Rajalakshmi Engineering College

Name: abinraj cr
Email: 241501007@rajalakshmi.edu.in
Roll no: 241501007
Phone: 7550268466
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
grades_list = []

with open("magical_grades.txt", "w") as file:
    while True:
        student_name = input().strip()
        if student_name.lower() == "done":
            break

        subject1 = input().strip()
        grade1 = input().strip()
        subject2 = input().strip()
        grade2 = input().strip()

        # Validate grades
        try:
            grade1 = float(grade1)
            grade2 = float(grade2)
            if not (0 <= grade1 <= 100 and 0 <= grade2 <= 100):
                # Ignore invalid grades (could also raise error, but not specified)
                continue
        except ValueError:
            continue

        # Write to file
        file.write(f"{student_name} {subject1} {grade1} {subject2} {grade2}\n")

        # Keep track of grades for GPA
        grades_list.append(grade1)
        grades_list.append(grade2)

if grades_list:
    gpa = sum(grades_list) / len(grades_list)
    print(f"{gpa:.2f}")
else:
    print("0.00")
```

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

2. Problem Statement

Write a program to read the Register Number and Mobile Number of a student. Create user-defined exception and handle the following:

If the Register Number does not contain exactly 9 characters in the specified format(2 numbers followed by 3 characters followed by 4 numbers) or if the Mobile Number does not contain exactly 10 characters, throw an IllegalArgumentException. If the Mobile Number contains any character other than a digit, raise a NumberFormatException.If the Register Number contains any character other than digits and alphabets, throw a NoSuchElementException.If they are valid, print the message ‘valid’ or else print an Invalid message.

Input Format

The first line of the input consists of a string representing the Register number.

The second line of the input consists of a string representing the Mobile number.

Output Format

The output should display any one of the following messages:

If both numbers are valid, print "Valid".

If an exception is raised, print "Invalid with exception message: ", followed by the specific exception message.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 19ABC1001
9949596920

Output: Valid

Answer

```
# You are using Python
class IllegalArgumentException(Exception):
   pass

class NoSuchElementException(Exception):
   pass

def validate_register_number(reg_num):
   if len(reg_num) != 9:
      raise IllegalArgumentException("Register Number should have exactly 9 
characters.")

   # Check all characters are alphanumeric (digits or letters)
   if not reg_num.isalnum():
      raise NoSuchElementException("Register Number should have only 
alphabets and digits.")

   # Check format: 2 digits, 3 letters, 4 digits
   part1 = reg_num[0:2]
   part2 = reg_num[2:5]
   part3 = reg_num[5:9]
```

```
    if not (part1.isdigit() and part2.isalpha() and part3.isdigit()):
        raise IllegalArgumentException("Register Number should have the format: 2
numbers, 3 characters, and 4 numbers.")

def validate_mobile_number(mobile):
    if len(mobile) != 10:
        raise IllegalArgumentException("Mobile Number should have exactly 10
characters.")

    if not mobile.isdigit():
        raise NumberFormatException("Mobile Number should only contain digits.")

class NumberFormatException(Exception):
    pass

try:
    reg_num = input().strip()
    mobile = input().strip()

    validate_register_number(reg_num)
    validate_mobile_number(mobile)

    print("Valid")

except (IllegalArgumentException, NoSuchElementException,
NumberFormatException) as e:
    print(f"Invalid with exception message: {e}")
```

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

3. 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

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

    # Define the expected format
    fmt = "%Y-%m-%d %H:%M:%S"

    # Try parsing both times
    datetime.strptime(start_time, fmt)
    datetime.strptime(end_time, fmt)

    # If successful, print the times
    print(start_time, end_time)
```

```
except ValueError:
  print("Event time is not in the format")
```

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

4. 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()
```

```
freq = OrderedDict()
for ch in text:
    freq[ch] = freq.get(ch, 0) + 1

with open("char_frequency.txt", "w") as f:
    f.write("Character Frequencies: ")
    for ch, count in freq.items():
        f.write(f"{ch}: {count} ")

print("Character Frequencies: ", end="")
for ch, count in freq.items():
    print(f"{ch}: {count} ", end="")
```

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