Student Name: Aakash Thapliyal **UID:** 24MCC20027
Branch: MCA(ccd) **Section/Group:** A 1
Semester: 1st **Date of Performance:** 04-11-2024
Subject Name: : Design and Analysis of Algorithms Lab **Subject Code: :** 24CAP-612

1. Title of minor project. **Quicksort Algorithm for Sorting Students by ID**

2. Introduction:

Sorting is a fundamental operation in computer science that organizes data in a specific order. This project implements the Quicksort algorithm, a highly efficient sorting algorithm, to sort a list of students based on their IDs. Each student is represented as a dictionary containing their name, ID, and email address.

3. Task to be done:

- To implement the Quicksort algorithm for sorting a list of dictionaries.
- To collect student data through user input.
- To display the sorted list of students in a readable format.

4. Implementation

4.1 Quick sort Function

The quick-sort function sorts an array of student dictionaries based on their IDs. The function follows these steps:

1. Base Case: If the length of the array is less than or equal to 1, return the array as it is already sorted.
2. Pivot Selection: The last element of the array is chosen as the pivot.
3. Partitioning: The remaining elements are compared to the pivot:
   * Elements with IDs greater than the pivot are placed in the high list.
   * Elements with IDs less than or equal to the pivot are placed in the low list.
4. Recursive Calls: The function recursively sorts the low and high lists and concatenates them with the pivot.

4.2 Input Function

The input_students function prompts the user for student details:

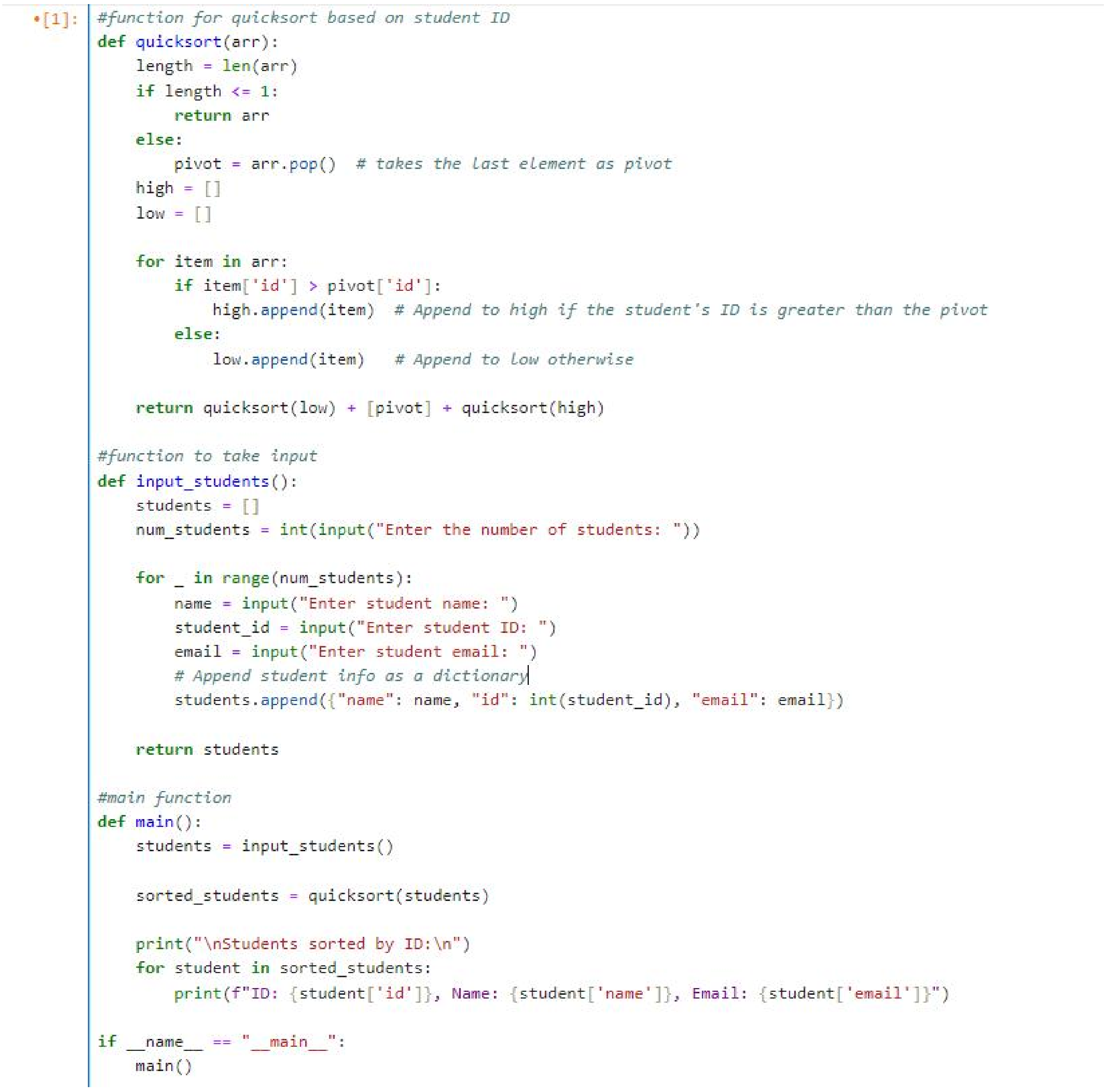
1. It first asks for the number of students to be entered.
2. For each student, it collects their name, ID, and email, storing this information in a list of dictionaries.

4.3 Main Function

The main function orchestrates the program's flow:

1. It calls input_students to gather student data.
2. It invokes the quicksort function to sort the students.
3. Finally, it prints the sorted list of students.

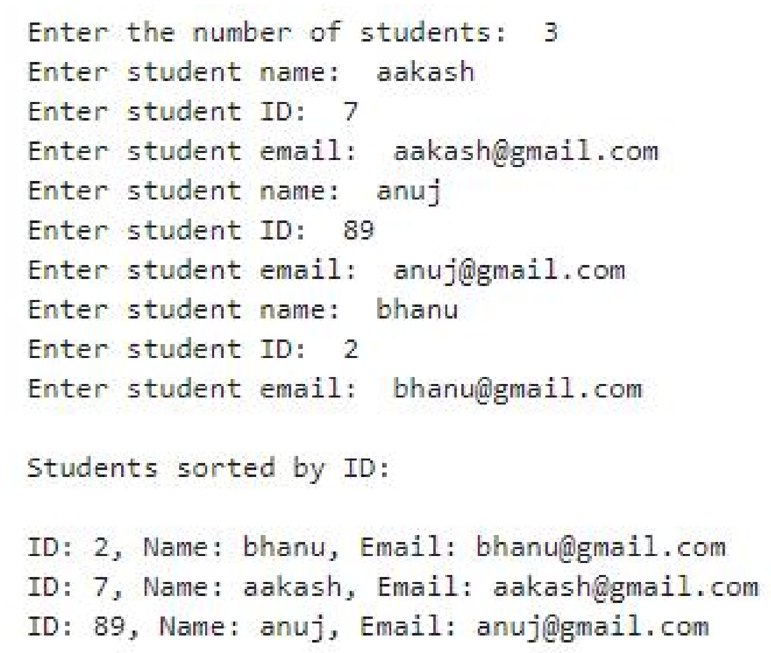
5. Code for experiment/project:



```
[1]:
#function for quicksort based on student ID
def quicksort(arr):
    length = len(arr)
    if length <= 1:
        return arr
    else:
        pivot = arr.pop()  # takes the last element as pivot
    high = []
    low = []

    for item in arr:
        if item['id'] > pivot['id']:
            high.append(item)  # Append to high if the student's ID is greater than the pivot
        else:
            low.append(item)   # Append to low otherwise

    return quicksort(low) + [pivot] + quicksort(high)

#function to take input
def input_students():
    students = []
    num_students = int(input("Enter the number of students: "))

    for _ in range(num_students):
        name = input("Enter student name: ")
        student_id = input("Enter student ID: ")
        email = input("Enter student email: ")
        # Append student info as a dictionary
        students.append({"name": name, "id": int(student_id), "email": email})

    return students

#main function
def main():
    students = input_students()

    sorted_students = quicksort(students)

    print("\nStudents sorted by ID:\n")
    for student in sorted_students:
        print(f"ID: {student['id']}, Name: {student['name']}, Email: {student['email']}")

if __name__ == "__main__":
    main()
```

6. Result/Output/Writing Summary:

Upon executing the program, users can enter the details of multiple students. After inputting the data, the program sorts the students by their IDs and displays the sorted list in a formatted output.



```
Enter the number of students:  3
Enter student name:  aakash
Enter student ID:  7
Enter student email:  aakash@gmail.com
Enter student name:  anuj
Enter student ID:  89
Enter student email:  anuj@gmail.com
Enter student name:  bhanu
Enter student ID:  2
Enter student email:  bhanu@gmail.com

Students sorted by ID:

ID: 2, Name: bhanu, Email: bhanu@gmail.com
ID: 7, Name: aakash, Email: aakash@gmail.com
ID: 89, Name: anuj, Email: anuj@gmail.com
```

7. Future Scope:

The project can be enhanced through:

- **User Interface**: Develop a GUI or web application for improved accessibility.
- **Advanced Sorting**: Implement multi-criteria sorting and additional sorting algorithms.
- **Data Handling**: Add input validation and integrate a database for persistent storage.
- **Performance Optimization**: Test with larger datasets and improve visualization of the sorting process.

8. Conclusion:

This project successfully implements the Quicksort algorithm to sort student data based on IDs. The use of a recursive approach for sorting demonstrates an efficient means of handling data organization. Future enhancements could include error handling for user input and extending functionality to sort by additional criteria such as names or emails.

9. Learning outcomes (What I have learnt):

Through the implementation of the Quicksort algorithm for sorting student records, I have gained the following insights:

Understanding of Sorting Algorithms: Developed a deeper understanding of the Quicksort algorithm, its mechanics, and its efficiency compared to other sorting methods.

Programming Skills: Enhanced my programming skills in Python, particularly in handling data structures like lists and dictionaries, and applying recursion.

User Input Handling: Learned how to effectively gather and process user input, including implementing input validation techniques to improve robustness.

Algorithm Implementation: Gained practical experience in implementing algorithms from scratch, reinforcing theoretical knowledge with practical application.

Problem-Solving: Developed critical thinking and problem-solving skills by addressing challenges encountered during implementation, such as partitioning data and managing recursion.

Future Development Awareness: Recognized the potential for future improvements and extensions of the project, fostering an understanding of iterative development.