

# Rajalakshmi Engineering College

Name: NEIL DANIEL A  
Email: 240701356@rajalakshmi.edu.in  
Roll no: 240701356  
Phone: 8925059757  
Branch: REC  
Department: I CSE FD  
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

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**

```
text=[]
flag=True
while(flag):
    str1=input()
    if str1=="done":
        flag=False
    else:
        text.append(str1)
num=[]
for i in range(len(text)):
    try:
        num.append(int(text[i]))
    except ValueError:
        pass
avg=sum(num)/len(num)
print(f"{avg:.2f}")
```

**Status : Correct**

**Marks : 10/10**

## 2. Problem Statement

Alex is creating an account and needs to set up a password. The program prompts Alex to enter their name, mobile number, chosen username, and desired password. Password validation criteria include:

Length between 10 and 20 characters. At least one digit. At least one special character from !@#\$%^&\* set. Display "Valid Password" if criteria are met; otherwise, raise an exception with an appropriate error message.

### ***Input Format***

The first line of the input consists of the name as a string.

The second line of the input consists of the mobile number as a string.

The third line of the input consists of the username as a string.

The fourth line of the input consists of the password as a string.

### ***Output Format***

If the password is valid (meets all the criteria), it will print "Valid Password"

If the password is weak (fails any one or more criteria), it will print an error message accordingly.

Refer to the sample outputs for the formatting specifications.

### ***Sample Test Case***

Input: John

9874563210

john

john1#nhøj

Output: Valid Password

### ***Answer***

```
def validate_password(password):  
    if len(password) < 10 or len(password) > 20:
```

```
return "Should be a minimum of 10 characters and a maximum of 20
characters"
```

```
if not any(char.isdigit() for char in password):
    return "Should contain at least one digit"
```

```
special_characters = {'!','@','#','$','%','^','&','*'}
if not any(char in special_characters for char in password):
    return "It should contain at least one special character"
```

```
return "Valid Password"
```

```
name = input().strip()
mobile_number = input().strip()
username = input().strip()
password = input().strip()

result = validate_password(password)
print(result)
```

**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**

```
import sys
days=int(input())
if(days>30):
    print("Exceeding limit!")
    sys.exit()
else:
    sales=[]
    sales_str=input().strip().split()
    rate=int(input())
    for i in sales_str:
        sales.append(rate * int(i))
```

```
for x in sales:  
    print(x)
```

**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**

```
import sys
def is_valid_triangle(a,b,c):
    if a+b > c and b+c >a and a+c>b:
        print ("It's a valid triangle")
    else:
        print("It's not a valid triangle")
a=int(input())
b=int(input())
c=int(input())
if a<1 or b<1 or c<1:
    print("ValueError: Side lengths must be positive")
    sys.exit()
else:
    is_valid_triangle(a,b,c)
```

**Status :** Correct

**Marks :** 10/10