Rajalakshmi Engineering College

Name: attana .s
Email: 241801030@rajalakshmi.edu.in
Roll no:
Phone: 6385736778
Branch: REC
Department: I AI & DS FA
Batch: 2028
Degree: B.E - AI & DS

Scan to verify results

NeoColab_REC_CS23221_Python Programming

REC_Python_Week 1_COD

Attempt : 1
Total Mark : 5
Marks Obtained : 5

Section 1 : Coding

1. Problem Statement

A science experiment produces a decimal value as the result. However, the scientist needs to convert this value into an integer so that it can be used in further calculations.

Write a Python program that takes a floating-point number as input and converts it into an integer.

Input Format

The input consists of a floating point number, F.

Output Format

The output prints "The integer value of F is: {result}", followed by the integer number equivalent to the floating point number.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 10.36
Output: The integer value of 10.36 is: 10

Answer

```
n = float(input())
print(f"The integer value of {n} is: {int(n)}")
```

Status : Correct ***Marks : 1/1***

2. Problem Statement

Quentin, a mathematics enthusiast, is exploring the properties of numbers. He believes that for any set of four consecutive integers, calculating the average of their fourth powers and then subtracting the product of the first and last numbers yields a constant value.

To validate his hypothesis, check if the result is indeed constant and display.

Example:

Input:

5

Output:

Constant value: 2064.5

Explanation:

Find the Average:

Average: (625 + 1296 + 2401 + 4096)/4 = 2104.5

Now, we calculate the product of a and (a + 3):

Product = 5 × (5 + 3) = 5 × 8 = 40

Final result: 2104.5 - 40 = 2064.5

Input Format

The input consists of an integer a, representing the first of four consecutive integers.

Output Format

The output displays "Constant value: " followed by the computed result based on Quentin's formula.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
Output: Constant value: 2064.5

Answer

```
a=int(input())
avg=((a)**4+(a+1)**4+(a+2)**4+(a+3)**4)/4
product=(a*(a+3))
total=avg-product
print("Constant value: ",total)
```

Status : Correct ***Marks : 1/1***

3. Problem Statement

In a family, two children receive allowances based on the gardening tasks they complete. The older child receives an allowance rate of Rs.5 for each task, with a base allowance of Rs.50. The younger child receives an allowance rate of Rs.3 for each task, with a base allowance of Rs.30.

Your task is to calculate and display the allowances for the older and younger children based on the number of gardening tasks they complete, along with the total allowance for both children combined.

Input Format

The first line of input consists of an integer n, representing the number of chores completed by the older child.

The second line consists of an integer m, representing the number of chores completed by the youngest child.

Output Format

The first line of output displays "Older child allowance: Rs." followed by an integer representing the allowance calculated for the older sibling.

The second line displays "Younger child allowance: Rs." followed by an integer representing the allowance calculated for the youngest sibling.

The third line displays "Total allowance: Rs." followed by an integer representing the sum of both siblings' allowances.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 10
5
Output: Older child allowance: Rs.100
Younger child allowance: Rs.45
Total allowance: Rs.145

Answer

```
# You are using Python
old = int(input())
young = int (input())

old = 50 + (old * 5)
young = 30 + (young * 3)
```

```
print(f"Older child allowance: Rs.{old}")
print(f"Younger child allowance: Rs.{young}")
print(f"Total allowance: Rs.{old+young}")
```

Status : Correct ***Marks : 1/1***

4. Problem Statement

A company has hired two employees, Alice and Bob. The company wants to swap the salaries of both employees. Alice's salary is an integer value and Bob's salary is a floating-point value.

Write a program to swap their salaries and print the new salary of each employee.

Input Format

The first line of input consists of an integer N, representing Alice's salary.

The second line consists of a float value F, representing Bob's salary.

Output Format

The first line of output displays "Initial salaries:"

The second line displays "Alice's salary = N", where N is Alice's salary.

The third line of output displays "Bob's salary = F", where F is Bob's salary.

After a new line space, the following line displays "New salaries after swapping:"

The next line displays "Alice's salary = X", where X is the swapped salary.

The last line displays "Bob's salary = Y", where Y is the swapped salary.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 10000
15400.55

Output: Initial salaries:
Alice's salary = 10000
Bob's salary = 15400.55

New salaries after swapping:
Alice's salary = 15400.55
Bob's salary = 10000

Answer

```
# You are using Python

alice_sal = int(input())
bob_sal = float(input())

print("Initial salaries:\n")
print("Alice's salary = ", alice_sal)
print("Bob's salary =", bob_sal)


alice_sal , bob_sal = bob_sal , alice_sal

print("New salaries after swapping:\n")
print("Alice's salary = ", alice_sal)
print("Bob's salary = ", bob_sal)
```

Status : Correct ***Marks : 1/1***

5. Problem Statement

Bob, the owner of a popular bakery, wants to create a special offer code for his customers. To generate the code, he plans to combine the day of the month with the number of items left in stock.

Help Bob to encode these two values into a unique offer code.

Note: Use the bitwise operator to calculate the offer code.

Example

Input:

15

9

Output:

Offer code: 6

Explanation:

Given the day of the month 15th day (binary 1111) and there are 9 items left (binary 1001), the offer code is calculated as 0110 which is 6.

Input Format

The first line of input consists of an integer D, representing the day of the month.

The second line consists of an integer S, representing the number of items left in stock.

Output Format

The output displays "Offer code: " followed by an integer representing the encoded offer code.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 15
9

Output: Offer code: 6

Answer

```
# You are using Python
a = int(input())
b = int(input())

print(f"Offer code: {a^b}")
```

Status : Correct ***Marks : 1/1***

Rajalakshmi Engineering College

Name: attana .s
Email: 241801030@rajalakshmi.edu.in
Roll no:
Phone: 6385736778
Branch: REC
Department: I AI & DS FA
Batch: 2028
Degree: B.E - AI & DS

Scan to verify results

NeoColab_REC_CS23221_Python Programming

REC_Python_Week 2_COD_Updated

Attempt : 1
Total Mark : 50
Marks Obtained : 50

Section 1 : Coding

1. Problem Statement

Emma, a mathematics enthusiast, is exploring a range of numbers and wants to count how many of them are not Fibonacci numbers.

Help Emma determine the count of non-Fibonacci numbers within the given range [start, end] using the continue statement.

Input Format

The first line of input consists of an integer, representing the starting number of the range.

The second line consists of an integer, representing the ending number of the range.

Output Format

The output prints a single integer, representing the count of numbers in the range that are not Fibonacci numbers.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1
10

Output: 5

Answer

```
# You are using Python
n1 = int(input())
n2 = int(input())
a , b = 0 , 1
fib = set()
while a <= n2:
    fib.add(a)
    a,b = b , a + b
count = 0
for num in range(n1 , n2 + 1):
    if num not in fib:
        count += 1
print(count)
```

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

2. Problem Statement

As a junior developer working on a text analysis project, your task is to create a program that displays the consonants in a sentence provided by the user, separated by spaces.

You need to implement a program that takes a sentence as input and prints the consonants while skipping vowels and non-alphabetic characters using only control statements.

Input Format

The input consists of a string representing the sentence.

Output Format

The output displays space-separated consonants present in the sentence.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: Hello World!
Output: H l l W r l d

Answer

```
# You are using Python
word = str(input())
vowel = 'aeiouAEIOU'
for i in word:
    if i.isalpha() and i not in vowel:
        print(i , end = " ")
    else:
        pass
```

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

3. Problem Statement

Ethan, a curious mathematician, is fascinated by perfect numbers. A perfect number is a number that equals the sum of its proper divisors (excluding itself). Ethan wants to identify all perfect numbers within a given range.

Help him write a program to list these numbers.

Input Format

The first line of input consists of an integer start, representing the starting

number of the range.

The second line consists of an integer end, representing the ending number of the range.

Output Format

The output prints all perfect numbers in the range, separated by a space.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1
100
Output: 6 28

Answer

```
# You are using Python
x = int(input())
y = int(input())
for i in range(x,y):
    total_sum = 0
    new_r = i // 2
    for j in range (1 , new_r + 1):
        if i % j == 0:
            total_sum += j
    if i == total_sum:
        print(i , end = " ")
```

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

4. Problem Statement

You work as an instructor at a math enrichment program, and your goal is to develop a program that showcases the concept of using control statements to manipulate loops. Your task is to create a program that takes an integer 'n' as input and prints the squares of even numbers from 1 to 'n', while skipping odd numbers.

Input Format

The input consists of a single integer, which represents the upper limit of the range.

Output Format

The output displays the square of even numbers from 1 to 'n' separated by lines.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 10

Output: 4
16
36
64
100

Answer

```
# You are using Python
n = int(input())
for i in range(1 , n+1):
    if i % 2 ==0:
        print(i**2)
    else:
        pass
```

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

5. Problem Statement

John, a software developer, is analyzing a sequence of numbers within a given range to calculate their digit sum. However, to simplify his task, he excludes all numbers that are palindromes (numbers that read the same backward as forward).

Help John find the total sum of the digits of non-palindromic numbers in

the range [start, end] (both inclusive).

Example:

Input:

10

20

Output:

55

Explanation:

Range [10, 20]: Non-palindromic numbers are 10, 12, 13, 14, 15, 16, 17, 18, 19 and 20.

Digit sums: 1+0 + 1+2 + 1+3 + 1+4 + 1+5 + 1+6 + 1+7 + 1+8 + 1+9 + 2+0 = 55.

Output: 55

Input Format

The first line of input consists of an integer, representing the starting number of the range.

The second line of input consists of an integer, representing the ending number of the range.

Output Format

The output prints a single integer, representing the total sum of the digits of all non-palindromic numbers in the range.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 10
20

Output: 55

Answer

```
# You are using Python
n1 = int(input())
n2 = int(input())
total_sum = 0
for i in range (n1 , n2+1):
    str_i = str(i)
    if str_i != str_i[::-1]:
        digit_sum = sum(int(digit) for digit in str_i)
        total_sum += digit_sum
print(total_sum)
```

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

Rajalakshmi Engineering College

Name: attana .s
Email: 241801030@rajalakshmi.edu.in
Roll no:
Phone: 6385736778
Branch: REC
Department: I AI & DS FA
Batch: 2028
Degree: B.E - AI & DS

Scan to verify results

NeoColab_REC_CS23221_Python Programming

REC_Python_Week 3_COD

Attempt : 1
Total Mark : 50
Marks Obtained : 50

Section 1 : Coding

1. Problem Statement

Dhruv wants to write a program to slice a given string based on user-defined start and end positions.

The program should check whether the provided positions are valid and then return the sliced portion of the string if the positions are within the string's length.

Input Format

The first line consists of the input string as a string.

The second line consists of the start position (0-based index) as an integer.

The third line consists of the end position (0-based index) as an integer.

Output Format

The output displays the following format:

If the start and end positions are valid, print the sliced string.

If the start and end positions are invalid, print "Invalid start and end positions".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: pythonprogramming
0
5
Output: python

Answer

```
# You are using Python
input_string = input().strip()
start = int(input())
end = int(input())
if 0 <= start <= end <len(input_string):
    print(input_string[start:end+1])
else:
    print("Invalid start and end positions")
```

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

2. Problem Statement

Given a list of positive and negative numbers, arrange them such that all negative integers appear before all the positive integers in the array. The order of appearance should be maintained.

Example

Input:

[12, 11, -13, -5, 6, -7, 5, -3, -6]

Output:

List = [-13, -5, -7, -3, -6, 12, 11, 6, 5]

Explanation:

The output is the arranged list where all the negative integers appear before the positive integers while maintaining the original order of appearance.

Input Format

The input consists of a single line containing a list of integers enclosed in square brackets separated by commas.

Output Format

The output displays "List = " followed by an arranged list of integers as required, separated by commas and enclosed in square brackets.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: [12, 11, -13, -5, 6, -7, 5, -3, -6]
Output: List = [-13, -5, -7, -3, -6, 12, 11, 6, 5]

Answer

```
# You are using Python
input_list = input().strip()
input_list = list(map(int,input_list.strip('[]').split(',')))
negatives = [num for num in input_list if num < 0]
postives = [num for num in input_list if num >= 0]
result = negatives + postives
print(f"List = {result}")
```

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

3. Problem Statement

You have a string containing a phone number in the format "(XXX) XXX-XXXX". You need to extract the area code from the phone number and create a new string that contains only the area code.

Write a Python program for the same.

Note

(XXX) - Area code

XXX-XXXX - Phone number

Input Format

The input consists of a string, representing the phone number in the format "(XXX) XXX-XXXX".

Output Format

The output displays "Area code: " followed by a string representing the area code for the given phone number.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: (123) 456-7890
Output: Area code: 123

Answer

```
# You are using Python
phone_number = input().strip()
area_code = phone_number[1:4]
print("Area code:", area_code)
```

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

4. Problem Statement

Alex is working on a Python program to manage a list of elements. He needs to append multiple elements to the list and then remove an element from the list at a specified index.

Your task is to create a program that helps Alex manage the list. The program should allow Alex to input a list of elements, append them to the existing list, and then remove an element at a specified index.

Input Format

The first line contains an integer n, representing the number of elements to be appended to the list.

The next n lines contain integers, representing the elements to be appended to the list.

The third line of input consists of an integer M, representing the index of the element to be popped from the list.

Output Format

The first line of output displays the original list.

The second line of output displays the list after popping the element of the index M.

The third line of output displays the popped element.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 5
64
98
-1
5
26
3
Output: List after appending elements: [64, 98, -1, 5, 26]

List after popping last element: [64, 98, -1, 26]
Popped element: 5

Answer

```
# You are using Python
n = int(input())
elements = []
for _ in range(n):
    num = int(input())
    elements.append(num)

M = int(input())
print("List after appending elements:", elements)
popped_elements = elements.pop(M)
print("List after popping last element:", elements)
print("Popped element:", popped_elements)
```

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

5. Problem Statement

Ram is working on a program to manipulate strings. He wants to create a program that takes two strings as input, reverses the second string, and then concatenates it with the first string.

Ram needs your help to design a program.

Input Format

The input consists of two strings in separate lines.

Output Format

The output displays a single line containing the concatenated string of the first string and the reversed second string.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: hello
word

Output: hellodrow

Answer

```
# You are using Python
str1 = input().strip()
str2 = input().strip()
reversed_str2 = str2[::-1]
result = str1 + reversed_str2
print(result)
```

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

Rajalakshmi Engineering College

Name: attana .s
Email: 241801030@rajalakshmi.edu.in
Roll no:
Phone: 6385736778
Branch: REC
Department: I AI & DS FA
Batch: 2028
Degree: B.E - AI & DS

Scan to verify results

NeoColab_REC_CS23221_Python Programming

REC_Python_Week 4_COD_Updated

Attempt : 1
Total Mark : 50
Marks Obtained : 40

Section 1 : Coding

1. Problem Statement

Implement a program that needs to identify Armstrong numbers. Armstrong numbers are special numbers that are equal to the sum of their digits, each raised to the power of the number of digits in the number.

Write a function is_armstrong_number(number) that checks if a given number is an Armstrong number or not.

Function Signature: armstrong_number(number)

Input Format

The first line of the input consists of a single integer, n, representing the number to be checked.

Output Format

The output should consist of a single line that displays a message indicating whether the input number is an Armstrong number or not.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 153

Output: 153 is an Armstrong number.

Answer

```
def armstrong_number(number):
    num_str = str(number)
    num_digits = len(num_str)
    sum_of_powers = sum(int(digit)**num_digits for digit in num_str)
    if sum_of_powers == number:
        print(f"{number} is an Armstrong number.")
    else:
        print(f"{number} is not an Armstrong number ." )
n = int(input())
armstrong_number(n)
```

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

2. Problem Statement

Imagine you are developing a text analysis tool for a cybersecurity company. Your task is to create a function that analyzes input strings to categorize and count the characters into four categories: uppercase letters, lowercase letters, digits, and special characters. The company needs this tool to process log files and identify potential security threats.

Function Signature: analyze_string(input_string)

Input Format

The input consists of a single string (without space), which may include uppercase letters, lowercase letters, digits, and special characters.

Output Format

The first line contains an integer representing the count of uppercase letters in the format "Uppercase letters: [count]".

The second line contains an integer representing the count of lowercase letters in the format "Lowercase letters: [count]".

The third line contains an integer representing the count of digits in the format "Digits: [count]".

The fourth line contains an integer representing the count of special characters in the format "Special characters: [count]".

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: Hello123

Output: Uppercase letters: 1
Lowercase letters: 4
Digits: 3
Special characters: 0

Answer

-

Status : Skipped ***Marks : 0/10***

3. Problem Statement

Sara is developing a text-processing tool that checks if a given string starts with a specific character or substring. She needs to implement a function that accepts a string and a character (or substring), and returns True if the string starts with the provided character/substring, or False otherwise.

Write a program that uses a lambda function to help Sara perform this check.

Input Format

The first line contains a string `str` representing the main string to be checked.

The second line contains a string `n`, which is the character or substring to check if the main string starts with it.

Output Format

The first line of output prints "True" if the string starts with the given character/ substring, otherwise prints "False".

Refer to the sample for the formatting specifications.

Sample Test Case

Input: Examly
e

Output: False

Answer

```
# You are using Python
# Read input
main_string = input().strip()
substring = input().strip()

# Lambda function to check if the string starts with the given substring
starts_with = lambda s, sub: s.startswith(sub)

# Output the result
print(starts_with(main_string, substring))
```

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

4. Problem Statement

Imagine you are building a messaging application, and you want to know the length of the messages sent by the users. You need to create a program that calculates the length of a message using the built-in function

len().

Input Format

The input consists of a string representing the message.

Output Format

The output prints an integer representing the length of the entered message.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: hello!!
Output: 7

Answer

```
message = input()


print(len(message))
```

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

5. Problem Statement

Sneha is building a more advanced exponential calculator. She wants to implement a program that does the following:

Calculates the result of raising a given base to a specific exponent using Python's built-in pow() function.Displays all intermediate powers from base¹ to base^exponent as a list.Calculates and displays the sum of these intermediate powers.

Help her build this program to automate her calculations.

Input Format

The input consists of line-separated two integer values representing base and exponent.

Output Format

The first line of the output prints the calculated result of raising the base to the exponent.

The second line prints a list of all powers from base^1 to base^exponent.

The third line prints the sum of all these powers.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 2
3
Output: 8
[2, 4, 8]
14

Answer

```
base = int(input())
exponent = int(input())


result = pow(base, exponent)


powers = [pow(base, i) for i in range(1, exponent + 1)]


powers_sum = sum(powers)


print(result)
print(powers)
print(powers_sum)
```

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

Rajalakshmi Engineering College

Name: attana .s
Email: 241801030@rajalakshmi.edu.in
Roll no:
Phone: 6385736778
Branch: REC
Department: I AI & DS FA
Batch: 2028
Degree: B.E - AI & DS

Scan to verify results

NeoColab_REC_CS23221_Python Programming

REC_Python_Week 5_COD

Attempt : 1
Total Mark : 50
Marks Obtained : 40

Section 1 : Coding

1. Problem Statement

Ella is analyzing the sales data for a new online shopping platform. She has a record of customer transactions where each customer's data includes their ID and a list of amounts spent on different items. Ella needs to determine the total amount spent by each customer and identify the highest single expenditure for each customer.

Your task is to write a program that computes these details and displays them in a dictionary.

Input Format

The first line of input consists of an integer n, representing the number of customers.

Each of the next n lines contains a numerical customer ID followed by integers representing the amounts spent on different items.

Output Format

The output displays a dictionary where the keys are customer IDs and the values are lists containing two integers: the total expenditure and the maximum single expenditure.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 2
101 100 150 200
102 50 75 100
Output: {101: [450, 200], 102: [225, 100]}

Answer

```
# You are using Python
n = int(input())  # Number of customers
data = list(map(int, input().split()))

sales = {}
i = 0
while i < len(data):
    customer_id = data[i]
    i += 1
    amounts = []
    # Collect amounts until the next customer ID or end of list
    while i < len(data) and not (101 <= data[i] <= 110 and len(sales) + 1 < n):
        amounts.append(data[i])
        i += 1
    sales[customer_id] = [sum(amounts), max(amounts)]

print(sales)
```

Status : Wrong ***Marks : 0/10***

2. Problem Statement

Gowshik is working on a task that involves taking two lists of integers as input, finding the element-wise sum of the corresponding elements, and then creating a tuple containing the sum values.

Write a program to help Gowshik with this task.

Example:

Given list:

[1, 2, 3, 4]

[3, 5, 2, 1]

An element-wise sum of the said tuples: (4, 7, 5, 5)

Input Format

The first line of input consists of a single integer n, representing the length of the input lists.

The second line of input consists of n integers separated by commas, representing the elements of the first list.

The third line of input consists of n integers separated by commas, representing the elements of the second list.

Output Format

The output is a single line containing a tuple of integers separated by commas, representing the element-wise sum of the corresponding elements from the two input lists.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 4
1, 2, 3, 4
3, 5, 2, 1

Output: (4, 7, 5, 5)

Answer

```
# You are using Python
# Read input
n = int(input())  # Length of the lists
list1 = list(map(int, input().split(',')))
list2 = list(map(int, input().split(',')))

# Compute element-wise sum
result = tuple(list1[i] + list2[i] for i in range(n))

# Print the result
print(result)
```

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

3. Problem Statement

Professor Adams needs to analyze student participation in three recent academic workshops. She has three sets of student IDs: the first set contains students who registered for the workshops, the second set contains students who actually attended, and the third set contains students who dropped out.

Professor Adams needs to determine which students who registered also attended, and then identify which of these students did not drop out.

Help Professor Adams identify the students who registered, attended, and did not drop out of the workshops.

Input Format

The first line of input consists of integers, representing the student IDs who registered for the workshops.

The second line consists of integers, representing the student IDs who attended the workshops.

The third line consists of integers, representing the student IDs who dropped out

of the workshops.

Output Format

The first line of output displays the intersection of the first two sets, which shows the IDs of students who registered and attended.

The second line displays the result after removing student IDs that are in the third set (dropped out), showing the IDs of students who both attended and did not drop out.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 1 2 3
2 3 4
3 4 5

Output: {2, 3}
{2}

Answer

```
# You are using Python
# Read input and convert to sets
registered = set(map(int, input().split()))
attended = set(map(int, input().split()))
dropped_out = set(map(int, input().split()))

# Step 1: Students who registered and attended
reg_and_att = registered & attended

# Step 2: Students who registered, attended, and did not drop out
final_students = reg_and_att - dropped_out

# Output the results
print(reg_and_att)
print(final_students)
```

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

4. Problem Statement

Liam is analyzing a list of product IDs from a recent sales report. He needs to determine how frequently each product ID appears and calculate the following metrics:

Frequency of each product ID: A dictionary where the key is the product ID and the value is the number of times it appears.Total number of unique product IDs.Average frequency of product IDs: The average count of all product IDs.

Write a program to read the product IDs, compute these metrics, and output the results.

Example

Input:

6 //number of product ID

101

102

101

103

101

102 //product IDs

Output:

{101: 3, 102: 2, 103: 1}

Total Unique IDs: 3

Average Frequency: 2.00

Explanation:

Input 6 indicates that you will enter 6 product IDs.

A dictionary is created to track the frequency of each product ID.

Input 101: Added with a frequency of 1.

Input 102: Added with a frequency of 1.

Input 101: Frequency of 101 increased to 2.

Input 103: Added with a frequency of 1.

Input 101: Frequency of 101 increased to 3.

Input 102: Frequency of 102 increased to 2.

The dictionary now contains 3 unique IDs: 101, 102, and 103.

Total Unique is 3.

The average frequency is 2.00.

Input Format

The first line of input consists of an integer n, representing the number of product IDs.

The next n lines each contain a single integer, each representing a product ID.

Output Format

The first line of output displays the frequency dictionary, which maps each product ID to its count.

The second line displays the total number of unique product IDs, preceded by "Total Unique IDs: ".

The third line displays the average frequency of the product IDs. This is calculated by dividing the total number of occurrences of all product IDs by the total number of unique product IDs, rounded to two decimal places. It is preceded by "Average Frequency: ".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 6
101

102
101
103
101
102

Output: {101: 3, 102: 2, 103: 1}
Total Unique IDs: 3
Average Frequency: 2.00

Answer

```
# You are using Python
n = int(input())  # Number of product IDs

# Initialize frequency dictionary
freq = {}

# Read product IDs and update frequencies
for _ in range(n):
    pid = int(input())
    freq[pid] = freq.get(pid, 0) + 1

# Compute total unique IDs
total_unique = len(freq)

# Compute average frequency
total_occurrences = sum(freq.values())
average_freq = total_occurrences / total_unique

# Output
print(freq)
print(f"Total Unique IDs: {total_unique}")
print(f"Average Frequency: {average_freq:.2f}")
```

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

5. Problem Statement

James is managing a list of inventory items in a warehouse. Each item is recorded as a tuple, where the first element is the item ID and the second

element is a list of quantities available for that item. James needs to filter out all quantities that are above a certain threshold to find items that have a stock level above this limit.

Help James by writing a program to process these tuples, filter the quantities from all the available items, and display the results.

Note:

Use the filter() function to filter out the quantities greater than the specified threshold for each item's stock list.

Input Format

The first line of input consists of an integer N, representing the number of tuples.

The next N lines each contain a tuple in the format (ID, [quantity1, quantity2, ...]), where ID is an integer and the list contains integers.

The final line consists of an integer threshold, representing the quantity threshold.

Output Format

The output should be a single line displaying the filtered quantities, space-separated. Each quantity is strictly greater than the given threshold.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 2
(1, [1, 2])
(2, [3, 4])
2

Output: 3 4

Answer

```
# You are using Python
import ast  # For safely evaluating tuple input
```

```
N = int(input())  # Number of tuples

all_quantities = []

# Read N tuples
for _ in range(N):
    item = ast.literal_eval(input())  # Convert string to tuple
    quantities = item[1]
    all_quantities.extend(quantities)

# Read threshold
threshold = int(input())

# Use filter to get quantities > threshold
filtered = list(filter(lambda x: x > threshold, all_quantities))

# Print space-separated results
print(*filtered)
```

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

Rajalakshmi Engineering College

Name: attana .s
Email: 241801030@rajalakshmi.edu.in
Roll no:
Phone: 6385736778
Branch: REC
Department: I AI & DS FA
Batch: 2028
Degree: B.E - AI & DS

Scan to verify results

NeoColab_REC_CS23221_Python Programming

REC_Python_Week 6_COD

Attempt : 1
Total Mark : 50
Marks Obtained : 47.5

Section 1 : Coding

1. 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
# Read input string from user
input_string = input()

# Write the input string to the file
with open("text_file.txt", "w") as file:
    file.write(input_string)

# Read the string back from the file
with open("text_file.txt", "r") as file:
    original_string = file.read()

# Display the required output
print(f"Original String: {original_string}")
print(f"Upper-Case String: {original_string.upper()}")
print(f"Lower-Case String: {original_string.lower()}")
```

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

2. 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 = input().strip()
    # Check if n is numeric integer
    if not n.isdigit():
        print("Error: You must enter a numeric value.")
        exit()

    n = int(n)

    # Check if n is positive integer (non-negative means > 0 here based on sample
tests)
    if n <= 0:
        print("Error: The length of the list must be a non-negative integer.")
        exit()

    numbers = []
    for _ in range(n):
        elem = input().strip()
        if not (elem.lstrip('-').isdigit()):
            print("Error: You must enter a numeric value.")
            exit()
        numbers.append(int(elem))

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

except Exception:
    print("Error: You must enter a numeric value.")
```

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

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

```
# You are using Python
# Read the input sentence
sentence = input()

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

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

# Count the words - split by whitespace and count non-empty strings
if content == "":
    count = 0
else:
    count = len(content.split())

print(count)
```

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

4. 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
try:
```

```
    price = float(input().strip())
    quantity = int(input().strip())

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

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

    total_cost = price * quantity

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

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

except ValueError as ve:
    print(ve)
except RuntimeError as re:
    print(re)
```

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 NotEligibleError(Exception):
    pass

try:
    age = int(input().strip())
    if age < 18:
        raise NotEligibleError("Not eligible to vote")
    print("Eligible to vote")
except NotEligibleError as e:
    print(e)
```

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

Rajalakshmi Engineering College

Name: attana .s
Email: 241801030@rajalakshmi.edu.in
Roll no:
Phone: 6385736778
Branch: REC
Department: I AI & DS FA
Batch: 2028
Degree: B.E - AI & DS

Scan to verify results

NeoColab_REC_CS23221_Python Programming

REC_Python_Week 7_COD

Attempt : 1
Total Mark : 50
Marks Obtained : 50

Section 1 : Coding

1. Problem Statement

Sita works as a sales analyst and needs to analyze monthly sales data for different cities. She receives lists of cities, months, and corresponding sales values and wants to create a pandas DataFrame using a MultiIndex of cities and months.

Help her to implement this task and calculate total sales for each city.

Input Format

The first line of input consists of an integer value, n, representing the number of records.

The second line of input consists of n space-separated city names.

The third line of input consists of n space-separated month names.

The fourth line of input consists of n space-separated float values representing sales for each city-month combination.

Output Format

The first line of output prints: "Monthly Sales Data with MultiIndex:"

The next lines print the DataFrame with MultiIndex (City, Month) and their corresponding sales values.

The following line prints: "\nTotal Sales Per City:"

The final lines print the total sales per city, computed by grouping the sales data on city names.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 4
NYC NYC LA LA
Jan Feb Jan Feb
100 200 300 400

Output: Monthly Sales Data with MultiIndex:

```
          Sales
City Month
NYC  Jan    100.0
     Feb    200.0
LA   Jan    300.0
     Feb    400.0

Total Sales Per City:
      Sales
City
LA    700.0
NYC   300.0
```

Answer

```
# You are using Python
import pandas as pd
```

```
# Read input
n = int(input())
cities = input().split()
months = input().split()
sales = list(map(float, input().split()))

# Create MultiIndex
index = pd.MultiIndex.from_tuples(zip(cities, months), names=["City", "Month"])

# Create DataFrame
df = pd.DataFrame({'Sales': sales}, index=index)

# Display the MultiIndex DataFrame
print("Monthly Sales Data with MultiIndex:")
print(df)

# Group by City and sum Sales
total_sales = df.groupby('City').sum()

# Display total sales per city
print("\nTotal Sales Per City:")
print(total_sales)
```

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

2. Problem Statement

Rekha works in hospital data management and receives patient records with missing or incomplete data. She needs to clean the records by performing the following tasks:

Calculate the mean of the available Age values.Replace any missing (NaN) values in the Age column with this mean age.Remove any rows where the Diagnosis value is missing (NaN).Reset the DataFrame index after removing these rows.

Implement this data cleaning task using the pandas package.

Input Format

The first line of input contains an integer n representing the number of patient records.

The second line contains the CSV header — comma-separated column names (e.g., "Name,Age,Diagnosis,Gender").

The next n lines each contain one patient record in comma-separated format.

Output Format

The first line of output is the text:

Cleaned Hospital Records:

The next lines print the cleaned pandas DataFrame (as produced by print(cleaned_df)).

This will include the updated values of the Age column (with missing ages filled by the mean age), and any rows with missing Diagnosis removed.

The DataFrame will be displayed using the default pandas print() representation.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 5
PatientID,Name,Age,Diagnosis
1,John Doe,45,Flu
2,Jane Smith,,Cold
3,Bob Lee,50,
4,Alice Green,38,Fever
5,Tom Brown,,Infection

Output: Cleaned Hospital Records:

	PatientID	Name	Age	Diagnosis
0	1	John Doe	45.000000	Flu
1	2	Jane Smith	44.333333	Cold
2	4	Alice Green	38.000000	Fever
3	5	Tom Brown	44.333333	Infection

Answer

```
# You are using Python
import pandas as pd
import sys
import numpy as np

# Read input
n = int(input())
header = input().split(',')

# Read patient records into a list
data = [input().split(',') for _ in range(n)]

# Create the DataFrame
df = pd.DataFrame(data, columns=header)

# Convert 'Age' to numeric (to handle missing/invalid values as NaN)
df['Age'] = pd.to_numeric(df['Age'], errors='coerce')

# Calculate mean of available Age values
mean_age = df['Age'].mean()

# Replace NaN values in Age with mean_age
df['Age'] = df['Age'].fillna(mean_age)

# Replace empty strings with NaN in Diagnosis and drop those rows
df['Diagnosis'].replace('', np.nan, inplace=True)
cleaned_df = df.dropna(subset=['Diagnosis'])

# Reset the index
cleaned_df = cleaned_df.reset_index(drop=True)

# Print the cleaned DataFrame
print("Cleaned Hospital Records:")
print(cleaned_df)
```

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

3. Problem Statement

Alex is a data scientist analyzing the relationship between two financial indicators over time. He has collected two time series datasets

representing daily values of these indicators over several months. Alex wants to understand how these two indicators correlate at different time lags to identify possible leading or lagging behaviors.

Your task is to help Alex compute the cross-correlation of these two time series using numpy, so he can analyze the similarity between the two signals at various time shifts.

Input Format

The first line of input consists of space-separated float values representing the first time series, array1.

The second line of input consists of space-separated float values representing the second time series, array2.

Output Format

The first line of output prints: "Cross-correlation of the two time series:"

The second line of output prints: the 1D numpy array cross_corr representing the cross-correlation of array1 and array2 across different lags.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 1.0 2.0 3.0
4.0 5.0 6.0
Output: Cross-correlation of the two time series:
[6. 17. 32. 23. 12.]

Answer

```
# You are using Python
import numpy as np

# Read the two time series from input
array1 = np.array(list(map(float, input().split())))
array2 = np.array(list(map(float, input().split())))

# Compute the cross-correlation using 'full' mode
```

```
cross_corr = np.correlate(array1, array2, mode='full')

# Center the result to show correlation by lag (optional: align center index)
# We're only printing it as-is per the sample format

print("Cross-correlation of the two time series:")
print(cross_corr)
```

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

4. Problem Statement

A company tracks the monthly sales data of various products. You are given a table where each row represents a product and each column represents its monthly sales in sequential months.

Your task is to compute the cumulative monthly sales for each product using numpy, where the cumulative sales for a month is the total sales from month 1 up to that month.

Input Format

The first line of input consists of two integer values, products and months, separated by a space.

Each of the next products lines consists of months integer values representing the monthly sales data of a product.

Output Format

The first line of output prints: "Cumulative Monthly Sales:"

The second line of output prints: the 2D numpy array cumulative_array that contains the cumulative sales data for each product.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 2 4

10 20 30 40
5 15 25 35

Output: Cumulative Monthly Sales:
[[10 30 60 100]
 [5 20 45 80]]

Answer

```
# You are using Python
import numpy as np

# Read number of products and months
products, months = map(int, input().split())

# Read sales data for each product
sales_data = [list(map(int, input().split())) for _ in range(products)]

# Convert to NumPy array
sales_array = np.array(sales_data)

# Compute cumulative sum along months (axis=1)
cumulative_array = np.cumsum(sales_array, axis=1)

# Print results
print("Cumulative Monthly Sales:")
print(cumulative_array)
```

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

5. Problem Statement

Sita is analyzing her company's daily sales data to find all sales values that are multiples of 5 and exceed 100. She wants to filter these specific sales values from the list.

Help her to implement the task using the numpy package.

Formula:

To filter sales values:

Select all values s from sales such that (s % 5 == 0) and (s > 100)

Input Format

The first line of input consists of an integer value, n, representing the number of sales entries.

The second line of input consists of n floating-point values, sales, separated by spaces, representing daily sales figures.

Output Format

The output prints: filtered_sales

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 5
50.0 100.0 105.0 150.0 99.0
Output: [105. 150.]

Answer

```
# You are using Python
import numpy as np

# Read number of sales entries
n = int(input())

# Read the sales values
sales = np.array(list(map(float, input().split())))

# Apply the filter: multiples of 5 and greater than 100
filtered_sales = sales[(sales % 5 == 0) & (sales > 100)]

# Print the filtered result
print(filtered_sales)
```

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