Rajalakshmi Engineering College

Name: Raghul M
Email: 240701409@rajalakshmi.edu.in
Roll no: 240701409
Phone: 9150457149
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 : 36.5

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

```
# You are using Python
from collections import Counter

file_name = "char_frequency.txt"

from collections import Counter

file_name = "char_frequency.txt"

text = input()

with open(file_name, 'w') as file:
    file.write(text)

with open(file_name, 'r') as file:
    text = file.read().strip()

char_count = Counter(text)

print("Character Frequencies:")
with open(file_name, 'w') as file:
    file.write("Character Frequencies:\n")


    for char in text:
        if char in char_count:
            print(f"{char}: {char_count[char]}")
```

```
        file.write(f"{char}: {char_count[char]}\n")
        del char_count[char]
```

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
file_name = "sorted_names.txt"

names = []

while True:
    name = input().strip()
    if name.lower() == 'q':
        break
    names.append(name)

names.sort()

with open(file_name, 'w') as file:
    for name in names:
        file.write(name + "\n")

with open(file_name, 'r') as file:
    sorted_names = file.read().strip()

print(sorted_names)
```

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

3. 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

```
# You are using Python
def validate_password(name, mobile, username, password):
    try:

        if not isinstance(password, str) or not password:
            raise ValueError("Should be a minimum of 10 characters and a maximum
of 20 characters")


        if not any(char.isdigit() for char in password):
            raise ValueError("Should contain at least one digit")


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


        if not (10 <= len(password) <= 20):
```

```
        raise ValueError("Should be a minimum of 10 characters and a maximum
of 20 characters")


        return "Valid Password"

    except ValueError as e:
        return str(e)


name = input()
mobile = input()
username = input()
password = input()


print(validate_password(name, mobile, username, password))
```

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

4. Problem Statement

Write a program to obtain the start time and end time for the stage event show. If the user enters a different format other than specified, an exception occurs and the program is interrupted. To avoid that, handle the exception and prompt the user to enter the right format as specified.

Start time and end time should be in the format 'YYYY-MM-DD HH:MM:SS'If the input is in the above format, print the start time and end time.If the input does not follow the above format, print "Event time is not in the format "

Input Format

The first line of input consists of the start time of the event.

The second line of the input consists of the end time of the event.

Output Format

If the input is in the given format, print the start time and end time.

If the input does not follow the given format, print "Event time is not in the format".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 2022-01-12 06:10:00
2022-02-12 10:10:12

Output: 2022-01-12 06:10:00
2022-02-12 10:10:12

Answer

```
# You are using Python
from datetime import datetime

def validate_time_format(time_str):
    try:
        datetime.strptime(time_str, "%Y-%m-%d %H:%M:%S")
        return True
    except ValueError:
        return False

start_time = input().strip()
end_time = input().strip()

if validate_time_format(start_time) and validate_time_format(end_time):
    print(start_time)
    print(end_time)
else:
    print("Event time is not in the format")
```

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