Rajalakshmi Engineering College

Name: SHIVANISREE K B
Email: 240701501@rajalakshmi.edu.in
Roll no: 240701501
Phone: 7358464804
Branch: REC
Department: I CSE FE
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 : 28.5

Section 1 : Coding

1. 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

```
# You are using Python

# Read input
file_name = input().strip()
content = input()
char_to_count = input()

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

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

# Count the occurrences of the character
c=char_to_count.lower()
d=data.lower()
count = d.count(c)
```

```
# Display the output based on the count
if count > 0:
    print(f"The character '{char_to_count}' appears {count} times in the file.")
else:
    print("Character not found in the file.")
```

Status : Correct **Marks : 10/10**

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

```
# You are using Python
# Read input (the numbers as a string)
content = input().strip()

# File name is fixed as per problem description
file_name = "user_input.txt"

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

try:
    # Read the file
    with open(file_name, "r") as f:
        line = f.readline().strip()
        parts = line.split()

        numbers = []
        for part in parts:
            try:
                number = float(part)
                numbers.append(number)
            except ValueError:
                print("Invalid data in the input.")
                break
        else:
            if numbers:
                avg = sum(numbers) / len(numbers)
                print(f"Average of the numbers is: {avg:.2f}")
            else:
                print("Invalid data in the input.")

except FileNotFoundError:
    print("Invalid data in the input.")
```

Status : Correct **Marks : 10/10**

3. 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

```
# You are using Python
MAX_CAPACITY = 30

n = int(input())
student_ids = set()
students_added = 0
db_full_flag = False  # To ensure the "database is full" message is printed only
once

for _ in range(n):
    entry = input().strip().split(maxsplit=1)

    if len(entry) != 2:
        continue  # Skip malformed input

    student_id_str, student_name = entry
    try:
        student_id = int(student_id_str)

        if students_added >= MAX_CAPACITY:
            if not db_full_flag:
                print("Exception caught. Error: Student database is full.")
                db_full_flag = True
            continue  # Skip further additions

        if student_id in student_ids:
            print("Exception caught. Error: Student ID already exists.")
            continue

        student_ids.add(student_id)
        students_added += 1
        print(f"Student with ID {student_id} added to the database.")

    except ValueError:
        continue  # Ignore invalid ID parsing
```

Status : Partially correct *Marks : 8.5/10*