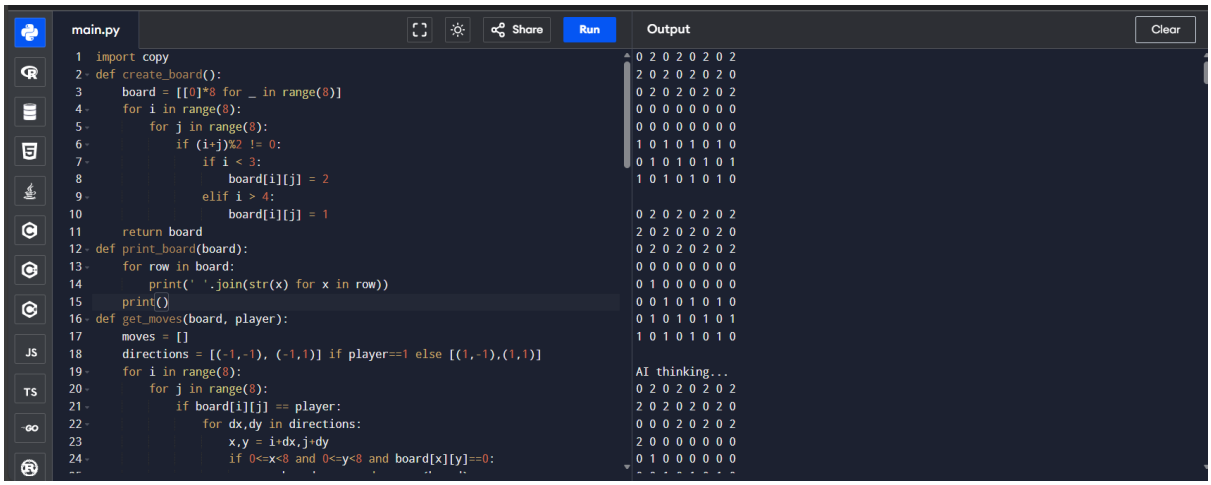
MLA0109 – Artificial Intelligence and Expert System

Python Programs – Day 2

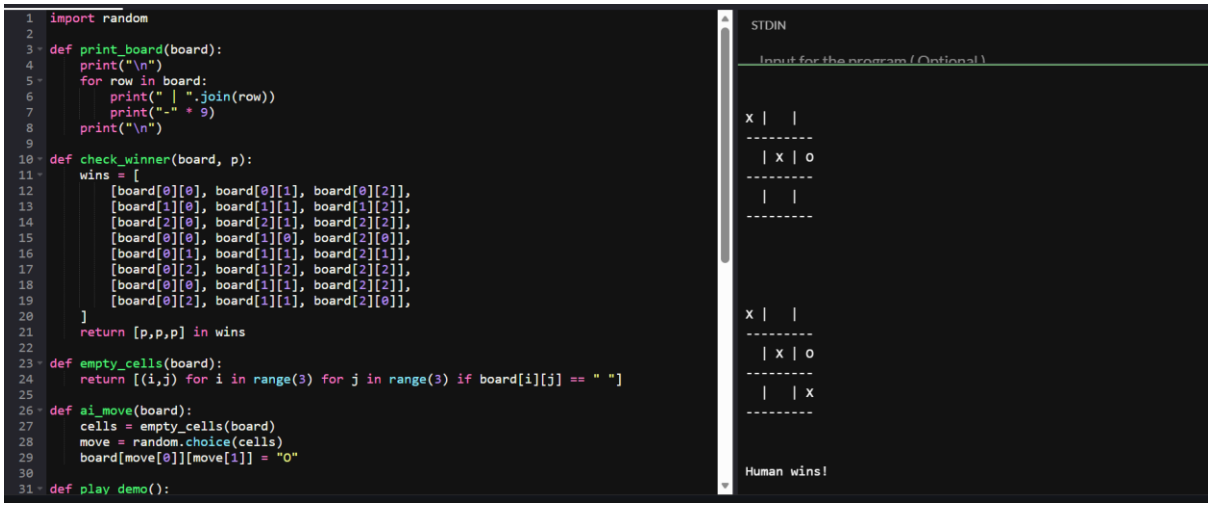
1. Implement a Python program for a Checkers AI agent using the Minimax algorithm with Alpha–Beta pruning. The program should:

   a) Represent the Checkers board and its legal moves.

   b) Allow two players (Human vs AI) or (AI vs AI) to play.
   c) Use Minimax with Alpha–Beta pruning for the AI's decision-making.
   d) Display the board after every move and declare the winner at the end.



2. Implement a Python program to simulate the Tic Tac Toe game. The program should:
   a) Represent the 3×3 game board.
   b) Allow a human player to play against an AI player.
   c) Print the board after each move.
   d) Declare the winner or draw at the end of the game



```python
import random

def print_board(board):
    print("\n")
    for row in board:
        print(" | ".join(row))
        print("-" * 9)
    print("\n")

def check_winner(board, p):
    wins = [
        [board[0][0], board[0][1], board[0][2]],
        [board[1][0], board[1][1], board[1][2]],
        [board[2][0], board[2][1], board[2][2]],
        [board[0][0], board[1][0], board[2][0]],
        [board[0][1], board[1][1], board[2][1]],
        [board[0][2], board[1][2], board[2][2]],
        [board[0][0], board[1][1], board[2][2]],
        [board[0][2], board[1][1], board[2][0]],
    ]
    return [p,p,p] in wins

def empty_cells(board):
    return [(i,j) for i in range(3) for j in range(3) if board[i][j] == " "]

def ai_move(board):
    cells = empty_cells(board)
    move = random.choice(cells)
    board[move[0]][move[1]] = "O"

def play_demo():
```

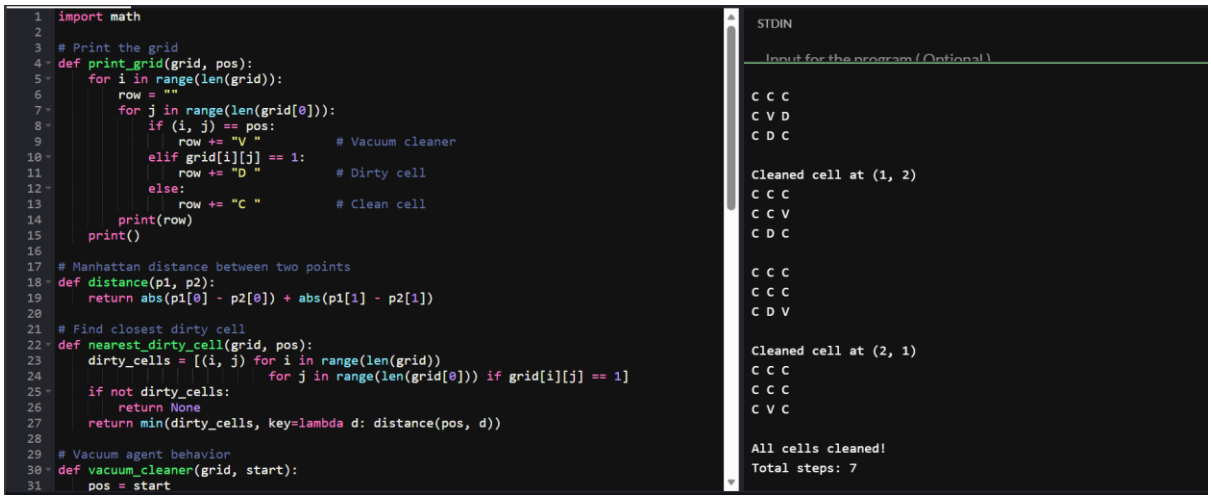
```
STDIN
Input for the program ( Optional )

x |   |
---------
  | x | o
---------
  |   |
---------

x |   |
---------
  | x | o
---------
  |   | x
---------

Human wins!
```

3. Imagine you have a simple autonomous vacuum cleaner, which can move around in a grid-based environment. The grid is divided into cells, some of which are dirty, and some are clean. The vacuum cleaner can move in four directions: up, down, left, and right. Its primary task is to clean all the dirty cells while minimizing its movements. Implement the above AI problem using python.



```python
import math

# Print the grid
def print_grid(grid, pos):
    for i in range(len(grid)):
        row = ""
        for j in range(len(grid[0])):
            if (i, j) == pos:
                row += "V "          # Vacuum cleaner
            elif grid[i][j] == 1:
                row += "D "          # Dirty cell
            else:
                row += "C "          # Clean cell
        print(row)
    print()

# Manhattan distance between two points
def distance(p1, p2):
    return abs(p1[0] - p2[0]) + abs(p1[1] - p2[1])

# Find closest dirty cell
def nearest_dirty_cell(grid, pos):
    dirty_cells = [(i, j) for i in range(len(grid))
                          for j in range(len(grid[0])) if grid[i][j] == 1]
    if not dirty_cells:
        return None
    return min(dirty_cells, key=lambda d: distance(pos, d))

# Vacuum agent behavior
def vacuum_cleaner(grid, start):
    pos = start
```

```
STDIN
Input for the program ( Optional )

C C C
C V D
C D C

Cleaned cell at (1, 2)
C C C
C C V
C D C

C C C
C C C
C D V

Cleaned cell at (2, 1)
C C C
C C C
C V C

All cells cleaned!
Total steps: 7
```