

Rajalakshmi Engineering College

Name: CHENTHAN AMUTHAN.D
Email: 240701090@rajalakshmi.edu.in
Roll no:
Phone: null
Branch: REC
Department: I CSE FA
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
from collections import OrderedDict
input_string = input()
char_count = OrderedDict()

for char in input_string:
    if char in char_count:
        char_count[char] += 1
    else:
        char_count[char] = 1

with open("char_frequency.txt", "w") as file:
    file.write("Character Frequencies:\n")
    for char, count in char_count.items():
        file.write(f"{char}: {count}\n")

print("Character Frequencies:")
for char, count in char_count.items():
    print(f"{char}: {count}")
```

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
datetime_format = "%Y-%m-%d %H:%M:%S"
```

try:

```
    start_time = input().strip()
```

```
end_time = input().strip()
```

```
start_dt = datetime.strptime(start_time, datetime_format)
end_dt = datetime.strptime(end_time, datetime_format)
```

```
print(start_time)
print(end_time)
```

```
except ValueError:
```

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

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
```

```
MAX_DAYS = 30
```

```
FILE_NAME = "sales.txt"
```

```
N = int(input())
```

```
if N > MAX_DAYS:
```

```
    print("Exceeding limit!")
```

```
else:
```

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

```
    M = int(input())
```

```
with open(FILE_NAME, "w") as file:
```

```
    for sold in items_sold:
```

```
        earning = sold * M
```

```
        file.write(str(earning) + "\n")
```

```
with open(FILE_NAME, "r") as file:
```

```
for line in file:  
    print(line.strip())
```

Status : Correct

Marks : 10/10

4. Problem Statement

In the enchanted realm of Academia, you, the Academic Alchemist, are bestowed with a magical quill and a parchment to weave the grades of aspiring students into a tapestry of academic brilliance.

The mission is to craft a Python program that empowers faculty members to enter student grades for any two subjects, stores these magical grades in a mystical file, and then, with a wave of your virtual wand, calculates the GPA to unveil the true essence of academic achievement.

Input Format

The input format is a string representing the student's name, any two subjects, and corresponding grades.

After entering grades, they can type 'done' when prompted for the student's name.

Output Format

The output should display the (average of grades) calculated GPA with a precision of two decimal places.

The magical grades will be saved in a mystical file named "magical_grades.txt".

Refer to the sample output for format specifications.

Sample Test Case

Input: Alice
Math
95
English

88

done

Output: 91.50

Answer

You are using Python

with open("magical_grades.txt", "w") as file:

while True:

name = input()

if name.lower() == 'done':

break

subject1 = input()

grade1 = int(input())

subject2 = input()

grade2 = int(input())

file.write(f"{name},{subject1},{grade1},{subject2},{grade2}\n")

gpa = (grade1 + grade2) / 2

print(f"{gpa:.2f}")

Status : Correct

Marks : 10/10