Rajalakshmi Engineering College

Name: Nishal P
Email: 241501130@rajalakshmi.edu.in
Roll no: 241501130
Phone: 9790965308
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_COD

Attempt : 1
Total Mark : 50
Marks Obtained : 48.5

Section 1 : Coding

1. 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.

Sample Test Case

Input: 20.0
5
Output: 100.0

Answer

```
# You are using Python
def calculate_total_cost():
    try:
        price = float(input())
        quantity = int(input())

        if price <= 0:
            raise ValueError("Invalid Price")

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

        total_cost = price * quantity

        if total_cost > 1000:
            raise RuntimeError("Excessive Cost")

        print(f"{total_cost:.1f}")
```

```
    except ValueError as e:
        print(e)
    except RuntimeError as e:
        print(e)

calculate_total_cost()
```

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

2. 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.

Sample Test Case

Input: #SpecialSymBoLs1234
Output: Original String: #SpecialSymBoLs1234
Upper-Case String: #SPECIALSYMBOLS1234
Lower-Case String: #specialsymbols1234

Answer

```
# You are using Python
def case_conversion():
    user_input = input().strip()

    with open("text_file.txt", "w") as file:
        file.write(user_input)

    with open("text_file.txt", "r") as file:
        original_string = file.read().strip()

    upper_case_string = original_string.upper()
    lower_case_string = original_string.lower()

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

case_conversion()
```

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

3. 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.

Sample Test Case

Input: Four Words In This Sentence
Output: 5

Answer

```
def count_words():
    sentence = input().strip()

    with open("sentence_file.txt", "w") as file:
        file.write(sentence)

    with open("sentence_file.txt", "r") as file:
        stored_sentence = file.read().strip()

    word_count = len(stored_sentence.split()) if stored_sentence else 0

    print(word_count)

count_words()
```

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

4. 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.

Sample Test Case

Input: -2
1
2
Output: Error: The length of the list must be a non-negative integer.

Answer

```
# You are using Python
try:
    n=int(input())
    if n<=0:
        print("Error: The length of the list must be a non-negative integer.")
    else:
        numbers=[]
        for _ in range(n):
```

```
        try:
            num = int(input())
            numbers.append(num)
        except ValueError:
            print("Error: You must enter a numeric value.")
            break
    else:
        average = sum(numbers)/n
        print(f"The average is: {average:.2f}")
except ValueError:
    print("Error: You must enter a numeric value.")
```

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

5. 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.

Sample Test Case

Input: 18
Output: Eligible to vote

Answer

```
# You are using Python
class NotEligibleToVote(Exception):
```

```
    pass

def check_voting_eligibility(age):

    if age < 18:
        raise NotEligibleToVote("Not eligible to vote")
    else:
        print("Eligible to vote")

try:
    age = int(input())  # Read age input
    if age < 1 or age > 100:
        print("Error: Age must be between 1 and 100.")
    else:
        check_voting_eligibility(age)
except ValueError:
    print("Error: You must enter a numeric value.")
except NotEligibleToVote as e:
    print(e)
```

Status : Partially correct ***Marks : 8.5/10***

Rajalakshmi Engineering College

Name: Nishal P
Email: 241501130@rajalakshmi.edu.in
Roll no: 241501130
Phone: 9790965308
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

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
def main():
    try:
        N = int(input())  # Number of days

        if N > 30:
            print("Exceeding limit!")
            return
```

```
        items_sold = list(map(int, input().split()))
        if len(items_sold) != N:
            raise ValueError("Number of items sold does not match number of days.")

        M = int(input())  # Price of the item

        # Calculate daily earnings
        daily_earnings = [items * M for items in items_sold]

        # Write to file
        with open("sales.txt", "w") as file:
            for earning in daily_earnings:
                file.write(f"{earning}\n")

        # Read from file and display earnings
        with open("sales.txt", "r") as file:
            for line in file:
                print(line.strip())

    except ValueError as ve:
        print(f"Invalid input: {ve}")
    except IOError:
        print("File operation failed.")
    except Exception as e:
        print(f"An unexpected error occurred: {e}")

if __name__ == "__main__":
    main()
```

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

2. 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#nhoj
Output: Valid Password

Answer

```
# Password validation program for account creation

def validate_password(password):
    """
    Validates password based on the given criteria:
    - Length between 10 and 20 characters
    - At least one digit
    - At least one special character from !@#$%^&*
    """
    special_chars = "!@#$%^&*"

    # Check length
```

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

    # Check for at least one digit
    has_digit = any(char.isdigit() for char in password)
    if not has_digit:
        raise Exception("Should contain at least one digit")

    # Check for at least one special character
    has_special = any(char in special_chars for char in password)
    if not has_special:
        raise Exception("It should contain at least one special character")

    return True

# Main program
try:
    # Input collection
    name = input().strip()
    mobile = input().strip()
    username = input().strip()
    password = input().strip()

    # Validate password
    if validate_password(password):
        print("Valid Password")

except Exception as e:
    print(str(e))
```

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

3. 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

```
def is_valid_triangle(a, b, c):
    try:
        # Check if all sides are positive
        if a <= 0 or b <= 0 or c <= 0:
            raise ValueError("Side lengths must be positive")

        # Check triangle inequality theorem
        if a + b > c and a + c > b and b + c > a:
            print("It's a valid triangle")
        else:
            print("It's not a valid triangle")
```

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

# Example usage with user input
try:
    a = int(input().strip())
    b = int(input().strip())
    c = int(input().strip())

    is_valid_triangle(a, b, c)

except ValueError:
    print("ValueError: Invalid input. Please enter integers only.")
```

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

```
# You are using Python
from collections import Counter

def analyze_character_frequency(text, filename="char_frequency.txt"):
    try:
        # Count character frequencies
        frequency = Counter(text)

        # Save results to a file
        with open(filename, "w") as file:
            file.write("Character Frequencies:\n")
            for char, count in frequency.items():
                file.write(f"{char}: {count}\n")

        # Display results
        print("Character Frequencies:")
        for char, count in frequency.items():
            print(f"{char}: {count}")

    except Exception as e:
        print(f"Error: {e}")

# Example usage
text = input().strip()
analyze_character_frequency(text)
```

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