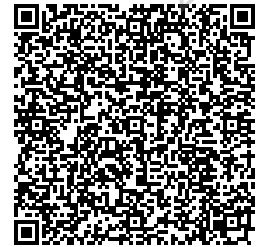


# Rajalakshmi Engineering College

Name: Jeevaveni S  
Email: 241501076@rajalakshmi.edu.in  
Roll no: 241501076  
Phone: 9342214985  
Branch: REC  
Department: I AIML AD  
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

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
start_time = input()
end_time = input()
try:
    start_dt = datetime.strptime(start_time, '%Y-%m-%d %H:%M:%S')
    end_dt = datetime.strptime(end_time, '%Y-%m-%d %H:%M:%S')
    print(start_time)
    print(end_time)
except:
    print("Event time is not in the format")
```

**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 NumberFormatException(Exception):
    pass
class NoSuchElementException(Exception):
    pass
reg_num = input()
mob_num = input()
try:
    if len(reg_num) != 9:
        raise IllegalArgumentException("Register Number should have exactly 9
characters.")
```

```

if len(mob_num) != 10:
    raise IllegalArgumentException("Mobile Number should have exactly 10
characters.")
if not mob_num.isdigit():
    raise NumberFormatException("Mobile Number should only contain digits.")
if not (reg_num[:2].isdigit() and reg_num[2:5].isalpha() and
reg_num[5:].isdigit()):
    raise IllegalArgumentException("Register Number should have the format: 2
numbers, 3 characters, and 4 numbers.")
if not reg_num.isalnum():
    raise NoSuchElementException("Register Number should only contain digits
and alphabets.")
print("Valid")
except Exception as e:
    print("Invalid with exception message:", e)

```

**Status :** Correct

**Marks :** 10/10

### 3. 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()
    if name == 'q':
        break
    names.append(name)
with open("sorted_names.txt", "w") as file:
    for name in names:
        file.write(name + "\n")
names.sort()
for name in names:
    print(name)
```

**Status :** Correct

**Marks : 10/10**

## **4. 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!")
    exit()
items = list(map(int, input().split()))
M = int(input())
```

```
with open("sales.txt", "w") as f:  
    for i in range(N):  
        f.write(str(items[i] * M) + "\n")  
with open("sales.txt", "r") as f:  
    for line in f:  
        print(line.strip())
```

**Status :** Correct

**Marks :** 10/10