Student: **kamali rj** Email id: **240701225@rajalakshmi.edu.in** Test: **REC_Python_Week 6_CY** Course: **NeoColab_REC_CS23221_Python Programming**

IP Address: 115.245.95.250, 2409:40f4:411d:fab9:4ca5:9... Tab Switches: --
OS Used: Windows Browser Used: Firefox
Test Duration: 00:16:20 Test Start Time: May 20, 2025 | 06:20 PM
Test Submit Time: May 23, 2025 | 09:19 PM Resume Count: 2

Summary Sections Filters

1 Coding (4)

Question No: 1

Single File Programming Question

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.

Code constraints :

In this scenario, the given test cases will fall under the following constraints:
1 ≤ side1,side2, side3 ≤ 1000

Sample test cases :

Input 1 :

```
3
4
5
```

Output 1 :

```
It's a valid triangle
```

Input 2 :

```
1
1
3
```

Output 2 :

```
It's not a valid triangle
```

Input 3 :

```
4
-5
6
```

Output 3 :

```
ValueError: Side lengths must be positive
```

Input 4 :

```
5
5
0
```

Output 4 :

```
ValueError: Side lengths must be positive
```

Fill your code here

Python

```python
def is_valid_triangle(a, b, c):
    """Check if the given sides form a valid triangle."""
    if a <= 0 or b <= 0 or c <= 0:
        raise ValueError("Side lengths must be positive")

    if a + b > c and a + c > b and b + c > a:
        return "It's a valid triangle"
    else:
        return "It's not a valid triangle"

try:
    a = int(input())
    b = int(input())
    c = int(input())

    print(is_valid_triangle(a, b, c))

except ValueError as e:
    print(f"ValueError: {e}")

```

Status **Correct** | Mark obtained **10/10** | Hints used **0** | Times compiled **2** | Times submitted **2** | Level **Medium** | Question type **Single File Programming** | Subject **Python**
Topic **Exception Handling** | Sub Topic **try except** | Blooms taxonomy **Apply**

☐ Show testcase scores ☐ Show solution

Question No: 2

Single File Programming Question

Problem Statement

Write a program to read the Register Number and Mobile Number of a student. Create user-defined exception and handle the following:

1. 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**.
2. If the Mobile Number contains any character other than a digit, raise a **NumberFormatException**.
3. If the Register Number contains any character other than digits and alphabets, throw a **NoSuchElementException**.
4. 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.

Code constraints :

In this scenario, the given test cases will fall under the following constraints:
The register number should have exactly 9 characters: 2 digits, 3 letters, and 4 digits.
The mobile number should have exactly 10 digits.

Sample test cases :

Input 1 :

```
19ABC1001
9949596920
```

Output 1 :

```
Valid
```

Input 2 :

```
19ABC1001
99495969209
```

Output 2 :

```
Invalid with exception message: Mobile Number should have exactly 10 characters.
```

Input 3 :

```
19ABC10019
9949596920
```

Output 3 :

```
Invalid with exception message: Register Number should have exactly 9 characters.
```

Input 4 :

```
195AC1001
9949596920
```

Output 4 :

```
Invalid with exception message: Register Number should have the format: 2 numbers, 3 character
```

Input 5 :

```
19ABC1001
994C596920
```

Output 5 :

```
Invalid with exception message: Mobile Number should only contain digits.
```

Fill your code here Python

```python
import re

class IllegalArgumentException(Exception):
    """Exception raised for invalid length or format issues."""
    pass

class NumberFormatException(Exception):
    """Exception raised when mobile number contains non-digit characters."""
    pass

class NoSuchElementException(Exception):
    """Exception raised when register number contains invalid characters."""
    pass

def validate_register_number(register_number):
    """Validate the register number format."""
    if len(register_number) != 9:
        raise IllegalArgumentException("Register Number should have exactly 9 characters.")

    if not re.match(r"^\d{2}[A-Za-z]{3}\d{4}$", register_number):
        raise IllegalArgumentException("Register Number should have the format: 2 numbers, 3 characters, and 4 numbers.")

    if not register_number.isalnum():
```

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

def validate_mobile_number(mobile_number):
    """Validate the mobile number format."""
    if len(mobile_number) != 10:
        raise IllegalArgumentException("Mobile Number should have exactly 10 characters.")

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

Status **Correct** | Mark obtained **10/10** | Hints used **0** | Times compiled **2** | Times submitted **2** | Level **Hard** | Question type **Single File Programming** | Subject **Python**
Topic **Exception Handling** | Sub Topic **User Defined Exception Handling** | Blooms taxonomy **Apply**

☐ Show testcase scores ☐ Show solution

Question No: 3

Single File Programming Question

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.

Code constraints :

The given test cases fall under the following constraints:
The grades should be from 0 to 100.

Sample test cases :

Input 1 :

```
Alice
Math
95
English
88
done
```

Output 1 :

```
91.50
```

Input 2 :

```
David
Literature
78
Spanish
92
done
```

Output 2 :

```
85.00
```

Fill your code here Python ...

```
def save_grades(student_data):
    """Save grades to the mystical file."""
    with open("magical_grades.txt", "w") as file:
        for student, subjects in student_data.items():
            file.write(f"{student}: {subjects}\n")

def calculate_gpa(grades):
    """Calculate GPA with enchanted precision."""
    return round(sum(grades) / len(grades), 2)

# Begin the academic ritual
student_data = {}

while True:
    student_name = input().strip()
    if student_name.lower() == "done":
        break

    subjects = {}
    for _ in range(2):  # Two subjects per student
        subject = input().strip()
        grade = int(input().strip())

        # Ensure grades are within the magical realm's limits
        if not (0 <= grade <= 100):
            print("Invalid grade! Please enter a number between 0 and 100.")
            continue

        subjects[subject] = grade

    student_data[student_name] = subjects
    gpa = calculate_gpa(list(subjects.values()))
    print(f"{gpa: .2f}")
```

Status **Correct** | Mark obtained **10/10** | Hints used **0** | Times compiled **20** | Times submitted **1** | Level **Hard** | Question type **Single File Programming** | Subject **Programming** | Topic **Python**
Sub Topic **File operation** | Blooms taxonomy **Apply**

Show testcase scores Show solution

Question No: 4

Single File Programming Question

Problem Statement

Write a program to obtain the start time and end time for the stage event show. If the user enters a different format other than specified, an exception occurs and the program is interrupted. To avoid that, handle the exception and prompt the user to enter the right format as specified.

- Start time and end time should be in the format **'YYYY-MM-DD HH:MM:SS'**
- If the input is in the above format, print the start time and end time.
- If the input does not follow the above format, print "**Event time is not in the format** "

Input format :

The first line of input consists of the start time of the event.
The second line of the input consists of the end time of the event.

Output format :

If the input is in the given format, print the start time and end time.
If the input does not follow the given format, print "**Event time is not in the format".**

Refer to the sample output for formatting specifications.

Code constraints :

The given test cases fall under the following constraints:
The input format: **YYYY-MM-DD HH:MM:SS**

Sample test cases :

Input 1 :

```
2022-01-12 06:10:00
2022-02-12 10:10:12
```

Output 1 :

```
2022-01-12 06:10:00
2022-02-12 10:10:12
```

Input 2 :

```
2022-01-12 06:10:00
2022-02-12 10:00:
```

Output 2 :

```
Event time is not in the format
```

Input 3 :

```
2022-01-12 06:10:00
2022-02-31 10:10:12
```

Output 3 :

```
Event time is not in the format
```

Input 4 :

```
2022-01-12 10:75:00
2022-02-12 10:10:80
```

Output 4 :

```
Event time is not in the format
```

Fill your code here Python _

```python
from datetime import datetime

def validate_datetime_format(date_str):
    """Validate if the given date string is in 'YYYY-MM-DD HH:MM:SS' format."""
    try:
        valid_date = datetime.strptime(date_str, "%Y-%m-%d %H:%M:%S")
        return valid_date
    except ValueError:
        return None

start_time = input().strip()
end_time = input().strip()

valid_start = validate_datetime_format(start_time)
valid_end = validate_datetime_format(end_time)

if valid_start and valid_end:
    print(start_time)
    print(end_time)
else:
    print("Event time is not in the format")

```

Status **Correct** | Mark obtained **10/10** | Hints used **0** | Times compiled **2** | Times submitted **2** | Level **Medium** | Question type **Single File Programming** | Subject **Python**
Topic **Exception Handling** | Sub Topic **In Built Exception Handling** | Blooms taxonomy **Apply**

Show testcase scores Show solution

Student: **kamali rj** Email id: **240701225@rajalakshmi.edu.in** Test: **REC_Python_Week 6_PAH** Course: **NeoColab_REC_CS23221_Python Programming**

IP Address: 115.245.95.250, 2409:40f4:411d:fab9:4ca5:9... Tab Switches: 1 OS Used: Windows Browser Used: Firefox

Test Duration: 00:32:20 Test Start Time: May 20, 2025 | 06:38 PM Test Submit Time: May 23, 2025 | 09:40 PM Resume Count: 5

Summary Sections Filters

1 Coding (3)

Question No: 1

Single File Programming Question

Problem Statement

Peter manages a student database and needs a program to add students. For each student, Alex inputs their ID and name. The program checks for duplicate IDs and ensures the database isn't full.

If a duplicate or a full database is detected, an appropriate error message is displayed. Otherwise, the student is added, and a confirmation message is shown. The database has a maximum capacity of 30 students, and each student must have a unique ID.

Input format :

The first line contains an integer n, representing the number of students to be added to the school database.
The next n lines each contain two space-separated values, representing the student's ID (integer) and the student's name (string).

Output format :

The output will depend on the actions performed in the code.

If a student is added to the database, the output will display: "Student with ID [ID number] added to the database."
If there is an exception due to a duplicate student ID, the output will display: "Exception caught. Error: Student ID already exists."
If there is an exception due to the database being full, the output will display: "Exception caught. Error: Student database is full."

Refer to the sample outputs for the formatting specifications.

Code constraints :

The given test case will fall under the following constraints:
MAX_CAPACITY = 30.
1 ≤ n ≤ MAX_CAPACITY.
1 ≤ student ID ≤ 10.
1 ≤ length of student's name ≤ 100
Each student ID must be a positive integer.

Sample test cases :

Input 1 :

```
3
16 Sam
87 Sabari
43 Dani
```

Output 1 :

```
Student with ID 16 added to the database.
Student with ID 87 added to the database.
Student with ID 43 added to the database.
```

Input 2 :

```
3
44 Udhesh
33 Sandy
44 Keerthi
```

Output 2 :

```
Student with ID 44 added to the database.
Student with ID 33 added to the database.
Exception caught. Error: Student ID already exists.
```

Input 3 :

```
32
32 zen
23 jazz
16 sam
87 santhiya
43 dominic
90 felicia
17 tera
85 wednesday
81 sayari
39 danny
55 udhesh
36 nani
21 cheenu
12 Sakshi
49 madhan
33 bons
41 Ambika
30 Sandy
47 Charu
59 Theju
34 Sabari
56 Udhesh
40 Babu
42 Sandeep
```

Output 3 :

```
Student with ID 32 added to the database.
Student with ID 23 added to the database.
Student with ID 16 added to the database.
Student with ID 87 added to the database.
Student with ID 43 added to the database.
Student with ID 90 added to the database.
Student with ID 17 added to the database.
Student with ID 85 added to the database.
Student with ID 81 added to the database.
Student with ID 39 added to the database.
Student with ID 55 added to the database.
Student with ID 36 added to the database.
Student with ID 21 added to the database.
Student with ID 12 added to the database.
Student with ID 49 added to the database.
Student with ID 33 added to the database.
Student with ID 41 added to the database.
Student with ID 30 added to the database.
Student with ID 47 added to the database.
Student with ID 59 added to the database.
Student with ID 34 added to the database.
Student with ID 56 added to the database.
Student with ID 40 added to the database.
Student with ID 42 added to the database.
Student with ID 102 added to the database.
```

```
102 nancy
26 saxy
13 doll
11 craven
211 kanaga
94 veronic
47 jansi
33 yalini
```

```
Student with ID 26 added to the database.
Student with ID 13 added to the database.
Student with ID 11 added to the database.
Student with ID 211 added to the database.
Student with ID 94 added to the database.
Exception caught. Error: Student database is full.
```

Fill your code here

Python _

```python
MAX_CAPACITY = 30

n = int(input())
database = set()

for _ in range(n):
    try:
        entry = input().strip()
        if not entry:
            continue
        parts = entry.split(maxsplit=1)
        if len(parts) != 2:
            continue  # Ignore malformed input
        student_id_str, student_name = parts
        student_id = int(student_id_str)

        if len(database) >= MAX_CAPACITY:
            raise Exception("Student database is full.")
        if student_id in database:
            raise Exception("Student ID already exists.")

        database.add(student_id)
        print(f"Student with ID {student_id} added to the database.")
    except Exception as e:
        print(f"Exception caught. Error: {e}")
        break
```

Status **Correct** | Mark obtained **10/10** | Hints used **0** | Times compiled **10** | Times submitted **4** | Level **Hard** | Question type **Single File Programming** | Subject **Python**
Topic **Exception Handling** | Sub Topic **Exception** | Blooms taxonomy **Apply**

☐ Show testcase scores ☐ Show solution

Question No: 2

Single File Programming Question

Problem Statement

John is a data analyst who often works with text files. He needs a program that can analyze the contents of a text file and count the number of times a specific character appears in the file.

John wants a simple program that allows him to specify a file and a character to count within that file.

Input format :

The first line of input consists of the file's name to be analyzed.
The second line of the input consists of the string they want to write within the file.
The third line of the input consists of a character to count within the file.

Output format :

If the character is found, the output displays "The character 'X' appears {Y} times in the file." where X is the character and Y i the count,

If the character does not appear in the file, the output displays "Character not found."

Refer to the sample output for the formatting specifications.

Code constraints :

The given test cases fall under the following constraints:
The character is to be counted as a single character (alphabetic, numeric, or special character).
5 ≤ length of the input string ≤ 500

Sample test cases :

Input 1 :

```
test.txt
This is a test file to check the character count.
e
```

Output 1 :

```
The character 'e' appears 5 times in the file.
```

Input 2 :

```
sample.txt
Lorem ipsum dolor sit amet, consectetur adipiscing elit.
L
```

Output 2 :

```
The character 'L' appears 3 times in the file.
```

Input 3 :

```
data.txt
Some random content with no special character.
$
```

Output 3 :

```
Character not found in the file.
```

Input 4 :

```
document.txt
This document contains multiple spaces   and a character count.
```

Output 4 :

```
The character ' ' appears 10 times in the file.
```

Fill your code here Python ...

```python

filename = input().strip()
text_to_write = input().lower()
char_to_count = input().lower()
with open(filename, 'w', encoding='utf-8') as file:
    file.write(text_to_write)
with open(filename, 'r', encoding='utf-8') as file:
    content = file.read()
    count = 0
    for i in content:
        if i==char_to_count:
            count+=1

if count > 0:
    print(f"The character '{char_to_count}' appears {count} times in the file.")
else:
    print("Character not found in the file.")

```

Status **Correct** | Mark obtained **10/10** | Hints used **0** | Times compiled **14** | Times submitted **1** | Level **Hard** | Question type **Single File Programming** | Subject **Python** | Topic **File Handling** | Sub Topic **File Handling** | Blooms taxonomy **Apply**

☐ Show testcase scores ☐ Show solution

Question No: 3

Single File Programming Question

Problem Statement

Reeta is playing with numbers. Reeta wants to have a file containing a list of numbers, and she needs to find the average of those numbers. Write a program to read the numbers from the file, calculate the average, and display it.

File Name: user_input.txt

Input format :

The input file will contain a single line of space-separated numbers (as a string).
These numbers may be integers or decimals.

Output format :

If all inputs are valid numbers, the output should print: "Average of the numbers is: X.XX" (where X.XX is the computed average rounded to two decimal places)
If the input contains invalid data, print: "Invalid data in the input."

Refer to the sample output for format specifications.

Code constraints :

The given test cases fall under the following constraints:
The file will have between 1 and 100 numbers ($1 \leq n \leq 100$)

Sample test cases :

Input 1 :

```
1 2 3 4 5
```

Output 1 :

```
Average of the numbers is: 3.00
```

Input 2 :

```
abc 1.1 def 2.2 ghi
```

Output 2 :

```
Invalid data in the input.
```

Input 3 :

```
5 7.9 6.5 9 10
```

Output 3 :

```
Average of the numbers is: 7.68
```

Fill your code here Python ...

```python
'''filename = "user_input.txt"

try:
    # Read the line from the file
    with open(filename, 'r', encoding='utf-8') as file:
        line = file.readline().strip()
        tokens = line.split()

    # Try converting all tokens to float
```

```
    numbers = []
    for token in tokens:
        try:
            number = float(token)
            numbers.append(number)
        except ValueError:
            print("Invalid data in the input.")
            break
    else:
        # Only runs if the loop didn't break (all numbers are valid)
        avg = sum(numbers) / len(numbers)
        print(f"Average of the numbers is: {avg:.2f}")

except Exception:
    print("Invalid data in the input.")'''
b = 1
s = input().split()
for i in s:
    if i.isalpha():
        s.remove(i)
        b = 0
if b:
    avg = 0.0
    for i in s:
```

Status **Partially correct** | Mark obtained **7.5/10** | Hints used **0** | Times compiled **10** | Times submitted **2** | Level **Medium** | Question type **Single File Programming** | Subject **Programming**

Topic **Python** | Sub Topic **File operation** | Blooms taxonomy **Apply**

☐ Show testcase scores ☐ Show solution

Result & Analysis

Attempt 1 of 01

Student: **kamali rj** | Email id: **240701225@rajalakshmi.edu.in** | Test: **REC_Python_Week 6_COD** | Course: **NeoColab_REC_CS23221_Python Programming**

IP Address: 2409:40f4:aa:94b2:30f6:d0a8:4911:3fde | Tab Switches: -- | OS Used: Windows | Browser Used: Firefox

Test Duration: 00:13:09 | Test Start Time: May 20, 2025 | 07:02 PM | Test Submit Time: May 23, 2025 | 11:08 PM | Resume Count: 3

Summary | Sections | Filters

1 Coding (5)

Question No: 1

Single File Programming Question

Problem Statement

Sophie enjoys playing with words and wants to count the number of words in a sentence. She inputs a sentence, saves it to a file, and then reads it from the file to count the words.

Write a program to determine the number of words in the input sentence.

File Name: sentence_file.txt

Input format :

The input consists of a single line of text containing words separated by spaces.

Output format :

The output displays the count of words in the sentence.

Refer to the sample output for the formatting specifications.

Code constraints :

The given test cases fall under the following constraints:
The length of the input string 'S' ≤ 1000

Sample test cases :

Input 1 :

```
Four Words In This Sentence
```

Output 1 :

```
5
```

Input 2 :

```
Word WordWord WordWordWord WordWordWordWord
```

Output 2 :

```
4
```

Input 3 :

Output 3 :

```
0
```

Fill your code here Python ...

```python
# You are using Python
# Step 1: Get user input
sentence = input()

# Step 2: Write the sentence to the file
with open("sentence_file.txt", "w") as file:
    file.write(sentence)

# Step 3: Read the sentence from the file
with open("sentence_file.txt", "r") as file:
    content = file.read()

# Step 4: Count the words
# Split by whitespace and filter out any empty strings
words = content.split()
word_count = len(words)

# Step 5: Print the word count
print(word_count)

```

Status **Correct** | Mark obtained **10/10** | Hints used **0** | Times compiled **3** | Times submitted **1** | Level **Easy** | Question type **Single File Programming** | Subject **Python** | Topic **File Handling**

Sub Topic **File Handling** | Blooms taxonomy **Apply**

Show testcase scores | Show solution

Question No: 2

Single File Programming Question

Problem Statement

Write a program that calculates the average of a list of integers. The program prompts the user to enter the length of the list (n) and each element of the list. It performs error handling to ensure that the length of the list is a non-negative integer and that each input element is a numeric value.

Input format :

The first line of the input is an integer n, representing the length of the list as a positive integer.
The second line of the input consists of an element of the list as an integer, separated by a new line.

Output format :

If the length of the list is not a positive integer or zero, the output displays "Error: The length of the list must be a non-negative integer."
If a non-numeric value is entered for the length of the list, the output displays "Error: You must enter a numeric value."
If a non-numeric value is entered for a list element, the output displays "Error: You must enter a numeric value."

If the inputs are valid, the program calculates and prints the average of the provided list of integers with two decimal places: "The average is: [average]".

Refer to the sample output for the formatting specifications.

Code constraints :

The given test cases fall under the following constraints:
$0 < n \leq 20$
The length of the list and each list element must be integers.

Sample test cases :

Input 1 :

```
-2
1
2
```

Output 1 :

```
Error: The length of the list must be a non-negative integer.
```

Input 2 :

```
3
1
2
3
```

Output 2 :

```
The average is: 2.00
```

Input 3 :

```
3
1
2
a
```

Output 3 :

```
Error: You must enter a numeric value.
```

Input 4 :

```
/
1
3
```

Output 4 :

```
Error: You must enter a numeric value.
```

Input 5 :

```
0
0
```

Output 5 :

```
Error: The length of the list must be a non-negative integer.
```

Fill your code here

Python _

```python
try:
    # Read and validate list length
    n_input = input()
    n = int(n_input)
    if n <= 0:
        print("Error: The length of the list must be a non-negative integer.")
    else:
        numbers = []
        for i in range(n):
            element = input()
            try:
                number = int(element)
                numbers.append(number)
            except ValueError:
                print("Error: You must enter a numeric value.")
                break
        else:
            # All inputs were valid; calculate average
            avg = sum(numbers) / n
            print(f"The average is: {avg:.2f}")
except ValueError:
    print("Error: You must enter a numeric value.")

```

Status **Correct** | Mark obtained **10/10** | Hints used **0** | Times compiled **7** | Times submitted **1** | Level **Medium** | Question type **Single File Programming** | Subject **Python**

Topic **Exception Handling** | Sub Topic **In Built Exception Handling** | Blooms taxonomy **Apply**

Topic: Exception Handling | Sub Topic: In Built Exception Handling | Blooms taxonomy: Apply

Show testcase scores Show solution

Question No: 3

Single File Programming Question

Problem Statement

In a voting system, a person must be at least 18 years old to be eligible to vote. If a user enters an age below 18, the system should raise a user-defined exception indicating that they are not eligible to vote.

Input format :

The input contains a positive integer representing age.

Output format :

If the age is less than 18, the output displays "Not eligible to vote".
Otherwise, the output displays "Eligible to vote".

Refer to the sample output for formatting specifications.

Code constraints :

The given test cases fall under the following constraints:
1 ≤ age ≤ 100

Sample test cases :

Input 1 :

```
18
```

Output 1 :

```
Eligible to vote
```

Input 2 :

```
12
```

Output 2 :

```
Not eligible to vote
```

Fill your code here — Python

```python
# You are using Python
# Define a custom exception
class NotEligibleToVote(Exception):
    pass

# Function to check voting eligibility
def check_voting_eligibility(age):
    if age < 18:
        raise NotEligibleToVote("Not eligible to vote")
    else:
        print("Eligible to vote")

# Main block to take input and check eligibility
try:
    age = int(input())
    check_voting_eligibility(age)
except NotEligibleToVote as e:
    print(e)

```

Status **Correct** | Mark obtained **10/10** | Hints used **0** | Times compiled **1** | Times submitted **1** | Level **Easy** | Question type **Single File Programming** | Subject **Python** | Topic **Exception Handling** | Sub Topic **User Defined Exception Handling** | Blooms taxonomy **Apply**

Show testcase scores Show solution

Question No: 4

Single File Programming Question

Problem Statement

Tara is a content manager who needs to perform case conversions for various pieces of text and save the results in a structured manner.

She requires a program to take a user's input string, save it in a file, and then retrieve and display the string in both **upper-case and lower-case versions.** Help her achieve this task efficiently.

File Name: text_file.txt

Input format :

The input consists of a single line containing a string provided by the user.

Output format :

The first line displays the original string read from the file in the format: "Original String: {original_string}".
The second line displays the upper-case version of the original string in the format: "Upper-Case String: {upper_case_string}".

The third line displays the lower-case version of the original string in the format: "Lower-Case String: {lower_case_string}".

Refer to the sample output for the formatting specifications.

Code constraints :

The input string can contain alphanumeric characters, spaces, and special symbols.
5 ≤ length of the input string ≤ 500

Sample test cases :

Input 1 :

```
#SpecialSymBoLs1234
```

Output 1 :

```
Original String: #SpecialSymBoLs1234
Upper-Case String: #SPECIALSYMBOLS1234
Lower-Case String: #specialsymbols1234
```

Input 2 :

```
LoReM iPsUm D010r
```

Output 2 :

```
Original String: LoReM iPsUm D010r
Upper-Case String: LOREM IPSUM D0L0R
Lower-Case String: lorem ipsum d010r
```

Input 3 :

```
1abc!@#AbC!@#123
```

Output 3 :

```
Original String: 1abc!@#AbC!@#123
Upper-Case String: 1ABC!@#ABC!@#123
Lower-Case String: 1abc!@#abc!@#123
```

Fill your code here — Python

```python
# You are using Python
# Define the file name
file_name = "text_file.txt"

# Get user input
user_input = input().strip()

# Validate input length
if not (5 <= len(user_input) <= 500):
    print("Error: The input string must be between 5 and 500 characters.")
    exit()

# Write the input string to the file
with open(file_name, "w") as file:
    file.write(user_input)

# Read the string from the file
with open(file_name, "r") as file:
    original_string = file.read().strip()

# Perform case conversions
upper_case_string = original_string.upper()
lower_case_string = original_string.lower()

# Display results
print(f"Original String: {original_string}")
print(f"Upper-Case String: {upper_case_string}")
print(f"Lower-Case String: {lower_case_string}")

```

Status **Correct** | Mark obtained **10/10** | Hints used **0** | Times compiled **3** | Times submitted **1** | Level **Medium** | Question type **Single File Programming** | Subject **Programming**
Topic **Programming** | Sub Topic **programming** | Blooms taxonomy **Understand**

☐ Show testcase scores ☐ Show solution

Question No: 5

Single File Programming Question

Problem Statement

A retail store requires a program to calculate the total cost of purchasing a product based on its price and quantity. The program performs validation to ensure valid inputs and handles specific error conditions using exceptions:

- Price Validation: If the price is zero or less, raise a ValueError with the message: "Invalid Price".
- Quantity Validation: If the quantity is zero or less, raise a ValueError with the message: "Invalid Quantity".
- Cost Threshold: If the total cost exceeds 1000, raise RuntimeError with the message: "Excessive Cost".

Input format :

The first line of input consists of a double value, representing the price of a product.
The second line consists of an integer, representing the quantity of the product.

Output format :

If the calculation is successful, print the total cost rounded to one decimal place.
If the price is zero or less prints "Invalid Price".
If the quantity is zero or less prints "Invalid Quantity".
If the total cost exceeds 1000, prints "Excessive Cost".

Refer to the sample output for formatting specifications.

Code constraints :

The given test cases fall under the following constraints:

-10.0 ≤ price ≤ 100.0
-1 ≤ quantity ≤ 10000

Sample test cases :

Input 1 :

```
20.0
5
```

Output 1 :

```
100.0
```

Input 2 :

```
-10.0
5
```

Output 2 :

```
Invalid Price
```

Input 3 :

```
20.0
60
```

Output 3 :

```
Excessive Cost
```

Input 4 :

```
70.0
0
```

Output 4 :

```
Invalid Quantity
```

Fill your code here

Python _

```python
# You are using Python
def calculate_total_cost(price, quantity):
    try:
        # Validate price
        if price <= 0:
            raise ValueError("Invalid Price")

        # Validate quantity
        if quantity <= 0:
            raise ValueError("Invalid Quantity")

        # Calculate total cost
        total_cost = price * quantity

        # Validate cost threshold
        if total_cost > 1000:
            raise RuntimeError("Excessive Cost")

        # Print total cost rounded to one decimal place
        print(f"{total_cost:.1f}")

    except ValueError as e:
        print(e)

    except RuntimeError as e:
        print(e)

# Get user input
try:
    price = float(input())
    quantity = int(input())

    # Call the function to calculate total cost
```

Status **Correct** | Mark obtained **10/10** | Hints used **0** | Times compiled **1** | Times submitted **1** | Level **Medium** | Question type **Single File Programming** | Subject **Programming** | Topic **Python** | Sub Topic **Exception Handling** | Blooms taxonomy **Apply**

☐ Show testcase scores ☐ Show solution

Result & Analysis

Attempt 1 of 01

Student: kamali rj | Email id: 240701225@rajalakshmi.edu.in | Test: REC_Python_Week 6_MCQ | Course: NeoColab_REC_CS23221_Python Programming

IP Address: 2401:4900:1730:a5c8:a877:630f:3738:3... | Tab Switches: -- | OS Used: Windows | Browser Used: Firefox

Test Duration: 00:23:47 | Test Start Time: May 23, 2025 | 10:06 PM | Test Submit Time: May 23, 2025 | 10:44 PM | Resume Count: 1

Summary | Sections | Filters

1 MCQ (20)

Question No: 11

Multi Choice Type Question

Which of the following is true about the finally block in Python?

- The finally block is optional and does not have to be used
- The finally block is executed only if an exception occurs
- The finally block is always executed, regardless of whether an exception occurs or not ✓
- The finally block is executed only if no exception occurs

Status **Correct** | Mark obtained **1/1** | Hints used **0** | Level **Easy** | Question type **MCQ Single Correct** | Subject **Programming** | Topic **Python** | Sub Topic **Exception Handling** | Blooms taxonomy **Remember**

Show solution

Question No: 12

Multi Choice Type Question

How do you create a user-defined exception in Python?

- By creating a new class that inherits from the Exception class ✓
- By using the raise keyword
- By writing a try block with custom logic
- By defining a function and using it in try-except

Status **Correct** | Mark obtained **1/1** | Hints used **0** | Level **Easy** | Question type **MCQ Single Correct** | Subject **Programming** | Topic **Python** | Sub Topic **Exception Handling** | Blooms taxonomy **Remember**

Show solution

Question No: 13

Multi Choice Type Question

What will be the output of the following Python code?

```
# Predefined lines to simulate the file content
lines = [
    "This is 1st line",
    "This is 2nd line",
    "This is 3rd line",
    "This is 4th line",
    "This is 5th line"
]

print("Name of the file: foo.txt")

# Print the first 5 lines from the predefined list
for index in range(5):
    line = lines[index]
    print("Line No %d - %s" % (index + 1, line.strip()))
```

Compile Time Error

Displays Output ✓

Syntax Error

None of the mentioned options

Status **Correct** | Mark obtained **1/1** | Hints used **0** | Level **Medium** | Question type **MCQ Single Correct** | Subject **Python** | Topic **File Handling** | Sub Topic **File Handling** | Blooms taxonomy **Understand**

Show solution

Question No: 14

Multi Choice Type Question

What is the correct way to raise an exception in Python?

throw Exception()

exit Exception()

raise Exception() ✓

catch Exception()

Status **Correct** | Mark obtained **1/1** | Hints used **0** | Level **Easy** | Question type **MCQ Single Correct** | Subject **Programming** | Topic **Python** | Sub Topic **Exception Handling** | Blooms taxonomy **Remember**

Show solution

Question No: 15

Multi Choice Type Question

What is the difference between r+ and w+ modes?

no difference

in w+ the pointer is initially placed at the beginning of the file and the pointer is at the end for r+

depends on the operating system

in r+ the pointer is initially placed at the beginning of the file and the pointer is at the end for w+ ✓

Status **Correct** | Mark obtained **1/1** | Hints used **0** | Level **Easy** | Question type **MCQ Single Correct** | Subject **Python** | Topic **File Handling** | Sub Topic **File Handling** | Blooms taxonomy **Analyse**

Show solution

Question No: 16

Multi Choice Type Question

Which of the following is true about
fp.seek(10,1)

Move file pointer ten characters behind from the current position

Move file pointer ten characters behind from the end of a file

Move file pointer ten characters ahead from the current position ✓

Move file pointer ten characters ahead from the beginning of a file

Status **Correct** | Mark obtained **1/1** | Hints used **0** | Level **Easy** | Question type **MCQ Single Correct** | Subject **Python** | Topic **File Handling** | Sub Topic **File Handling**
Blooms taxonomy **Remember**

Show solution

Question No: 17

Multi Choice Type Question

What happens if no arguments are passed to the seek function?

file position remains unchanged ✓

file position is set to the start of file

error ✗

file position is set to the end of file

Status **Wrong** | Mark obtained **0/1** | Hints used **0** | Level **Easy** | Question type **MCQ Single Correct** | Subject **Python** | Topic **File Handling** | Sub Topic **File Handling**
Blooms taxonomy **Remember**

Show solution

Question No: 18

Multi Choice Type Question

What is the output of the following code?

```
try:
    x = 1 / 0
except ZeroDivisionError:
    print("Caught division by zero error")
finally:
    print("Executed")
```

Executed

ZeroDivisionError followed by Executed

ZeroDivisionError

Caught division by zero error
Executed ✓

Status **Correct** | Mark obtained **1/1** | Hints used **0** | Level **Easy** | Question type **MCQ Single Correct** | Subject **Programming** | Topic **Python** | Sub Topic **Exception Handling**
Blooms taxonomy **Understand**

Show solution

Question No: 19

Multi Choice Type Question

What is the purpose of the except clause in Python?

To handle exceptions during code execution ✓

To exit the program abruptly

To display the final output of the program

To define a custom exception

Status **Correct** | Mark obtained **1/1** | Hints used **0** | Level **Easy** | Question type **MCQ Single Correct** | Subject **Programming** | Topic **Python** | Sub Topic **Exception Handling** | Blooms taxonomy **Remember**

Show solution

Question No: 20

Multi Choice Type Question

Match the following:

a) f.seek(5,1) i) Move file pointer five characters behind from the current position
b) f.seek(-5,1) ii) Move file pointer to the end of a file
c) f.seek(0,2) iii) Move file pointer five characters ahead from the current position
d) f.seek(0) iv) Move file pointer to the beginning of a file

a-i, b-iii, c-iv, d-ii

a-iii, b-i, c-ii, d-iv ✓

a-i, b-iii, c-ii, d-iv

a-iii, b-i, c-iv, d-ii

Status **Correct** | Mark obtained **1/1** | Hints used **0** | Level **Medium** | Question type **MCQ Single Correct** | Subject **Python** | Topic **File Handling** | Sub Topic **File Handling** | Blooms taxonomy **Remember**

Result & Analysis

Attempt 1 of 01

Student: kamali rj | Email id: 240701225@rajalakshmi.edu.in | Test: REC_Python_Week 6_MCQ | Course: NeoColab_REC_CS23221_Python Programming

IP Address: 2401:4900:1730:a5c8:a877:630f:3738:3... | Tab Switches: -- | OS Used: Windows | Browser Used: Firefox

Test Duration: 00:23:47 | Test Start Time: May 23, 2025 | 10:06 PM | Test Submit Time: May 23, 2025 | 10:44 PM | Resume Count: 1

Summary | Sections | Filters

1 MCQ (20)

Question No: 1

Multi Choice Type Question

What happens if an exception is not caught in the except clause?

- ✓ The program will display a traceback error and stop execution
- raise CustomException()
- The program will exit automatically
- The exception is ignored, and the program continues

Status **Correct** | Mark obtained **1/1** | Hints used **0** | Level **Easy** | Question type **MCQ Single Correct** | Subject **Programming** | Topic **Python** | Sub Topic **Exception Handling** | Blooms taxonomy **Remember**

Show solution

Question No: 2

Multi Choice Type Question

What will be the output of the following Python code?

```
f = None
for i in range (5):
    with open("data.txt", "w") as f:
        if i > 2:
            break
print(f.closed)
```

- False
- ✓ True
- ✗ Compile Time Error
- None of the mentioned options

Status **Wrong** | Mark obtained **0/1** | Hints used **0** | Level **Medium** | Question type **MCQ Single Correct** | Subject **Python** | Topic **File Handling** | Sub Topic **File Handling** | Blooms taxonomy **Understand**

Show solution

Question No: 3

Multi Choice Type Question

How do you rename a file?

- os.set_name(existing_name, new_name)
- ✓ os.rename(existing_name, new_name)

fp.name = 'new_name.txt'

os.rename(fp, new_name)

Status **Correct** | Mark obtained **1/1** | Hints used **0** | Level **Medium** | Question type **MCQ Single Correct** | Subject **Python** | Topic **File Handling** | Sub Topic **File Handling**
Blooms taxonomy **Remember**

Show solution

Question No: 4

Multi Choice Type Question

Fill in the blanks in the following code of writing data in binary files.

```
1  import _____________ (1)
2  rec=[]
3  while True:
4      rn=int(input("Enter"))
5      nm=input("Enter")
6      temp=[rn, nm]
7      rec.append(temp)
8      ch=input("Enter choice (y/N)")
9      if ch.upper=="N":
10         break
11 f.open("stud.dat","_____________")(2)
12 ___________.dump(rec,f)(3)
13 ___________.close()(4)
```

(f,r,wb,pickle)

(pickle,wb,pickle,f) ✓

(pickle,w,pickle,r)

(f,rw,pickle,pickle)

Status **Correct** | Mark obtained **1/1** | Hints used **0** | Level **Medium** | Question type **MCQ Single Correct** | Subject **Python** | Topic **File Handling** | Sub Topic **File Handling**
Blooms taxonomy **Understand**

Show solution

Question No: 5

Multi Choice Type Question

Fill the code to in order to read file from the current position.
Assuming exp.txt file has following 3 lines, consider current file position is beginning of 2nd line
Meri,25
John,21
Raj,20
Ouptput:
['John,21\n','Raj,20\n']

```
1  f = open("exp.txt", "w+")
2  ___________(1)
3  print _____________(2)
```

1) f.seek(0, 1)
2) f.readlines() ✓

1) f.seek(0, 2)
2) f.readlines()

1) f.seek(0, 1)
2) f.rline()

1) f.seek(0, 0)
2) f.rlines()

Status **Correct** | Mark obtained **1/1** | Hints used **0** | Level **Medium** | Question type **MCQ Single Correct** | Subject **Python** | Topic **File Handling** | Sub Topic **File Handling**
Blooms taxonomy **Understand**

Show solution

Question No: 6

Multi Choice Type Question

Fill in the code in order to get the following output:

Output:
Name of the file: ex.txt

```
fo = open(_______(1), "wb")
print("Name of the file: ",_______)(2)
```

- [x] 1) "ex.txt"
  2) fo.name

- [ ] 1) "ex.txt"
  2) fo.name()

- [ ] 1) ex.txt
  2) fo.name()

- [ ] 1) ex.txt
  2) fo.name

Status **Correct** | Mark obtained **1/1** | Hints used **0** | Level **Medium** | Question type **MCQ Single Correct** | Subject **Python** | Topic **File Handling** | Sub Topic **Python directory and file management** | Blooms taxonomy **Understand**

Show solution

Question No: 7

Multi Choice Type Question

Which clause is used to clean up resources, such as closing files in Python?

- [ ] else
- [ ] except
- [ ] try
- [x] finally

Status **Correct** | Mark obtained **1/1** | Hints used **0** | Level **Easy** | Question type **MCQ Single Correct** | Subject **Programming** | Topic **Python** | Sub Topic **Exception Handling** | Blooms taxonomy **Remember**

Show solution

Question No: 8

Multi Choice Type Question

What is the default value of reference_point in the following code?

```
file_object.seek(offset [,reference_point])
```

- [x] 0
- [ ] -1
- [ ] null
- [ ] garbage

Status **Correct** | Mark obtained **1/1** | Hints used **0** | Level **Easy** | Question type **MCQ Single Correct** | Subject **Python** | Topic **File Handling** | Sub Topic **File Handling** | Blooms taxonomy **Understand**

Show solution

Question No: 9

Multi Choice Type Question

What is the output of the following code?

```
try:
    x = "hello" + 5
except TypeError:
    print("Type Error occurred")
finally:
    print("This will always execute")
```

- Executed
- ✓ Type Error occurred
  This will always execute
- Type Error occurred
- This will always execute

Status **Correct** | Mark obtained **1/1** | Hints used **0** | Level **Easy** | Question type **MCQ Single Correct** | Subject **Programming** | Topic **Python** | Sub Topic **Exception Handling** | Blooms taxonomy **Understand**

Show solution

Question No: 10

Multi Choice Type Question

What is the output of the following code?

```
class MyError(Exception):
    pass

try:
    raise MyError("Something went wrong")
except MyError as e:
    print(e)
```

- ✓ Something went wrong
- Type Error occurred
- Executed
- Nothing, the code will fail

Status **Correct** | Mark obtained **1/1** | Hints used **0** | Level **Easy** | Question type **MCQ Single Correct** | Subject **Programming** | Topic **Python** | Sub Topic **Exception Handling** | Blooms taxonomy **Understand**

Show solution