Rajalakshmi Engineering College

Name: Prasham Jaganathan
Email: 241501148@rajalakshmi.edu.in
Roll no: 241501148
Phone: 9445840008
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 : 50

Section 1 : Coding

1. 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
def calculate_average():
    try:
        n = int(input())
        if n <= 0:
            print("Error: The length of the list must be a non-negative integer.")
            return
    except ValueError:
        print("Error: You must enter a numeric value.")
        return

    numbers = []
    for _ in range(n):
        try:
            num = int(input())
```

```
            numbers.append(num)
        except ValueError:
            print("Error: You must enter a numeric value.")
            return

    average = sum(numbers) / n
    print(f"The average is: {average:.2f}")

calculate_average()
```

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

2. 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

try:
    age = int(input())
    if 1 <= age <= 100:
        check_voting_eligibility(age)
        print("Eligible to vote")
    else:
        print("Not eligible to vote")
except NotEligibleToVote:
    print("Not eligible to vote")
except ValueError:
    print("Invalid input")
```

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

3. 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(price, quantity):
    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")

    return total_cost

try:
    price = float(input())
    quantity = int(input())
    total_cost = calculate_total_cost(price, quantity)
    print(f"{total_cost:.1f}")
except ValueError as e:
    print(e)
except RuntimeError as e:
    print(e)
```

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

4. 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

```
# You are using Python
def count_words_in_sentence(sentence):
    words = sentence.split()
    return len(words)

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

with open("sentence_file.txt", "r") as file:
    content = file.read()
    word_count = count_words_in_sentence(content)

print(word_count)
```

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

5. 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
```

```
input_string = input()

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

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

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

Rajalakshmi Engineering College

Name: Prasham Jaganathan
Email: 241501148@rajalakshmi.edu.in
Roll no: 241501148
Phone: 9445840008
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 : 37.5

Section 1 : Coding

1. 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.

Sample Test Case

Input: 2022-01-12 06:10:00
2022-02-12 10:10:12
Output: 2022-01-12 06:10:00
2022-02-12 10:10:12

Answer

```
# You are using Python
from datetime import datetime

def get_event_time():
    try:
        start_time = input()
        end_time = input()

        datetime.strptime(start_time, "%Y-%m-%d %H:%M:%S")
        datetime.strptime(end_time, "%Y-%m-%d %H:%M:%S")

        print(start_time, end_time)
    except ValueError:
        print("Event time is not in the format")

get_event_time()
```

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

2. 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

```
# You are using Python
def name_sorter():
    filename = "sorted_names.txt"
    names = []


    while True:
        name = input().strip()
        if name.lower() == 'q':
            break
```

```
        if 3 <= len(name) <= 30:
            names.append(name)

    names.sort()

    with open(filename, "w") as file:
        for name in names:
            file.write(name + "\n")

    for name in names:
        print(name)

# Run the program
if __name__ == "__main__":
    name_sorter()
```

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

3. 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

```
import re

class IllegalArgumentException(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 reg_no.isalnum():
        raise NoSuchElementException("Register Number should only contain letters and digits.")

    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.")

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 contain only digits.")

class NumberFormatException(Exception):
    pass

def main():
    try:
        reg_no = input().strip()
        mobile = input().strip()

        validate_register_number(reg_no)
        validate_mobile_number(mobile)

        print("Valid")

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


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

Status : Partially correct ***Marks : 7.5/10***

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

```
from collections import Counter

def analyze_character_frequency(text):
    frequency = Counter(text)

    with open("char_frequency.txt", "w") as file:
        file.write("Character Frequencies:\n")
        print("Character Frequencies:")

        for char, count in frequency.items():
            print(f"{char}: {count}")
            file.write(f"{char}: {count}\n")

def main():
    text = input().strip()

    analyze_character_frequency(text)

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

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