Laporan Semifinal IMPACT ITB
Problem Solving Komputer
Disusun oleh Tim Altair

A. Kamus (Fungsi, Prosedur, dan Variabel)

Prosedur:

a. **load**→ memuat data dari file statistik yang diberikan user
b. **save**→ menyimpan data yang ditambahkan ke file yang diberikan user
c. **show_prov**→ menampilkan data statistik berdasarkan provinsi yang diberikan
d. **show_date**→ menampilkan data statistik berdasarkan tanggal yang diberikan
e. **sort** → menampilkan data statistik pada tanggal yang diberikan yang diurutkan berdasarkan parameter tertentu, yaitu antara penderita, sembuh, atau jumlah kematian
f. **average** → menampilkan rata-rata penambahan, baik penambahan jumlah penderita, sembuh, atau kematian dari seluruh provinsi pada tanggal tertentu
g. **top**→ menampilkan provinsi serta data terbanyak dan tersedikit pada tanggal tertentu
h. **laju**→ menampilkan laju pertambahan dari suatu provinsi hingga tanggal tertentu
i. **add**→ menambahkan data penambahan kasus penderita, sembuh, maupun jumlah kematian

Fungsi bawaan python yang digunakan:

1. **Import** → memanggil file lain dalam satu module yang sama
2. **If, elif, else**→ fungsi percabangan, untuk menjalankan fungsi dalam suatu kondisi tertentu yang diberikan
3. **Print** → fungsi yang berfungsi untuk menuliskan statement pada terminal
4. **Input** → fungsi yang berfungsi mendapatkan input dari pengguna melalui terminal
5. **For** → fungsi perulangan, untuk melakukan perulangan tugas tertentu selama dalam kondisi tertentu
6. **Open** → membuka file untuk menjalankan fungsi tertentu, yaitu membaca atau menulis file
7. **While** → fungsi perulangan, untuk melakukan perulangan tugas tertentu selama dalam kondisi tertentu
8. **Lower** → fungsi untuk mengubah suatu data string menjadi huruf kecil (lowercase)
9. **Append** → menambahkan data pada suatu list
10. **Reader (csv.reader)** → fungsi untuk membaca file csv

11. **Writer (csv.writer)** → fungsi untuk menulis sesuatu pada file csv
12. **Writerow** → fungsi untuk menulis suatu baris / row dalam suatu csv
13. **Copy** → fungsi untuk menyalin data dalam suatu tipe data dictionary
14. **Clear** → fungsi untuk menghapus semua elemen dalam suatu dictionary
15. **Len** → fungsi untuk mengembalikan nilai dari panjang suatu data string
16. **Sorted** → fungsi untuk mengurutkan suatu data berdasarkan parameter tertentu
17. **Min** → fungsi untuk mengembalikan nilai terendah dalam suatu iterable (misalnya, list)
18. **Max** → fungsi untuk mengembalikan nilai tertinggi dalam suatu iterable (misalnya, list)
19. **Int** → fungsi untuk mengubah tipe suatu variabel menjadi integer
20. **Str** → fungsi untuk mengubah tipe suatu variabel menjadi string
21. **Slice** → fungsi untuk memecah suatu variabel string menjadi beberapa bagian dengan suatu karakter pemisah.

B. Penjelasan source code step by step

Source code terdiri dari 10 fungsi yang dibuat, salah satunya adalah fungsi main sebagai titik awal dan akhir eksekusi program.

Mengimport file module

```
import csv
```

Bagian code ini untuk membuka file lain di module yang sama, yakni file csv, untuk menjalankan fungsi yang berguna untuk memodifikasi file csv. Dalam hal ini, file csv digunakan untuk menyimpan data statistik (COVID-19)

Mendeklarasikan variable global

```
data_covid = []
provinsi_list = []
data_new = []
```

Bagian code ini bertujuan untuk mendeklarasikan 3 variabel global yang digunakan, yaitu:

a. data_covid : data bertipe list untuk menyimpan data statistik yang diambil dari file csv statistik
b. provinsi_list : data bertipe list untuk menyimpan dat provinsi yang ada (provinsi yang ada ini diambil hanya dari data provinsi yang ada dalam data csv statistik)
c. data_new : data bertipe list untuk menyimpan data yang baru ditambahkan

pengguna yang nantinya akan disimpan

Fungsi main

```
def main(): //mendeklarasikan fungsi main
```

Bagian kode ini adalah bagian untuk mendeklarasikan fungsi main(), yaitu fungsi utama dalam program.

```
read = False
    while read == False:
        x = load()
        if x != 1:
            read = True
        else:
            read = False
```

Bagian pertama dari fungsi main adalah untuk memuat file statistik yang akan digunakan dalam program ini. Variabel bertipe boolean yang dideklarasikan, yaitu `read = False` bertujuan sebagai parameter bahwa file sudah dimuat.

Jika file belum dimuat, yaitu ditandai dengan fungsi load() yang mengembalikan nilai 1, maka variabel read akan tetap bernilai false, dan jila file sudah berhasil dimuat maka maka read akan bernilai true.

Program akan terus memanggil fungsi load() jika read bernilai false, yakni menandakan file belum berhasil dimuat.

```
run = True
    while run == True:
        print("Key: ", end='')
        cmd = input()
```

Bagian kode diatas merupakan bagian utama dari program ini, yakni untuk mendapatkan key, atau command yang ingin dijalankan pengguna.

Variabel run, yang dideklarasikan sebagai boolean yang bernilai true, menandakan bahwa program masih berjalan. Variable ini digunakan sebagai kondisi untuk perulangan program untuk mendapatkan key, atau command dari pengguna.

Ketika run bernilai true, maka pengguna akan diminta memasukkan key. Jika run bernilai false, maka program akan menghentikan perulangan dan berhenti meminta pengguna memasukkan key.
Key atau command didapatkan dengan memberi nilai pada variable cmd, dengan input dari pengguna, yaitu dengan code: `cmd = input()`

```
if cmd.lower() == 'exit':
            run = False
```

Setelah pengguna memasukkan key, maka program melakukan fungsi percabangan. Ketika key (atau cmd) yang dimasukkan adalah `exit`, maka program akan mengubah nilai run menjadi false dan menghentikan perulangan untuk meminta key kepada pengguna.

`.lower()` digunakan untuk menghindari kesalahan pengguna menggunakan huruf kapital (membuat case-insensitive menggunakan fungsi lower())

```
elif cmd.lower() == 'help':
            print("add: ...")
            print("average: ...")
            print("exit: ...")
            print("help: ...")
            print("laju: ...")
            print("save: ...")
            print("sort: ...")
            print("show: ...")
            print("sho: ...")
            print("top: ...")
```

Key berikutnya adalah `help`, ketika key yang dimasukkan adalah help, maka program akan menampilkan opsi key yang bisa dimasukkan pengguna beserta cara penggunaan dan fungsinya.

```
elif cmd.lower() == 'show':
            arg1 = input("Tampilkan berdasarkan: ")
            if arg1.lower() == "provinsi":
                arg2 = input("Provinsi: ")
                print("----------")
```

```
            show_prov(arg2)
            print("----------")
        elif arg1.lower() == "tanggal":
            arg2 = input("Tanggal: ")
            print("----------")
            show_date(arg2)
            print("----------")
        else://Penjelasan c
            print("Usage: show ...")
```

Key berikutnya adalah show.

Key show digunakan untuk memanggil fungsi show, yakni menampilkan data berdasarkan tanggal atau provinsi tertentu.

Ketika pengguna memasukkan show, maka program akan meminta parameter untuk menjalankan fungsi show. Parameter pertama adalah untuk cara penampilan data, yakni berdasarkan provinsi atau tanggal. Parameter pertama dimasukkan ke dalam variabel arg1.

Kemudian, program akan menjalankan percabangan if-else berdasarkan arg1.

- Jika arg1 adalah **provinsi**, maka program akan meminta argumen kedua berupa nama provinsi yang ingin ditampilkan, lalu menjalankan fungsi show_prov() dengan memasukkan parameter arg1, dan arg2. (a)
- Jika arg1 adalah **tanggal**, maka program akan meminta argumen kedua berupa tanggal yang datanya ingin ditampilkan, lalu menjalankan fungsi show_date() dengan memasukkan parameter arg1, dan arg2. (b)
- Jika arg1 adalah selain tanggal atau provinsi, program akan menampilkan penggunaan key [show] yang benar, yakni harus memasukkan hanya “provinsi” atau “tanggal” ke dalam arg1. (c)

```
elif cmd.lower() == 'sort':
            arg1 = input("Urutkan berdasarkan: ")
            if arg1.lower() == "penderita":
                arg2 = input("Tanggal: ")
                print("----------")
                sort(arg1.lower(), arg2)
                print("----------")
            elif arg1.lower() == "sembuh":
```

```
            arg2 = input("Tanggal: ")
            print("----------")
            sort(arg1.lower(), arg2)
            print("----------")
        elif arg1.lower() == "kematian":
            arg2 = input("Tanggal: ")
            print("----------")
            sort(arg1.lower(), arg2)
            print("----------")
        else:
            print("Usage: sort ...")
```

Key berikutnya adalah `sort`.

Key sort digunakan untuk memanggil fungsi sort, yakni menampilkan data pada tanggal tertentu yang sudah diurutkan berdasarkan parameter tertentu.

Ketika pengguna memasukkan sort, maka program akan meminta parameter untuk menjalankan fungsi sort. Parameter pertama adalah untuk cara pengurutan, yakni berdasarkan provinsi atau tanggal. Parameter pertama dimasukkan ke dalam variabel arg1.

Kemudian, program akan menjalankan percabangan if-else berdasarkan arg1.

- Jika arg1 adalah **penderita**, maka program akan memasukkan “penderita” sebagai parameter pertama fungsi sort, sehingga sort akan mengurutkan data berdasarkan jumlah penderita. (a)
- Jika arg1 adalah **sembuh**, maka program akan memasukkan “sembuh” sebagai parameter pertama fungsi sort, sehingga sort akan mengurutkan data berdasarkan jumlah penderita yang sembuh. (b)
- Jika arg1 adalah **kematian**, maka program akan memasukkan “kematian” sebagai parameter pertama fungsi sort, sehingga sort akan mengurutkan data berdasarkan jumlah kematian. (c)
- Jika arg1 adalah selain penderita, sembuh, atau kematian, program akan menampilkan penggunaan key [sort] yang benar, yakni harus memasukkan hanya “penderita”, “sembuh”, atau “kematian” ke dalam arg1. (d)

Jika arg1 yang dimasukkan pengguna sesuai dengan pilihan yang ada, yaitu “penderita” atau “sembuh” atau “kematian” maka program kemudian akan meminta pengguna memasukkan input variable arg2 berupa tanggal yang datanya ingin

ditampilkan.
Kemudian, jika penggunaan benar maka program akan memanggil fungsi sort dengan menggunakan parameter arg1 dan arg2.

```
elif cmd.lower() == 'average':
            arg1 = input("Tanggal: ")
            if arg1.lower() != "":
                print("----------")
                average(arg1)
                print("----------")
            else:
                print("Usage: average [tanggal (dd/mm/yyyy)]")
elif cmd.lower() == 'top':
            arg1 = input("Tanggal: ")
            if arg1.lower() != "":
                print("----------")
                top(arg1)
                print("----------")
            else:
                print("Usage: top [tanggal (dd/mm/yyyy)]")
```

Key berikutnya adalah `average` dan top.

Key average digunakan untuk memanggil fungsi average, yakni menampilkan rata rata penambahan kasus COVID-19 pada tanggal tertentu.

Key top digunakan untuk memanggil fungsi top, yakni menampilkan data pertambahan kasus COVID-19 terbanyak dan tersedikit pada tanggal tertentu.

Jika pengguna memasukkan average ataupun top, maka program akan meminta input tanggal yang ingin datanya ditampilkan sebagai variabel arg1, untuk digunakan sebagai parameter fungsi average ataupun top.

Jika arg1 yang dimasukkan kosong, maka program akan menampilkan penggunaan yang benar. Kemudian, jika penggunaan benar maka program akan memanggil fungsi average untuk key [average], dan top untuk key [top] dengan parameter arg1

```
 elif cmd.lower() == 'laju':
             arg1 = input("Provinsi: ")
```

```
            arg2 = input("Hingga tanggal: ")
            if arg1.lower() != "" and arg2.lower() != "":
                print("----------")
                laju(arg1, arg2)
                print("----------")
            else:
                print("Usage: laju ...")
```

Key berikutnya adalah `laju`

Key laju digunakan untuk memanggil fungsi laju, yakni menampilkan laju penambahan rata rata kasus COVID-19 di suatu provinsi hingga tanggal tertentu.

```
elif cmd.lower() == 'add':
            print("----------")
            add()
            print("----------")
```

Key berikutnya adalah `add`

Key add digunakan untuk memanggil fungsi add, yakni menambahkan data baru.

```
elif cmd.lower() == 'save':
            print("----------")
            save()
            data_new.clear()
            print("----------")
```

Key berikutnya adalah `save`

Key save digunakan untuk menyimpan data baru yang ditambahkan dengan memanggil fungsi save. Kemudian, setelah data disimpan di csv, maka data temporary (data_new) akan dihapus elemennya dengan menggunakan fungsi clear()

```
else:
            print("Command tidak ditemukan")
            print("----------")
            print("Daftar command yang ada:")
```

```
            print("..... (Daftar command)") //Print opsi command
            print("----------")
```

Jika key yang dimasukkan tidak ada dalam list, maka program akan menunjukkan daftar program-program yang tersedia.

```
if len(data_new) != 0:
        save()
```

Sebelum program menutup perulangan untuk meminta key kepada pengguna (atau ketika run bernilai false karena pengguna menjalankan fungsi `exit()`) maka program akan memastikan bahwa tidak ada data tambahan yang belum disimpan (ketika `len(data_new) != 0`). Jika ada data tambahan yang belum disimpan, maka program akan memanggil fungsi save() untuk menyimpan data. Setelah itu, maka program kemudian akan ditutup.

Deskripsi Fungsi yang akan digunakan

Fungsi-fungsi tambahan yang digunakan, dideklarasikan pada awal program, guna menghindari kesalahan saat menjalankan program python.
Fungsi yang tersedia adalah:

Fungsi Load

```
def load():
    filename = input("Masukkan nama file data statistik: ")
    try:
        file = open(filename, "r")
    except:
        print("File tidak ditemukan")
        return 1
```

Pertama kali fungsi load dijalankan, program akan meminta nama file data statistik dari input pengguna. Lalu program akan mencoba membuka file csv yang telah diinput dengan metode “r” atau read.

Jika file tidak berhasil dibuka (mengembalikan error) maka program akan menampilkan pesan error “File tidak ditemukan” dan akan mengembalikan nilai 1 (berarti error)

```
    else:
        file_reader = csv.reader(file)
        for provinsi, tanggal, penderita, sembuh, kematian in file_reader:
            if provinsi.lower() != "provinsi":
                data = {"provinsi" : provinsi, "tanggal" : tanggal,
                        "penderita" : int(penderita), "sembuh" :
                         int(sembuh), "kematian" : int(kematian)}
                if provinsi not in provinsi_list:
                    provinsi_list.append(provinsi.lower())
                data_covid.append(data)
        file.close()
        print("File data statistik sudah terbaca")
        return 0
```

Jika file berhasil dibuka, lalu fungsi reader dijalankan untuk membaca file csv yang telah dibuka. Kemudian program akan mengiterasi setiap row dalam csv untuk membaca data yang ada, yakni data provinsi, tanggal, penderita, sembuh, dan kematian.

`if provinsi.lower() != "provinsi":` digunakan agar row pertama sebagai judul dari suatu kolom di tabel csv tidak dimasukkan ke dalam data statistik yang ada di program.

Kemudian, sebuah variabel bertipe dictionary dibuat untuk menyimpan setiap data yang telah dibaca. Data penderita, sembuh, dan kematian akan diubah menjadi integer dengan fungsi `int()` karena tipe yang sesuai untuk data tersebut adalah integer.

Lalu program juga akan menyimpan data list provinsi yang ada. Jika nama provinsi belum tersimpan, maka program akan menambahkan nama provinsi tersebut pada list provinsi_list yang telah dideklarasikan.

Kemudian, data dict yang telah dibuat akan ditambahkan ke dalam data list `data_covid`, Hal ini kemudian diulang dihingga program mencapai akhir dari file csv yang dibaca.

Setelah semua file disimpan dalam program, file csv akan ditutup dengan fungsi close(). Lalu program akan menampilkan pesan sukses, dan mengakhiri program dengan mengembalikan nilai 0 (berarti program yang dijalankan berhasil.

Fungsi save

```
def save():
    filename = input("Masukkan nama file data statistik: ")
    try:
        file = open(filename, "a")
    except:
        print("File tidak ditemukan")
        return 1
```

Pertama kali fungsi save dijalankan, program akan meminta nama file data statistik dari input pengguna. Lalu program akan mencoba membuka file csv yang telah diinput dengan metode “a” atau append.

Jika file tidak berhasil dibuka (mengembalikan error) maka program akan menampilkan pesan error “File tidak ditemukan” dan akan mengembalikan nilai 1 (berarti error)

```
    else:
        file_writer = csv.writer(file, lineterminator='\n')
        for x in range(len(data_new)):
            data_add = [data_new[x]['provinsi'],
                        data_new[x]['tanggal'],
                        data_new[x]['penderita'],
                        data_new[x]['sembuh'],
                        data_new[x]['kematian']]
            file_writer.writerow(data_add)
        file.close()
        print("File data statistik sudah tersimpan")
        for x in range(len(data_new)):
            data_covid.append(data_new[x].copy())
        return 0
```

Jika file berhasil dibuka, lalu fungsi write dijalankan untuk membaca file csv yang telah dibuka. Program kemudian akan mengiterasi seluruh data baru yang tersimpan di list `data_new`, lalu memasukkan elemen dari setiap elemen list `data_new` berupa data dictionary ke dalam variabel `data_add`.

Kemudian, program akan menjalankan fungsi writerow untuk menulis data yang ada di dalam `data_add` ke dalam csv yang diberikan pengguna.

Ketika program sudah selesai mengcopy data baru yang sebelumnya sudah tersimpan ke dalam file csv, maka program akan menjalankan fungsi `close()` untuk menutup file.

Kemudian program akan mengiterasi lagi setiap data di dalam `data_new` untuk menyalinnya ke dalam data seluruhnya, yaitu variabel `data_covid`. Setelah selesai, fungsi diakhiri dan mengembalikan nilai 0 (berarti fungsi berjalan dengan baik)

Fungsi show_prov

```
def show_prov(prov):
    if prov.lower() not in provinsi_list:
        print(f"Data tidak ditemukan untuk provinsi {prov}")
```

Dalam fungsi show_prov, pertama-tama program akan mengecek nama provinsi dengan list provinsi yang tersedia. Jika nama provinsi yang dimasukkan pengguna tidak ada dalam list provinsi maka akan menampilkan pesan error.

```
    else:
        data = []
        data.clear()
        for x in range(len(data_covid)):
            if data_covid[x]["provinsi"].lower() == prov.lower():
                data.append(data_covid[x].copy())
```

Jika provinsi ada dalam list, maka program akan mendeklarasi list data yang akan ditampilkan. Data didapat dengan mengiterasi seluruh data yang ada dalam variabel data_covid, dan jika data tersebut memiliki elemen provinsi sama dengan provinsi yang diminta, maka program akan menyalin data tersebut ke list data.

```
        for y in range(len(data)):
            data[y]['tanggal'] = data[y]['tanggal'].split('/')
        data = sorted(data, key=lambda x: (x['tanggal'][2],
                      x['tanggal'][1], x['tanggal'][0]))
```

Untuk bisa mengurutkan data, pertama-tama program harus mengubah data tanggal menjadi integer untuk bisa disortir, maka program mengiterasi seluruh data yang telah didapat (yang akan ditampilkan), lalu memecah elemen tanggal menjadi list berisi 3 elemen, yaitu tahun, bulan dan tanggal menggunakan fungsi `split()`. Lalu program akan mengurutkan data menggunakan algoritma pengurutan yang ada di python yaitu dengan fungsi `sorted()`, yang parameternya diurutkan berdasarkan tahun, kemudian bulan, dan terakhir tanggal (agar sesuai dengan urutan tanggal, dari data terlama hingga data terbaru).

```
        print(f"Data penyebaran COVID-19 di Provinsi {prov}")
        print("Tanggal    | Penderita | Sembuh | Kematian ")
        for x in range(len(data)):
            print(data[x]['tanggal'][0] + "/" +
                  data[x]['tanggal'][1] + "/" +
                  data[x]['tanggal'][2], "|", data[x]['penderita'],
                          " "*(8 - len(str(data[x]['penderita']))),
                  "|",data[x]['sembuh'],
                           " "*(5 - len(str(data[x]['sembuh']))),
                  "|",data[x]['kematian'])
        data.clear()
```

Langkah terakhir di fungsi `show_prov` adalah menampilkan data yang sudah disortir berdasarkan provinsi dan diurutkan berdasarkan tanggal. Program akan menampilkan seluruh data dengan mengiterasi seluruh data yang terdapat pada variabel data, kemudian mencetaknya di terminal.

Untuk membuat tampilan seperti tabel, yang dilakukan adalah dengan menambahkan spasi di belakang karakter dengan total spasi sebesar total karakter dalam satu kolom yang terbesar, dikurang total karakter dari elemen dari satu kolom. Misalnya, jika satu kolom memiliki karakter terbesar dengan 8 karakter, dan pada suatu baris di kolom itu telah memiliki elemen dengan 4 karakter, maka akan diberi 4 spasi agar posisi kolom berikutnya sejajar.

Fungsi show_date

```
def show_date(date):
    data = []
    data.clear()
```

```
    for x in range(len(data_covid)):
        if data_covid[x]["tanggal"] == date:
            data.append(data_covid[x].copy())
    if len(data) == 0:
        print(f"Data tidak ditemukan pada tanggal {date}")
```

Dalam fungsi `show_date`, hal yang dilakukan kurang lebih sama dengan fungsi `show_prov`. Hanya saja pertama-tama program mengecek keberadaan data dengan tanggal yang dimasukkan bukan provinsi, dengan mengiterasi seluruh data yang ada dalam `data_covid`, jika tanggal sama maka data akan disalin ke dalam list `data`.

```
    else:
        data = sorted(data, key=lambda x: x['provinsi'])
        print(f"Berikut adalah data statistik COVID-19 pada tanggal
                {date}")
        print("Provinsi   | Penderita | Sembuh | Kematian ")
        for x in range(len(data)):
            print(data[x]['provinsi'],
                  " "*(9 - len(data[x]['provinsi'])),
                  "|",data[x]['penderita'],
                  " "*(8 - len(str(data[x]['penderita']))),
                  "|",data[x]['sembuh'],
                  " "*(5 - len(str(data[x]['sembuh']))),
                  "|",data[x]['kematian'])
    data.clear()
```

Langkah berikutnya setelah data ditemukan adalah menyortir data yang ada. Namun, parameter yang digunakan berbeda dengan penyortiran pada show_prov, yaitu menggunakan parameter elemen provinsi. Ketika telah disortir, program akan mengiterasi seluruh data yang telah didapat, kemudian mencetaknya dengan algoritma yang sama dengan show_prov, untuk mendapatkan tampilan tabel.

Fungsi sort

```
def sort(by, date):
    data = []
    data.clear()
    for x in range(len(data_covid)):
        if data_covid[x]["tanggal"] == date:
```

```
            data.append(data_covid[x].copy())
    if len(data) == 0:
        print(f"Data tidak ditemukan pada tanggal {date}")
```

Pada fungsi sort, program kembali menggunakan algoritma yang sama untuk menemukan data, yakni dengan iterasi seluruh data dan mencocokkan elemen tanggal dengan tanggal yang diberikan pengguna, jika sama maka data akan disalin.

```
    else:
        data = sorted(data, key=lambda x: x[by])
        if by == "penderita":
            print(f"Berikut adalah data statistik COVID-19 pada
                    tanggal {date} terurut menaik berdasarkan jumlah
                    kumulatif penderita")
        elif by == "sembuh":
            print(f"Berikut adalah data statistik COVID-19 pada
                    tanggal {date} terurut menaik berdasarkan jumlah
                    kumulatif penderita yang sembuh")
        else:
            print(f"Berikut adalah data statistik COVID-19 pada
                    tanggal {date} terurut menaik berdasarkan angka
                    kematian")
```

Langkah berikutnya jika data ditemukan pada tanggal tersebut adalah melakukan pengurutan dengan fungsi sorted. Kali ini, parameter yang digunakan sorted bersifat dinamis, berdasarkan argumen yang diberikan oleh pengguna, yakni variabel “by”. Kemudian setelah disortir, program akan pertama-tama mencetak judul tabel sesuai dengan cara pengurutan yang dilakukan.

```
        print("Provinsi   | Penderita | Sembuh | Kematian ")
        for x in range(len(data)):
            print(data[x]['provinsi'],
                  " "*(9 - len(data[x]['provinsi'])),
                  "|",data[x]['penderita'],
                  " "*(8 - len(str(data[x]['penderita']))),
                  "|",data[x]['sembuh'],
                  " "*(5 - len(str(data[x]['sembuh']))),
                  "|",data[x]['kematian'])
```

```
    data.clear()
```

Selanjutnya, program mencetak tabel data dengan cara yang sama seperti sebelumnya, yaitu mengiterasi seluruh data, dan mencetak dengan tambahan spasi untuk membuat tampilan seperti tabel.

Fungsi average

```
def average(date):
    penderita = 0
    sembuh = 0
    kematian = 0
    total = 0
    for x in range(len(data_covid)):
        if data_covid[x]["tanggal"] == date:
            penderita += data_covid[x]["penderita"]
            sembuh += data_covid[x]["sembuh"]
            kematian += data_covid[x]["kematian"]
            total += 1
    if total == 0:
        print(f"Data tidak ditemukan pada tanggal {date}")
```

Pada fungsi average, pertama-tama program akan mengecek total dari penderita, sembuh, dan kematian dalam data dict di list data_covid yang memiliki tanggal sesuai dengan yang diberikan pengguna. Jika tanggal cocok, maka data penderita, sembuh, dan kematian akan ditambahkan ke dalam jumlah setiap variabel, serta total data akan dijumlahkan setiap data ditemukan. Jika data tidak ditemukan maka program akan menampilkan error bahwa data tidak ditemukan.

```
else:
        avg_penderita = int(round(penderita/total, 0))
        avg_sembuh = int(round(sembuh/total, 0))
        avg_kematian = int(round(kematian/total, 0))
        print(f"Berikut adalah rata-rata data statistik COVID-19
                pada tanggal {date}")
        print(f"Jumlah rata-rata penderita = {avg_penderita}")
        print(f"Jumlah rata-rata penderita yang sembuh =
                {avg_sembuh}")
        print(f"Angka kematian rata-rata = {avg_kematian}")
```

Kemudian, jika data ditemukan, untuk mencari rata-rata dalam suatu tanggal, caranya adalah dengan menggunakan jumlah penderita/sembuh/kematian, lalu dibagi dengan total jumlah seluruh data yang didapat. Kemudian, program akan mencetaknya ke dalam terminal.

Fungsi add

```
def add():
    print("Masukkan informasi statistik COVID-19 yang
           ditambahkan:")
    prov = "falsedata"
    while prov.lower() not in provinsi_list:
        prov = input("Masukkan nama provinsi: ")
        if prov.lower() not in provinsi_list:
            print(f"Nama provinsi tidak ditemukan, masukkan nama
                    provinsi yang sesuai")
    date = input("Masukkan tanggal pencatatan (DD/MM/YYYY): ")
    penderita = input("Masukkan jumlah penderita yang tercatat: ")
    sembuh = input("Masukkan jumlah penderita yang sudah sembuh: ")
    kematian = input("Masukkan angka kematian akibat COVID-19: ")
    data = {"provinsi" : prov, "tanggal" : date,
            "penderita" : penderita, "sembuh" : sembuh,
            "kematian" : kematian}
    data_new.append(data)
```

Jika fungsi add dijalankan, pertama-tama program akan meminta pengguna untuk memasukan nama provinsi yang datanya ingin ditambahkan. Jika nama provinsi yang dituliskan oleh pengguna tidak terdapat pada list `provinsi_list`, maka program akan memberikan pesan error, dan meminta pengguna untuk menuliskan ulang nama provinsi sampai nama provinsi sesuai dengan yang ada pada data.

Jika nama provinsi yang dituliskan oleh pengguna terdapat pada data, maka program akan meminta lagi kepada pengguna untuk menginput tanggal, jumlah penderita, jumlah sembuh, dan jumlah kematian dari data yang ingin ditambahkan. Setelah semua data telah dimasukan, program akan membuat data dict berisi elemen yang telah diminta kepada pengguna, lalu ditambahkan ke list sementara, data yang akan ditambahkan, yaitu `data_new` sebelum data tersebut disimpan ke file csv.

Fungsi Top

```
def top(date):
    total = 0
    penderita = []
    sembuh = []
    kematian = []

    for x in range(len(data_covid)):
        if data_covid[x]['tanggal'] == date:
            penderita.append(data_covid[x]['penderita'])
            sembuh.append(data_covid[x]['sembuh'])
            kematian.append(data_covid[x]['kematian'])
            total += 1
    if total == 0:
        print(f"Data tidak ditemukan pada tanggal {date}")
```

Pertama, di fungsi top, untuk mencari data penambahan kasus terbanyak dan tersedikit, program akan mengecek data yang memiliki elemen tanggal sesuai yang diinginkan pengguna. Jika tanggal sama, maka program akan menambahkan data jumlah penderita, sembuh, dan kematian ke list data yang sesuai. Jika total data dengan tanggal sama adalah nol, maka program akan mengirimkan pesan error kepada pengguna.

Pada saat fungsi top dijalankan, maka pengguna pertama diminta untuk mengisi data tanggal untuk pencetakan data terbanyak dan tersedikit. Apabila tanggal tidak terdapat pada data, maka program akan mencetak `Data tidak ditemukan pada tanggal {date}`.

```
    else:
            penderita_banyak = []
            sembuh_banyak = []
            kematian_banyak = []
            penderita_sedikit = []
            sembuh_sedikit = []
            kematian_sedikit = []
```

```
lowest_penderita = min(penderita)
highest_penderita = max(penderita)
lowest_sembuh = min(sembuh)
highest_sembuh = max(sembuh)
lowest_kematian = min(kematian)
highest_kematian = max(kematian)
```

Jika tanggal yang dituliskan oleh pengguna terdaftar pada data, program akan mencari jumlah penderita terbanyak dengan fungsi max, dan tersedikit dengan fungsi min, yang dilihat berdasarkan list data jumlah penderita, sembuh dan kematian yang telah dibuat sebelumnya.

```
for x in range(len(data_covid)):
    if data_covid[x]['penderita'] == lowest_penderita:
        penderita_sedikit.append(data_covid[x])
    elif data_covid[x]['penderita'] == highest_penderita:
        penderita_banyak.append(data_covid[x])

    if data_covid[x]['sembuh'] == lowest_sembuh:
        sembuh_sedikit.append(data_covid[x])
    elif data_covid[x]['sembuh'] == highest_sembuh:
        sembuh_banyak.append(data_covid[x])

    if data_covid[x]['kematian'] == lowest_kematian:
        kematian_sedikit.append(data_covid[x])
    elif data_covid[x]['kematian'] == highest_kematian:
        kematian_banyak.append(data_covid[x])
```

Kemudian, setelah menemukan jumlah terbanyak dan tersedikit, program akan mengiterasi seluruh data `data_covid` yang memiliki angka penderita/sembuh/kematian terbanyak dan tersedikit. Jika nilainya sama, maka data itu akan dicopy ke masing-masing variabel yang berhubungan, misalnya `penderita_banyak` untuk penderita terbanyak, dst.

```
print(f"Berikut adalah data statistik COVID-19 pada tanggal
        {date}")
print("Dengan :")
```

```
print("1. Jumlah Kumulatif Penderita Terbanyak")
for x in range(len(penderita_banyak)):
    print(penderita_banyak[x]['provinsi'],
      " "*(9 - len(penderita_banyak[x]['provinsi'])),
      "|",penderita_banyak[x]['penderita'],
      " "*(8 - len(str(penderita_banyak[x]['penderita']))),
      "|",penderita_banyak[x]['sembuh'],
      " "*(5 - len(str(penderita_banyak[x]['sembuh']))),
      "|",penderita_banyak[x]['kematian'])
print("2. Jumlah Kumulatif Penderita Tersedikit")
for x in range(len(penderita_sedikit)):
      print(penderita_sedikit[x]['provinsi'],
      " "*(9 - len(penderita_sedikit[x]['provinsi'])),
      "|",penderita_sedikit[x]['penderita'],
      " "*(8 -len(str(penderita_sedikit[x]['penderita']))),
      "|",penderita_sedikit[x]['sembuh'],
      " "*(5 - len(str(penderita_sedikit[x]['sembuh']))),
      "|",penderita_sedikit[x]['kematian'])
print("3. Jumlah Kumulatif Penderita yang Sembuh
       Terbanyak")
for x in range(len(sembuh_banyak)):
      print(sembuh_banyak[x]['provinsi'],
      " "*(9 - len(sembuh_banyak[x]['provinsi'])),
      "|",sembuh_banyak[x]['penderita'],
      " "*(8 - len(str(sembuh_banyak[x]['penderita']))),
      "|",sembuh_banyak[x]['sembuh'],
      " "*(5 - len(str(sembuh_banyak[x]['sembuh']))),
      "|",sembuh_banyak[x]['kematian'])
print("4. Jumlah Kumulatif Penderita yang Sembuh
      Tersedikit")
for x in range(len(sembuh_sedikit)):
    print(sembuh_sedikit[x]['provinsi'],
    " "*(9 - len(sembuh_sedikit[x]['provinsi'])),
    "|",sembuh_sedikit[x]['penderita'],
    " "*(8 - len(str(sembuh_sedikit[x]['penderita']))),
    "|",sembuh_sedikit[x]['sembuh'],
    " "*(5 - len(str(sembuh_sedikit[x]['sembuh']))),
    "|",sembuh_sedikit[x]['kematian'])
```

```
print("5. Angka Kematian Terbanyak")
for x in range(len(kematian_banyak)):
    print(kematian_banyak[x]['provinsi'],
    " "*(9 - len(kematian_banyak[x]['provinsi'])),
    "|",kematian_banyak[x]['penderita'],
    " "*(8 - len(str(kematian_banyak[x]['penderita']))),
    "|",kematian_banyak[x]['sembuh'],
    " "*(5 - len(str(kematian_banyak[x]['sembuh']))),
    "|",kematian_banyak[x]['kematian'])
print("6. Angka Kematian Tersedikit")
for x in range(len(kematian_sedikit)):
    print(kematian_sedikit[x]['provinsi'],
    " "*(9 - len(kematian_sedikit[x]['provinsi'])),
    "|",kematian_sedikit[x]['penderita'],
    " "*(8 - len(str(kematian_sedikit[x]['penderita']))),
    "|",kematian_sedikit[x]['sembuh'],
    " "*(5 - len(str(kematian_sedikit[x]['sembuh']))),
    "|",kematian_sedikit[x]['kematian'])
```

Langkah terakhir dalam fungsi top, adalah mencetak daftar data terbanyak dan tersedikit. Program akan mencetak semua data yang memiliki nilai tersedikit atau terbanyak, walaupun misalnya ada data yang jumlahnya sama-sama tersedikit. Dan dengan cara yang sama seperti sebelumnya, program akan mencetak dengan tampilan seperti tabel.

Fungsi Laju

```
def laju(prov, date):
    if prov.lower() not in provinsi_list:
        print(f"Tidak ditemukan data untuk provinsi {prov}")
```

Fungsi terakhir adalah `laju`. Pertama-tama fungsi mengambil argumen provinsi dan tanggal untuk menghitung laju rata-rata penambahan. Kemudian program akan mengecek apakah daftar provinsi ada di list, jika tidak ada maka akan mengembalikan pesan error.

```
    else:
        date_print = date
        data = []
        data_laju = []
        penderita = 0
        sembuh = 0
        kematian = 0
        total_laju = 0
        date = date.split('/')
        for x in range(len(data_covid)):
            if data_covid[x]['provinsi'].lower() == prov.lower():
                data.append(data_covid[x].copy())
```

Untuk mendapatkan laju, program akan mulai mencari data dengan elemen provinsi yang sama, jika data tersebut memiliki provinsi yang sama, maka data akan disalin ke data sementara, yaitu variabel `data`.

```
        for y in range(len(data)):
            data[y]['tanggal'] = data[y]['tanggal'].split('/')
            if int(data[y]['tanggal'][2]) < int(date[2]):
                data_laju.append(data[y])
                penderita += data[y]['penderita']
                sembuh += data[y]['sembuh']
                kematian += data[y]['kematian']
                total_laju += 1
            elif int(data[y]['tanggal'][2]) == int(date[2]):
                if int(data[y]['tanggal'][1]) < int(date[1]):
                    data_laju.append(data[y])
                    penderita += data[y]['penderita']
```

```
                sembuh += data[y]['sembuh']
                kematian += data[y]['kematian']
                total_laju += 1
            elif int(data[y]['tanggal'][1]) == int(date[1]):
                if int(data[y]['tanggal'][0]) <= int(date[0]):
                    data_laju.append(data[y])
                    penderita += data[y]['penderita']
                    sembuh += data[y]['sembuh']
                    kematian += data[y]['kematian']
                    total_laju += 1
```

Kemudian, karena yang akan dicari adalah laju rata rata “hingga tanggal tertentu” maka program harus mencari data penambahan kasus sebelum tanggal tersebut. Maka program akan menyaring data yang sudah ada berdasarkan tanggal. Hal pertama yang disaring adalah tahun, kemudian bulan, hingga tanggal. Tidak lupa program juga menghitung jumlah data yang ada.

```
if total_laju <= 0:
    print(f"Data tidak ditemukan untuk laju hingga tanggal
            {date_print}")
else:
    laju_penderita = int(round(penderita/total_laju, 0))
    laju_sembuh = int(round(sembuh/total_laju, 0))
    laju_kematian = int(round(kematian/total_laju, 0))

    print(f"Laju rata-rata penyebaran dan pencegahan
            COVID-19 di provinsi {prov} sampai dengan
            tanggal {date_print}")
    print(f"Penderita : {laju_penderita}")
    print(f"Sembuh    : {laju_sembuh}")
    print(f"Kematian  : {laju_kematian}")
```

Langkah terakhir adalah mencetak laju. Laju didapat dengan hasil penjumlahan semua data sebelum tanggal yang dimasukkan lalu dibagi dengan total data yang ada.