**Program source Code:**

**Lab\_06\_guru.py**

import os  
import sys  
  
class Student\_grades:  
 scores = {} # Class variable to store scores for all students  
  
 def \_\_init\_\_(self, name, grade):  
 self.nameofstudent = name # Initialize the student's name  
 self.gradeofstudent = grade # Initialize the student's grade  
 self.old\_gradeofstudent = grade # Initialize the student's old grade  
  
 def calculate\_scores(self): # Define the method to calculate scores  
 try:  
 answer\_key = [] # Get the answer key from the directory  
 with open("answers.txt", 'r') as answer\_file:  
 answer\_key = answer\_file.read().splitlines() # Read answer key from file in the directory  
  
 student\_answers = []  
 with open("data.txt", 'r') as data\_file:  
 student\_answers = [line.strip().split(',') for line in data\_file] # Read student answers from file for comparison  
  
 total\_score = 100 # Initialize total score  
  
 for answer in student\_answers:  
 if answer[0] == self.get\_name(): # Check if the answer belongs to the current student  
 incorrect\_count = sum([answer[i] != answer\_key[i - 1] for i in range(1, len(answer))]) # Count incorrect answers  
 deduction = min(incorrect\_count \* 10, 100 - self.gradeofstudent) # Calculate deduction based on incorrect answers  
 total\_score -= deduction # Deduct the calculated deduction from the total score  
  
 self.old\_gradeofstudent = self.gradeofstudent # Update the old grade to the current grade  
 self.gradeofstudent = total\_score # Update the current grade with the calculated score  
 Student\_grades.scores[self.get\_name()] = self.gradeofstudent # Store the current grade in the class variable  
  
 except FileNotFoundError:  
 print("File not found. Please check if 'answers.txt' and 'data.txt' exist in the directory.")  
 sys.exit() # Exit the program if file not found  
 except Exception as e:  
 print("An error occurred:", str(e))  
 sys.exit() # Exit the program for other exceptions  
  
 def get\_name(self): # Return the student's name  
 return self.nameofstudent  
  
 def get\_old\_grade(self): # Return the student's old grade  
 return self.old\_gradeofstudent  
  
 def get\_letter\_grade(self): # Check the grade and return the corresponding letter grade  
 if self.gradeofstudent >= 90:  
 return "A"  
 elif self.gradeofstudent >= 80:  
 return "B"  
 elif self.gradeofstudent >= 70:  
 return "C"  
 elif self.gradeofstudent >= 60:  
 return "D"  
 else:  
 return "F" # If the grade is less than 60, mark as fail  
  
 @staticmethod  
 def sort\_scores():  
 return sorted(Student\_grades.scores.items(), key=lambda x: x[1])  
  
 @staticmethod  
 def calculate\_statistics(students):  
 all\_grades = [student.get\_old\_grade() for student in students] # Get the old grades of all students  
 overall\_average = sum(all\_grades) / len(all\_grades) # Calculate the overall average grade  
 overall\_grade\_range = max(all\_grades) - min(all\_grades) # Calculate the overall grade range  
 student\_average\_grades = [(student.get\_old\_grade() + student.gradeofstudent) / 2 for student in students]  
 student\_grade\_ranges = [abs(student.get\_old\_grade() - student.gradeofstudent) for student in students]  
  
 return overall\_average, overall\_grade\_range, student\_average\_grades, student\_grade\_ranges  
  
  
students = [  
 Student\_grades('Sammy Student', 65), # Student name and old marks  
 Student\_grades('Betty Sanchez', 45),  
 Student\_grades('Alice Brown', 100),  
 Student\_grades('Tom Schulz', 50),  
 Student\_grades('guruteja kanderi', 65),  
]  
  
students[4].gradeofstudent = 60 # Update Guruteja's new score to 60  
  
# Calculate scores and print individual results  
for student in students:  
 student.calculate\_scores()  
 print("Name:", student.get\_name()) # Print student's name  
 print("Old Score:", student.get\_old\_grade()) # Print student's old grade  
 print("New Score:", student.gradeofstudent) # Print student's new grade  
 print("Average:", (student.get\_old\_grade() + student.gradeofstudent) / 2) # Print average of old and new grades  
 print("Grade Range:", abs(student.get\_old\_grade() - student.gradeofstudent)) # Print grade range  
 print("Student Grade:", student.get\_letter\_grade()) # Print student's letter grade  
 print()  
  
# Calculate overall statistics and student statistics  
try:  
 overall\_average, overall\_grade\_range, student\_average\_grades, student\_grade\_ranges = Student\_grades.calculate\_statistics(  
 students)  
  
 # Print student statistics  
 print("......................") # To put the statistics in a better format.  
 print("Student Statistics:")  
 print("......................")  
 for i, student in enumerate(students):  
 print("Name:", student.get\_name())  
 print("Average Grade:", student\_average\_grades[i])  
 print("Grade Range:", student\_grade\_ranges[i]) # Print student's grade range  
 print()  
  
 # Print overall statistics  
 print("Overall Statistics:")  
 print("Overall Average Grade:", overall\_average) # Print overall average grade  
 print("Overall Grade Range:", overall\_grade\_range) # Print overall grade range  
  
except Exception as e: # Exception case  
 print("An error occurred while calculating statistics:", str(e))  
 sys.exit() # Exit the program if exception occurs

**OUTPUT :**

**data.txt:**

**A screenshot of a computer

Description automatically generated**

**OUTPUT:**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**Student Statistics:  
A screenshot of a computer

Description automatically generated**

**Exceptional errors if file not found in directory:**

**A screenshot of a computer program

Description automatically generated**

**(1)** What is the purpose of this function?

**def \_\_init\_\_()**

In this program we haven’t used the Main() function to call the methods. In that case we can use constructor to call or execute automatically.

It allows us to set the values of parameters with initial stage or you can set the values in the set with respect to program requirement.

You can assigns values to instance variables specific to the object being created. These instance variables hold the data that belongs to each individual object.

**(2)** Is **self** considered a keyword in Python?

No , It is not. There are 35 key words in python and self is not declared or considered as keyword.  
But it is recommended to use the best available attributes or parameters while assigning the names in the code. To not confuse with the code and to ensure the readability of the code.

It is a conventional name used as the first parameter in instance methods within a class. By convention, self is used to refer to the instance of the class itself. That’s what we have done in the code. But widely used as first parameter of the instance methods, to reference the instance class.

**(3)** What is meant by object - oriented programming?

The Concept is to use the objects which are instances of the classes which is utilized or reused in the code multiple times to make the code better and significant.

The OOPS concept is widely classified as Encapsulation, Polymorphism, Inheritance, and Abstraction. The Classes and Objects are also comes under this concept and plays a key role.

Object-oriented programming provides benefits such as modularity, reusability, code organization, and abstraction, making it easier to manage and maintain complex software systems.

**(4)** How are text files used in your program code for this project?

The text files are the source to compare the answers and check with the student data stored inside.

The answers.txt file used to store the answer key for the exam. It contains the correct answers to the quiz questions. The program reads the contents of this file to compare the student's responses with the correct answers and calculate the scores.

The data.txt has the records of students which they marked as their answers, now this answers given by students will be compared with the answers.txt key sheet and we will calculate the marks and then we will make the average with old score and new score available.

**(5)** What have you learned from performing and coding for this lab assignment?

Learned to call the functions without main function in the code with the help of constructors.

Along with that I have experienced to handle the input and output files. Which is answers.txt and data.txt sheet. Learnt opening the files in directory and read the data and compare it to do or perform some statistics.

And this lab deals with some grading and statistics, so gained some mathematical knowledge to grade students and get the statistics out of some data.