CSV, Lamda function, Map, Reduce

Sidheswar Routray
Department of Computer Science & Engineering
School of Technology

Example

NAME	MATHS	SCIENCE	ENGLISH
John	85	78	92
Alice	89	90	93
Bob	76	90	85
Eve	88	78	88

- **You are given a CSV file named "student_grades.csv," which contains the following data:**

Your task is to write a Python program that reads this CSV file, calculates the average score for each student, and then creates a new CSV file named "student_average_grades.csv"

Steps to Solve

1. Read the data from "student_grades.csv" using CSV file handling in Python.

2. For each student, calculate their average score across all subjects (Maths, Science, and English).

3. Create average functions to calculate the average for each student.

4. Store the student's name and their corresponding average score in a new dictionary.

5. Write the data from the dictionary into a new CSV file named "student_average_grades.csv" with two columns: "Name" and "Average."

```python
import csv

input_file = "student_grades.csv"
output_file = "student_average_grades.csv"
student_grades = {}

# Read data from CSV file
with open(input_file, "r") as csvfile:
    reader = csv.DictReader(csvfile)
    for row in reader:
        name = row["Name"]
        math_score = int(row["Maths"])
        science_score = int(row["Science"])
        english_score = int(row["English"])
        score=[math_score, science_score, english_score]
        average_score = sum(score) / len(score)
        student_grades[name] = average_score
```

Write data to new CSV file

```python
    with open(output_file, "w", newline="") as csvfile:
        fieldnames = ["Name", "Average"]
        writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
        writer.writeheader()
        for name, average in student_grades.items():
            writer.writerow({"Name": name, "Average": average})

    print("Successfully calculated and saved average grades in
'student_average_grades.csv'.")
```

Note:
writer.writeheader() # Writes the header row (field names) to the CSV file
writer.writerows(data) # Writes the data rows to the CSV file

Use pandas to solve this problem

```python
import pandas as pd
# Function to calculate average using lambda function

calculate_average = lambda scores: sum(scores) / len(scores)
def main():

    input_file = "student_grades.csv"
    # Read data from CSV file using pandas

    df = pd.read_csv(input_file)
    # Calculate average for each student and add "Average" column

    df["Average"] = df.apply(lambda row: calculate_average([row["Maths"], row["Science"], row["English"]]), axis=1)
```

```python
# Write data back to the input CSV file with "Average" column
df.to_csv(input_file, index=False)

print("Successfully calculated and appended average grades as a new column in 'student_grades.csv'.")

if __name__ == "__main__":
    main()
```

Lambda Function

- Lambda functions in Python are small anonymous functions defined with the lambda keyword. They can have any number of arguments but only one expression. The expression is evaluated and returned.

- **lambda arguments: expression**

Basic Usage: A lambda function to add two numbers.

```python
add = lambda x, y: x + y
print(add(2, 3))
# Output: 5
```

```python
x = lambda a, b : a * b
print(x(5, 6))
```

```python
def cube(y):
    return y*y*y;
g = lambda x: x*x*x
print(g(7))
print(cube(5))
```

Lambda Function

- **The map() function** in Python applies a given function to each item of an iterable (like a list) and returns an iterator of the results.

- When combined with a lambda function, map() allows for concise and functional-style transformations.

Basic Syntax

map(function, iterable)

- function: The function to apply to each item.
- iterable: The iterable whose items are to be processed.

Example 1: Squaring Numbers
numbers = [1, 2, 3, 4]
squared = map(lambda x: x ** 2, numbers)
print(list(squared))
# Output: [1, 4, 9, 16]

In this example:
•lambda x: x ** 2 is a lambda function that squares its input.
•map() applies this lambda function to each element in the numbers list.
•The result is an iterator, which we convert to a list using list() to display the output.

```python
a = [1,2,3,4]
b = [17,12,11,10]
c = [-1,-4,5,9]
S1 = list(map(lambda x,y:x+y, a,b))
S2 = list(map(lambda x,y,z:x+y+z, a,b,c))
S3 = list(map(lambda x,y,z:x+y-z, a,b,c))
print(S1)
print(S2)
print(S3)
```

Problem

I have a list which contains temperature in Celsius. Task is to convert Celcius to faherenheit.

i/p- [39.2, 45.5,67.9,38.3]

o/p- corresponding faherenheit

map

```python
Celsius = [39.2, 36.5, 37.3, 37.8]
Fahrenheit = list(map(lambda x: (float(9)/5)*x + 32, Celsius))
print (Fahrenheit)
C = list(map(lambda x: (float(5)/9)*(x-32), Fahrenheit))
print (C)
```

Lambda Function

- **The filter() function** in Python is used to filter elements of an iterable based on a function that returns either True or False. When combined with a lambda function, filter() allows you to easily apply custom filtering criteria.

Basic Syntax

filter(function, iterable)

- function: A function that returns True or False for each item in the iterable.
- iterable: The iterable whose items are to be processed.

Example 1: Filtering Even Numbers
```
numbers = [1, 2, 3, 4, 5, 6]
even_numbers = filter(lambda x: x % 2 == 0, numbers)
print(list(even_numbers))
 # Output: [2, 4, 6]
```

In this example:
- lambda x: x % 2 == 0 is a lambda function that checks if a number is even.
- filter() applies this lambda function to each element in the numbers list.
- Only the elements for which the lambda function returns True (even numbers) are included in the result.

Example 2: Filtering Strings by Length

words = ['apple', 'banana', 'cherry', 'date']
long_words = filter(lambda word: len(word) > 5, words)
print(list(long_words))
# Output: ['banana', 'cherry']

In this example:
- lambda word: len(word) > 5 is a lambda function that checks if the length of a string is greater than 5.
- filter() applies this lambda function to each element in the words list.
- Only the words with more than 5 characters are included in the result.

```python
li = [5, 7, 22, 97, 54, 62, 77, 23, 73]
result = list(filter(lambda x: (x>=25) , li))
print(result)
```

- **The reduce() function** in Python is used to apply a binary function (a function that takes two arguments) cumulatively to the items of an iterable, from left to right, so as to reduce the iterable to a single result.

- It's part of the functools module.

Basic Syntax
from functools import reduce
reduce(function, iterable[, initializer])
- **function:** A binary function that takes two arguments and returns a single result.
- **iterable:** The iterable whose items are to be reduced.
- **initializer** (optional): A value that is used as the initial value for the reduction. If not provided, the first item of the iterable is used as the initial value, and the reduction starts from the second item.

Example 1: Calculating the Product of a List
```
from functools import reduce
numbers = [1, 2, 3, 4]
product = reduce(lambda x, y: x * y, numbers)
print(product)
# Output: 24
```
In this example:
•lambda x, y: x * y is a lambda function that multiplies two numbers.
•reduce() applies this function cumulatively to the items in the numbers list, resulting in the product of all elements.

Example 2: Finding the Maximum Value
from functools import reduce
numbers = [5, 1, 8, 3, 7]
max_value = reduce(lambda x, y: x if x > y else y, numbers)
print(max_value)
 # Output: 8

In this example:
•lambda x, y: x if x > y else y is a lambda function that returns the maximum of two numbers.
•reduce() applies this function cumulatively to find the maximum value in the numbers list.

Problem-

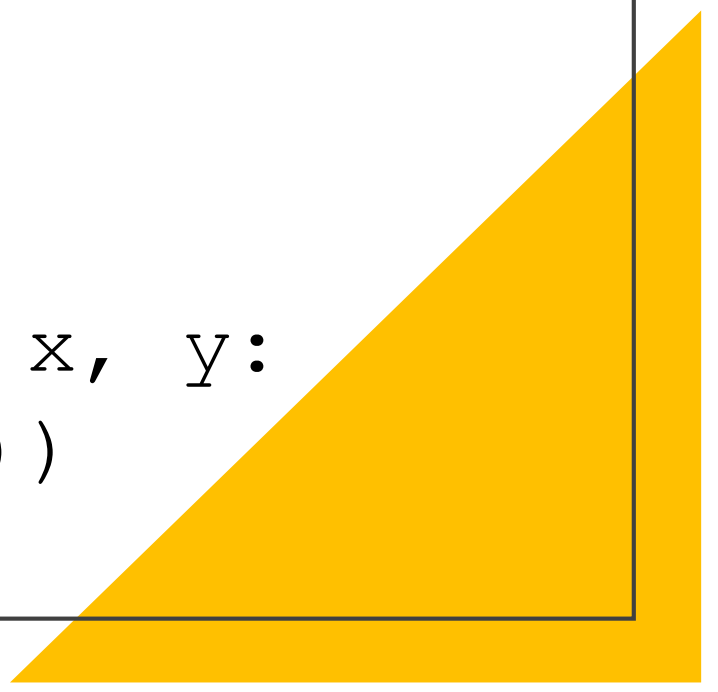In a company I am doing some task where I have to find sum of squares of the given list.

I/p- [2,3,6,5]

o/p- 2*2 +3*3 + 6*6 + 5*5 = 74

Reduce

```python
from functools import reduce
li=[1,2,3,4,5,6,7]
sum= reduce((lambda x,y:x*x+y*y),li)
print(sum)

from functools import reduce
li = [1, 2, 3, 4, 5, 6, 7]
sum_of_squares = reduce(lambda x, y: x + y, map(lambda x: x * x, li))
print(sum_of_squares)
```

Reduce

```
from functools import reduce
li=[1,2,3,4,5,6,7]
sum= reduce((lambda x,y:x*x+y*y),li)
print(sum)
```

3560020598205630145296938

```
from functools import reduce

li = [1, 2, 3, 4, 5, 6, 7]
sum_of_squares = reduce(lambda x, y: x + y, map(lambda x: x * x, li))
print(sum_of_squares)
```

140

```python
from functools import reduce
f=lambda a,b: a if(a>b) else b
r = reduce(f, [47,111,42,102,13])
print(r)
```

Problem Statement

You are working as a data engineer for a large retail company. Your team is responsible for processing and analyzing sales data from multiple stores across the country. The sales data is stored in CSV files, and each file represents sales data for a specific month and year. Each CSV file has the following columns:

- Date (in the format "YYYY-MM-DD")

- Store ID (a unique alphanumeric code)

- Product ID (a unique alphanumeric code)

- Quantity sold (an integer representing the number of products sold on that date)

The "product_names.csv" file has two columns: "Product ID" and "Product Name," and it contains the mapping for all products in the sales data.

Your task is to write a Python program that performs the following operations:

- Read the sales data from all the CSV files in a given directory and its subdirectories.
- Calculate the total sales (quantity sold) for each product across all stores and all months.
- Determine the top 5 best-selling products in terms of the total quantity sold.

Create a new CSV file named "sales_summary.csv" and write the following information into it:

- Product ID
- Product Name
- Total Quantity Sold
- Average Quantity Sold per month (considering all months available in the data)