Rajalakshmi Engineering College

Name: Darshan Abinav R.K
Email: 241501039@rajalakshmi.edu.in
Roll no: 241501039
Phone: 7010796406
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

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

```
from collections import OrderedDict

text = input()

frequency = OrderedDict()

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

print("Character Frequencies:")
with open("char_frequency.txt", "w") as file:
    for char, count in frequency.items():
        line = f"{char}: {count}"
        print(line)
        file.write(line + "\n")
```

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

2. 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

```
def is_valid_triangle(a, b, c):
    if a <= 0 or b <= 0 or c <= 0:
        raise ValueError("Side lengths must be positive")
    return (a + b > c) and (a + c > b) and (b + c > a)

try:
    a = int(input())
    b = int(input())
```

```
    c = int(input())

    if is_valid_triangle(a, b, c):
        print("It's a valid triangle")
    else:
        print("It's not a valid triangle")
except ValueError as ve:
    print(f"ValueError: {ve}")
```

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

```
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()
end_time = input()

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***

4. 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

```
import re

class IllegalArgumentException(Exception):
    pass

class NumberFormatException(Exception):
    pass

class NoSuchElementException(Exception):
    pass

def validate_register_number(reg_no):
    if len(reg_no) != 9:
        raise IllegalArgumentException("Register Number should have exactly 9
characters.")
    if not re.match(r'^\d{2}[A-Za-z]{3}\d{4}$', reg_no):
        if re.match(r'^[0-9A-Za-z]{9}$', reg_no):
```

```
        raise IllegalArgumentException("Register Number should have the format:
2 numbers, 3 characters, and 4 numbers.")
      else:
        raise NoSuchElementException("Register Number should only contain
digits and alphabets.")

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.")

try:
  reg_no = input().strip()
  mobile = input().strip()
  validate_register_number(reg_no)
  validate_mobile_number(mobile)
  print("Valid")
except (IllegalArgumentException, NumberFormatException,
NoSuchElementException) as e:
  print("Invalid with exception message:", e)
```

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