File Edit Format Run Options Window Help

```python
def is_safe(queens, row, col):
    return all(queens[i] != col and abs(queens[i] - col) != row - i for i in range(row))

def place_queens(queens, row, size, solutions):
    if row == size:
        solutions.append(queens[:])
        return True

    for col in range(size):
        if is_safe(queens, row, col):
            queens[row] = col
            if place_queens(queens, row + 1, size, solutions):
                return True

    return False

def show_solution(solution, size):
    for row in range(size):
        print("".join("Q" if col == solution[row] else "." for col in range(size)))
    print("-" * size)

size = 8
queens, solutions = [-1] * size, []
place_queens(queens, 0, size, solutions)

print("First valid solution:\n")
show_solution(solutions[0], size)
```

File Edit Shell Debug Options Window Help

```
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (AMD64)]
Type "help", "copyright", "credits" or "license()" for more information
>>>
==================== RESTART: D:/Akshaya (python)/A star.py ===========
First valid solution:

Q.......
....Q...
.......Q
.....Q..
..Q.....
......Q.
.Q......
...Q....
--------
>>>
```

```python
import heapq

def a_star(grid, start, goal):
    open_list = [(0, start)]
    came_from = {}
    g_score = {start: 0}

    while open_list:
        _, curr = heapq.heappop(open_list)

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

        for dx, dy in [(0, 1), (0, -1), (1, 0), (-1, 0)]:
            nxt = (curr[0] + dx, curr[1] + dy)
            if 0 <= nxt[0] < len(grid) and 0 <= nxt[1] < len(grid[0]) and grid[nxt[0]][nxt[1]] == 0:
                if nxt not in g_score or g_score[nxt] > g_score[curr] + 1:
                    g_score[nxt] = g_score[curr] + 1
                    heapq.heappush(open_list, (g_score[nxt], nxt))
                    came_from[nxt] = curr

    return None

grid = [[0, 0, 0], [0, 1, 0], [0, 0, 0]]
start, goal = (0, 0), (2, 2)

print(a_star(grid, start, goal))
```

File Edit Shell Debug Options Window Help

```
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [M
Type "help", "copyright", "credits" or "licens
>>>
==================== RESTART: D:/Akshaya (pyth
[(0, 1), (0, 2), (1, 2), (2, 2)]
>>>
```

File Edit Format Run Options Window Help

```python
def dfs(graph, node, visited=None):
    if visited is None:
        visited = set()
    visited.add(node)
    print(node, end=" ")

    for neighbor in graph[node]:
        if neighbor not in visited:
            dfs(graph, neighbor, visited)

graph = {
    'A': ['B', 'C'],
    'B': ['A', 'D', 'E'],
    'C': ['A', 'F'],
    'D': ['B'],
    'E': ['B', 'F'],
    'F': ['C', 'E']
}

dfs(graph, 'A')
```

File Edit Shell Debug Options Window Help

```
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
==================== RESTART: D:/Akshaya (python)/A star.py ====================
A B D E F C
>>> S
```

```python
rules = {
    "A": ["B", "C"],
    "B": ["D"],
    "C": ["E"],
    "D": [],
    "E": []
}

def backward_chaining(goal, facts):
    if goal in facts:
        return True
    for condition in rules.get(goal, []):
        if not backward_chaining(condition, facts):
            return False
    return True

facts = {"D", "E"}
print("Goal A:", backward_chaining("A", facts))  # Output: True
```

File Edit Shell Debug Options Window Help

```
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933
Type "help", "copyright", "credits" or "license()" for
>>>
==================== RESTART: D:/Akshaya (python)/A sta
Goal A: True
>>>
```

```python
rules = {
    "D": ["B"],
    "E": ["C"],
    "B": ["A"],
    "C": ["A"]
}

def forward_chaining(facts, goal):
    inferred = set(facts)
    while True:
        new_facts = set()
        for conclusion, conditions in rules.items():
            if all(cond in inferred for cond in conditions):
                new_facts.add(conclusion)
        if not new_facts - inferred:
            break
        inferred.update(new_facts)
    return goal in inferred

facts = {"D", "E"}
print("Goal A:", forward_chaining(facts, "A"))  # Output: True
```

File Edit Shell Debug Options Window Help

```
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.19
Type "help", "copyright", "credits" or "license()" fo
>>>
==================== RESTART: D:/Akshaya (python)/A s
Goal A: False
>>>
```

```python
class FuzzySet:
    def __init__(self, name, min_val, max_val):
        self.name = name
        self.min_val = min_val
        self.max_val = max_val

    def membership(self, value):
        if value <= self.min_val:
            return 0
        elif value >= self.max_val:
            return 1
        else:
            return (value - self.min_val) / (self.max_val - self.min_val)

# Define fuzzy sets
cold = FuzzySet("Cold", 0, 20)
warm = FuzzySet("Warm", 15, 30)
hot = FuzzySet("Hot", 25, 40)

# Example input
temp_value = 25
print(f"Cold: {cold.membership(temp_value):.2f}")
print(f"Warm: {warm.membership(temp_value):.2f}")
print(f"Hot: {hot.membership(temp_value):.2f}")
```

File Edit Shell Debug Options Window Help

```
Python 3.11.0 (main, Oct 24 2022,
Type "help", "copyright", "credit
>>>
==================== RESTART: D:/
Cold: 1.00
Warm: 0.67
Hot: 0.00
>>>
```

```python
def unify(term1, term2, subst={}):
    if term1 == term2:
        return subst
    if isinstance(term1, str) and term1.islower():
        return unify_var(term1, term2, subst)
    if isinstance(term2, str) and term2.islower():
        return unify_var(term2, term1, subst)
    if isinstance(term1, list) and isinstance(term2, list) and len(term1) == len(term2):
        for t1, t2 in zip(term1, term2):
            subst = unify(t1, t2, subst)
            if subst is None:
                return None
        return subst
    return None

def unify_var(var, term, subst):
    if var in subst:
        return unify(subst[var], term, subst)
    if term in subst:
        return unify(var, subst[term], subst)
    subst[var] = term
    return subst

# Resolution function
def resolve(clause1, clause2):
    for literal1 in clause1:
        for literal2 in clause2:
            if literal1 == f"~{literal2}" or literal2 == f"~{literal1}":
                new_clause = list(set(clause1 + clause2) - {literal1, literal2})
                return new_clause if new_clause else ["Contradiction"]
    return None

# Example usage
term1 = ["P", "x"]
term2 = ["P", "a"]
print("Unification:", unify(term1, term2))

clause1 = ["P(x)", "~Q(x)"]
clause2 = ["Q(a)"]
print("Resolution:", resolve(clause1, clause2))
```

File Edit Shell Debug Options Window Help

```
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 b
Type "help", "copyright", "credits" or "license()" for more
>>>
==================== RESTART: D:/Akshaya (python)/A star.py
Unification: {'x': 'a'}
Resolution: None
>>>
```

```python
class BlocksWorld:
    def __init__(self, initial, goal):
        self.state = initial
        self.goal = goal

    def move(self, block, destination):
        if block in self.state and destination in self.state:
            self.state[destination].append(self.state[block].pop())
            return True
        return False

    def is_goal_reached(self):
        return self.state == self.goal

# Example setup
initial_state = {"A": ["B"], "B": [], "Table": ["A"]}
goal_state = {"A": [], "B": ["A"], "Table": ["B"]}

bw = BlocksWorld(initial_state, goal_state)
bw.move("A", "B")
print("Goal reached:", bw.is_goal_reached())
```

```
File  Edit  Shell  Debug  Options  Window  Help
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC
Type "help", "copyright", "credits" or "license()
>>>
==================== RESTART: D:/Akshaya (python)
Goal reached: False
>>>
```

```python
def decide_weather(weather):
    if weather == "sunny":
        return "Go for a walk!"
    elif weather == "rainy":
        return "Stay indoors!"
    else:
        return "Check the forecast!"

print(decide_weather("sunny"))
print(decide_weather("rainy"))
print(decide_weather("cloudy"))
```

File Edit Shell Debug Options Window Help

```
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (AMD64)] on win3
Type "help", "copyright", "credits" or "license()" for more information.
>>>
==================== RESTART: D:/Akshaya (python)/A star.py ===================
Go for a walk!
Stay indoors!
Check the forecast!
>>>
```