Rajalakshmi Engineering College

Name: shri vishal.sv
Email: 241501202@rajalakshmi.edu.in
Roll no: 241501202
Phone: 9342648703
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 1_MCQ

Attempt : 1
Total Mark : 15
Marks Obtained : 14

Section 1 : MCQ

1. What is the output of the below expression?

print(3*1**3)

Answer

3

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

2. Which of the following expressions results in an error?

Answer

int('10.8')

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

3. What is the output of the following number conversion?

```
z = complex(1.25)
print(z)
```

Answer

(1.25+0j)

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

4. What is the value of the following expression?

```
float(22//3+3/3)
```

Answer

8.0

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

5. What will the following code output?

```
z = 3 + 4j
print(abs(z))
```

Answer

5.0

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

6. Which of these is not a core data type?

Answer

Class

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

7. What will be the value of the following Python expression?

```
4 + 3 % 5
```

Answer

7

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

8. The value of the expressions 4/(3*(2-1)) and 4/3*(2-1) is the same. True or False?

Answer

True

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

9. What is the output of the following program?

```
print((1, 2) + (3, 4))
```

Answer

(1, 2, 3, 4)

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

10. Which of the following represents the bitwise XOR operator?

Answer

|

Status : Wrong ***Marks : 0/1***

11. Which is the correct operator for power(xy)?

Answer

x**y

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

12. What will be the output of the following code?

```
x = int(34.56 - 2 * 2)
print(x)
```

Answer

30

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

13. Which of the following functions converts a string to a float in Python?

Answer

float(x)

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

14. Evaluate the expression given below if A= 16 and B = 15

A % B // A

Answer

0

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

15. What is the return type of the function id?

Answer

int

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

Rajalakshmi Engineering College

Name: shri vishal.sv
Email: 241501202@rajalakshmi.edu.in
Roll no: 241501202
Phone: 9342648703
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 4_COD_Updated

Attempt : 1
Total Mark : 50
Marks Obtained : 50

Section 1 : Coding

1. 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

```
def analyze_string(input_string):

    uppercase_count = 0
    lowercase_count = 0
    digit_count = 0
    special_count = 0

    for char in input_string:
        if char.isupper():
            uppercase_count += 1
        elif char.islower():
            lowercase_count += 1
        elif char.isdigit():
            digit_count += 1
        else:
            special_count += 1
```

```
    return uppercase_count, lowercase_count, digit_count, special_count

input_string = input()
uppercase_count, lowercase_count, digit_count, special_count =
analyze_string(input_string)

print("Uppercase letters:", uppercase_count)
print("Lowercase letters:", lowercase_count)
print("Digits:", digit_count)
print("Special characters:", special_count)
```

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

2. 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

```
# You are using Python
def is_armstrong_number(number):
  # Convert the number to string to easily iterate over digits
  str_number = str(number)
  num_digits = len(str_number)

   # Calculate the sum of each digit raised to the power of num_digits
  armstrong_sum = sum(int(digit) ** num_digits for digit in str_number)

  # Check if the calculated sum is equal to the original number
  return armstrong_sum == number

def armstrong_number(number):
  if is_armstrong_number(number):
     print(f"{number} is an Armstrong number.")
  else:
     print(f"{number} is not an Armstrong number.")

# Example usage:
n = int(input())
armstrong_number(n)
```

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

3. 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

```
# You are using Pytho
a=input()
b=len(a)
print(b)
```

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

4. 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

```
a = int(input())
b = int(input())


RES = pow(a, b)


POW= [pow(a, i) for i in range(1, b + 1)]


SUM = sum(POW)


print(RES)
print(POW)
print(SUM)
```

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

5. 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

```
def check_starts_with(main_string, substring):
    starts_with = lambda s, sub: s.startswith(sub)

    return starts_with(main_string, substring)

main_string = input().strip()
substring = input().strip()

result = check_starts_with(main_string, substring)
```

```
print(result)
```

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

Rajalakshmi Engineering College

Name: shri vishal.sv
Email: 241501202@rajalakshmi.edu.in
Roll no: 241501202
Phone: 9342648703
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 5_COD

Attempt : 1
Total Mark : 50
Marks Obtained : 46

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

```
n = int(input())
result = {}

for _ in range(n):
    customer_data = list(map(int, input().split()))
    customer_id = customer_data[0]
    amounts = customer_data[1:]
    total = sum(amounts)
    max_spent = max(amounts)
    result[customer_id] = [total, max_spent]

print(result)
```

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

2. 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
# Read the number of tuples
```

```
N = int(input())

filtered_quantities = []

for _ in range(N):
    item = eval(input())
    item_id, quantities = item


    filtered_quantities.extend(quantities)


threshold = int(input())


filtered_quantities = list(filter(lambda x: x > threshold, filtered_quantities))


print(" ".join(map(str, filtered_quantities)))
```

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

```
registered = set(map(int, input().split()))
attended = set(map(int, input().split()))
dropped_out = set(map(int, input().split()))

registered_and_attended = registered & attended

final_students = registered_and_attended - dropped_out

print(registered_and_attended)
print(final_students)
```

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

4. 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

```
n = int(input())

list1 = list(map(int, input().split(', ')))
list2 = list(map(int, input().split(', ')))

sum_tuple = tuple(a + b for a, b in zip(list1, list2))

print(sum_tuple)
```

Status : Partially correct ***Marks : 6/10***

5. 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

```
from collections import Counter


n = int(input())

product_ids = []


for _ in range(n):
    product_ids.append(int(input()))


frequency_dict = dict(Counter(product_ids))
```

```
unique_count = len(frequency_dict)
total_frequency = sum(frequency_dict.values())
average_frequency = total_frequency / unique_count if unique_count > 0 else 0
print(frequency_dict)
print(f"Total Unique IDs: {unique_count}")
print(f"Average Frequency: {average_frequency:.2f}")
```

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

Rajalakshmi Engineering College

Name: shri vishal.sv
Email: 241501202@rajalakshmi.edu.in
Roll no: 241501202
Phone: 9342648703
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

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:
        # Read and convert inputs
        price = float(input())
        quantity = int(input())

        # 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

        # Check 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 ve:
        print(ve)
    except RuntimeError as re:
        print(re)

# Execute the function
calculate_total_cost()
```

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
a=int(input())
if a>=18:
   print("Eligible to vote")
else:
   print("Not eligible to vote")
```

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

3. 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 = input()
    n = int(n_input)

    if n <= 0:
        print("Error: The length of the list must be a non-negative integer.")
    elif n > 20:
        print("Error: The length of the list must not exceed 20.")
    else:
        numbers = []
        for _ in range(n):
            try:
                num_input = input()
                num = int(num_input)
                numbers.append(num)
            except ValueError:
                print("Error: You must enter a numeric value.")
                exit()

        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***

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

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


with open("sentence_file.txt", "r") as file:
    content = file.read()
    words = content.strip().split()
    print(len(words))
```

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

```
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} Upper-Case String: {upper_case_string}
Lower-Case String: {lower_case_string}")
```

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

Rajalakshmi Engineering College

Name: shri vishal.sv
Email: 241501202@rajalakshmi.edu.in
Roll no: 241501202
Phone: 9342648703
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 7_COD

Attempt : 1
Total Mark : 50
Marks Obtained : 41

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
```

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

# Create MultiIndex
index = pd.MultiIndex.from_arrays([cities, months], names=['City', 'Month'])

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

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

# Group by City and calculate total sales
total_sales = df.groupby('City').sum()

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 numpy as np

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

data = []
for _ in range(n):
    line = input().split(',')
    data.append(line)

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

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

# Calculate mean age excluding NaN
mean_age = df['Age'].mean()

# Fill missing ages with mean
df['Age'].fillna(mean_age, inplace=True)

# Remove rows with missing Diagnosis
df = df.dropna(subset=['Diagnosis'])

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

# Output
print("Cleaned Hospital Records:")
print(cleaned_df)
```

Status : Partially correct **Marks : 1/10**

3. 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 input data
products, months = map(int, input().split())
sales_data = []

for _ in range(products):
    sales_row = list(map(int, input().split()))
```

```
    sales_data.append(sales_row)

# Convert sales data to a numpy array
sales_array = np.array(sales_data)

# Compute cumulative sales
cumulative_array = sales_array.cumsum(axis=1)

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

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

4. 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 the number of sales entries
n = int(input())

# Read the sales figures and convert them into a numpy array
sales = np.array(list(map(float, input().split())))

# Filter sales values that are multiples of 5 and exceed 100
filtered_sales = sales[(sales % 5 == 0) & (sales > 100)]

# Print the filtered sales values
print(filtered_sales)
```

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

5. 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 input
array1 = np.array(list(map(float, input().strip().split())))
array2 = np.array(list(map(float, input().strip().split())))

# Compute cross-correlation using numpy's correlate function
cross_corr = np.correlate(array1, array2, mode='full')

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

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