Foundation Certificate in Higher Education

Module: DOC 333-Introduction to Programming Principles

Module Leader: Mr. Nishan Saliya Harankahawa

Assessment Type: Individual Course Work

Work Title: The Kingdom of Miranda

Student Details:

Student ID	Student Name
20222171	Disini Ruhansa Kodagoda Hettige

I. Abstract

"DropMe" is a console application designed for the Kingdom of Miranda, offering a cab service with Trishaw, Car, and Van options for travel between its five main cities. The application allows users to view trip details, apply promo codes, and enjoy occasional auto-generated promotions. Invoices are generated after each ride, providing date, time, start and end cities, total amount, applied promo code, random reduction (if any), and the final payment. The application aims to provide efficient and convenient transportation for passengers in the Kingdom of Miranda.

II. Acknowledgements

I would like to take this opportunity to express my sincere gratitude to all the programming lecturers for providing valuable guidance. Their expertise and knowledge have been instrumental in shaping the direction of this report and helping me navigate the various challenges that I have encountered along the way. I would like to express my gratitude to my family, whose unwavering support, love, and encouragement have been a constant source of motivation and inspiration throughout my academic journey. I am truly grateful for their presence in my life. This report may have not become a reality if it was not for the support I received.

Table of Contents

List of Figures

The Scenario

The “The Kingdom of Miranda” is a circular shaped small island with 5 main cities named as Alvin (Capital), Jamz, Razi, Mali and Zuhar.

The official currency of the country is KMD (Kingdom of Miranda Dollars). The country uses a single cab service known as DropMe™. The DropMe™ price chart for travelling is shown below. All the prices are in KMD.

City Name	Alvin	Jamz	Razi	Mali	Zuhar
Alvin	0	20	40	40	20
Jamz	20	0	20	40	40
Razi	40	20	0	20	40
Mali	40	40	20	0	20
Zuhar	20	40	40	20	0

The country uses only Trishaws, Cars and Vans for cab services. The default transport is Trishaws while passengers can ask for cars or vans if they prefer. Car prices have doubled while van prices have tripled. The vehicles will always take the shortest path for traveling.

The president of The Kingdom of Miranda needs us to develop a console application which will allow users to view trip details and generate invoices. The application is used by the passenger.

The promo codes are available under especial circumstances or issued individually. The format of these codes is as below.

- pro2 (2 KMDs are reduced from the total bill)
- pro5 (5 KMDs are reduced from the total bill)
- pro10 (10 KMDs are reduced from the total bill)

The above three are samples while the promotion amounts can range from 1 KMD to maximum of 15 KMD.

Figure 1: Taxi service

The application itself will generate price reductions for customers who use the application. These auto-generated promotions are always at a fixed price of 5 KMD. These are not taking place if the passenger uses a promotional code. This feature is a random feature! Only the lucky passengers will get this special promotion time-to-time.

Every time a passenger uses the app, the invoice shown will be saved in a text file.

1. Algorithm for the program

1. Import necessary modules
2. Initialize price_chart, promo_code and vehicle dictionaries. These dictionaries containing the prices between the cities, promotion codes with their discounts, the vehicle types and their prices.
3. Define the save function to save details and save those details in to a text file. (Date, time, start, end, amount, promo, random reduction, final payment)
4. Define the invoice_cal function to calculate the details of the invoice.
5. Define the invoice function to display the invoice details. (Start, end, amount, promo, random reduction, final payment)
6. Process user commands. This arguments are providing by the user when running the program.
7. Enter the error handling and closing message.

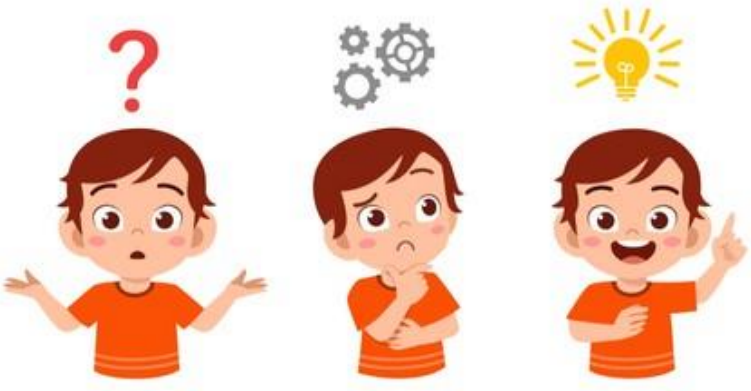

Figure 4: Thinking of an algorithm

2. Actual python codes for the program

To implement this program, there are two different python files.

- dm.py
- Function.py

dm.py file containing all the arguments and error handling messages while **function.py** containing all the functions that are help to calculate the invoice and write those details in a text file.

2.1. Python file 1 (dm.py)

```
#Import necessary modules
from datetime import date, datetime
import random
import function as f  #Import a custom module names 'function' as 'f'
import sys


# DropMe price chart for traveling
price_chart = {
    'alvin': {'alvin': 0, 'jamz': 20, 'razi': 40, 'mali': 40, 'zuhar': 20},
    'jamz': {'alvin': 20, 'jamz': 0, 'razi': 20, 'mali': 40, 'zuhar': 40},
    'razi': {'alvin': 40, 'jamz': 20, 'razi': 0, 'mali': 20, 'zuhar': 40},
    'mali': {'alvin': 40, 'jamz': 40, 'razi': 20, 'mali': 0, 'zuhar': 20},
    'zuhar': {'alvin': 20, 'jamz': 40, 'razi': 40, 'mali': 20, 'zuhar': 0}
    }


#Promo codes and their corresponding discounts
promo_code = {
    "/pro1" : 1,
    "/pro2" : 2,
    "/pro3" : 3,
    "/pro4" : 4,
```

```
    "/pro5" : 5,
    "/pro6" : 6,
    "/pro7" : 7,
    "/pro8" : 8,
    "/pro9" : 9,
    "/pro10" : 10,
    "/pro11" : 11,
    "/pro12" : 12,
    "/pro13" : 13,
    "/pro14" : 14,
    "/pro15" : 15
}
#Vehicle types and their corresponding costs
vehicle = {
    "/c": 2,
    "/v": 3
}

#Print DropMe title
print("****Drop-Me****")
try:
    # If the user runs the program with the argument '/?', show available commands
    if len(sys.argv) == 2 and sys.argv[1] == '/?':
        print("* Keep a space between every command")
        print("* /price to show the price chart")
        print("* vehicle: </c> for car    </v> for van")
        print("* promo code: </pro2> 2KDM, <pro5> 5KDM, <pro10> 10KDM")
        print("* <city_name> <space> <city_name> To get the ride with a Trishaw")
        print("* <city_name> <space> <city_name> <space> </c> To get the ride with a chosen vehicle")
        print("* <city_name> <space> <city_name> <space> </v> To get the ride with a chosen vehicle")
```

```
        print("* <city_name> <space> <city_name> <space> </v> <space> <promo_code> to add a promo code")
        print("* <city_name> <space> <city_name> <space> <promo_code> <space> </vehicle_letter> to add a promo code")
        print("* Do not use <> brackets in commands")

    #If the user runs the program with the argument '/price', display the price chart
    elif len(sys.argv) == 2 and sys.argv[1] == '/price':
        #Print the price chart for traveling via Trishaw
        print("Price Chart for Traveling (Trishaw):\n")
        cities = list(price_chart.keys())
        header_row = ["Cities"] + cities

        print("\t".join(header_row))

        for city in cities:
            row = [city] + [str(price_chart[city][dest]) for dest in cities]
            print("\t".join(row))

        #Print the price chart for traveling via Car
        print("Price Chart for Traveling (Car):\n")
        cities = list(price_chart.keys())
        header_row = ["Cities"] + cities

        print("\t".join(header_row))

        for city in cities:
            row = [city] + [str(price_chart[city][dest]*2) for dest in cities]
            print("\t".join(row))

        #Print the price chart for traveling via Van
```

```
    print("Price Chart for Traveling (Van):\n")
    cities = list(price_chart.keys())
    header_row = ["Cities"] + cities

    print("\t".join(header_row))

    for city in cities:
        row = [city] + [str(price_chart[city][dest]*3) for dest in cities]
        print("\t".join(row))


#If the user provides two city names as arguments, calculate the ride cost via Trishaw
elif len(sys.argv) == 3 and sys.argv[1].lower() in price_chart and sys.argv[2].lower() in price_chart:
    start = sys.argv[1].lower()
    end = sys.argv[2].lower()
    print(f.invoice(start,end))

#If the user provides three arguments with a vehicle type or promo code, calculate the ride cost accordingly
elif len(sys.argv) == 4 and sys.argv[1].lower() in price_chart and sys.argv[2].lower() in price_chart:
    if sys.argv[3].lower() in vehicle:
        start = sys.argv[1].lower()
        end = sys.argv[2].lower()
        vehi = sys.argv[3].lower()
        print(f.invoice(start,end,vehi))
    elif sys.argv[3] in promo_code:
        start = sys.argv[1].lower()
        end = sys.argv[2].lower()
        vehi = ""
        promoc = sys.argv[3].lower()
```

```
            print(f.invoice(start,end,promoc))


    #If the user provides four arguments with a combination of vehicle type and promo code, calculate the cost
    elif len(sys.argv) == 5 and sys.argv[1].lower() in price_chart and sys.argv[2].lower() in price_chart:
        if sys.argv[3].lower() in vehicle and sys.argv[4].lower() in promo_code:
            start = sys.argv[1].lower()
            end = sys.argv[2].lower()
            vehi = sys.argv[3].lower()
            promoc = sys.argv[4].lower()
            print(f.invoice(start,end,promoc,vehi))
        elif sys.argv[4].lower() in vehicle and sys.argv[3].lower() in promo_code:
            start = sys.argv[1].lower()
            end = sys.argv[2].lower()
            vehi = sys.argv[4].lower()
            promoc = sys.argv[3].lower()
            print(f.invoice(start,end,promoc,vehi))


    else:
        # If the user doesn't provide valid arguments, show the help message
        print("* Enter /? To show all the commands")


except :
    # If any error occurs during the execution, show the help message
    print("* Enter /? To show all the commands")


finally :
    # Print the DropMe title at the end
    print("****Drop-Me****")
```

2.2. Python file 2 (functions.py)

```
#Import necessary modules
import random
from datetime import date, datetime
import os

#Price chart for traveling between
price_chart = {
    'alvin': {'alvin': 0, 'jamz': 20, 'razi': 40, 'mali': 40, 'zuhar': 20},
    'jamz': {'alvin': 20, 'jamz': 0, 'razi': 20, 'mali': 40, 'zuhar': 40},
    'razi': {'alvin': 40, 'jamz': 20, 'razi': 0, 'mali': 20, 'zuhar': 40},
    'mali': {'alvin': 40, 'jamz': 40, 'razi': 20, 'mali': 0, 'zuhar': 20},
    'zuhar': {'alvin': 20, 'jamz': 40, 'razi': 40, 'mali': 20, 'zuhar': 0}
    }

# Promo codes and their corresponding discounts
promo_code = {
    "/pro1" : 1,
    "/pro2" : 2,
    "/pro3" : 3,
    "/pro4" : 4,
    "/pro5" : 5,
    "/pro6" : 6,
    "/pro7" : 7,
    "/pro8" : 8,
    "/pro9" : 9,
    "/pro10" : 10,
    "/pro11" : 11,
    "/pro12" : 12,
```

```
    "/pro13" : 13,
    "/pro14" : 14,
    "/pro15" : 15
}
#Vehicle types and their corresponding costs
vehicle = {
    "/c": 2,
    "/v": 3
}



#Function to save the invoice details to a text file
def save(start,end,first_price,promo,random_reduct,price):
    fn = date.today()
    if not os.path.exists("invoices"):
        os.makedirs("invoices")
    filename = datetime.now().strftime("%Y-%m-%d %H_%M_%S_%f") + ".txt"
    file_path = os.path.join("invoices", filename)

    with open(file_path, "w") as f:
        f.write("Date              : " + str(date.today()) + "\n")
        f.write("Time              : " + datetime.now().strftime("%H:%M:%S") + "\n")
        f.write("Start             : " + start + "\n")
        f.write("End               : " + end + "\n")
        f.write("Amount            : " + str(first_price) +" KMD" + "\n")
        f.write("Promo             : " + str(promo) +" KMD" + "\n")
        f.write("Random Reduction  : " + str(random_reduct) +" KMD" + "\n")
        f.write("Final Payment     : " + str(price) +" KMD" + "\n")
        f.write("\n")
        f.close()
```

```
    pass


#Function to calculate the invoice details based on start and end cities, promo code, and vehicle type
def invoice_cal(start, end, promoc=None, vehi=None):
    amount = price_chart[start][end] * vehicle[vehi] if vehi in vehicle else price_chart[start][end]
    prom = promo_code[promoc] if promoc in promo_code else 0
    ran = random.choice([5, 0]) if not prom else 0
    famount = amount - prom - ran
    return amount, prom, ran, famount


#Function to generate the invoice and save it to a file
def invoice(start, end, promoc=None, vehi=None):
    amount, prom, ran, famount = invoice_cal(start, end, promoc, vehi)
    save(start,end,amount,prom,ran,famount)
    print("Start           : ", start)
    print("End             : ", end)
    print("Amount          : ", amount, " KMD")
    print("Promo           : ", prom, " KMD")
    print("Random Reduct   : ", ran, " KMD")
    print("Final Amount    : ", famount, " KMD")
    return
```

3. Test cases for the program

	Test case	Input	Expected output	Actual output	Result
1	Shows the price between the two cities	D:\Desktop\cw>dm.py alvin razi	****Drop-Me**** Start : alvin End : razi Amount : 40 KMD Promo : 0 KMD Random Reduct : 5 KMD Final Amount : 35 KMD None ****Drop-Me****	****Drop-Me**** Start : alvin End : razi Amount : 40 KMD Promo : 0 KMD Random Reduct : 5 KMD Final Amount : 35 KMD None ****Drop-Me****	Pass
2	Shows the price between the two cities after applying the promo code (pro2)	D:\Desktop\cw>dm.py alvin razi /pro2	****Drop-Me**** Start : alvin End : razi Amount : 40 KMD Promo : 2 KMD Random Reduct : 0 KMD Final Amount : 38 KMD None ****Drop-Me****	****Drop-Me**** Start : alvin End : razi Amount : 40 KMD Promo : 2 KMD Random Reduct : 0 KMD Final Amount : 38 KMD None ****Drop-Me****	Pass
3	Shows the price between the two cities after applying the promo code (pro13)	D:\Desktop\cw>dm.py jamz mali /pro13	****Drop-Me**** Start : jamz End : mali Amount : 40 KMD Promo : 13 KMD Random Reduct : 0 KMD Final Amount : 27 KMD None ****Drop-Me****	****Drop-Me**** Start : jamz End : mali Amount : 40 KMD Promo : 13 KMD Random Reduct : 0 KMD Final Amount : 27 KMD None ****Drop-Me****	Pass

4	Show the full price plan for the whole country (for all 3 vhicles)	D:\Desktop\ cw>dm.py /price	****Drop-Me**** Price Chart for Traveling (Trishaw): Cities alvin jamz razi mali zuhar alvin 0 20 40 40 20 jamz 20 0 20 40 40 razi 40 20 0 20 40 mali 40 40 20 0 20 zuhar 20 40 40 20 0 Price Chart for Traveling (Car): Cities alvin jamz razi mali zuhar alvin 0 40 80 80 40 jamz 40 0 40 80 80 razi 80 40 0 40 80 mali 80 80 40 0 40 zuhar 40 80 80 40 0 Price Chart for Traveling (Van): Cities alvin jamz razi mali zuhar alvin 0 60 120 120 60 jamz 60 0 60 120 120 razi 120 60 0 60 120 mali 120 120 60 0 60	****Drop-Me**** Price Chart for Traveling (Trishaw): Cities alvin jamz razi mali zuhar alvin 0 20 40 40 20 jamz 20 0 20 40 40 razi 40 20 0 20 40 mali 40 40 20 0 20 zuhar 20 40 40 20 0 Price Chart for Traveling (Car): Cities alvin jamz razi mali zuhar alvin 0 40 80 80 40 jamz 40 0 40 80 80 razi 80 40 0 40 80 mali 80 80 40 0 40 zuhar 40 80 80 40 0 Price Chart for Traveling (Van): Cities alvin jamz razi mali zuhar alvin 0 60 120 120 60 jamz 60 0 60 120 120 razi 120 60 0 60 120 mali 120 120 60 0 60	Pass

			zuhar 60 120 120 60 0 ****Drop-Me****	zuhar 60 120 120 60 0 ****Drop-Me****	
5	Shows the price between the two cities This uses a car	D:\Desktop\ cw>dm.py alvin razi /c	****Drop-Me**** Start : alvin End : razi Amount : 40 KMD Promo : 0 KMD Random Reduct : 0 KMD Final Amount : 40 KMD None ****Drop-Me****	****Drop-Me**** Start : alvin End : razi Amount : 40 KMD Promo : 0 KMD Random Reduct : 0 KMD Final Amount : 40 KMD None ****Drop-Me****	Pass
6	Shows the price between the two cities while apply a 10 KMD reduction to total bill. The rider prefers a van.	D:\Desktop\ cw>dm.py alvin razi /pro10 /v	****Drop-Me**** Start : alvin End : razi Amount : 120 KMD Promo : 10 KMD Random Reduct : 0 KMD Final Amount : 110 KMD None ****Drop-Me****	****Drop-Me**** Start : alvin End : razi Amount : 120 KMD Promo : 10 KMD Random Reduct : 0 KMD Final Amount : 110 KMD None ****Drop-Me****	Pass
7	Program help command	D:\Desktop\ cw>dm.py /?	****Drop-Me**** * Keep a space between every command * /price to show the price chart * vehicle: </c> for car </v> for van * promo code: </pro2> 2KDM, <pro5> 5KDM, <pro10> 10KDM * <city_name> <space> <city_name> To get the ride with a Trishaw	****Drop-Me**** * Keep a space between every command * /price to show the price chart * vehicle: </c> for car </v> for van * promo code: </pro2> 2KDM, <pro5> 5KDM, <pro10> 10KDM * <city_name> <space> <city_name> To get the ride with a Trishaw	Pass

			* <city_name> <space> <city_name> <space> </c> To get the ride with a chosen vehicle * <city_name> <space> <city_name> <space> </v> To get the ride with a chosen vehicle * <city_name> <space> <city_name> <space> </v> <space> <promo_code> to add a promo code * <city_name> <space> <city_name> <space> <promo_code> <space> </vehicle_letter> to add a promo code * Do not use <> brackets in commands ****Drop-Me****	* <city_name> <space> <city_name> <space> </c> To get the ride with a chosen vehicle * <city_name> <space> <city_name> <space> </v> To get the ride with a chosen vehicle * <city_name> <space> <city_name> <space> </v> <space> <promo_code> to add a promo code * <city_name> <space> <city_name> <space> <promo_code> <space> </vehicle_letter> to add a promo code * Do not use <> brackets in commands ****Drop-Me****	
8	When we enter Invalid city names	D:\Desktop\ cw>dm.py alvin ra	Invalid cities. Please provide valid starting and ending cities as provided above. You can then check city names and list of prices by the command, dm /price.	Invalid cities. Please provide valid starting and ending cities as provided above. You can then check city names and list of prices by the command, dm /price.	Pass
9	Selecting the same start and end location	D:\Desktop\ cw>dm.py alvin alvin	****Drop-Me**** Start : alvin End : alvin Amount : 0 KMD Promo : 0 KMD Random Reduct : 5 KMD Final Amount : -5 KMD None ****Drop-Me****	****Drop-Me**** Start : alvin End : alvin Amount : 0 KMD Promo : 0 KMD Random Reduct : 5 KMD Final Amount : -5 KMD None ****Drop-Me****	Pass

10	Random reduction from bill	D:\Desktop\ cw>dm.py alvin mali	****Drop-Me**** Start : alvin End : mali Amount : 40 KMD Promo : 0 KMD Random Reduct : 5 KMD Final Amount : 35 KMD None ****Drop-Me****	****Drop-Me**** Start : alvin End : mali Amount : 40 KMD Promo : 0 KMD Random Reduct : 5 KMD Final Amount : 35 KMD None ****Drop-Me****	Pass
11	When we enter a wrong command line	D:\Desktop\ cw>dm.py price	****Drop-Me**** * Enter /? To show all the commands ****Drop-Me****	****Drop-Me**** * Enter /? To show all the commands ****Drop-Me****	Pass
12	Saving each invoice in different text files after printing	D:\Desktop\ cw>dm.py jamz zuhar /pro6 /v	Date : 2023-07-31 Time : 19:58:07 Start : jamz End : zuhar Amount : 120 KMD Promo : 6 KMD Random Reduction : 0 KMD Final Payment : 114 KMD	Date : 2023-07-31 Time : 19:58:07 Start : jamz End : zuhar Amount : 120 KMD Promo : 6 KMD Random Reduction : 0 KMD Final Payment : 114 KMD	Pass

4. Screenshots of the working program

4.1. Shows the price between the two cities

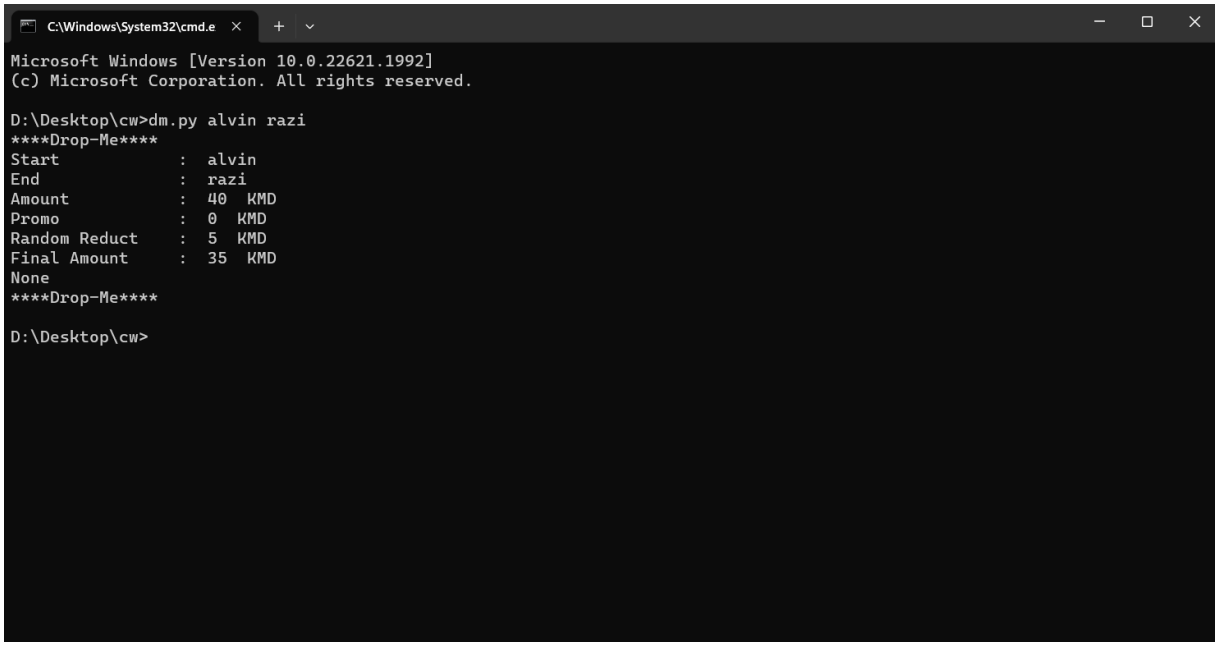

Figure 5: Shows the price between the two cities

4.2. Shows the price between the two cities after applying the promo code

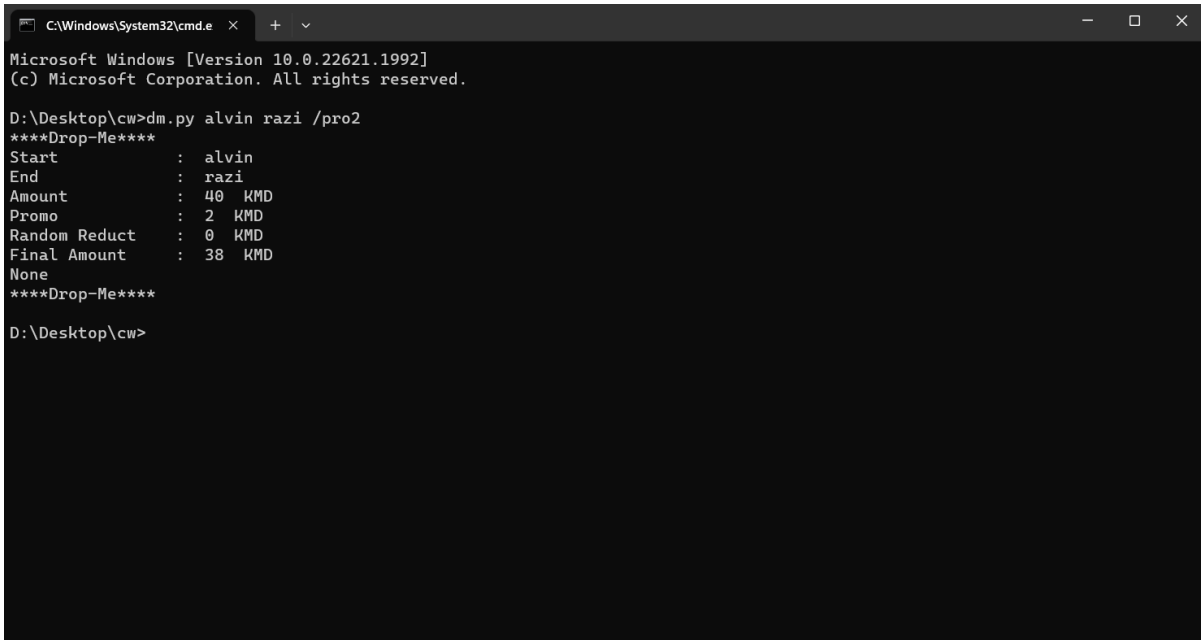

Figure 6:Shows the price between the two cities after applying the promo code

4.3. Show the full price plan for the whole country (for all 3 vehicles)

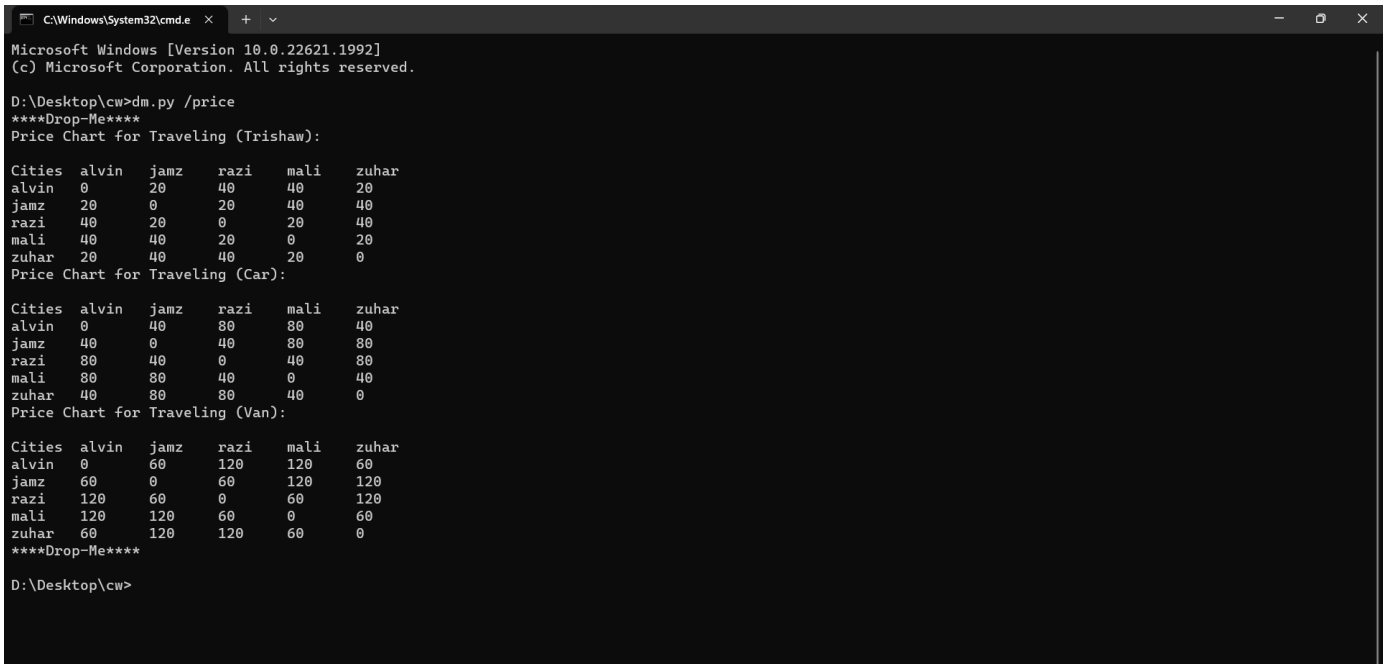

Figure 7: Show the full price plan for the whole country (for all 3 vehicles)

4.4. Shows the price between the two cities after applying the promo code (pro13)

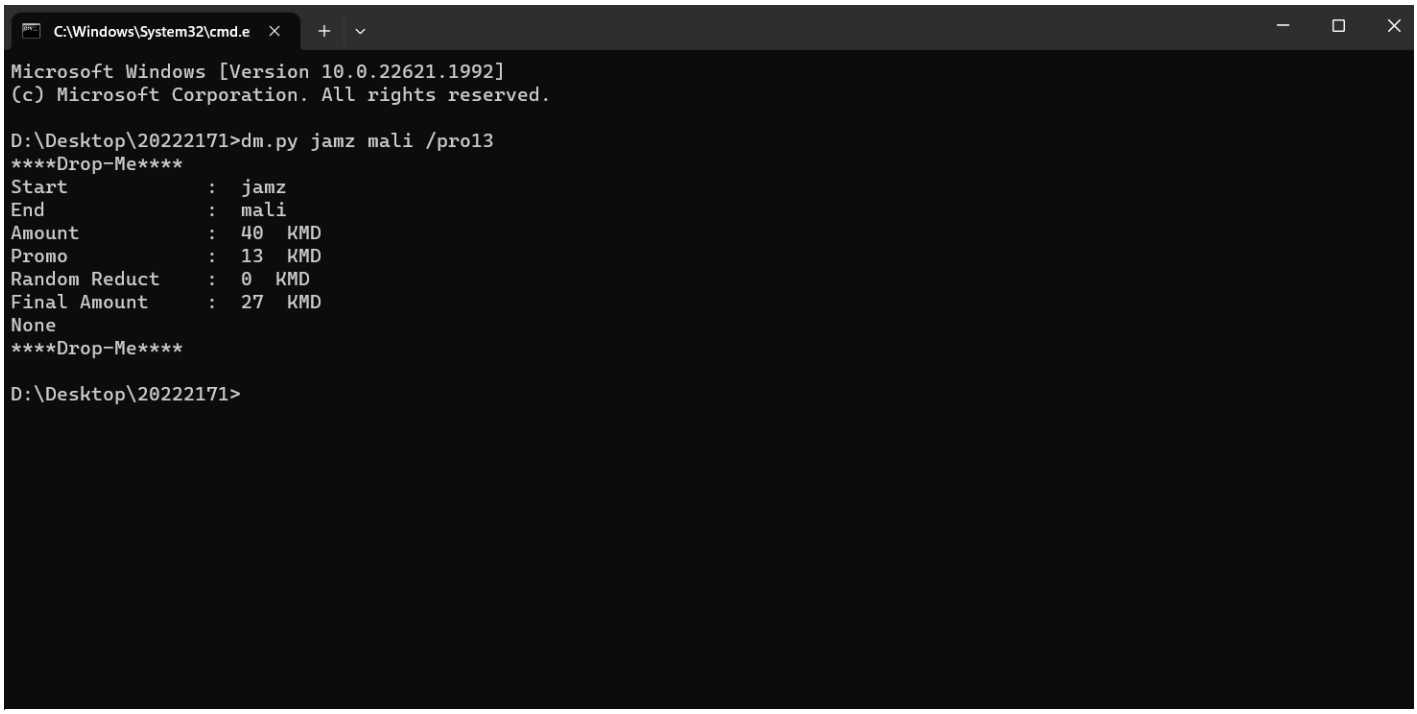

Figure 8: Shows the price between the two cities after applying the promo code (pro13)

4.5. Shows the price between the two cities and generates an invoice file for the trip. This uses a car

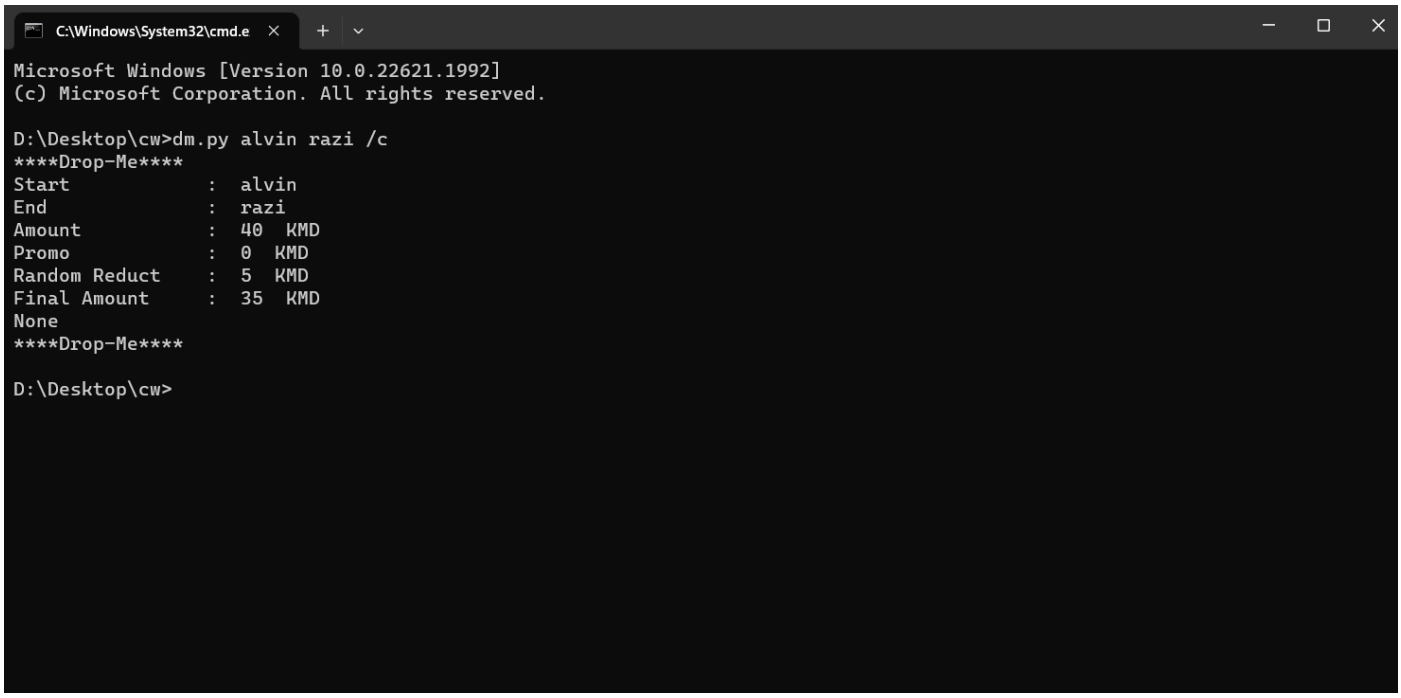

Figure 9: Shows the price between the two cities and generates an invoice file for the trip. This uses a car

4.6. Shows the price between the two cities while apply a 10 KMD reduction to total bill. The rider prefers a van.

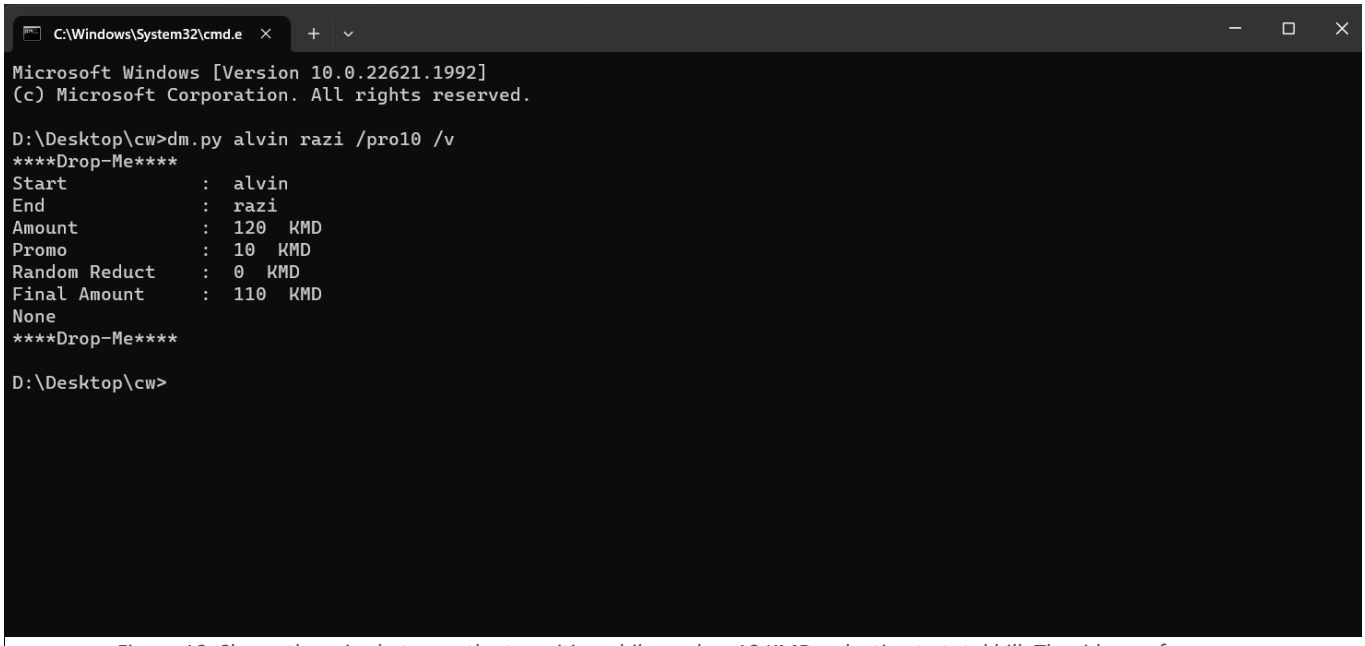

Figure 10: Shows the price between the two cities while apply a 10 KMD reduction to total bill. The rider prefers a van

4.7. Program help command

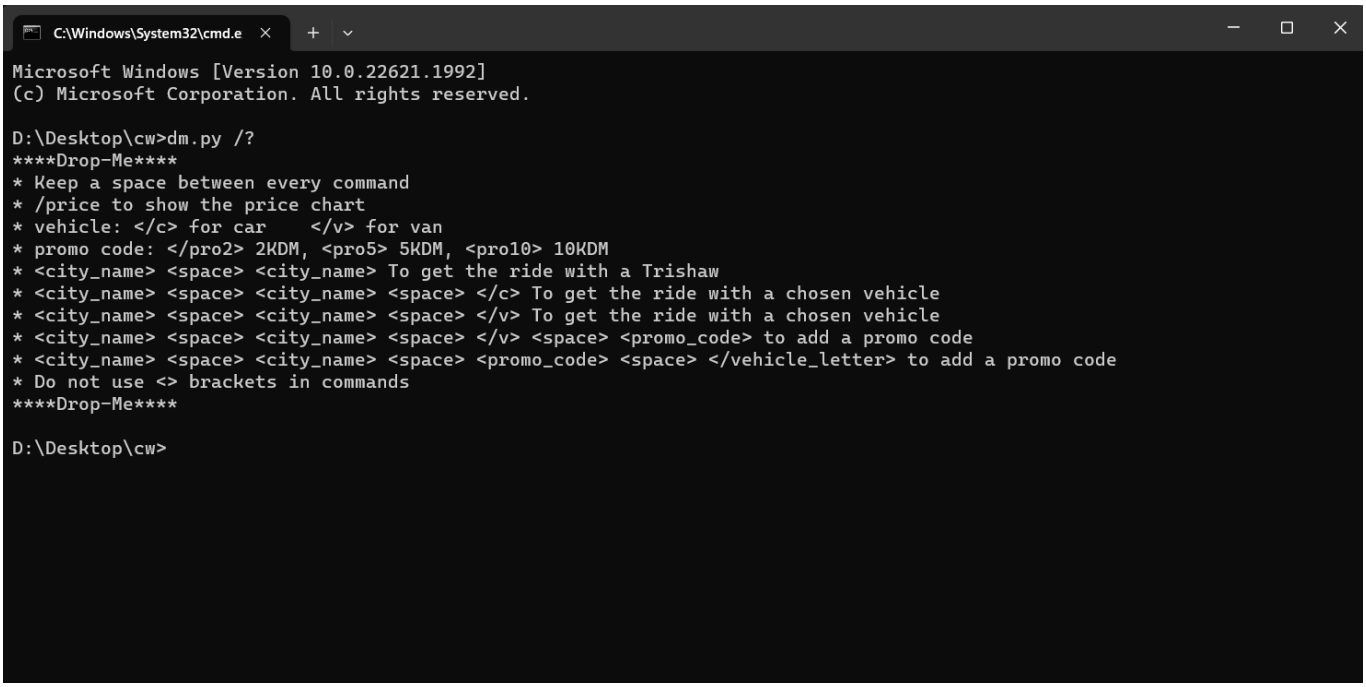

Figure 11: Program help command

4.8. When we enter Invalid city names

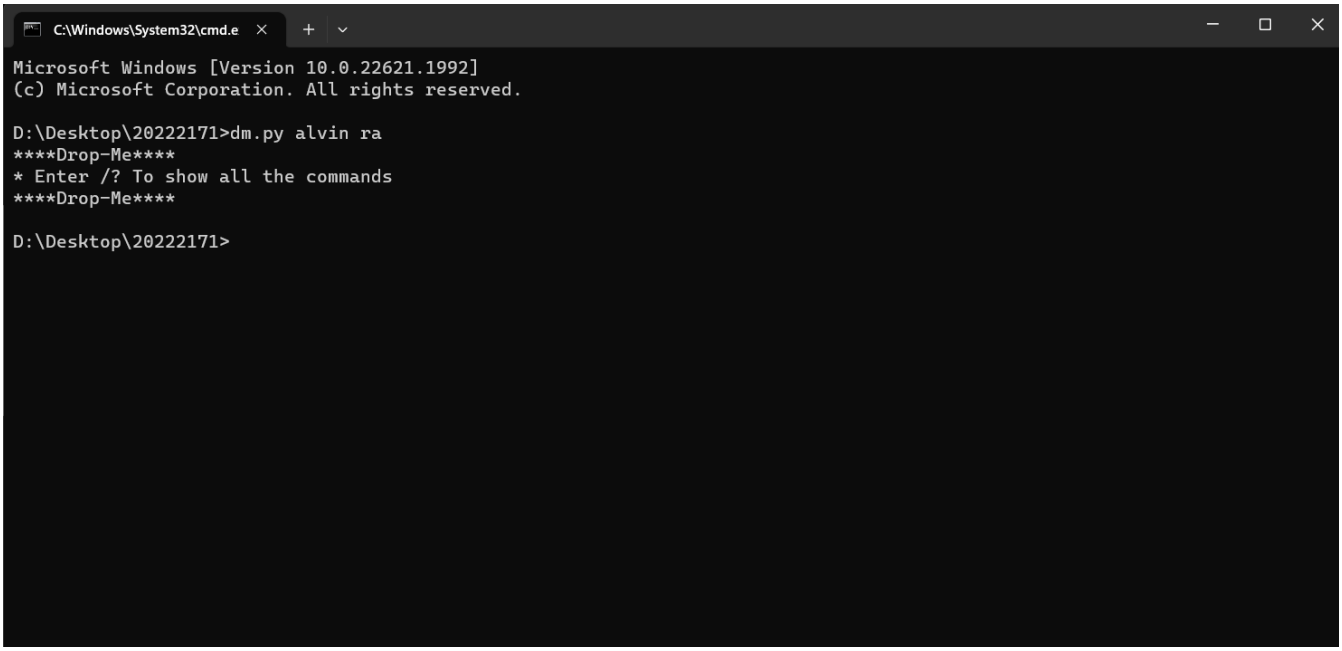

Figure 12: When we enter Invalid city names

4.9. Selecting the same start and end location

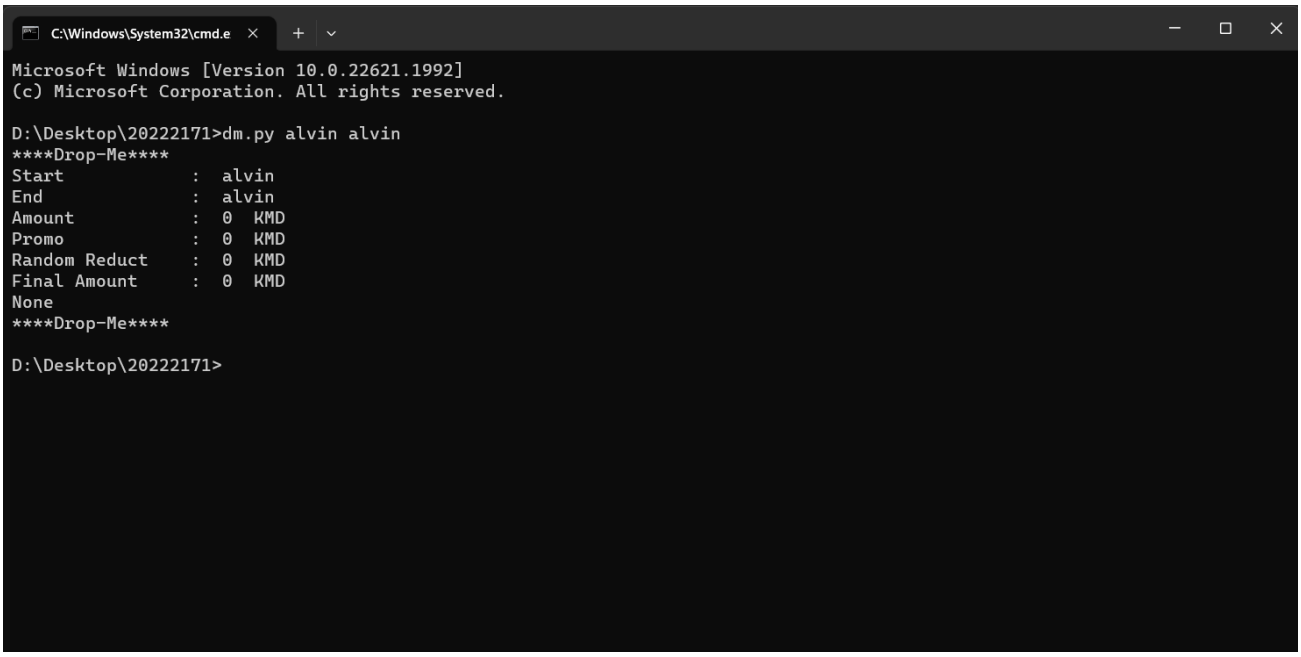

Figure 13: Selecting the same start and end location

4.10. Random reduction from bill

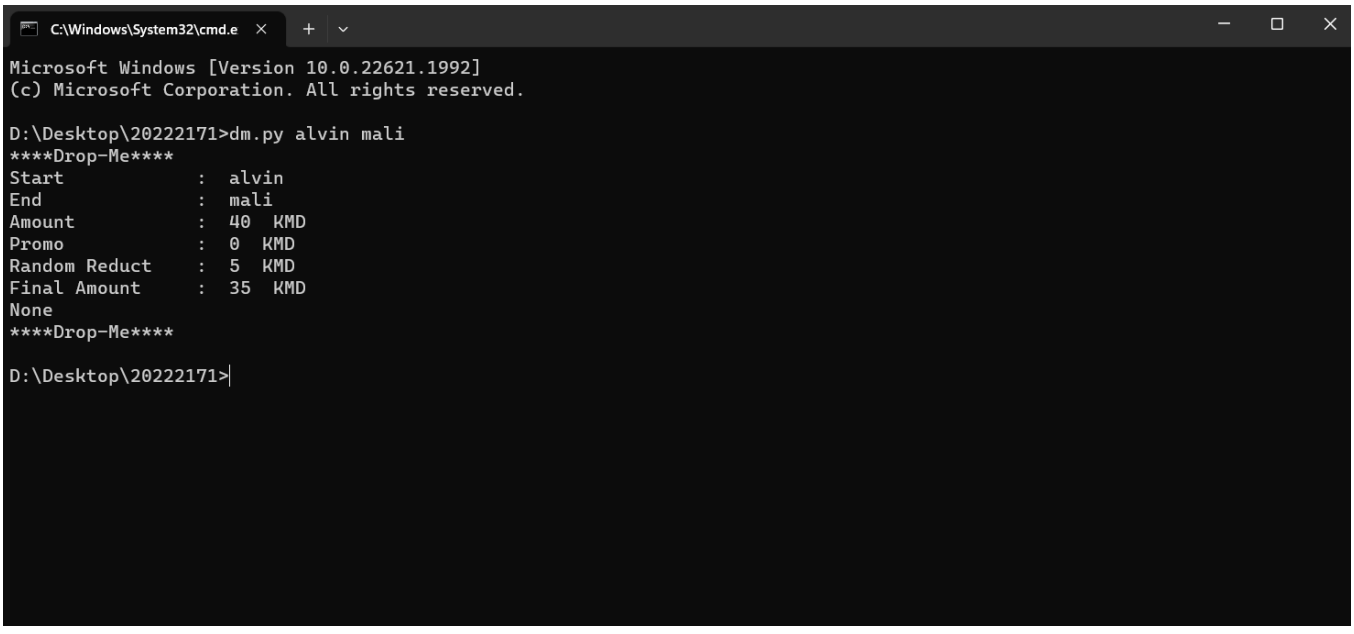

Figure 14: Random reduction from bill

4.11. When we enter a wrong command line

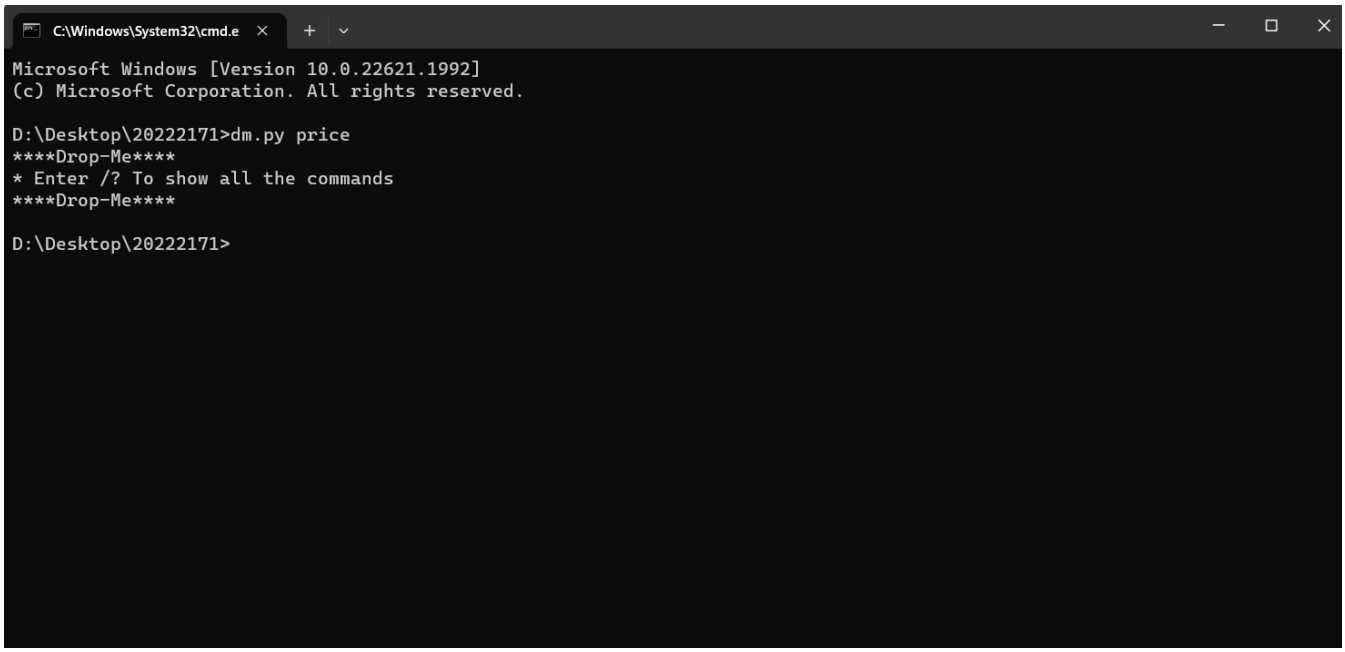

Figure 15: When we enter a wrong command line

4.12. Saving each invoice in different text files after printing

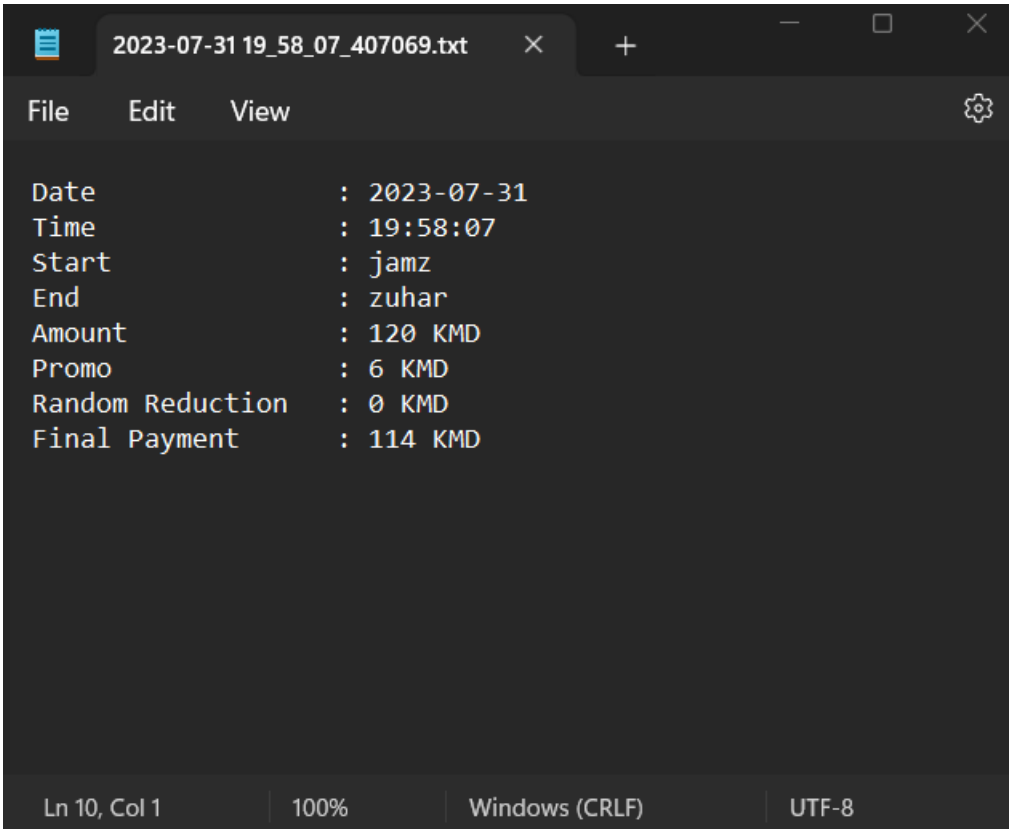

Figure 16: Saving each invoice in different text files after printing

Conclusion

In conclusion, the console application developed for Miranda Kingdom is a user-friendly and efficient tool for travelers to access travel information, generate invoices and take advantage of promotions in their taxi trips using DropMe™ services. With various features like distance-based fare calculation, transport option and discount codes, the app offers passengers a smooth and cost-effective transport experience. The ability to save invoices in text files ensures transparency and easy storage. All in all, the application contributes a lot to the smooth operation of the transport system in the kingdom of Miranda.