|  |  |
| --- | --- |
| Blue letters on a white background  Description automatically generated | Text  Description automatically generated with medium confidence |

# Student Assessment Submission and Declaration

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

|  |  |  |  |
| --- | --- | --- | --- |
| Student name: | Pawel Mateusz Czerepski | ESL ID No | 120071 |
| BSU ID No | 575378 |
| Submission date: | | | 07.02.2025 |
|  | | |  |
| Programme: | Computting foundation | | |
| Module name and code: | Software foundations CPUF001 | | |
| Title: | Data processing programme | | |
| Assessor name: |  | | |

**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: | Pawel Mateusz Czerepski | Date: | 07.02.2025 |

**Introduction**

This project involved developing a Data Processing Program that takes grade information for students in a file in CSV format and performs calculation operations, including mean calculation of scores, assigning a letter grade, and generating a report summary. Requirements were examined to preserve organized Python programming, command-line run (CLI), script modularity, and checks for errors. Best practice in software development was utilized in its development, and it is designed for ease, efficiency, and maintainability. The script and documentation for the entire project can be viewed at GitHub: [GitHub Repository Link].

**Solution Design**

**Flowchart Representation**

A diagram of a diagram

AI-generated content may be incorrect.

### **Walkthrough of the Program and Scripts**

### The program employs a pipeline with a specific sequence for effective information processing. The process starts with reading and checking information. Student grade information is first read in the format of a CSV file. Before processing, file integrity checking is conducted to confirm that values are in the correct format and that no value gaps exist. If any mistakes, including non-numeric values for scores and value gaps, arise, correct messages of errors are printed and displayed to a user. Wrong values are repaired or removed for future processing.

### After validation, the processing stage comes in for the program. It takes individual subject marks for a student and extracts several values important for academic performance analysis. It takes an average mark by summing the individual subject marks and then dividing them by a subject's subject. It also identifies a top and a minimum mark for a student in a specific topic to understand a student's strong and weak subjects individually. It then utilizes a letter grading scheme with predefined thresholds for grading student performance in categories, with an opportunity for uniform evaluation of results.

### Once data processing is completed, the program shifts to printing out results. Student information processed, including individual marks, mean, high and low marks, and grades, is printed out to an output file for convenient retrieval and further analysis of results. In addition, a report summary is generated that provides a high-level picture of students' performance, including key statistics such as students processed, best-scoring students, and worst-scoring students. This report is in a standalone text file for a readable and concise view of the data set.

### Finally, the program is executed via a command line, with an option for specifying dynamically both output and input file locations. The automated sequence of execution is supported via a shell script, guaranteeing that the software will execute an unproblematic ally with no intervention at all. With automation, efficiency is increased via the effective processing of large datasets and the avoidance of any user errors. With such an arrangement of sequence of execution, the software provides effective data processing, effective reporting, and effective automation for real-life scenarios.

### **Technical Breakdown of Key Features**

### File Handling and Error Handling

### The program will verify that files have been entered for processing.

### If an input file doesn't exist, an error message will be displayed, and processing will terminate.

### Example code section:

**A screen shot of a computer program

AI-generated content may be incorrect.**

### Data Calculation and Analysis

### The program computes each student's average mark via a summation of individual subjects and then a subject count divide.

### The highest and lowest subjects' scores are computed, too.

### Example implementation:

**A screen shot of a computer program

AI-generated content may be incorrect.**

1. **Grade Assignment**

The grading system follows a structured threshold.

Example grading function:

A screen shot of a computer program

AI-generated content may be incorrect.

1. **Report Creation**

The program generates a report with a key observation, including:

1. Total students processed
2. The student with the best and poorest average mark
3. Grade distribution

This enhances usability, making performance insights easily accessible.

Example report output:

A screen shot of a computer code

AI-generated content may be incorrect.

**. Command-Line Argument Handling (argparse)**

* The program uses argparse to accept user-defined input and output file paths.

Example implementation:

A screen shot of a computer code

AI-generated content may be incorrect.

**6. Scalability and Future Updates**

The program is designed with modularity in its function, and it can expand simply.

Potential future development:

1. Integration with databases (e.g., MySQL, PostgreSQL) for storing persistent information.
2. Web-based interface for real-time report visualization and real-time data input (Zhang, 2019).
3. Graphical representation of student performance trends with **Matplotlib** and/or **Seaborn.**

The solution is efficient, well-designed, and can handle real-life student processing with minimum human intervention. Its modularity allows it to adapt quickly; new capabilities can be added whenever possible.

## **Reflective Evaluation**

Developing this project helped me better grasp structured programming, data processing techniques, and automation. Having a robust mechanism for dealing with errors was one of the most significant experiences in that it kept missing or corrupt information from impacting execution. A proper mechanism for dealing with exceptions and producing useful error messages added to usability and tolerance for unforeseen input errors. In addition, appreciating the value of a modular structure became apparent by sectioning off the program into discrete chunks for reading in, processing, and outputting information, improving maintainability and expandability. Implementing the argparse module for command-line input was an effective mechanism for providing ease in running the program with different datasets and, thus, a flexible and adaptable tool.

A key difficulty that was encountered included processing large datasets efficiently. Handling long data files highlighted the need for optimizations, such as efficiently using data structures and minimizing repetitive computations. Optimizations in the future can include using caching algorithms or utilizing database management to deal with large sets of students' records more efficiently. Having tools for data visualization will have a significant impact on output readability. To date, numerical output and summary reports have been produced. Still, graphical visualization through packages such as Matplotlib or Seaborn can make student performance trends easier to interpret and enable effective decision-making for instructors and school administration (Sakib, 2022).

Additionally, gaining machine learning and database management expertise can make the program even smarter in its predictive capabilities, offering students personalized learning recommendations.

**Conclusion**

The student processing program successfully reads student scores in an input file in CSV format, calculates mean, minimum, and maximum, and a grade following a predefined grading scheme, and stores them in an output file in CSV format. It generates a summary report, providing a general picture of student performance.

**Appendix**

Data\_Processing.py:

Data\_Processing.py:

import csv  # Importing CSV module to handle file reading/writing

import sys  # Importing sys module to handle program exit on critical errors

import argparse  # Importing argparse to handle command-line arguments

import os  # Importing os module for file existence checking

# Deliverables and Application Requirements Included

#

# This program reads student grades from a CSV file passed via the command line,

# processes their average scores, finds the highest and lowest scores,

# assigns letter grades, and outputs results to another CSV file.

# Additionally, a report summarizing processing activity is generated.

# Function to read student data from a CSV file

def read\_student\_data(filename):

    """Reads student data from a CSV file passed via the command line with error handling."""

    data = []

    if not os.path.exists(filename):

        print(f"Error: The file '{filename}' does not exist.")

        sys.exit(1)

    try:

        with open(filename, 'r', encoding='utf-8') as file:

            csv\_reader = csv.reader(file)

            header = next(csv\_reader)

            for row in csv\_reader:

                if len(row) < 6:

                    print(f"Warning: Incomplete data in row: {row}")

                    continue

                try:

                    data.append({

                        'Name': row[0],

                        'Art': int(row[1]),

                        'Chemistry': int(row[2]),

                        'Science': int(row[3]),

                        'English': int(row[4]),

                        'Math': int(row[5])

                    })

                except ValueError as e:

                    print(f"Error: Invalid data in row {row}: {e}")

                    sys.exit(1)

    except FileNotFoundError:

        print(f"Error: {filename} not found.")

        sys.exit(1)

    except Exception as e:

        print(f"An unexpected error occurred: {e}")

        sys.exit(1)

    return data

def calculate\_average(student):

    """Calculates the average score of a student."""

    total = sum([student['Art'], student['Chemistry'], student['Science'], student['English'], student['Math']])

    return total / 5

def find\_highest\_lowest(student):

    """Finds the highest and lowest scores for a student."""

    scores = [student['Art'], student['Chemistry'], student['Science'], student['English'], student['Math']]

    return max(scores), min(scores)

def assign\_grade(score):

    """Assigns a letter grade based on score."""

    if score >= 90:

        return "A"

    elif score >= 80:

        return "B"

    elif score >= 70:

        return "C"

    elif score >= 60:

        return "D"

    elif score >= 50:

        return "E"

    else:

        return "F"

def process\_student\_results(students\_data):

    """Processes student data to compute averages, highest/lowest scores, and grades."""

    results = []

    for student in students\_data:

        average = calculate\_average(student)

        highest, lowest = find\_highest\_lowest(student)

        results.append({

            'Name': student['Name'],

            'Art': student['Art'],

            'Chemistry': student['Chemistry'],

            'Science': student['Science'],

            'English': student['English'],

            'Math': student['Math'],

            'Average': average,

            'Highest\_Score': highest,

            'Lowest\_Score': lowest,

            'Overall\_Grade': assign\_grade(average),

            'Art\_Grade': assign\_grade(student['Art']),

            'Chemistry\_Grade': assign\_grade(student['Chemistry']),

            'Science\_Grade': assign\_grade(student['Science']),

            'English\_Grade': assign\_grade(student['English']),

            'Math\_Grade': assign\_grade(student['Math'])

        })

    return results

def write\_results\_to\_csv(results, filename):

    """Writes student results to an output CSV file with error handling."""

    try:

        if results:

            header = ['Name', 'Art', 'Chemistry', 'Science', 'English', 'Math', 'Average', 'Highest\_Score', 'Lowest\_Score', 'Overall\_Grade', 'Art\_Grade', 'Chemistry\_Grade', 'Science\_Grade', 'English\_Grade', 'Math\_Grade']

            with open(filename, 'w', newline='', encoding='utf-8') as file:

                writer = csv.DictWriter(file, fieldnames=header)

                writer.writeheader()

                writer.writerows(results)

            print(f"Results successfully written to {filename}")

        else:

            print("No results to write.")

    except Exception as e:

        print(f"Error: Failed to write to {filename}: {e}")

        sys.exit(1)

def generate\_summary\_report(results, report\_filename):

    """Generates a summary report of the processing activity with error handling."""

    try:

        total\_students = len(results)

        highest\_avg = max(results, key=lambda x: x['Average'])

        lowest\_avg = min(results, key=lambda x: x['Average'])

        with open(report\_filename, 'w', encoding='utf-8') as file:

            file.write("Summary Report\n")

            file.write("====================\n")

            file.write(f"Total Students Processed: {total\_students}\n")

            file.write(f"Highest Average Score: {highest\_avg['Name']} - {highest\_avg['Average']}\n")

            file.write(f"Lowest Average Score: {lowest\_avg['Name']} - {lowest\_avg['Average']}\n")

        print(f"Summary report successfully written to {report\_filename}")

    except Exception as e:

        print(f"Error: Failed to write report to {report\_filename}: {e}")

        sys.exit(1)

if \_\_name\_\_ == "\_\_main\_\_":

    parser = argparse.ArgumentParser(description="Process student grades from a CSV file.")

    parser.add\_argument("input\_file", help="Path to the input CSV file")

    parser.add\_argument("output\_file", help="Path to the output CSV file")

    parser.add\_argument("report\_file", help="Path to the summary report file")

    args = parser.parse\_args()

    students\_data = read\_student\_data(args.input\_file)

    results = process\_student\_results(students\_data)

    write\_results\_to\_csv(results, args.output\_file)

    generate\_summary\_report(results, args.report\_file)

    print("Processing complete. Goodbye!")

Student\_grades.csv:

Name,Art,Chemistry,Science,English,Math

Iris,57,60,3,97,44

Zane,76,45,92,23,100

Quinn,3,81,20,74,95

Maeve,73,37,74,33,94

Arlo,4,88,13,2,35

Hazel,98,58,75,12,7

Owen,7,68,48,55,86

Nora,28,50,57,61,21

Levi,86,96,96,32,64

Ava,57,44,91,24,42

Elias,61,22,6,45,93

Ruby,14,8,18,83,91

Felix,80,96,26,5,30

Isla,98,5,11,44,67

Kai,39,9,91,30,59

Run\_script.bat:

#!/bin/bash

# Script to run the data processing Python program

# Define input, output, and report filenames

INPUT\_FILE="students\_grades.csv"

OUTPUT\_FILE="student\_results.csv"

REPORT\_FILE="summary\_report.txt"

# Check if Python is installed

if ! command -v python3 &> /dev/null

then

    echo "Python3 is not installed. Please install Python3 to continue."

    exit 1

fi

# Execute the Python program with the correct arguments

echo "Running Data Processing Program..."

python3 data\_processing.py "$INPUT\_FILE" "$OUTPUT\_FILE" "$REPORT\_FILE"

echo "Program execution complete."

**Output:**

**Student\_results.csv:**

Name,Art,Chemistry,Science,English,Math,Average,Highest\_Score,**Lowest\_Score,**Overall\_Grade,Art\_Grade,Chemistry\_Grade,Science\_Grade,English\_Grade,Math\_Grade

Iris,57,60,3,97,44,52.2,97,**3,**E,E,D,F,A,F

Zane,76,45,92,23,100,67.2,100,**23,**D,C,F,A,F,A

Quinn,3,81,20,74,95,54.6,95,**3,**E,F,B,F,C,A

Maeve,73,37,74,33,94,62.2,94,**33,**D,C,F,C,F,A

Arlo,4,88,13,2,35,28.4,88,**2,**F,F,B,F,F,F

Hazel,98,58,75,12,7,50.0,98,**7,**E,A,E,C,F,F

Owen,7,68,48,55,86,52.8,86,**7,**E,F,D,F,E,B

Nora,28,50,57,61,21,43.4,61,**21,**F,F,E,E,D,F

Levi,86,96,96,32,64,74.8,96,**32,**C,B,A,A,F,D

Ava,57,44,91,24,42,51.6,91,**24,**E,E,F,A,F,F

Elias,61,22,6,45,93,45.4,93,**6,**F,D,F,F,F,A

Ruby,14,8,18,83,91,42.8,91,**8,**F,F,F,F,B,A

Felix,80,96,26,5,30,47.4,96,**5,**F,B,A,F,F,F

Isla,98,5,11,44,67,45.0,98,**5,**F,A,F,F,F,D

Kai,39,9,91,30,59,45.6,91,**9,**F,F,F,A,F,E

Summaart\_report.txt:

Summary Report

====================

Total Students Processed: 15

Highest Average Score: Levi - 74.8

Lowest Average Score: Arlo - 28.4

To run this, Open Visual Studio Code. Install Python if you don’t have it in your machine. Create a folder, e.g., Python, open the vscode, and create data\_processing.py, students\_grade.csv, and run\_script.bat. Load the content as I have done above. Open the vscode terminal, head to Gitbash, and run this command: bash run\_script.bat. This will be the Output:

The Output should be Student\_results.csv and a Summary\_report.txt.

A screenshot of a computer program

AI-generated content may be incorrect.

**References**

Zhang, Q., 2019. Web-based medical data visualization and information sharing towards application in distributed diagnosis. *Informatics in medicine unlocked*, *14*, pp.69-81. <https://doi.org/10.1016/j.imu.2018.10.010>

Sakib, S.N., 2022. Data Visualization in Data Science. <https://doi.org/10.33774/coe-2022-c3kk5-v2>