Rajalakshmi Engineering College

Name: Muhammed Qaiser
Email: 240701621@rajalakshmi.edu.in
Roll no: 240701621
Phone: 9363147459
Branch: REC
Department: I CSE FF
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 : 40

Section 1 : Coding

1. Problem Statement

In the enchanted realm of Academia, you, the Academic Alchemist, are bestowed with a magical quill and a parchment to weave the grades of aspiring students into a tapestry of academic brilliance.

The mission is to craft a Python program that empowers faculty members to enter student grades for any two subjects, stores these magical grades in a mystical file, and then, with a wave of your virtual wand, calculates the GPA to unveil the true essence of academic achievement.

Input Format

The input format is a string representing the student's name, any two subjects, and corresponding grades.

After entering grades, they can type 'done' when prompted for the student's name.

Output Format

The output should display the (average of grades) calculated GPA with a precision of two decimal places.

The magical grades will be saved in a mystical file named "magical_grades.txt".

Refer to the sample output for format specifications.

Sample Test Case

Input: Alice
Math
95
English
88
done

Output: 91.50

Answer

```
# You are using Python
def calculate_gpa():
    grades_data = []
    while True:
        name = input()
        if name.lower() == 'done':
            break
        try:
            subject1 = input()
            grade1 = float(input())
            subject2 = input()
            grade2 = float(input())
            if not (0 <= grade1 <= 100 and 0 <= grade2 <= 100):
                print("Hark! Grades must be between 0 and 100.")
                continue
            grades_data.append((name, subject1, grade1, subject2, grade2))
            average_grade = (grade1 + grade2) / 2
            print(f"{average_grade:.2f}")
```

```
        except ValueError:
            print("By the mystic rivers! Invalid input. Please enter numeric grades.")
    with open("magical_grades.txt", "w") as f:
        for entry in grades_data:
            f.write(f"{entry[0]},{entry[1]},{entry[2]},{entry[3]},{entry[4]}\n")

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

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

2. 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 name_sorter():
    names = []
    while True:
        name = input()
        if name.lower() == 'q':
            break
        names.append(name)
    with open("sorted_names.txt", "w") as f:
        for name in names:
            f.write(name + "\n")
    sorted_names = sorted(names)
    for name in sorted_names:
        print(name)

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

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

3. 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 character_frequency_analyzer():
    input_string = input()
    char_counts = {}
    for char in input_string:
        char_counts[char] = char_counts.get(char, 0) + 1
    with open("char_frequency.txt", "w") as f:
        f.write("Character Frequencies:\n")
        for char, count in char_counts.items():
            f.write(f"{char}: {count}\n")
    print("Character Frequencies:")
    for char, count in char_counts.items():
        print(f"{char}: {count}")

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

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

4. 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 main():
    name = input().strip()
    mobile_number = input().strip()
    username = input().strip()
    password = input().strip()

    # Validate length first
    if not (10 <= len(password) <= 20):
```

```
        print("Should be a minimum of 10 characters and a maximum of 20
characters")
        return

    # Validate digit presence
    if not any(char.isdigit() for char in password):
        print("Should contain at least one digit")
        return

    # Validate special character presence
    special_chars = "!@#$%^&*"
    if not any(char in special_chars for char in password):
        print("It should contain at least one special character")
        return

    # If all validations pass
    print("Valid Password")


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

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