

1. Grade Checker

Take a score as input and print the grade based on the following:

90+ : "A"

80-89 : "B"

70-79 : "C"

60-69 : "D"

Below 60 : "F"

here we used a basic if else statement to carry out marks and all.

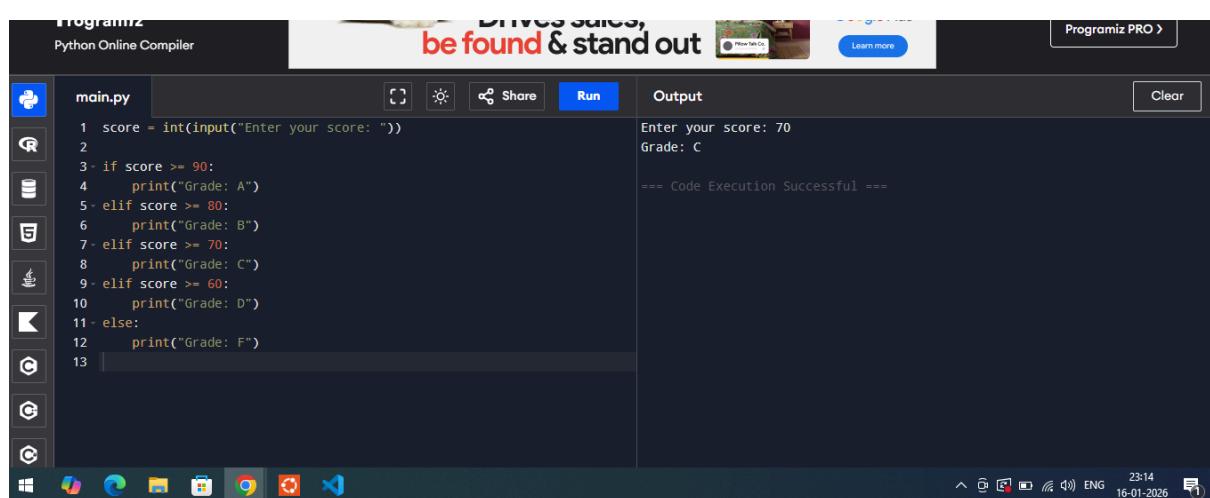
Ans :

User enters marks

Program checks conditions from top to bottom

First matching condition executes

Output grade is printed



The screenshot shows a Python online compiler interface. On the left, there is a code editor window titled "main.py" containing the following Python code:

```
1 score = int(input("Enter your score: "))
2
3 if score >= 90:
4     print("Grade: A")
5 elif score >= 80:
6     print("Grade: B")
7 elif score >= 70:
8     print("Grade: C")
9 elif score >= 60:
10    print("Grade: D")
11 else:
12     print("Grade: F")
```

On the right, there is an "Output" terminal window showing the results of running the code. It displays the user's input "Enter your score: 70" and the program's output "Grade: C". Below the output, it says "Code Execution Successful".

2 Student Grades

Create a dictionary where the keys are student names and the values are their grades. Allow the user to:

Add a new student and grade.

Update an existing student's grade.

Print all student grades.

Used dictionary and basic operations. Using if else:

Ans

```
main.py
1 students = {}
2 while True:
3     print("\n1. Add Student")
4     print("2. Update Grade")
5     print("3. View All Students")
6     print("4. Exit")
7     choice = input("Enter choice: ")
8
9     if choice == "1":
10         name = input("Enter student name: ")
11         grade = input("Enter grade: ")
12         students[name] = grade
13         print("Student added successfully.")
14
15     elif choice == "2":
16         name = input("Enter student name: ")
17         if name in students:
18             grade = input("Enter new grade: ")
19             students[name] = grade
20             print("Grade updated.")
21
22     elif choice == "3":
23         for name, grade in students.items():
24             print(name, ":", grade)
25
26     elif choice == "4":
27         break
28
29 else:
30     print("Invalid choice")
```

Output

```
1. Add Student
2. Update Grade
3. View All Students
4. Exit
Enter choice: 1
Enter student name: Anshu Tamarkar
Enter grade: 59
Student added successfully.

1. Add Student
2. Update Grade
3. View All Students
4. Exit
Enter choice: 3
Anshu Tamarkar : 59

1. Add Student
2. Update Grade
3. View All Students
4. Exit
Enter choice: 4
```

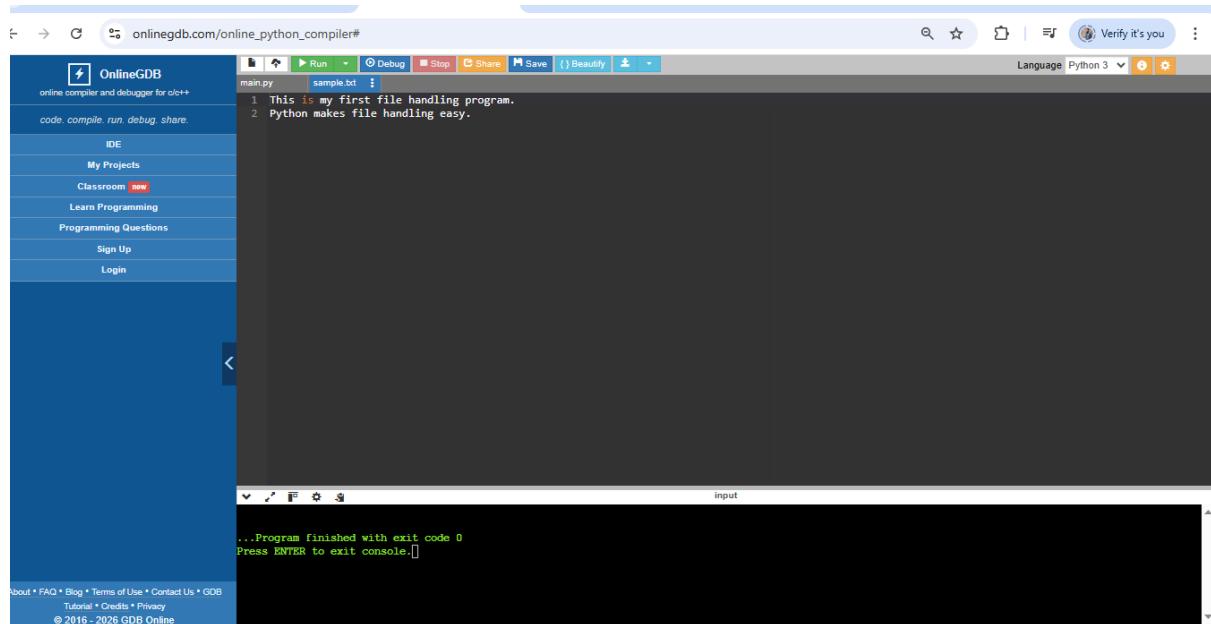
Dictionary stores student data Menu-driven program using while

If else handles operations in keyword checks existing students

3. Write to a File

Write a program to create a text file and write some content to it.

Using file functions like write and open.



The screenshot shows the OnlineGDB Python compiler interface. On the left, there's a sidebar with links for IDE, My Projects, Classroom, Learn Programming, Programming Questions, Sign Up, and Login. The main area has tabs for 'main.py' and 'sample.txt'. The 'main.py' tab contains the following Python code:

```
1 This is my first file handling program.
2 Python makes file handling easy.
```

Below the code editor is a terminal window labeled 'input' which displays the output of the program:

```
...Program finished with exit code 0
Press ENTER to exit console.[]
```

At the bottom of the page, there are footer links for About, FAQ, Blog, Terms of Use, Contact Us, and GDB. It also includes a copyright notice: © 2016 - 2026 GDB Online.

The program creates a text file named sample.txt and writes content into it using Python file handling.

The file was successfully created and the written content is visible.you can see in the right side of the [main.py](#) file.

4. Read from a File

We used open in read mode and file.read to read and print to display.

Ans :

The screenshot shows the OnlineGDB IDE interface. On the left sidebar, there are links for 'IDE', 'My Projects', 'Classroom' (which is highlighted in red), 'Learn Programming', 'Programming Questions', 'Sign Up', and 'Login'. The main workspace has two tabs: 'main.py' and 'sample.txt'. The 'main.py' tab contains the following Python code:

```
1 file = open("sample.txt", "r")
2 content = file.read()
3 print(content)
4 file.close()
```

The 'sample.txt' tab shows the contents of the file: "This is my first file handling program. Python makes file handling easy." Below the code editor is a terminal window with the output:

```
This is my first file handling program.
Python makes file handling easy.

...Program finished with exit code 0
Press ENTER to exit console.[]
```

The file sample.txt is opened in read mode ("r") file.read() reads the entire content of the file.

The content is printed on the screen using printf() file.close() closes the file after reading

Submission Guidelines :- Attach Screenshots or command along with explanation and submit in doc(google doc or microsoft doc) format or share github link