# Explain the Code, How and Why

## Explain the Code

This Python program implements the A\* (A-star) search algorithm to find the shortest path between two nodes in a weighted graph. The program uses a class-based structure to organize the graph data and logic.

## How the Code Works

1. A class named 'Graph' is defined with an initializer (\_\_init\_\_) that takes a graph map (dictionary) representing nodes and edges.  
2. The 'neigh(x)' function returns the neighbors of a given node.  
3. The 'h(x)' function provides a heuristic (estimated cost to reach the goal). In this example, all nodes have a heuristic value of 1.  
4. The 'astar(s, t)' function performs the A\* algorithm:  
 • It maintains sets for open (nodes to explore) and closed (visited nodes).  
 • It calculates the total cost for each node using 'cost[node] + h(node)'.  
 • It repeatedly picks the node with the smallest total estimated cost.  
 • When the target node is reached, it reconstructs the path using the parent dictionary ('par').

## Why the Code is Written This Way

The code follows the A\* algorithm’s principle of combining actual and heuristic costs to efficiently find the optimal path. Using a class-based design makes the implementation clean, modular, and reusable. The use of sets and dictionaries ensures fast lookup and path tracking. This approach is ideal for AI pathfinding problems in graphs or maps where both distance and estimated cost are important.

## Screenshot of output:

