# Rajalakshmi Engineering College

Name: Arjunraj M
Email: 240701049@rajalakshmi.edu.in
Roll no: 240701049
Phone: 6381538294
Branch: REC
Department: I CSE FA
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 : 50

### Section 1 : Coding

1. 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***

```
import numpy as np

# Read input
rows = int(input())
cols = int(input())

# Read the matrix elements
matrix = []
for _ in range(rows):
    row = list(map(int, input().split()))
    matrix.append(row)

# Convert to numpy array
image = np.array(matrix, dtype=float)
```

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

# Normalize
if min_pixel == max_pixel:
    normalized_image = np.zeros_like(image)
else:
    normalized_image = (image - min_pixel) / (max_pixel - min_pixel)

# Print normalized image
print(normalized_image)
```

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

2. 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***

```
import pandas as pd

n = int(input())

data = []
for _ in range(n):
    line = input().strip()
    parts = line.split(maxsplit=1)
    respondent_id = parts[0]
    feedback = parts[1].strip() if len(parts) > 1 and parts[1].strip() else None
    data.append({'RespondentID': respondent_id, 'Feedback': feedback})


df = pd.DataFrame(data)

df['Feedback'] = df['Feedback'].fillna('No Response')

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

***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***

```
import numpy as np
import math


n = int(input())
wait_times = list(map(float, input().split()))


max_time = max(wait_times)
upper_limit = math.ceil(max_time / 5) * 5
bins = list(range(0, upper_limit + 5, 5))


indices = np.digitize(wait_times, bins, right=False)


counts = [0] * (len(bins) - 1)
for idx in indices:
    if 1 <= idx <= len(counts):
        counts[idx - 1] += 1


print("Wait Time Buckets and Counts:")
for i in range(len(counts)):
    print(f"{bins[i]}-{bins[i+1]}: {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***

```
import pandas as pd

n = int(input())

data = []
for _ in range(n):
    parts = input().split()
    name = parts[0]
    years = float(parts[1])
    data.append({'Name': name, 'Years at Company': years})

df = pd.DataFrame(data)

# Define a function for experience classification
def classify_experience(years):
    if years < 3:
        return 'Junior'
    elif 3 <= years < 6:
        return 'Mid'
    else:
        return 'Senior'

# Apply classification
df['Experience Level'] = df['Years at Company'].apply(classify_experience)

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

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

5. 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***

```
import numpy as np

rows, cols = map(int, input().split())

data = []
for _ in range(rows):
    data.append(list(map(float, input().split())))

returns = np.array(data)
```

```
positive_days = np.all(returns > 0, axis=1)

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

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