Parul University

Name: Rajveersinh Desai
Email: 2203031080071@paruluniversity.ac.in
Roll no: 2203031080071
Phone: 9773227698
Branch: Parul University
Department: ITB_Batch 2
Batch: 2026
Degree: B.Tech - IT

Scan to verify results

PIET_Machine learning using python_Course

PIET_Machine Learning using Python_Week 1_CY

Attempt : 1
Total Mark : 30
Marks Obtained : 30

Section 1 : Coding

1. Problem Statement

Kamal is working on a data transformation tool in a manufacturing unit where he needs to process a 1D array of sensor readings. His task is to create an extended version of the array by appending its reverse to itself and then reshaping it into a 2D array with a single row.

Write a program to help Kamal by taking the size and elements of a 1D array, reversing the array, appending the reversed version to the original, and reshaping the result into a single row 2D array using NumPy.

Input Format

The first line of input is an integer, n, representing the size of the 1D array.

The next n lines of input consist of integers, each representing an element of the

array.

Output Format

The first line prints "Original array: "

The second line prints a 1D array printed using standard NumPy formatting

The third line prints "Expanded array: "

The fourth line prints the expanded array printed using standard NumPy formatting (2D array with one row)

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 3
10
60
70
Output: Original array:
[10 60 70]
Expanded array:
[[10 60 70 70 60 10]]

Answer

```
import numpy as np
def solve():
    n = int(input())
    elements = []
    for _ in range(n):
        elements.append(int(input()))

    original_array = np.array(elements)

    print("Original array:")
    print(original_array)

    reversed_array = np.flip(original_array)
```

```
    concatenated_array = np.concatenate((original_array, reversed_array))

    expanded_array = np.expand_dims(concatenated_array, axis=0)

    print("Expanded array:")
    print(expanded_array)

solve()
```

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

2. Problem Statement

Ava is analyzing electricity consumption (in kWh) data collected from various households. She wants to group the data into standard intervals and summarize the frequency and average usage for each interval. Additionally, she needs to compute the overall statistics, including mean, variance, standard deviation, and total count of values.

Write a program that classifies the input consumption values into the intervals: 0–100, 101–200, 201–300, 301–400, and 400+. For each interval, compute the frequency and mean consumption, rounded to 2 decimal places. If no values fall in an interval, display '-' for the mean. The program must also compute the overall mean, variance, standard deviation (sample), and the total number of values, and present all results in a structured format.

Input Format

The input consists of a single line containing space-separated integers representing electricity consumption values (kWh).

Output Format

First line prints "Electricity Consumption Summary"

Then, a table with three columns:

- Interval: one of "0-100", "101-200", "201-300", "301-400", "400+", "Total"
- Frequency: number of values in that interval
- Mean Consumption: mean of values in that interval, rounded to 2 decimal places, or '-' if none

After the table, print:

- "Overall Mean Consumption: <value> kWh"
- "Variance: <value>"
- "Standard Deviation: <value>"
- The mean in the Total row should reflect the overall mean of all input values, rounded to 2 decimal places.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 10 50 90 110 150 200 220 250

Output: Electricity Consumption Summary

Interval	Frequency	Mean Consumption
0-100	3	50.0
101-200	3	153.33
201-300	2	235.0
301-400	0	-
400+	0	-
Total	8	135.0

Overall Mean Consumption: 135.0 kWh
Variance: 7200.0
Standard Deviation: 84.85

Answer

```
import pandas as pd
import numpy as np
def analyze_electricity_consumption(consumption_values):
    intervals = {
        "0-100" : [],
        "101-200": [],
        "201-300": [],
```

```
        "301-400": [],
        "400+" : []
    }
    for val in consumption_values:
        if 0<= val <= 100:
            intervals["0-100"].append(val)
        elif 101 <= val <= 200:
            intervals["101-200"].append(val)
        elif 201 <= val <= 300:
            intervals["201-300"].append(val)
        elif 301 <= val <= 400:
            intervals["301-400"].append(val)
        elif val > 400:
            intervals["400+"].append(val)
    data = []
    for interval, values in intervals.items():
        frequency = len(values)
        mean_consumption = round(np.mean(values),2) if values else '-'
        data.append([interval, frequency, mean_consumption])
    total_count = len(consumption_values)
    overall_mean = round(np.mean(consumption_values), 2) if
consumption_values else 0.0
    data.append(["Total",total_count, overall_mean])
    df = pd.DataFrame(data, columns = ["Interval", "Frequency", "Mean
Consumption"])
    print("Electricity Consumption Summary")
    print(df.to_string(index=False))
    if consumption_values:
        overall_variance = round(np.var(consumption_values, ddof=1), 2)
        overall_std_dev = round(np.std(consumption_values, ddof=1), 2)
    else:
        overall_variance = 0.0
        overall_std_dev = 0.0
    print(f"\nOverall Mean Consumption: {overall_mean:} kwh")
    print(f"Variance: {overall_variance:}")
    print(f"Standard Deviation: {overall_std_dev:}")


input_line = input()
consumption_values = list(map(int, input_line.split()))
analyze_electricity_consumption(consumption_values)
```

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

3. Problem Statement

Rohan has a 2D NumPy array with a specific number of rows and columns. He wants to sort this array in three different ways:

Sort each row of the array in ascending order.Flatten the array and sort it in ascending order.Sort each column of the array in ascending order.

Write a program to help Rohan sort the array in these three ways.

Input Format

The first line contains two space-separated integers, n and m, representing the number of rows and columns of the array, respectively.

The next n lines contain m space-separated integers, each representing the elements of the array.

Output Format

The first line of output prints "Sorted based on last axis(1):" followed by the array sorted row-wise.

The second line prints "Sorted flattened array:" followed by the flattened and sorted array.

The third line prints "Sorted based on axis 0:" followed by the array sorted column-wise.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 3 3
3 10 2
1 5 7
2 7 5
Output: Sorted based on last axis(1):
[[2 3 10]
 [1 5 7]
 [2 5 7]]
Sorted flattened array:
[1 2 2 3 5 5 7 7 10]
Sorted based on axis 0:
[[1 5 2]
 [2 7 5]
 [3 10 7]]

Answer

```
import numpy as np

def solve():
    n, m = map(int, input().split())

    arr = []
    for _ in range(n):
        arr.append(list(map(int, input().split())))

    np_arr = np.array(arr)

    sorted_row_wise = np.sort(np_arr, axis = 1)
    print("Sorted based on last axis(1):")
    print(sorted_row_wise)

    flattened_sorted = np.sort(np_arr.flatten())
    print("Sorted flattened array:")
    print(flattened_sorted)
```

```
    sorted_column_wise = np.sort(np_arr, axis = 0)
    print("Sorted based on axis 0:")
    print(sorted_column_wise)

solve()
```

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