Recommended Project Implementation Steps in the **Python Programming Language**

These are ONLY recommendations!! Feel free to use your own approach if you like!!

**A. Package-Data steps:**

1. Create *HashTable* data structure [(See C950 - Webinar-1 - Let’s Go Hashing webinar)](https://srm--c.na127.visual.force.com/apex/CourseArticle?id=kA03x000000spz6CAA&groupId=&searchTerm=&courseCode=C950&rtn=/apex/CommonsExpandedSearch).
2. Create a *Package* object with variables such as *id, address, city, state, zip, deliveryTime, date, status.*
3. Create a *Truck* object with variables such as *currentLocation, packages*.
4. Have *packageCSV,* *distanceCSV* and *addressCSV*files ready. (These are the CSV files in the ZIP folder attached to the Welcome Email)
5. Define the function *loadPackageData(HashTable)* to read the *packageCSV* file. You would need to import the *csv* package in Python. Here is a [VIDEO](https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwiM-Nbwovj8AhXTk4kEHVsLAjcQtwJ6BAgEEAI&url=https%3A%2F%2Fwww.youtube.com%2Fwatch%3Fv%3DefSjcrp87OY&usg=AOvVaw1sVweoJ86N4ZxzeCbJsAvx) that is helpful.
6. Insert all your packages (*Package* objects) into *HashTable* with *key = PackageID* and *item = Package Information*.

**B. Distance-Data steps:**

1. Upload Distances:
   1. Create the *distanceData* list in Python via *distanceData = [ ].*
   2. Define *loadDistanceData( )* function to read the *distanceCSV* file, row-wise.
   3. Append each row to *distanceData*, making it a two-dimensional list or a list of lists.
2. Upload Addresses:
   1. Create the *addressData* list in Python via *addressData = [ ].*
   2. Define loadAddressData( ) function to read *addressCSV* file by row.
   3. Read each row of the addresses and append to the *addressData* list.

**C. Load Packages Steps and Algorithm:**

1. Define a function to return the distance between two addresses or the function *distanceBetween(address1, address2)*. This function will use the *distanceData* list from B.1. above and return the distance via *distanceData[addressData.index(address1)][addressData.index(address2)]*.
2. Define a function that returns the package destination address, from among all the *packages* that are currently on a truck at that time, that is closest to the truck’s *currentLocation*. This could be a function *minDistanceFrom(truck)* and could be implemented using the *distanceBetween* function from C.1.
3. Define a function *loadPackages(truck)* for example that loads packages into a truck based on the provided constraints in section F below. One can hand load packages into the trucks based on the constraints in section F for example.

**D. Deliver Packages Steps and Algorithm:**

1. Define a function such as *truckDeliverPackages(truck)* for example to deliver packages in a truck. Here, one can loop through all the truck packages and use the *minDistanceFrom* function from C.2. for all the addresses the truck has not visited yet.
2. Update the delivery status and time delivered for each delivered package in the Hash Table.
3. Keep track of the total mileage for each truck. Remember the truck travels at 18 miles per hour or 0.3 miles per minute. One may need to import the *datetime* package from Python to help us here.

**E. Command Line User Interface (UI) to Interact with the Users:**

Create a Command Line User-Interface (UI) to interact and report the results based on the requirements. An example follows below:

Possible Menu Options:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. Print All Packages’ Information (including Statuses [“At Hub”, “En Route”, “Delivered’]) and Total Mileage of all Trucks
2. Print All Information (including Status) of a given Package at a Given Time
3. Print All Information (including Statuses) of All the Packages at a Given Time
4. Exit the Program

Possible output example:

Package ID: 1, Address: 195 W Oakland Ave, ... Delivered at 08:46:20

Package ID: 2, Address: 2530 S 500 E, ... AtHub

Package ID: 3, Address: 233 Canyon Rd, ... InRoute

....................................

....................................

....................................

**F. Constraints and Objectives of Project**

1. The goal is to meet all constraints while trying to minimize total truck mileage. If the total mileage is more than 140 miles, we do not have a sufficiently efficient solution. Both heuristics and optimizing algorithms are important to get a good solution.
2. Two drivers and three trucks are available. So, no more than two trucks can be away from the hub at the same time.
3. The trucks move at a constant speed of 18 miles per hour or 0.3 miles per minute.
4. Each truck can carry a maximum of 16 packages.
5. Trucks can leave the hub no sooner than 8:00 a.m.
6. Trucks can be loaded only at the hub.
7. You only need to account for the time spent driving. You can ignore the time spent on all other activities, such as loading trucks and dropping off packages.
8. The wrong delivery address for package #9, Third District Juvenile Court, will be corrected at 10:20 a.m. The correct address is “410 S State St., Salt Lake City, UT 84111”. You may assume that WGUPS knows the address is correct and when the correction will be available. You can treat this package as a delayed package.
9. Packages #13, #14, #15. #16, #19, and #20 must go out for delivery on the same truck.
10. Packages #3, #18, #36, and #38 may only be delivered by truck 2.
11. Packages #6, #25, #28, #32 arrived late on a flight and are not available to leave the hub before 9:05 a.m.
12. Packages #1, 6, 13, 14, 16, 20, 25, 29, 30, 31, 34, 37 and 40 need to be delivered on or before 10:30 am.
13. Package #15 needs to be delivered on or before 9:00 am.
14. Packages #2-5, #7-12, #17-19, #21-24, #26-28, #32-33, #35-36, and #38-39 can be delivered by the end of the day (EOD) which would be 5:00 pm, so there can be some flexibility exercised as regards these packages.