Description

Once we are sure that the input information is correct, let's add a few new features to the program. This time, we will add users to the data store and update their records as the new learning data becomes available.

There will be some restrictions on adding students, though. First, you should assign a unique ID to each student so that you can access their records using the id rather than using their sensitive personal data. You may use a number or a string as an ID. Second, make sure that each user has only one account on the platform. Some users may have the same names, that's why you have their email addresses as their identity. This means that any email address can be used only once for registration.

The learning platform offers four courses: Python, Data Structures and Algorithms (DSA), Databases, and Flask. Each student can take any (or all) of these courses, complete tasks, pass tests, and submit their homework to receive points. By completing any task of any course, a student earns credit points that will be added to the student's total course score. These points can be zero, but they can't be negative.

Use the following input format to update records: a line of five elements separated by blank spaces. The first element is a student's ID, and the rest four elements are points earned by the student in the courses. For example:

```
25841 4 10 5 0
28405 0 8 7 5
```

Further, your program should be able to output information about the learning progress of any registered students. Keeping these conditions and restrictions in mind, don't forget to use unit tests to make sure your program works as intended.

Objectives

In addition to the features of the previous stages, your program should:

1. Check if the provided email has been already used when adding information about students. If so, respond with the following message: `This email is already taken.`
2. Recognize the new `list` command to print the `Students:` a header followed by the student IDs. The students must be listed in the order they were added. Remember, each ID must be unique. If there are no students to list, print `No students found`.
3. Recognize the new `add points` and print the following message in response: `Enter an id and points or 'back' to return`. After that, the program must read learning progress data

in the following format: `studentId number number number number`. The numbers correspond to the courses (Python, DSA, Databases, Flask). `Number` is a non-negative integer number. If there is no student with the specified ID, the program should print `No student is found for id=%s`. where `%s` is the invalid ID. Also, if any of the numbers are missing, or there is an extra number or any of the numbers do not meet the requirements mentioned above, the program should print `Incorrect points format`. If the learning progress data is entered in the correct format, and the specified user exists, the program should update the student's record and print `Points updated`. Once `back` is entered, the program must stop reading learning progress data.
4. Recognize the `find` command and print the following message: `Enter an id or 'back' to return`. After that, if an ID is entered, the program should either print details of the student with the specified ID in this format: `id points: Python=%d; DSA=%d; Databases=%d; Flask=%d`where `%d` is the respective number of points earned by the student. If the ID cannot be found, print the error message: `No student is found for id=%s`. where `%s` is the invalid ID.

Examples

The greater-than symbol followed by a space (`> `) represents the user input. Note that it's not part of the input.

Example 1: *entering a non-unique email*

```
Learning Progress Tracker
> add students
Enter student credentials or 'back' to return
> John Doe johnd@yahoo.com
The student has been added.
> Jane Spark jspark@gmail.com
The student has been added.
> Jane Sprocket jspark@gmail.com
This email is already taken.
> back
Total 2 students have been added.
> exit
Bye!
```

Example 2: *entering points in an incorrect format*

```
Learning Progress Tracker
> add students
Enter student credentials or 'back' to return:
> John Doe jdoe@yahoo.com
The student has been added.
> Jane Spark janes@gmail.com
The student has been added.
```

```
> back
Total 2 students have been added.
> list
Students:
10000
10001
> add points
Enter an id and points or 'back' to return:
> 1000 10 10 5 8
No student is found for id=1000.
> 10001 10 10 5 8
Points updated.
> 10001 5 8 7 3
Points updated.
> 10000 7 7 7 7 7
Incorrect points format.
> 10000 -1 2 2 2
Incorrect points format.
> 10000 ? 1 1 1
Incorrect points format.
> back
> exit
Bye!
```

Example 3: *entering points and printing information about a student*

```
Learning Progress Tracker
> add students
Enter student credentials or 'back' to return:
> John Doe jdoe@yahoo.com
The student has been added.
> back
Total 1 students have been added.
> list
Students:
10000
> add points
Enter an id and points or 'back' to return:
> 10000 5 5 5 5
Points updated.
> 10000 7 8 9 10
Points updated.
> back
> find
Enter an id or 'back' to return:
> 10000
10000 points: Python=12; DSA=13; Databases=14; Flask=15
> 10001
No student is found for id=10001.
> back
```

```
> exit
Bye!
```