

# Rajalakshmi Engineering College

Name: Sweta Suresh  
Email: 240701549@rajalakshmi.edu.in  
Roll no: 2116240701549  
Phone: 9344119216  
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

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
```

```
n = int(sys.stdin.readline().strip())
```

```
header = sys.stdin.readline().strip().split(',')
```

```
data = [sys.stdin.readline().strip().split(',') for _ in range(n)]
```

```
df = pd.DataFrame(data, columns=header)
```

```
df['Purchase Date'] = pd.to_datetime(df['Purchase Date'], errors='coerce')
```

```
df['Purchase Month'] = df['Purchase Date'].dt.month
```

```
df['Purchase Day'] = df['Purchase Date'].dt.day
```

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

```
print(df)
```

**Status :** Correct

**Marks : 10/10**

## 2. 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
```

```
sensors, samples = map(int, input().split())
data = [list(map(float, input().split())) for _ in range(sensors)]
```

```
arr = np.array(data)
```

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

```
stds[stds == 0] = 1
```

```
normalized = (arr - means) / stds
```

```
print("Normalized Sensor Data:", normalized)
```

**Status :** Correct

**Marks :** 10/10

### 3. 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 total number of readings
n = int(input())
```

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

```
# Convert to numpy array
temps_arr = np.array(temps)
```

```
# Reshape into rows with 24 readings each (days x 24)
temps_reshaped = temps_arr.reshape(-1, 24)
```

```
# Calculate mean of each row (day)
avg_per_day = temps_reshaped.mean(axis=1)
```

```
# Print the result as numpy array
print(avg_per_day)
```

**Status :** Correct

**Marks :** 10/10

## **4. 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, each line has 12 floats
data = [list(map(float, input().split())) for _ in range(5)]
```

```
# Convert to numpy array of shape (5, 12)
rainfall = np.array(data)

# Calculate yearly total rainfall (sum of 12 months per year)
yearly_totals = rainfall.sum(axis=1)

# Find index of month with max rainfall per year (0-based)
max_rainfall_months = rainfall.argmax(axis=1)

# Print results as per output format
print(yearly_totals, max_rainfall_months)
```

**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

# You are using Python

```
import pandas as pd
```

```
import sys
```

```
n = int(sys.stdin.readline().strip())
```

```
data = []
```

```
for _ in range(n):
```

```
    line = sys.stdin.readline().strip()
```

```
    product_id, quantity_str = line.split()
```

```
    quantity = int(quantity_str)
```

```
    data.append([product_id, quantity])
```

```
df = pd.DataFrame(data, columns=["Product_ID", "Quantity"])
```

```
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