Rajalakshmi Engineering College

Name: I Mohammed Hamza
Email: 240701326@rajalakshmi.edu.in
Roll no: 240701326
Phone: 7358328592
Branch: REC
Department: I CSE AH
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

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

```
n = int(input())
if n > 30:
    print("Exceeding limit!")
else:
    items = list(map(int, input().split()))
    price = int(input())
    earnings = [i * price for i in items]
    with open("sales.txt", "w") as f:
        for e in earnings:
```

```
        f.write(str(e) + "\n")
  with open("sales.txt", "r") as f:
      for line in f:
          print(line.strip())
```

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

```
class IllegalArgumentException(Exception):
    pass

class NoSuchElementException(Exception):
    pass

try:
    reg_no = input().strip()
    mobile_no = input().strip()

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

    if not reg_no[:2].isdigit() or not reg_no[2:5].isalpha() or not reg_no[5:].isdigit():
        raise IllegalArgumentException("Register Number should have the format: 2 
numbers, 3 characters, and 4 numbers.")

    if not reg_no.isalnum():
        raise NoSuchElementException("Register Number should contain only digits 
and alphabets.")

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

    if not mobile_no.isdigit():
        raise ValueError("Mobile Number should only contain digits.")

    print("Valid")

except (IllegalArgumentException, NoSuchElementException, ValueError) as e:
    print("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

```
from datetime import datetime

try:
    start = input().strip()
    end = input().strip()
    start_time = datetime.strptime(start, '%Y-%m-%d %H:%M:%S')
    end_time = datetime.strptime(end, '%Y-%m-%d %H:%M:%S')

    print(start, end)

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

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

4. 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

```
names = []

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


names.sort()

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

# Read from file and print sorted names
with open("sorted_names.txt", "r") as file:
    for line in file:
        print(line.strip())
```

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