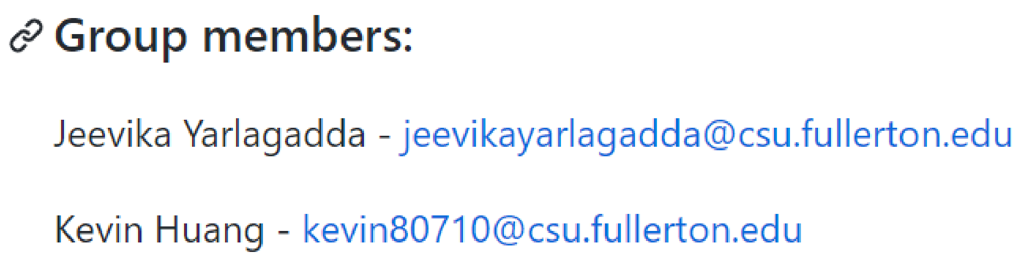
CPSC 535 Project Report

Project: Electric Car Traveler

Group members:



Group members:

Jeevika Yarlagadda - jeevikayarlagadda@csu.fullerton.edu

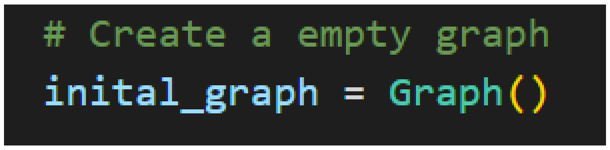
Kevin Huang - kevin80710@csu.fullerton.edu

Summary:

Electric cars need to be charged periodically. Say if we want to travel long distance from Santa Monica, CA to Tallahassee,FL. If we charge the car at the starting city we don't get it till our destination. It has to be recharged in some other stops on the way. If any of the charge stations doesn't work, we should be able to travel back to the previous station. Here is an algorithm to get the list of cities to be halted to get the car recharged till it reaches the destination.

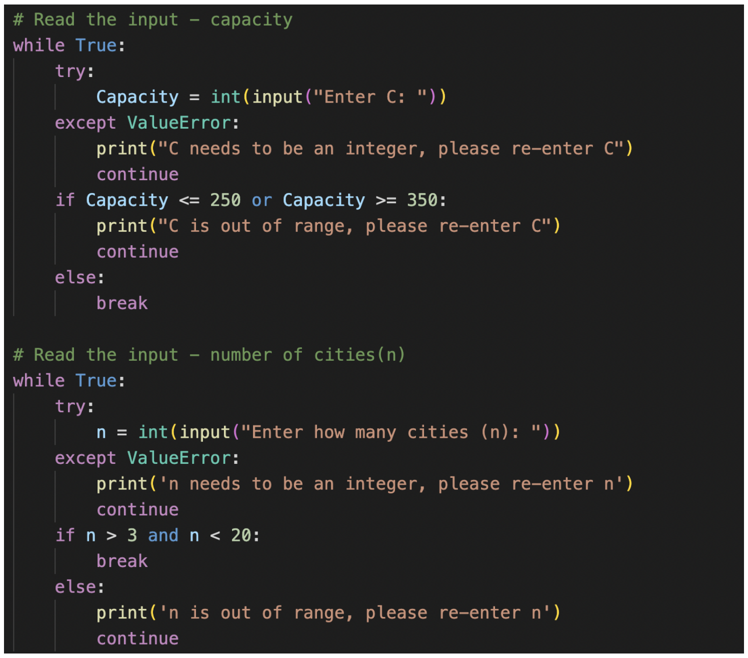
Algorithm and Pseudocode:

1. Initialize an empty graph using Graph() available in python

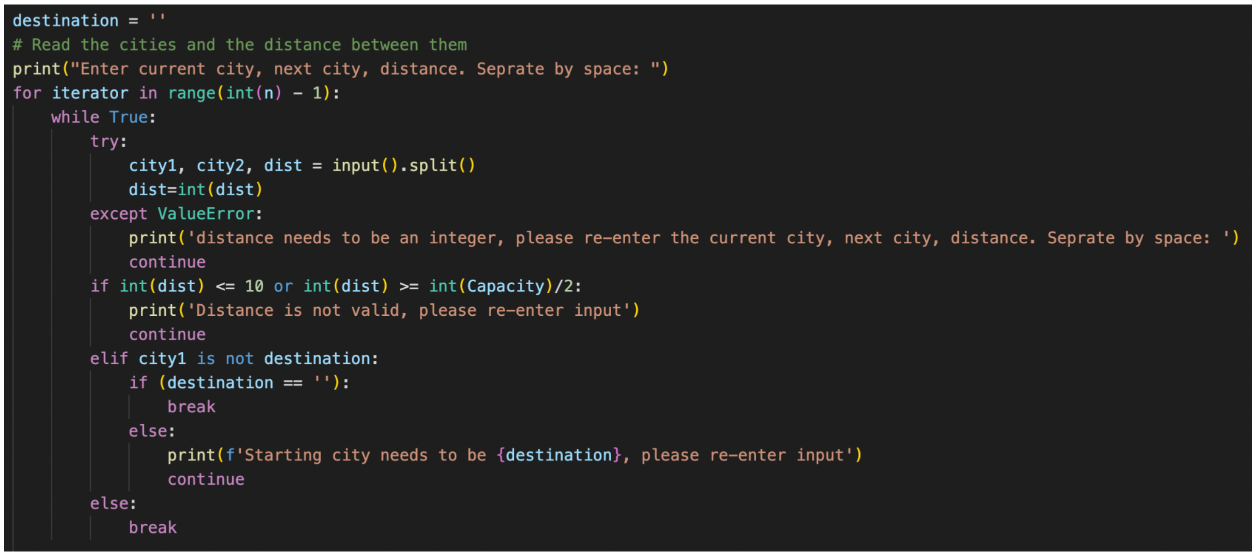


```
# Create a empty graph
inital_graph = Graph()
```

2. Read the inputs: C, n, cities and distances between them

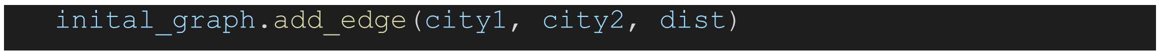


```
# Read the input - capacity
while True:
    try:
        Capacity = int(input("Enter C: "))
    except ValueError:
        print("C needs to be an integer, please re-enter C")
        continue
    if Capacity <= 250 or Capacity >= 350:
        print("C is out of range, please re-enter C")
        continue
    else:
        break

# Read the input - number of cities(n)
while True:
    try:
        n = int(input("Enter how many cities (n): "))
    except ValueError:
        print('n needs to be an integer, please re-enter n')
        continue
    if n > 3 and n < 20:
        break
    else:
        print('n is out of range, please re-enter n')
        continue
```



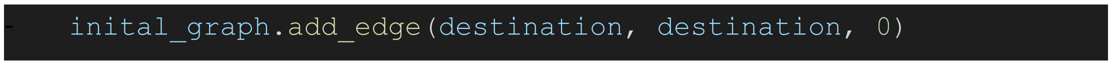
```
destination = ''
# Read the cities and the distance between them
print("Enter current city, next city, distance. Seprate by space: ")
for iterator in range(int(n) - 1):
    while True:
        try:
            city1, city2, dist = input().split()
            dist=int(dist)
        except ValueError:
            print('distance needs to be an integer, please re-enter the current city, next city, distance. Seprate by space: ')
            continue
        if int(dist) <= 10 or int(dist) >= int(Capacity)/2:
            print('Distance is not valid, please re-enter input')
            continue
        elif city1 is not destination:
            if (destination == ''):
                break
            else:
                print(f'Starting city needs to be {destination}, please re-enter input')
                continue
        else:
            break
```

3. Update the graph with the cities as nodes and the distance as an edge



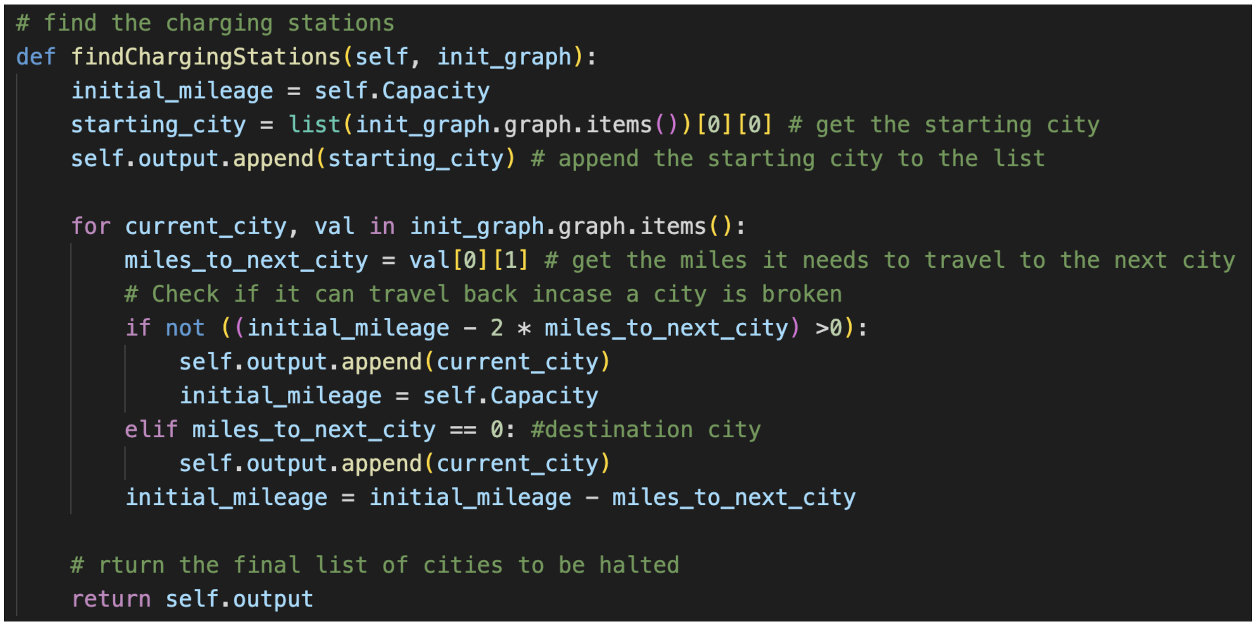
```
    inital_graph.add_edge(city1, city2, dist)
```

- When last city is input, add the destination edge to itself with the mileage as 0



```
    inital_graph.add_edge(destination, destination, 0)
```

4. Run a for loop by taking each node and its distance to next node
    a. Check if we can travel back if the current station is not working, If you cannot travel back from the next station to the current station then it is the place where he needs the car to be recharged
    b. Add to the output (a list that keeps track of cities where we need to halt)
    c. Update the mileage to the given initial capacity
    d. If the mileage to next city is 0, it means that the city is the destination
    e. Add the city to the output list
    f. Update the capacity of car once it crosses a station



```
# find the charging stations
def findChargingStations(self, init_graph):
    initial_mileage = self.Capacity
    starting_city = list(init_graph.graph.items())[0][0] # get the starting city
    self.output.append(starting_city) # append the starting city to the list

    for current_city, val in init_graph.graph.items():
        miles_to_next_city = val[0][1] # get the miles it needs to travel to the next city
        # Check if it can travel back incase a city is broken
        if not ((initial_mileage - 2 * miles_to_next_city) >0):
            self.output.append(current_city)
            initial_mileage = self.Capacity
        elif miles_to_next_city == 0: #destination city
            self.output.append(current_city)
        initial_mileage = initial_mileage - miles_to_next_city

    # rturn the final list of cities to be halted
    return self.output
```

Pseudocode for algorithm:

```
Function findChargingStations:
    Input: graph contains cities with distance between previous and next cities
    Output: list contains minimum cities

    Output = List[]
    Initial_mileage = input Capacity
    starting_city = first key in graph
    Output add starting_city
    For current_city, distance_to_next_city in graph:
        If (Initial_mileage - 2*distance_to_next_city not > 0):
            Output add current city
            Initial_mileage = C
```

```
          Elif distance_to_next_city is 0:
                    Output add current_city
          Initial_mileage -= distance_to_next_city
Endfor
```

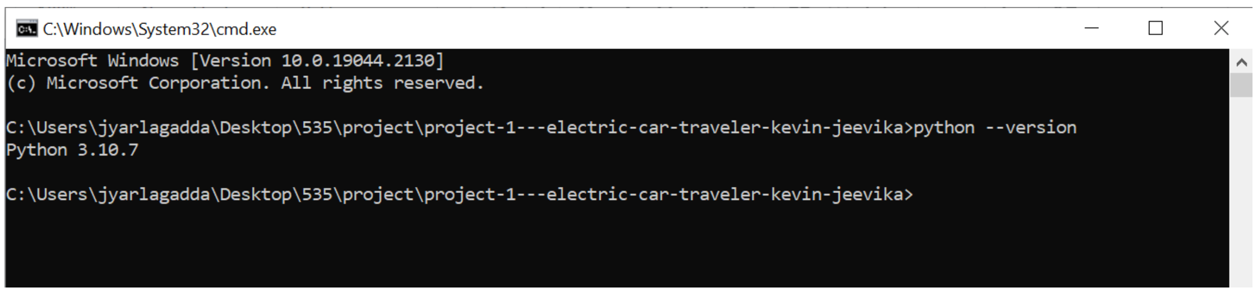
Time complexity would be O(n), it has one for-loop which iterates n times

Steps to Execute the code:

We run the code using Python
Expected version of Python:
Python 3.10.7

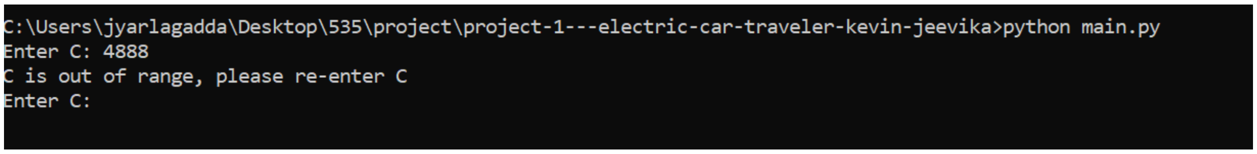
- can check this using the command: python --version on the terminal

1. Please download the zip file of the code and open a command prompt in the main.py file location



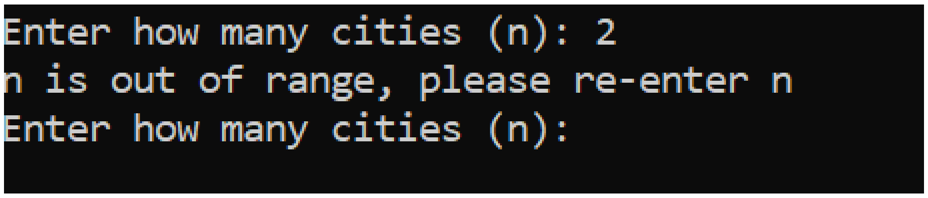
```
Microsoft Windows [Version 10.0.19044.2130]
(c) Microsoft Corporation. All rights reserved.

C:\Users\jyarlagadda\Desktop\535\project\project-1---electric-car-traveler-kevin-jeevika>python --version
Python 3.10.7

C:\Users\jyarlagadda\Desktop\535\project\project-1---electric-car-traveler-kevin-jeevika>
```

2. Run the code using the command: python main.py
3. You will see a screen that prompts the user to enter C. Please enter the value of C.
   If you enter the value of C which is out of range, it asks you to re-enter C



```
C:\Users\jyarlagadda\Desktop\535\project\project-1---electric-car-traveler-kevin-jeevika>python main.py
Enter C: 4888
C is out of range, please re-enter C
Enter C:
```

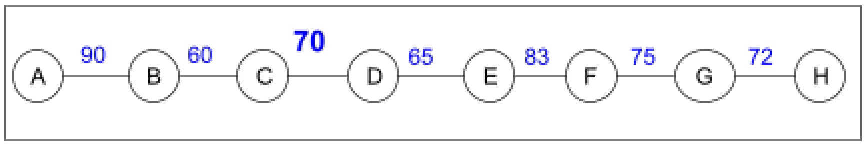
4. Then, please enter the value of n
   If you enter the value of n which is out of range, it asks you to re-enter n



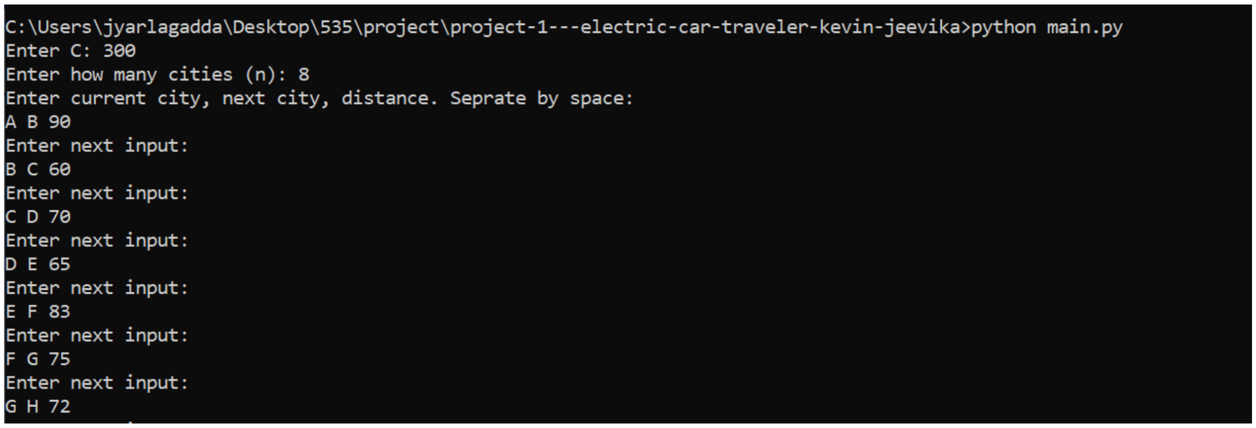
```
Enter how many cities (n): 2
n is out of range, please re-enter n
Enter how many cities (n):
```

5. Now, please enter the current city, the destination city and the distance. All of these should be separated by space. A sample is shown below:

   For a graph like this:

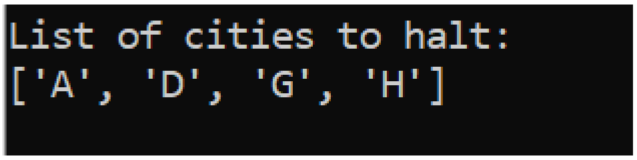


   This is how the input is to be given:



```
C:\Users\jyarlagadda\Desktop\535\project\project-1---electric-car-traveler-kevin-jeevika>python main.py
Enter C: 300
Enter how many cities (n): 8
Enter current city, next city, distance. Seprate by space:
A B 90
Enter next input:
B C 60
Enter next input:
C D 70
Enter next input:
D E 65
Enter next input:
E F 83
Enter next input:
F G 75
Enter next input:
G H 72
```

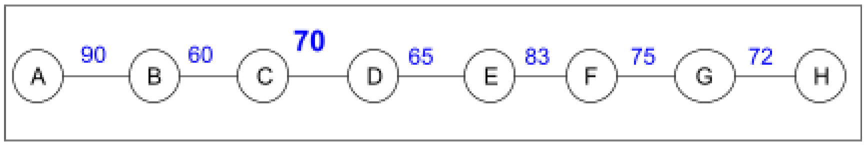
6. Once the input is done, we get the output as a list. A sample is here:

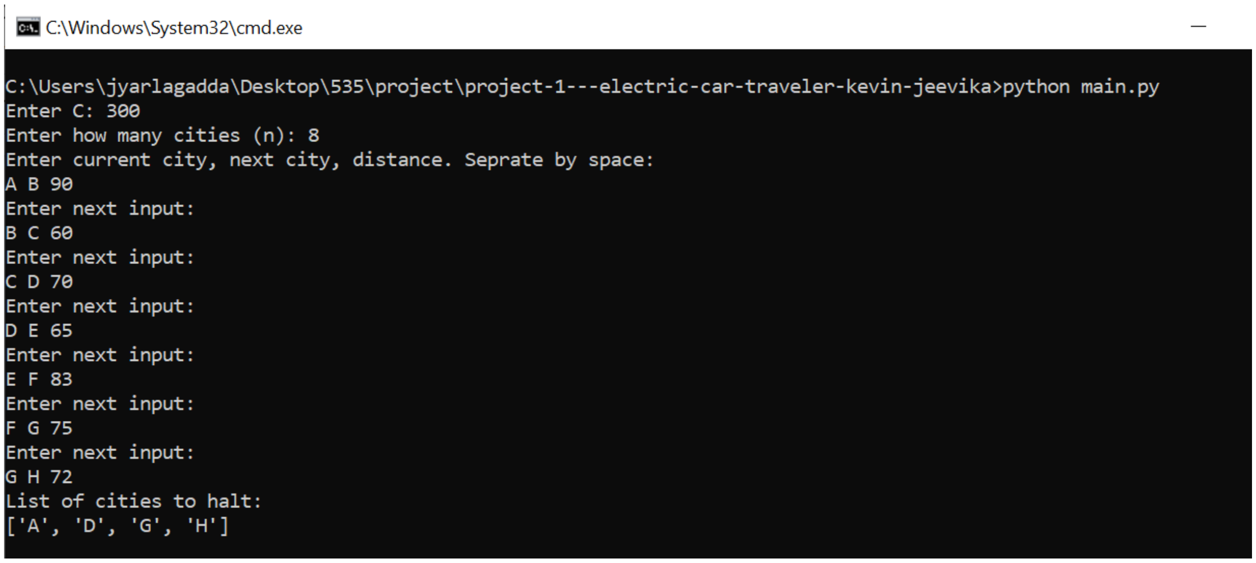


```
List of cities to halt:
['A', 'D', 'G', 'H']
```

Testcases:

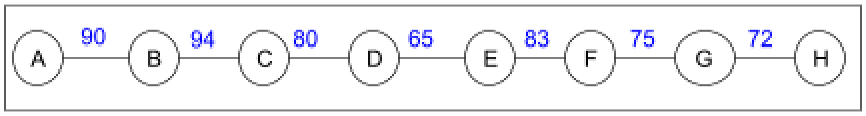
Sample 1:

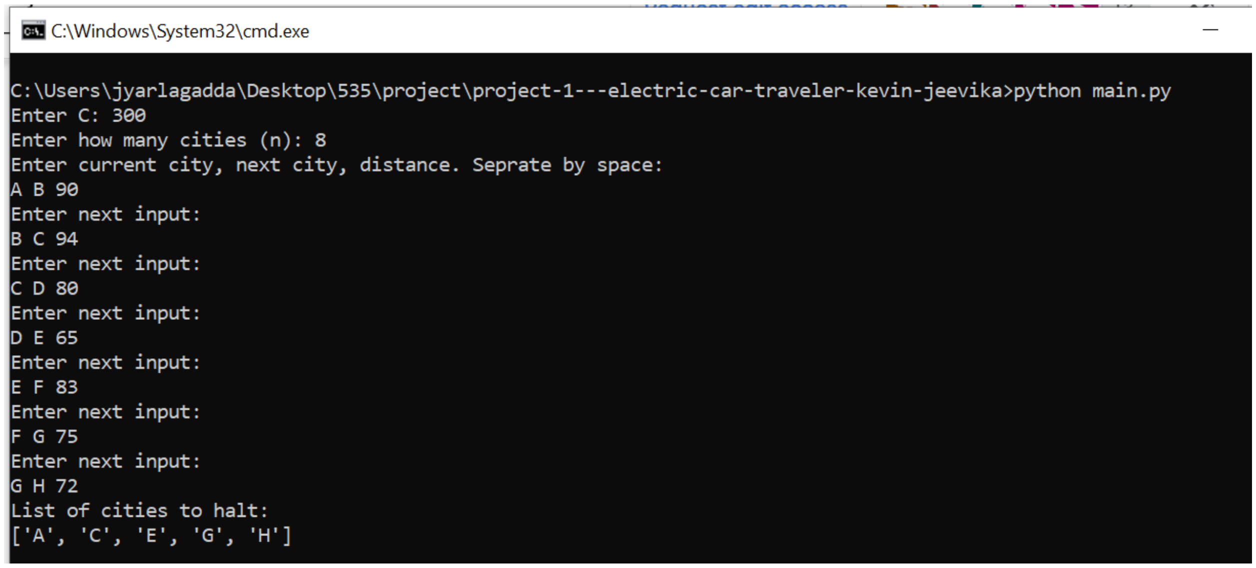



C:\Windows\System32\cmd.exe

```
C:\Users\jyarlagadda\Desktop\535\project\project-1---electric-car-traveler-kevin-jeevika>python main.py
Enter C: 300
Enter how many cities (n): 8
Enter current city, next city, distance. Seprate by space:
A B 90
Enter next input:
B C 60
Enter next input:
C D 70
Enter next input:
D E 65
Enter next input:
E F 83
Enter next input:
F G 75
Enter next input:
G H 72
List of cities to halt:
['A', 'D', 'G', 'H']
```

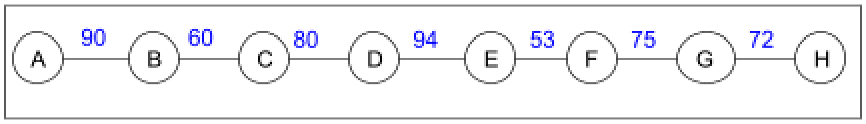
Sample 2:

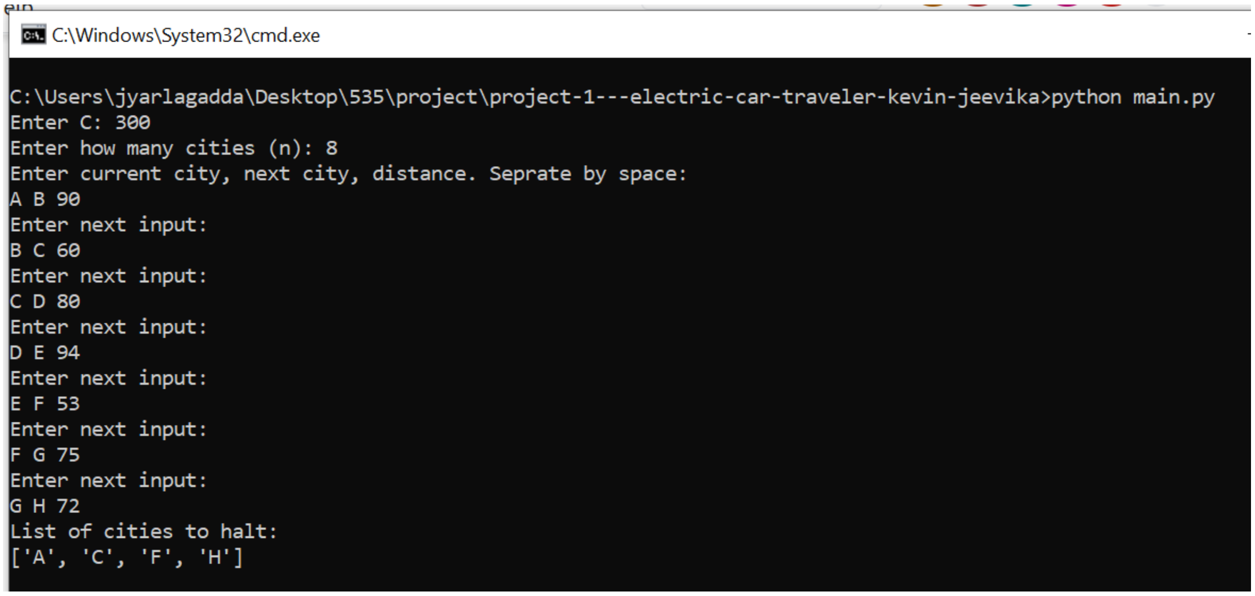



C:\Windows\System32\cmd.exe

```
C:\Users\jyarlagadda\Desktop\535\project\project-1---electric-car-traveler-kevin-jeevika>python main.py
Enter C: 300
Enter how many cities (n): 8
Enter current city, next city, distance. Seprate by space:
A B 90
Enter next input:
B C 94
Enter next input:
C D 80
Enter next input:
D E 65
Enter next input:
E F 83
Enter next input:
F G 75
Enter next input:
G H 72
List of cities to halt:
['A', 'C', 'E', 'G', 'H']
```

Sample 3:





C:\Windows\System32\cmd.exe

```
C:\Users\jyarlagadda\Desktop\535\project\project-1---electric-car-traveler-kevin-jeevika>python main.py
Enter C: 300
Enter how many cities (n): 8
Enter current city, next city, distance. Seprate by space:
A B 90
Enter next input:
B C 60
Enter next input:
C D 80
Enter next input:
D E 94
Enter next input:
E F 53
Enter next input:
F G 75
Enter next input:
G H 72
List of cities to halt:
['A', 'C', 'F', 'H']
```