Write the python program to implement A* algorithm

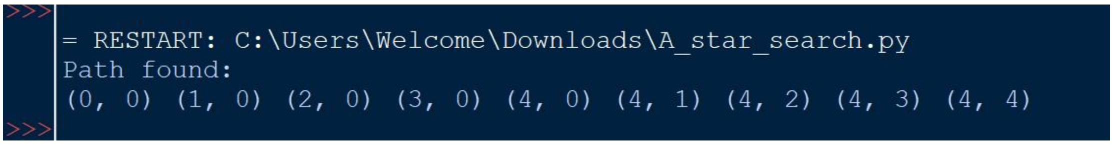
AIM:

To write a python code to implement A* algorithm.

PROGRAM:

```
import heapq

class Node:
   def __init__(self, x, y, obstacle=False):
      self.x = x
      self.y = y
      self.obstacle = obstacle
      self.g = float('inf')
      self.h = 0
      self.f = 0
      self.parent = None

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

def calculate_heuristic(current, goal):

   return abs(current.x - goal.x) + abs(current.y - goal.y)

def get_neighbors(grid, node):
   neighbors = []
   rows, cols = len(grid), len(grid[0])
   directions = [(1, 0), (-1, 0), (0, 1), (0, -1)]

   for dx, dy in directions:
      x, y = node.x + dx, node.y + dy
      if 0 <= x < rows and 0 <= y < cols and not grid[x][y].obstacle:
           neighbors.append(grid[x][y])

   return neighbors

def astar(grid, start, goal):
   open_set = []
   heapq.heappush(open_set, start)
   start.g = 0
   start.h = calculate_heuristic(start, goal)
   start.f = start.g + start.h

   while open_set:
      current = heapq.heappop(open_set)

      if current == goal:
           path = []
           while current:
               path.append((current.x, current.y))
               current = current.parent
           return path[::-1]

      for neighbor in get_neighbors(grid, current):
           tentative_g = current.g + 1
           if tentative_g < neighbor.g:
               neighbor.parent = current
               neighbor.g = tentative_g
               neighbor.h = calculate_heuristic(neighbor, goal)
               neighbor.f = neighbor.g + neighbor.h
               if neighbor not in open_set:
                   heapq.heappush(open_set, neighbor)

   return None

if __name__ == "__main__":
   grid = [[Node(x, y, obstacle=False) for y in range(5)] for x in range(5)]
   grid[1][2].obstacle = True
   grid[2][2].obstacle = True
   grid[3][2].obstacle = True


   start_node = grid[0][0]
   goal_node = grid[4][4]


   path = astar(grid, start_node, goal_node)

   if path:
      print("Path found:")
      for x, y in path:
           print(f"({x}, {y})", end=" ")
   else:
      print("No path found.")
```

OUTPUT:



```
>>>
= RESTART: C:\Users\Welcome\Downloads\A_star_search.py
Path found:
(0, 0) (1, 0) (2, 0) (3, 0) (4, 0) (4, 1) (4, 2) (4, 3) (4, 4)
>>>
```

RESULT:

The program was executed successfully and results was obtained.