Jinesh · N
Roll No : 6 Topic :
Batch C2

Page No. :
Date. : / /

Assignment - 02 [AIES]

* Tittle :
Implementation of Minmax Algo for tic-tac toe game.

* Aim :
Solving tic tac toe game using minmax algo.

* Objective :
To study and implement minmax algo.

* Theory :

(1) Adversial search :
It is a search strategy used in decision making problems where multiple agents compete against each other.
It is commonly used in the field of game theory and AI.

(2) Tic Tac Toe solving steps : Game represn. initial state, move generation, evaluation of game states, minmax algo., alpha beta prunning.

Topic : _____ Page No. : ____ Date. : / /

3. Data Structures :

Key features
Game tree : nodes, edge
Board repres^n : A 2D array
Move list
Evaluate func^n

Key Concepts :

-> depth and pruning
-> optimal strategy
-> utility values.

* Input :
Initial state

* Output :
Soln / goal state with optimal path

* Algorithm :
Minmax

* Platform :
Linux

Page No. :
Date. : / /

Topic :

* FAQ's :

Q.1. Compare informed search and adversial search.

→ Informed search :
- Utilises domain specific knowledge
- Goal oriented : Focuses on finding the shortest costly path to a goal state.
- Explores path in a state space.
- Heuristic : used to estimate cost to goal.
- Single agent problems.

Adversial search
- Represents all possible moves in the game from current game.
- Alpha beta pruning
- Multi agent problems
- Explore game trees.

Q.2. Explain alpha-beta pruning.

→ Alpha bet is an optimisn technique for the minmax algo used in adversial search scenarios.

Topic : _____ Page No. : ____ Date. : / /

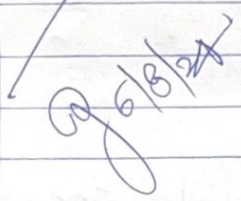
Key Concepts :

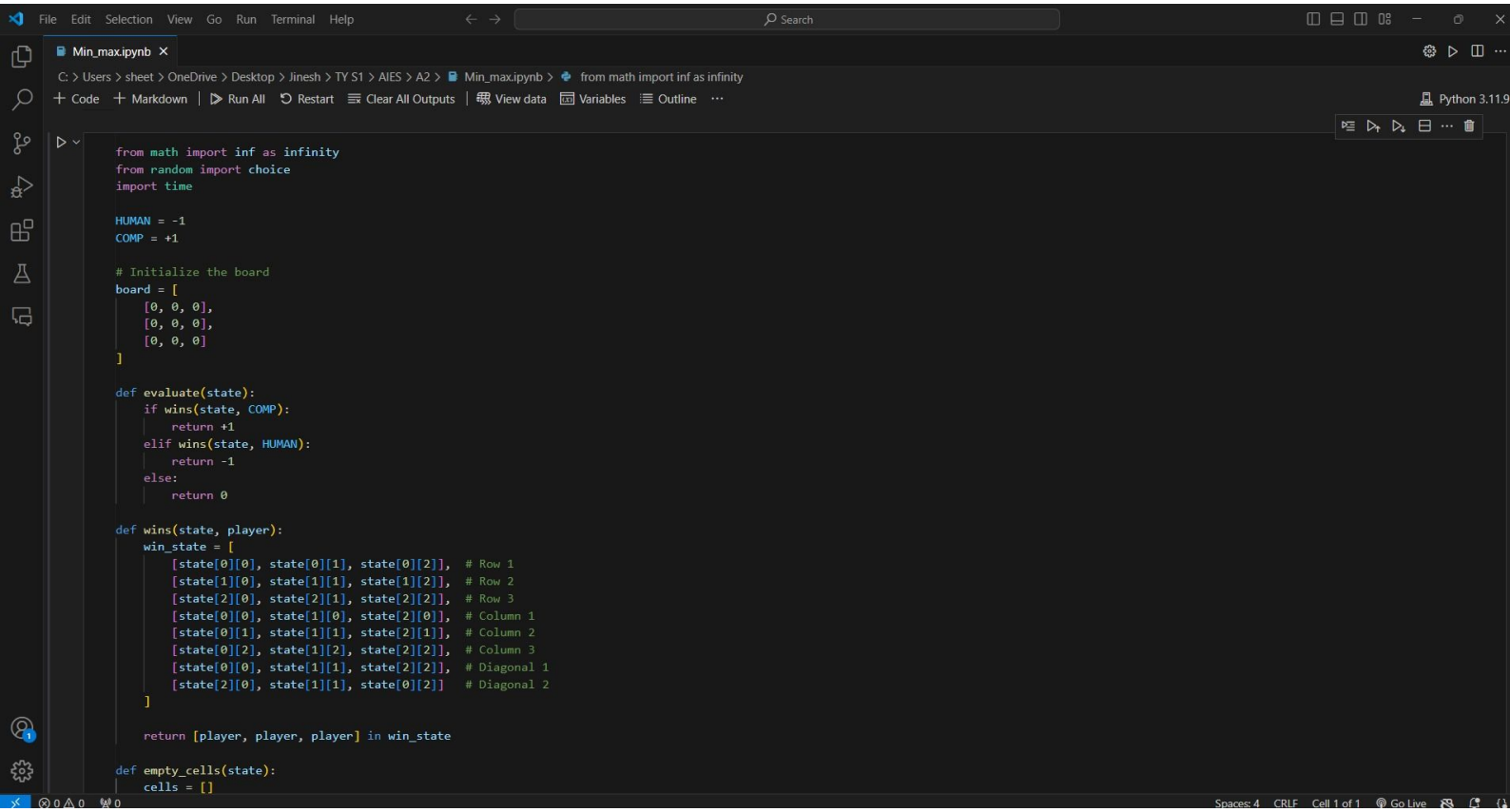
(1) Alpha
(2) Beta
(3) Pruning

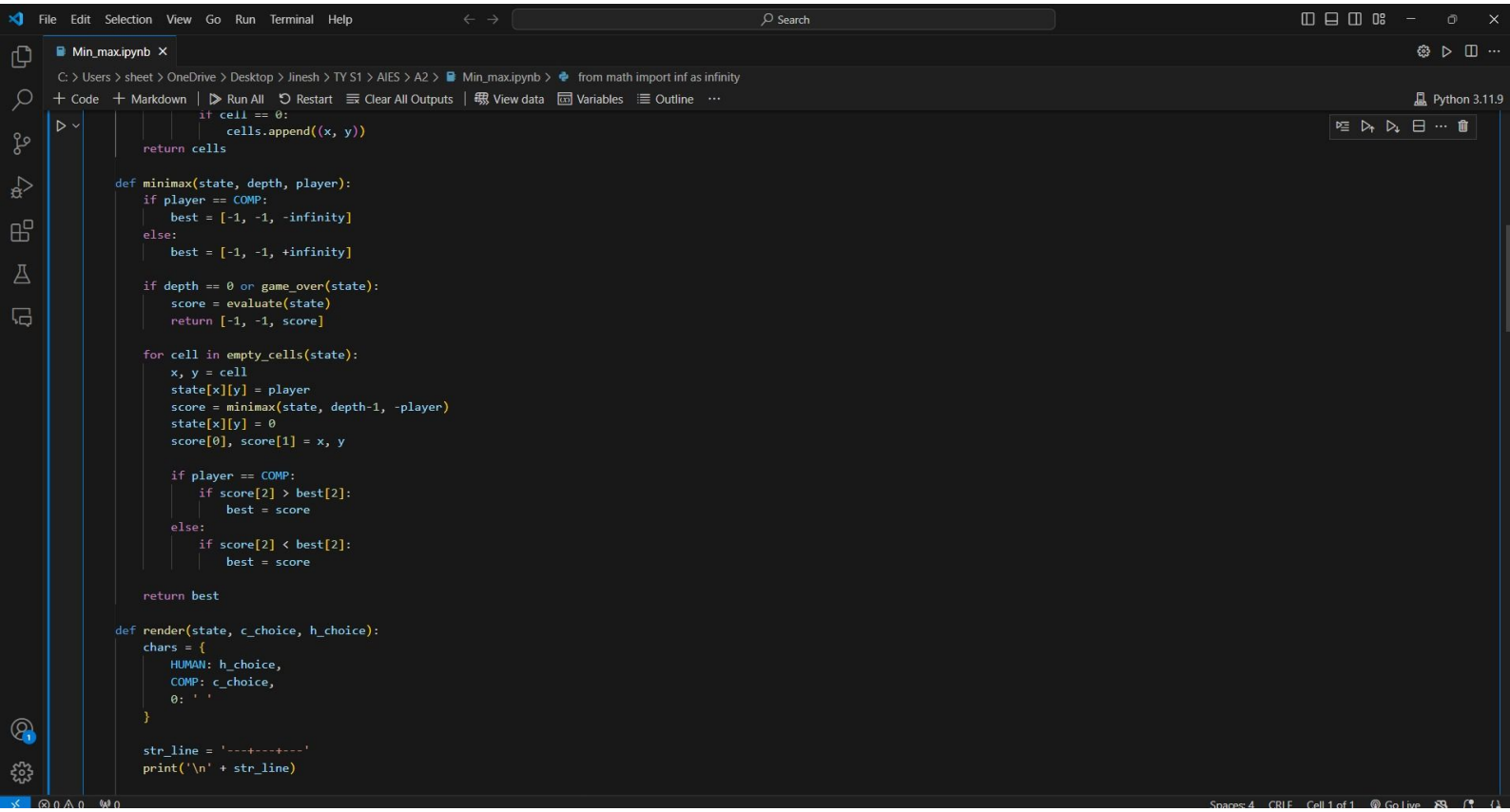
How it works :

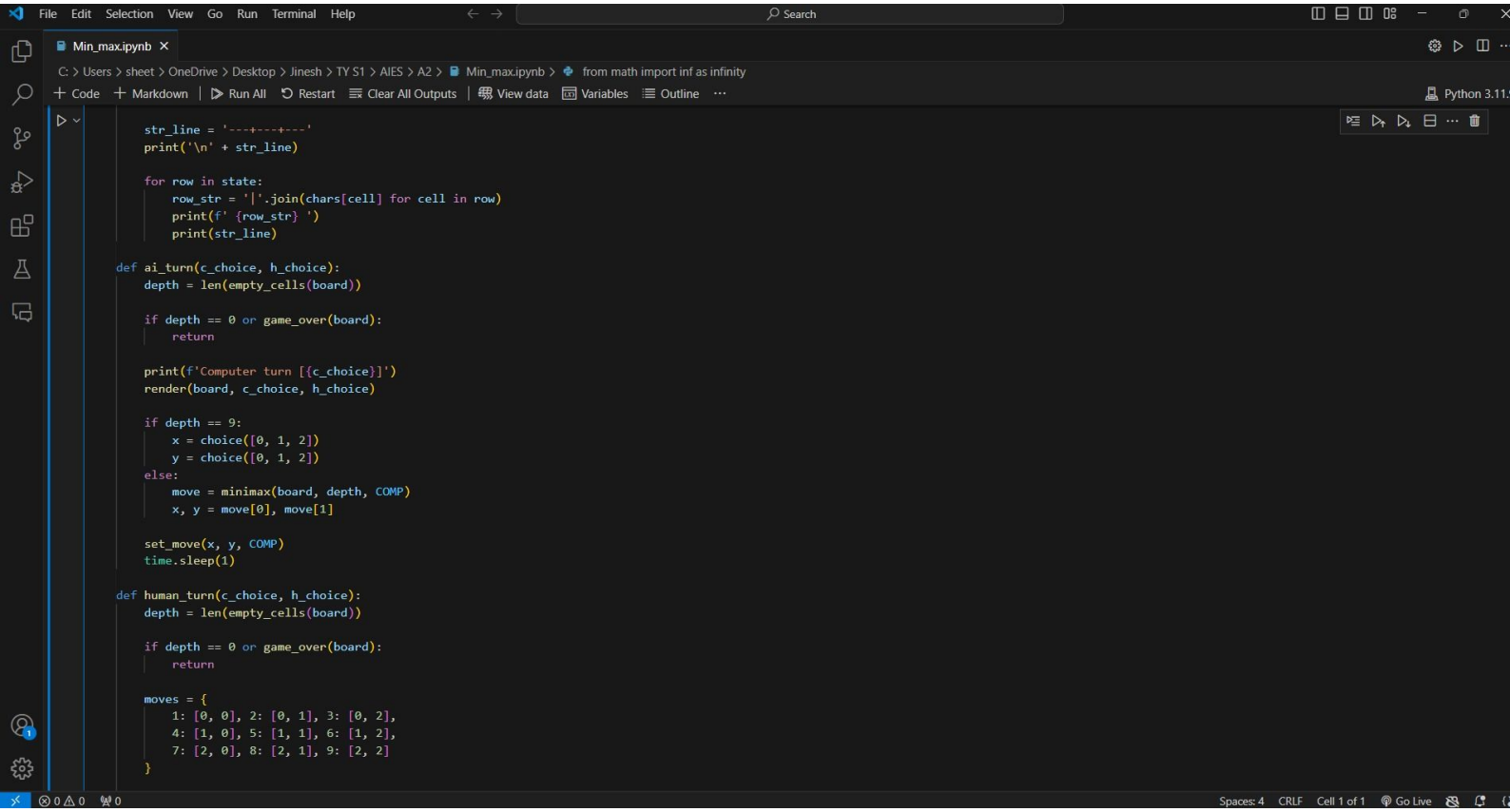
Initial call
recursive search
pruning search

Max Node




File Edit Selection View Go Run Terminal Help
Search
Min_max.ipynb
C: > Users > sheet > OneDrive > Desktop > Jinesh > TY S1 > AIES > A2 > Min_max.ipynb > from math import inf as infinity
Code Markdown Run All Restart Clear All Outputs View data Variables Outline
Python 3.11.9
from math import inf as infinity
from random import choice
import time
HUMAN = -1
COMP = +1
# Initialize the board
board = [
[0, 0, 0],
[0, 0, 0],
[0, 0, 0]
]
def evaluate(state):
if wins(state, COMP):
return +1
elif wins(state, HUMAN):
return -1
else:
return 0
def wins(state, player):
win_state = [
[state[0][0], state[0][1], state[0][2]], # Row 1
[state[1][0], state[1][1], state[1][2]], # Row 2
[state[2][0], state[2][1], state[2][2]], # Row 3
[state[0][0], state[1][0], state[2][0]], # Column 1
[state[0][1], state[1][1], state[2][1]], # Column 2
[state[0][2], state[1][2], state[2][2]], # Column 3
[state[0][0], state[1][1], state[2][2]], # Diagonal 1
[state[2][0], state[1][1], state[0][2]] # Diagonal 2
]
return [player, player, player] in win_state
def empty_cells(state):
cells = []
Spaces: 4 CRLF Cell 1 of 1 Go Live


File Edit Selection View Go Run Terminal Help
Search
Min_max.ipynb
C: > Users > sheet > OneDrive > Desktop > Jinesh > TY S1 > AIES > A2 > Min_max.ipynb > from math import inf as infinity
Code Markdown Run All Restart Clear All Outputs View data Variables Outline
Python 3.11.9
if cell == 0:
cells.append((x, y))
return cells
def minimax(state, depth, player):
if player == COMP:
best = [-1, -1, -infinity]
else:
best = [-1, -1, +infinity]
if depth == 0 or game_over(state):
score = evaluate(state)
return [-1, -1, score]
for cell in empty_cells(state):
x, y = cell
state[x][y] = player
score = minimax(state, depth-1, -player)
state[x][y] = 0
score[0], score[1] = x, y
if player == COMP:
if score[2] > best[2]:
best = score
else:
if score[2] < best[2]:
best = score
return best
def render(state, c_choice, h_choice):
chars = {
HUMAN: h_choice,
COMP: c_choice,
0: ' '
}
str_line = '---+---+---'
print('\n' + str_line)
Spaces: 4 CRLF Cell 1 of 1 Go Live


File Edit Selection View Go Run Terminal Help
Search
Min_max.ipynb
C: > Users > sheet > OneDrive > Desktop > Jinesh > TY S1 > AIES > A2 > Min_max.ipynb > from math import inf as infinity
Code Markdown Run All Restart Clear All Outputs View data Variables Outline
Python 3.11
str_line = '---+---+---'
print('\n' + str_line)
for row in state:
row_str = '|'.join(chars[cell] for cell in row)
print(f' {row_str} ')
print(str_line)
def ai_turn(c_choice, h_choice):
depth = len(empty_cells(board))
if depth == 0 or game_over(board):
return
print(f'Computer turn [{c_choice}]')
render(board, c_choice, h_choice)
if depth == 9:
x = choice([0, 1, 2])
y = choice([0, 1, 2])
else:
move = minimax(board, depth, COMP)
x, y = move[0], move[1]
set_move(x, y, COMP)
time.sleep(1)
def human_turn(c_choice, h_choice):
depth = len(empty_cells(board))
if depth == 0 or game_over(board):
return
moves = {
1: [0, 0], 2: [0, 1], 3: [0, 2],
4: [1, 0], 5: [1, 1], 6: [1, 2],
7: [2, 0], 8: [2, 1], 9: [2, 2]
}
Spaces: 4 CRLF Cell 1 of 1 Go Live

File Edit Selection View Go Run Terminal Help

Search

Min_max.ipynb

C: > Users > sheet > OneDrive > Desktop > Jinesh > TY S1 > AIES > A2 > Min_max.ipynb > from math import inf as infinity

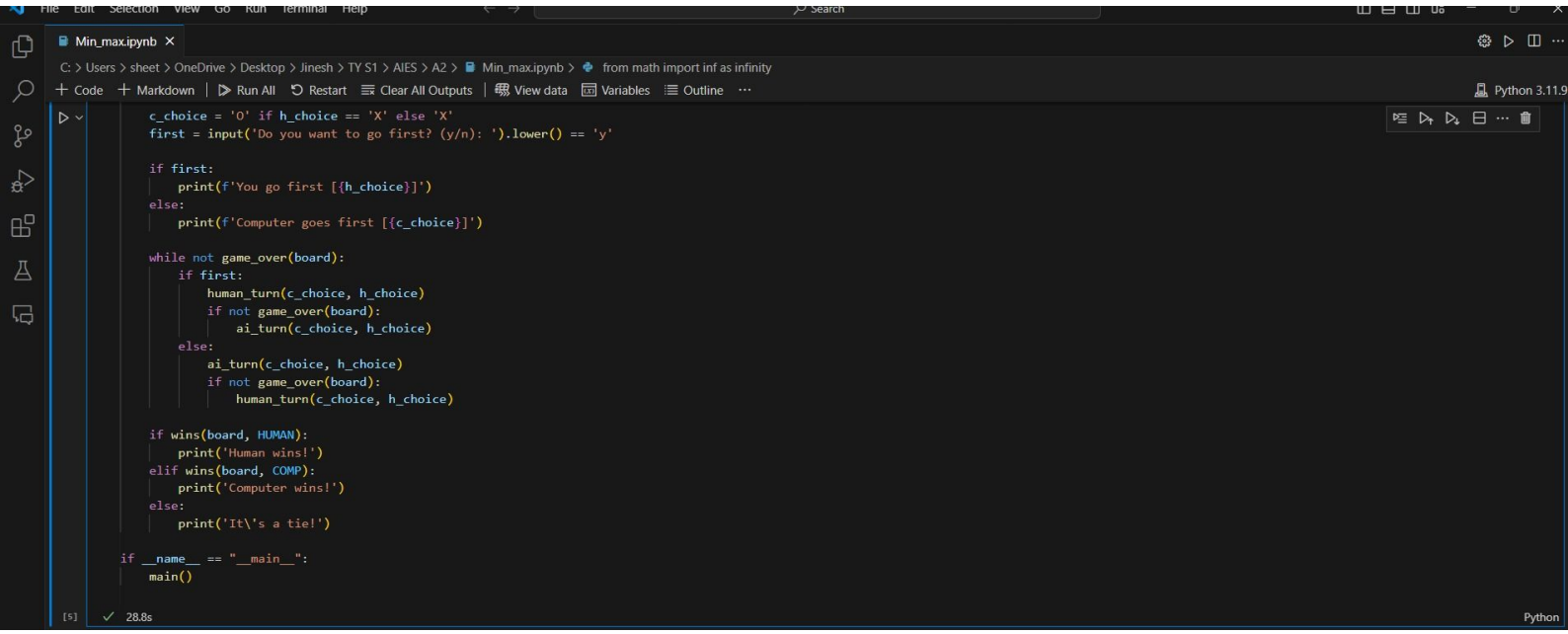
+ Code + Markdown | Run All Restart Clear All Outputs | View data Variables Outline ...

Python 3.11.9

```
        print(f'Human turn [{h_choice}]')
        render(board, c_choice, h_choice)

        move = -1
        while move < 1 or move > 9:
            try:
                move = int(input('Use numpad (1..9): '))
                coord = moves[move]
                if set_move(coord[0], coord[1], HUMAN):
                    break
                else:
                    print('Bad move')
                    move = -1
            except (EOFError, KeyboardInterrupt):
                print('Bye')
                exit()
            except (KeyError, ValueError):
                print('Bad choice')

    def set_move(x, y, player):
        if board[x][y] == 0:
            board[x][y] = player
            return True
        return False

    def game_over(state):
        return wins(state, HUMAN) or wins(state, COMP) or len(empty_cells(state)) == 0

    def main():
        # Customize your game start here
        h_choice = input('Choose your symbol (X or O): ').upper()
        while h_choice not in ['X', 'O']:
            print('Invalid choice. Choose X or O.')
            h_choice = input('Choose your symbol (X or O): ').upper()

        c_choice = 'O' if h_choice == 'X' else 'X'
        first = input('Do you want to go first? (y/n): ').lower() == 'y'

        if first:
```

0 0 0

Spaces: 4 CRLF Cell 1 of 1 Go Live


File Edit Selection View Go Run Terminal Help
Search
Min_max.ipynb
C: > Users > sheet > OneDrive > Desktop > Jinesh > TY S1 > AIES > A2 > Min_max.ipynb > from math import inf as infinity
Code Markdown Run All Restart Clear All Outputs View data Variables Outline
Python 3.11.9
c_choice = 'O' if h_choice == 'X' else 'X'
first = input('Do you want to go first? (y/n): ').lower() == 'y'
if first:
print(f'You go first [{h_choice}]')
else:
print(f'Computer goes first [{c_choice}]')
while not game_over(board):
if first:
human_turn(c_choice, h_choice)
if not game_over(board):
ai_turn(c_choice, h_choice)
else:
ai_turn(c_choice, h_choice)
if not game_over(board):
human_turn(c_choice, h_choice)
if wins(board, HUMAN):
print('Human wins!')
elif wins(board, COMP):
print('Computer wins!')
else:
print('It\'s a tie!')
if __name__ == "__main__":
main()
[5]
28.8s
Python