SUBMITTED TO: MR. MUBASHIR IQBAL

SUBMITTED BY: AHMED SALEEM RANA

REG ID: 24-CYS-023

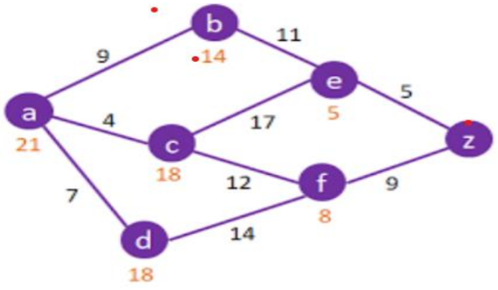
SECTION: B

HITEC UNIVERSITY TAXILA, CANTT

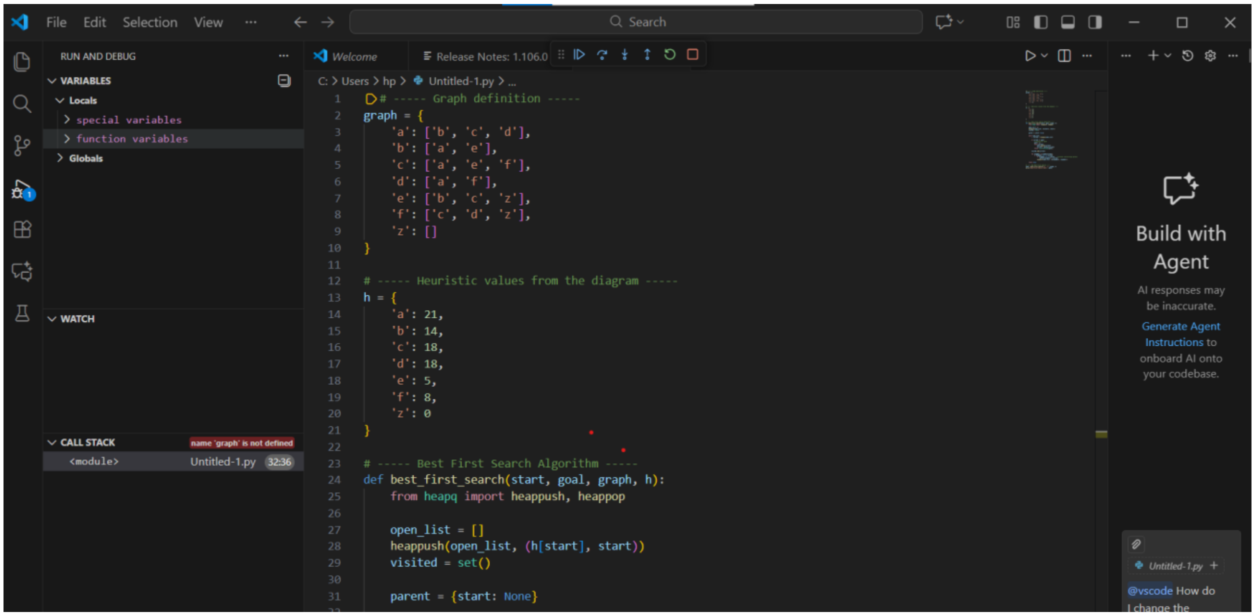
TASK 01

PART A:

a) Apply the Best First Search and Beam Search algorithm on the graph below. Root Node is (a) and goal node is (z).



BFS - CODE:



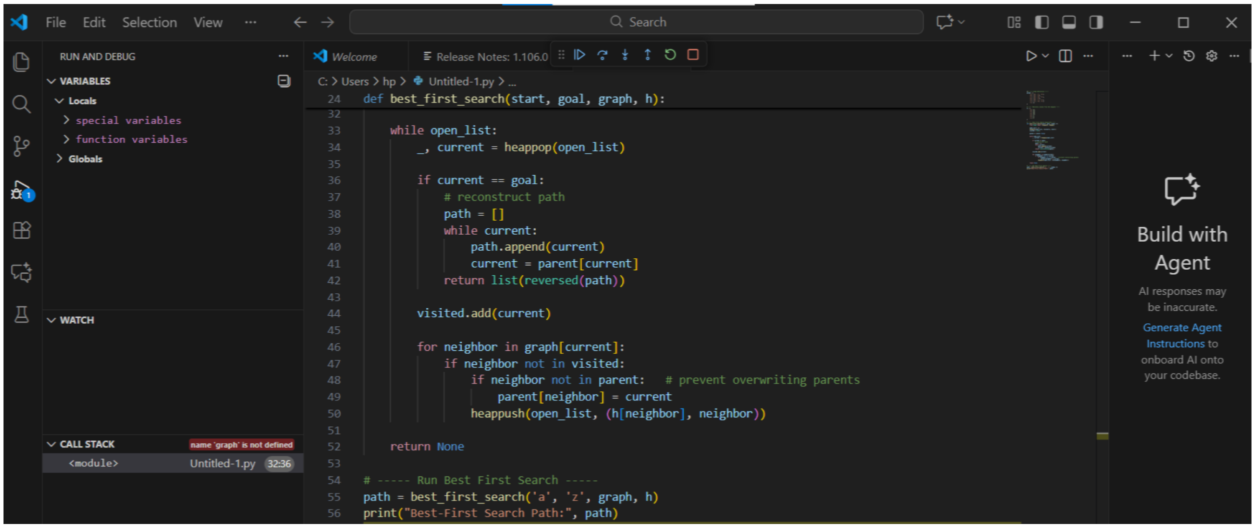
name 'graph' is not defined

AI responses may be inaccurate.

Generate Agent Instructions to onboard AI onto your codebase.

@vscode How do I change the

```python
# ----- Graph definition -----
graph = {
    'a': ['b', 'c', 'd'],
    'b': ['a', 'e'],
    'c': ['a', 'e', 'f'],
    'd': ['a', 'f'],
    'e': ['b', 'c', 'z'],
    'f': ['c', 'd', 'z'],
    'z': []
}

# ----- Heuristic values from the diagram -----
h = {
    'a': 21,
    'b': 14,
    'c': 18,
    'd': 18,
    'e': 5,
    'f': 8,
    'z': 0
}

# ----- Best First Search Algorithm -----
def best_first_search(start, goal, graph, h):
    from heapq import heappush, heappop

    open_list = []
    heappush(open_list, (h[start], start))
    visited = set()

    parent = {start: None}
```



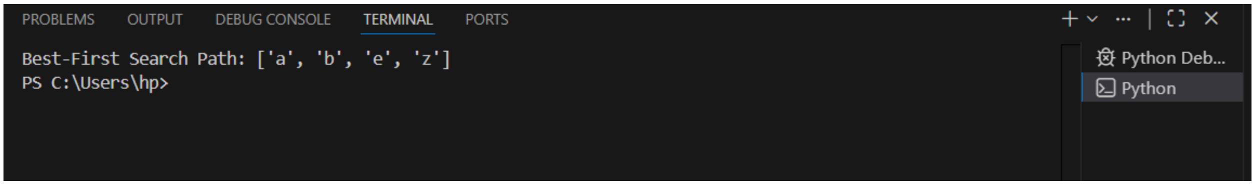
name 'graph' is not defined

AI responses may be inaccurate.

Generate Agent Instructions to onboard AI onto your codebase.

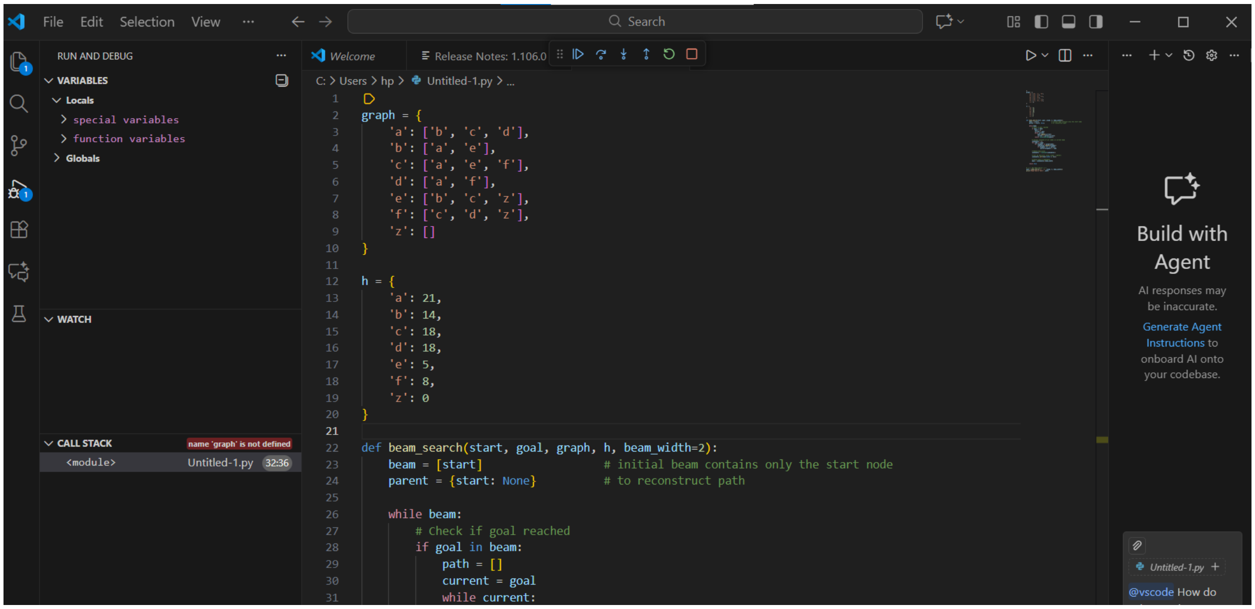
```python
def best_first_search(start, goal, graph, h):

    while open_list:
        _, current = heappop(open_list)

        if current == goal:
            # reconstruct path
            path = []
            while current:
                path.append(current)
                current = parent[current]
            return list(reversed(path))

        visited.add(current)

        for neighbor in graph[current]:
            if neighbor not in visited:
                if neighbor not in parent:   # prevent overwriting parents
                    parent[neighbor] = current
                heappush(open_list, (h[neighbor], neighbor))

    return None

# ----- Run Best First Search -----
path = best_first_search('a', 'z', graph, h)
print("Best-First Search Path:", path)
```

OUTPUT:

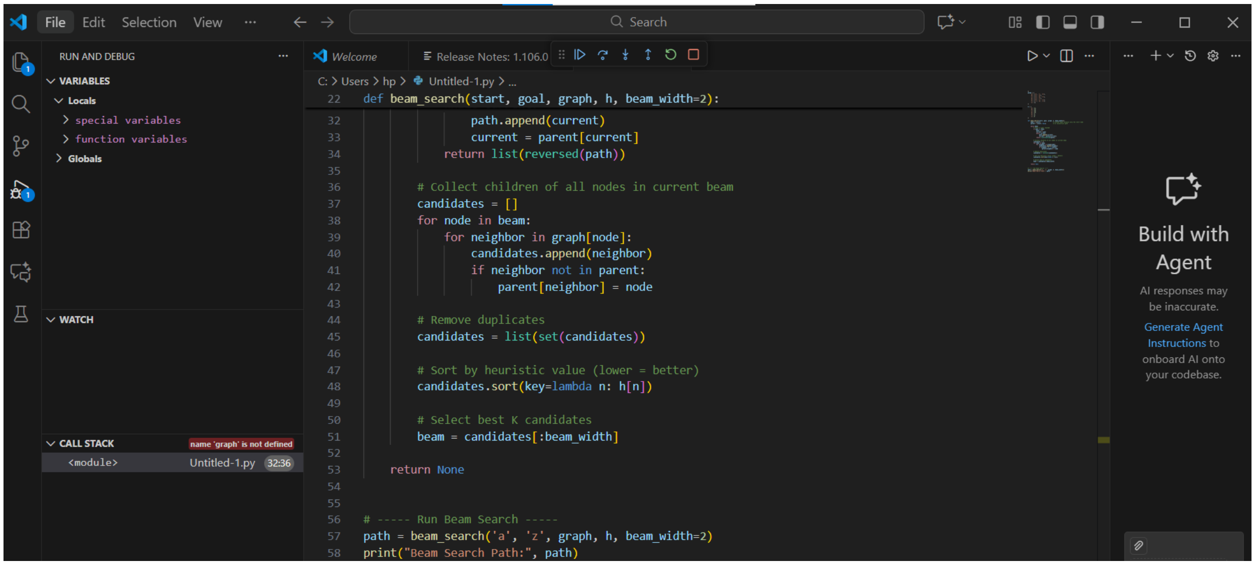


```
Best-First Search Path: ['a', 'b', 'e', 'z']
PS C:\Users\hp>
```

BEAM SEARCH ALGORITHM CODE:

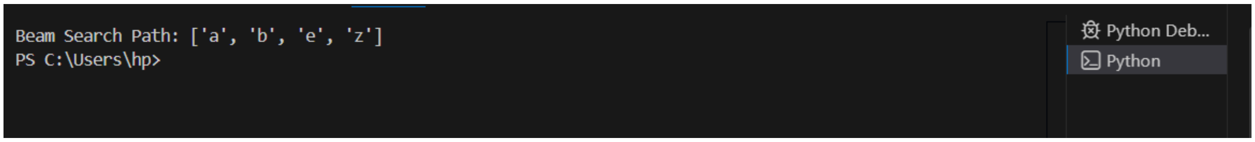


```python

graph = {
    'a': ['b', 'c', 'd'],
    'b': ['a', 'e'],
    'c': ['a', 'e', 'f'],
    'd': ['a', 'f'],
    'e': ['b', 'c', 'z'],
    'f': ['c', 'd', 'z'],
    'z': []
}

h = {
    'a': 21,
    'b': 14,
    'c': 18,
    'd': 18,
    'e': 5,
    'f': 8,
    'z': 0
}

def beam_search(start, goal, graph, h, beam_width=2):
    beam = [start]                  # initial beam contains only the start node
    parent = {start: None}          # to reconstruct path

    while beam:
        # Check if goal reached
        if goal in beam:
            path = []
            current = goal
            while current:
```



```python
def beam_search(start, goal, graph, h, beam_width=2):
                path.append(current)
                current = parent[current]
            return list(reversed(path))

        # Collect children of all nodes in current beam
        candidates = []
        for node in beam:
            for neighbor in graph[node]:
                candidates.append(neighbor)
                if neighbor not in parent:
                    parent[neighbor] = node

        # Remove duplicates
        candidates = list(set(candidates))

        # Sort by heuristic value (lower = better)
        candidates.sort(key=lambda n: h[n])

        # Select best K candidates
        beam = candidates[:beam_width]

    return None


# ----- Run Beam Search -----
path = beam_search('a', 'z', graph, h, beam_width=2)
print("Beam Search Path:", path)
```

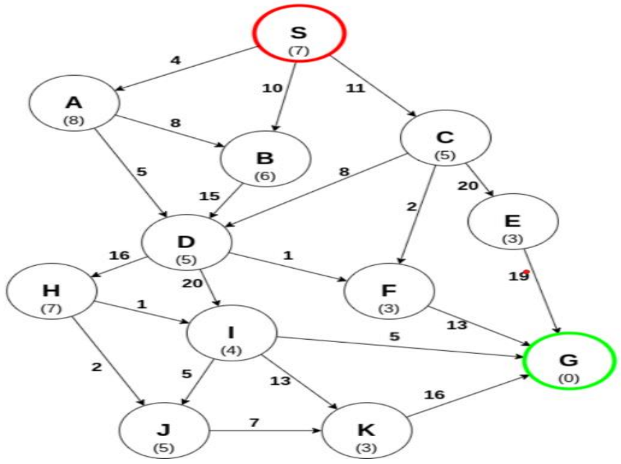
OUTPUT:



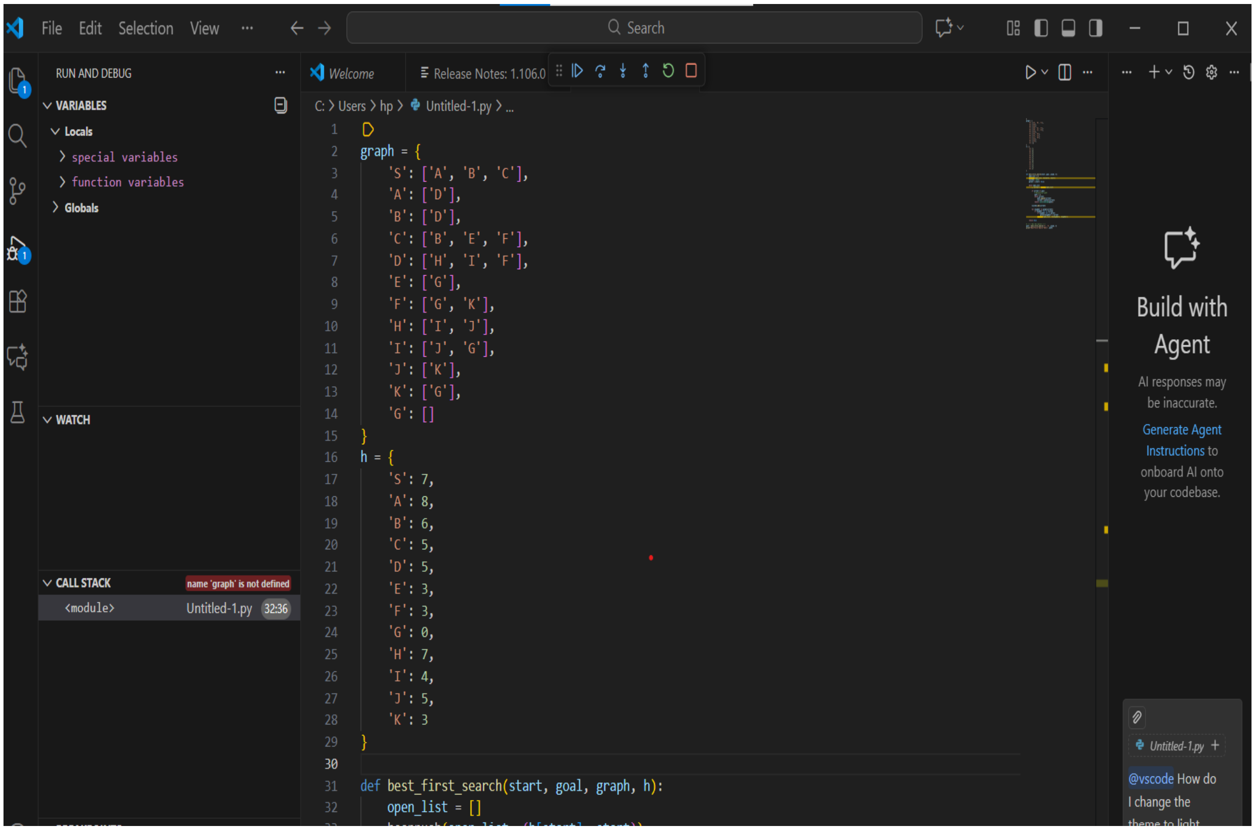
```
Beam Search Path: ['a', 'b', 'e', 'z']
PS C:\Users\hp>
```

PART B

b) Apply Best First Search, Beam Search algorithm and A* on the graph below.

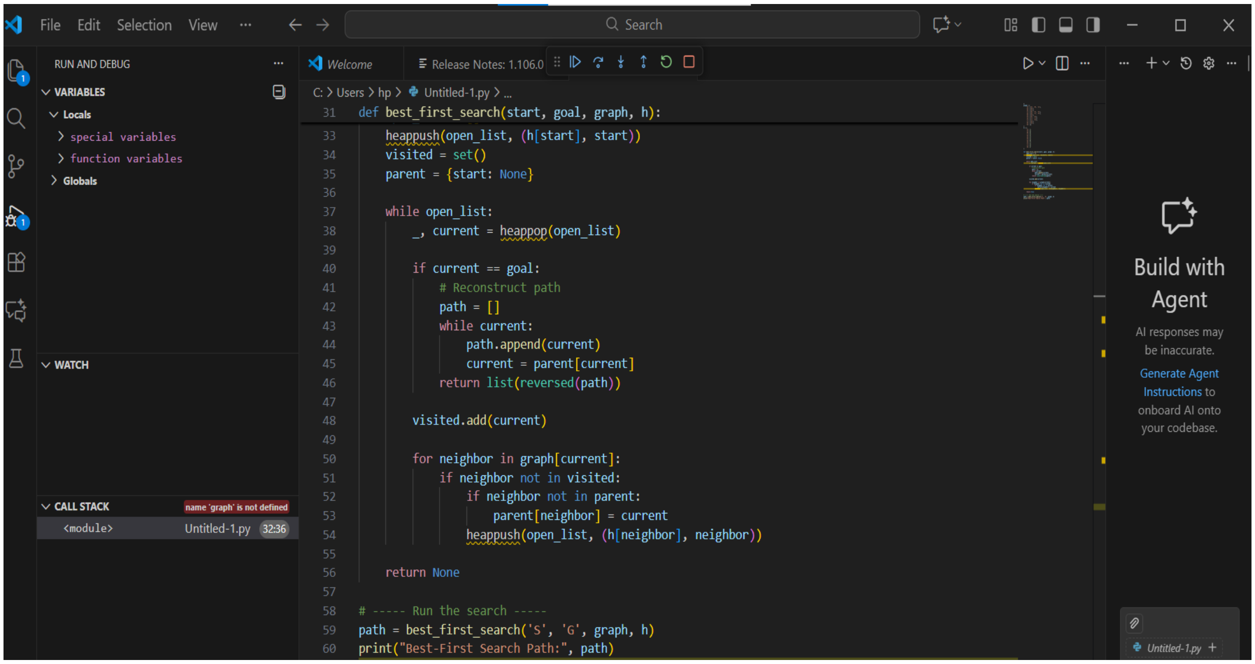


BFS – CODE:


File Edit Selection View ···
Search
RUN AND DEBUG
VARIABLES
Locals
special variables
function variables
Globals
WATCH
CALL STACK
<module>
Untitled-1.py 32:36
Welcome
Release Notes: 1.106.0
C: > Users > hp > Untitled-1.py > ...
Build with Agent
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.
Untitled-1.py +
@vscode How do I change the theme to light

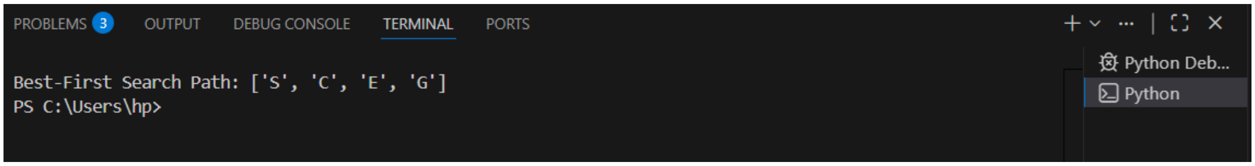
name 'graph' is not defined

```python
graph = {
    'S': ['A', 'B', 'C'],
    'A': ['D'],
    'B': ['D'],
    'C': ['B', 'E', 'F'],
    'D': ['H', 'I', 'F'],
    'E': ['G'],
    'F': ['G', 'K'],
    'H': ['I', 'J'],
    'I': ['J', 'G'],
    'J': ['K'],
    'K': ['G'],
    'G': []
}
h = {
    'S': 7,
    'A': 8,
    'B': 6,
    'C': 5,
    'D': 5,
    'E': 3,
    'F': 3,
    'G': 0,
    'H': 7,
    'I': 4,
    'J': 5,
    'K': 3
}

def best_first_search(start, goal, graph, h):
    open_list = []
    heappush(open_list, (h[start], start))
```


File Edit Selection View
Search
RUN AND DEBUG
VARIABLES
Locals
special variables
function variables
Globals
WATCH
CALL STACK
name 'graph' is not defined
<module> Untitled-1.py 32:36
Welcome
Release Notes: 1.106.0
C: > Users > hp > Untitled-1.py > ...
Build with Agent
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.
Untitled-1.py

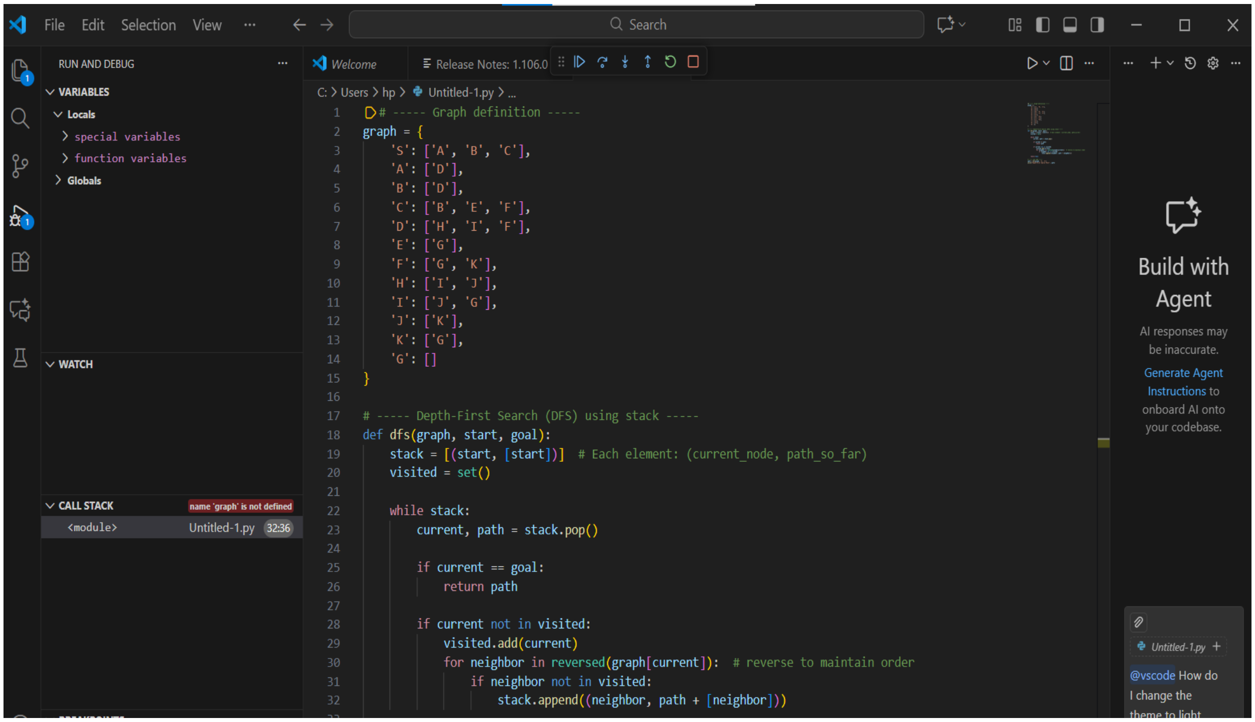
```python
def best_first_search(start, goal, graph, h):
    heappush(open_list, (h[start], start))
    visited = set()
    parent = {start: None}

    while open_list:
        _, current = heappop(open_list)

        if current == goal:
            # Reconstruct path
            path = []
            while current:
                path.append(current)
                current = parent[current]
            return list(reversed(path))

        visited.add(current)

        for neighbor in graph[current]:
            if neighbor not in visited:
                if neighbor not in parent:
                    parent[neighbor] = current
                heappush(open_list, (h[neighbor], neighbor))

    return None

# ----- Run the search -----
path = best_first_search('S', 'G', graph, h)
print("Best-First Search Path:", path)
```

OUTPUT:

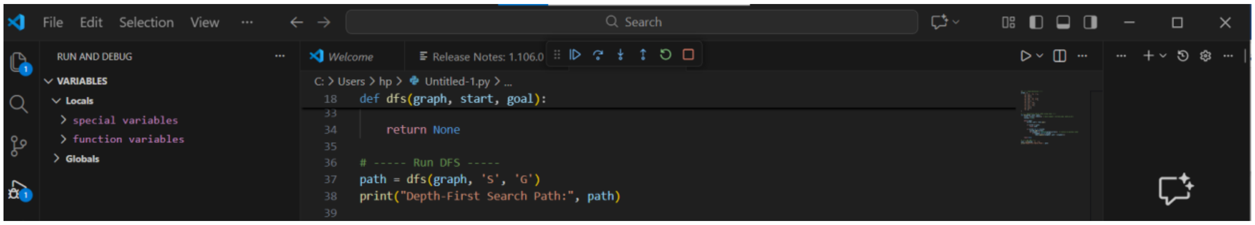


```
Best-First Search Path: ['S', 'C', 'E', 'G']
PS C:\Users\hp>
```

DEEP SEARCH ALGORITHM CODE:

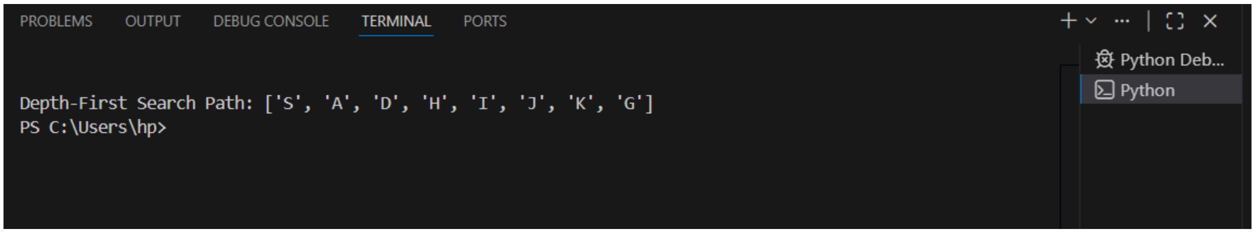

File Edit Selection View
Search
RUN AND DEBUG
VARIABLES
Locals
special variables
function variables
Globals
WATCH
CALL STACK
name 'graph' is not defined
<module> Untitled-1.py 32:36
Welcome
Release Notes: 1.106.0
C: > Users > hp > Untitled-1.py > ...
Build with Agent
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.
Untitled-1.py
@vscode How do I change the theme to light

```python
# ----- Graph definition -----
graph = {
    'S': ['A', 'B', 'C'],
    'A': ['D'],
    'B': ['D'],
    'C': ['B', 'E', 'F'],
    'D': ['H', 'I', 'F'],
    'E': ['G'],
    'F': ['G', 'K'],
    'H': ['I', 'J'],
    'I': ['J', 'G'],
    'J': ['K'],
    'K': ['G'],
    'G': []
}

# ----- Depth-First Search (DFS) using stack -----
def dfs(graph, start, goal):
    stack = [(start, [start])]  # Each element: (current_node, path_so_far)
    visited = set()

    while stack:
        current, path = stack.pop()

        if current == goal:
            return path

        if current not in visited:
            visited.add(current)
            for neighbor in reversed(graph[current]):  # reverse to maintain order
                if neighbor not in visited:
                    stack.append((neighbor, path + [neighbor]))
```



```python
def dfs(graph, start, goal):

    return None

# ----- Run DFS -----
path = dfs(graph, 'S', 'G')
print("Depth-First Search Path:", path)
```

OUTPUT:



```
Depth-First Search Path: ['S', 'A', 'D', 'H', 'I', 'J', 'K', 'G']
PS C:\Users\hp>
```

CONCLUSION:

We have applied several search algorithms on two different graphs to identify paths from the start to the goal. In the first graph, **Best-First Search (DFS)** algorithm is used which relies on heuristics to prioritize nodes closer to the goal, while **Beam Search** algorithm restricted the number of nodes explored at each level to find a path efficiently. In the second graph, we implemented **Depth-First Search (DFS)**, which traversed the graph deeply before backtracking, successfully reaching the goal node **'S' to 'G'.**