Rajalakshmi Engineering College

Name: Saii Anish R
Email: 241501175@rajalakshmi.edu.in
Roll no: 241501175
Phone: 8438920387
Branch: REC
Department: I AI & ML FB
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

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'.

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

```
names_list = []
while True:
    name_input = input()
    if name_input.lower() == 'q':
        break
    names_list.append(name_input)

file_name = "sorted_names.txt"
try:
    with open(file_name, 'w') as f:
        for name in names_list:
            f.write(name + "\n")
except IOError:

    pass

names_list.sort()

for name in names_list:
    print(name)
```

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

2. 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

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

```
class NumberFormatException(Exception):
    pass

class NoSuchElementException(Exception):
    pass

def validate_register_number(reg_num):
    if len(reg_num) != 9:
        raise IllegalArgumentException("Register Number should have exactly 9
characters.")

    if not reg_num.isalnum():
        raise NoSuchElementException("Register Number should only contain digits
and alphabets.")

    if not (reg_num[0:2].isdigit() and \
            reg_num[2:5].isalpha() and \
            reg_num[5:9].isdigit()):
        raise IllegalArgumentException("Register Number should have the format: 2
numbers, 3 characters, and 4 numbers.")

def validate_mobile_number(mobile_num):
    if len(mobile_num) != 10:
        raise IllegalArgumentException("Mobile Number should have exactly 10
characters.")

    if not mobile_num.isdigit():
        raise NumberFormatException("Mobile Number should only contain digits.")

try:
    register_number_input = input()
    mobile_number_input = input()

    validate_register_number(register_number_input)
    validate_mobile_number(mobile_number_input)

    print("Valid")

except IllegalArgumentException as e:
    print(f"Invalid with exception message: {e}")
except NumberFormatException as e:
    print(f"Invalid with exception message: {e}")
```

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

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

3. 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

```
all_grades_numeric = []
file_name = "magical_grades.txt"

with open(file_name, 'w') as f:
    while True:
        student_name = input()
        if student_name.lower() == 'done':
            break

        subject1_name = input()
        grade1_str = input()
        grade1 = float(grade1_str)

        subject2_name = input()
        grade2_str = input()
        grade2 = float(grade2_str)

        all_grades_numeric.append(grade1)
        all_grades_numeric.append(grade2)

        f.write(student_name + "\n")
        f.write(subject1_name + "\n")
        f.write(grade1_str + "\n")
        f.write(subject2_name + "\n")
        f.write(grade2_str + "\n")

gpa = 0.0
if all_grades_numeric:
    total_sum_of_grades = sum(all_grades_numeric)
    count_of_grades = len(all_grades_numeric)
    gpa = total_sum_of_grades / count_of_grades

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

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

4. 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

```
program_data_container = [input()]
program_data_container.append({})

for char_iterator in program_data_container[0]:
    program_data_container[1][char_iterator] =
program_data_container[1].get(char_iterator, 0) + 1

program_data_container.append("char_frequency.txt")

with open(program_data_container[2], 'w') as file_handle:
```

```
    for key_char, value_count in program_data_container[1].items():
        file_handle.write(f"{key_char}: {value_count}\n")

print("Character Frequencies:")
for key_char, value_count in program_data_container[1].items():
    print(f"{key_char}: {value_count}")
```

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