



“Python Programming”

Assignment-2

Topic – Analysing and Reporting Student Grades

Submitted by – Priyam Sharma

Roll no - 2501730184

Course – B. Tech CSE (AI & ML)

Section – B

Faculty name- Mr Sameer Farooq

Introduction

The objective of this Python Lab assignment is to develop a GradeBook Analyser, a program capable of reading student marks, performing statistical analysis, assigning grades, and presenting results clearly. The project demonstrates Python skills such as:

- File handling
- Functions and modular programming
- Data validation
- Statistics (mean, median)
- Grade assignment logic
- Table formatting for output

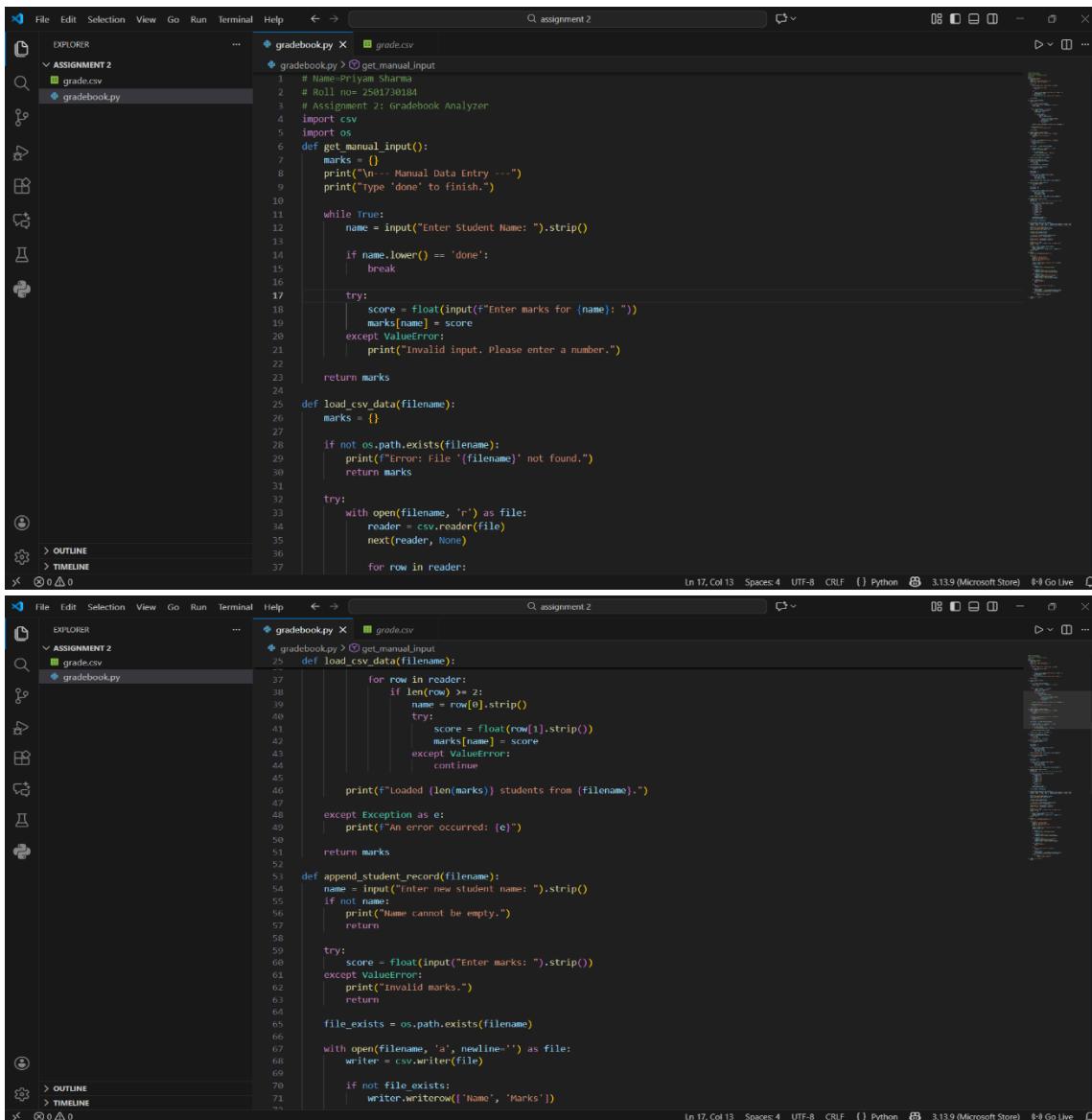
Objectives

- To develop a Python program that can efficiently analyse student marks.
- To implement file handling using CSV files
- To practice modular programming using functions
- To design a grading system based on score ranges
- To generate summaries for academic performance evaluation

Program Description

The program developed (gradebook_analyzer.py) is designed to analyse student marks and generate performance summaries. It allows users to either enter marks manually or load them from a CSV file.

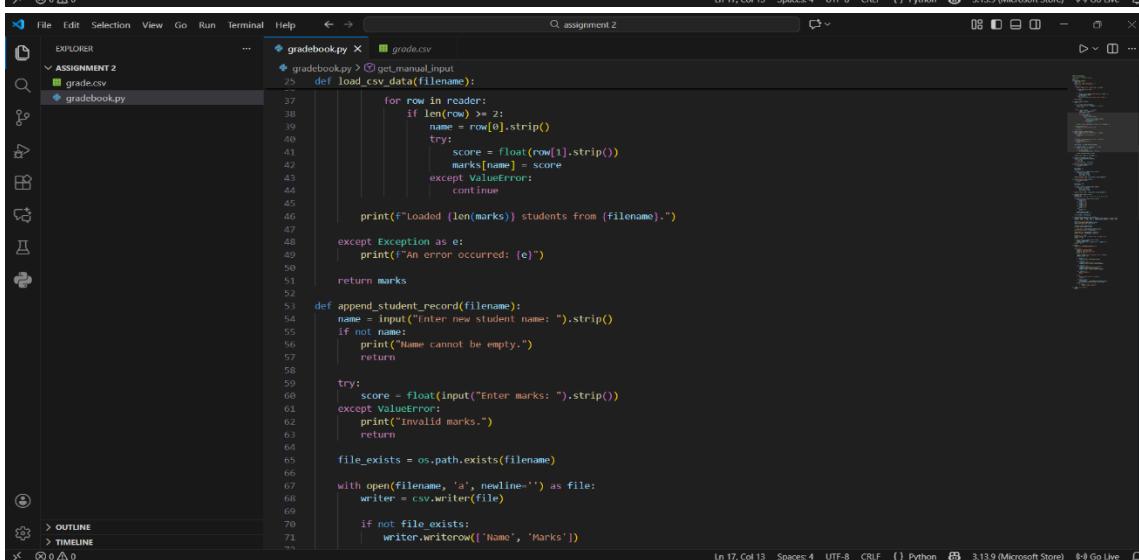
Program Code



The image shows two instances of Microsoft Visual Studio Code side-by-side, both displaying the same Python script: gradebook_analyzer.py. The script is used for managing student marks, either through manual input or by reading from a CSV file.

```
File Edit Selection View Go Run Terminal Help <- > Q assignment 2
EXPLORER gradebook.py x grade.csv
ASSIGNMENT 2 grade.csv
gradebook.py
gradebook.py > get_manual_input
1 # Name-Priyam Sharma
2 # Roll no- 2501730184
3 # Assignment 2: Gradebook Analyzer
4 import csv
5 import os
6 def get_manual_input():
7     marks = {}
8     print("\n-- Manual Data Entry --")
9     print("Type 'done' to finish.")
10
11    while True:
12        name = input("Enter Student Name: ").strip()
13
14        if name.lower() == 'done':
15            break
16
17    try:
18        score = float(input(f"Enter marks for {name}: "))
19        marks[name] = score
20    except ValueError:
21        print("Invalid input. Please enter a number.")
22
23    return marks
24
25 def load_csv_data(filename):
26     marks = {}
27
28     if not os.path.exists(filename):
29         print(f"Error: File '{filename}' not found.")
30         return marks
31
32     try:
33         with open(filename, 'r') as file:
34             reader = csv.reader(file)
35             next(reader, None)
36
37             for row in reader:
38                 if len(row) >= 2:
39                     name = row[0].strip()
40
41                     try:
42                         score = float(row[1].strip())
43                         marks[name] = score
44                     except ValueError:
45                         continue
46
47                     print(f"Loaded {len(marks)} students from {filename}.")
48
49     except Exception as e:
50         print(f"An error occurred: {e}")
51
52     return marks
53
54 def append_student_record(filename):
55     name = input("Enter new student name: ").strip()
56
57     if not name:
58         print("Name cannot be empty.")
59         return
60
61     try:
62         score = float(input("Enter marks: ").strip())
63     except ValueError:
64         print("Invalid marks.")
65
66     file_exists = os.path.exists(filename)
67
68     with open(filename, 'a', newline='') as file:
69         writer = csv.writer(file)
70
71         if not file_exists:
72             writer.writerow(['Name', 'Marks'])
```

Ln 17, Col 13 | Spaces: 4 | UTF-8 | CRLF | Python | 3.13.9 (Microsoft Store) | Go Live



```
File Edit Selection View Go Run Terminal Help <- > Q assignment 2
EXPLORER gradebook.py x grade.csv
ASSIGNMENT 2 grade.csv
gradebook.py
gradebook.py > get_manual_input
25 def load_csv_data(filename):
26     marks = {}
27
28     for row in reader:
29         if len(row) >= 2:
30             name = row[0].strip()
31
32             try:
33                 score = float(row[1].strip())
34                 marks[name] = score
35             except ValueError:
36                 continue
37
38             print(f"Loaded {len(marks)} students from {filename}.")
39
40     except Exception as e:
41         print(f"An error occurred: {e}")
42
43     return marks
44
45 def append_student_record(filename):
46     name = input("Enter new student name: ").strip()
47
48     if not name:
49         print("Name cannot be empty.")
50         return
51
52     try:
53         score = float(input("Enter marks: ").strip())
54     except ValueError:
55         print("Invalid marks.")
56
57     file_exists = os.path.exists(filename)
58
59     with open(filename, 'a', newline='') as file:
60         writer = csv.writer(file)
61
62         if not file_exists:
63             writer.writerow(['Name', 'Marks'])
64
65         writer.writerow([name, score])
```

Ln 17, Col 13 | Spaces: 4 | UTF-8 | CRLF | Python | 3.13.9 (Microsoft Store) | Go Live

```
File Edit Selection View Go Run Terminal Help ← → Q_assignment 2

EXPLORER
ASSIGNMENT 2
gradebook.py
grade.csv

gradebook.py > get.manual.input
53 def append_student_record(filename):
54     writer.writerow([name, score])
55
56     print(f"Added {name} to {filename}.")
57
58 def calculate_average(marks_dict):
59     scores = list(marks_dict.values())
60     if not scores:
61         return 0.0
62     return sum(scores) / len(scores)
63
64 def find_max_score(student_scores):
65     if not student_scores:
66         return
67
68     max_score = -1
69     max_student = ""
70
71     for name, score in student_scores.items():
72         if score > max_score:
73             max_score = score
74             max_student = name
75
76     print(f"HIGHEST SCORE: {max_score} by {max_student}")
77
78 def find_min_score(student_scores):
79     if not student_scores:
80         return
81
82     min_score = 101
83     min_student = ""
84
85     for name, score in student_scores.items():
86         if score < min_score:
87             min_score = score
88             min_student = name
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
```

The screenshot shows a code editor with the following details:

- File Explorer:** Shows a folder named "ASSIGNMENT 2" containing "grade.csv".
- Code Editor:** The file "gradebook.py" is open, displaying Python code for a gradebook system.
- Code Content:**

```
File Edit Selection View Go Run Terminal Help ← → Q assignment 2

gradebook.py X grade.csv

gradebook.py > get_manual_input
97 def find_min_score(student_scores):
98     print("LOWEST SCORE: {min_score} by {min_student}")
99
100
101 def assign_grades(student_scores):
102     grades = {}
103     distribution = {"A": 0, "B": 0, "C": 0, "D": 0, "F": 0}
104
105     for name, score in student_scores.items():
106         if score >= 90:
107             grade = "A"
108         elif score >= 80:
109             grade = "B"
110         elif score >= 70:
111             grade = "C"
112         elif score >= 60:
113             grade = "D"
114         else:
115             grade = "F"
116
117         grades[name] = grade
118         distribution[grade] += 1
119
120     return grades, distribution
121
122 def print_summary(student_scores, grades):
123     passed = [name for name, score in student_scores.items() if score >= 40]
124     failed = [name for name, score in student_scores.items() if score < 40]
125
126     print("\n--- class Statistics ---")
127     avg = calculate_average(student_scores)
128     print(f"Average Score: {avg:.2f}")
129
130     find_max_score(student_scores)
131     find_min_score(student_scores)
132
133     final_dist = assign_grades(student_scores)
134     print(f"\nGrade Counts: {final_dist}")

Line 17, Col 13 Spaces: 4 UTF-8 CRLF () Python 3.13.9 (Microsoft Store) ⚡ Go Live
```

```
File Edit Selection View Go Run Terminal Help <- > Q assignment 2
EXPLORER ASSIGNMENT 2 grade.csv gradebook.py
gradebook.py > get_manual_input
132     def print_summary(student_scores, grades):
133         ...
134         print("Passed: {len(passed)} students")
135         print("Failed: {len(failed)} students")
136
137         print("\n" + "="*40)
138         print("{name:<20} | {Marks:<10} | {Grade:<5}")
139         print("." * 40)
140
141         for name, score in student_scores.items():
142             grade = grades[name]
143             print("{name:<20} | {score:<10.1f} | {grade:<5}")
144             print("." * 40)
145
146     def main():
147         print("\n*** GRADEBOOK ANALYZER ***")
148
149         while True:
150             print("1. Manual Entry")
151             print("2. Load from CSV")
152             print("3. Add Student to CSV")
153             print("4. Exit")
154
155             choice = input("Select an option (1-4): ").strip()
156             student_scores = {}
157
158             if choice == '1':
159                 student_scores = get_manual_input()
160
161             elif choice == '2':
162                 filename = input("Enter CSV filename: ")
163                 student_scores = load_csv_data(filename)
164
165             elif choice == '3':
166                 filename = input("Enter CSV filename: ")
167                 append_student_record(filename)
168                 student_scores = load_csv_data(filename)
169
170             elif choice == '4':
171                 print("Goodbye!")
172                 break
173
174             else:
175                 print("Invalid choice. Try again.")
176                 continue
177
178             if student_scores:
179                 final_grades, _ = assign_grades(student_scores)
180                 print_summary(student_scores, final_grades)
181
182             else:
183                 if choice in ['1', '2', '3']:
184                     print("No data loaded.")
185
186             if __name__ == "__main__":
187                 main()
```

Ln 17, Col 13 Spaces: 4 UTF-8 CRLF () Python 3.13.9 (Microsoft Store) ⚡ Go Live

```
File Edit Selection View Go Run Terminal Help <- > Q assignment 2
EXPLORER ASSIGNMENT 2 grade.csv gradebook.py
gradebook.py > get_manual_input
158     def main():
159         ...
160         print("1. Manual Entry")
161         print("2. Load from CSV")
162         print("3. Add Student to CSV")
163         print("4. Exit")
164
165         choice = input("Select an option (1-4): ").strip()
166         student_scores = {}
167
168         if choice == '1':
169             student_scores = get_manual_input()
170
171         elif choice == '2':
172             filename = input("Enter CSV filename: ")
173             student_scores = load_csv_data(filename)
174
175         elif choice == '3':
176             filename = input("Enter CSV filename: ")
177             append_student_record(filename)
178             student_scores = load_csv_data(filename)
179
180         elif choice == '4':
181             print("Goodbye!")
182             break
183
184         else:
185             print("Invalid choice. Try again.")
186             continue
187
188         if student_scores:
189             final_grades, _ = assign_grades(student_scores)
190             print_summary(student_scores, final_grades)
191
192         else:
193             if choice in ['1', '2', '3']:
194                 print("No data loaded.")
195
196         if __name__ == "__main__":
197             main()
```

Ln 17, Col 13 Spaces: 4 UTF-8 CRLF () Python 3.13.9 (Microsoft Store) ⚡ Go Live

Sample Output

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface with the following details:

- File Explorer:** Shows files: `gradebook.py`, `grade.csv`, and `gradebook.py`.
- Code Editor:** Displays the `gradebook.py` script:

```
gradebook.py
gradebook.py > ...
160 def main():
161     ...
162     else:
163         if choice in ["1", "2", "3"]:
164             print("No data loaded.")
165
166     if __name__ == "__main__":
167         main()
```
- Terminal:** Shows the execution of the script and its output:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Python Debug Console + v x

Enter CSV filename: grade.csv
Loaded 3 students from grade.csv.

--- Class Statistics ---
Average Score: 73.33
HIGHEST SCORE: 94.0 by Prince
LOWEST SCORE: 45.0 by shaurya
Grade Counts: {'A': 1, 'B': 1, 'C': 0, 'D': 0, 'F': 1}
Passed: 3 students
Failed: 0 students

=====
Name      | Marks | Grade
-----
Prince   | 94.0  | A
aryan    | 81.0  | B
shaurya | 45.0  | F

1. Manual Entry
2. Load from CSV
3. Add Student to CSV
4. Exit
Select an option (1-4):
```
- Status Bar:** Shows the current file is `gradebook.py`, line 200, column 11, with 200 lines of code, 11 spaces, and using UTF-8 encoding.

Conclusion

The *GradeBook Analyser* is an effective Python program that reads student marks, performs statistical analysis, assigns grades, and displays results clearly. It demonstrates essential programming skills such as file handling, functions, and data processing, providing a practical solution for evaluating student performance.