Rajalakshmi Engineering College

Name: Manju sri N
Email: 241801151@rajalakshmi.edu.in
Roll no: 241801151
Phone: 8946059431
Branch: REC
Department: I AI & DS FC
Batch: 2028
Degree: B.E - AI & DS

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

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 NumberFormatException(Exception):
   pass

from collections import deque

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 (reg_no[:2].isdigit() and reg_no[2:5].isalpha() and reg_no[5:].isdigit()):
      raise IllegalArgumentException("Register Number should have the format: 2 
numbers, 3 characters, and 4 numbers.")
   for ch in reg_no:
        if not (ch.isdigit() or ch.isalpha()):
            raise NoSuchElementException("Register Number should contain only
alphabets and digits.")

def validate_mobile_number(mobile_no):
    if len(mobile_no) != 10:
        raise IllegalArgumentException("Mobile Number should have exactly 10
characters.")
    if not mobile_no.isdigit():
        raise NumberFormatException("Mobile Number should only contain digits.")

def main():
    try:
        reg_no = input()
        mobile_no = input()

        validate_register_number(reg_no)
        validate_mobile_number(mobile_no)

        print("Valid")

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

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

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

```
# 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) and (a + c > b) and (b + c > a):
        return True
    else:
        return False

def main():
    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}")

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

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

3. 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
def main():
    N = int(input())

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

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

    # Calculate total earnings and write to file
    with open("sales.txt", "w") as file:
        for items in items_sold:
            earnings = items * M
            file.write(str(earnings) + "\n")

    # Read from file and display earnings
    with open("sales.txt", "r") as file:
        for line in file:
            print(line.strip(), end=" ")
```

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

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

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

        # Try to parse the datetime strings
        start_dt = datetime.strptime(start_time, format)
        end_dt = datetime.strptime(end_time, format)

        # If both parse successfully, print them
        print(start_time, end_time)

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

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

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