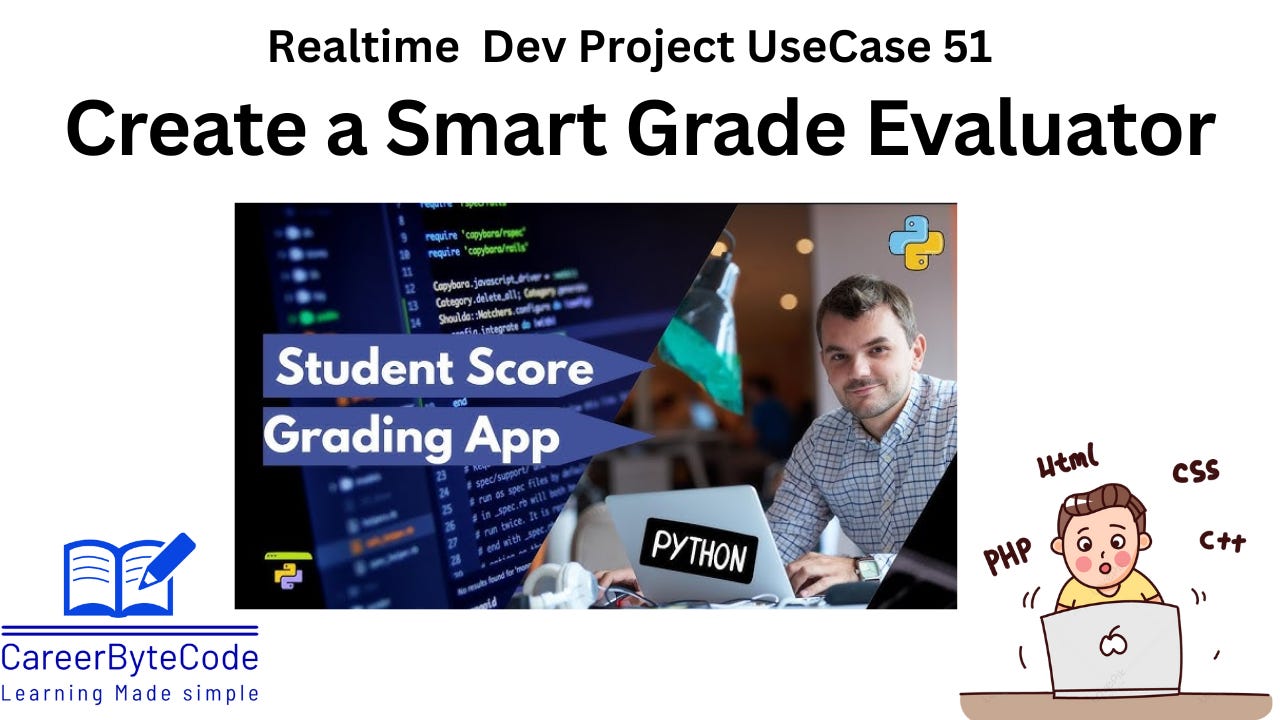
**Learn Python Logic by Creating a Student Marks Evaluator**

In academic systems, it's essential to evaluate a student’s performance based on their subject-wise marks.

[[](https://substackcdn.com/image/fetch/$s_!nc-J!,f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2Fe328392b-ba6b-48c6-bf7d-210761ae789e_1280x720.png)](https://substackcdn.com/image/fetch/$s_!nc-J!,f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2Fe328392b-ba6b-48c6-bf7d-210761ae789e_1280x720.png" \t "_blank)

**1. Detailed Problem Statement**

In academic systems, it's essential to evaluate a student’s performance based on their subject-wise marks. A basic way to do this is by calculating the average of marks from a set number of subjects and checking whether the student has passed or failed based on a threshold. In this case, if the average is **greater than or equal to 50**, the student passes; otherwise, they fail.

This simple logic forms the base of many real-world applications, such as school report card systems, student dashboards, and academic analytics. Implementing this in Python introduces new learners to essential programming constructs.

**2. Why We Need This Use Case**

* **Educational Relevance:** Every student and institution deals with grading. Automating grade calculation ensures consistency, transparency, and quick feedback.
* **Beginner-Level Logic Building:** This use case is perfect for programming beginners. It introduces variables, data structures (lists), loops, conditional logic, and input/output—all essential for foundational coding.
* **Scalability:** The logic can be extended to multiple students, a GUI-based app, or even a full-fledged grading system.
* **Accuracy and Speed:** Manual calculation is prone to human error. This code eliminates such issues.
* **Training Ground for Data-Driven Thinking:** It simulates how software uses data to make decisions.

**3. When We Need This Use Case**

* During **beginner programming lessons** to teach input, list, and conditional logic.
* In **online examination systems** to automate pass/fail status.
* While creating **school grading software** or academic report generators.
* As a **practice project for students** preparing for technical interviews or hackathons.
* In **tutorial videos or coding workshops** to explain real-life logic building with Python.

**4. Challenge Questions**

1. **Scenario:** You have marks of a student in 5 subjects. Write a program to store them and check if they passed.

*Expected: Accept inputs, calculate average, compare with 50.*

1. **Scenario:** The list of marks is [40, 60, 70, 45, 55]. What is the average? Will the student pass or fail?

*Hint: Calculate sum and average, then compare.*

1. **Scenario:** A student scored below 50 in 3 subjects but still passed overall. How is that possible?

*Hint: It's about the overall average, not per-subject minimum.*

1. **Scenario:** Modify the logic so that if the student scores less than 35 in any subject, they automatically fail.

*Hint: Use a loop to check individual marks.*

1. **Scenario:** You need to process data for 5 students, each with 5 marks. How would you structure the data?

*Hint: Use a list of lists or dictionary.*

**5. Prerequisites for the Lab**

* Basic Python knowledge
* Python 3.8 or above installed
* Basic understanding of Python
* Pycharm download and install - [Link](https://www.jetbrains.com/pycharm/download/download-thanks.html?platform=windows&code=PCC)
* Python download - [Link](https://www.python.org/ftp/python/3.13.3/python-3.13.3-amd64.exe)
* Familiarity with:
  + List data structures.
  + Loops (for loop)
  + Conditional statements (if-else)
  + Input/output functions
  + input() function
  + List operations

**6. Advantages and Disadvantages of This Use Case**

**✅ Advantages:**

* Very beginner-friendly and quick to implement
* Demonstrates real-world application of conditionals and loops
* No external libraries required
* Can be easily modified for advanced use cases

**❌ Disadvantages:**

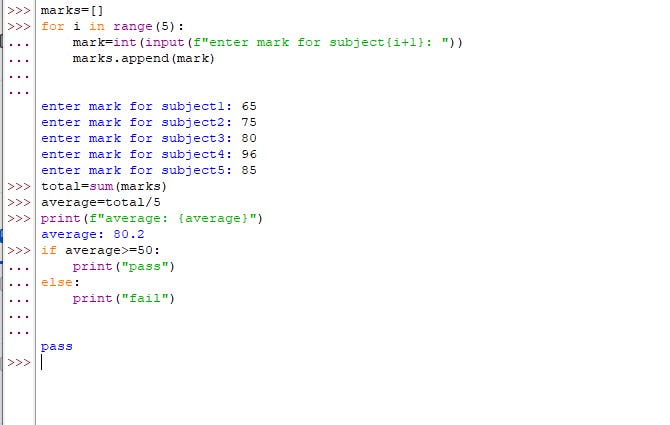
* Only suitable for small-scale or single-student data
* Doesn’t consider subject-wise passing criteria
* No error handling for non-integer input

**7. Step-by-Step Implementation Instructions**

Student Grade Calculation is a basic programming use case where a student's marks in five subjects are taken as input, the average is calculated, and a decision is made on whether the student has passed or failed. It introduces key concepts like input handling, loops, lists, conditional statements, and basic arithmetic. This use case is widely applicable in educational software and helps beginners learn how to write logical, real-world programs.

Accept 5 integer inputs from the user, representing marks.  
Store these values in a list.

Use a loop or built-in function to calculate the total of the marks.  
Calculate the average by dividing the total by 5.  
Use an if condition to check if the average is greater than or equal to 50.  
Print "Pass" if condition is met, else print "Fail".

[[](https://substackcdn.com/image/fetch/$s_!Litw!,f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2F77f454fb-3271-4ba5-92af-2cc4d6842985_670x425.png)](https://substackcdn.com/image/fetch/$s_!Litw!,f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2F77f454fb-3271-4ba5-92af-2cc4d6842985_670x425.png" \t "_blank)

**Approach:**

Take 5 integers as input (marks for each subject).  
Store them in a list.  
Use a loop to calculate the total.

Find the average.  
Use an if statement to check if the average >= 50.  
  
**Python Modules Used:**

No external modules. Only core Python.

**8. Conclusion**

The **Student Grade Calculation** program is an ideal beginner-level use case that connects core Python concepts with a familiar, real-world scenario. It demonstrates how to collect input, store data in lists, calculate averages using loops or functions, and make decisions using conditionals. Despite its simplicity, the logic can be scaled and modified for more complex educational software.

It helps learners develop confidence in logical thinking and Python coding, while also introducing them to the importance of clean input handling and real-time decision-making based on data.