C950 Task-2 WGUPS Write-Up

Vincent Messina

WGU Email: vmessi1@wgu.edu

7/26/25

C950 Data Structures and Algorithms II

# A. Hash Table

# 

# 

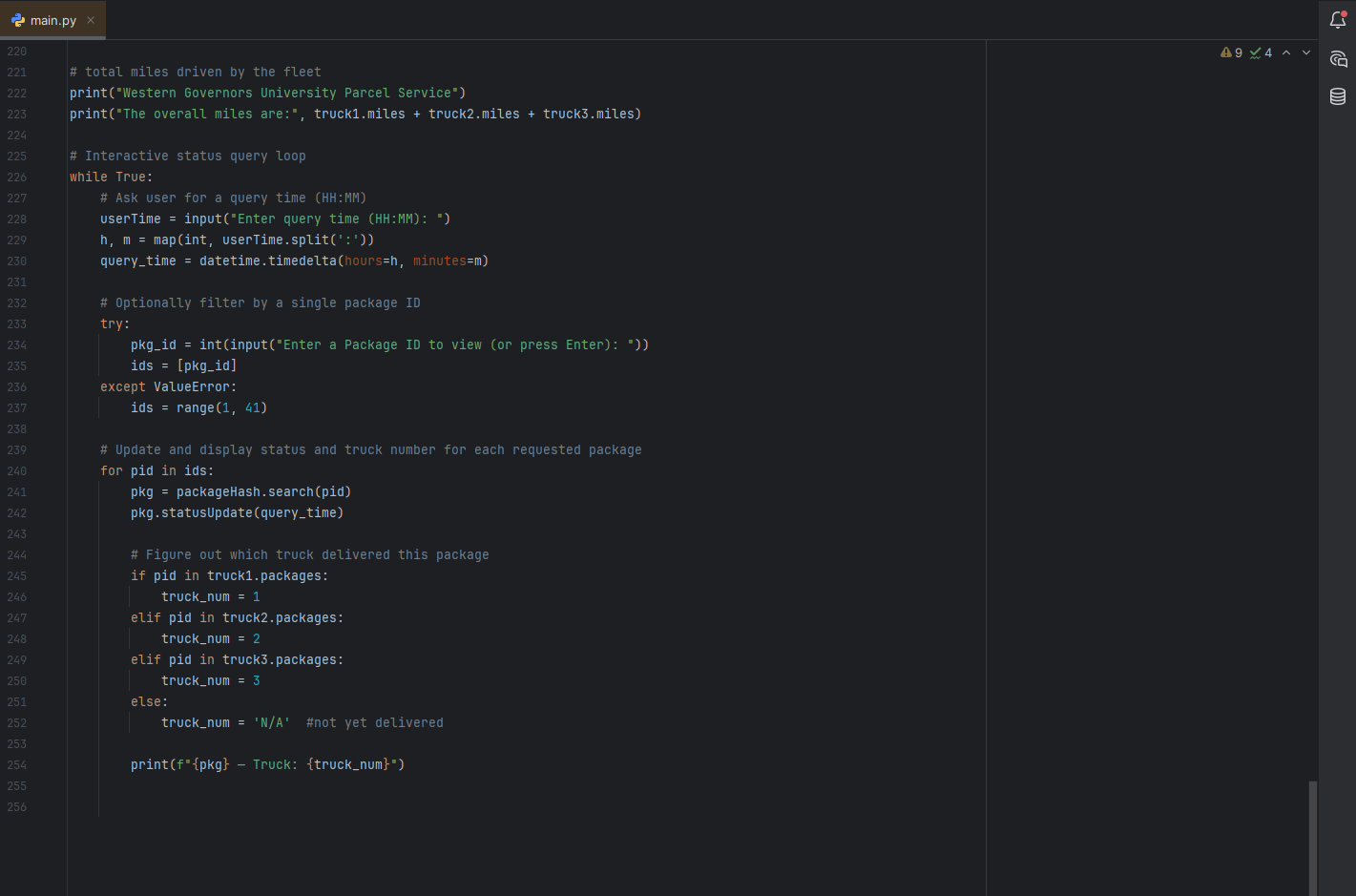
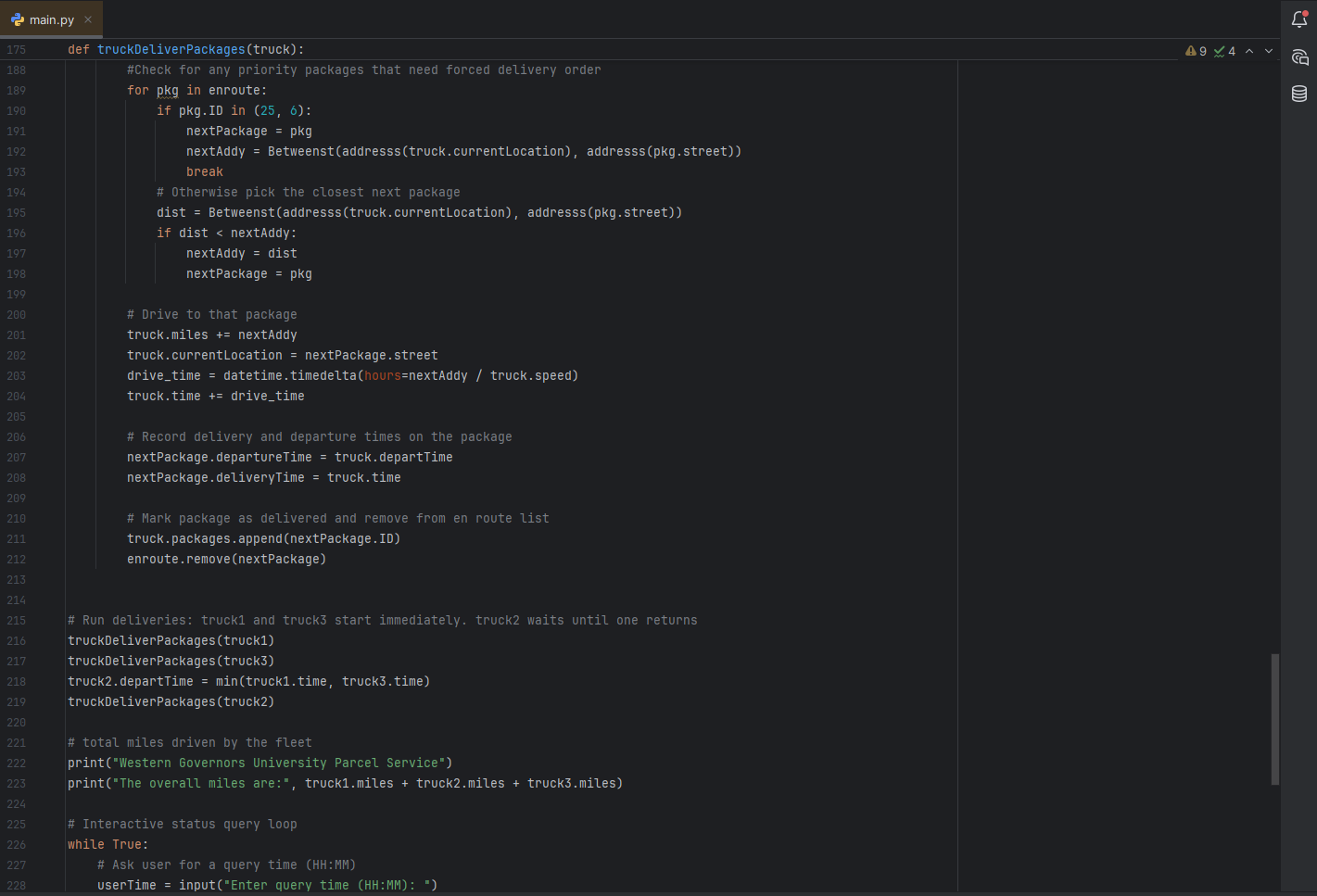
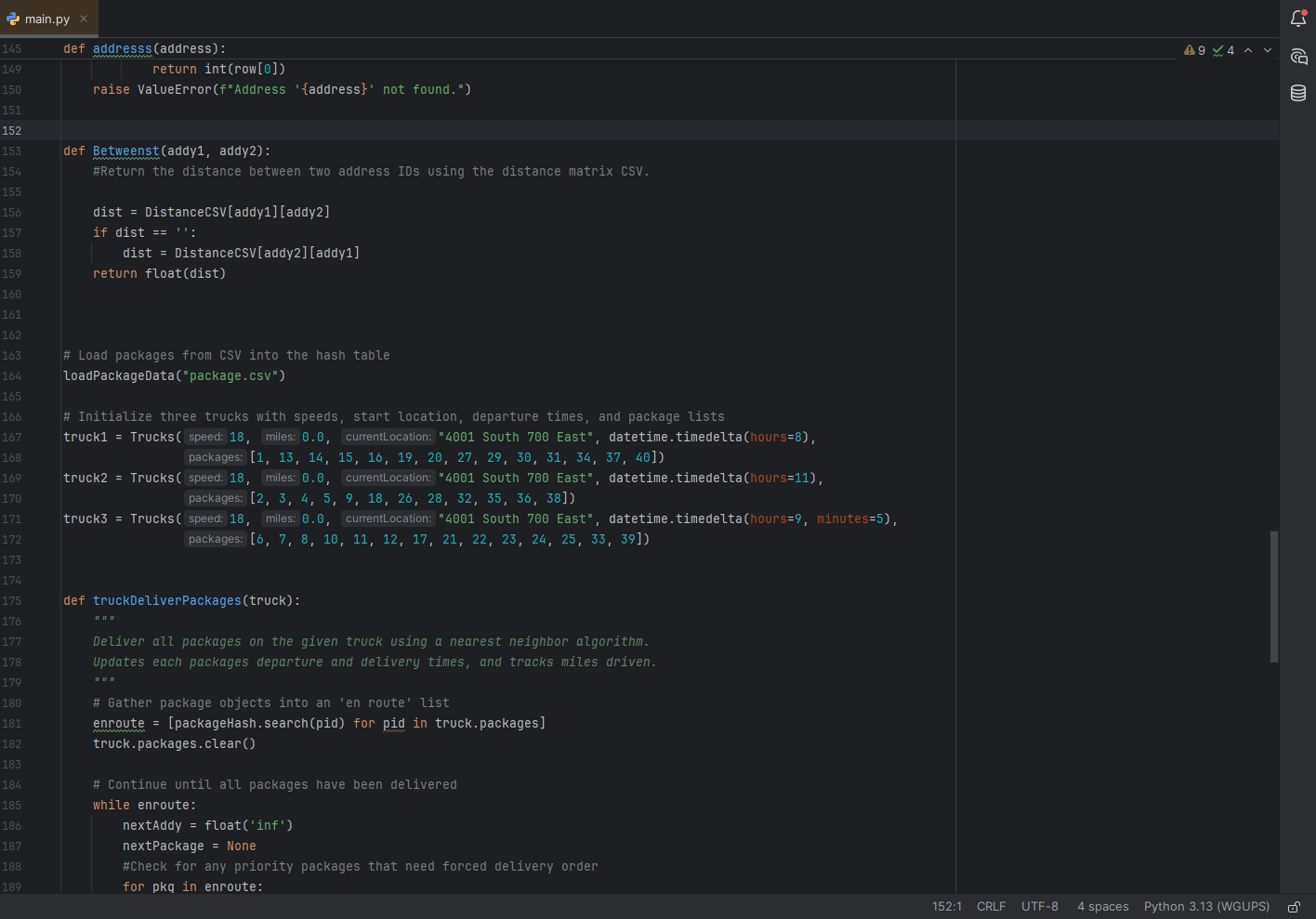
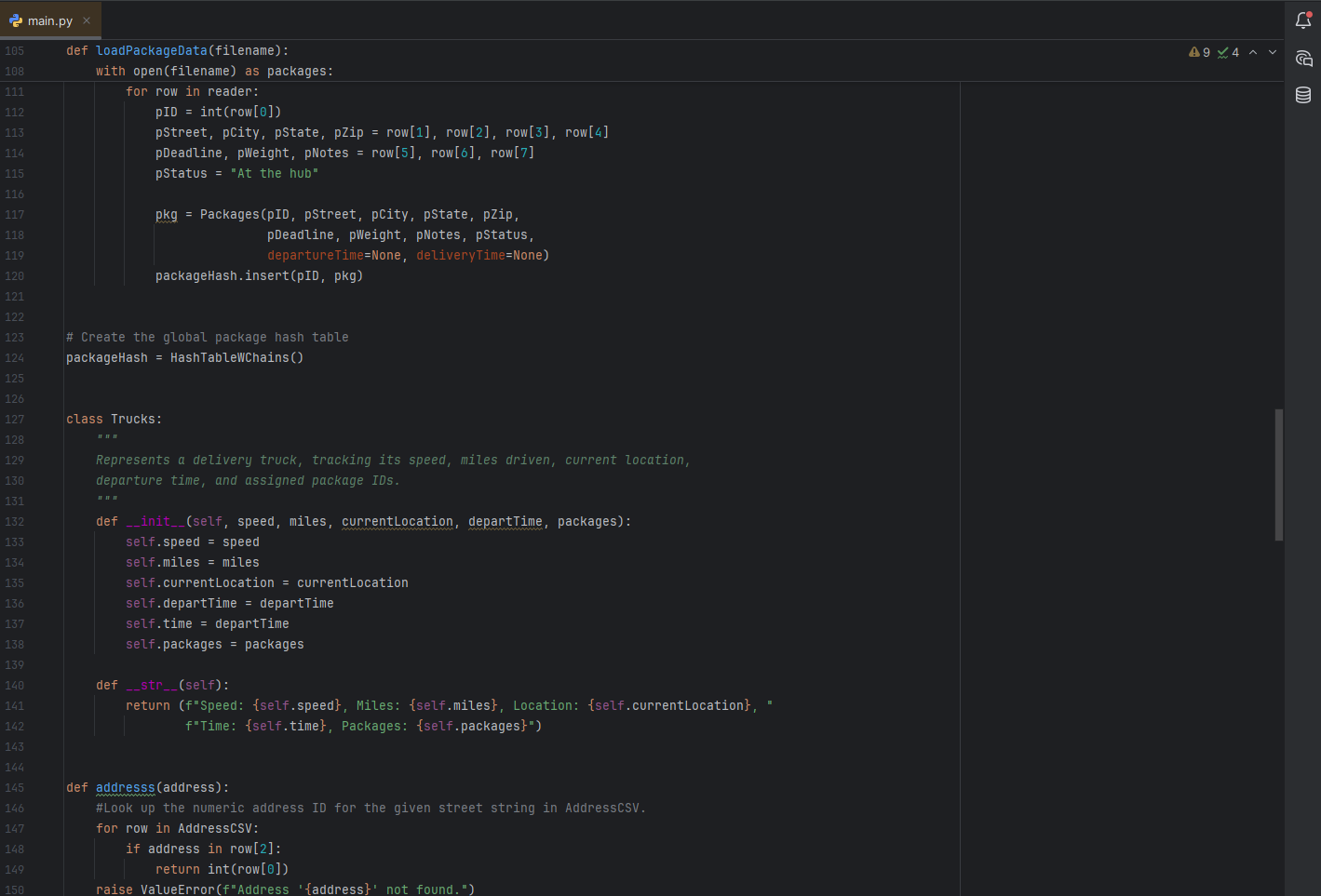
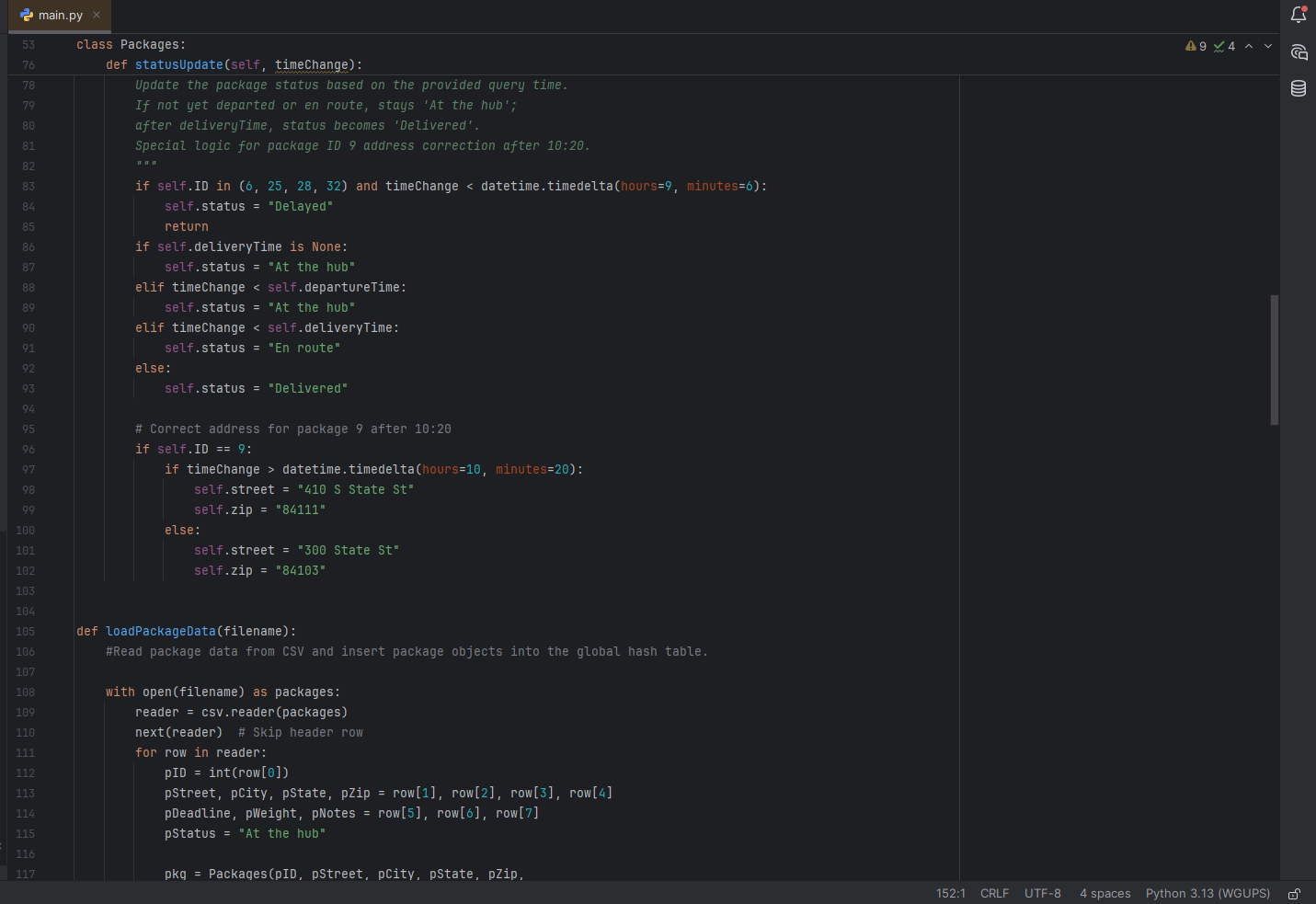
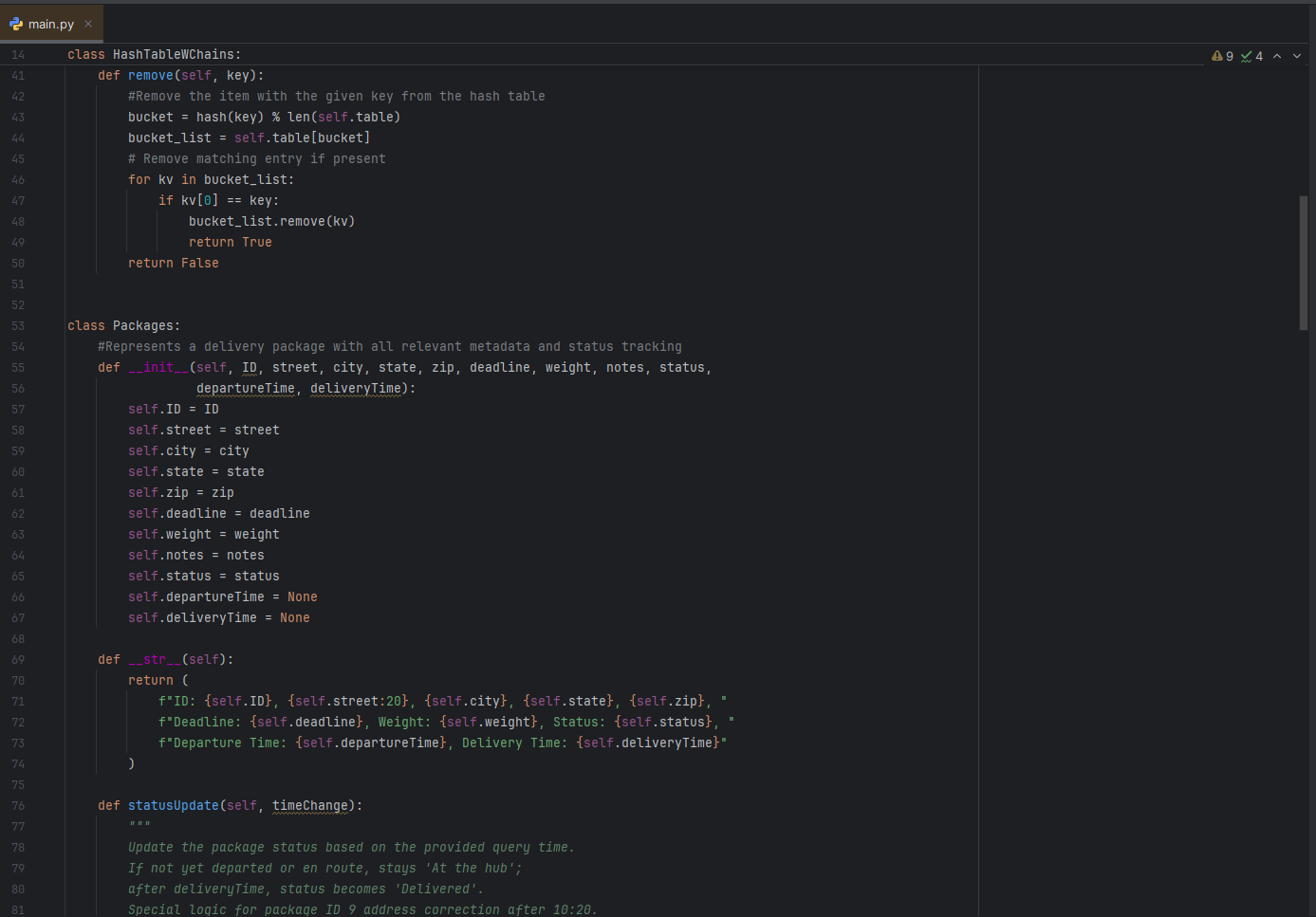
# B. Look-Up Functions

I named my look-up function “search”

# 

# C. Original Code

# 



# C1. Identification Information

# 

# The fist line of code includes my student ID.

# C2. Process and Flow Comments

# 

# D. Interface

# 

# D1. First Status Check

First Status check for 09:00AM

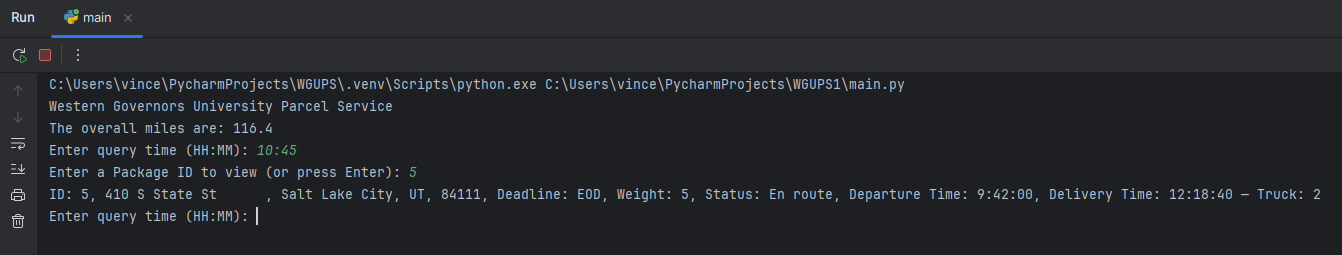
# D2. Second Status Check

Second Status Check for 10:00AM

# D3. Third Status Check

Third Status Check for 12:30PM

# E. Screenshot of Code Execution



Execution of Code

# F1. Strengths of the Chosen Algorithm

One strength of the Nearest Neighbor Algorithm is it’s simplicity and ease of implementation. The algorithm chooses the closest undelivered package and goes there next.

Another strength of the Nearest Neighbor Algorithm it is fast. It runs in a complexity of O(n^2)

# F2. Verification of Algorithm

This scenario required me to do all the following: deliver every package (1-40), track departure and delivery times, handle an address correction for package 9 after 10:20AM, handles delayed packages 6, 25, 28, and 32, allow querying status at different times, compute the total miles driven for all three trucks. And use priority deliveries for certain package IDs.

The algorithm visits every package once per truck, records each packages departure time and delivertime, applies the logic switch for package 9’s address and zip, prints total miles, and forces packages 25 and 6 to be delivered next if they remain on the truck. All the scenario requirements were satisfied.

# F3. Other Possible Algorithms

Two other algorithms that could have worked for this scenario are the Clarke-Wright Savings Algorithm and Christofides Approximation Algorithm.

# F3a. Algorithm Differences

There are key differences between each algorithm.

|  |  |  |  |
| --- | --- | --- | --- |
| Feature | Nearest Neighbor | Clarke-Wright Savings | Christofides |
| Approach | Greedy local decision only | Builds routes by merging savings from combining route pairs | Builds a minimum spanning tree and perfect matching |
| Quality of Solution | Suboptimal but fast | Better than greedy | Best |
| Complexity | O(n^2) | O(n^2logn) | O(n^3) |

# G. Different Approach

There are a few things I would do differently. I would implement time windows and deadlines instead of ignoring deadlines in routing to make sure the packages are dropped off before the deadline.

Another thing I would do differently would be to pre cluster deliveries by neighborhoods. Then have the trucks map the appropriate math using Christofides Algorithm.

# H. Verification of Data Structure

The Hash Table needed to support O(1) average insert/search/remove, handle collisions, and allow fast lookup by package ID. I inserted all 40 packages, each lookup by ID search take O(1) average, and chaining ensures there are no lost entries in case of collision. All requirements are met.

# H1. Other Data Structures

Two other data structures that could have worked in this scenario are: Direct Address Table (Array), and Binary Search Tree.

# H1a. Data Structure Differences

|  |  |  |
| --- | --- | --- |
| Data Structure | Pros | Cons |
| Hash Table | -O(1) average  - Space efficient for key ranges  - Simple collision handling | - Worst case O(n)  - Requires a good hash function  - No ordering of keys |
| Array | * O(1) worst case * Easy implementation | * Wastes memory if ID range is sparse * Must pre allocate |
| Binary Search Tree | * O(log n) worst case * Maintains keys in a sorted order` | * Slower than hash on average * More complex insert/remove |