Rajalakshmi Engineering College

Name: Rakesh H
Email: 240701415@rajalakshmi.edu.in
Roll no: 240701415
Phone: 7305737702
Branch: REC
Department: I CSE FD
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23221_Python Programming

REC_Python_Week 6_PAH

Attempt : 1
Total Mark : 30
Marks Obtained : 18.5

Section 1 : Coding

1. Problem Statement

Peter manages a student database and needs a program to add students. For each student, Alex inputs their ID and name. The program checks for duplicate IDs and ensures the database isn't full.

If a duplicate or a full database is detected, an appropriate error message is displayed. Otherwise, the student is added, and a confirmation message is shown. The database has a maximum capacity of 30 students, and each student must have a unique ID.

Input Format

The first line contains an integer n, representing the number of students to be added to the school database.

The next n lines each contain two space-separated values, representing the student's ID (integer) and the student's name (string).

Output Format

The output will depend on the actions performed in the code.

If a student is added to the database, the output will display: "Student with ID [ID number] added to the database."

If there is an exception due to a duplicate student ID, the output will display: "Exception caught. Error: Student ID already exists."

If there is an exception due to the database being full, the output will display: "Exception caught. Error: Student database is full."

Refer to the sample outputs for the formatting specifications.

Sample Test Case

Input: 3
16 Sam
87 Sabari
43 Dani

Output: Student with ID 16 added to the database.
Student with ID 87 added to the database.
Student with ID 43 added to the database.

Answer

```
class StudentDatabase:
    def __init__(self):
        self.MAX_CAPACITY = 30
        self.students = {}

    def add_student(self, student_id, name):
        try:

            if len(self.students) >= self.MAX_CAPACITY:
```

```
            print("Exception caught. Error: Student database is full.")
            return False


        if student_id in self.students:
            raise Exception("Student ID already exists")


        self.students[student_id] = name
        print(f"Student with ID {student_id} added to the database.")
        return True

    except Exception as e:
        print(f"Exception caught. Error: {str(e)}.")
        return True


def main():
    db = StudentDatabase()


    n = int(input())


    for _ in range(n):

        student_id, name = input().split()
        student_id = int(student_id)
        # Stop processing if database is full
        if not db.add_student(student_id, name):
            break

# Run the program
if __name__ == "__main__":
    main()
```

Status : Partially correct **Marks : 8.5/10**

2. Problem Statement

Reeta is playing with numbers. Reeta wants to have a file containing a list

of numbers, and she needs to find the average of those numbers. Write a program to read the numbers from the file, calculate the average, and display it.

File Name: user_input.txt

Input Format

The input file will contain a single line of space-separated numbers (as a string).

These numbers may be integers or decimals.

Output Format

If all inputs are valid numbers, the output should print: "Average of the numbers is: X.XX" (where X.XX is the computed average rounded to two decimal places)

If the input contains invalid data, print: "Invalid data in the input."

Refer to the sample output for format specifications.

Sample Test Case

Input: 1 2 3 4 5
Output: Average of the numbers is: 3.00

Answer

```
def calculate_average():
    try:
        # Open and read the file
        with open('user_input.txt', 'r') as file:
            # Read the single line and split into numbers
            numbers = file.readline().strip().split()

        # Check if the number of inputs is within constraints (1 ≤ n ≤ 100)
        if not (1 <= len(numbers) <= 100):
            print("Invalid data in the input.")
            return

        # Convert strings to floats and validate
        num_list = []
```

```
        for num in numbers:
            # Remove any unexpected whitespace
            num = num.strip()
            try:
                num_list.append(float(num))
            except ValueError:
                print("Invalid data in the input.")
                return

        # Calculate average
        average = sum(num_list) / len(num_list)

        # Print average rounded to two decimal places
        print(f"Average of the numbers is: {average:.2f}")

    except FileNotFoundError:
        print("Invalid data in the input.")
    except Exception as e:
        # For debugging, print the exception (remove in final submission)
        # print(f"Debug: Exception occurred: {str(e)}")
        print("Invalid data in the input.")

# Run the program
if __name__ == "__main__":
    calculate_average()
```

Status : Partially correct ***Marks : 5/10***

3. Problem Statement

John is a data analyst who often works with text files. He needs a program that can analyze the contents of a text file and count the number of times a specific character appears in the file.

John wants a simple program that allows him to specify a file and a character to count within that file.

Input Format

The first line of input consists of the file's name to be analyzed.

The second line of the input consists of the string they want to write within the file.

The third line of the input consists of a character to count within the file.

Output Format

If the character is found, the output displays "The character 'X' appears {Y} times in the file." where X is the character and Y i the count,

If the character does not appear in the file, the output displays "Character not found."

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: test.txt
This is a test file to check the character count.
e
Output: The character 'e' appears 5 times in the file.

Answer

```
def count_character():
    try:
        # Read input
        file_name = input().strip()
        content = input().strip()
        char_to_count = input()

        # Validate character input (single character, including space)
        char_to_count = char_to_count.strip()
        if len(char_to_count) != 1:
            print("Invalid input: Please enter a single character.")
            return

        # Validate string length (5 ≤ length ≤ 500)
        if not (5 <= len(content) <= 500):
```

```
            print("Invalid input: String length must be between 5 and 500 characters.")
            return

        # Write content to the file
        with open(file_name, 'w') as file:
            file.write(content)

        # Read content from the file
        with open(file_name, 'r') as file:
            text = file.read()

        # Count occurrences of the character
        count = text.count(char_to_count)

        # Output result
        if count > 0:
            print(f"The character '{char_to_count}' appears {count} times in the file.")
        else:
            print("Character not found in the file.")

    except Exception:
        print("An error occurred while processing the file.")

# Run the program
if __name__ == "__main__":
    count_character()
```

Status : Partially correct **Marks : 5/10**