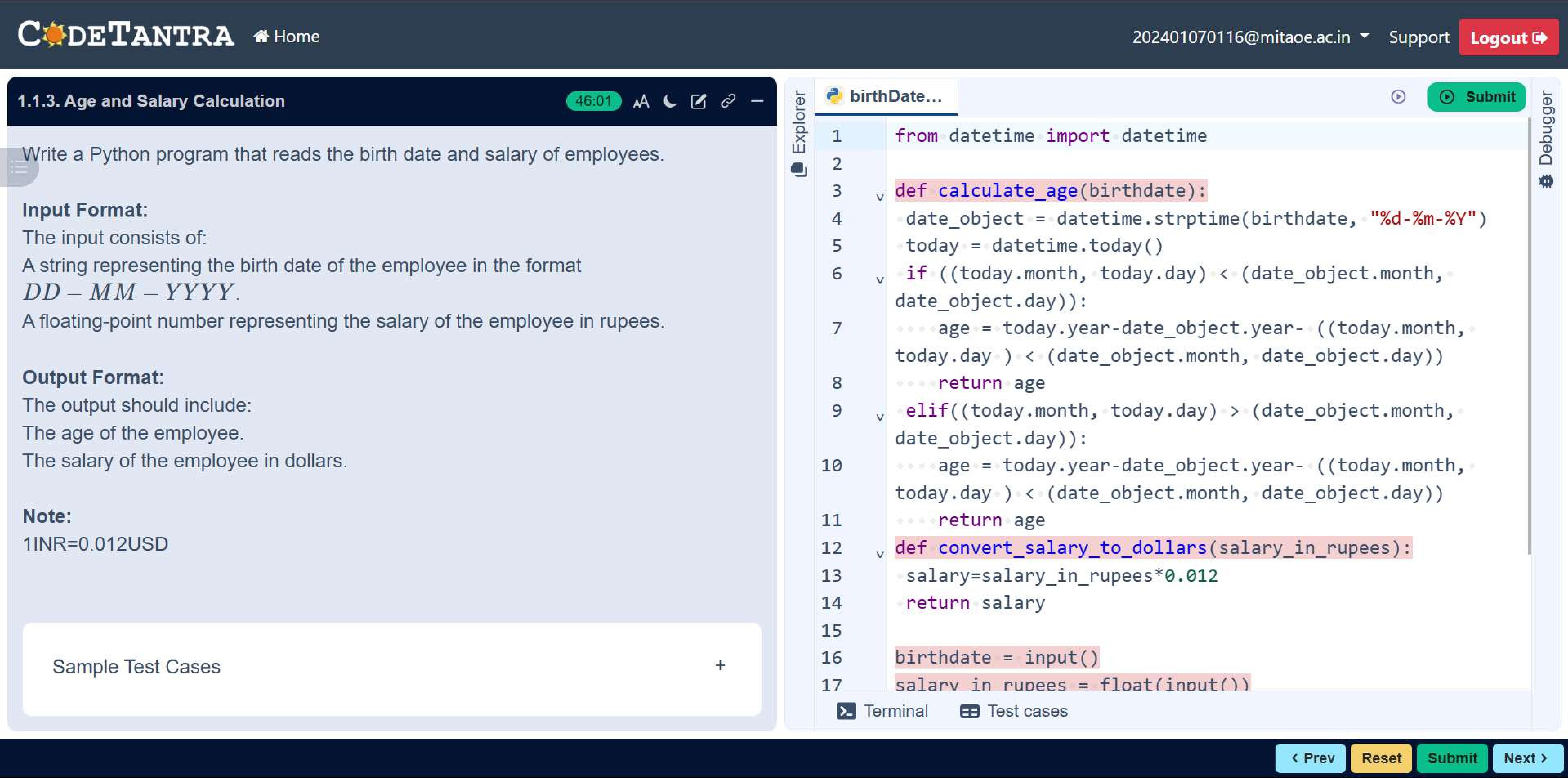


1.1.3. Age and Salary Calculation

Write a Python program that reads the birth date and salary of employees.

Input Format:

The input consists of:

A string representing the birth date of the employee in the format $DD-MM-YYYY$.

A floating-point number representing the salary of the employee in rupees.

Output Format:

The output should include:

The age of the employee.

The salary of the employee in dollars.

Note:

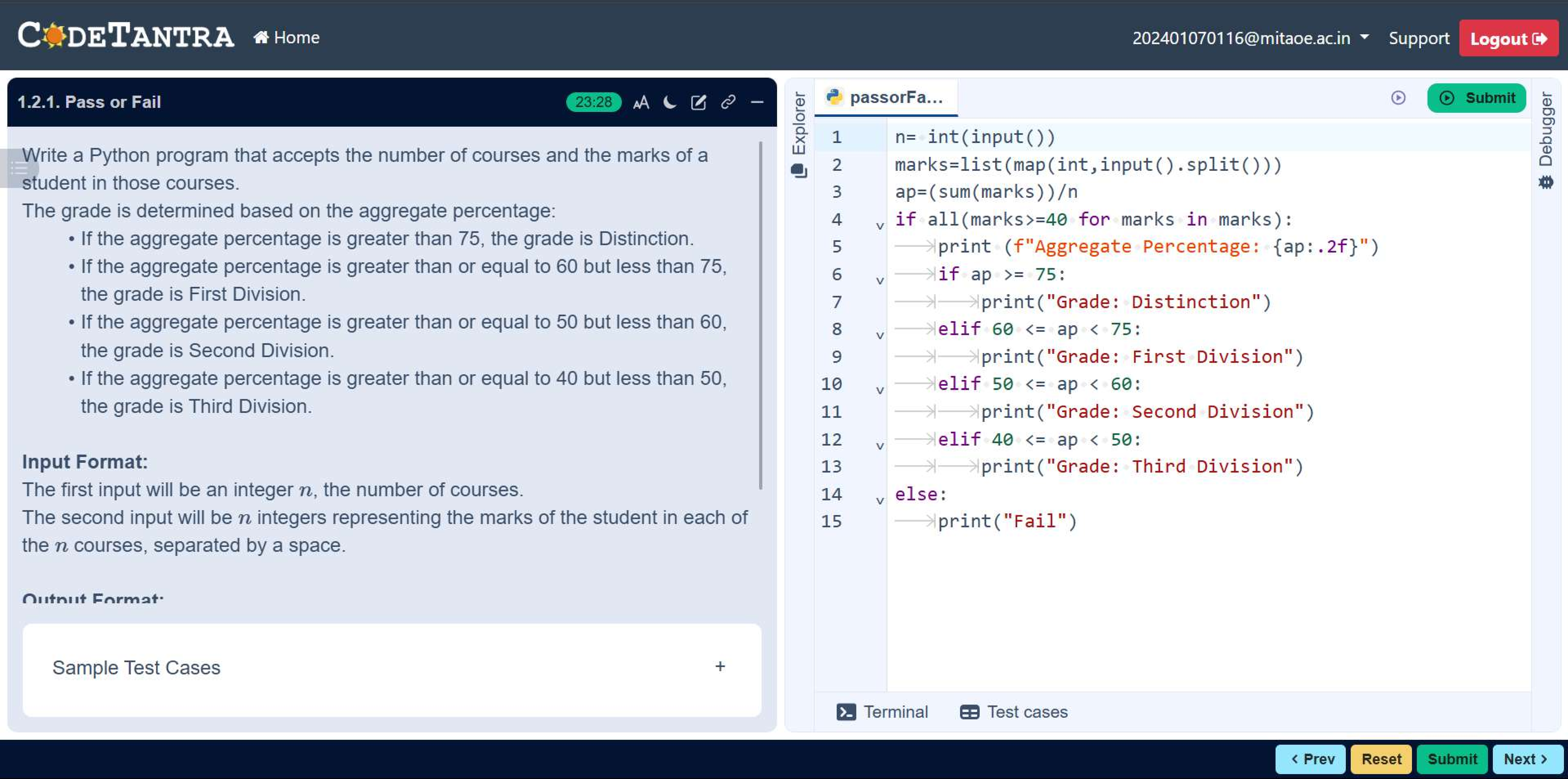
1INR=0.012USD

Sample Test Cases +

```python
from datetime import datetime

def calculate_age(birthdate):
 date_object = datetime.strptime(birthdate, "%d-%m-%Y")
 today = datetime.today()
 if ((today.month, today.day) < (date_object.month, date_object.day)):
    age = today.year-date_object.year- ((today.month, today.day ) < (date_object.month, date_object.day))
    return age
 elif((today.month, today.day) > (date_object.month, date_object.day)):
    age = today.year-date_object.year- ((today.month, today.day ) < (date_object.month, date_object.day))
    return age
def convert_salary_to_dollars(salary_in_rupees):
 salary=salary_in_rupees*0.012
 return salary

birthdate = input()
salary_in_rupees = float(input())
```



1.2.1. Pass or Fail

Write a Python program that accepts the number of courses and the marks of a student in those courses.

The grade is determined based on the aggregate percentage:

- If the aggregate percentage is greater than 75, the grade is Distinction.
- If the aggregate percentage is greater than or equal to 60 but less than 75, the grade is First Division.
- If the aggregate percentage is greater than or equal to 50 but less than 60, the grade is Second Division.
- If the aggregate percentage is greater than or equal to 40 but less than 50, the grade is Third Division.

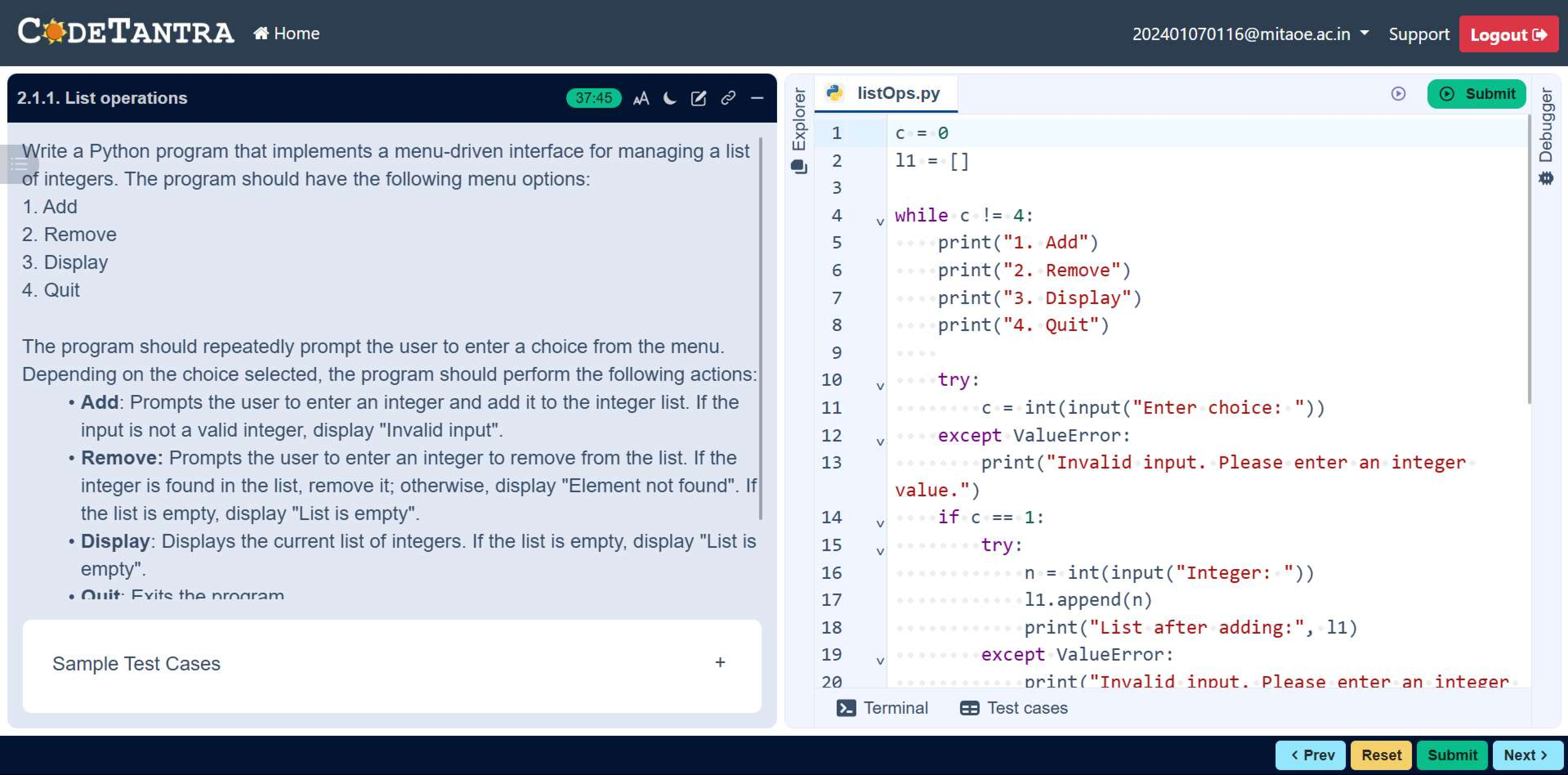
Input Format:

The first input will be an integer n, the number of courses.

The second input will be n integers representing the marks of the student in each of the n courses, separated by a space.

Output Format:

```python
n= int(input())
marks=list(map(int,input().split()))
ap=(sum(marks))/n
if all(marks>=40 for marks in marks):
    print (f"Aggregate Percentage: {ap:.2f}")
    if ap >= 75:
        print("Grade: Distinction")
    elif 60 <= ap < 75:
        print("Grade: First Division")
    elif 50 <= ap < 60:
        print("Grade: Second Division")
    elif 40 <= ap < 50:
        print("Grade: Third Division")
else:
    print("Fail")
```



2.1.1. List operations

Write a Python program that implements a menu-driven interface for managing a list of integers. The program should have the following menu options:

1. Add
2. Remove
3. Display
4. Quit

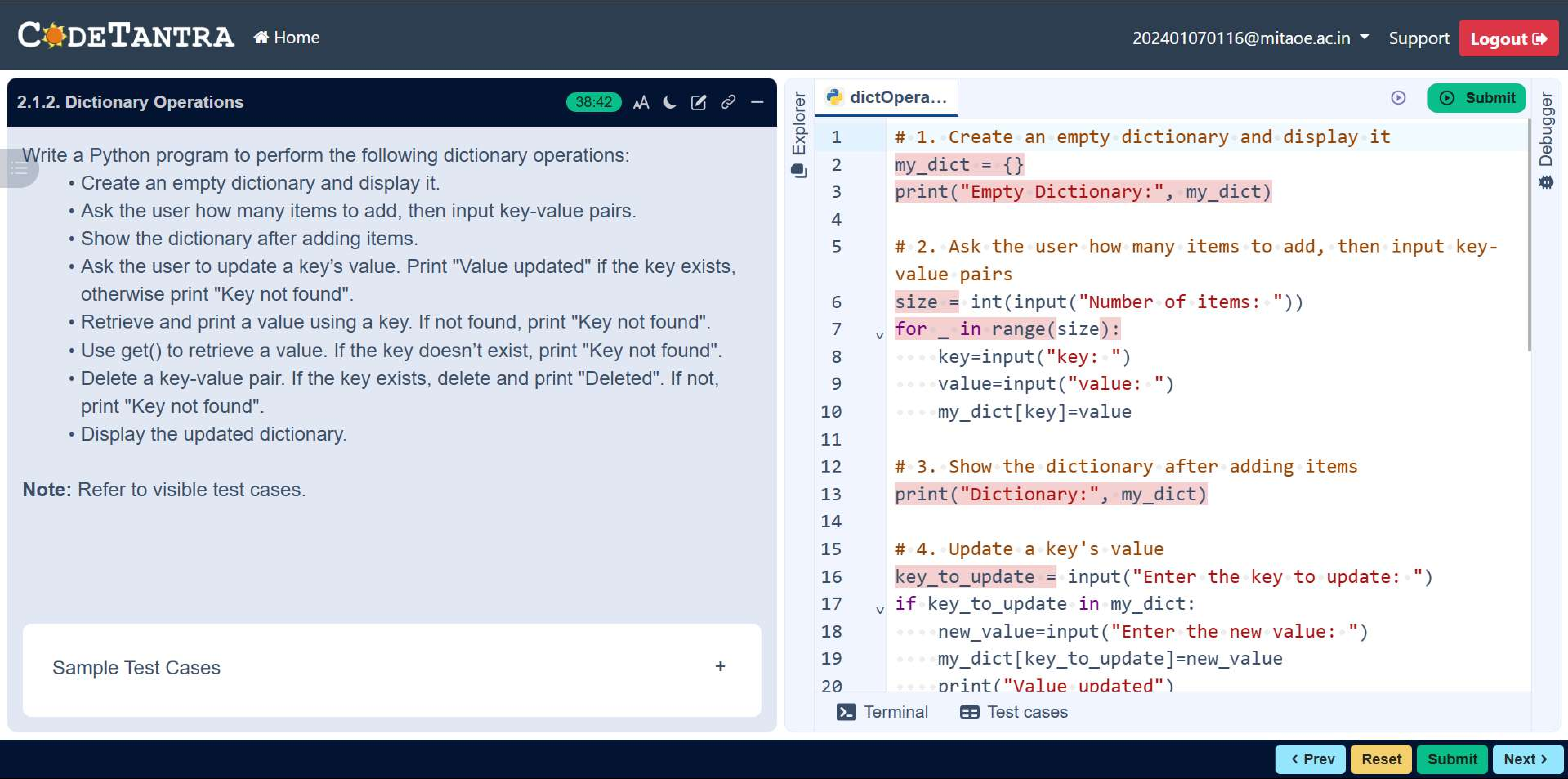
The program should repeatedly prompt the user to enter a choice from the menu. Depending on the choice selected, the program should perform the following actions:

- **Add**: Prompts the user to enter an integer and add it to the integer list. If the input is not a valid integer, display "Invalid input".
- **Remove:** Prompts the user to enter an integer to remove from the list. If the integer is found in the list, remove it; otherwise, display "Element not found". If the list is empty, display "List is empty".
- **Display**: Displays the current list of integers. If the list is empty, display "List is empty".
- **Quit**: Exits the program

Sample Test Cases +

```python
c = 0
l1 = []

while c != 4:
    print("1. Add")
    print("2. Remove")
    print("3. Display")
    print("4. Quit")
    
    try:
        c = int(input("Enter choice: "))
    except ValueError:
        print("Invalid input. Please enter an integer value.")
    if c == 1:
        try:
            n = int(input("Integer: "))
            l1.append(n)
            print("List after adding:", l1)
        except ValueError:
            print("Invalid input. Please enter an integer
```



2.1.2. Dictionary Operations

Write a Python program to perform the following dictionary operations:

- Create an empty dictionary and display it.
- Ask the user how many items to add, then input key-value pairs.
- Show the dictionary after adding items.
- Ask the user to update a key's value. Print "Value updated" if the key exists, otherwise print "Key not found".
- Retrieve and print a value using a key. If not found, print "Key not found".
- Use get() to retrieve a value. If the key doesn't exist, print "Key not found".
- Delete a key-value pair. If the key exists, delete and print "Deleted". If not, print "Key not found".
- Display the updated dictionary.

Note: Refer to visible test cases.

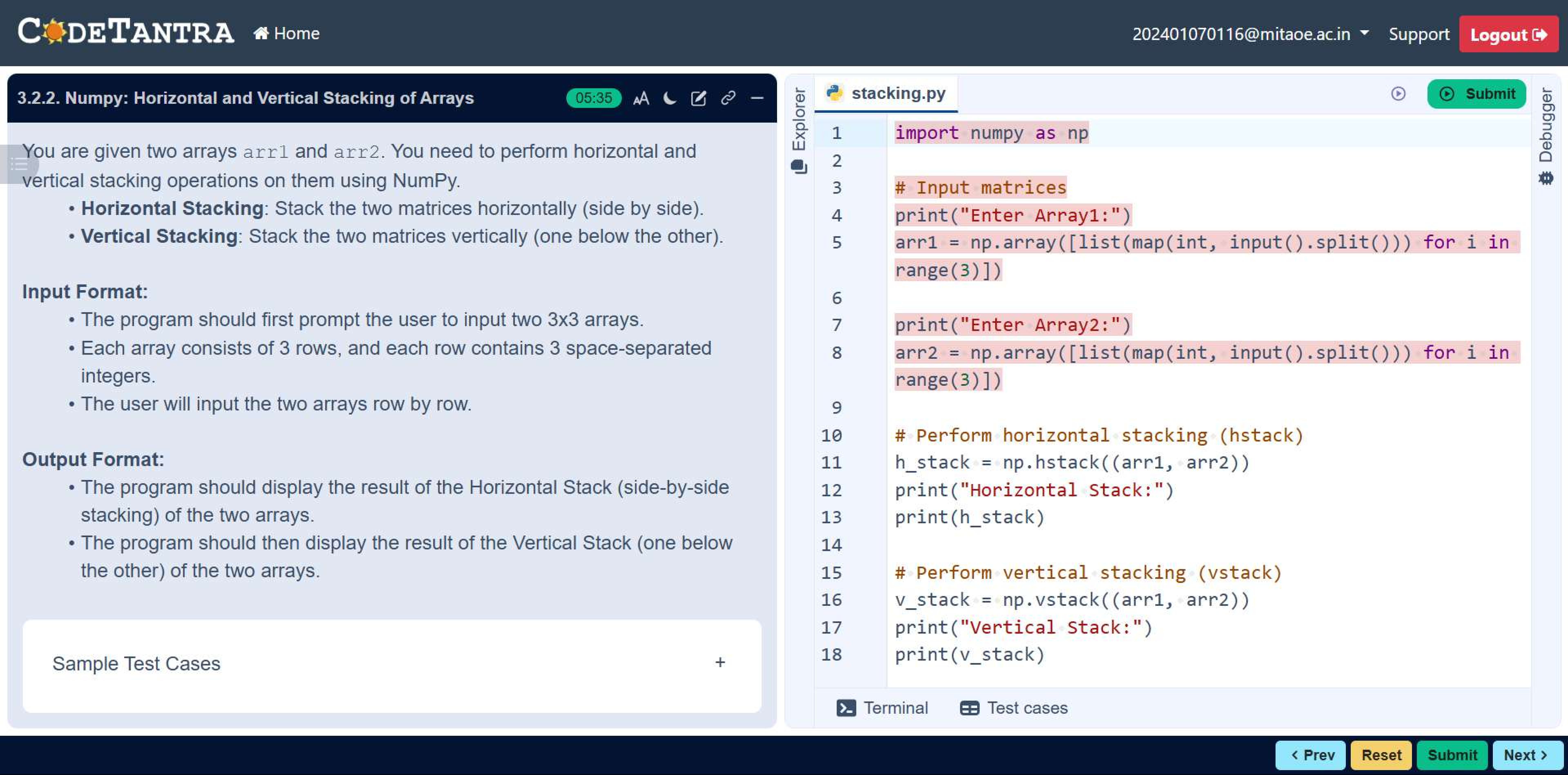
Sample Test Cases +

```python
# 1. Create an empty dictionary and display it
my_dict = {}
print("Empty Dictionary:", my_dict)

# 2. Ask the user how many items to add, then input key-value pairs
size = int(input("Number of items: "))
for _ in range(size):
    key=input("key: ")
    value=input("value: ")
    my_dict[key]=value

# 3. Show the dictionary after adding items
print("Dictionary:", my_dict)

# 4. Update a key's value
key_to_update = input("Enter the key to update: ")
if key_to_update in my_dict:
    new_value=input("Enter the new value: ")
    my_dict[key_to_update]=new_value
    print("Value updated")
```



3.2.2. Numpy: Horizontal and Vertical Stacking of Arrays

You are given two arrays `arr1` and `arr2`. You need to perform horizontal and vertical stacking operations on them using NumPy.

- **Horizontal Stacking**: Stack the two matrices horizontally (side by side).
- **Vertical Stacking**: Stack the two matrices vertically (one below the other).

Input Format:

- The program should first prompt the user to input two 3x3 arrays.
- Each array consists of 3 rows, and each row contains 3 space-separated integers.
- The user will input the two arrays row by row.

Output Format:

- The program should display the result of the Horizontal Stack (side-by-side stacking) of the two arrays.
- The program should then display the result of the Vertical Stack (one below the other) of the two arrays.

Sample Test Cases +

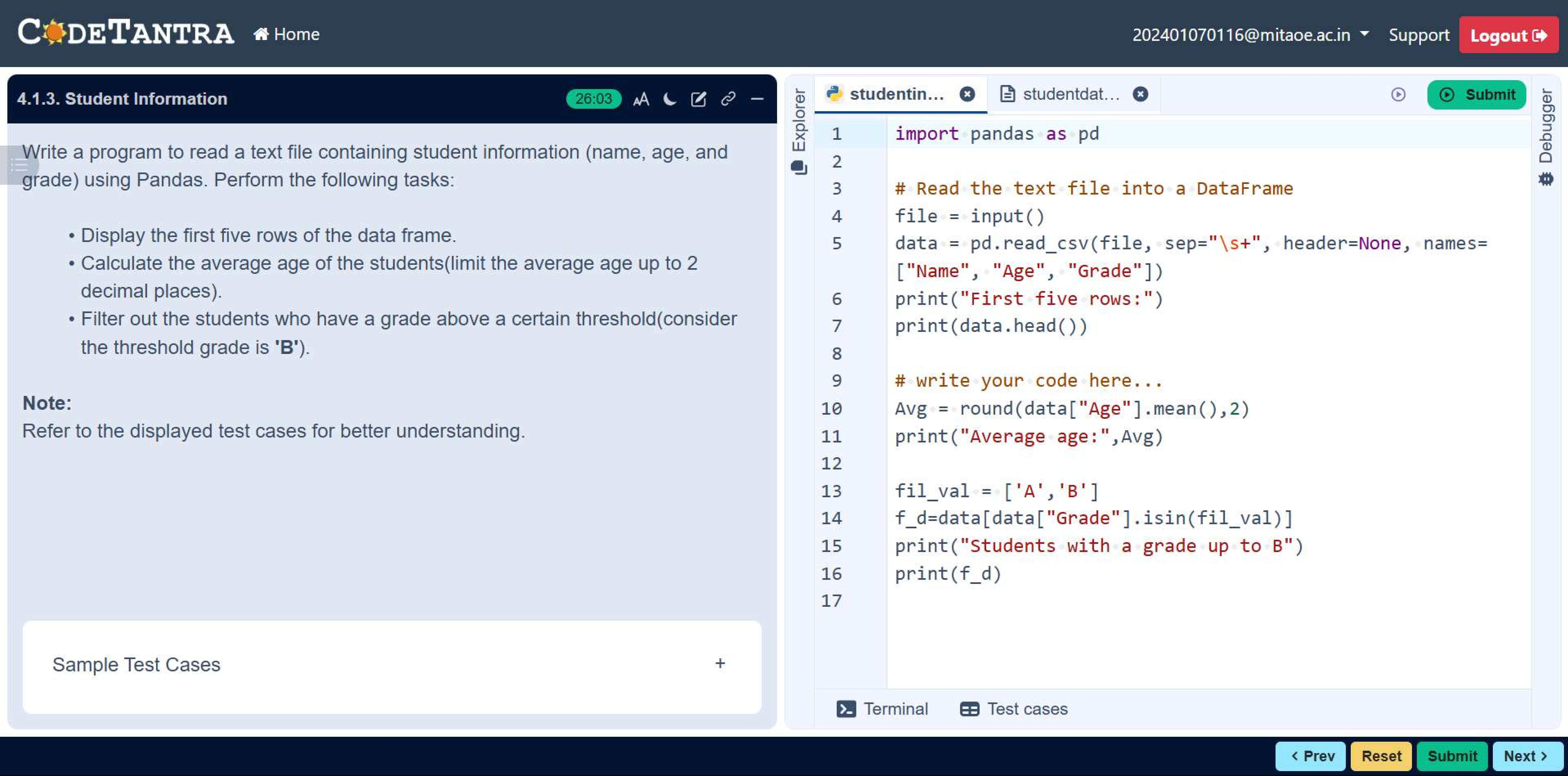
```python
import numpy as np

# Input matrices
print("Enter Array1:")
arr1 = np.array([list(map(int, input().split())) for i in range(3)])

print("Enter Array2:")
arr2 = np.array([list(map(int, input().split())) for i in range(3)])

# Perform horizontal stacking (hstack)
h_stack = np.hstack((arr1, arr2))
print("Horizontal Stack:")
print(h_stack)

# Perform vertical stacking (vstack)
v_stack = np.vstack((arr1, arr2))
print("Vertical Stack:")
print(v_stack)
```



4.1.3. Student Information

Write a program to read a text file containing student information (name, age, and grade) using Pandas. Perform the following tasks:

- Display the first five rows of the data frame.
- Calculate the average age of the students(limit the average age up to 2 decimal places).
- Filter out the students who have a grade above a certain threshold(consider the threshold grade is **'B'**).

Note:

Refer to the displayed test cases for better understanding.

Sample Test Cases +

```python
import pandas as pd

# Read the text file into a DataFrame
file = input()
data = pd.read_csv(file, sep="\s+", header=None, names=["Name", "Age", "Grade"])
print("First five rows:")
print(data.head())

# write your code here...
Avg = round(data["Age"].mean(),2)
print("Average age:",Avg)

fil_val = ['A','B']
f_d=data[data["Grade"].isin(fil_val)]
print("Students with a grade up to B")
print(f_d)

```