

Rajalakshmi Engineering College

Name: aakash velan

Email: 241501001@rajalakshmi.edu.in

Roll no:

Phone: 7358618861

Branch: REC

Department: I AI & ML FA

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

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

```
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

2. 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
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

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

```
class IllegalArgumentException(Exception):  
    pass
```

```
class NoSuchElementException(Exception):  
    pass
```

```
register_number = input().strip()  
mobile_number = input().strip()
```

```
try:
```

```
    # Check length of Register Number  
    if len(register_number) != 9:  
        raise IllegalArgumentException("Register Number should have exactly 9  
characters.")
```

```
    # Check Register Number format: 2 digits + 3 letters + 4 digits
```

```
    if not (register_number[:2].isdigit() and  
            register_number[2:5].isalpha() and  
            register_number[5:].isdigit()):
```

```
        raise IllegalArgumentException("Register Number should have the format: 2
```

```

numbers, 3 characters, and 4 numbers.")

# Check for only alphanumeric characters
if not register_number.isalnum():
    raise NoSuchElementException("Register Number should contain only
letters and digits.")

# Check length of Mobile Number
if len(mobile_number) != 10:
    raise IllegalArgumentException("Mobile Number should have exactly 10
characters.")

# Check Mobile Number contains only digits
if not mobile_number.isdigit():
    raise ValueError("Mobile Number should only contain digits.")

print("Valid")

except (IllegalArgumentException, ValueError, NoSuchElementException) as e:
    print(f"Invalid with exception message: {e}")

```

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
```

```
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