

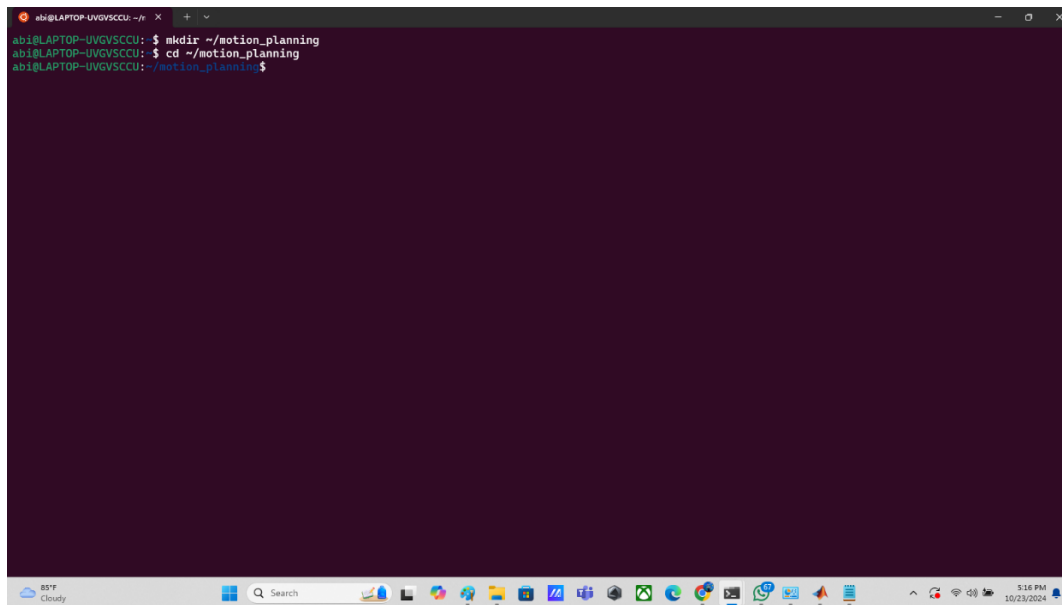
Nama : Ketut Satria Wibisana

NIM : 1103213148

Kelas : TK-45-G09

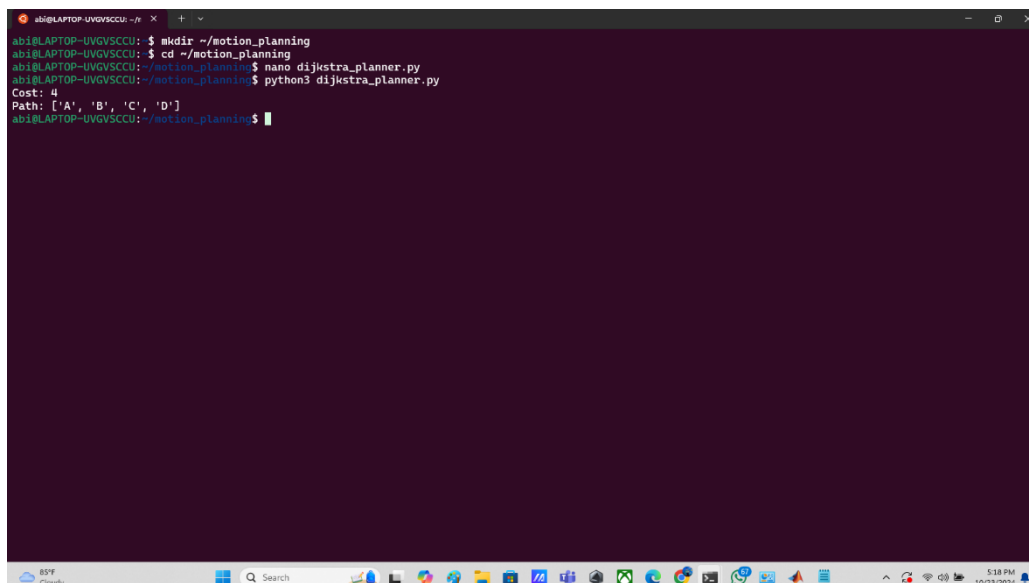
### Tiga Algoritma Perencanaan Jalur

1. Jalankan syntax **mkdir ~/motion\_planning** untuk membuat direktori baru dan masukkan syntax **cd ~/motion\_planning** untuk masuk ke dalam direktori yang baru dibuat.

A terminal window with a dark purple background. The prompt is 'abi@LAPTOP-UVGVSCCU: ~'. The user enters 'mkdir ~/motion\_planning', and the prompt changes to 'abi@LAPTOP-UVGVSCCU: ~/motion\_planning'. The user then enters 'cd ~/motion\_planning', and the prompt changes to 'abi@LAPTOP-UVGVSCCU: ~/motion\_planning\$'. The terminal window is titled 'abi@LAPTOP-UVGVSCCU: ~'. The taskbar at the bottom shows various application icons and the system clock indicating 5:18 PM on 10/23/2024.

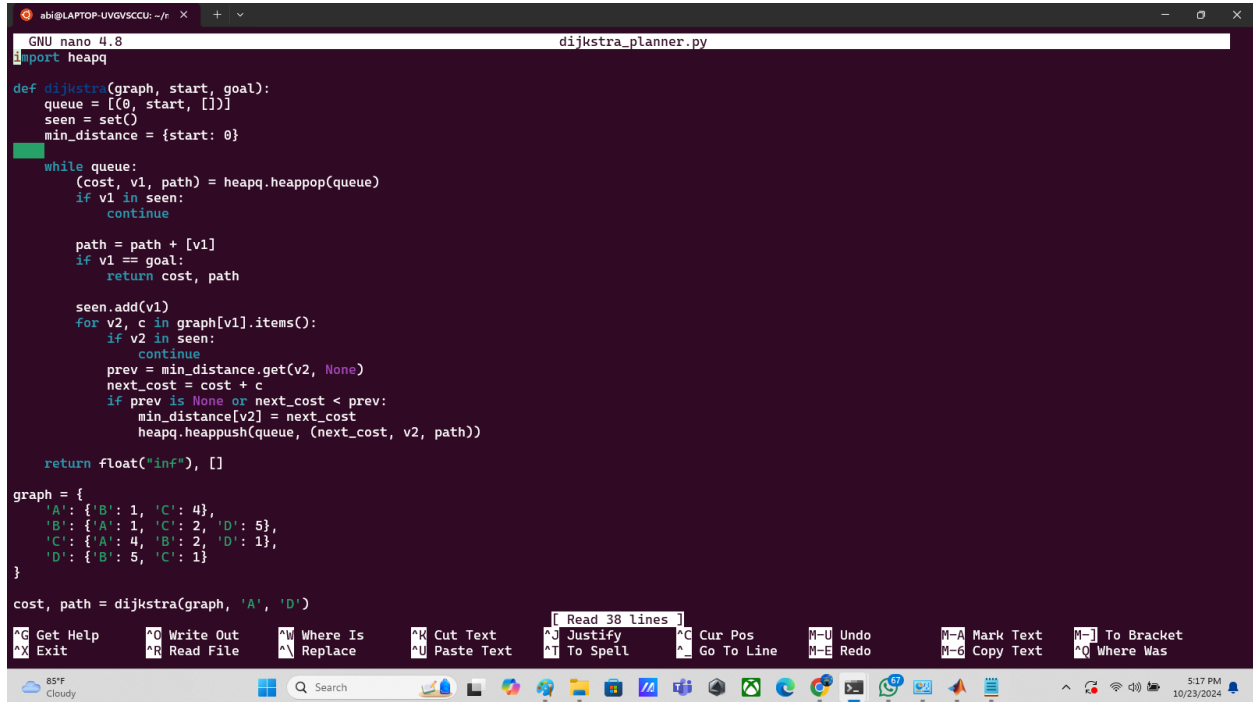
```
abi@LAPTOP-UVGVSCCU: ~  
abi@LAPTOP-UVGVSCCU:~$ mkdir ~/motion_planning  
abi@LAPTOP-UVGVSCCU:~$ cd ~/motion_planning  
abi@LAPTOP-UVGVSCCU:~/motion_planning$
```

2. Jalankan syntax **nano dijkstra\_planner.py** untuk membuat file python algoritma dijkstra.

A terminal window with a dark purple background. The prompt is 'abi@LAPTOP-UVGVSCCU: ~/motion\_planning'. The user enters 'nano dijkstra\_planner.py', and the prompt changes to 'abi@LAPTOP-UVGVSCCU: ~/motion\_planning\$ nano dijkstra\_planner.py'. The user then enters 'python3 dijkstra\_planner.py', and the output is 'Cost: 4' and 'Path: ['A', 'B', 'C', 'D']'. The prompt returns to 'abi@LAPTOP-UVGVSCCU: ~/motion\_planning\$'. The terminal window is titled 'abi@LAPTOP-UVGVSCCU: ~/motion\_planning'. The taskbar at the bottom shows various application icons and the system clock indicating 5:18 PM on 10/23/2024.

```
abi@LAPTOP-UVGVSCCU: ~/motion_planning  
abi@LAPTOP-UVGVSCCU:~/motion_planning$ nano dijkstra_planner.py  
abi@LAPTOP-UVGVSCCU:~/motion_planning$ python3 dijkstra_planner.py  
Cost: 4  
Path: ['A', 'B', 'C', 'D']  
abi@LAPTOP-UVGVSCCU:~/motion_planning$
```

3. Buat kode python untuk menjalankan algoritma dijkstra lalu save



```
GNU nano 4.8 dijkstra_planner.py
import heapq

def dijkstra(graph, start, goal):
    queue = [(0, start, [])]
    seen = set()
    min_distance = {start: 0}

    while queue:
        (cost, v1, path) = heapq.heappop(queue)
        if v1 in seen:
            continue

        path = path + [v1]
        if v1 == goal:
            return cost, path

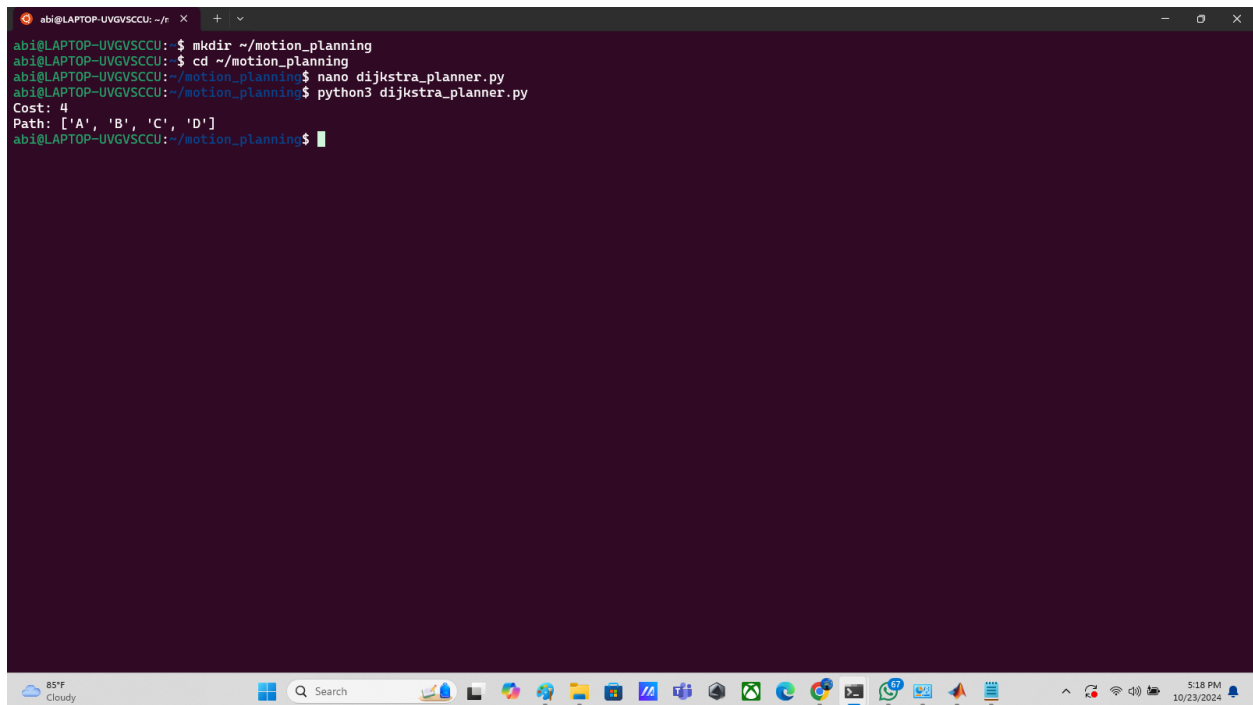
        seen.add(v1)
        for v2, c in graph[v1].items():
            if v2 in seen:
                continue
            prev = min_distance.get(v2, None)
            next_cost = cost + c
            if prev is None or next_cost < prev:
                min_distance[v2] = next_cost
                heapq.heappush(queue, (next_cost, v2, path))

    return float("inf"), []

graph = {
    'A': {'B': 1, 'C': 4},
    'B': {'A': 1, 'C': 2, 'D': 5},
    'C': {'A': 4, 'B': 2, 'D': 1},
    'D': {'B': 5, 'C': 1}
}

cost, path = dijkstra(graph, 'A', 'D')
```

4. Jalankan syntax **python3 dijkstra\_planner.py** untuk menjalankan kodenya.



```
abi@LAPTOP-UVGVSCCU: ~/motion_planning
abi@LAPTOP-UVGVSCCU:~/motion_planning$ python3 dijkstra_planner.py
Cost: 4
Path: ['A', 'B', 'C', 'D']
abi@LAPTOP-UVGVSCCU:~/motion_planning$
```

5. Jalankan syntax **nano a\_star\_planner.py** untuk membuat file python algoritma a\*.

```
abi@LAPTOP-UVGVSCCU: ~/motion_planning
abi@LAPTOP-UVGVSCCU:~/motion_planning$ mkdir ~/motion_planning
abi@LAPTOP-UVGVSCCU:~/motion_planning$ cd ~/motion_planning
abi@LAPTOP-UVGVSCCU:~/motion_planning$ nano dijkstra_planner.py
abi@LAPTOP-UVGVSCCU:~/motion_planning$ python3 dijkstra_planner.py
Cost: 4
Path: ['A', 'B', 'C', 'D']
abi@LAPTOP-UVGVSCCU:~/motion_planning$ nano a_star_planner.py
abi@LAPTOP-UVGVSCCU:~/motion_planning$ python3 a_star_planner.py
Path: [(0, 0), (1, 0), (2, 0), (2, 1), (3, 1), (3, 2), (3, 3)]
abi@LAPTOP-UVGVSCCU:~/motion_planning$
```

6. Buat kode python untuk menjalankan algoritma a\* lalu save.

```
GNU nano 4.8 a_star_planner.py
def get_neighbors(node, grid):
    neighbors = []
    directions = [(-1, 0), (1, 0), (0, -1), (0, 1)]
    for d in directions:
        neighbor = (node[0] + d[0], node[1] + d[1])
        if 0 <= neighbor[0] < len(grid) and 0 <= neighbor[1] < len(grid[0]) and grid[neighbor[0]][neighbor[1]] == 0:
            neighbors.append(neighbor)
    return neighbors

def a_star(start, goal, grid):
    open_list = []
    closed_list = set()
    open_list.append(start)

    g = {start: 0}
    f = {start: heuristic(start, goal)}

    came_from = {}

    while open_list:
        current = min(open_list, key=lambda x: f[x])

        if current == goal:
            path = []
            while current in came_from:
                path.append(current)
                current = came_from[current]
            path.append(start)
            return path[::-1]

        open_list.remove(current)
        closed_list.add(current)

        for neighbor in get_neighbors(current, grid):
            if neighbor in closed_list:
                continue
```

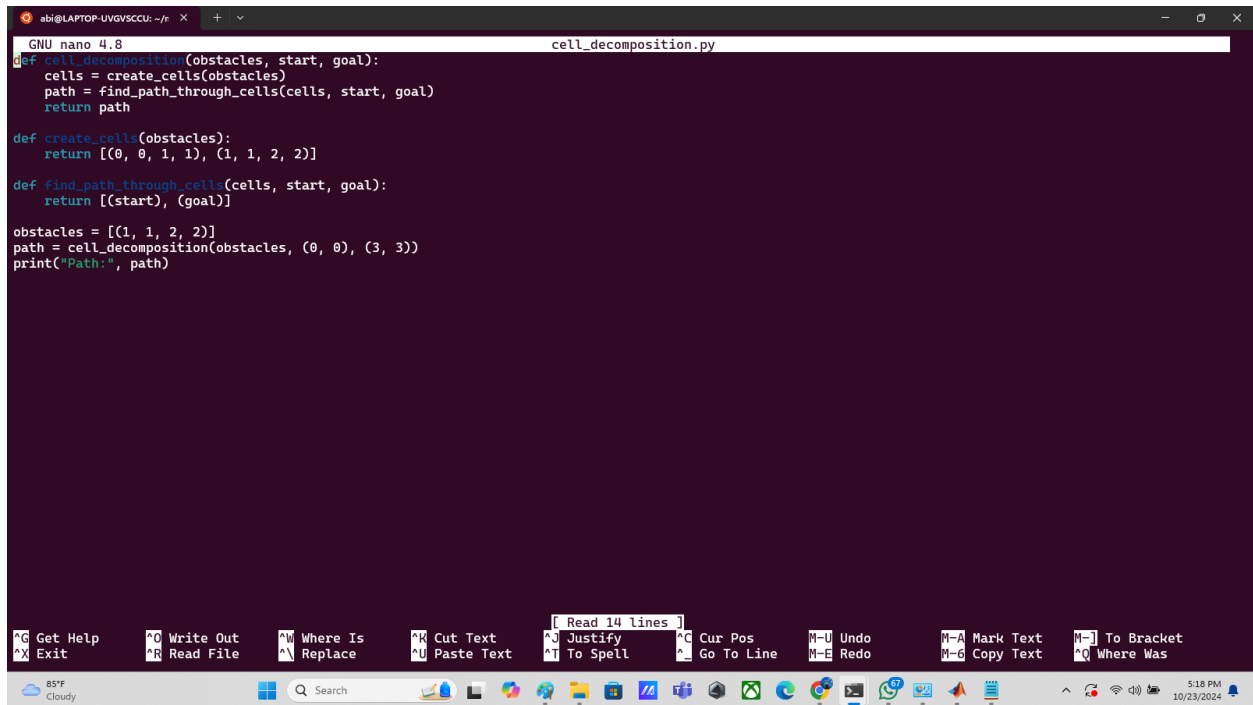
7. Jalankan syntax **python3a\_star\_planner.py** untuk menjalankan kodenya.

```
abi@LAPTOP-UVGVSCCU: ~/motion_planning
abi@LAPTOP-UVGVSCCU:~/motion_planning$ mkdir ~/motion_planning
abi@LAPTOP-UVGVSCCU:~/motion_planning$ cd ~/motion_planning
abi@LAPTOP-UVGVSCCU:~/motion_planning$ nano dijkstra_planner.py
abi@LAPTOP-UVGVSCCU:~/motion_planning$ python3 dijkstra_planner.py
Cost: 4
Path: ['A', 'B', 'C', 'D']
abi@LAPTOP-UVGVSCCU:~/motion_planning$ nano a_star_planner.py
abi@LAPTOP-UVGVSCCU:~/motion_planning$ python3 a_star_planner.py
Path: [(0, 0), (1, 0), (2, 0), (2, 1), (3, 1), (3, 2), (3, 3)]
abi@LAPTOP-UVGVSCCU:~/motion_planning$
```

8. Jalankan syntax **nano cell\_descomposition.py** untuk membuat file python algoritma cell decomposition.

```
abi@LAPTOP-UVGVSCCU: ~/motion_planning
abi@LAPTOP-UVGVSCCU:~/motion_planning$ mkdir ~/motion_planning
abi@LAPTOP-UVGVSCCU:~/motion_planning$ cd ~/motion_planning
abi@LAPTOP-UVGVSCCU:~/motion_planning$ nano dijkstra_planner.py
abi@LAPTOP-UVGVSCCU:~/motion_planning$ python3 dijkstra_planner.py
Cost: 4
Path: ['A', 'B', 'C', 'D']
abi@LAPTOP-UVGVSCCU:~/motion_planning$ nano a_star_planner.py
abi@LAPTOP-UVGVSCCU:~/motion_planning$ python3 a_star_planner.py
Path: [(0, 0), (1, 0), (2, 0), (2, 1), (3, 1), (3, 2), (3, 3)]
abi@LAPTOP-UVGVSCCU:~/motion_planning$ nano cell_decomposition.py
abi@LAPTOP-UVGVSCCU:~/motion_planning$ python3 cell_decomposition.py
Path: [(0, 0), (3, 3)]
abi@LAPTOP-UVGVSCCU:~/motion_planning$
```

9. Buat kode python untuk menjalankan algoritma cell decomposition lalu save.



```
GNU nano 4.8 cell_decomposition.py
def cell_decomposition(obstacles, start, goal):
    cells = create_cells(obstacles)
    path = find_path_through_cells(cells, start, goal)
    return path

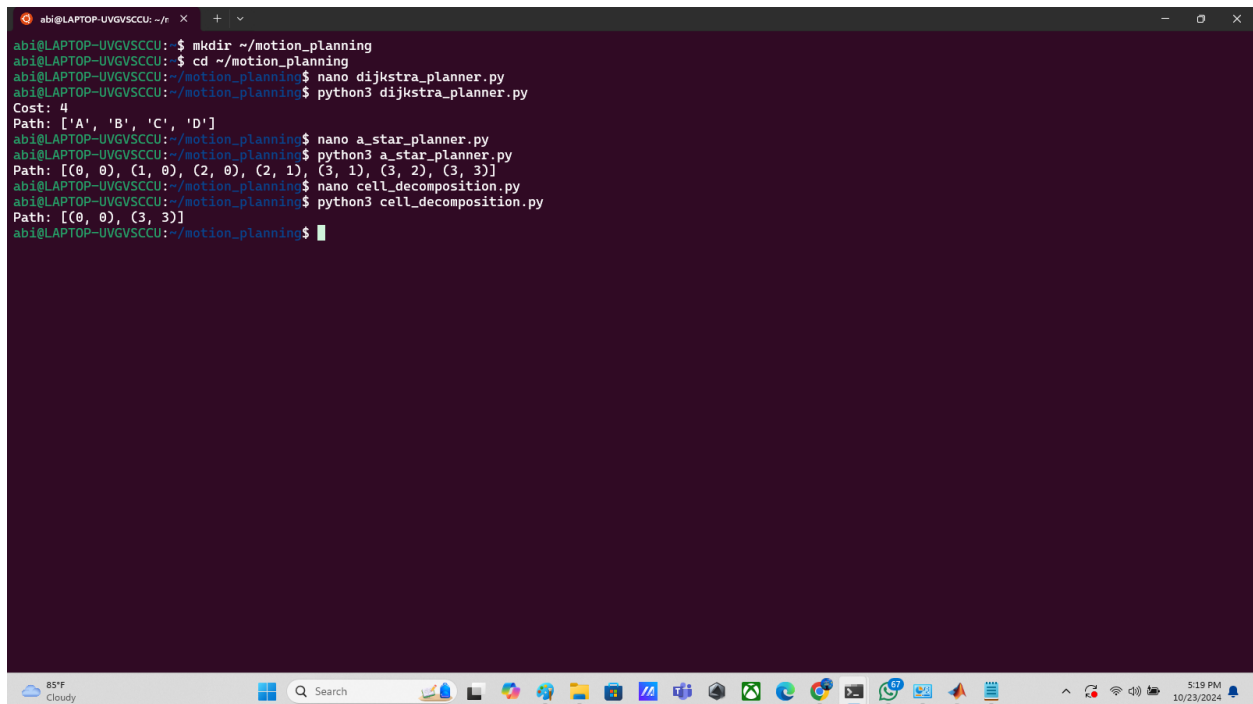
def create_cells(obstacles):
    return [(0, 0, 1, 1), (1, 1, 2, 2)]

def find_path_through_cells(cells, start, goal):
    return [(start), (goal)]

obstacles = [(1, 1, 2, 2)]
path = cell_decomposition(obstacles, (0, 0), (3, 3))
print("Path:", path)
```

The screenshot shows a terminal window with the GNU nano 4.8 editor. The file being edited is cell\_decomposition.py. The code defines two functions: cell\_decomposition and create\_cells. The cell\_decomposition function calls create\_cells to generate a set of cells and then finds a path through them. The create\_cells function returns a list of cells. The main code block sets obstacles to [(1, 1, 2, 2)], calls cell\_decomposition with start (0, 0) and goal (3, 3), and prints the resulting path.

10. Jalankan syntax **python3 cell\_decomposition.py** untuk menjalankan kodenya.



```
abi@LAPTOP-UVGVSCCU: ~/motion_planning
abi@LAPTOP-UVGVSCCU:~/motion_planning$ mkdir ~/motion_planning
abi@LAPTOP-UVGVSCCU:~/motion_planning$ cd ~/motion_planning
abi@LAPTOP-UVGVSCCU:~/motion_planning$ nano dijkstra_planner.py
abi@LAPTOP-UVGVSCCU:~/motion_planning$ python3 dijkstra_planner.py
Cost: 4
Path: ['A', 'B', 'C', 'D']
abi@LAPTOP-UVGVSCCU:~/motion_planning$ nano a_star_planner.py
abi@LAPTOP-UVGVSCCU:~/motion_planning$ python3 a_star_planner.py
Path: [(0, 0), (1, 0), (2, 0), (2, 1), (3, 1), (3, 2), (3, 3)]
abi@LAPTOP-UVGVSCCU:~/motion_planning$ nano cell_decomposition.py
abi@LAPTOP-UVGVSCCU:~/motion_planning$ python3 cell_decomposition.py
Path: [(0, 0), (3, 3)]
abi@LAPTOP-UVGVSCCU:~/motion_planning$
```

The screenshot shows a terminal window with the following commands and output:

- `mkdir ~/motion_planning`
- `cd ~/motion_planning`
- `nano dijkstra_planner.py`
- `python3 dijkstra_planner.py` outputs: `Cost: 4` and `Path: ['A', 'B', 'C', 'D']`
- `nano a_star_planner.py`
- `python3 a_star_planner.py` outputs: `Path: [(0, 0), (1, 0), (2, 0), (2, 1), (3, 1), (3, 2), (3, 3)]`
- `nano cell_decomposition.py`
- `python3 cell_decomposition.py` outputs: `Path: [(0, 0), (3, 3)]`

