Student Assessment Submission and Declaration

When submitting evidence for assessment, you must sign a declaration confirming that the work is your own.

Student name:	Rosen Penchev Kyurkchiev	ESL ID No	120021
		BSU ID No	575573
Submission date:			**05/02/2025**
Program:	BSc Computing		
Module name and code:	**Software Foundation CPUF001**		
Title:	**Development Project** **Data Processing programme**		
Assessor name:	Lahiru Sooriyabandara		

Plagiarism

Plagiarism is a form of cheating. Plagiarism must be avoided at all costs and students who break the rules, however innocently, may be penalised. It is your responsibility to ensure that you understand correct referencing practices. As a university-level student, you are expected to use appropriate references throughout and keep carefully detailed notes of all your sources of materials for material you have used in your work, including any material downloaded from the Internet. Please consult the relevant unit lecturer or your course tutor if you need any further advice.

Student declaration I certify that the assignment submission is entirely my own work. I fully understand the consequences of plagiarism. I understand that making a false declaration is a form of malpractice.			
Student signature:	Rosen Kyurkchiev	Date:	05/02/2025

Contents

Development Document

1. Introduction

The project was developed based on the requirements. The idea is taken from what I do as part of my job. I calculate journey prices manually, and it has always fascinated me if I could create some tool to help me with that. As a result, I created the Phyton program, which calculates taxi fares based on trip length. The program reads trip information from a CSV file via the command line, calculates trip costs, and writes the processed data to an output CSV file.

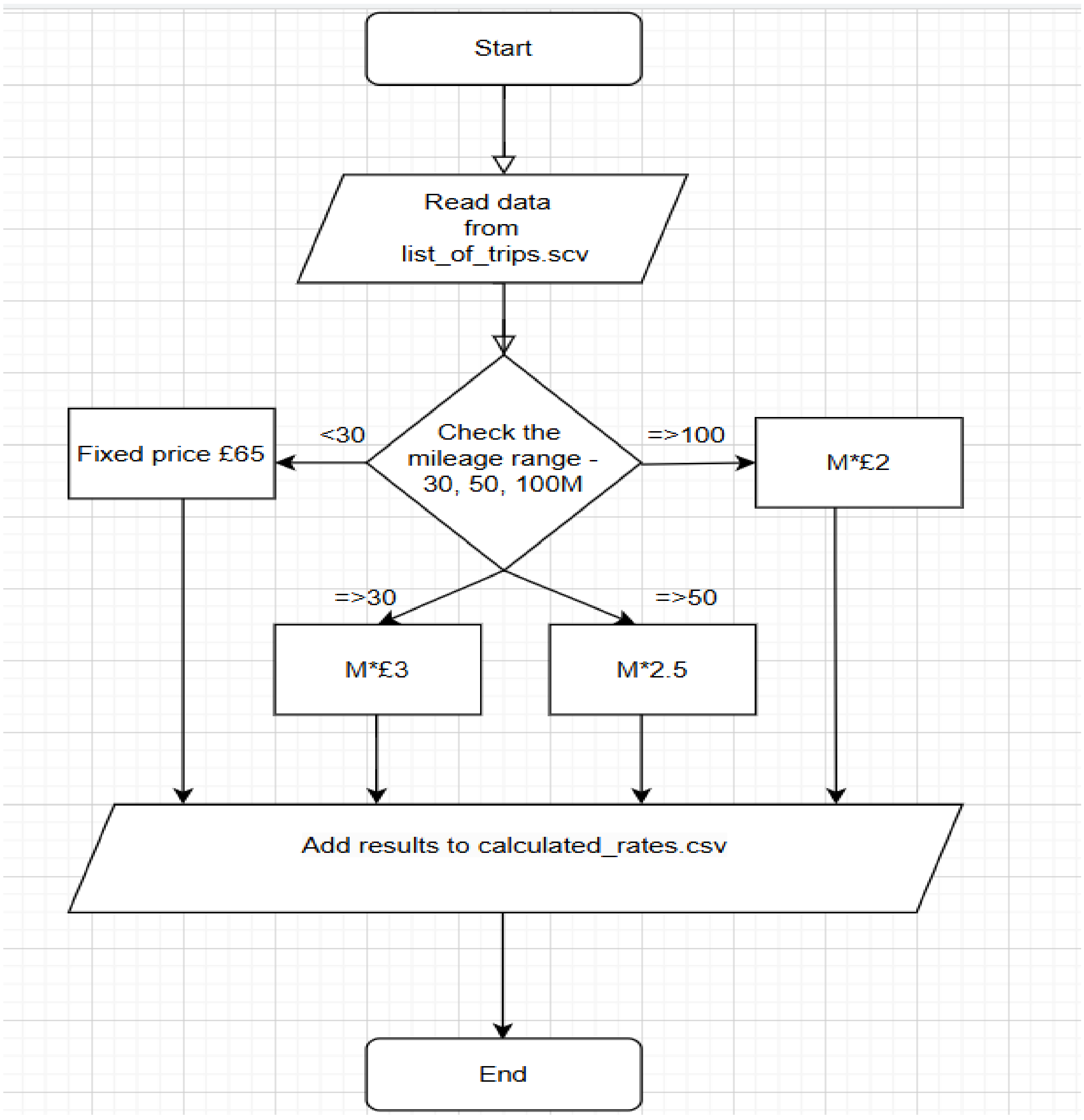
The program was designed with a modular structure to ensure maintainability and readability. Key functionalities were encapsulated in separate functions. This modular approach promotes code reusability and simplifies understanding and modifying the program's logic. The use of the CSV module simplified the reading and writing of CSV data, streamlining the input/output operations.

The link to my GitHub repository ***https://github.com/Rosen82/SoftwareFoundation***

2. Solution Design

The Python code processes trip data from a CSV file. It uses two functions, 'calculate_price' and 'check_the_lent_trip', to determine trip costs based on mileage and fares. The program reads input data, calculates costs, and writes the results to a new CSV file.

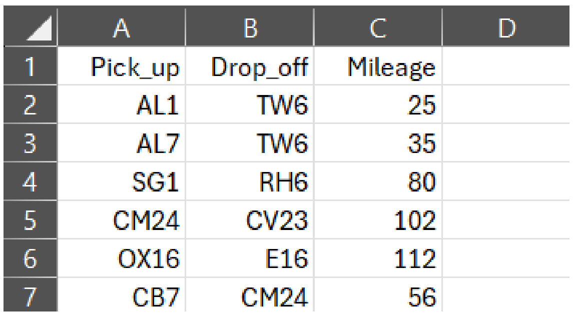
Flowchart:


Start
Read data from list_of_trips.scv
Check the mileage range - 30, 50, 100M
<30
Fixed price £65
=>100
M*£2
=>30
M*£3
=>50
M*2.5
Add results to calculated_rates.csv
End

Walkthrough:

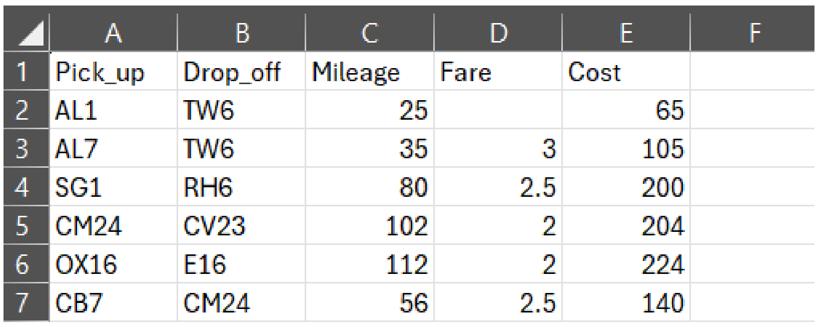
1. **Initialisation:** The code starts by importing the CSV module to handle CSV files and initialising an empty list called trips to store the processed trip data.
2. **Read Trip Data:** It opens the input CSV file ("listoftrips.csv") and reads the data row by row, extracting the pick-up point, drop-off point, and mileage for each trip.
3. **Calculate Fare:** The code determines the fare per mile based on the mileage using the 'check_the_lent_trip' function for each trip. Different fare rates apply to different mileage ranges.
4. **Calculate Total Cost:** The 'calculate_price' function calculates the total cost of each trip. It applies a fixed price of £65 for trips under 30 miles and calculates the cost based on mileage and fare for longer trips.
5. **Store Processed Data:** The processed trip information, including pick-up, drop-off, mileage, fare, and total cost, is stored in the trip list.
6. **Write to Output File:** Finally, the code opens a new CSV file ("calculated_rates.csv") and writes the processed trip data, including a header row with column names.

CSV files:



	A	B	C	D
1	Pick_up	Drop_off	Mileage	
2	AL1	TW6	25	
3	AL7	TW6	35	
4	SG1	RH6	80	
5	CM24	CV23	102	
6	OX16	E16	112	
7	CB7	CM24	56	

Python outputted CSV file:



	A	B	C	D	E	F
1	Pick_up	Drop_off	Mileage	Fare	Cost	
2	AL1	TW6	25		65	
3	AL7	TW6	35	3	105	
4	SG1	RH6	80	2.5	200	
5	CM24	CV23	102	2	204	
6	OX16	E16	112	2	224	
7	CB7	CM24	56	2.5	140	

The calculations that Python has performed are:

1. Fare Calculation:

It determines the fare per mile based on the mileage of the trip.

Different fares apply to different mileage ranges:

£2 per mile for trips 100 miles or longer.

£2.50 per mile for trips 50 miles or longer (but less than 100 miles).

£3 per mile for trips 30 miles or longer (but less than 50 miles).

2. Cost Calculation:

Trips under 30 miles: A fixed cost of £65 is applied.

Trips 30 miles or longer: The total cost is calculated by multiplying the trip's mileage by the fare per mile determined in the previous step.

3. Reflective Evaluation

Developing this Python program for processing trip data and calculating fares was a valuable learning experience.

I broke down the program into functions **(calculate_price, check_the_lent_trip),** making the code more organised, readable, and easier to test. This approach promotes code reuse and simplifies future modifications or use in different projects.

I learned to create a logical flow for reading input, processing data, and writing output, which makes it easy to understand the program's execution. The CSV module simplified the handling of CSV files, making the input/output operations efficient and straightforward.

Error handling is a new experience for me. It was interesting to find out where and how it should be implemented in the code for the best performance. Initially, there was confusion about where to implement the fixed price logic for trips under 30 miles. This highlighted the importance of careful planning and consideration of the program's flow.

While the code includes initial notes, more comprehensive documentation and comments could enhance readability and maintainability.

Future Improvements:

I use a dispatch system daily, which performs similar tasks. The model I created is very primitive, and to be used in real situations or how I calculate prices should have many changes and improvements. In reality, the function should be like minimum charge 0 to 29 miles £45, 30 to 50 miles £2.5 per mile, 50 to 100 miles £2, and over 100 miles £1.8. Then, a trip of 90 miles should be calculated as (29 miles - £45) + (20miles * £2.5) + (40 miles * £2) = £175, my program simplifies the calculation, and the price will be 90 miles * £2.5 = £225.

My knowledge wasn't enough to create a program which exactly performs how I calculate prices. Additionally, the program could automatically calculate mileage from point A to point B, which would be beneficial. That would automate the whole process from gaining the distance and calculating the price based on it.

More robust data validation could be added to handle potential errors in the input CSV file, such as missing values or incorrect data types.

Conclusion

This project provided valuable experience in Python programming and data manipulation. It highlighted the importance of modular design, efficient data handling, and logical program flow. While the current program successfully calculates fares based on mileage, further improvements would enhance its accuracy and real-world applicability. Specifically, refining the fare calculation logic, automating mileage determination, and adding robust error handling would make the program more powerful. This project is a solid foundation for future development and demonstrates a clear understanding of key programming concepts.

References:

Beecher, K. (2018). *Computational thinking a beginner's guide to problem-solving and programming*. Swindon Bcs: British Computer Society, The Chartered Institute For It.

Beri Rydhm (2019). *Python Made Simple*. BPB Publications.

Knapp, M. (2017). *Python Programming for Beginners*. CreateSpace Independent Publishing Platform.

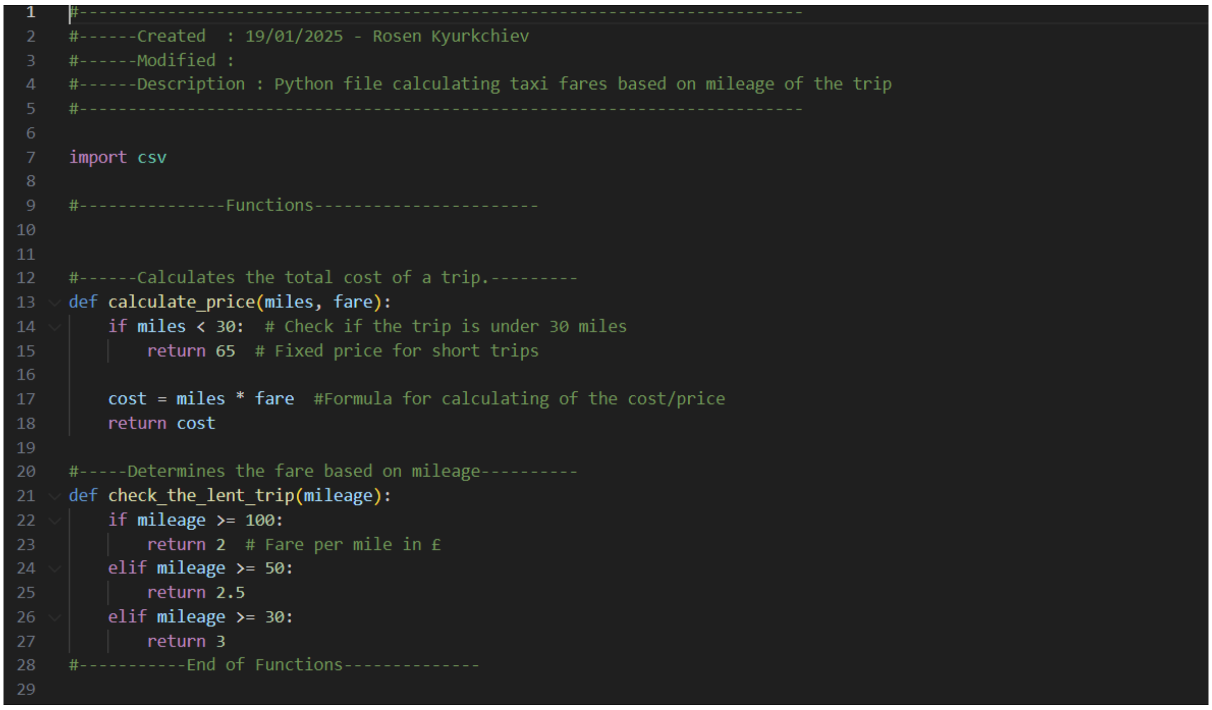
W3Schools (1999). *W3Schools online web tutorials*. [online] W3schools.com. Available at: https://www.w3schools.com/.

Appendix:

Screenshots of the code:

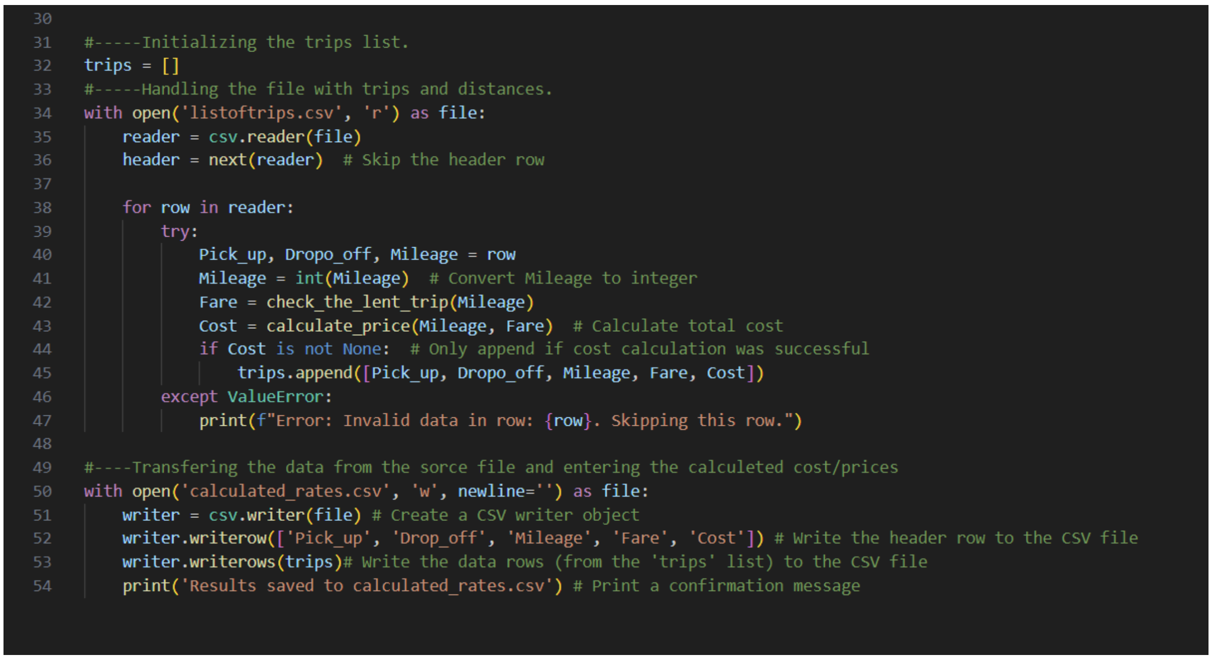
Figure 1- The two functions that calculate the price based on the length of the trip and assign the fare for the journey.

Figure 2— Part of the code that opens the data file. It activates the functions and calculates the price for the trips. In the end, it writes the results in the empty file.



```python
#------------------------------------------------------------------------
#------Created   : 19/01/2025 - Rosen Kyurkchiev
#------Modified :
#------Description : Python file calculating taxi fares based on mileage of the trip
#------------------------------------------------------------------------

import csv

#--------------Functions----------------------


#------Calculates the total cost of a trip.---------
def calculate_price(miles, fare):
    if miles < 30:  # Check if the trip is under 30 miles
        return 65  # Fixed price for short trips

    cost = miles * fare  #Formula for calculating of the cost/price
    return cost

#-----Determines the fare based on mileage----------
def check_the_lent_trip(mileage):
    if mileage >= 100:
        return 2  # Fare per mile in £
    elif mileage >= 50:
        return 2.5
    elif mileage >= 30:
        return 3
#----------End of Functions-------------

```

Fig 1.



```python

#-----Initializing the trips list.
trips = []
#-----Handling the file with trips and distances.
with open('listoftrips.csv', 'r') as file:
    reader = csv.reader(file)
    header = next(reader)  # Skip the header row

    for row in reader:
        try:
            Pick_up, Dropo_off, Mileage = row
            Mileage = int(Mileage)  # Convert Mileage to integer
            Fare = check_the_lent_trip(Mileage)
            Cost = calculate_price(Mileage, Fare)  # Calculate total cost
            if Cost is not None:  # Only append if cost calculation was successful
                trips.append([Pick_up, Dropo_off, Mileage, Fare, Cost])
        except ValueError:
            print(f"Error: Invalid data in row: {row}. Skipping this row.")

#----Transfering the data from the sorce file and entering the calculeted cost/prices
with open('calculated_rates.csv', 'w', newline='') as file:
    writer = csv.writer(file) # Create a CSV writer object
    writer.writerow(['Pick_up', 'Drop_off', 'Mileage', 'Fare', 'Cost']) # Write the header row to the CSV file
    writer.writerows(trips)# Write the data rows (from the 'trips' list) to the CSV file
    print('Results saved to calculated_rates.csv') # Print a confirmation message
```

Fig 2.