Google Account | New announcement | Dear all, Lab program | Week-3-AI Lab.pptx | van37 | Dontpad | Intro | A* algorithm pseudo | 1BM23CS253_AI/w1 | Convert JPG to PDF

jupyter.org/try-jupyter/notebooks/?path=notebooks/Intro.ipynb

Guest

JupyterLite Intro Last Checkpoint: 2 months ago

File Edit View Run Kernel Settings Help

Trusted

Code

JupyterLab | Python (Pyodide)

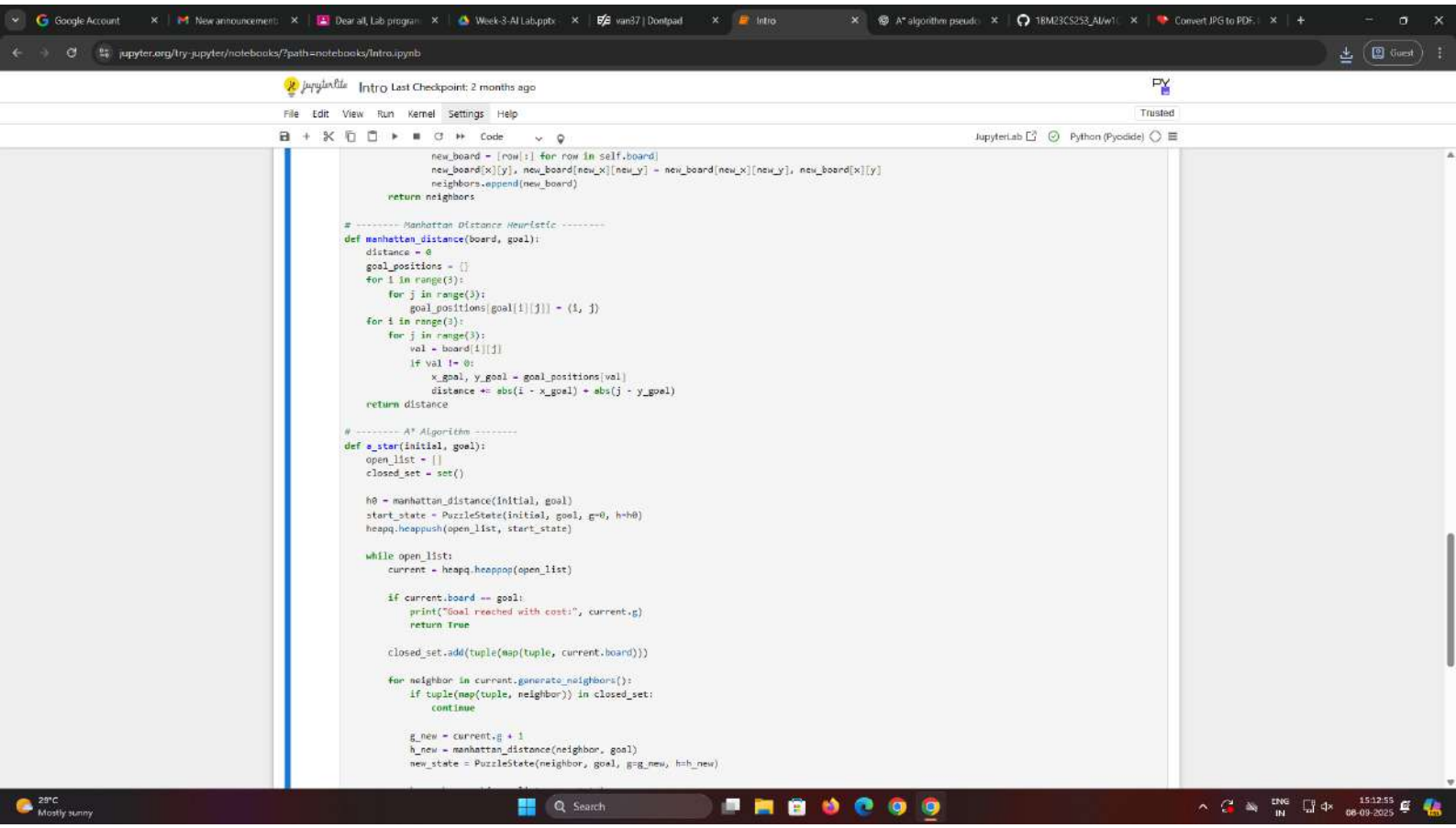
[2]:

```
import heapq

class PuzzleState:
    def __init__(self, board, goal, g=0, h=0):
        self.board = board
        self.goal = goal
        self.g = g
        self.h = h
        self.f = g + h

    def __lt__(self, other):
        return self.f < other.f

    def __eq__(self, other):
        return self.board == other.board

    def __hash__(self):
        return hash(str(self.board))

    def get_blank_pos(self):
        for i in range(3):
            for j in range(3):
                if self.board[i][j] == 0:
                    return i, j

    def generate_neighbors(self):
        neighbors = []
        x, y = self.get_blank_pos()
        moves = [(0, 1), (0, -1), (1, 0), (-1, 0)]
        for dx, dy in moves:
            new_x, new_y = x + dx, y + dy
            if 0 <= new_x < 3 and 0 <= new_y < 3:
                new_board = [row[:] for row in self.board]
                new_board[x][y], new_board[new_x][new_y] = new_board[new_x][new_y], new_board[x][y]
                neighbors.append(new_board)
        return neighbors

# -------- Manhattan Distance Heuristic --------
def manhattan_distance(board, goal):
    distance = 0
    goal_positions = {}
    for i in range(3):
        for j in range(3):
            goal_positions[goal[i][j]] = (i, j)
    for i in range(3):
```

28°C Mostly sunny

Search

ENG IN 15:12:50 08-09-2025


Intro Last Checkpoint: 2 months ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python (Pyodide)
new_board = [row[:] for row in self.board]
new_board[x][y], new_board[new_x][new_y] = new_board[new_x][new_y], new_board[x][y]
neighbors.append(new_board)
return neighbors
# -------- Manhattan Distance Heuristic --------
def manhattan_distance(board, goal):
distance = 0
goal_positions = {}
for i in range(3):
for j in range(3):
goal_positions[goal[i][j]] = (i, j)
for i in range(3):
for j in range(3):
val = board[i][j]
if val != 0:
x_goal, y_goal = goal_positions[val]
distance += abs(i - x_goal) + abs(j - y_goal)
return distance
# -------- A* Algorithm --------
def a_star(initial, goal):
open_list = []
closed_set = set()
h0 = manhattan_distance(initial, goal)
start_state = PuzzleState(initial, goal, g=0, h=h0)
heapq.heappush(open_list, start_state)
while open_list:
current = heapq.heappop(open_list)
if current.board == goal:
print("Goal reached with cost:", current.g)
return True
closed_set.add(tuple(map(tuple, current.board)))
for neighbor in current.generate_neighbors():
if tuple(map(tuple, neighbor)) in closed_set:
continue
g_new = current.g + 1
h_new = manhattan_distance(neighbor, goal)
new_state = PuzzleState(neighbor, goal, g=g_new, h=h_new)
28°C
Mostly sunny
Search
ENG
IN
15:12:55
08-09-2025

Google Account | New announcement | Dear all, Lab program | Week-3-AI Lab.pptx | van37 | Dontpad | Intro | A* algorithm pseudo | 1BM23CS253_AI/w1 | Convert JPG to PDF

jupyter.org/try-jupyter/notebooks/?path=notebooks/Intro.ipynb

Guest

JupyterLite Intro Last Checkpoint: 2 months ago

File Edit View Run Kernel Settings Help

Trusted

Code

JupyterLab | Python (Pyodide)

```
            if current.board == goal:
                print("Goal reached with cost:", current.g)
                return True

            closed_set.add(tuple(map(tuple, current.board)))

            for neighbor in current.generate_neighbors():
                if tuple(map(tuple, neighbor)) in closed_set:
                    continue

                g_new = current.g + 1
                h_new = manhattan_distance(neighbor, goal)
                new_state = PuzzleState(neighbor, goal, g=g_new, h=h_new)

                heapq.heappush(open_list, new_state)

        return False

    # -------- Example Run --------
    initial_state = [
        [1, 2, 3],
        [4, 0, 6],
        [7, 5, 8]
    ]

    goal_state = [
        [1, 2, 3],
        [4, 5, 6],
        [7, 8, 0]
    ]

    print("A* with Manhattan Distance Heuristic")
    a_star(initial_state, goal_state)
```

```
A* with Manhattan Distance Heuristic
Goal reached with cost: 2
```

[2]: True

28°C Mostly sunny

Search

ENG IN 15:12:59 08-09-2025