Rajalakshmi Engineering College

Name: Elango G
Email: 241501055@rajalakshmi.edu.in
Roll no: 241501055
Phone: 7010568330
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

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 main():
    while True:
        student_name = input().strip()
        if student_name.lower() == 'done':
            break
        subjects = []
        grades = []
        for _ in range(2):
            subject = input().strip()
            subjects.append(subject)
            grade = float(input().strip())
            if grade < 0 or grade > 100:
                return
            grades.append(grade)
        gpa = sum(grades) / len(grades)
        with open("magical_grades.txt", "a") as file:
            file.write(f"{student_name}: {subjects[0]} {grades[0]}, {subjects[1]}
{grades[1]}\n")
```

```
    print(f"{gpa:.2f}")

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

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 main():
    try:
        start_time = input().strip()
        end_time = input().strip()
        datetime.strptime(start_time, '%Y-%m-%d %H:%M:%S')
        datetime.strptime(end_time, '%Y-%m-%d %H:%M:%S')
        print(start_time)
        print(end_time)
    except ValueError:
        print("Event time is not in the format")

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

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

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

    names.sort()

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

    for name in names:
        print(name)

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

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 !@#$%^&amp;* 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
# Read inputs
name = input()
```

```
mobile_number = input()
username = input()
password = input()

# Function to validate the password
def validate_password(password):
    special_characters = set("!@#$%^&*")

    length_valid = 10 <= len(password) <= 20
    has_digit = any(char.isdigit() for char in password)
    has_special = any(char in special_characters for char in password)

    # Perform checks in this order
    if not length_valid:
        raise Exception("Should be a minimum of 10 characters and a maximum of
20 characters")
    if not has_digit:
        raise Exception("Should contain at least one digit")
    if not has_special:
        raise Exception("It should contain at least one special character")

# Main block
try:
    validate_password(password)
    print("Valid Password")
except Exception as e:
    print(e)
```

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