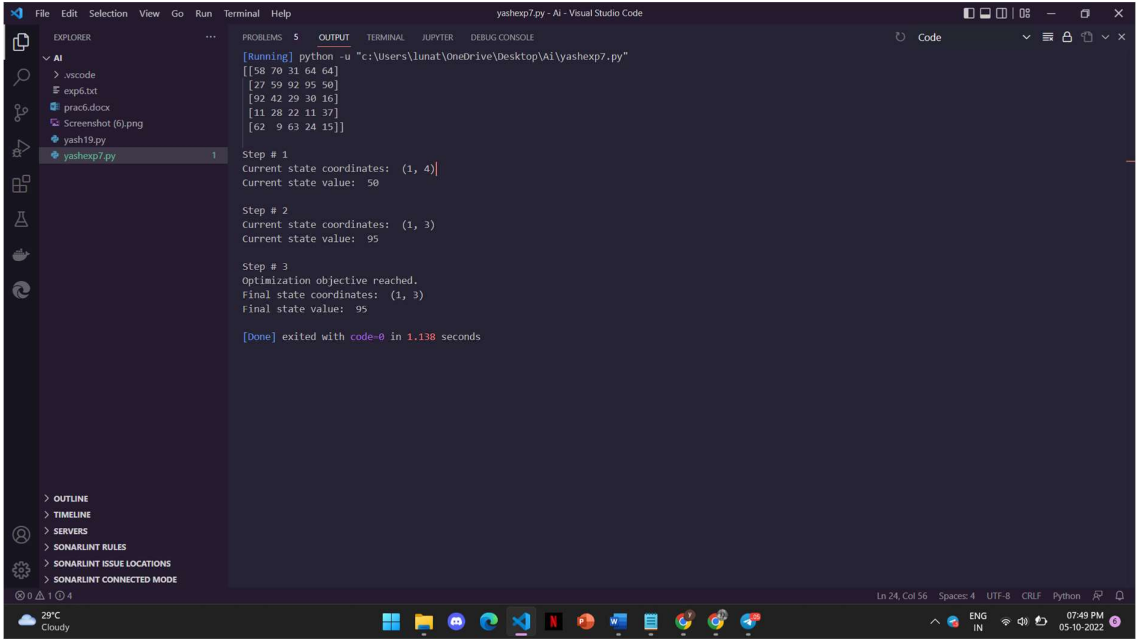
Code:

```
import numpy as np
def find_neighbours(state, landscape):
    neighbours = []
    dim = landscape.shape

    # left neighbour
    if state[0] != 0:
        neighbours.append((state[0] - 1, state[1]))

    # right neighbour
    if state[0] != dim[0] - 1:
        neighbours.append((state[0] + 1, state[1]))

    # top neighbour
    if state[1] != 0:
        neighbours.append((state[0], state[1] - 1))

    # bottom neighbour
    if state[1] != dim[1] - 1:
        neighbours.append((state[0], state[1] + 1))

    # top left
    if state[0] != 0 and state[1] != 0:
        neighbours.append((state[0] - 1, state[1] - 1))

    # bottom left
    if state[0] != 0 and state[1] != dim[1] - 1:
        neighbours.append((state[0] - 1, state[1] + 1))

    # top right
    if state[0] != dim[0] - 1 and state[1] != 0:
        neighbours.append((state[0] + 1, state[1] - 1))

    # bottom right
    if state[0] != dim[0] - 1 and state[1] != dim[1] - 1:
        neighbours.append((state[0] + 1, state[1] + 1))

    return neighbours

# Current optimization objective: local/global maximum
def hill_climb(curr_state, landscape):
    neighbours = find_neighbours(curr_state, landscape)
    bool
    ascended = False
    next_state = curr_state
    for neighbour in neighbours: #Find the neighbour with the greatest value
        if landscape[neighbour[0]][neighbour[1]] > landscape[next_state[0]][next_state[1]]:
            next_state = neighbour
            ascended = True

    return ascended, next_state

def _main_():
    landscape = np.random.randint(1, high=100, size=(5, 5))
    print(landscape)
    start_state = (1, 4)  # matrix index coordinates
    current_state = start_state
    count = 1
    ascending = True
    while ascending:
        print("\nStep #", count)
        print("Current state coordinates: ", current_state)
        print("Current state value: ", landscape[current_state[0]][current_state[1]])
        count += 1
        ascending, current_state = hill_climb(current_state, landscape)

    print("\nStep #", count)
    print("Optimization objective reached.")
    print("Final state coordinates: ", current_state)
    print("Final state value: ", landscape[current_state[0]][current_state[1]])

_main_()
```

Output:



```
[Running] python -u "c:\Users\lunat\OneDrive\Desktop\Ai\yashexp7.py"
[[58 70 31 64 64]
 [27 59 92 95 50]
 [92 42 29 30 16]
 [11 28 22 11 37]
 [62  9 63 24 15]]

Step # 1
Current state coordinates:  (1, 4)
Current state value:  50

Step # 2
Current state coordinates:  (1, 3)
Current state value:  95

Step # 3
Optimization objective reached.
Final state coordinates:  (1, 3)
Final state value:  95

[Done] exited with code=0 in 1.138 seconds
```