Rajalakshmi Engineering College

Name: VISHNU D
Email: 240701601@rajalakshmi.edu.in
Roll no: 2116240701601
Phone: null
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

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.

2116240701601

2116240701601

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

```
import re
def validate_register_number(reg_num):
    if len(reg_num) != 9:
        raise IllegalArgumentException("Register Number should have exactly 9 characters.")
    if not re.match(r'^\d{2}[A-Za-z]{3}\d{4}$', reg_num):
        raise IllegalArgumentException("Register Number should have the format: 2 numbers, 3 characters, and 4 numbers.")
    if not reg_num.isalnum():
        raise NoSuchElementException("Register Number should contain only digits and alphabets.")
def validate_mobile_number(mobile):
    if len(mobile) != 10:
        raise IllegalArgumentException("Mobile Number should have exactly 10 characters.")
    if not mobile.isdigit():
        raise NumberFormatException("Mobile Number should only contain digits.")
class IllegalArgumentException(Exception):
    pass
```

```
class NoSuchElementException(Exception):
   pass

class NumberFormatException(Exception):
   pass
try:
   reg_num = input().strip()
   mobile = input().strip()
   validate_register_number(reg_num)
   validate_mobile_number(mobile)
   print("Valid")
except (IllegalArgumentException, NoSuchElementException,
NumberFormatException) as e:
   print(f"Invalid with exception message: {e}")
```

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

2. 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".

2116240701601

Refer to the sample output for format specifications.

Sample Test Case

Input: Alice
Math
95
English
88
done
Output: 91.50

Answer

```
f=open("magical_grades.txt", "w")
while True:
   n=input()
   if n=="done":
      break
   s1=input()
   g1=int(input())
   s2=input()
   g2=int(input())
   a=(g1+g2)/2
   f.write(f"{n},{s1},{g1},{s2},{g2}\n")
   print(f"{a:.02f}")
   f.close()
```

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

3. 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 !@#$%^&amp;* set. Display "Valid Password" if

2116240701601

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#nhoj
Output: Valid Password

Answer

```
class l(Exception):
   pass
class d(Exception):
   pass
class c(Exception):
   pass
n=input()
p=int(input())
u=input()
try:
```

2116240701601

```
    pa=input().strip()
    da,s=0,0
    for i in pa:
        if i.isdigit():
            da+=1
        if i.isalnum():
            s+=1
    if len(pa) <10 or len(pa)>=20:
        raise l
    if not any(ch.isdigit() for ch in pa):
        raise d
    if pa.isalnum():
        raise c
    print("Valid Password")
except l:
    print("Should be a minimum of 10 characters and a maximum of 20
characters")
except d:
    print("Should contain at least one digit")
except c:
    print("It should contain at least one special character")
```

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

4. 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'.

2116240701601

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

```
with open("sorted_named.txt","w") as f:
    while True:
        e=input()
        if e=='q':
            break
        f.write(e.strip()+'\n')
with open("sorted_named.txt","r") as f:
    l=list(line for line in f)
    l.sort()
    for i in l:
        print(i,end='')
```

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

2116240701601