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_CY

Attempt : 1
Total Mark : 40
Marks Obtained : 34

Section 1 : Coding

1. Problem Statement

Bob, a data analyst, requires a program to automate the process of analyzing character frequency in a given text. This program should allow the user to input a string, calculate the frequency of each character within the text, save these character frequencies to a file named "char_frequency.txt," and display the results.

Input Format

The input consists of the string.

Output Format

The first line prints "Character Frequencies:".

The following lines print the character frequency in the format: "X: Y" where X is the character and Y is the count.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: aaabbbccc

Output: Character Frequencies:
a: 3
b: 3
c: 3

Answer

```
def analyze_character_frequency():
    try:
        # Read input string
        text = input().strip()

        # Validate string length (1 ≤ length ≤ 1000)
        if not (1 <= len(text) <= 1000):
            print("Invalid input: String length must be between 1 and 1000
characters.")
            return

        # Calculate character frequencies and maintain order of appearance
        char_freq = {}
        char_order = []
        for char in text:
            if char not in char_freq:
                char_order.append(char)  # Track order of first appearance
            char_freq[char] = char_freq.get(char, 0) + 1

        # Save frequencies to char_frequency.txt
        with open('char_frequency.txt', 'w') as file:
            for char in char_order:
                file.write(f"{char}: {char_freq[char]}\n")

        # Display frequencies
        print("Character Frequencies:")
        for char in char_order:
```

```
        print(f"{char}: {char_freq[char]}")

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

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

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

2. Problem Statement

Alex is creating an account and needs to set up a password. The program prompts Alex to enter their name, mobile number, chosen username, and desired password. Password validation criteria include:

Length between 10 and 20 characters.At least one digit.At least one special character from !@#$%^&* set. Display "Valid Password" if criteria are met; otherwise, raise an exception with an appropriate error message.

Input Format

The first line of the input consists of the name as a string.

The second line of the input consists of the mobile number as a string.

The third line of the input consists of the username as a string.

The fourth line of the input consists of the password as a string.

Output Format

If the password is valid (meets all the criteria), it will print "Valid Password"

If the password is weak (fails any one or more criteria), it will print an error message accordingly.

Refer to the sample outputs for the formatting specifications.

Sample Test Case

Input: John
9874563210
john
john1#nhoj

Output: Valid Password

Answer

```
def validate_password(password):
    # Check for at least one digit
    if not any(char.isdigit() for char in password):
        raise Exception("Should contain at least one digit")

    # Check for at least one special character
    special_characters = "!@#$%^&*"
    if not any(char in special_characters for char in password):
        raise Exception("It should contain at least one special character")

    # Check length
    if len(password) < 10 or len(password) > 20:
        raise Exception("Should be a minimum of 10 characters and a maximum of
20 characters")

    print("Valid Password")

# Main input
try:
    name = input()
    mobile = input()
    username = input()
    password = input()

    validate_password(password)

except Exception as e:
    print(e)
```

Status : Partially correct ***Marks : 6.5/10***

3. Problem Statement

Alice is developing a program called "Name Sorter" that helps users organize and sort names alphabetically.

The program takes names as input from the user, saves them in a file, and then displays the names in sorted order.

File Name: sorted_names.txt.

Input Format

The input consists of multiple lines, each containing a name represented as a string.

To end the input and proceed with sorting, the user can enter 'q'.

Output Format

The output displays the names in alphabetical order, each name on a new line.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: Alice Smith
John Doe
Emma Johnson
q
Output: Alice Smith
Emma Johnson
John Doe

Answer

```
# You are using Python
def is_valid_name(name):
    # Check if name length is between 3 and 30 characters
    if not (3 <= len(name) <= 30):
        return False
    # Check if name contains only alphabetic characters and spaces
```

```
    return all(char.isalpha() or char.isspace() for char in name)

def name_sorter():
    try:
        # List to store names
        names = []

        # Read names until 'q' is entered
        while True:
            name = input().strip()
            if name.lower() == 'q':
                break
            names.append(name)

        # Validate number of names (3 ≤ n ≤ 20)
        if not (3 <= len(names) <= 20):
            print("Invalid input: Number of names must be between 3 and 20.")
            return

        # Validate each name
        for name in names:
            if not is_valid_name(name):
                print("Invalid input: Each name must be 3 to 30 characters long and
contain only alphabetic characters or spaces.")
                return

        # Sort names alphabetically
        names.sort()

        # Save sorted names to sorted_names.txt
        with open('sorted_names.txt', 'w') as file:
            for name in names:
                file.write(f"{name}\n")

        # Display sorted names
        for name in names:
            print(name)

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

# Run the program
```

```
if __name__ == "__main__":
    name_sorter()
```

Status : Partially correct ***Marks : 7.5/10***

4. Problem Statement

A shopkeeper is recording the daily sales of an item for N days, where the price of the item remains the same for all days. Write a program to calculate the total sales for each day and save them in a file named sales.txt that can store the data for a maximum of 30 days. Then, read the file and display the total earnings for each day.

Note: Total Earnings for each day = Number of Items sold in that day × Price of the item.

Input Format

The first line of input consists of an integer N, representing the number of days.

The second line of input consists of N space-separated integers representing the number of items sold each day.

The third line of input consists of an integer M, representing the price of the item that is common for all N days.

Output Format

If the number of days entered exceeds 30 (N > 30), the output prints "Exceeding limit!" and terminates.

Otherwise, the code reads the contents of the file and displays the total earnings for each day on separate lines.

Contents of the file: The total earnings for N days, with each day's earnings appearing on a separate line.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 4
5 10 5 0
20

Output: 100
200
100
0

Answer

```
# Step 1: Get the input values
N = int(input())  # Number of days

# Step 2: Check constraint
if N > 30:
    print("Exceeding limit!")
else:
    items_sold = list(map(int, input().split()))
    M = int(input())  # Price of the item

    # Step 3: Calculate total earnings for each day
    total_earnings = [sold * M for sold in items_sold]

    # Step 4: Write to file "sales.txt"
    with open("sales.txt", "w") as file:
        for earning in total_earnings:
            file.write(str(earning) + "\n")

    # Step 5: Read from file and display output
    with open("sales.txt", "r") as file:
        for line in file:
            print(line.strip(), end=' ')
```

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