Tugas Individu 7 Praktikum Pemrograman Berbasis Fungsi

March 31, 2023

Penjual jam tangan mewah telah meminta bantuan Anda untuk menyelesaikan permasalahan berikut, Menggabungkan lambda, map, filter, atau reduce, tulis program python ke:

- Jam tangan termahal
- Inventaris baru tempat jam tangan disortir menurut harganya dictionary yang di dalamnya setiap kunci mewakili merek jam tangan dan nilainya adalah biaya total merek tersebut dalam inventaris (kuantitas * harga)
- List yang berisi semua jam tangan dengan jumlah lebih dari 30
- Inventaris baru dengan harga diskon untuk item yang tidak terjual habis. Dalam inventaris baru ini, jika jumlah jam tangan dari inventaris lama lebih besar dari 30, kurangi harga sebesar 50 EUR jika harga lama lebih besar dari 500 EUR atau kurangi sebesar 30 EUR

[ ]:
```
import openpyxl
from functools import reduce

wb = openpyxl.load_workbook('hm.xlsx')
sheet = wb.active
rows = sheet.max_row
cols = sheet.max_column
watch = []
for i in range(4, rows + 1):
    for j in range(2, cols + 1, 2):
        cell = sheet.cell(row=i, column=j)
        if cell.value is None:
            continue
        watch.append(str(cell.value).strip())
id = []
item = []
quantity = []
unit_price = []
for i in range(0, len(watch), 4):
    id.append(int(watch[i]))
    item.append(watch[i + 1].strip())
    quantity.append(int(watch[i + 2]))
    unit_price.append(float(watch[i + 3].strip()))
inventory = []
for i, v in enumerate(id):
```

```
    inventory.append((id[i], item[i], quantity[i], unit_price[i]))
print(inventory)
```

```
[(1, 'Audemars Piguet', 65, 450.5), (2, 'Vacheron Constantin', 1, 790.0), (3,
'Patek Philippe', 18, 664.99), (4, 'Blancpain', 45, 356.0), (5, 'Chopard', 23,
1264.3), (6, 'IWC Schaffhausen', 11, 99.99), (7, 'Rolex', 0, 520.0), (8, 'Ulysse
Nardin', 34, 340.0), (9, 'Jaeger-LeCoultre', 2, 956.0), (10, 'Panerai', 8,
120.0)]
```

Menghitung harga termahal

[ ]:
```
print('---The most expensive watch---')
expensive_watch = reduce(max, unit_price)
print(f'The most expensive watch = {expensive_watch}')
```

```
---The most expensive watch---
The most expensive watch = 1264.3
```

Sortir Harga dan kemudian menyimpan nya dalam inventory

[ ]:
```
print('---Sorted by price---')
s = sorted(unit_price, key=lambda x: x)
print(s)
print('--or--')
s = sorted(inventory, key=lambda x: x[3])
print(s)
```

```
---Sorted by price---
[99.99, 120.0, 340.0, 356.0, 450.5, 520.0, 664.99, 790.0, 956.0, 1264.3]
--or--
[(6, 'IWC Schaffhausen', 11, 99.99), (10, 'Panerai', 8, 120.0), (8, 'Ulysse
Nardin', 34, 340.0), (4, 'Blancpain', 45, 356.0), (1, 'Audemars Piguet', 65,
450.5), (7, 'Rolex', 0, 520.0), (3, 'Patek Philippe', 18, 664.99), (2, 'Vacheron
Constantin', 1, 790.0), (9, 'Jaeger-LeCoultre', 2, 956.0), (5, 'Chopard', 23,
1264.3)]
```

Memetakan harga dengan jam

[ ]:
```
print('---Dictionary---')
d = dict(map(lambda x: [x[1], round((x[2]) * (x[3]), 2)], inventory))
print(d)
print('--or--')
def make_dict(inventory):
    d2 = {}
    for i in inventory:
        watch_brand = i[1]
        total_cost = round((i[2] * i[3]), 2)
        d2.update({watch_brand: total_cost})
    return d2
print(make_dict(inventory))
```

```
---Dictionary---
{'Audemars Piguet': 29282.5, 'Vacheron Constantin': 790.0, 'Patek Philippe':
11969.82, 'Blancpain': 16020.0, 'Chopard': 29078.9, 'IWC Schaffhausen': 1099.89,
'Rolex': 0.0, 'Ulysse Nardin': 11560.0, 'Jaeger-LeCoultre': 1912.0, 'Panerai':
960.0}
--or--
{'Audemars Piguet': 29282.5, 'Vacheron Constantin': 790.0, 'Patek Philippe':
11969.82, 'Blancpain': 16020.0, 'Chopard': 29078.9, 'IWC Schaffhausen': 1099.89,
'Rolex': 0.0, 'Ulysse Nardin': 11560.0, 'Jaeger-LeCoultre': 1912.0, 'Panerai':
960.0}
```

[ ]:
```python
print("---New inventory---")
def new_inventory(inventory):
    not_sold_out = []
    new_inv = []
    for i in inventory:
        if i[2] == 0:
            continue
        not_sold_out.append((i[0], i[1], i[2], i[3]))
    for i, v in enumerate(not_sold_out):
        not_sold_out[i] = list(not_sold_out[i])
        if not_sold_out[i][2] > 30:  # if quantity of watches > 30
           if not_sold_out[i][3] > 500:  # if price > 500
               not_sold_out[i][3] -= 50
           else:                  # if price <= 500
               not_sold_out[i][3] -= 30
        not_sold_out[i] = tuple(not_sold_out[i])
    inventory = not_sold_out
    for i, v in enumerate(inventory):
        print('(',inventory[i][1], '-', inventory[i][2],'items)', ' -> ',␣
 ↪inventory[i][3],'eur')
        new_inv.append((inventory[i][0], inventory[i][1], inventory[i][2],␣
 ↪inventory[i][3]))
    return new_inv

print(new_inventory(inventory))
```

```
---New inventory---
( Audemars Piguet - 65 items)  ->  420.5 eur
( Vacheron Constantin - 1 items)  ->  790.0 eur
( Patek Philippe - 18 items)  ->  664.99 eur
( Blancpain - 45 items)  ->  326.0 eur
( Chopard - 23 items)  ->  1264.3 eur
( IWC Schaffhausen - 11 items)  ->  99.99 eur
( Ulysse Nardin - 34 items)  ->  310.0 eur
( Jaeger-LeCoultre - 2 items)  ->  956.0 eur
( Panerai - 8 items)  ->  120.0 eur
[(1, 'Audemars Piguet', 65, 420.5), (2, 'Vacheron Constantin', 1, 790.0), (3,
```

```
'Patek Philippe', 18, 664.99), (4, 'Blancpain', 45, 326.0), (5, 'Chopard', 23,
1264.3), (6, 'IWC Schaffhausen', 11, 99.99), (8, 'Ulysse Nardin', 34, 310.0),
(9, 'Jaeger-LeCoultre', 2, 956.0), (10, 'Panerai', 8, 120.0)]
```

Cara Lain:

[ ]:
```
id =[]
for cell in sheet["B"]:
     if cell.value is not None:
         id.append(cell.value)
print(id)
item =[]
for cell in sheet["D"]:
     if cell.value is not None:
         cell.value = cell.value.strip()
         item.append(cell.value)
quantity =[]
for cell in sheet["F"]:
     if cell.value is not None:
         quantity.append(cell.value)
unit_price =[]
for cell in sheet["H"]:
     if cell.value is not None:
         unit_price.append(cell.value)
inventory = []
for i, v in enumerate(id):
     if i == 0 or i == 1:
         continue
     inventory.append((id[i], item[i], quantity[i], float(unit_price[i])))
```

```
['id          ', '------------', 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

[ ]:
```
# The most expensive watch
print('---The most expensive watch---')
res = reduce(max, map(lambda x: x[3], inventory))
print(f"The most expensive watch -> {res}")
price = []

for i in inventory:
     price.append(i[3])
res2 = reduce(max, price)
print(f"The most expensive watch -> {res2}")
```

```
---The most expensive watch---
The most expensive watch -> 1264.3
The most expensive watch -> 1264.3
```

[ ]:
```
print('---Sorted by price---')
 # A new inventory where the watches are sorted according to their price
```

```
s = sorted(inventory, key=lambda x: x[3])
print('sorted-price ->', s)
for i, v in enumerate(s):
    print(f'{s[i][1]} -> {s[i][3]}eur')
```

```
---Sorted by price---
sorted-price -> [(6, 'IWC Schaffhausen', 11, 99.99), (10, 'Panerai', 8, 120.0),
(8, 'Ulysse Nardin', 34, 340.0), (4, 'Blancpain', 45, 356.0), (1, 'Audemars
Piguet', 65, 450.5), (7, 'Rolex', 0, 520.0), (3, 'Patek Philippe', 18, 664.99),
(2, 'Vacheron Constantin', 1, 790.0), (9, 'Jaeger-LeCoultre', 2, 956.0), (5,
'Chopard', 23, 1264.3)]
IWC Schaffhausen -> 99.99eur
Panerai -> 120.0eur
Ulysse Nardin -> 340.0eur
Blancpain -> 356.0eur
Audemars Piguet -> 450.5eur
Rolex -> 520.0eur
Patek Philippe -> 664.99eur
Vacheron Constantin -> 790.0eur
Jaeger-LeCoultre -> 956.0eur
Chopard -> 1264.3eur
```

[ ]:
```
print('---Dictionary---')
# A dictionary within which each key represents a watch brand and the value is␣
 ↪the total cost of that
# brand in our inventory (quantity * price)
d = dict(map(lambda x: [x[1], round(x[2]*x[3], 2)], inventory))
print('dict ->', d)
```

```
---Dictionary---
dict -> {'Audemars Piguet': 29282.5, 'Vacheron Constantin': 790.0, 'Patek
Philippe': 11969.82, 'Blancpain': 16020.0, 'Chopard': 29078.9, 'IWC
Schaffhausen': 1099.89, 'Rolex': 0.0, 'Ulysse Nardin': 11560.0, 'Jaeger-
LeCoultre': 1912.0, 'Panerai': 960.0}
```

[ ]:
```
print('---All watches with quantity greater than 30---')
 # A list containing all watches with quantity greater than 30
gt = list(filter(lambda x: x[2] > 30, inventory))
print('quantity > 30 ->', gt)
for i, v in enumerate(gt):
    print(f'{gt[i][1]} - {gt[i][2]}')
```

```
---All watches with quantity greater than 30---
quantity > 30 -> [(1, 'Audemars Piguet', 65, 450.5), (4, 'Blancpain', 45,
356.0), (8, 'Ulysse Nardin', 34, 340.0)]
Audemars Piguet - 65
Blancpain - 45
Ulysse Nardin - 34
```

```
[ ]: print("---New inventory---")
not_sold_out = list(filter(lambda x: x[2] > 0, inventory))
for i, v in enumerate(not_sold_out):
     not_sold_out[i] = list(not_sold_out[i])
     if not_sold_out[i][2] > 30:  # if quantity of watches > 30
         if not_sold_out[i][3] > 500:  # if price > 500
             not_sold_out[i][3] -= 50
         else:                  # if price <= 500
             not_sold_out[i][3] -= 30
     not_sold_out[i] = tuple(not_sold_out[i])
inventory = not_sold_out
for i, v in enumerate(inventory):
     print('(',inventory[i][1], '-', inventory[i][2],'items)', ' -> ',␣
  ↪inventory[i][3],'eur')
```

```
---New inventory---
( Audemars Piguet - 65 items)  ->  420.5 eur
( Vacheron Constantin - 1 items)  ->  790.0 eur
( Patek Philippe - 18 items)  ->  664.99 eur
( Blancpain - 45 items)  ->  326.0 eur
( Chopard - 23 items)  ->  1264.3 eur
( IWC Schaffhausen - 11 items)  ->  99.99 eur
( Ulysse Nardin - 34 items)  ->  310.0 eur
( Jaeger-LeCoultre - 2 items)  ->  956.0 eur
( Panerai - 8 items)  ->  120.0 eur
```

1 Tugas Kelompok

Download data disini :

https://drive.google.com/file/d/1yRYkjsF9CuC4RXueZQCHuJnXRSXliBjS/view?usp=sharing

Langkah dalam menyelesaikan tugas ini adalah:

- Buatlah terlebih dahulu list dictionary yang kosong
- Buatlah looping untuk melakukan kalkulasi dengan fungsi Map atau Filter atau Reduce kemudian simpan ke dalam list dictionary yang sudah di definisikan

Bagian 1: Model Populasi Polisi Detroit

Baca data dari file Laporan Polisi Detroit menggunakan modul csv dan terjemahkan data ke dalam daftar dictionary. Menggunakan Filter dengan fungsi lambda untuk mengecualikan dictionary (baris CSV) yang memiliki data yang hilang di kolom Zip, atau kolom Neighborhood. Dengan menggunakan fungsi lambda dan Reduce, hitung total waktu respons rata-rata, waktu pengiriman rata-rata, dan total waktu rata-rata untuk kepolisian Detroit.

Bagian 2: Modelkan Neighborhood Samples

Menggunakan fungsi lambda dan map, atau lambda dan Filter, bagi list dictionary menjadi list dictionary yang lebih kecil yang dipisahkan oleh neighborhood. Dengan menggunakan lambda

dan Reduce, temukan total waktu respons rata-rata untuk setiap neighborhood, waktu pengiriman rata-rata untuk setiap neighborhood, dan total waktu rata-rata untuk setiap neighborhood dan simpanlah ke dalam list dictionary. Tambahkan item dictioanary untuk menyertakan data populasi untuk semua Detroit dalam daftar gabungan Anda.

Bagian 3: Buat file Output JSON

Menggunakan modul JSON, format list dictionary Anda sebagai JSON