A white and black background

Description automatically generated with medium confidenceA black text on a white background

Description automatically generated

**University of Westminster**

**Module Name:**  Software Development 1

**Module leader**: Guhanathan Poravi

**Assignment type:** Individual coursework

**Task : E,D**

**Group ID**: 24

**Submission date:** 24.12.2024

|  |  |  |
| --- | --- | --- |
| **Student Id** | **Uow ID** | **Student Name** |
| 20231259 | w2120290 | H. I. A. Haneef |

**School of Computing**

**Informatics Institute of Technology (IIT Campus)**

Table of Contents

[Pseudocode for CSV Data processing program 1](#_Toc185844787)

[Program description: 1](#_Toc185844788)

[Steps of the Program: 1](#_Toc185844789)

[Test cases 8](#_Toc185844790)

[Test cases and their outcomes. 8](#_Toc185844791)

[Test case 1 outcomes 9](#_Toc185844792)

[Test case 2 outcomes 10](#_Toc185844793)

[Test case 3 outcomes 10](#_Toc185844794)

[Test case 4 outcomes 11](#_Toc185844795)

[Test case 5 outcomes 12](#_Toc185844796)

[Test case 6 outcomes 12](#_Toc185844797)

Table of figures

[Figure 1: Case 1 outcomes 9](#_Toc185844798)

[Figure 2: case 2 outcomes 10](#_Toc185844799)

[Figure 3 :case 3 outcomes 10](#_Toc185844800)

[Figure 4: case 4 outcomes 1 11](#_Toc185844801)

[Figure 5: case 4 outcomes 2 11](#_Toc185844802)

[Figure 6: case 4 outcome 3 11](#_Toc185844803)

[Figure 7: case 4 outcome 4 12](#_Toc185844804)

[Figure 8: Case 5 outcomes 12](#_Toc185844805)

[Figure 9 : case 6 outcomes 1 13](#_Toc185844806)

[Figure 10 : case 6 outcomes 2 13](#_Toc185844807)

# Pseudocode for CSV Data processing program

## Program description:

The program processes traffic data from a CSV file and calculates various metrics, such as the number of vehicles, trucks, buses, electric vehicles, and speed violations. It interacts with the user to validate input, display results, and save them to a file and then the program creates a histogram to display the number of vehicles passing through the two junctions each hour on the selected date. Users can choose to process multiple datasets in a single session.

## Steps of the Program:

# PROGRAM Traffic Data Analysis and Visualization

BEGIN

CLASS HistogramApp

METHOD \_\_init\_\_(traffic\_data, date)

SET self.traffic\_data = traffic\_data

SET self.date = date

# Extract hour values for Hanley Highway/Westway

INITIALIZE list\_of\_times\_hanley\_highway as empty list

FOR each inner\_list in traffic\_data

IF inner\_list[0] == 'Hanley Highway/Westway'

APPEND inner\_list[2].split(":")[0] to list\_of\_times\_hanley\_highway

# Count vehicles per hour for Hanley Highway

INITIALIZE hour\_count\_Hanley\_Highway as empty dictionary

FOR each hour in list\_of\_times\_hanley\_highway

INITIALIZE count = 0

FOR each j in list\_of\_times\_hanley\_highway

IF hour == j

INCREMENT count

SET hour\_count\_Hanley\_Highway[hour] = count

# Extract hour values for Elm Avenue/Rabbit Road

INITIALIZE list\_of\_times\_Elm\_Avenue as empty list

FOR each inner\_list in traffic\_data

IF inner\_list[0] == 'Elm Avenue/Rabbit Road'

APPEND inner\_list[2].split(":")[0] to list\_of\_times\_Elm\_Avenue

# Count vehicles per hour for Elm Avenue

INITIALIZE hour\_count\_Elm\_Avenue as empty dictionary

FOR each hour in list\_of\_times\_Elm\_Avenue

INITIALIZE count = 0

FOR each j in list\_of\_times\_Elm\_Avenue

IF hour == j

INCREMENT count

SET hour\_count\_Elm\_Avenue[hour] = count

# Store the hour counts in a nested dictionary

SET self.traffic\_data = {"Elm\_Avenue": hour\_count\_Elm\_Avenue, "Hanley\_Highway": hour\_count\_Hanley\_Highway}

METHOD setup\_window()

CREATE a window with title "Histogram" and dimensions 1200x600

SET background color to black

METHOD draw\_histogram()

DRAW horizontal line for histogram

SET parameters for grouped histogram (num\_groups, line\_length, space\_between\_groups)

CALCULATE group\_length

SET max\_height = 400

GET max\_traffic\_value from both datasets

CALCULATE scaler = max\_height / max\_traffic\_value

FOR group from 0 to num\_groups - 1

CALCULATE group\_start\_x

SET group\_key = formatted group number (e.g., '00', '01', etc.)

TRY

GET height\_for\_Elm\_Avenue\_triangle from traffic\_data

GET height\_for\_Hanley\_Highway\_triangle from traffic\_data

EXCEPT KeyError

SET height\_for\_Elm\_Avenue\_triangle = 0

SET height\_for\_Hanley\_Highway\_triangle = 0

# Scale the heights for better visualization

SET scaled\_height\_for\_Elm\_Avenue\_rectangle = height\_for\_Elm\_Avenue\_triangle \* scaler

SET scaled\_height\_for\_Hanley\_Highway\_rectangle = height\_for\_Hanley\_Highway\_triangle \* scaler

# Draw rectangles for Elm Avenue

SET x1 = group\_start\_x

SET x2 = x1 + group\_length / 2

SET y\_base = 525

CREATE rectangle\_for\_Elm\_Avenue

SET rectangle color and outline

DRAW rectangle\_for\_Elm\_Avenue

# Draw rectangles for Hanley Highway

SET x1 = x2

SET x2 = x1 + group\_length / 2

CREATE rectangle\_for\_Hanley\_Highway

SET rectangle color and outline

DRAW rectangle\_for\_Hanley\_Highway

# Add labels for vehicle counts

CREATE label1 for Elm Avenue

CREATE label2 for Hanley Highway

CREATE label3 for hour count

METHOD add\_legend()

SET legend\_x and legend\_y for positioning

CREATE legend boxes and labels for Elm Avenue and Hanley Highway

ADD title and hour labels to the legend

METHOD run()

TRY

CALL setup\_window()

CALL draw\_histogram()

CALL add\_legend()

WAIT for mouse click

CLOSE the window

EXCEPT GraphicsError

PASS

CLASS MultiCSVProcessor

METHOD \_\_init\_\_()

INITIALIZE current\_data as None

INITIALIZE traffic\_data as empty dictionary

INITIALIZE date as empty list

METHOD validate\_date\_input()

INITIALIZE prompts as list of tuples (day, month, year)

INITIALIZE date as empty list

FOR each prompt, min\_limit, max\_limit, format in prompts

WHILE True

PROMPT user for input

TRY

CONVERT input to integer

IF value is out of range (min\_limit, max\_limit)

PRINT "Out of range" message

CONTINUE

ELSE

APPEND value to date

BREAK

EXCEPT ValueError

PRINT "integer required" message

CONTINUE

SET file\_path = formatted string using date

RETURN file\_path

METHOD load\_csv\_file(file\_path)

TRY

OPEN file\_path in read mode

READ lines from file

PROCESS lines into current\_data (skip header)

# Calculate various metrics

SET total\_vehicles = length of current\_data

SET total\_trucks = count of 'Truck' in current\_data

SET total\_electric\_vehicles = count of 'True' in electric vehicle column

SET total\_two\_wheeled\_vehicles = count of 'Bicycle', 'Motorcycle', 'Scooter' in vehicle column

SET total\_busses\_leaving\_Elm\_Avenue\_heading\_north = count of 'Bus' heading north from Elm Avenue

SET total\_vehicles\_not\_turning\_left\_or\_right = count where direction in and out are equal

# Calculate percentages and averages

SET percentage\_of\_trucks = round((total\_trucks / total\_vehicles) \* 100)

SET total\_bicycles = count of 'Bicycle' in current\_data

SET average\_bicycle = round(total\_bicycles / 24)

SET total\_vehicle\_over\_speed\_limit = count of vehicles over speed limit

SET only\_Elm\_Avenue\_vehicles = filter current\_data for Elm Avenue

SET only\_Hanley\_Highway\_vehicles = filter current\_data for Hanley Highway

SET average\_Elm\_avenue\_scooters = round((count of 'Scooter' in Elm Avenue / length of only\_Elm\_Avenue\_vehicles) \* 100)

# Calculate busiest hour

INITIALIZE list\_of\_times\_hanley\_highway as empty list

FOR each inner\_list in current\_data

IF inner\_list[0] == 'Hanley Highway/Westway'

APPEND hour from timestamp to list\_of\_times\_hanley\_highway

INITIALIZE hour\_count\_Hanley\_Highway as empty dictionary

FOR each hour in list\_of\_times\_hanley\_highway

INITIALIZE count = 0

FOR each j in list\_of\_times\_hanley\_highway

IF hour == j

INCREMENT count

SET hour\_count\_Hanley\_Highway[hour] = count

SET total\_vehicles\_in\_busiest\_hour\_passing\_hanley\_highway = max of hour\_count\_Hanley\_Highway values

SET busy\_hours = formatted string of hours with max count

# Calculate rainy hours

INITIALIZE rainy\_hours as empty set

FOR each inner\_list in current\_data

IF inner\_list[5] is 'Light Rain' or 'Heavy Rain'

ADD hour to rainy\_hours

SET rainy\_hours = length of unique rainy\_hours

# Prepare outcomes

INITIALIZE outcomes as list of formatted strings with calculated metrics

EXCEPT FileNotFoundError

RETURN "File not found. Try again."

EXCEPT ZeroDivisionError

RETURN "File is empty"

EXCEPT IndexError

PASS

PRINT outcomes

APPEND outcomes to results.txt

METHOD clear\_previous\_data()

SET current\_data to None

SET traffic\_data to None

SET date to None

METHOD handle\_user\_interaction()

WHILE True

PROMPT user to load another dataset (Y/N)

CONVERT input to uppercase

IF input is 'Y'

BREAK

ELSE IF input is 'N'

BREAK

ELSE

PRINT "Type 'Y' or 'N'" message

CONTINUE

RETURN user input

METHOD process\_files()

WHILE True

CALL clear\_previous\_data()

SET file\_path = CALL validate\_date\_input()

SET outcomes = CALL load\_csv\_file(file\_path)

IF outcomes indicate file not found or empty

PRINT error message

ELSE

CREATE HistogramApp instance with current\_data and date

CALL app.run()

SET validation = CALL handle\_user\_interaction()

IF validation is 'Y'

CONTINUE

ELSE IF validation is 'N'

PRINT "Thank you for using. Goodbye!"

BREAK

IF \_\_name\_\_ == "\_\_main\_\_":

# Create an instance of MultiCSVProcessor

SET processor = new instance of MultiCSVProcessor

# Call the process\_files method to start the application

CALL processor.process\_files()

END

# Test cases

# Test cases and their outcomes.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test case No. | Input | Expected Outcome | Actual Outcome | Result |
| 1 | When user Input’s  “I” as the date  “L” as the month  “O” as year | The program should display ‘**Integer required**’ if a date input is the wrong data type and loops. | As Expected, | Pass |
| 2 | When user Input’s  “32” as the day  “13” as the month  “2025” as the year | The program should display ‘**Out of range**’ if date values entered are not in the correct range **(01-31 for day, 01-12 for month and 2000-2024 for year)** | As Expected, | Pass |
| 3 | When user input’s  Date is other than the name of the existing file | The program should display “**File not found, Try again**” and reloops to validate date again | As Expected, | Pass |
| 4 | When user input  Date is valid and the csv file exists | Processes the csv file and displays the results in idle and saves the results in a text file named result.txt, and then the histogram pops up to show the vehicle frequency for the selected date | As Expected, | Pass |
| 5 | When the user input for the load another data set is other than ‘y’ or ‘n’ | Displays Type Y or N and asks again to input | As Expected, | Pass |
| 6 | When the user inputs for the load another data set is ‘y’ or ‘n’ | If y the program loops again to the start to date validation  and if n the program ends | As Expected, | Pass |

## Test case 1 outcomes

A screenshot of a computer

Description automatically generated

Figure 1: Case 1 outcomes

## Test case 2 outcomes

A screenshot of a computer screen

Description automatically generated

Figure 2: case 2 outcomes

## Test case 3 outcomes

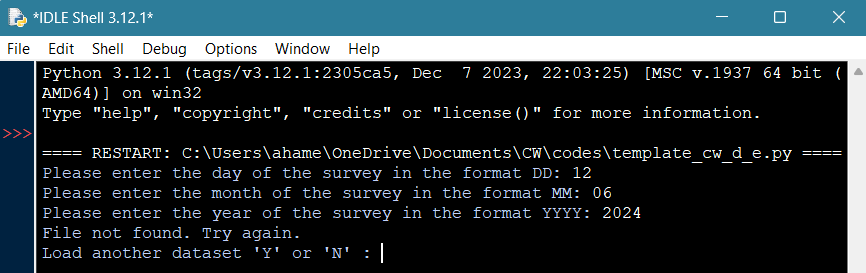


Figure 3 :case 3 outcomes

## Test case 4 outcomes

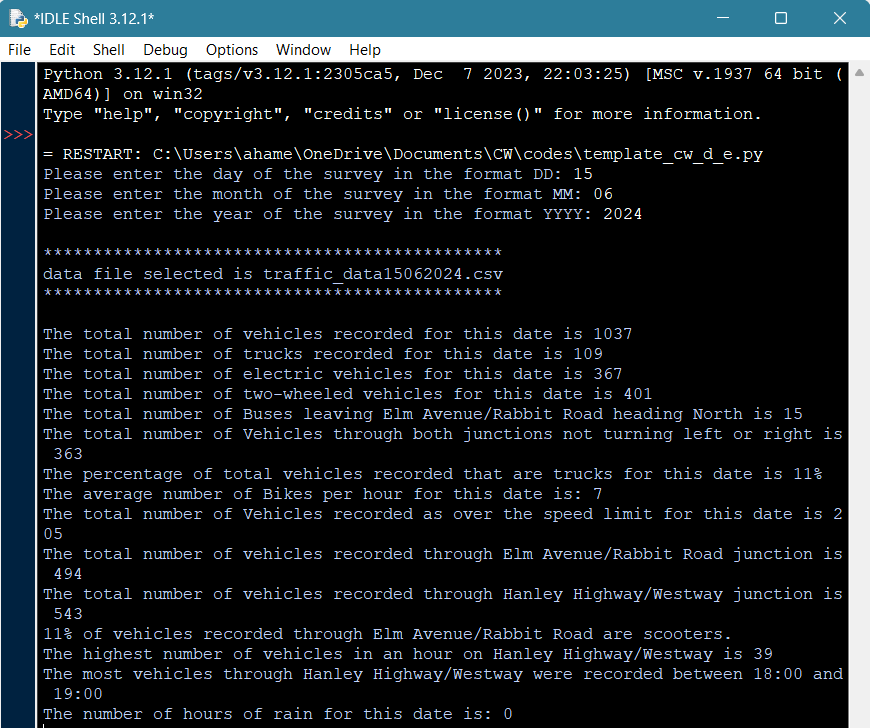


Figure 4: case 4 outcomes 1



Figure 5: case 4 outcomes 2

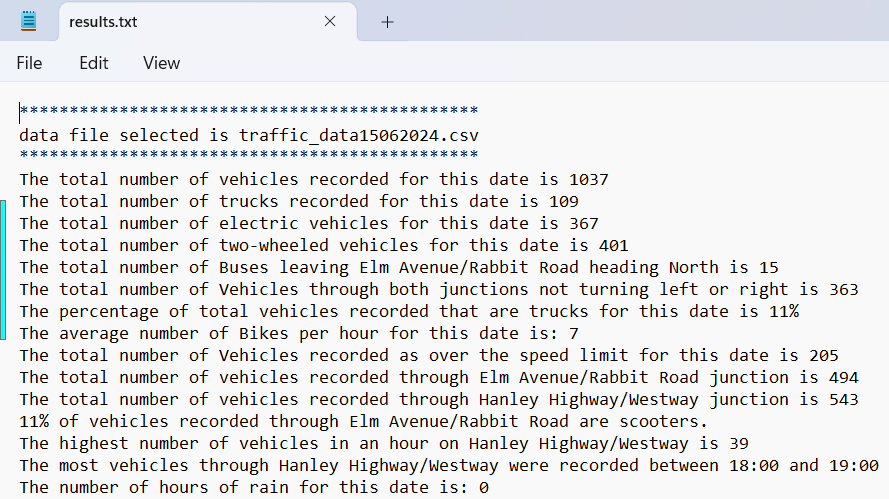


Figure 6: case 4 outcome 3

A screen shot of a graph

Description automatically generated

Figure 7: case 4 outcome 4

## Test case 5 outcomes

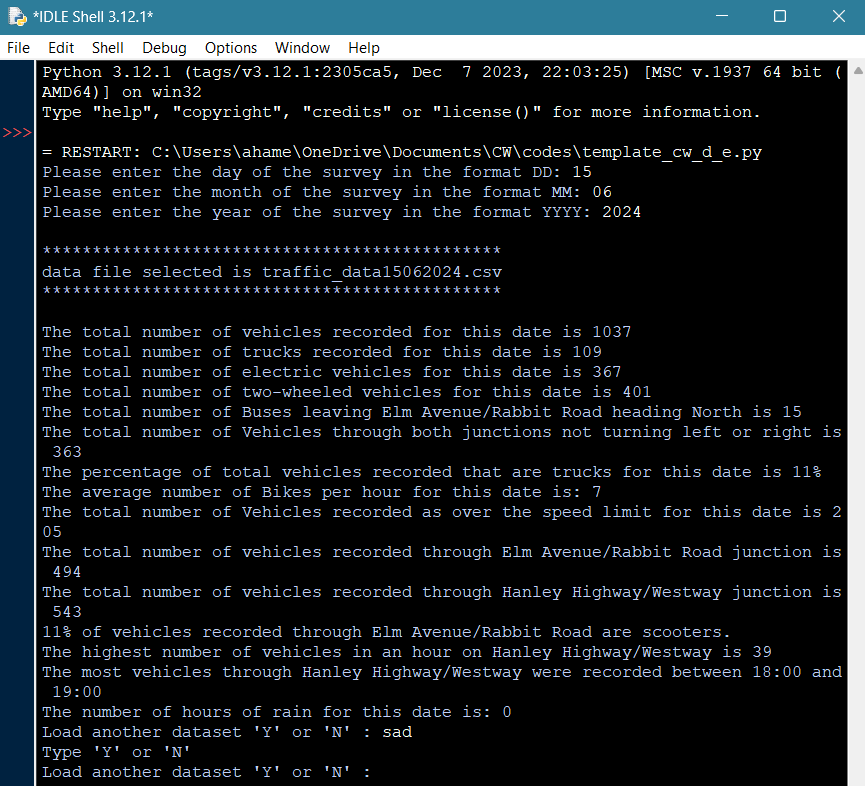


Figure 8: Case 5 outcomes

## Test case 6 outcomes

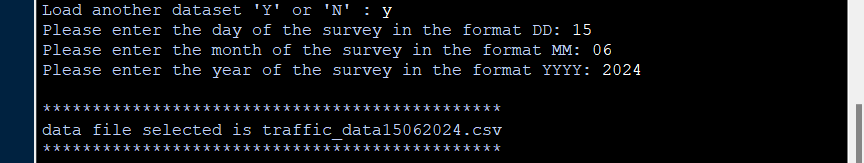


Figure 9 : case 6 outcomes 1

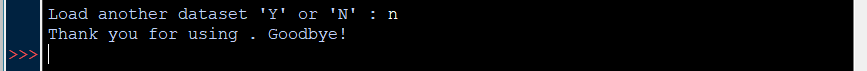


Figure 10 : case 6 outcomes 2