Neema Peter
77366

User guide for the student record system program.

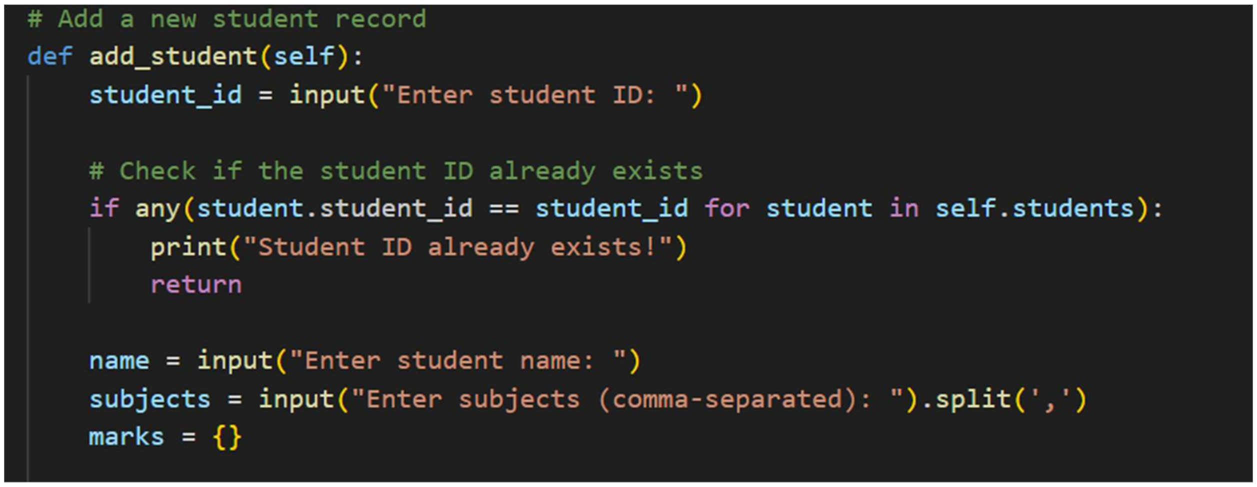
This student record system is a command line python application that allows users to manage student records efficiently. It allows the user to add, view, search and identify top-performing students, with data stored in a CSV file.

Before running the program make sure you have Python 3.x installed on your system, and in this case we will need an external library which is tabulate. Then you can run the program

Menu options

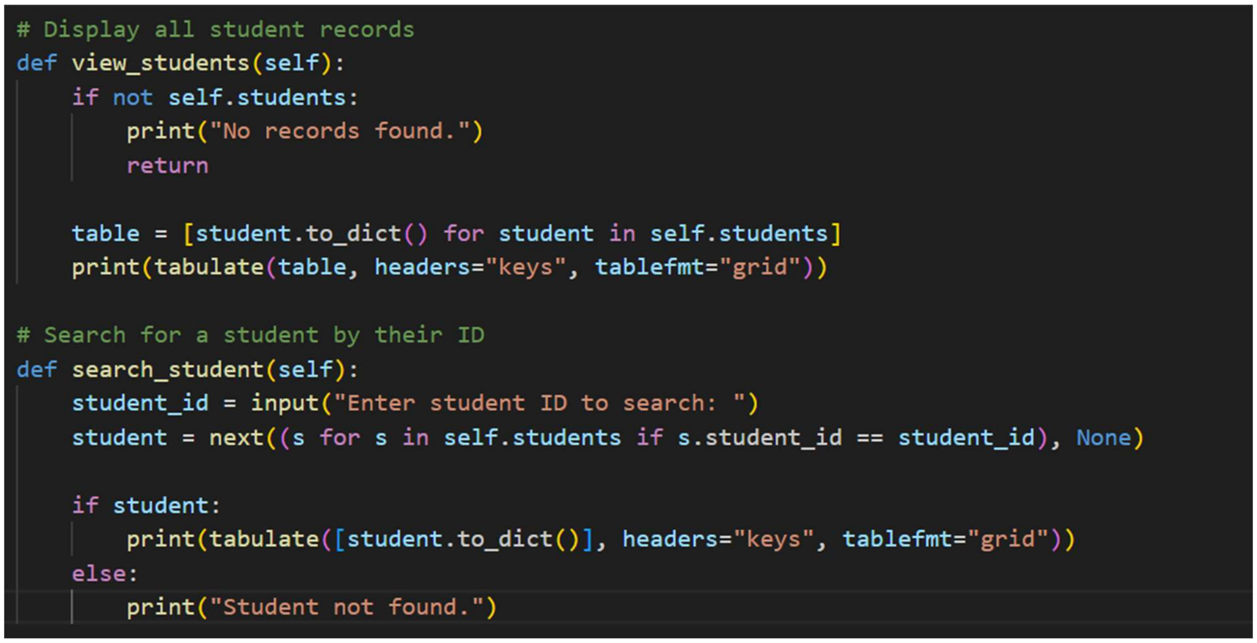
The menu options displayed will be

1. **Add student** which allows you to enter a student tID, name, subject, and marks



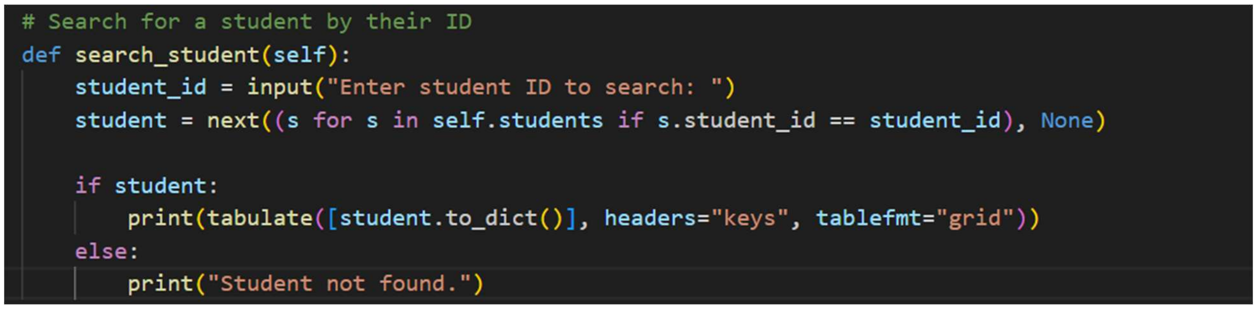
```python
# Add a new student record
def add_student(self):
    student_id = input("Enter student ID: ")

    # Check if the student ID already exists
    if any(student.student_id == student_id for student in self.students):
        print("Student ID already exists!")
        return

    name = input("Enter student name: ")
    subjects = input("Enter subjects (comma-separated): ").split(',')
    marks = {}
```

2. **View All students,** displays all stored student records in a table.



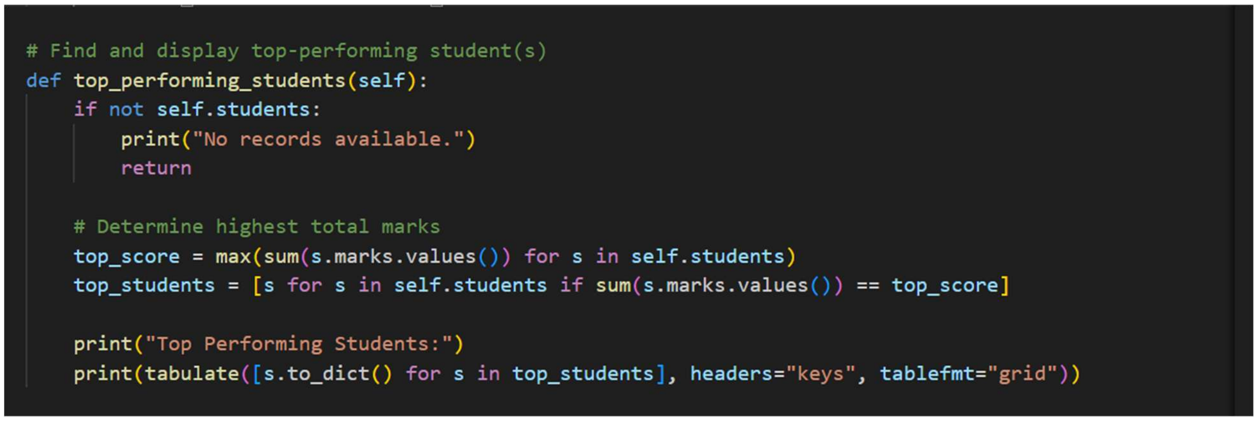
```python
# Display all student records
def view_students(self):
    if not self.students:
        print("No records found.")
        return

    table = [student.to_dict() for student in self.students]
    print(tabulate(table, headers="keys", tablefmt="grid"))

# Search for a student by their ID
def search_student(self):
    student_id = input("Enter student ID to search: ")
    student = next((s for s in self.students if s.student_id == student_id), None)

    if student:
        print(tabulate([student.to_dict()], headers="keys", tablefmt="grid"))
    else:
        print("Student not found.")
```

3. **Search Students by ID** finds and displays details of a student using their unique ID.



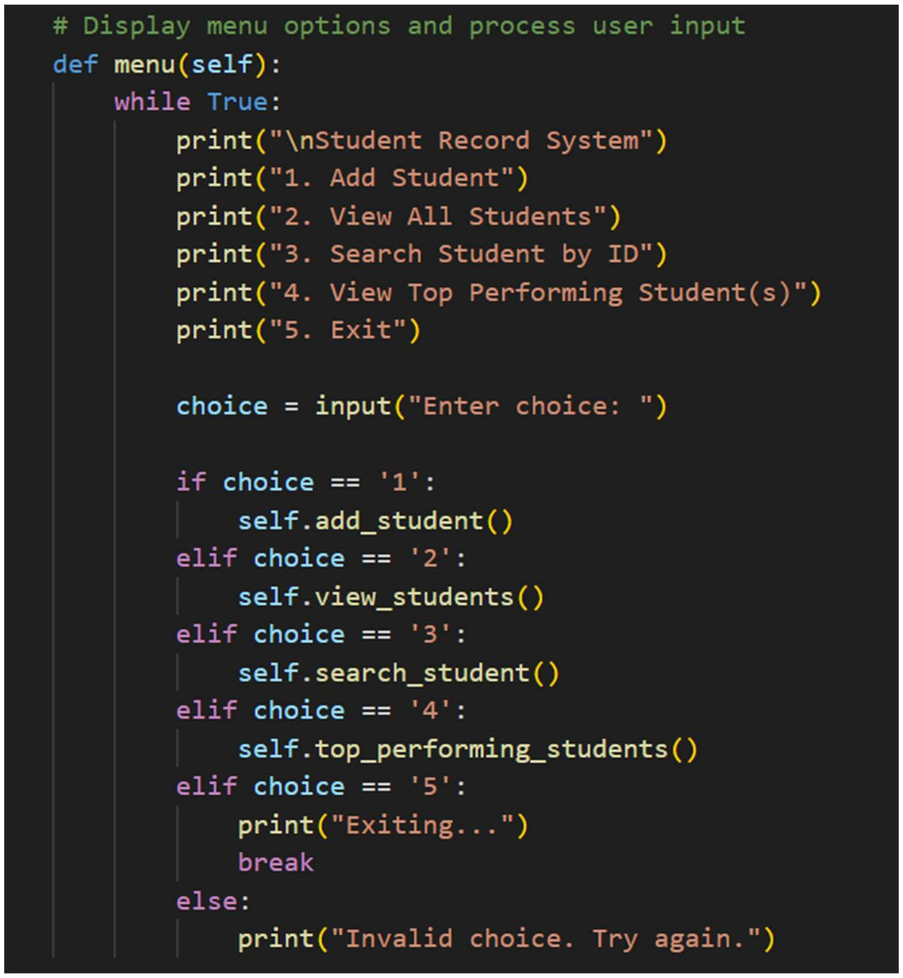
```python
# Search for a student by their ID
def search_student(self):
    student_id = input("Enter student ID to search: ")
    student = next((s for s in self.students if s.student_id == student_id), None)

    if student:
        print(tabulate([student.to_dict()], headers="keys", tablefmt="grid"))
    else:
        print("Student not found.")
```

4. **View Top Performing Student** identifies and displays students with the highest total marks.



```python
# Find and display top-performing student(s)
def top_performing_students(self):
    if not self.students:
        print("No records available.")
        return

    # Determine highest total marks
    top_score = max(sum(s.marks.values()) for s in self.students)
    top_students = [s for s in self.students if sum(s.marks.values()) == top_score]

    print("Top Performing Students:")
    print(tabulate([s.to_dict() for s in top_students], headers="keys", tablefmt="grid"))
```

5. **Exit** Closes the program. The program keeps running and asks you to choose an option until you choose the option exit due to the while loop being used and displays “invalid choice” when a choice that isn’t in the menu is chose.

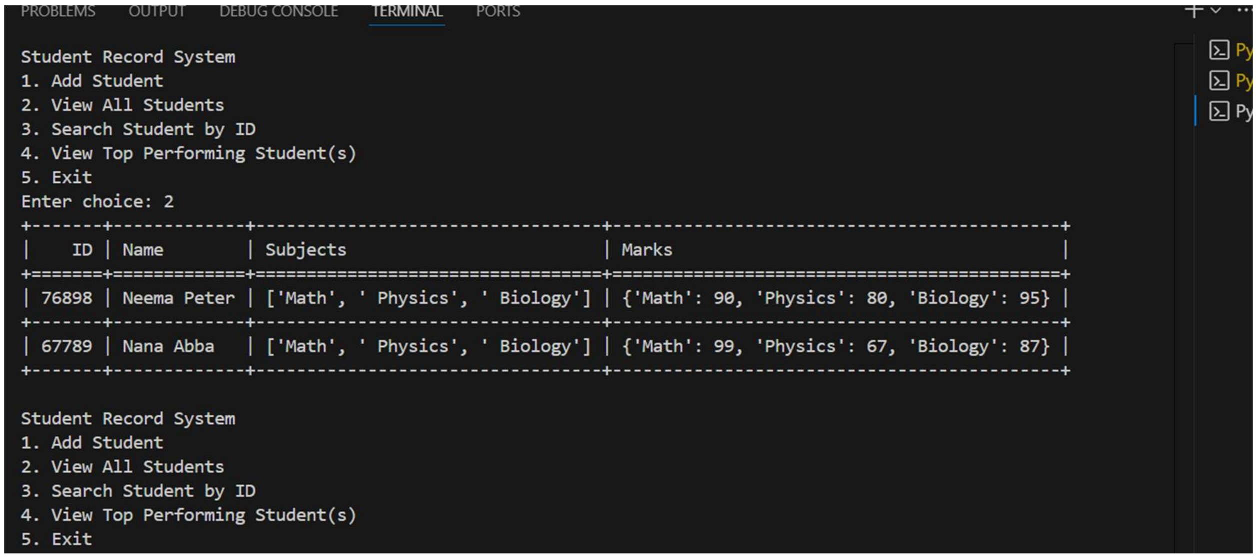


```python
# Display menu options and process user input
def menu(self):
    while True:
        print("\nStudent Record System")
        print("1. Add Student")
        print("2. View All Students")
        print("3. Search Student by ID")
        print("4. View Top Performing Student(s)")
        print("5. Exit")

        choice = input("Enter choice: ")

        if choice == '1':
            self.add_student()
        elif choice == '2':
            self.view_students()
        elif choice == '3':
            self.search_student()
        elif choice == '4':
            self.top_performing_students()
        elif choice == '5':
            print("Exiting...")
            break
        else:
            print("Invalid choice. Try again.")
```

Testing.

Below is a screenshot that describes the table formed using this program by entering two student's details and choosing the second option that helps us view all the records. The system stores these student records in a CSV file (students.csv) and this file is updated whenever a new student is added to the system.

Neema Peter
77366



```
Student Record System
1. Add Student
2. View All Students
3. Search Student by ID
4. View Top Performing Student(s)
5. Exit
Enter choice: 2
+-------+------------+----------------------------------+-----------------------------------------+
|    ID | Name       | Subjects                         | Marks                                   |
+=======+============+==================================+=========================================+
| 76898 | Neema Peter | ['Math', ' Physics', ' Biology'] | {'Math': 90, 'Physics': 80, 'Biology': 95} |
+-------+------------+----------------------------------+-----------------------------------------+
| 67789 | Nana Abba   | ['Math', ' Physics', ' Biology'] | {'Math': 99, 'Physics': 67, 'Biology': 87} |
+-------+------------+----------------------------------+-----------------------------------------+

Student Record System
1. Add Student
2. View All Students
3. Search Student by ID
4. View Top Performing Student(s)
5. Exit
```

Error handling.

This program also minimizes errors by ensuring unique user IDs are entered, it also validates marks to prevent negative values. And lastly it handles file-related errors such as missing files or read errors.

Additional Notes.

- To start fresh, delete student.csv file before running the program.
- If the program does not display correctly, make sure tabulate is installed.