Rajalakshmi Engineering College

Name: Neha R N
Email: 240701354@rajalakshmi.edu.in
Roll no: 240701354
Phone: 9080137196
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 : 40

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 OrderedDict

input_text = input()

char_freq = OrderedDict()

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

with open("char_frquency.txt", "w") as file:
    for char, freq in char_freq.items():
        file.write(f"{char}: {freq}")

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

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

2. 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 validate_grade(grade):
    try:
        grade = float(grade)
        if not 0<= grade <= 100:
            raise ValueError("Grade must be between 0 and 100")
        return grade
    except ValueError as e:
        if "Grade must" in str(e):
            raise
        raise ValueError("Invalid grade input")

def main():
    grades_data = []
    while True:
        student_name = input().strip()

        if student_name.lower() == 'done':
            break
        subject1 = input().strip()
        grade1 = input().strip()
        subject2 = input().strip()
        garde2 = input().strip()

        try:
            g1 = validate_grade(grade1)
            g2  =validate_grade(garde2)

            grades_data.append({
                'name': student_name,
                'subject1': subject1,
                'grade1': g1,
                'subject2': subject1,
                'grade2': g2
            })
        except ValueError as e:
            print(f"Error: {str(e)}")
            return
    if grades_data:
        last_student = grades_data[-1]
        gpa = (last_student['grade1'] + last_student['grade2']) / 2
        print(f"{gpa:.2f}")

        with open('magical_grades.txt', 'a') as file:
            for student in grades_data:
                file.write(f"Stduent: {student['name']}\n")
                file.write(f"{student['subject2']}: {student['grade2']}\n")
        file.write(f"GPA: {(student['grade1'] + student['grade2']) / 2:.2f}\n\n")

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

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

3. Problem Statement

Implement a program that checks whether a set of three input values can form the sides of a valid triangle. The program defines a function is_valid_triangle that takes three side lengths as arguments and raises a ValueError if any side length is not a positive value. It then checks whether the sum of any two sides is greater than the third side to determine the validity of the triangle.

Input Format

The first line of input consists of an integer A, representing side1.

The second line of input consists of an integer B, representing side2.

The third line of input consists of an integer C, representing side3.

Output Format

The output prints either "It's a valid triangle" if the input side lengths form a valid triangle,

or "It's not a valid triangle" if they do not.

If there is a ValueError, it should print "ValueError: <error_message>".

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 3
4

5
Output: It's a valid triangle

Answer

```
# You are using Python
def is_valid_triangle(a,b,c):
   if a<=0 or b<= 0 or c<= 0:
      raise ValueError("Side lengths must be positive")

   if(a+b <= c) or(b+c <= a) or (a+c <= b):
      return False
   return True

try:
   side1 = int(input())
   side2 = int(input())
   side3 = int(input())

   if is_valid_triangle(side1, side2, side3):
      print("It's a valid triangle")
   else:
      print("It's not a valid triangle")

except ValueError as e:
   if str(e) == "Side lengths must be positive":
      print(f"valueError: {str(e)}")
   else:
      print("ValueError: Invalid input")
```

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

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

try:
    start_time = input()
    end_time = input()

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

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