Rajalakshmi Engineering College

Name: Saii Anish R
Email: 241501175@rajalakshmi.edu.in
Roll no: 241501175
Phone: 8438920387
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_PAH

Attempt : 1
Total Mark : 50
Marks Obtained : 50

Section 1 : Coding

1. 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

df_survey = pd.DataFrame(
    [[p[0], p[1] if len(p) > 1 else ''] for p in [input().split(' ', 1) for _ in
range(int(input()))]],
    columns=['RespondentID', 'Feedback']
).assign(
    Feedback=lambda x: x['Feedback'].replace('', 'No Response')
)

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

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

2. 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

def determine_level(years_at_company):
    if years_at_company < 3:
```

```
    return "Junior"
  elif years_at_company < 6:
    return "Mid"
  else:
    return "Senior"

num_employees = int(input())

employee_data_list = []
for _ in range(num_employees):
  name, years_str = input().split()
  employee_data_list.append({'Name': name, 'Years at Company':
float(years_str)})

employee_df = pandas.DataFrame(employee_data_list)

employee_df['Experience Level'] = employee_df['Years at
Company'].apply(determine_level)

print("Employee Data with Experience Level:")
print(employee_df.to_string(index=False))
```

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

3. 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

input_line_str = input()
parts_list = input_line_str.split()
num_rows = int(parts_list[0])
num_cols = int(parts_list[1])

data_list_of_lists = []
i = 0
while i < num_rows:
    row_input_str = input()
    row_str_elements = row_input_str.split()

    current_row_float_values = []
    j = 0
    while j < num_cols:
        element_as_float = float(row_str_elements[j])
        current_row_float_values.append(element_as_float)
        j += 1

    data_list_of_lists.append(current_row_float_values)
```

```
    i += 1

daily_returns_array = numpy.array(data_list_of_lists)

all_positive_days_flags = numpy.all(daily_returns_array > 0, axis=1)

header_text = "Days where all asset returns were positive:"
print(header_text)
print(all_positive_days_flags)
```

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

4. 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

n_customers_input = int(input())
wait_times_str_list_input = input().split()
```

```
wait_times_data_array = numpy.array(wait_times_str_list_input, dtype=float)

fixed_bucket_size = 5

max_recorded_wait_time = numpy.max(wait_times_data_array)

bins_overall_upper_edge = numpy.ceil(max_recorded_wait_time /
fixed_bucket_size) * fixed_bucket_size
if bins_overall_upper_edge == 0.0:
    bins_overall_upper_edge = float(fixed_bucket_size)

num_buckets_for_output = int(bins_overall_upper_edge / fixed_bucket_size)

digitize_bin_definitions = numpy.linspace(0.0, bins_overall_upper_edge,
num=num_buckets_for_output + 1)

assigned_indices_per_wait_time = numpy.digitize(wait_times_data_array,
digitize_bin_definitions, right=False)

counts_per_assigned_index = numpy.bincount(assigned_indices_per_wait_time)

print("Wait Time Buckets and Counts:")

for current_bucket_loop_idx in range(num_buckets_for_output):
    current_bucket_label_lower = current_bucket_loop_idx * fixed_bucket_size
    current_bucket_label_upper = (current_bucket_loop_idx + 1) *
fixed_bucket_size
    lookup_idx_for_bincount = current_bucket_loop_idx + 1

    count_for_current_bucket_label = 0

    if lookup_idx_for_bincount < len(counts_per_assigned_index):
        count_for_current_bucket_label =
counts_per_assigned_index[lookup_idx_for_bincount]

    print(f"{current_bucket_label_lower}-{current_bucket_label_upper}:
{count_for_current_bucket_label}")
```

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

5. 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.]]

2415011175

Answer

```
import numpy

rows_input_value = int(input())
cols_input_value = int(input())

image_data_as_list_of_lists = []
row_counter_var = 0
while row_counter_var < rows_input_value:
    input_row_str_values = input().split()
    current_row_integer_pixels = []
    col_counter_var = 0
    while col_counter_var < cols_input_value:
        pixel_val_int = int(input_row_str_values[col_counter_var])
        current_row_integer_pixels.append(pixel_val_int)
        col_counter_var += 1
    image_data_as_list_of_lists.append(current_row_integer_pixels)
    row_counter_var += 1

image_matrix_main_array = numpy.array(image_data_as_list_of_lists,
dtype=float)

min_pixel_value_overall = numpy.min(image_matrix_main_array)
max_pixel_value_overall = numpy.max(image_matrix_main_array)

if min_pixel_value_overall == max_pixel_value_overall:
    image_matrix_main_array = numpy.zeros_like(image_matrix_main_array,
dtype=float)
else:
    image_matrix_main_array = (image_matrix_main_array -
min_pixel_value_overall) / (max_pixel_value_overall - min_pixel_value_overall)

print(image_matrix_main_array)
```

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

2415011175