Student Grade Analyzer — Instructor & Student Guide

Print / Save as PDF

A Python CLI app to compute, classify, and export student performance data.

> **Table of Contents**
>
>

1. Overview & Learning Goals

The Student Grade Analyzer helps automate report generation for student test results. Users can enter names and scores, view statistics (mean, highest, lowest), assign letter grades, and export results as a CSV file.

- Learn to use Python's `csv` module for data storage.
- Reinforce loops, lists, dictionaries, and functions.
- Practice input validation and conditional logic.
- Simulate real-world data processing workflows.

2. Requirements & Setup

- Python 3.8 or newer.
- Basic understanding of functions and loops.
- Any text editor (e.g., IDLE, VS Code, Sublime).

> **Tip:** For testing, you can start with a small dataset (3–5 students) and expand later.

3. Folder Structure

```
StudentGradeAnalyzer/


├─ grade_analyzer.py
└─ students.csv (auto-created on export)
```

4. Annotated Sample Code

```
 # grade_analyzer.py


import csv

def get_grade(score):
if score >= 70: return "A"
elif score >= 60: return "B"
elif score >= 50: return "C"
elif score >= 45: return "D"
else: return "F"

def analyze_students():
students = []
while True:
name = input("Enter student name (or 'done' to finish): ").strip()
if name.lower() == "done":
break
try:
score = float(input(f"Enter {name}'s score (0–100): "))
if not 0 <= score <= 100:
print("Score must be between 0 and 100.")
continue
except ValueError:
print("Invalid score! Please enter a number.")
continue
students.append({"Name": name, "Score": score, "Grade": get_grade(score)})

if not students:
    print("No data entered. Exiting program.")
    return

total = sum(s["Score"] for s in students)
average = total / len(students)
top_student = max(students, key=lambda x: x["Score"])
low_student = min(students, key=lambda x: x["Score"])

print("\n--- Class Summary ---")
print(f"Total Students: {len(students)}")
print(f"Average Score: {average:.2f}")
print(f"Highest: {top_student['Name']} ({top_student['Score']})")
print(f"Lowest: {low_student['Name']} ({low_student['Score']})")

with open("students.csv", "w", newline="") as f:
    writer = csv.DictWriter(f, fieldnames=["Name", "Score", "Grade"])
    writer.writeheader()
    writer.writerows(students)
print("\nData exported successfully to students.csv")
```

```
if name == "main":
analyze_students()
```

5. Step-by-Step Build Instructions

1. Create a folder named `StudentGradeAnalyzer`.
2. Inside it, create a new file named `grade_analyzer.py`.
3. Copy the sample code above.
4. Run the program in your terminal or IDLE (press F5).
5. Input names and scores. Type `done` when finished.
6. View the summary report in your terminal.
7. Open `students.csv` in Excel or a text editor to see exported results.

6. Testing & Output Verification

- Try entering both valid and invalid scores (to test validation).
- Confirm grades follow the correct thresholds.
- Ensure the CSV file updates correctly after export.
- Check for correct average, highest, and lowest results.

7. Submission & Grading Rubric

Deliverables

```
Firstname_Lastname_GradeAnalyzer/


├─ grade_analyzer.py
└─ students.csv
```

Rubric

- **Functionality (50%) –** Correctly calculates and exports results.
- **Code readability (25%) –** Proper indentation and variable names.
- **User interaction (15%) –** Clean prompts and feedback.
- **Innovation (10%) –** Optional bonus for enhancements (e.g. JSON, sorting, graphs).

8. Extensions & Challenges

- Add grade distribution visualization using `matplotlib`.
- Implement ranking (1st, 2nd, 3rd) by score.
- Include grade boundaries editable by user.
- Load data from and append to existing CSV files.

> **Instructor Tip:** Encourage students to re-run the app with new data and compare CSV results.

Instructor Checklist

Python installed ✅

Data validation tested ✅

CSV export verified ✅

Output accuracy reviewed

Files to Add

- `python-guides/student-grade-analyzer-guide.pdf`
- `python-samples/grade_analyzer.py`
- `site/projects/python-project-3.html`

> **Quick Student Walkthrough**
>
> 1. Run program in IDLE.
> 2. Enter student data.
> 3. Type "done" when finished.
> 4. Check `students.csv` output.