Rajalakshmi Engineering College

Name: talapaneni sujitha
Email: 240701551@rajalakshmi.edu.in
Roll no: 2116240701551
Phone: 9445179891
Branch: REC
Department: I CSE FF
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

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

```
# You are using Python
# 1. Read the input string
input_string = input()

# 2. Calculate character frequencies
# Use a dictionary to store character counts
char_counts = {}
for char in input_string:
    # If the character is already in the dictionary, increment its count
    if char in char_counts:
        char_counts[char] += 1
    # If it's a new character, add it to the dictionary with a count of 1
    else:
        char_counts[char] = 1

# Prepare the output content for both console and file
output_lines = ["Character Frequencies:"]
for char, count in char_counts.items():
    output_lines.append(f"{char}: {count}")

# Convert the list of output lines to a single string with newlines
output_content = "\n".join(output_lines)

# 3. Print the frequencies to the console
print(output_content)

# 4. Save the frequencies to a file named "char_frequency.txt"
```

```
file_name = "char_frequency.txt"
with open(file_name, 'w') as f:
    f.write(output_content)
```

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

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

# Flag to track if any input is in the wrong format
is_format_correct = True

# Read and parse start time
start_time_str = input()
try:
    # Attempt to parse the start time string
    datetime.strptime(start_time_str, DATE_TIME_FORMAT)
except ValueError:
    # If parsing fails, set the flag to False
    is_format_correct = False

# Read and parse end time
end_time_str = input()
try:
    # Attempt to parse the end time string
    datetime.strptime(end_time_str, DATE_TIME_FORMAT)
except ValueError:
    # If parsing fails, set the flag to False
    is_format_correct = False

# Determine output based on the flag
if is_format_correct:
    print(start_time_str)
    print(end_time_str)
else:
    print("Event time is not in the format")
```

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

3. 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(side1, side2, side3):
    """
    Checks if three given side lengths can form a valid triangle.
    Raises ValueError if any side length is not a positive value.
    """
    # Check if all side lengths are positive
    if side1 <= 0 or side2 <= 0 or side3 <= 0:
```

```
        raise ValueError("Side lengths must be positive")

    # Check the triangle inequality theorem
    # The sum of any two sides must be greater than the third side
    if (side1 + side2 > side3) and \
       (side1 + side3 > side2) and \
       (side2 + side3 > side1):
        return True
    else:
        return False

# Main part of the program
if __name__ == "__main__":
    try:
        # Read the three side lengths from input
        A = int(input())
        B = int(input())
        C = int(input())

        # Call the function and print the result
        if is_valid_triangle(A, B, C):
            print("It's a valid triangle")
        else:
            print("It's not a valid triangle")

    except ValueError as e:
        # Catch ValueError raised by is_valid_triangle or by int(input())
        # The problem statement implies ValueError for non-positive sides from the
function
        # and not for non-integer inputs. Assuming valid integer inputs.
        print(f"ValueError: {e}")
```

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

```
# You are using Python
# Define the maximum allowed days
MAX_DAYS = 30
FILE_NAME = "sales.txt"

# 1. Read Inputs
N = int(input())
items_sold_str = input().split()
M = int(input())

# Convert items sold to a list of integers
items_sold_per_day = [int(item) for item in items_sold_str]

# 2. Check for Limit
if N > MAX_DAYS:
    print("Exceeding limit!")
else:
    # 3. Calculate Daily Earnings and Write to File
    daily_earnings = []
    for items in items_sold_per_day:
        earnings = items * M
        daily_earnings.append(str(earnings)) # Convert to string for writing to file

    # Write daily earnings to sales.txt
    # 'w' mode truncates the file if it exists or creates it if it doesn't
    with open(FILE_NAME, 'w') as file:
        file.write("\n".join(daily_earnings))

    # 4. Read from File and Display
    # 'r' mode opens the file for reading
    with open(FILE_NAME, 'r') as file:
        # Read all lines from the file
        content = file.read()
        # Print the content directly, which already contains newlines
        print(content.strip()) # .strip() removes any trailing newline if present, to
match exact output
```

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