Rajalakshmi Engineering College

Name: Vishaali S
Email: 240701596@rajalakshmi.edu.in
Roll no: 240701596
Phone: 7550088033
Branch: REC
Department: I CSE FF
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23221_Python Programming

REC_Python_Week 7_MCQ

Attempt : 1
Total Mark : 20
Marks Obtained : 19

Section 1 : MCQ

1. What is the output of the following NumPy code snippet?

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5])
r = arr[arr > 2]
print(r)
```

Answer

[3 4 5]

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

2. Which NumPy function is used to calculate the standard deviation of an array?

Answer

numpy.std()

Status : Correct *Marks : 1/1*

3. What is the purpose of the following NumPy code snippet?

```
import numpy as np
arr = np.zeros((3, 4))
print(arr)
```

Answer

Displays a 3x4 matrix filled with zeros

Status : Correct *Marks : 1/1*

4. What is the output of the following NumPy code?

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5])
r = arr[2:4]
print(r)
```

Answer

[3 4]

Status : Correct *Marks : 1/1*

5. The important data structure of pandas is/are ____.

Answer

None of the mentioned options

Status : Wrong *Marks : 0/1*

6. Which NumPy function is used to create an identity matrix?

Answer

numpy.identity()

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

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

```
import pandas as pnd
pnd.Series([1,2], index= ['a','b','c'])
```

Answer

Value Error

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

8. What does NumPy stand for?

Answer

Numerical Python

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

9. What does the np.arange(10) function in NumPy do?

Answer

Creates an array with values from 1 to 9

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

10. What will be the output of the following code snippet?

```
import numpy as np
arr = np.array([1, 2, 3])
result = np.concatenate((arr, arr))
print(result)
```

Answer

```
[1 2 3 1 2 3]
```

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

11. In NumPy, how do you access the first element of a one-dimensional array arr?

Answer

arr[0]

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

12. What is the result of the following NumPy operation?

```
import numpy as np
arr = np.array([1, 2, 3])
r = arr + 5
print(r)
```

Answer

[6 7 8]

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

13. In the DataFrame created in the code, what is the index for the row containing the data for 'Jack'?

```
import pandas as pd

data = {'Name': ['Tom', 'Jack', 'nick', 'juli'],
   'marks': [99, 98, 95, 90]}

df = pd.DataFrame(data, index=['rank1',
            'rank2',
            'rank3',
            'rank4'])
print(df)
```

Answer

rank2

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

14. Which function is used to create a Pandas DataFrame?

Answer

pd.DataFrame()

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

15. Which of the following is a valid way to import NumPy in Python?

Answer

import numpy as np

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

16. What is the primary purpose of Pandas DataFrame?

Answer

To store data in tabular form for analysis and manipulation

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

17. Which NumPy function is used to find the indices of the maximum and minimum values in an array?

Answer

argmax() and argmin()

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

18. Minimum number of argument we require to pass in pandas series ?

Answer

1

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

19. What is the output of the following code?

```
import numpy as np
a = np.arange(10)
print(a[2:5])
```

Answer

[2, 3, 4]

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

20. What is the primary data structure used in NumPy for numerical computations?

Answer

Array

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

Rajalakshmi Engineering College

Name: Vishaali S
Email: 240701596@rajalakshmi.edu.in
Roll no: 240701596
Phone: 7550088033
Branch: REC
Department: I CSE FF
Batch: 2028
Degree: B.E - CSE

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 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
k=input()
l=list(map(float,input().split()))
n=np.array(l)




fl=[n[i] for i in range(len(n)) if n[i]>100.0 ]


print('[',end="")
for i in range(len(fl)):
   print(f"{fl[i]:.0f}. ",end="")
print(']',end="")
```

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

2. 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 pandas as pd
import numpy as np

n, m = map(int, input().split())
```

```
data = [list(map(int, input().split())) for i in range(n)]

p = pd.DataFrame(data)
cd = p.cumsum(axis=1)

print("Cumulative Monthly Sales:")
print(cd.to_numpy())
```

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

3. 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())

if n == 0:
    print("Monthly Sales Data with MultiIndex:")
    print(pd.DataFrame(columns=["Sales"]))
    print("\nTotal Sales Per City:")
    print(pd.DataFrame(columns=["Sales"]))
    exit()

cities = input().split()
months = input().split()
sales = list(map(float, input().split()))
```

```
# Validate lengths
if not (len(cities) == len(months) == len(sales) == n):
    raise ValueError("Input lengths do not match 'n'.")

# Create MultiIndex and DataFrame
index = pd.MultiIndex.from_arrays([cities, months], names=["City", "Month"])
df = pd.DataFrame({"Sales": sales}, index=index)

# Print multi-indexed DataFrame
print("Monthly Sales Data with MultiIndex:")
print(df)

# Group and print total sales
print("\nTotal Sales Per City:")
print(df.groupby("City").sum())
```

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

4. 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
import sys

# Input
n = int(input())
```

```
columns = input().split(',')

data = []
for _ in range(n):
    row = input().split(',')
    # Fill missing values as NaN
    row = [val if val.strip() != '' else np.nan for val in row]
    # If row has fewer values than columns, pad with NaN
    while len(row) < len(columns):
        row.append(np.nan)
    data.append(row)

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

# Handle Age column
if 'Age' in df.columns:
    try:
        df['Age'] = pd.to_numeric(df['Age'], errors='coerce')  # Convert to float, set
bad data to NaN
        if df['Age'].notna().sum() > 0:
            mean_age = df['Age'].mean()
            df['Age'] = df['Age'].fillna(mean_age)
    except Exception as e:
        print("Error processing 'Age' column:", e)
        sys.exit()
else:
    print("No 'Age' column found.")
    sys.exit()

# Handle missing Diagnosis
if 'Diagnosis' in df.columns:
    df = df[df['Diagnosis'].notna()]
else:
    print("No 'Diagnosis' column found.")
    sys.exit()

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

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

print(df)

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

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

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

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

Rajalakshmi Engineering College

Name: Vishaali S
Email: 240701596@rajalakshmi.edu.in
Roll no: 240701596
Phone: 7550088033
Branch: REC
Department: I CSE FF
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23221_Python Programming

REC_Python_Week 7_PAH

Attempt : 1
Total Mark : 50
Marks Obtained : 48.5

Section 1 : Coding

1. Problem Statement

You're analyzing the daily returns of a set of financial assets over a period of time. Each day is represented as a row in a 2D array, where each column represents the return of a specific asset on that day.

Your task is to identify which days had all positive returns across every asset using numpy, and output a boolean array indicating these days.

Input Format

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

Each of the next rows lines consists of cols float values representing the returns of the assets for that day.

Output Format

The first line of output prints: "Days where all asset returns were positive:"

The second line of output prints: the boolean array positive_days, indicating True for days where all asset returns were positive and False otherwise.

Refer to the sample output for the formatting specifications.

Sample Test Case

```
Input: 3 4
0.01 0.02 0.03 0.04
0.05 0.06 0.07 0.08
-0.01 0.02 0.03 0.04
Output: Days where all asset returns were positive:
[ True  True False]
```

Answer

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

# Read dimensions
rows, cols = map(int, input().split())

# Read matrix of returns
data = [list(map(float, input().split())) for _ in range(rows)]
returns = np.array(data)

# Identify days where all asset returns are positive
positive_days = np.all(returns > 0, axis=1)

# Output
print("Days where all asset returns were positive:")
print(positive_days)
```

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

2. Problem Statement

Arjun is a data scientist working on an image processing task. He needs to normalize the pixel values of a grayscale image matrix to scale between 0 and 1. The input image data is provided as a matrix of integers.

Help him to implement the task using the numpy package.

Formula:

To normalize each pixel value in the image matrix:

normalized_pixel = (pixel - min_pixel) / (max_pixel - min_pixel)

where min_pixel and max_pixel are the minimum and maximum pixel values in the image matrix, respectively. If all pixel values are the same, the normalized image matrix should be filled with zeros.

Input Format

The first line of input consists of an integer value, rows, representing the number of rows in the image matrix.

The second line of input consists of an integer value, cols, representing the number of columns in the image matrix.

The next rows lines each consist of cols integer values separated by a space, representing the pixel values of the image matrix.

Output Format

The output prints: normalized_image

Refer to the sample output for the formatting specifications.

Sample Test Case

```
Input: 2
3
1 2 3
4 5 6
Output: [[0.  0.2 0.4]
 [0.6 0.8 1. ]]
```

Answer

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

# Input
rows = int(input())
cols = int(input())
data = [list(map(int, input().split())) for _ in range(rows)]
image = np.array(data)

# Compute min and max
min_pixel = np.min(image)
max_pixel = np.max(image)

# Normalize using given formula, handle edge case where min == max
if min_pixel == max_pixel:
    normalized_image = np.zeros((rows, cols), dtype=float)
else:
    normalized_image = (image - min_pixel) / (max_pixel - min_pixel)

# Output
print(normalized_image)
```

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

3. Problem Statement

Arjun manages a busy customer service center and wants to analyze the distribution of customer wait times to improve service efficiency. He decides to group the wait times into intervals of 5 minutes each and count how many customers fall into each interval bucket.

Help him implement this bucketing and counting task using NumPy.

Bucketing Logic:

Divide the wait times into intervals (buckets) of size 5 minutes, e.g.:

[0–5), [5–10), [10–15), …

Use NumPy's digitize function to determine which bucket each wait time falls into.

Count the number of wait times in each bucket and generate bucket labels.

Input Format

The first line contains an integer n, the number of customer wait times recorded.

The second line contains n space-separated floating-point numbers representing the wait times (in minutes).

Output Format

The first line of output is the text:

Wait Time Buckets and Counts:

Each subsequent line prints the bucket range and the number of wait times in that bucket, formatted as:

<bucket_range>: <count>

where <bucket_range> is the lower and upper bound of the bucket (inclusive lower bound, exclusive upper bound), for example:

0-5: 3

5-10: 2

10-15: 1

The output uses the default string formatting of Python's print() function (no extra spaces, no special formatting beyond the specified lines).

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 10

2.0 3.0 7.0 8.0 12.0 14.0 18.0 19.0 21.0 25.0
Output: Wait Time Buckets and Counts:
0-5: 2
5-10: 2
10-15: 2
15-20: 2
20-25: 1

Answer

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

# Input
n = int(input())
wait_times = np.array(list(map(float, input().split())))

# Determine the highest wait time
max_time = np.max(wait_times)

# Calculate number of required buckets
num_buckets = math.ceil(max_time / 5)

# Create bin edges (e.g., [0, 5, 10, ..., 25])
bins = np.arange(0, (num_buckets + 1) * 5, 5)

# Digitize the wait times
bucket_indices = np.digitize(wait_times, bins, right=False)

# Count occurrences per bucket index
counts = np.bincount(bucket_indices, minlength=len(bins))

# Output
print("Wait Time Buckets and Counts:")
for i in range(1, num_buckets + 1):  # only up to the last required bucket
    lower = bins[i - 1]
    upper = bins[i]
    print(f"{lower}-{upper}: {counts[i]}")
```

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

4. Problem Statement

A software development company wants to classify its employees based on their years of service at the company. They want to categorize employees into three experience levels: Junior (less than 3 years), Mid (3 to 6 years, inclusive), and Senior (more than 6 years).

Experience Level Classification:

Junior: Years at Company < 3

Mid: 3 ≤ Years at Company < 6

Senior: Years at Company > 5

You need to create a Python program using the pandas library that reads employee data, processes it into a DataFrame, and adds a new column "Experience Level" to display the appropriate classification for each employee.

Input Format

First line: an integer n representing the number of employees.

Next n lines: each line has a string Name and a floating-point number Years at Company (space-separated).

Output Format

First line: "Employee Data with Experience Level:"

The employee data table printed with no index column, and with columns: Name, Years at Company, Experience Level.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 5
Alice 2
Bob 4

```
Charlie 7
Diana 3
Evan 6
```

```
Output: Employee Data with Experience Level:
   Name  Years at Company Experience Level
  Alice              2.0           Junior
    Bob              4.0              Mid
Charlie              7.0           Senior
  Diana              3.0              Mid
   Evan              6.0           Senior
```

Answer

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

# Read input
n = int(input())
data = [input().split() for _ in range(n)]

# Extract Name and Years
names = [row[0] for row in data]
years = [float(row[1]) for row in data]

# Create DataFrame
df = pd.DataFrame({
    "Name": names,
    "Years at Company": years
})

# Corrected classification function
def classify(years):
    if years < 3:
        return "Junior"
    elif years < 6:
        return "Mid"
    else:
        return "Senior"

df["Experience Level"] = df["Years at Company"].apply(classify)

# Print output
print("Employee Data with Experience Level:")
print(df.to_string(index=False))
```

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

5. Problem Statement

A company conducted a customer satisfaction survey where each respondent provides their RespondentID and an optional textual Feedback. Sometimes, respondents submit their ID without any feedback or with empty feedback.

Your task is to process the survey responses using pandas to replace any missing or empty feedback with the phrase "No Response". Finally, print the cleaned survey responses exactly as shown in the sample output.

Input Format

The first line contains an integer n, the number of survey responses.

Each of the next n lines contains:

A RespondentID (a single alphanumeric string without spaces),

Followed optionally by a Feedback string, which may be empty or missing.

If no feedback is provided after the RespondentID, treat it as missing.

Output Format

Print the line:

Survey Responses with Missing Feedback Filled:

Then print the cleaned survey data as a table with two columns: RespondentID and Feedback.

The table should have the headers exactly as:

RespondentID Feedback

Print each respondent's data on a new line, aligned to match the output produced by pandas.DataFrame.to_string(index=False).

For any missing or empty feedback, print "No Response" in the Feedback column.

Maintain the spacing and alignment exactly as shown in the sample outputs.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 4
101 Great service
102
103 Loved it
104

Output: Survey Responses with Missing Feedback Filled:
```
RespondentID      Feedback
         101 Great service
         102   No Response
         103      Loved it
         104   No Response
```

Answer

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

# Read number of records
n = int(sys.stdin.readline())

data = []
for _ in range(n):
    line = sys.stdin.readline().rstrip()
    # Split only once: RespondentID and optional feedback
    parts = line.split(maxsplit=1)

    respondent_id = parts[0]
    feedback = parts[1].strip() if len(parts) > 1 and parts[1].strip() != '' else "No 
Response"

    data.append([respondent_id, feedback])

# Create DataFrame
df = pd.DataFrame(data, columns=["RespondentID", "Feedback"])

# Print as expected
print("Survey Responses with Missing Feedback Filled:")
print(df.to_string(index=False))
```

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

Rajalakshmi Engineering College

Name: Vishaali S
Email: 240701596@rajalakshmi.edu.in
Roll no: 240701596
Phone: 7550088033
Branch: REC
Department: I CSE FF
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23221_Python Programming

REC_Python_Week 7_CY

Attempt : 1
Total Mark : 50
Marks Obtained : 47.5

Section 1 : Coding

1. Problem Statement

Arjun is developing a system to monitor environmental sensors installed in different rooms of a smart building. Each sensor records multiple temperature readings throughout the day. To compare sensor data fairly despite differing scales, Arjun needs to normalize each sensor's readings so that they have a mean of zero and standard deviation of one.

Help him implement this normalization using numpy.

Normalization Formula:

Input Format

The first line of input consists of two integers: sensors (number of sensors) and samples (number of readings per sensor).

The next sensors lines each contain samples space-separated floats representing the sensor readings.

Output Format

The first line of output prints: "Normalized Sensor Data:"

The next lines print the normalized readings as a numpy array, where each row corresponds to a sensor's normalized values.

Refer to the sample output for the formatting specifications.

Sample Test Case

```
Input: 3 3
1.0 2.0 3.0
4.0 5.0 6.0
7.0 8.0 9.0
Output: Normalized Sensor Data:
[[-1.22474487  0.          1.22474487]
 [-1.22474487  0.          1.22474487]
 [-1.22474487  0.          1.22474487]]
```

Answer

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

# Read input
sensors, samples = map(int, input().split())
data = []

for _ in range(sensors):
    readings = list(map(float, input().split()))
    data.append(readings)

# Convert to NumPy array
arr = np.array(data)

# Calculate mean and std along each row (axis=1)
```

```
means = arr.mean(axis=1, keepdims=True)
stds = arr.std(axis=1, keepdims=True)

# Avoid division by zero: set stds to 1 where std is 0
stds[stds == 0] = 1

# Normalize
normalized = (arr - means) / stds

# Output
print("Normalized Sensor Data:")
print(normalized)
```

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

2. Problem Statement

Rekha is a meteorologist analyzing rainfall data collected over 5 years, with monthly rainfall recorded for each year. She wants to find the total rainfall each year and also identify the month with the maximum rainfall for every year.

Help her to implement the task using the numpy package.

Formula:

Yearly total rainfall = sum of all 12 months' rainfall for each year

Month with max rainfall = index of the maximum rainfall value within the 12 months for each year (0-based index)

Input Format

The input consists of 5 lines.

Each line contains 12 floating-point values separated by spaces, representing the rainfall data (in mm) for each month of that year.

Output Format

The first line of output prints: yearly_totals

The second line of output prints: max_rainfall_months

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  7.0  8.0  9.0  10.0  11.0  12.0
2.0  3.0  4.0  5.0  6.0  7.0  8.0  9.0  10.0 11.0  12.0  13.0
3.0  4.0  5.0  6.0  7.0  8.0  9.0  10.0 11.0 12.0  13.0  14.0
4.0  5.0  6.0  7.0  8.0  9.0  10.0 11.0 12.0 13.0  14.0  15.0
5.0  6.0  7.0  8.0  9.0  10.0 11.0 12.0 13.0 14.0  15.0  16.0
Output: [ 78.  90. 102. 114. 126.]
[11 11 11 11 11]
```

Answer

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

# Read 5 lines of input with 12 float values each
data = []
for _ in range(5):
    row = list(map(float, input().split()))
    data.append(row)

# Convert to NumPy array
rainfall = np.array(data)

# Compute total rainfall per year (row-wise sum)
yearly_totals = np.sum(rainfall, axis=1)

# Compute month with max rainfall per year (row-wise argmax)
max_rainfall_months = np.argmax(rainfall, axis=1)

# Output
print(yearly_totals)
print(max_rainfall_months)
```

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

3. Problem Statement

Rekha works as an e-commerce data analyst. She receives transaction data containing purchase dates and needs to extract the month and day from these dates using the pandas package.

Help her implement this task by performing the following steps:

Convert the Purchase Date column to datetime format, treating invalid date entries as NaT (missing).

Create two new columns:

Purchase Month, containing the month (as an integer) extracted from the Purchase Date.

Purchase Day, containing the day (as an integer) extracted from the Purchase Date. Keep the rest of the data as is.

Input Format

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

The second line contains the CSV header — comma-separated column names.

The next n lines each contain a transaction record in comma-separated format.

Output Format

The first line of output is the text:

Transformed E-commerce Transaction Data:

The next lines print the pandas DataFrame with:

The original columns (including Purchase Date, which is now in datetime format or NaT if invalid).

Two additional columns: Purchase Month and Purchase Day.

The output uses the default pandas DataFrame string representation as produced by print(transformed_df).

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 3
Customer,Purchase Date
Alice,2023-05-15
Bob,2023-06-20
Charlie,2023-07-01

Output: Transformed E-commerce Transaction Data:

```
  Customer Purchase Date  Purchase Month  Purchase Day
0    Alice    2023-05-15               5            15
1      Bob    2023-06-20               6            20
2  Charlie    2023-07-01               7             1
```

Answer

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

# Read number of records
n = int(input())

# Read header and records
data = [input().strip() for _ in range(n + 1)]  # includes header

# Convert to DataFrame
from io import StringIO
df = pd.read_csv(StringIO('\n'.join(data)))

# Convert 'Purchase Date' to datetime, invalid entries to NaT
df['Purchase Date'] = pd.to_datetime(df['Purchase Date'], errors='coerce')

# Create new columns
df['Purchase Month'] = df['Purchase Date'].dt.month
df['Purchase Day'] = df['Purchase Date'].dt.day

# Output
print("Transformed E-commerce Transaction Data:")
print(df)
```

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

4. Problem Statement

Arjun is monitoring hourly temperature data recorded continuously for multiple days. He needs to calculate the average temperature for each day based on 24 hourly readings.

Help him to implement the task using the numpy package.

Formula:

Reshape the temperature readings into rows where each row has 24 readings (one day).

Average temperature per day = mean of 24 hourly readings in each row.

Input Format

The first line of input consists of an integer value, n, representing the total number of temperature readings.

The second line of input consists of n floating-point values separated by spaces, representing hourly temperature readings.

Output Format

The output prints: avg_per_day

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 30
30.0 30.0 30.0 30.0 30.0 30.0 30.0 30.0 30.0 30.0 30.0 30.0 30.0 30.0 30.0 30.0
30.0 30.0 30.0 30.0 30.0 30.0 30.0 30.0
Output: [30.]

Answer

```
# You are using Python
```

```
import numpy as np

# Read number of readings
n = int(input())

# Read all temperature readings
temps = list(map(float, input().split()))

# Convert to numpy array
temp_array = np.array(temps)

# Reshape to rows of 24 readings each (1 row = 1 day)
reshaped = temp_array.reshape(-1, 24)

# Compute mean temperature per row/day
avg_per_day = reshaped.mean(axis=1)

# Output
print(avg_per_day)
```

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

5. Problem Statement

You are working as a data analyst for a small retail store that wants to track the stock levels of its products. Each product has a unique Name (such as "Toothpaste", "Shampoo", "Soap") and an associated Quantity in stock. Management wants to identify which products have zero stock so they can be restocked.

Write a Python program using the pandas library to help with this task. The program should:

Read the number of products, n.Read n lines, each containing the Name of the product and its Quantity, separated by a space.Convert this data into a pandas DataFrame.Identify and display the Name and Quantity of products with zero stock.If no products have zero stock, display: No products with zero stock.

Input Format

The first line contains an integer n, the number of products.

The next n lines each contain:

<Product_ID> <Quantity>

where <Product_ID> is a single word (e.g., "Shampoo") and <Quantity> is a non-negative integer (e.g., 5).

Output Format

The first line of output prints:

Products with Zero Stock:

If there are any products with zero stock, the following lines print the pandas DataFrame showing those products with two columns: Product_ID and Quantity.

The column headers Product_ID and Quantity are printed in the second line.

Each subsequent line shows the product's name and quantity, aligned under the respective headers, with no index column.

The output formatting (spacing and alignment) follows the default pandas to_string(index=False) style.

If no products have zero stock, print:

No products with zero stock.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 3
P101 10
P102 0
P103 5

Output: Products with Zero Stock:
```
Product_ID  Quantity
     P102        0
```

Answer

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

# Read number of products
n = int(input())

products = []

for _ in range(n):
    line = input().strip()
    if not line:
        continue
    parts = line.split()
    if len(parts) != 2:
        continue
    product_id, quantity_str = parts
    try:
        quantity = int(quantity_str)
        products.append((product_id, quantity))
    except ValueError:
        continue

# Create DataFrame
df = pd.DataFrame(products, columns=["Product_ID", "Quantity"])

# Filter products with zero stock
zero_stock_df = df[df["Quantity"] == 0]
```

```
print("Products with Zero Stock:")
if zero_stock_df.empty:
    print("No products with zero stock.")
else:
    print(zero_stock_df.to_string(index=False))
```

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