CPE4040_Homework_221

September 25, 2024

CPE 4040: Homework 2

1. Write your answer in the cell provided under each question.
2. **Important: Write comments** in the code to explain your thoughts. **You may lose points without comments.**
3. Show your execution result.
4. Do your own work. **Do not copy-and-paste other people's (or Generative AI's) codes.**

0.0.1 Submission:

- **Submit this notebook file and the pdf version** - remember to add your name in the filename. #### You will write a small function, `"second"`, to achieve this purpose.

0.1 Q1: Functions and Lambda (15 Points)

Q1.1: Sort the tuples in the list based on the second item (that is, the state name), in descending order alphabetically.

You will write a small function, "second", to achieve this purpose.

[1300]:
```
state=[(19542209, "New York") ,(4887871, "Alabama"), (1420491, "Wyoming"),
       (626299, "Vermont"), (1805832, "Utah"), (39865590, "California")]
```

[1301]:
```
def second(states):
    n = len(states)

    # Simple bubble sort algorithm
    for i in range(n - 1):
        for j in range(n - i - 1):
            if states[j][1] > states[j + 1][1]:
                states[j], states[j + 1] = states[j + 1], states[j]
    return states

print(second(state))
```

```
[(4887871, 'Alabama'), (39865590, 'California'), (19542209, 'New York'),
(1805832, 'Utah'), (626299, 'Vermont'), (1420491, 'Wyoming')]
```

Q1.2 Perform the same task as in Q1.1, however, you will write a `lambda` function to replace `"second"`.

[1302]:
```
print(sorted(state, key=lambda x: x[1]))
```

```
[(4887871, 'Alabama'), (39865590, 'California'), (19542209, 'New York'),
(1805832, 'Utah'), (626299, 'Vermont'), (1420491, 'Wyoming')]
```

Q1.3 Use the sorted method and lambda to sort the tuples in the list based on the next to last character of the second items.

[ ]:
```
```

[1303]:
```
def second(states):
    n = len(states)

    # Modified Bubble Sort to sort based off second to last letter
    for i in range(n - 1):
        for j in range(n - i - 1):
            state1 = states[j][1][len(states[j][1]) - 2]        # 3 dimensional␣
 ↪index to grab the second to last letter
            state2 = states[j + 1][1][len(state[j + 1][1]) - 2] # 3 dimensional␣
 ↪index to grab the second to last letter

            if state1 > state2: # Compares the two
                states[j], states[j + 1] = states[j + 1], states[j]
    return states

print(second(state))

print(sorted(state, key=lambda x: x[1][len(x[1]) - 2]))
```

```
[(1805832, 'Utah'), (39865590, 'California'), (4887871, 'Alabama'), (626299,
'Vermont'), (1420491, 'Wyoming'), (19542209, 'New York')]
[(1805832, 'Utah'), (39865590, 'California'), (4887871, 'Alabama'), (626299,
'Vermont'), (1420491, 'Wyoming'), (19542209, 'New York')]
```

0.2 Q2: List Comprehension vs. Map/Filter + Lambda (15 Points)

For each question below, use both list comprehension and Python `map()` or `filter( )` function.

Use `%%timeit` magic command to profile the computation time. Make some comments about the profiling results.

Q2.1 For each of the floating number in the list, calcuate the square of it and round the result to 2 decimal places.

[1304]:
```
num = [4.35, 6.09, 3.25, 9.77, 2.16, 8.88, 4.59]
```

[1305]:
```
## List comprehension
numComp = [round(x**2, 2) for x in num]
print(numComp)
```

```
[18.92, 37.09, 10.56, 95.45, 4.67, 78.85, 21.07]
```

[1306]:
```
# map() function
def square(n):
    return round(n**2, 2)

numMap = list(map(square, num))
print(numMap)
```

```
[18.92, 37.09, 10.56, 95.45, 4.67, 78.85, 21.07]
```

Q2.2 For each of the name in the list, print only the names with less than 7 letters.

[1307]:
```
names = ["olumide", "akinremi", "alex", "josiah", "john", "temidayo", "emily",␣
 ↪"omoseun"]
```

[1308]:
```
# Use List Comprehension
namesComp = [name for name in names if len(name) < 7]
print(namesComp)
```

```
['alex', 'josiah', 'john', 'emily']
```

[1309]:
```
# Use filter() function
def nameFilter(name):
    if len(name) < 7:
        return name
nameFil = list(filter(nameFilter, namesComp))
print(nameFil)
```

```
['alex', 'josiah', 'john', 'emily']
```

0.3 Q3: Dictionary Operations (20 Points)

Q3.1 Shopping Lists You are given two shopping lists (with items and quantity), one from your mom and one for yourself: * mom's list: banana 5, avacado 8, eggs 12, milk 2, orange 6 * your list: Red Bull 12, potato chips 5, pineapple 3, eggs 6, avacado 4

Please construct two dictionaries, one for each shopping list.

[1310]:
```
momList = {"banana" : 5, "avocado" : 8, "eggs" : 12, "milk" : 2, "orange" : 6}
myList = {"Red Bull" : 12, "potato chips" : 5, "pineapple" : 3, "eggs" : 6,␣
 ↪"avocado" : 4}
```

Q3.2 Merging the Shopping Lists You realize there are some overlapping items on the lists, so you decide to merge the two.

Please write a `__function, merge_shop_list(a, b)__`, that takes two dictionaries **a** and **b** as inputs, and merge them together. If there is a common item, you should add the quantities together. The function shall return the merged dictionary (or the shopping list).

Show your result by printing the merged shopping list.

[1311]:
```
def merge_shop_list(a, b):
    c = {i: a.get(i, 0) + b.get(i, 0) for i in set(a).union(b)}
    return c

newList = merge_shop_list(momList, myList)
print(newList)
```

```
{'eggs': 18, 'milk': 2, 'orange': 6, 'banana': 5, 'potato chips': 5,
'pineapple': 3, 'Red Bull': 12, 'avocado': 12}
```

0.4 Q4: AppleStore Dataset (50 points)

Q4.1 Genres and Content Ratings With the AppleStore dataset, we found that "Game" is the most popular genre with more than 3000+ apps. We also realized that there are four different age-based content ratings (4+, 9+, 12+, and 17+).

Write a Python program that analyzes the content rating distribution for the top three genres (Game, Entertainment and Education). That is, we want to know, for example, in the "Game" genere, how many apps are 4+, 9+, 12+, and 17+, respectively.

There are different ways to do this. Try your best to come up with a solution.

[1312]:
```
opened_file = open('AppleStore1.csv', encoding="utf-8")
from csv import reader

read_file = reader(opened_file)
apps_data = list(read_file)
```

[1313]:
```
import pandas as pd

dataframe = pd.read_csv("AppleStore1.csv")

## Game Data
game_data = dataframe.loc[dataframe["prime_genre"] == "Games"]
game_rating_count = game_data["cont_rating"].value_counts()
print("Games by rating: \n", game_rating_count)

## Entertainment Data
ent_data = dataframe.loc[dataframe["prime_genre"] == "Entertainment"]
ent_rating_count = ent_data["cont_rating"].value_counts()
print("\nEntertainment Apps by rating: \n", ent_rating_count)

## Education data
edu_data = dataframe.loc[dataframe["prime_genre"] == "Education"]
edu_rating_count = edu_data["cont_rating"].value_counts()
print("\nEducation Apps by rating: \n", edu_rating_count)
```

```
Games by rating:
 cont_rating
4+      2079
```

```
9+      865
12+     741
17+     177
Name: count, dtype: int64

Entertainment Apps by rating:
 cont_rating
4+     285
12+    108
17+     98
9+      44
Name: count, dtype: int64

Education Apps by rating:
 cont_rating
4+     432
12+      8
17+      7
9+       6
Name: count, dtype: int64
```

Q4.2 We are also interested in the price of the apps. Specifically, we want to know the number of applications in five different price ranges, that is, 1. free 2. $0 <$ price < 5 3. $5 <=$ price < 10 4. $10 <=$ price < 50 5. price $>=50$.

Please write a code that shows the price range distribution.

[1314]:
```
# Initialize the price range dictionary. You will write the rest.

price_range = {'Free': 0, '0 to $5': 0, '$5 to $10': 0, '$10 to $50': 0, '>=␣
 ↪$50': 0}
```

[1315]:
```
for price in price_range:
    if price == 'Free':
        price_range[price] = len(dataframe[(dataframe["price"] == 0.0)])
    elif price == '0 to $5':
        price_range[price] = len(dataframe[(dataframe["price"] > 0.0) &␣
 ↪(dataframe["price"] <= 5.0)])
    elif price == '$5 to $10':
        price_range[price] = len(dataframe[(dataframe["price"] > 5.0) &␣
 ↪(dataframe["price"] <= 10.0)])
    elif price == '$10 to $50':
        price_range[price] = len(dataframe[(dataframe["price"] > 10.0) &␣
 ↪(dataframe["price"] <= 50.0)])
    elif price == '>= $50':
        price_range[price] = len(dataframe[(dataframe["price"] > 50.0)])

print(price_range)
```

```
{'Free': 4056, '0 to $5': 2703, '$5 to $10': 341, '$10 to $50': 90, '>= $50': 7}
```

Q4.3 How many of the applications are free? That is, find out how many applications are priced at $0.0.

[1316]:
```
print("Number of Free apps: ", price_range['Free'])
```

```
Number of Free apps:  4056
```

[1317]:
```
max_price = dataframe[["track_name", "price"]]
max_price = max_price.sort_values(by="price", ascending=False)
print(max_price.head(3))
```

```
                         track_name   price
5184             LAMP Words For Life  299.99
2828  Proloquo2Go - Symbol-based AAC  249.99
4769                     KNFB Reader   99.99
```