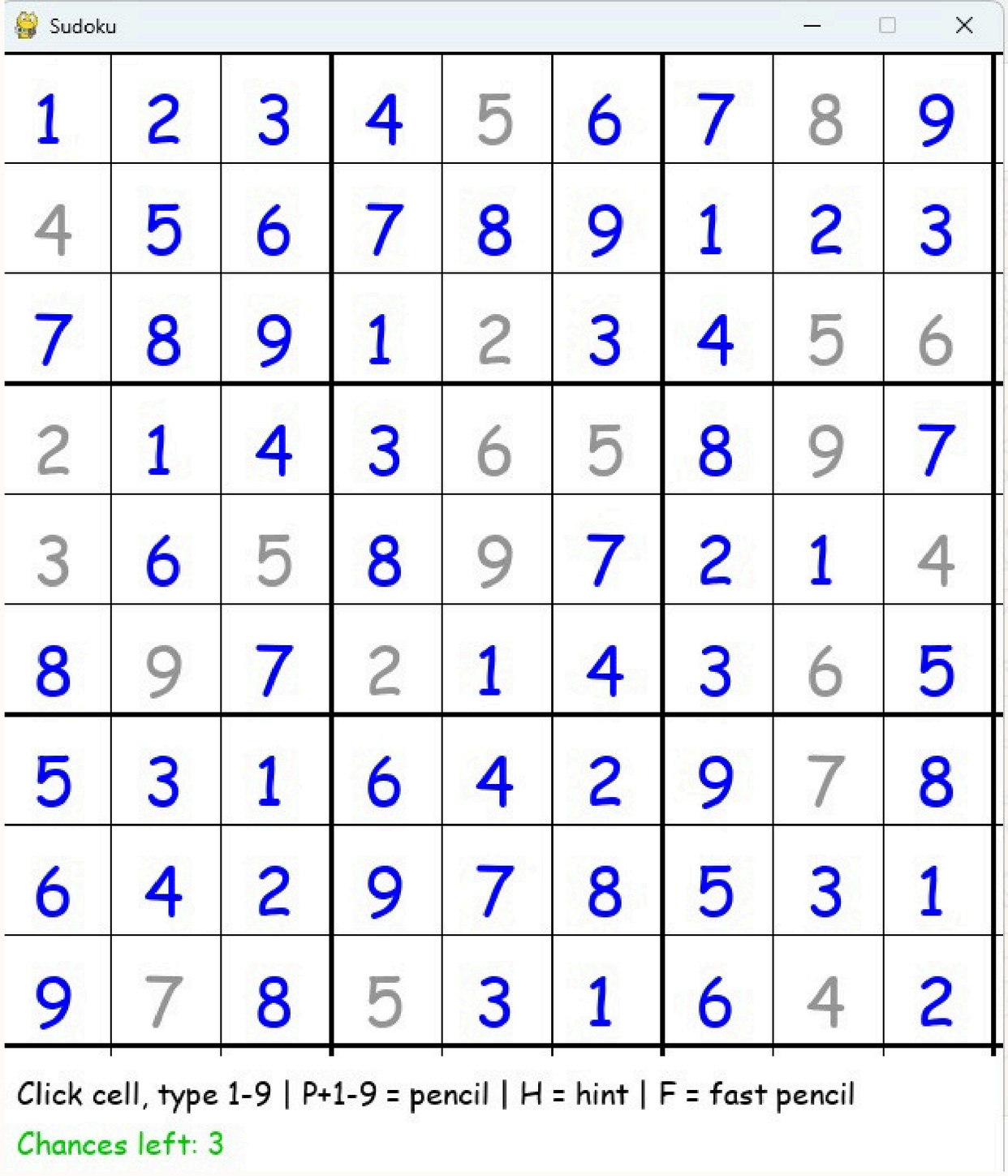
Tracks current selection and pencil mode.

Game State

Responds to hint and autofill keys.

Game End

Ends when puzzle is solved or chances are exhausted.



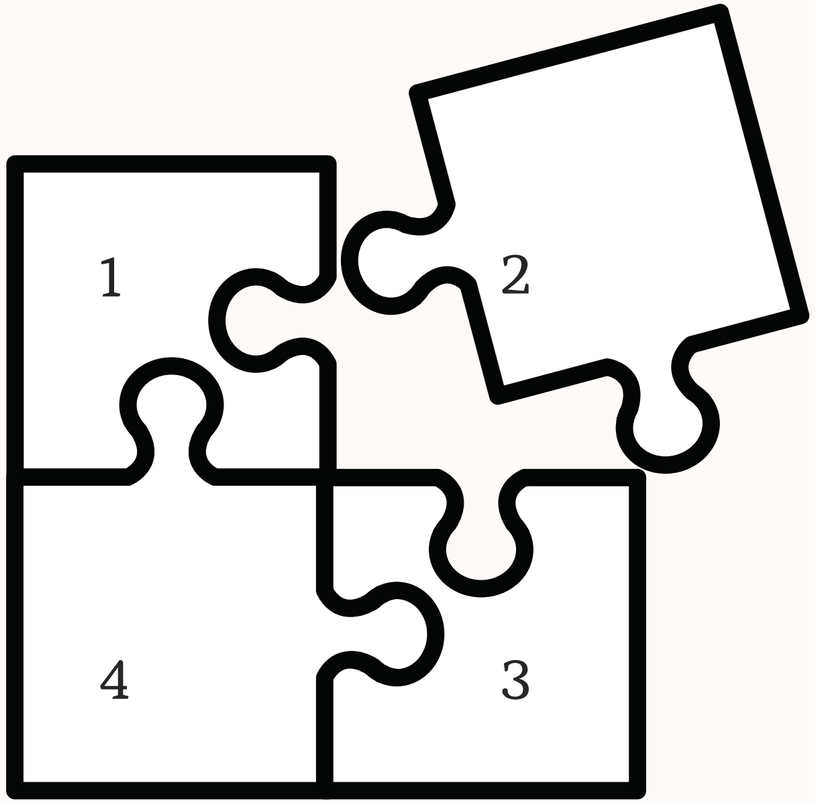
Challenges

Real-time Updates

Managing board updates in real-time was challenging.

Feature Integration

Integrating pencil mark features smoothly took time.



Unique Puzzles

Ensuring unique puzzle generation was difficult.

Balancing Act

Balancing UI design with logic handling needed work.

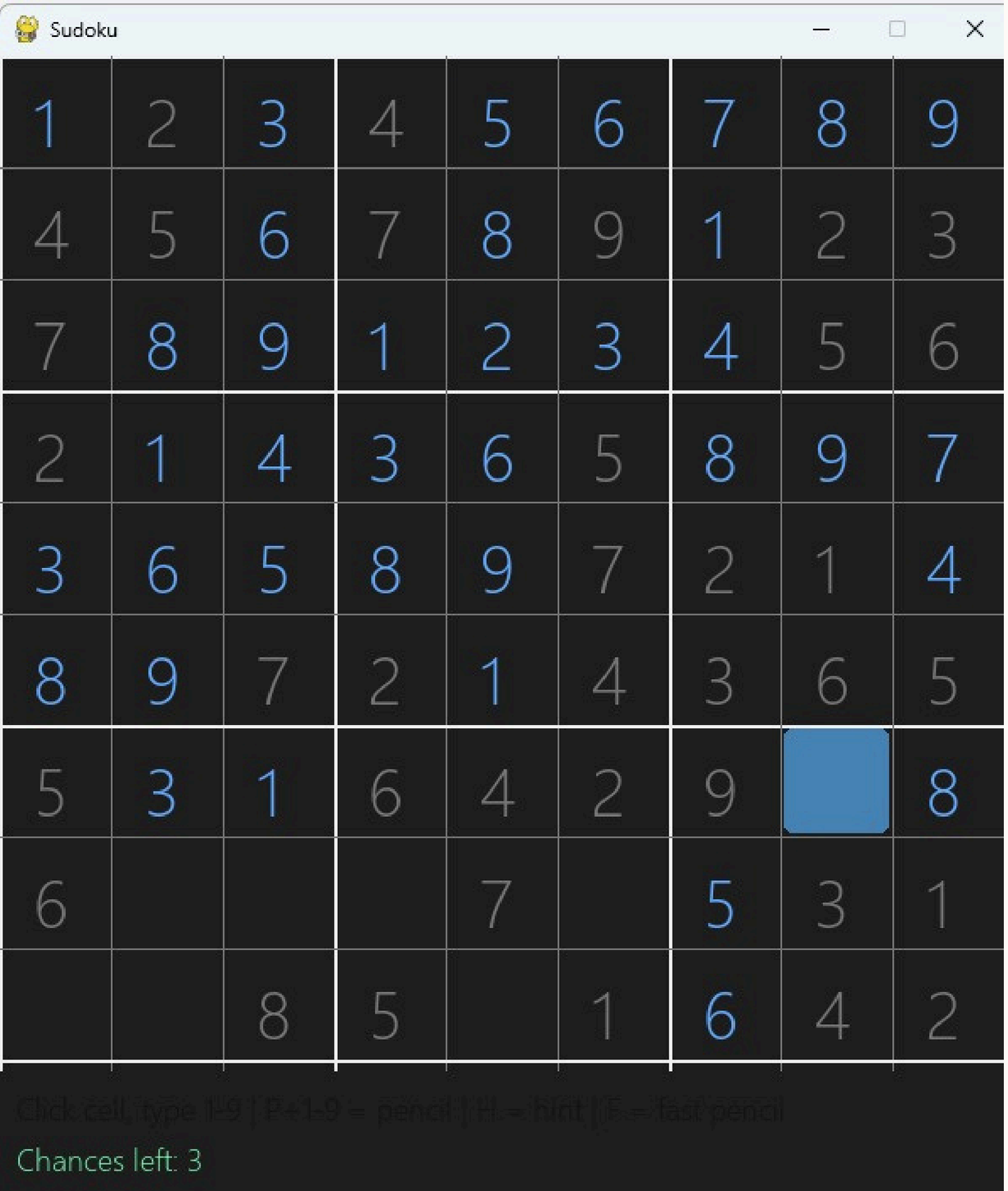
Learnings & Further Changes

Key Learnings

- Backtracking Algorithms
- Practical Pygame Use
- Efficient Game Loop
- User-Friendly Design

Future Enhancements

- Add timer and scoring
- Save/load progress
- Touch screen support
- Online leaderboard



Conclusion

The Python Sudoku project achieved its goals.

It provides a functional game with a clean interface. It serves as an educational tool and a portfolio piece.

This project demonstrates practical application of game development principles.

THANK YOU