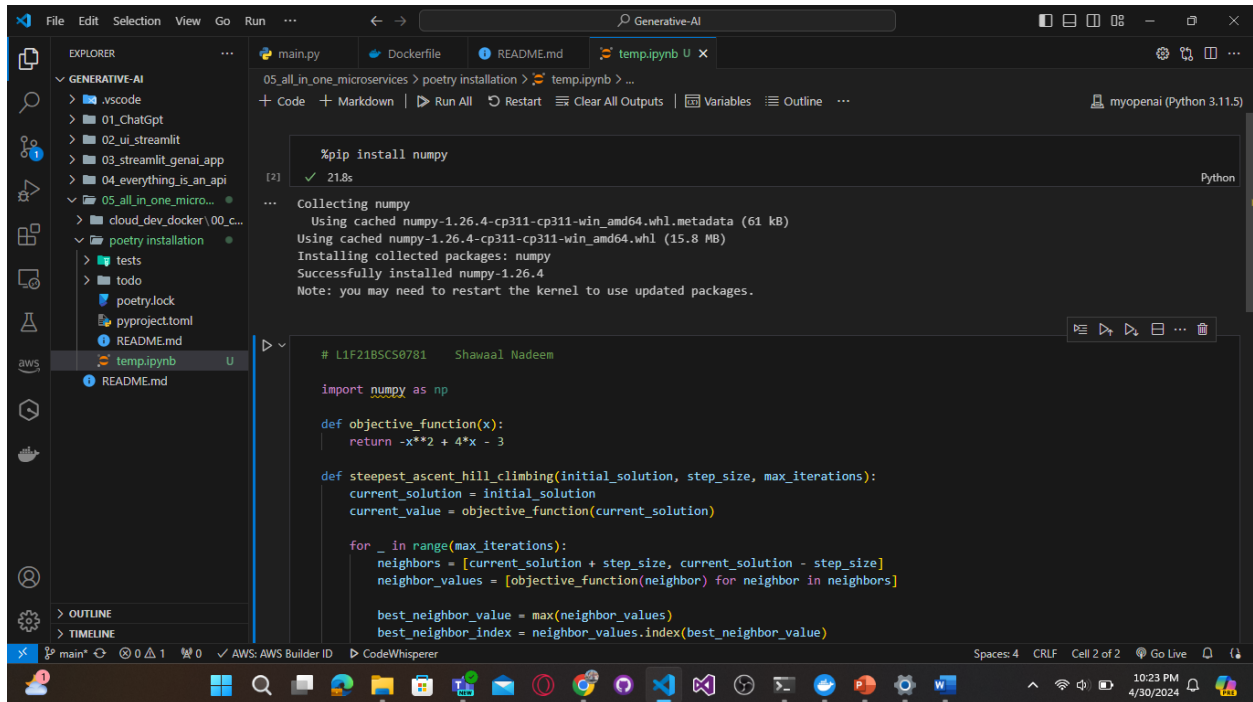


Name: Shawaal Nadeem

Reg no: L1F21BSCS0781

## Hill Climb Steepest Ascent



The screenshot shows a Jupyter Notebook environment within VS Code. The left sidebar displays the Explorer view with a project structure under 'GENERATIVE-AI'. The main editor area shows a Jupyter Notebook with two cells. The first cell contains a command to install numpy, which has been executed successfully. The second cell contains Python code for a Hill Climb Steepest Ascent algorithm.

```
05_all_in_one_microservices > poetry installation > temp.ipynb > ...
+ Code + Markdown | ▶ Run All ⌂ Restart | Clear All Outputs | Variables Outline ...
myopenai (Python 3.11.5)

[2] ✓ 218s
...
Collecting numpy
  Using cached numpy-1.26.4-cp311-win_amd64.whl.metadata (61 kB)
  Using cached numpy-1.26.4-cp311-win_amd64.whl (15.8 MB)
Installing collected packages: numpy
Successfully installed numpy-1.26.4
Note: you may need to restart the kernel to use updated packages.

# L1F21BSCS0781 Shawaal Nadeem

import numpy as np

def objective_function(x):
    return -x**2 + 4*x - 3

def steepest_ascent_hill_climbing(initial_solution, step_size, max_iterations):
    current_solution = initial_solution
    current_value = objective_function(current_solution)

    for _ in range(max_iterations):
        neighbors = [current_solution + step_size, current_solution - step_size]
        neighbor_values = [objective_function(neighbor) for neighbor in neighbors]

        best_neighbor_value = max(neighbor_values)
        best_neighbor_index = neighbor_values.index(best_neighbor_value)
```

```
05_all_in_one_microservices > poetry installation > temp.ipynb > ...
+ Code + Markdown | Run All | Restart | Clear All Outputs | Variables | Outline ...
myopenai (Python 3.11.5)

current_solution = initial_solution
current_value = objective_function(current_solution)

for _ in range(max_iterations):
    neighbors = [current_solution + step_size, current_solution - step_size]
    neighbor_values = [objective_function(neighbor) for neighbor in neighbors]

    best_neighbor_value = max(neighbor_values)
    best_neighbor_index = neighbor_values.index(best_neighbor_value)

    if best_neighbor_value <= current_value:
        break

    current_solution = neighbors[best_neighbor_index]
    current_value = best_neighbor_value

return current_solution, current_value

initial_solution = 0
step_size = 0.1
max_iterations = 100

optimal_solution, optimal_value = steepest_ascent_hill_climbing(initial_solution, step_size, max_iterations)
print("Optimal solution:", optimal_solution)
print("Optimal value:", optimal_value)
```

[3] ✓ 0.3s

Optimal solution: 2.0000000000000004  
Optimal value: 1.0

Python

main\* 0 AWS: AWS Builder ID CodeWhisperer Spaces: 4 CRLF Cell 2 of 2 Go Live 10:23 PM 4/30/2024