IMPLEMENTASI FUNGSI REDUCE, FILTER, DAN MAP UNTUK MENGOLAH DATA POLISI DETROIT

Arafi Ramadhan Maulana (122450002)[1)], Dwi Ratna Anggraeni (122450008)[2)], Raid Muhammad Naufal (122450027)[3)], Rayan Koemi Karuby (122450038)[4)], Muhammad Deriansyah Okutra (122450101)[5)]

Program Studi Sains Data, Fakultas Sains, Institut Teknologi Sumatera

Email :
arafi.122450002@student.itera.ac.id [1] , *dwi.122450008@student.itera.ac.id* [2] ,
raid.122450027@student.itera.ac.id [3] , *rayan.122450038@student.itera.ac.id* [4] ,
mderiansyah.122450101@student.itera.ac.id [5]

1. Pendahuluan

Seiring berjalannya era digital yang terus berkembang, pengolahan data menjadi krusial dalam berbagai bidang. Dengan menggunakan python sebagai bahasa pemrograman, *Python* memudahkan pengguna dengan cara menyediakan banyak modul dan fungsi yang dapat memudahkan pengolahan data, seperti modul *CSV* dan *JSON* yang digunakan untuk mengelola data dengan format yang lebih dari satu.

JSON (JavaScript Object Notation) merupakan format file berbasis teks yang digunakan untuk melakukan pertukaran data antar server dan klien. File *JSON* memiliki ekstensi sendiri, yaitu .json dan menggunakan teks yang dapat dibaca oleh manusia dan dipahami oleh komputer. (Faradilla, 2022)

Dalam jurnal kali ini, kami menggunakan data dari laporan polisi detroit. Menggunakan fungsi *DictReader(),* pembacaan file *CSV* menjadi lebih mudah dengan cara membacanya menjadi *dictionary,* sehingga manipulasi data dan akses menjadi lebih mudah. *JSON* memungkinkan pengguna untuk melakukan konversi data dari python menjadi *JSON* agar pertukaran data menjadi ideal antar aplikasi.

Jurnal ini akan membahas penggunaan berbagai fungsi dan modul tersebut dalam pengolahan data, menekankan efisiensi dan kemudahan yang ditawarkan oleh *Python.*

2. Metode

2.1. CSV

Modul CSV digunakan untuk menangani file CSV, CSV adalah file dimana setiap bagian data berada di barisnya sendiri, dan setiap baris dipisahkan oleh koma atau titik koma. File ini membantu memindahkan informasi dari satu aplikasi ke aplikasi lainnya. CSV berguna sebagai jenis sumber data eksternal yang dapat digunakan untuk mengimpor dan mengekspor data dari spreadsheet yang lebih mudah untuk dilakukan manipulasi ataupun analisis.

2.1.1. *DictReader()*

DictReader() adalah fungsi yang terdapat dalam modul csv digunakan untuk membaca CSV dalam bentuk *dictionary.*

2.2. JSON

Modul JSON digunakan untuk menangani file JSON, JSON (*JavaScript Object Notation*) adalah format data yang digunakan untuk pertukaran dan penyimpanan data. JSON adalah turunan dari JavaScript. JSON dapat dibaca oleh berbagai jenis bahasa pemrograman misalnya C, C++, C#, *Java, Javascript Perl, Python*, dan lain-lain. Hal ini menjadikan JSON ideal untuk pertukaran data dari satu aplikasi ke aplikasi lainnya.

2.2.1. *dumps()*

Fungsi *dumps()* akan mengonversi sebagian objek *Python* menjadi string json. Namun tidak semua objek dapat dikonversi dan di sisi lain, mungkin perlu dilakukan pembuatan kamus dari data yang ingin ditampilkan sebelum membuat serialisasi ke JSON.

2.3. *Functools*

Modul *Functools* dimaksudkan untuk fungsi pada tingkat tinggi yang digunakan pada fungsi lainnya. Modul ini menyediakan fungsi-fungsi yang bekerja menggunakan fungsi dan objek lain yang bisa dipanggil untuk dipakai atau diperluas tanpa harus menulis ulang secara keseluruhan.

2.3.1. *reduce()*

Fungsi *reduce()* dalam *python* mengulang pada setiap item dalam sebuah daftar, atau jenis data yang dapat berubah dan mengembalikan nilai tunggal. Fungsi ini merupakan salah satu metode dari kelas *functools* dalam *Python.*

2.4. *open()*

Fungsi *open()* digunakan untuk mengakses sebuah file dan mengembalikannya dalam bentuk objek file. Dengan menggunakan objek file ini, kita bisa membuat, mengupdate, membaca, dan menghapus file.

2.5. *write()*

Fungsi *write()* adalah fungsi yang ada di dalam *python* yang memungkinkan kita untuk menulis data ke sebuah file. Fungsi ini menerima string sebagai input dan mulai memasukkan string yang disediakan ke dalam file yang ditentukan. fungsi *write()* mampu memasukkan berbagai tipe data ke dalam file, termasuk teks, nilai numerik, dan bahkan data biner.

2.6. *list()*

List adalah tipe data yang mengumpulkan beberapa item dalam satu variabel. Item dalam daftar bersifat berurutan, dapat diubah, dan diduplikasi.

2.6.1. *items()*

Dalam Python, setiap nilai yang disimpan dalam list disebut item atau elemen.

2.6.2. *append()*

Pada *list* yang sudah memiliki elemen di dalamnya, kita bisa menggunakan *.append()* untuk melakukan penambahan elemen baru di bagian akhir, atau elemen yang berada di bagian paling kanan.

2.7. *filter()*

Fungsi *filter()* mengekstrak elemen dari sebuah iterable (*list, tuple*, dll.) yang fungsi mengembalikan nilai *True*. Jika *True* maka elemen tetap berada di dalam iterable jika tidak maka elemen akan dihapus / tidak dipilih untuk iterasi selanjutnya.

2.8. *float()*

Float() adalah fungsi yang mengembalikan hasil konversi menjadi tipe data untuk objek berupa angka desimal, seperti 1.22, 2.33, 4.55 dan seterusnya. Perlu diperhatikan, untuk membuat angka desimal menggunakan simbol titik (.) pada *python*.

2.9. *len()*

Fungsi *len()* digunakan untuk menghitung dan mengetahui berapa panjang banyaknya item dalam suatu baris di *DataFrame*. Jika objek bertipe *string* maka fungsi ini akan mengembalikan jumlah karakter pada objek tersebut. Penerapan fungsi *len()* dapat dilakukan

pada beragam jenis data misalnya data *sequence* yang berupa string, list, tuple, dan range dan data *collection* berupa *dictionary, set dan frozenset*.

2.10. *set()*

Set() dalam python merupakan tipe data kolektif yang berguna untuk menyimpan beberapa nilai dengan ketentuan bahwa nilai bersifat unik (tidak ada duplikasi), nilai item tidak dapat diubah lagi, item tidak dapat diakses melalui indeks dan kemudian disimpan dalam sebuah variabel.

2.11. *map()*

Fungsi *map()* adalah fungsi yang sudah tersedia di dalam *Python* yang digunakan untuk menerapkan sebuah fungsi pada setiap item dalam sebuah iterable(misalnya *list, tupel*, dll.) tanpa menggunakan *for loop*.

2.12. menghitung_rata2_waktu()

Waktu rata-rata adalah nilai statistik yang menggambarkan nilai rata-rata atau nilai yang diharapkan dari sejumlah waktu yang diukur. Nilai ini diperoleh dengan menjumlahkan semua waktu yang diukur lalu membaginya dengan jumlah total waktu yang diukur. Dengan cara ini, waktu rata-rata memberikan gambaran tentang waktu yang mungkin diharapkan atau biasanya terjadi dalam sekelompok waktu yang diukur.

3. Pembahasan

Pada aplikasi ini, kami menggunakan dataset 911 Calls for Service, dataset ini menunjukkan semua panggilan tanggap darurat polisi 911 dan panggilan yang diprakarsai oleh petugas untuk layanan di Kota Detroit dalam 30 hari terakhir. Panggilan tanggap darurat adalah hasil dari orang-orang yang menelepon 911 untuk meminta layanan polisi. Panggilan yang diprakarsai oleh petugas termasuk penghentian lalu lintas, investigasi jalanan dan kegiatan kepolisian lainnya (seperti mengamati kejahatan yang sedang berlangsung) di mana petugas polisi memprakarsai respons. Dataset ini mencakup semua panggilan yang diterima, pengiriman, perjalanan, dan total waktu respons untuk panggilan yang dilayani oleh lembaga kepolisian. Data tersebut juga mencakup lembaga yang merespons, unit, jenis panggilan, dan kategori setiap panggilan.

```
[1]  import csv
     import json
     from functools import reduce
```

Gambar 1. *Import Library*

Pada Gambar 1. *library* yang digunakan aplikasi ini adalah *library csv* dan *json, csv* digunakan untuk membaca dan menulis file *csv*, *json* digunakan untuk bekerja dengan data dalam format *json*. Kedua *library* ini biasanya untuk memudahkan proses pembacaan, penulisan, dan import data. Selain kedua *library* tersebut digunakan juga fungsi *reduce* dari *library functools* yang berfungsi untuk mengulang pada setiap item dalam sebuah daftar, atau jenis data yang dapat berubah dan mengembalikan nilai tunggal.

```
[2]  with open('911_Calls_for_Service_(Last_30_Days).csv', 'r') as file:
         reader = csv.DictReader(file)
         data = list(reader)
```

Gambar 2. Membaca Dataset

Selanjutnya adalah membaca dataset, pada kasus kali ini kita menggunakan dataset 911_calls_for_service dan merubahnya ke dalam bentuk *list dictionary* untuk memudahkan penggunaan *high order function*.

```
[4]  filtered_data = list(filter(lambda row: row['zip_code'] != ''
                                 and row['neighborhood'] != ''
                                 and row['totalresponsetime'] != ''
                                 and row['traveltime'] != ''
                                 and row['totaltime'] != '', data))
```

Gambar 3. Memfilter Data

Selanjutnya, kita akan memfilter hanya baris-baris yang memenuhi kriteria dari data dari file csv yang telah kita import, baris yang memiliki nilai kosong pada ('zip_code', 'neighborhood', 'totalresponsetime', 'traveltime', dan 'totaltime'), jadi hanya baris yang memiliki nilai yang akan ditampilkan.

```
[6]  rata2_totalresponsetime = reduce(lambda x, row: x + float(row['totalresponsetime']),
                                      filtered_data, 0) / len(filtered_data)
     rata2_traveltime = reduce(lambda x, row: x + float(row['traveltime']),
                               filtered_data, 0) / len(filtered_data)
     rata2_totaltime = reduce(lambda x, row: x + float(row['totaltime']),
                              filtered_data, 0) / len(filtered_data)

     print(f"Total waktu respons rata-rata: {rata2_totalresponsetime}")
     print(f"Waktu pengiriman rata-rata   : {rata2_traveltime}")
     print(f"Total waktu rata-rata        : {rata2_totaltime}")

     Total waktu respons rata-rata: 8.306335351073573
     Waktu pengiriman rata-rata   : 2.3263520404784326
     Total waktu rata-rata        : 29.530862153463513
```

Gambar 4. Menghitung Rata-Rata Waktu

Selanjutnya kita akan menghitung dan menampilkan rata-rata dari kolom ('*totalresponsetime*', '*traveltime*', '*totaltime*') dan data yang sudah di filter sebelumnya. Pada bagian ini memberikan total waktu rata-rata panggilan layanan 911 dalam periode 30 hari.

```
[7]  samples_neighborhood = {neighborhood: list(filter(lambda row:
                            row['neighborhood'] == neighborhood, filtered_data))
                            for neighborhood in set(map(lambda row: row['neighborhood']
                                                        filtered_data))}
```

Gambar 5. Memisahkan Data

Selanjutnya, kita akan mengelompokkan data yang sudah difilter berdasarkan nilai dari kolom '*neighborhood*', lalu hasilnya adalah sebuah *dictionary*, dimana setiap nilai adalah daftar baris-baris dari '*filtered_data*' yang memiliki nilai '*neighborhood*'.

```
[8]  def menghitung_rata2_waktu(data):
       rata2_totalresponsetime = reduce(lambda x, row: x +
                                        float(row['totalresponsetime']),
                                        data, 0) / len(data)
       rata2_traveltime = reduce(lambda x, row: x + float(row['traveltime']),
                                 data, 0) / len(data)
       rata2_totaltime = reduce(lambda x, row: x + float(row['totaltime']),
                                data, 0) / len(data)

       return {
           'rata2_waktu_respons': rata2_totalresponsetime,
           'rata2_waktu_pengiriman': rata2_traveltime,
           'rata2_total_waktu': rata2_totaltime
           }

     rata2_waktu_neighborhood = []
     for neighborhood, entries in samples_neighborhood.items():
       rata2_waktu_neighborhood.append({'neighborhood': neighborhood,
                                        **menghitung_rata2_waktu(entries)})

     rata2_waktu_neighborhood.append({'neighborhood': 'All Detroit',
                                      **menghitung_rata2_waktu(filtered_data)})

     for row in rata2_waktu_neighborhood:
       print(f"Neighborhood                : {row['neighborhood']}")
       print(f"Total waktu respons rata-rata: {row['rata2_waktu_respons']}")
       print(f"Waktu pengiriman rata-rata   : {row['rata2_waktu_pengiriman']}")
       print(f"Total waktu rata-rata        : {row['rata2_total_waktu']}\n")
```

Gambar 6. Menghitung dan Mencetak Rata-rata

Selanjutnya, kita akan menganalisis efisiensi waktu respons layanan darurat berdasarkan '*neighborhood*'. Kita akan menghitung dan menampilkan rata-rata waktu untuk berbagai wilayah, lalu diterapkannya pada data yang sudah dikelompokkan.

```
[8]  Neighborhood                : Lafayette Park
     Total waktu respons rata-rata: 5.13793103448276
     Waktu pengiriman rata-rata   : 1.313103448275862
     Total waktu rata-rata        : 28.60793103448275

     Neighborhood                : Northeast Central District
     Total waktu respons rata-rata: 9.064705882352941
     Waktu pengiriman rata-rata   : 1.6768907563025202
     Total waktu rata-rata        : 29.97352941176469

     Neighborhood                : Gold Coast
     Total waktu respons rata-rata: 8.719897959183674
     Waktu pengiriman rata-rata   : 2.1260204081632654
     Total waktu rata-rata        : 32.45663265306123
```

Gambar 7. *Output* dari menghitung dan mencetak rata-rata

Gambar diatas adalah output dari *neighborhood,* rata-rata total waktu respons, waktu pengiriman, dan total waktu rata-rata, ini bertujuan untuk menganalisis lebih lanjut.

```
[9]  data_json = json.dumps(rata2_waktu_neighborhood, indent=3)

     print(data_json)

     with open("rata2_waktu.json", "w") as json_file:
       json_file.write(data_json)
```

Gambar 8. Mengonversi ke File *JSON*

Selanjutnya, kita akan mengonversi hasil dari data rata-rata menjadi format file *json*, lalu menyimpannya ke dalam file dengan nama file 'rata2_waktu.json'.


```
    {
        "neighborhood": "LaSalle College Park",
        "rata2_waktu_respons": 14.326666666666668,
        "rata2_waktu_pengiriman": 3.5758974358974345,
        "rata2_total_waktu": 38.37846153846152
    },
    {
        "neighborhood": "Grandmont #1",
        "rata2_waktu_respons": 16.213888888888885,
        "rata2_waktu_pengiriman": 3.995833333333333,
        "rata2_total_waktu": 45.19305555555556
    },
    {
        "neighborhood": "Schaefer 7/8 Lodge",
        "rata2_waktu_respons": 18.592831541218636,
        "rata2_waktu_pengiriman": 5.259139784946239,
        "rata2_total_waktu": 50.00071684587814
    },
    {
        "neighborhood": "All Detroit",
        "rata2_waktu_respons": 8.306335351073573,
        "rata2_waktu_pengiriman": 2.3263520404784326,
        "rata2_total_waktu": 29.530862153463513
    }
```


Gambar 9. *Output* dari konversi

Gambar di atas merupakan *output* dari file yang sudah dikonversi menjadi *json* dari hasil data rata-rata waktu.

4. Kesimpulan

Dari aplikasi yang digunakan kami menggunakan dataset 911_calls_for_service dimulai dengan penggunaan *library csv* dan *json* untuk membaca dan menulis file *csv* serta file *json*. Data kemudian di filter untuk menghilangkan baris yang memiliki nilai kosong pada kolom '*zip_code*', '*neighborhood*', '*totalresponsetime*', '*traveltime*', dan '*totaltime*'. Rata-rata total waktu respons, waktu pengiriman, dan total waktu rata-rata dihitung dan ditampilkan untuk menganalisis efisiensi waktu respons layanan darurat. Data dibuat kelompok berdasarkan nilai '*neighborhood*' dan rata-rata waktu dihitung untuk setiap wilayah. Hasil analisis akhir akan dibuat dalam format *json* dan disimpan ke dalam file 'rata2_waktu.json'. Nilai rata-rata waktu untuk seluruh kawasan kota Detroit yaitu 8,306 untuk total waktu respons rata-rata, 2,326 untuk waktu pengiriman rata-rata, dan 29,531 untuk total waktu rata-rata.

5. Daftar Pustaka

CSV File - Definition and Examples. (2021, June 5). Parse.ly. Retrieved May 15, 2024, from https://www.parse.ly/glossary/csv/

Faradilla. (2022, December 7). *Apa Itu JSON? Penjelasan, Penggunaan, dan Contoh JSON*. Hostinger Tutorial. https://www.hostinger.co.id/tutorial/apa-itu-json

Jain, S. (2022, September 14). *Functools module in Python*. GeeksforGeeks. Retrieved May 15, 2024, from https://www.geeksforgeeks.org/functools-module-in-python/