Rajalakshmi Engineering College

Name: SONASREE RP
Email: 240701521@rajalakshmi.edu.in
Roll no: 240701521
Phone: 7305340666
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_CY

Attempt : 1
Total Mark : 40
Marks Obtained : 33.5

Section 1 : Coding

1. 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

```
# You are using Python

N = int(input())

if N > 30:
    print("Exceeding limit!")
    exit()

items_sold = list(map(int, input().split()))
```

```
M = int(input())

with open("sales.txt", "w") as file:
    for sold in items_sold:
        earnings = sold * M
        file.write(str(earnings) + "\n")

with open("sales.txt", "r") as file:
    for line in file:
        print(line.strip())
```

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

2. 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(input_time):
    try:

        parsed_time = datetime.strptime(input_time, "%Y-%m-%d %H:%M:%S")
        return parsed_time
    except ValueError:

        raise


start_time_input = input()
end_time_input = input()

try:
    start_time = validate_time(start_time_input)
    end_time = validate_time(end_time_input)
    print(start_time_input)
    print(end_time_input)
except:
    print("Event time is not in the format")
```

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

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

text = input()

char_freq = OrderedDict()

for char in text:
    if char in char_freq:
        char_freq[char] += 1
    else:
        char_freq[char] = 1

with open("char_frequency.txt", "w") as file:
    file.write("Character Frequencies:\n")
```

```
    for char, freq in char_freq.items():
        file.write(f"{char}: {freq}\n")

print("Character Frequencies:")
for char, freq in char_freq.items():
    print(f"{char}: {freq}")
```

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

```
# You are using Python
def validate_password(password):
    special_characters = "!@#$%^&*"

    if len(password) < 10:
        raise Exception("Should be a minimum of 10 char")
    if len(password) > 20:
        raise Exception("Should be a maximum of 20 char")

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

    if not any(char in special_characters for char in password):
        raise Exception("It should contain at least one special character")

    print("Valid Password")

name = input()
mobile_number = input()
username = input()
password = input()

try:
    validate_password(password)
except Exception as e:
    print(e)
```

Status : Partially correct ***Marks : 3.5/10***