4COSC006C: Software Development I – Coursework specification (2025/26)	
Module leader	Guhanathan Poravi
Weighting:	50%
Qualifying mark:	30%
Description:	Coursework
Learning Outcomes Covered in this Assignment:	The coursework rationale is: **LO1** Analyse specific problems and design their solutions by applying appropriate algorithmic techniques; **LO2** Apply programming concepts to implement solutions in the taught programming language; **LO3** Implement and manipulate simple data structures; **LO4** Use an integrated development environment to create programs to satisfy a simple specification.
Handed Out:	Friday 24th October 2025
Due Date:	**Coursework Deadline:** Monday 24th November **BEFORE** 1:00 pm
Expected deliverables:	a) Submit your Python program code **Important: Submit a single python code file to run in IDLE shell** using the name convention: "student_id.py", e.g. w1234567.py - **DO NOT** submit your code as a word, notepad or PDF document. -You **do not** need to submit the **graphics.py** module or any CSV files
Method of Submission:	Submitted online via Blackboard
Type of Feedback and Due Date:	Feedback during viva and written feedback and marks by 1st of Feb 2026

Assessment regulations
Refer to the following for clarification on what constitutes plagiarism, collusion and penalties for late submissions.

This is an individual coursework. You should not share your coursework or parts of your coursework with another student as this can cause you both to receive an allegation of collusion:

https://www.westminster.ac.uk/current-students/guides-and-policies/academic-matters/academic misconduct/collusion

Clarification on what constitutes plagiarism: https://www.westminster.ac.uk/current-students/guides-and-policies/academic-matters/academic misconduct/plagiarism

Penalty for Late Submission

If you submit your coursework late but within 24 hours or one working day of the specified deadline, 10 marks will be deducted from the final mark, as a penalty for late submission, except for work which obtains a mark in the range 40 – 49%, in which case the mark will be capped at the pass mark (40%). If you submit your coursework more than 24 hours or more than one working day after the specified deadline you will be given a mark of zero for the work in question unless a claim of Mitigating Circumstances has been submitted and accepted as valid.

It is recognised that on occasion, illness or a personal crisis can mean that you fail to submit a piece of work on time. In such cases you must inform the Campus Office in writing on a mitigating circumstances form, giving the reason for your late or non-submission. You must provide relevant documentary evidence with the form. This information will be reported to the relevant Assessment Board that will decide whether the mark of zero shall stand. For more detailed information regarding University Assessment Regulations, please refer to the following website:
http://www.westminster.ac.uk/study/current-students/resources/academic-regulation

This document contains:

- Coursework Brief and Description
- Important Notes on Your Submission
- CW Specification Tasks A to E
- Example of a successful code run
- Mark Scheme.

1. Coursework Brief and Description:

A European Airports Survey has collected a number of datasets detailing air traffic departures from a number of European airports for the same single twelve-hour period over the last 25 years.

The data has been tabulated and saved as a number of *.CSV* files (**test CSV files** are supplied – each one for a particular 12-hour period at a single Airport in a particular year).

You have been tasked with writing a program which analyses the data and returns selected information to allow the survey group to make informed decisions on European Air traffic.

Your program should allow the user to select and load **any** single CSV file (not just those supplied) as long as it has the correct structure and naming convention (Airport Code and year). It should *validate* the users file selection input as being in the correct format. **(Task A).**

It should analyse the data and summarise the specific outcomes required **(Task B).**

It should also save the outcome results as a text file **(Task C)** and use the 'graphics.py' module to display the results of some specific analysis in graphical form **(Task D: Histogram).** It should be able to loop and load a new CSV file **(Task E)**.

The CSV files

The data structure of the csv files is shown below with *each row* representing *one aircraft* departing the selected airport at a specific time over a 12hr period in the selected year (**below is just a possible first two rows as an example, see the supplied CSV files for full data list**).

Table1 – example of CSV data structure (two rows only)

Airport Code	Flight Num	Scheduled Depature	Actual Departure	Destination	Distance miles	Scheduled Arrival	Actual Arrival	Departure Terminal	Runway Num	Weather Conditions
LHR	BA880	00:01	00:01	FRA	407	01:36	01:36	2	1	18°C clear
LHR	LH632	00:03	00:03	AMS	230	01:23	01:23	2	1	18°C clear

Meaning of the Elements in Each Row (all are *string* data types)

1. **AirportCode:** The selected departure airport's three letter code (this is the same for all rows).
2. **FlightNum:** The flight Number of the departing aircraft, the *first two* characters are the *airline code* (see table 3).
3. **ScheduledDeparture:** Scheduled time of departure of this aircraft (Greenwich Mean Time).
4. **ActualDeparture:** Actual departure time of the aircraft (same as Scheduled departure if departed on time-GMT)
5. **Destination**: The three-letter airport code of the destination airport (see table 2)
6. **Distance miles:** The distance in miles from the selected departure airport to the destination airport.
7. **ScheduledArrival:** Scheduled arrival time of the aircraft (GMT).
8. **ActualArrival:** Actual arrival time of the aircraft (same as Scheduled Arrival if the aircraft arrived on time - GMT).
9. **DepartureTerminal:** The departing passenger terminal for the flight
10. **RunwayNum:** Departing runway number.
11. **WeatherConditions:** Temperature and weather for the hour of departure.

You are provided with a template python file ("cw_template.py") which *already loads* one of the supplied CSV files and converts the full dataset into a nested python list.

You should start with this template, rename and edit it to create your solution.

You are also provided with CSV files to use for testing purposes (these may not be the ones used to test your solution). **For any CSV files to be imported by your python file, they must be saved in the same directory (folder) as your python file.**

2. Important Notes on Your Submission

The code to load a CSV file and add it to a python list is already included and is executed by the template. You should gather and process the data from *this* list in your solution (the list is called 'data_list' in the template). You do not need to access any elements directly from the CSV file again once it has been loaded into data_list (until you load new CSV file when the code loops).

Your submitted python file must run stand alone in the Idle Shell window when downloaded to a folder containing the appropriate CSV files and the graphics.py module on a university computer (please check this before submission).

A.I. tools may be used only to help with your understanding of coding and approach. All of your submitted code should be written directly by you, using the techniques and ideas taught in the module.

- Use user-defined functions in your solution as appropriate (use docstring to add function description comments).
- Use descriptive names for your variables and functions.
- Comment your code extensively with descriptive comments explaining the code usage and structure.
- Reference within your code any code adapted from external or other sources.
- Use only the **"graphics.py"** module to render the histogram.
- You should not use any libraries or modules not already included in the template to complete your solution.
- The documentation for the graphics.py module is included in the brief folder.

The **only valid airports** for which data has been collected, are shown in the table below

Table 2 – All airport codes used in the survey

Airport Code	Full Airport Name
LHR	London Heathrow
MAD	Madrid Adolfo Suárez-Barajas
CDG	Charles De Gaulle International
IST	Istanbul Airport International
AMS	Amsterdam Schiphol
LIS	Lisbon Portela
FRA	Frankfurt Main
FCO	Rome Fiumicino
MUC	Munich International
BCN	Barcelona International

The **only valid airlines** for which data has been collected, are shown in the table below

Table 3 - all airline IATA codes used in the survey

IATA Code	Full Airline Name
BA	British Airways
AF	Air France
AY	Finnair
KL	KLM
SK	Scandinavian Airlines
TP	TAP Air Portugal
TK	Turkish Airlines
W6	Wizz Air
U2	easyJet
FR	Ryanair
A3	Aegean Airlines
SN	Brussels Airlines
EK	Emirates
QR	Qatar Airways
IB	Iberia
LH	Lufthansa

3. The Project Tasks

Task A. Input Validation (15 marks)

The template uses a string variable called '**selected_data_file**' set to a fixed, hard coded CSV file name (which is one of the supplied files - "CDG2021.csv". This CSV contains a deliberately short dataset for ease of testing).

The first task is to adapt and re-write this code such that the user may input ***any*** valid three letter departure *city code* followed by *any* valid *year* integer value and the application will load any appropriate CSV file in the same folder that has the correct naming format and data structure.

Note: Your code does not need to check whether or not a selected csv file exists in the folder.

This should be achieved using two prompts (validating the input each time). So, for example, if the user enters BCN to the first prompt and 2023 to the second, the program should load the file *BCN2023.csv* (the CSV with the data for Barcelona airport from 2023).

- For the city code prompt: The program should display "**Wrong code length - please enter a three-letter city code**" if the input is not three characters long.
- The program should display "**Unavailable city code - please enter a valid city code**" if the input is not one of the city codes in table 2.

- For the year prompt: the program should display "**Wrong data type - please enter a four-digit year value**" if the input is not a four-digit value (for example letters instead of integers).
- The program should display "**Out of range - please enter a value from 2000 to 2025**" if the input falls before 2000 or later than 2025.

- For task D (histogram) the program should validate the *airline code* input by checking that the entry is one of the airlines included in the data (see table 3) and ask the user to re-enter the code if it is not.

Your solution should be able to accept and process valid city and airline codes even if they are input as lower case. **You should Include code to catch and convert lower case letter input.**

Your program should then show the selected file name *Full* airport name and year before returning the Task 2 'outcomes' data (see example run below).

An Example of the Program Validation Task Running with User Input (shown in bold):

INPUT prompts

```
please enter a three-letter city code:CDGE
Wrong code length - please enter a three-letter city code:XCH
Unavailable city code - please enter a valid city code:cdg        <-allow for lowercase entry

Please enter the year required in the format YYYY:March
Wrong data type - please enter a four-digit year value:1997
Out of range - please enter a value from 2000 to 2025:2024
```

OUTPUT

```
**************************************************************************
CDG2024.csv selected - Planes departing Charles De Gaulle International 2024
**************************************************************************
```

The variable "selected_data_file" should now be set to **CDG2024.py**

Task B. Outcomes (25 marks)

The program should now process the data from the list 'data_list' calculate the results outcomes below and show them clearly to the shell window.

The specific information required to be extracted from the selected CSV

1. **The total number of departure flights for the selected file recorded in the 12-hour period.**
2. **The total number of flights departing from Terminal 2.**
3. **The total number of departures of flights that are under 600 miles.**
4. **The *total number* of departure flights by Air France aircraft.**
5. **The total number of flights departing in temperatures below 15^{O}C.**
6. **The average number of *British Airways* departures per hour (rounded to two decimal places).**
7. **The *percentage* of total departures that are *British Airways* aircraft (rounded to two decimal places).**
8. **The *percentage* of *Air France* flights with a delayed departure (rounded to two decimal places).**
9. **The total number of hours of rain in the twelve hours (rain values are recorded once every hour).**
10. **The full name of the *least* common destination (or names if more than one).**

Format for Task B results

The output to the shell window for task B should be *formatted* as below

```
********************************************************************************
File CDG2024.csv selected - Planes departing Charles De Gaulle International 2024
********************************************************************************

The total number of flights from this airport was 95
The total number of flights departing Terminal Two was 17
The total number of departures on flights under 600 miles was 57
There were 25 Air France flights from this airport
There were 0 flights departing in temperatures below 15 degrees
There was an average of 1.25 British Airways flights per hour from this airport
British Airways planes made up 15.79% of all departures
10.53% of Air France departures were delayed
There were 2 hours in which rain fell
The least common destinations are ['Lisbon Portela', 'Frankfurt Main']
```

Task C. Save Results as a Text File (10 Marks)

The program should collect The Selected airport name, year and all of the "outcomes" (part B) results. It should then save them to a text file with the filename "results.txt" in the format shown below.

When task E (Loop the program, load process a new CSV file) is implemented, the new results should be *added* to the existing results.txt file *without* overwriting the previously saved data.

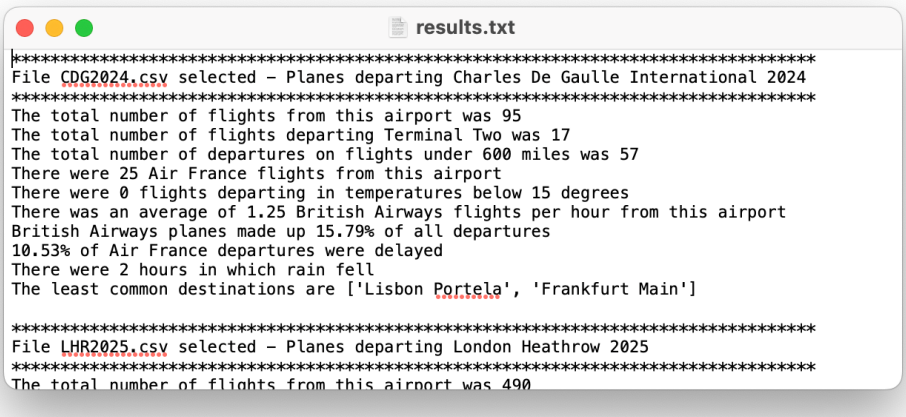


Figure 1 Example of .txt file content

Task D) Histogram (25 marks)

The program should now prompt:

```
Enter a two-character Airline code to plot a histogram:
```

The program should verify that the airline code entered is one of the valid airlines (those in table 3). If not, it should return…

```
Unavailable Airline code please try again.
Enter the two-character Airline code to plot a histogram:
```

When a valid code is entered, use the **"graphics.py"** module to pop up a window with a full horizontal histogram of the data in the format shown below with labeled y axis including hours (you should use your own look and feel and your own design and colour scheme).

The histogram should show the **total number of departing flights for the *selected* airline, for each hour of the twelve-hour survey** including numerical values for the number of flights each hour

The histogram should also include a title with the *full* selected **airline** name, the *full* departure **airport** name and the **year** of the CSV (see image below - results shown are illustrative only)

Bars should scale depending on the highest number of flights in a single hour so that it can accommodate any number of flights. It should make good use of colour and be clear and legible.

It *must* use the **graphics.py** module ***(do not use any other graphing module for this task).***

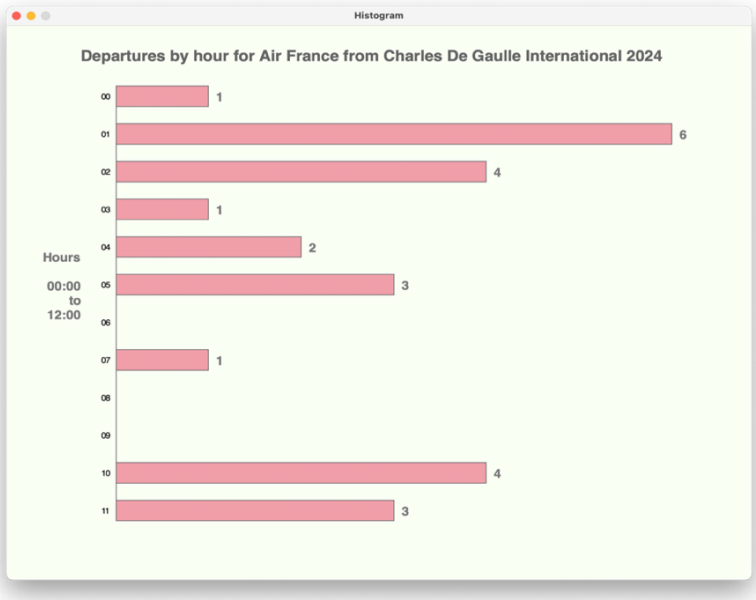


Figure 2 Example of a Histogram pop up

Task E) Program Loops on Request, Loads and Processes a New CSV file (10 Marks)

The program should then allow the user to select a **new** data set (CSV) for a different city code and year input and run a new analysis. After the first run has completed, it should ask…

```
Do you want to select a new data file? Y/N:
```

If the user inputs **"Y"**, The program should clear all data from the previous run (and clear data-list) and re-run to import and process the data from a new CSV file selected by the user. It should then output the new results (updating the text file) and histogram. The program should loop and process any loaded CSV files until the user enters **'N'** to the question "Do you want to select a new data file? Y/N". **You should Include code to catch lower case letter input.**

4. Example of a Single Full Code Run (user inputs shown in bold)

INPUT prompts

```
Please enter the three-letter code for the departure city required:MUNICH
Wrong code length - please enter a three-letter city code:XCH
Unavailable city code - please enter a valid city code:MUC

Please enter the year required in the format YYYY:March
Wrong data type - please enter a four-digit year value:1997
Out of range - please enter a value from 2000 to 2025:2025
```

OUTPUT

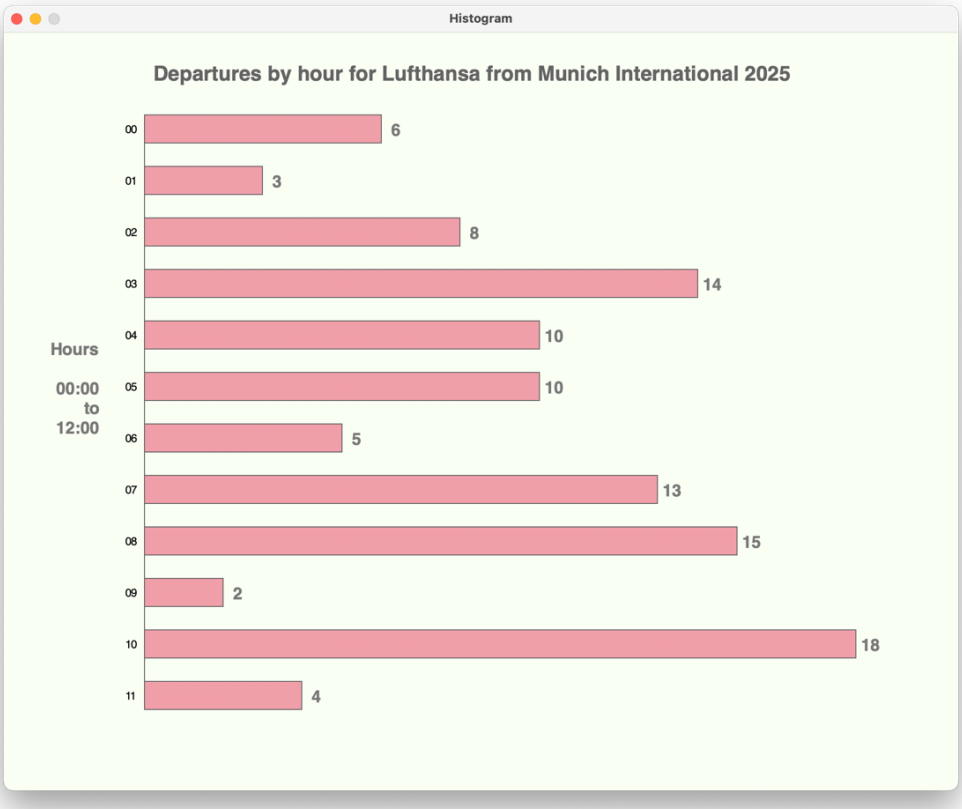
```
**********************************************************************
File MUC2025.csv selected - Planes departing Munich International 2025
**********************************************************************
The total number of flights from this airport was 397
The total number of flights departing Terminal Two was 193
The total number of departures on flights under 600 miles was 200
There were 46 Air France flights from this airport
There were 11 flights departing in temperatures below 15 degrees
There was an average of 7.42 British Airways flights per hour from this airport
British Airways planes made up 22.42% of all departures
4.79% of Air France departures were delayed
There were 6 hours in which rain fell
The least common destination is ['Madrid']
```

<–save outcomes to a text file

INPUT prompt

```
Enter a two-character Airline code to plot a histogram:QU
Unavailable Airline code please try again:LH
```

OUTPUT (pop up window)



INPUT prompt

```
Do you want to select a new data file? Y/N:N
```

OUTPUT

```
Thank you. End of run
```

5. Mark Scheme

Table 4 Breakdown of Marks for Coursework	
Task and Success Criteria	**Marks per Task**
Task A: Validation (15 Marks)	
Validation A1. Identifies when selected departure airport code input is the wrong length or not one of the available codes from table 2 (a three-letter airport code). (2) When input is incorrect - prompts user for the correct format. (1)	**3**
Validation A2. Identifies when year input is the wrong data type or range (a four-digit integer in the range 2000 to 2025) (2) When input is incorrect - prompts user for the correct format and range (1)	**3**
Validation A3. Combines the airport code and date and loads the correct, selected CSV file.	**6**
Validation A4. For task D - Identifies when selected *airline* code for the histogram is not one of those in table 3. (2) When the airline code input is invalid – informs the user prompts them for the correct two-character airline code format. (1)	**3**
Task B: Output Outcomes (25 Marks)	
Outputs B1. Data processed correctly Returns the correct outcome values for the selected CSV file to the shell window.	**20**
Outputs B2. Outputs are formatted well and correctly rendered with supporting text	**5**
Task C: Compile and save processed results to a text file (10 Marks)	
Text File C1. All results correctly saved to a text file called "results.txt"	**6**
Text File C2. When a new CSV is processed, the new outcomes should be appended to results.txt without overwriting the results from the previous program run.	**4**
Task D: Histogram (25 Marks)	
Histogram D1. Program requests a two-character airline code from the user, saves and uses the input.	**2**
Histogram D2. Histogram window pops up with title, including the *full* airport and airline names, and year details.	**4**
Histogram D3. The y-axis with a label and hour values is rendered to the screen.	**2**
Histogram D4. Histogram bars are drawn that reflect the data accurately (with numerical values).	**9**
Histogram D5. The bars scale changes depending on maximum value (so bars are always on screen).	**4**
Histogram D6. The pop-up window and content are clear with good user experience design.	**4**
Task E: Program Loops for Multiple CSV Files (10 Marks)	
Program Loop E1. When user inputs "Y" to the query, the program asks for new input. When user inputs **"N"** the program ends (lower case input should be allowed)	**2**
Program Loop E2. The program **reruns** clearing all variables and returns correct outputs and plots the histogram for the **new** selected CSV.	**8**
Good Programming Practice (15 Marks)	
Programming Practice 1. Good use of useful naming conventions and well formatted and commented code.	**5**
Programming Practice 2. Code makes effective use of user defined functions, loops and conditionals	**10**