**Assignment No . :- 1A**

**Aim :**

Create a dictionary to store student names and their corresponding scores in a test. Write functions to add new students, update scores, delete students, and find the student with the highest score.

**Theory :**

In Python, a dictionary is a built-in data type that allows you to store and manage data in key-value pairs. It is an unordered, mutable, and indexed collection where each key is unique. Dictionaries are versatile and widely used for various applications due to their efficient data retrieval capabilities.

**1.Key-Value Pairs:**

* **Key :** A key is a unique, immutable identifier (like a string, number, or tuple) used to access its associated value in a dictionary..
* **Value :** A value is the data associated with a key in a dictionary and can be of any data type, including other dictionaries.

**2.Unordered Collection:**

Dictionaries do not maintain a specific order of elements, so the order of key-value pairs stored or retrieved is not guaranteed.

**3.Mutable:**

Dictionaries are mutable, meaning you can modify them after creation by adding, updating, or removing key-value pairs.

**4.Indexed by Keys:**

Elements are accessed via unique keys in dictionaries, enabling fast lookups and modifications.

**Algorithm :**

**Initialize**

* Create an empty dictionary: student\_scores = {}

**Add a New Student**

* Define add\_student(name, score) function
* If name is in student\_scores:
  + Print: "Student {name} already exists. Use update\_score to update their score."
* Else:
  + Add name and score to student\_scores
  + Print: "Added student {name} with score {score}."

**Update a Student’s Score**

* Define update\_score(name, score) function
* If name is in student\_scores:
  + Update name's score in student\_scores
  + Print: "Updated {name}'s score to {score}."
* Else:
  + Print: "Student {name} does not exist. Use add\_student to add them first."

**Delete a Student**

* Define delete\_student(name) function
* If name is in student\_scores:
  + Remove name from student\_scores
  + Print: "Deleted student {name}."
* Else:
  + Print: "Student {name} does not exist."

**Find the Student with the Highest Score**

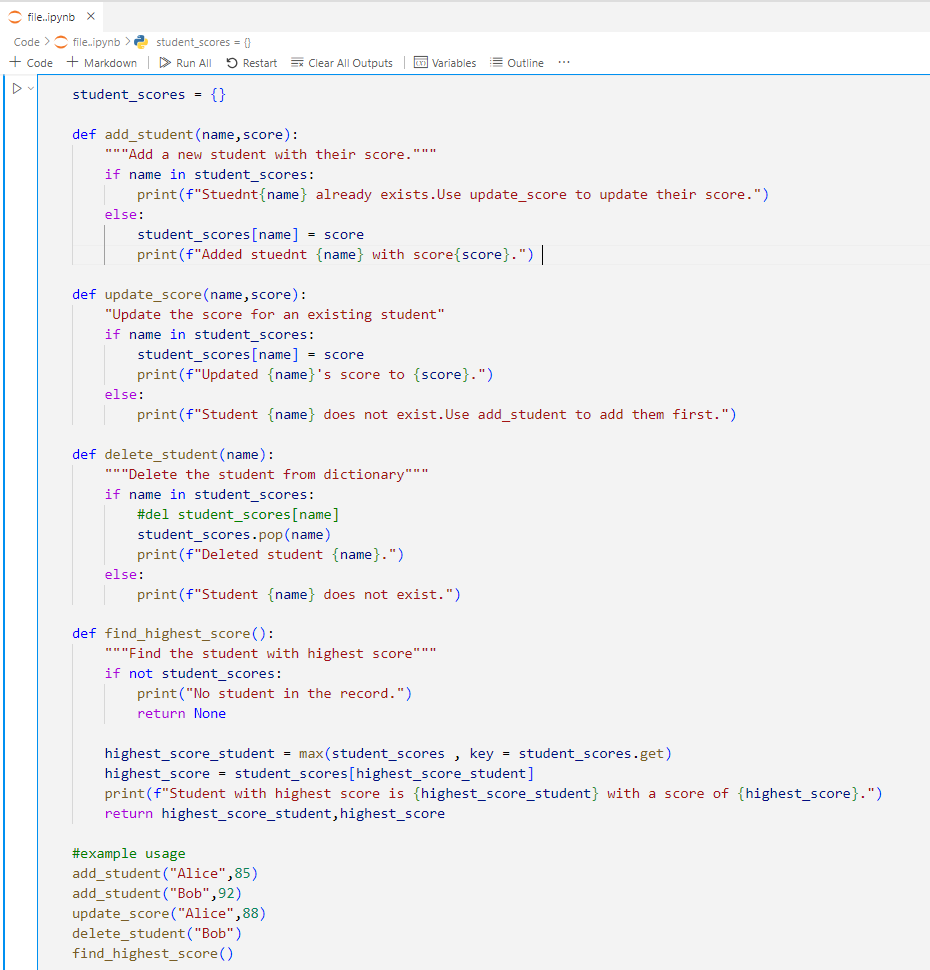
* Define find\_max() function
* If student\_scores is empty:
  + Print: "No records found."
  + Return None
* Else:
  + Find the student with the highest score
  + Print: "Student with the highest score is {highest\_score\_student} with a score of {highest\_score}."
  + Return (highest\_score\_student, highest\_score)

**Datasets :**

The code uses a single dataset, a Python dictionary named student\_scores, to store student names as keys and their corresponding scores as values. This dataset is manipulated through functions to add, update, delete, and query student scores.

**Conclusion:**

Implementing these functions allows efficient management of student scores using a dictionary. The add\_student function adds new students, update\_score modifies existing scores, delete\_student removes students, and find\_max identifies the student with the highest score. This approach leverages the dictionary's fast key-based operations for streamlined record management.

**Code:**

**Output :**

