Rajalakshmi Engineering College

Name: Mithran VT
Email: 240701312@rajalakshmi.edu.in
Roll no:
Phone: 9952919350
Branch: REC
Department: I CSE FC
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

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

```
# You are using Python
def save_and_sort_names(filename):
    try:
        names = []

        while True:
            name = input().strip()
            if name.lower() == 'q':
                break
            if 3 <= len(name) <= 30:
                names.append(name)
            else:
                print("Error: Name must be between 3 and 30 characters.")

        if len(names) < 3 or len(names) > 20:
            print("Error: Number of names must be between 3 and 20.")
            return

        names.sort()

        with open(filename, 'w') as file:
            for name in names:
                file.write(name + '\n')
```

```
        # Display sorted names without extra text
        for name in names:
            print(name)

    except Exception as e:
        print(f"Error: {e}")

# Run the function
save_and_sort_names("sorted_names.txt")
```

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

2. 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 validate_datetime_format(date_time_str):
    """Function to validate date-time format."""
    try:
        datetime.strptime(date_time_str, '%Y-%m-%d %H:%M:%S')
        return True
    except ValueError:
        return False

def get_event_times():
    """Function to obtain and validate start and end times."""
    start_time = input().strip()
    end_time = input().strip()

    if validate_datetime_format(start_time) and
validate_datetime_format(end_time):
        print(start_time)
        print(end_time)
    else:
        print("Event time is not in the format")

# Run the function
get_event_times()
```

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

3. 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
import re

class IllegalArgumentException(Exception):
    """Raised when the input does not meet the required format."""
    pass
```

```
class NumberFormatException(Exception):
    """Raised when a mobile number contains non-digit characters."""
    pass

class NoSuchElementException(Exception):
    """Raised when a register number contains invalid characters."""
    pass

def validate_register_number(register_number):
    """Validate register number format (2 numbers, 3 letters, 4 numbers)."""
    if len(register_number) != 9:
        raise IllegalArgumentException("Register Number should have exactly 9
characters.")

    if not re.match(r"^\d{2}[A-Za-z]{3}\d{4}$", register_number):
        raise IllegalArgumentException("Register Number should have the format: 2
numbers, 3 characters, and 4 numbers.")

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

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

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

def main():
    try:
        register_number = input().strip()
        mobile_number = input().strip()

        validate_register_number(register_number)
        validate_mobile_number(mobile_number)

        print("Valid")

    except (IllegalArgumentException, NumberFormatException,
```

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

# Run the program
main()
```

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

4. 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):
    """Function to check if three sides form a valid triangle."""
    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:
        print("It's a valid triangle")
    else:
        print("It's not a valid triangle")

def main():
    try:
        side1 = int(input().strip())
        side2 = int(input().strip())
        side3 = int(input().strip())

        is_valid_triangle(side1, side2, side3)

    except ValueError as e:
        print(f"ValueError: {e}")

# Run the program
main()
```

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