main.py

Share Run

```python
import heapq
def neighbors(node): return [(node[0]+1,node[1]),(node[0]-1,node[1]),(node[0],node[1]+1),(node[0],node[1]-1)]
def h(n,goal): return abs(n[0]-goal[0])+abs(n[1]-goal[1])
def a_star(start,goal,neighbors,h):
    open_set,g_score,came_from=[(0,start)],{start:0},{}
    while open_set:
        _,current=heapq.heappop(open_set)
        if current==goal:
            path=[]
            while current in came_from: path.append(current);current=came_from[current]
            return path[::-1]
        for n in neighbors(current):
            t_g_score=g_score[current]+1
            if t_g_score<g_score.get(n,float('inf')):
                came_from[n],g_score[n]=current,t_g_score
                heapq.heappush(open_set,(t_g_score+h(n,goal),n))
    return None
print("Path:", a_star((0,0),(3,3),neighbors,h))

```

Output Clear

```
Path: [(0, 1), (0, 2), (0, 3), (1, 3), (2, 3), (3, 3)]

=== Code Execution Successful ===
```

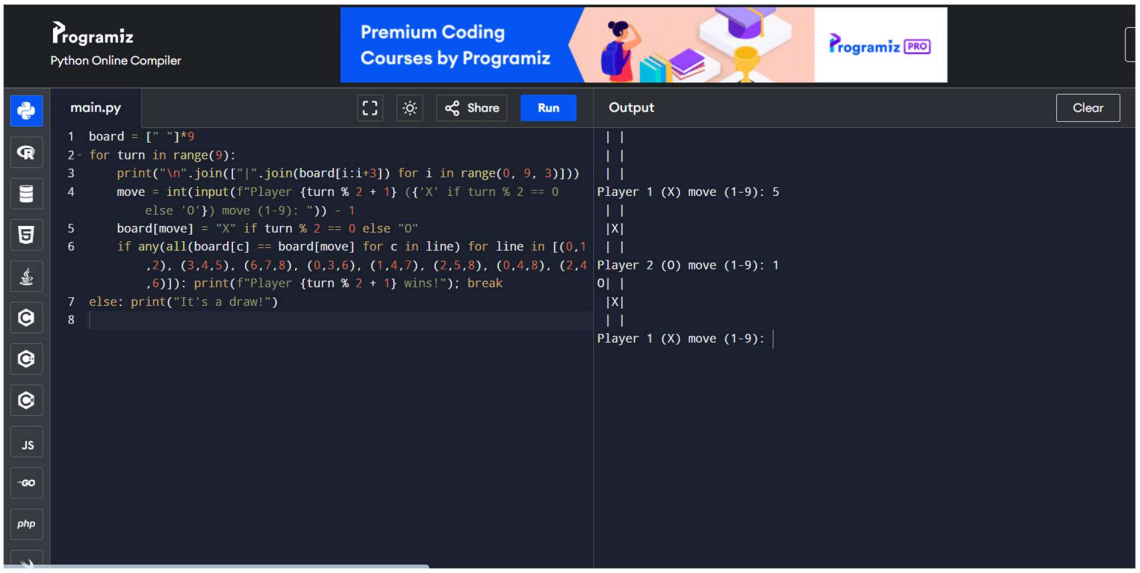

main.py

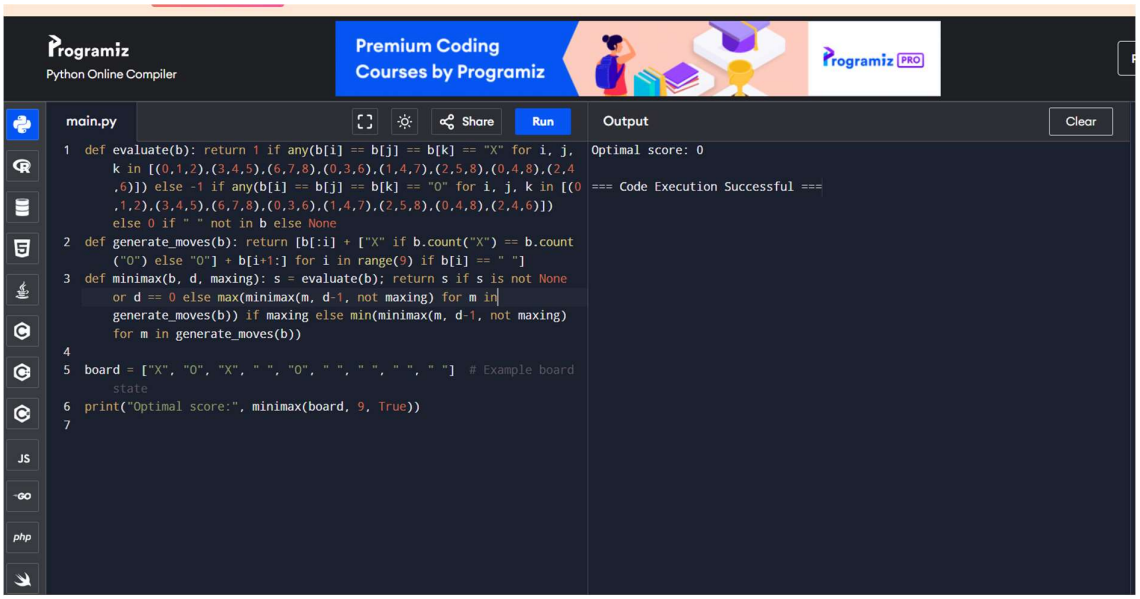
Share Run

```python
regions, colors, neighbors = ["WA", "NT", "SA", "Q", "NSW", "V", "T"], ["Red", "Green", "Blue"], {"WA": ["NT", "SA"], "NT": ["WA", "SA", "Q"], "SA": ["WA", "NT", "Q", "NSW", "V"], "Q": ["NT", "SA", "NSW"], "NSW": ["SA", "Q", "V"], "V": ["SA", "NSW"], "T": []}
def color_map(assignment={}, i=0): return assignment if i == len(regions) else next((res for color in colors if all(assignment.get(n) != color for n in neighbors[regions[i]]) and (res := color_map({**assignment, regions[i]: color}, i + 1))), None)
solution = color_map()
print("Solution:", solution)

```

Output Clear

```
Solution: {'WA': 'Red', 'NT': 'Green', 'SA': 'Blue', 'Q': 'Red', 'NSW': 'Green', 'V': 'Red', 'T': 'Red'}

=== Code Execution Successful ===
```


Programiz
Python Online Compiler
Premium Coding
Courses by Programiz
Programiz PRO
main.py
Share
Run
Output
Clear
board = [" "]*9
for turn in range(9):
print("\n".join(["|".join(board[i:i+3]) for i in range(0, 9, 3)]))
move = int(input(f"Player {turn % 2 + 1} ({'X' if turn % 2 == 0 else 'O'}) move (1-9): ")) - 1
board[move] = "X" if turn % 2 == 0 else "O"
if any(all(board[c] == board[move] for c in line) for line in [(0,1,2), (3,4,5), (6,7,8), (0,3,6), (1,4,7), (2,5,8), (0,4,8), (2,4,6)]): print(f"Player {turn % 2 + 1} wins!"); break
else: print("It's a draw!")
| |
| |
| |
Player 1 (X) move (1-9): 5
| |
|X|
| |
Player 2 (O) move (1-9): 1
O| |
|X|
| |
Player 1 (X) move (1-9):


Programiz
Python Online Compiler
Premium Coding
Courses by Programiz
Programiz PRO
main.py
Share
Run
Output
Clear
def evaluate(b): return 1 if any(b[i] == b[j] == b[k] == "X" for i, j, k in [(0,1,2),(3,4,5),(6,7,8),(0,3,6),(1,4,7),(2,5,8),(0,4,8),(2,4,6)]) else -1 if any(b[i] == b[j] == b[k] == "O" for i, j, k in [(0,1,2),(3,4,5),(6,7,8),(0,3,6),(1,4,7),(2,5,8),(0,4,8),(2,4,6)]) else 0 if " " not in b else None
def generate_moves(b): return [b[:i] + ["X" if b.count("X") == b.count("O") else "O"] + b[i+1:] for i in range(9) if b[i] == " "]
def minimax(b, d, maxing): s = evaluate(b); return s if s is not None or d == 0 else max(minimax(m, d-1, not maxing) for m in generate_moves(b)) if maxing else min(minimax(m, d-1, not maxing) for m in generate_moves(b))
board = ["X", "O", "X", " ", "O", " ", " ", " ", " ", " "] # Example board state
print("Optimal score:", minimax(board, 9, True))
Optimal score: 0
=== Code Execution Successful ===

main.py

Share Run

```
def evaluate(b): return 1 if any(b[i] == b[j] == b[k] == "X" for i, j, k in [(0,1,2),(3,4,5),(6,7,8),(0,3,6),(1,4,7),(2,5,8),(0,4,8),(2,4,6)]) else -1 if any(b[i] == b[j] == b[k] == "O" for i, j, k in [(0,1,2),(3,4,5),(6,7,8),(0,3,6),(1,4,7),(2,5,8),(0,4,8),(2,4,6)]) else 0 if " " not in b else None
def generate_moves(b): t = "X" if b.count("X") == b.count("O") else "O"; return [b[:i] + [t] + b[i+1:] for i in range(9) if b[i] == " "]
def alphabeta(b, d, a, b_, maxing): s = evaluate(b); return s if s is not None or d == 0 else (max(alphabeta(m, d-1, a, b_, 0) for m in generate_moves(b)) if maxing else min(alphabeta(m, d-1, a, b_, 1) for m in generate_moves(b)))
board = ["X", "O", "X", " ", "O", " ", " ", " ", " "]
print("Optimal score:", alphabeta(board, 9, -float("inf"), float("inf"), True))

```

Output Clear

```
Optimal score: 0

=== Code Execution Successful ===
```