

# Rajalakshmi Engineering College

Name: Akshaya Kirubakarraj  
Email: 241501016@rajalakshmi.edu.in  
Roll no: 241501016  
Phone: 9940136549  
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

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  
import re

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

```
class NumberFormatException(Exception):  
    pass
```

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

```
def validate_register_number(reg_no):  
    if len(reg_no) != 9:  
        raise IllegalArgumentException("Register Number should have exactly 9  
characters.")  
    if not re.match(r"^\d{2}[A-Za-z]{3}\d{4}$", reg_no):  
        raise IllegalArgumentException("Register Number should have the format: 2  
numbers, 3 characters, and 4 numbers.")  
    if not reg_no.isalnum():
```

```

        raise NoSuchElementException("Register Number should only contain
alphabets and digits.")

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.")

try:
    reg_no = input().strip()
    mobile = input().strip()
    validate_register_number(reg_no)
    validate_mobile_number(mobile)

    print("Valid")
except Exception as e:
    print(f"Invalid with exception message: {e}")

```

**Status :** Correct

**Marks :** 10/10

## 2. 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

```
text = input()
```

```
frequency = {}
```

```
for char in text:
```

```
    frequency[char] = frequency.get(char, 0) + 1
```

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

```
    file.write("Character Frequencies:\n")
```

```
    for char, count in frequency.items():
```

```
        file.write(f"{char}: {count}\n")
```

```
print("Character Frequencies:")
```

```
for char, count in frequency.items():
```

```
    print(f"{char}: {count}", end=' ')
```

**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 !@#\$%^&\* 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***

```
# You are using Python
name = input()
mobile = input()
username = input()
password = input()
```

```
try:
```

```
    if len(password) < 10 or len(password) > 20:
        raise Exception("Should be a minimum of 10 characters and a maximum of
        20 characters")
```

```
if not any(char.isdigit() for char in password):
    raise Exception("Should contain at least one digit")
if not any(char in "!@#$%^&*" for char in password):
    raise Exception("It should contain at least one special character")
print("Valid Password")
except Exception as e:
    print(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

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**