A* IMPLEMENTATION

FINDING SHORTEST PATH

FREDERICK YONATAN / 5025211121

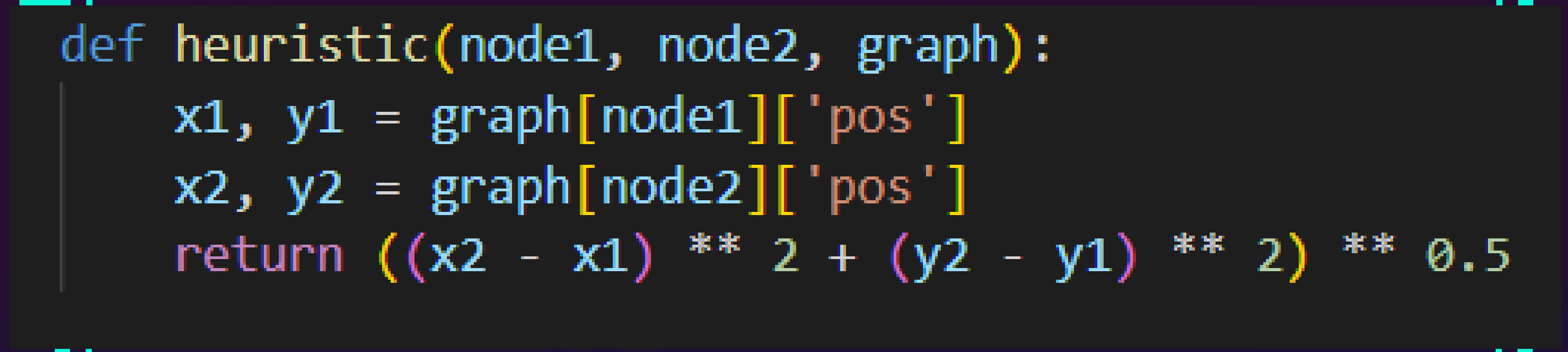
START NOW

GAME START

PLAY

MENU

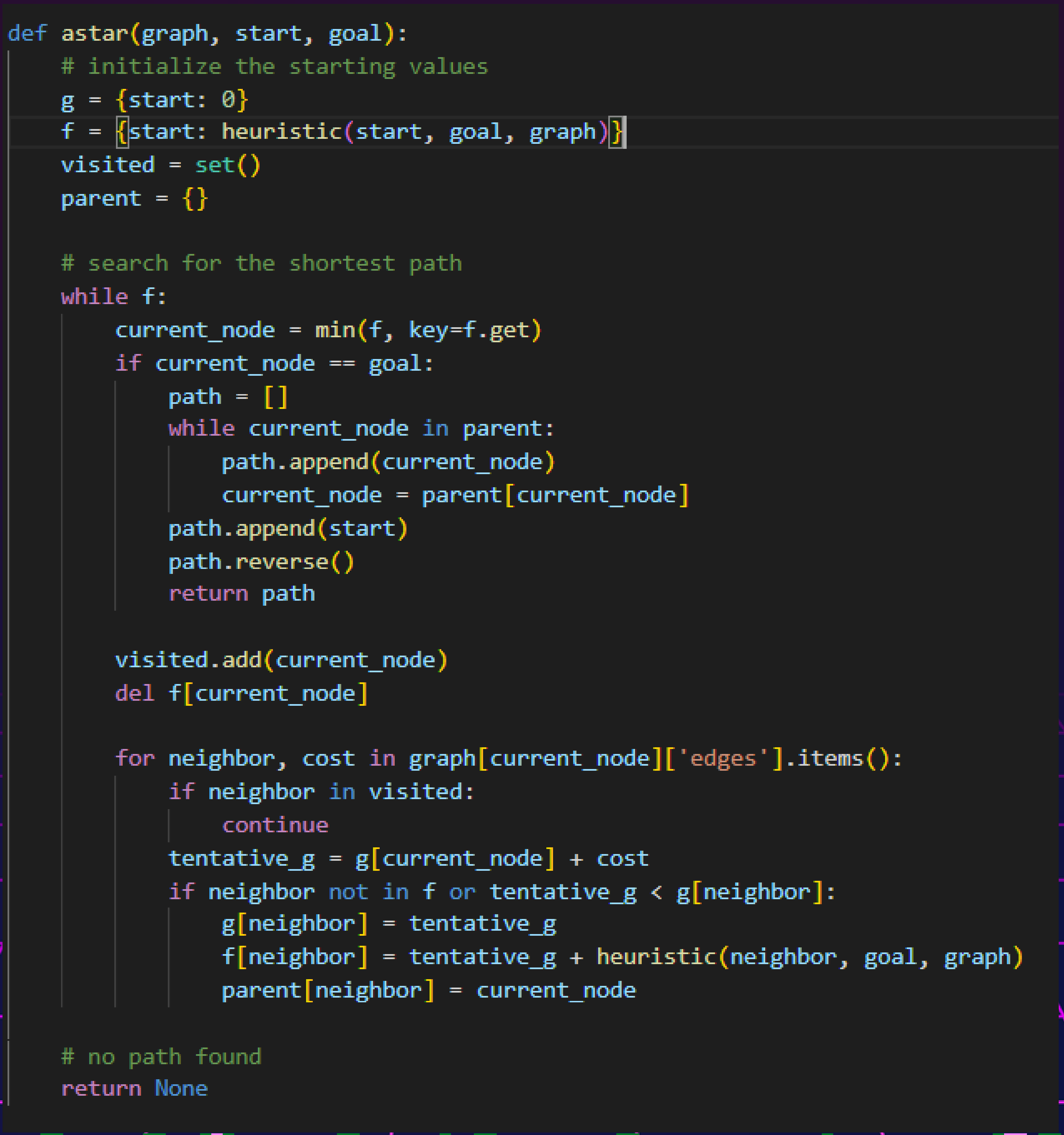
EXIT



```python
def heuristic(node1, node2, graph):
    x1, y1 = graph[node1]['pos']
    x2, y2 = graph[node2]['pos']
    return ((x2 - x1) ** 2 + (y2 - y1) ** 2) ** 0.5
```

FUNGSI HEURISTIC DIGUNAKAN UNTUK MENGHITUNG HEURISTIK (JARAK PERKIRAAN) ANTARA DUA SIMPUL PADA GRAF DENGAN MENGGUNAKAN KOORDINAT SIMPUL-SIMPUL TERSEBUT. FUNGSI INI MEMBUTUHKAN TIGA ARGUMEN YAITU NODE1 DAN NODE2 YANG MERUPAKAN SIMPUL YANG AKAN DICARI JARAKNYA, SERTA GRAPH YANG MERUPAKAN GRAF YANG BERISI INFORMASI KOORDINAT SIMPUL-SIMPUL TERSEBUT. FUNGSI INI AKAN MENGHITUNG JARAK ANTARA DUA SIMPUL MENGGUNAKAN RUMUS JARAK EUCLIDEAN (SQRT((X2-X1)^2 + (Y2-Y1)^2)) DAN MENGEMBALIKAN HASIL PERHITUNGAN TERSEBUT.

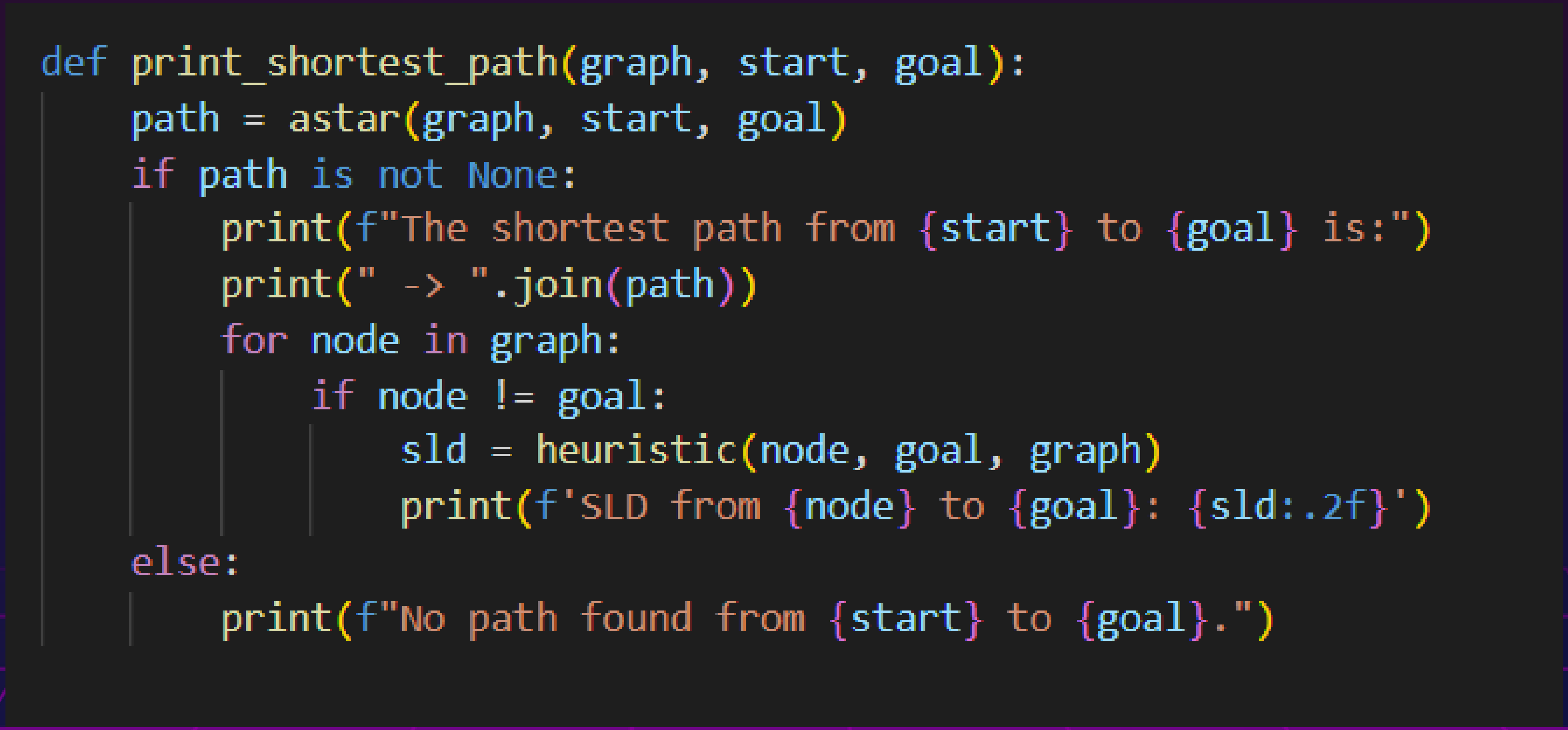
EXIT



```python
def astar(graph, start, goal):
    # initialize the starting values
    g = {start: 0}
    f = {start: heuristic(start, goal, graph)}
    visited = set()
    parent = {}

    # search for the shortest path
    while f:
        current_node = min(f, key=f.get)
        if current_node == goal:
            path = []
            while current_node in parent:
                path.append(current_node)
                current_node = parent[current_node]
            path.append(start)
            path.reverse()
            return path

        visited.add(current_node)
        del f[current_node]

        for neighbor, cost in graph[current_node]['edges'].items():
            if neighbor in visited:
                continue
            tentative_g = g[current_node] + cost
            if neighbor not in f or tentative_g < g[neighbor]:
                g[neighbor] = tentative_g
                f[neighbor] = tentative_g + heuristic(neighbor, goal, graph)
                parent[neighbor] = current_node

    # no path found
    return None
```

FUNGSI ASTAR DIGUNAKAN UNTUK MENCARI JALUR TERPENDEK PADA GRAF MENGGUNAKAN ALGORITMA A*. FUNGSI INI MEMBUTUHKAN TIGA ARGUMEN YAITU GRAPH YANG MERUPAKAN GRAF, START YANG MERUPAKAN SIMPUL AWAL, DAN GOAL YANG MERUPAKAN SIMPUL TUJUAN. FUNGSI INI AKAN MENGINISIALISASI VARIABEL G, F, VISITED, DAN PARENT, LALU MENCARI JALUR TERPENDEK DARI SIMPUL AWAL KE SIMPUL TUJUAN DENGAN MEMANFAATKAN NILAI G, F, VISITED, DAN PARENT YANG DIUPDATE PADA SETIAP ITERASI. JIKA JALUR TERPENDEK BERHASIL DITEMUKAN, MAKA FUNGSI INI AKAN MENGEMBALIKAN JALUR TERPENDEK TERSEBUT DALAM BENTUK LIST. JIKA TIDAK DITEMUKAN JALUR TERPENDEK, MAKA FUNGSI INI AKAN MENGEMBALIKAN NILAI NONE.

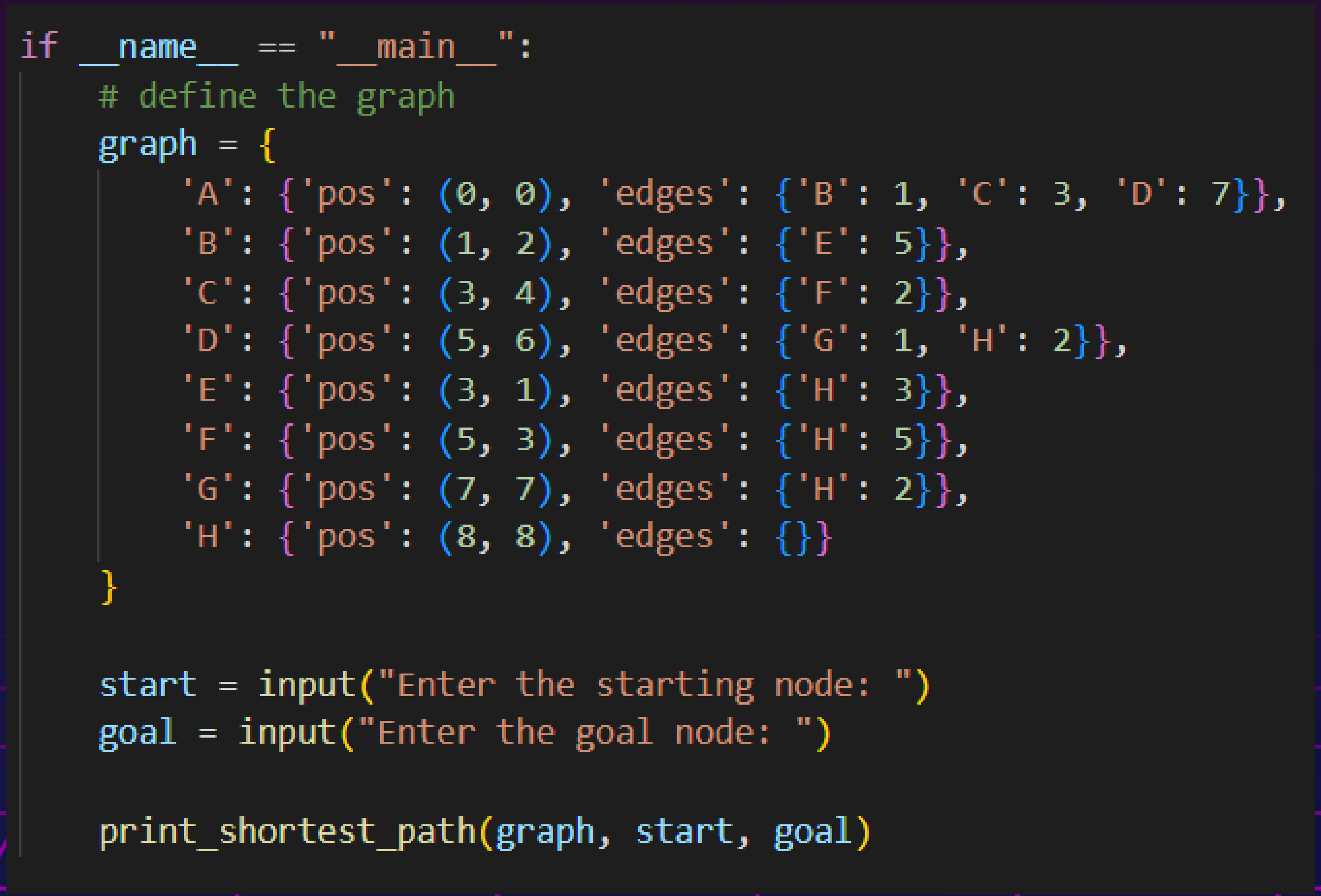
EXIT



```python
def print_shortest_path(graph, start, goal):
    path = astar(graph, start, goal)
    if path is not None:
        print(f"The shortest path from {start} to {goal} is:")
        print(" -> ".join(path))
        for node in graph:
            if node != goal:
                sld = heuristic(node, goal, graph)
                print(f'SLD from {node} to {goal}: {sld:.2f}')
    else:
        print(f"No path found from {start} to {goal}.")
```

FUNGSI PRINT_SHORTEST_PATH DIGUNAKAN UNTUK MENCETAK JALUR TERPENDEK DAN JARAK STRAIGHT LINE DISTANCE (SLD) DARI SEMUA SIMPUL DALAM GRAF GRAPH DARI SIMPUL START KE SIMPUL GOAL. FUNGSI INI MENGGUNAKAN FUNGSI ASTAR DAN HEURISTIC YANG SUDAH DIDEFINISIKAN SEBELUMNYA. JIKA TIDAK ADA JALUR YANG DITEMUKAN DARI START KE GOAL, FUNGSI INI MENCETAK PESAN YANG SESUAI.

EXIT



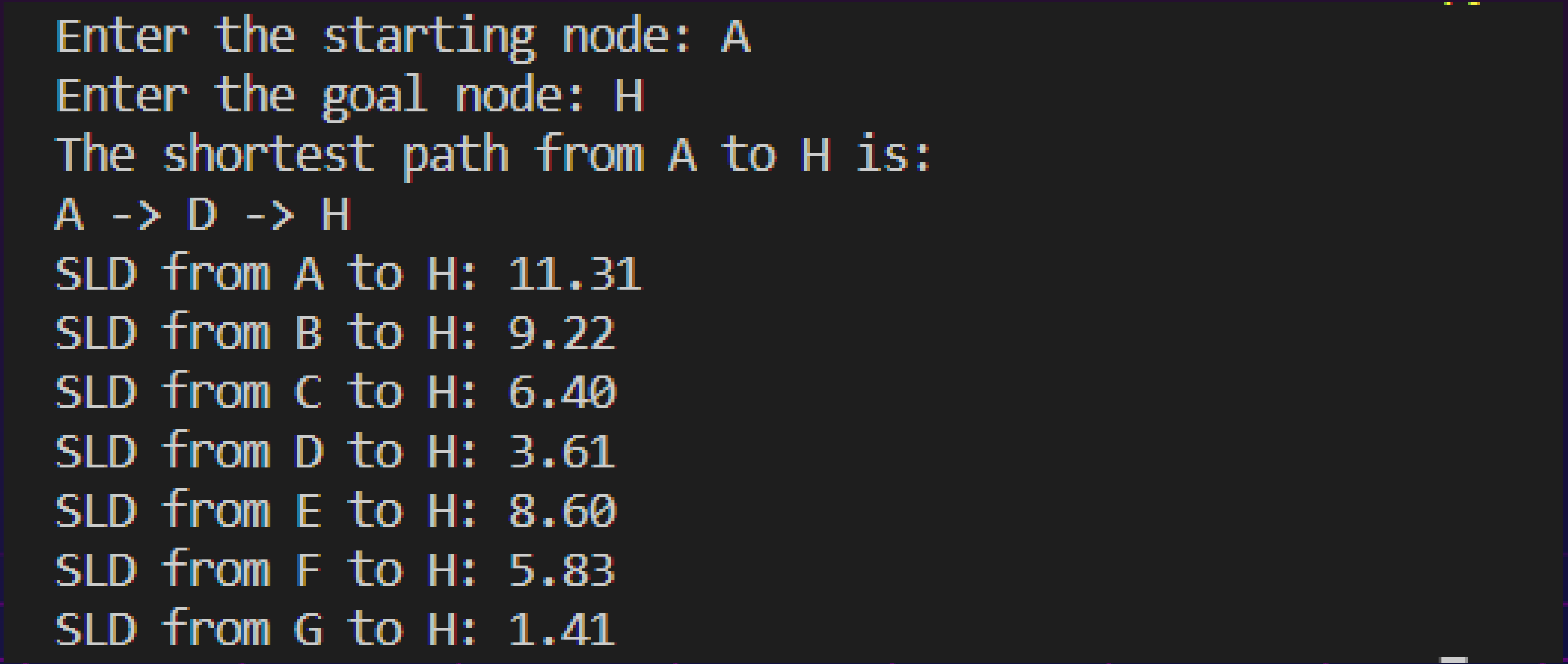
```python
if __name__ == "__main__":
    # define the graph
    graph = {
        'A': {'pos': (0, 0), 'edges': {'B': 1, 'C': 3, 'D': 7}},
        'B': {'pos': (1, 2), 'edges': {'E': 5}},
        'C': {'pos': (3, 4), 'edges': {'F': 2}},
        'D': {'pos': (5, 6), 'edges': {'G': 1, 'H': 2}},
        'E': {'pos': (3, 1), 'edges': {'H': 3}},
        'F': {'pos': (5, 3), 'edges': {'H': 5}},
        'G': {'pos': (7, 7), 'edges': {'H': 2}},
        'H': {'pos': (8, 8), 'edges': {}}
    }

    start = input("Enter the starting node: ")
    goal = input("Enter the goal node: ")

    print_shortest_path(graph, start, goal)
```

EXIT

FUNGSI MAIN DI ATAS MERUPAKAN PROGRAM UTAMA YANG MENJALANKAN APLIKASI UNTUK MENEMUKAN JALUR TERPENDEK DI DALAM GRAF YANG DIDEFINISIKAN DALAM BENTUK KAMUS. PROGRAM INI MEMINTA PENGGUNA MEMASUKKAN NODE AWAL DAN AKHIR DARI GRAF, KEMUDIAN MEMANGGIL FUNGSI "PRINT_SHORTEST_PATH" UNTUK MENEMUKAN JALUR TERPENDEK MENGGUNAKAN ALGORITMA A* SERTA JARAK SLD-NYA DAN MENCETAK HASILNYA. JIKA TIDAK ADA JALUR YANG DITEMUKAN ANTARA DUA NODE, MAKA AKAN DICETAK PESAN "NO PATH FOUND".

EXIT

CONTOH OUTPUT



```
Enter the starting node: A
Enter the goal node: H
The shortest path from A to H is:
A -> D -> H
SLD from A to H: 11.31
SLD from B to H: 9.22
SLD from C to H: 6.40
SLD from D to H: 3.61
SLD from E to H: 8.60
SLD from F to H: 5.83
SLD from G to H: 1.41
```

EXIT

THANK YOU VERY MUCH !