

# Kata Pengantar

**Modul Pembelajaran Pemrograman Berorientasi Objek (PBO)**

Sebagai langkah awal dalam pembelajaran dunia pemrograman, Modul Pembelajaran Pemrograman Berorientasi Objek (PBO) ini disusun dengan tujuan utama memberikan landasan yang kokoh bagi peserta didik dalam memahami konsep-konsep dasar Pemrograman Berorientasi Objek. Pemrograman Berorientasi Objek telah menjadi pijakan utama dalam pengembangan perangkat lunak modern, dan pemahaman yang mendalam terhadap paradigma ini merupakan hal yang esensial bagi setiap calon pengembang perangkat lunak.

Modul ini didesain untuk membimbing peserta didik dari pengenalan dasar hingga pemahaman mendalam tentang prinsip-prinsip Pemrograman Berorientasi Objek. Dalam perjalanan pembelajaran ini, peserta didik akan diajak untuk menggali konsep kelas, objek, pewarisan, polimorfisme, abstraksi, dan konsep-konsep penting lainnya. Setiap materi disajikan dengan pendekatan yang jelas dan aplikatif, dilengkapi dengan contoh-contoh praktis untuk memudahkan pemahaman. Modul ini juga menekankan pada pengaplikasian konsep PBO dalam situasi dunia nyata, dengan proyek-proyek mini yang dirancang untuk memberikan pengalaman praktis kepada peserta didik.Pengembangan perangkat lunak tidak hanya tentang sintaks dan struktur kode, tetapi juga tentang pemahaman mendalam terhadap masalah dan kebutuhan pengguna. Oleh karena itu, modul ini tidak hanya membahas aspek teknis PBO, tetapi juga menyoroti pentingnya berpikir analitis, merancang solusi yang efektif, dan berkolaborasi dalam pengembangan perangkat lunak.Kami berharap bahwa modul ini dapat menjadi panduan yang bermanfaat bagi peserta didik dalam memahami dan menguasai Pemrograman Berorientasi Objek. Semoga perjalanan pembelajaran ini dapat membuka pintu bagi pengetahuan lebih lanjut dan memberikan dasar yang solid bagi pengembangan keterampilan pemrograman berkelanjutan.

Rabu, 24 Januari 2024

Kelompok 3

# DAFTAR ISI

[Kata Pengantar 2](#_Toc157331889)

[DAFTAR ISI 3](#_Toc157331890)

[BAB I 5](#_Toc157331891)

[Pendahuluan 5](#_Toc157331892)

[1. LATAR BELAKANG 5](#_Toc157331893)

[1.1Pemrograman Berorientasi Objek (PBO) 5](#_Toc157331894)

[1.2 RUANG LINGKUP 6](#_Toc157331895)

[1.3 Pemrograman Secara Modular 8](#_Toc157331896)

[1.4 Keuntungan Pemrograman Modular 8](#_Toc157331897)

[BAB II 9](#_Toc157331898)

[PEMBAHASAN 9](#_Toc157331899)

[PERTEMUAN 1 9](#_Toc157331900)

[2.1 Pengertian Modul Pada Python 9](#_Toc157331901)

[2.2 Paket Pada Python 12](#_Toc157331902)

[2.3 Membuat Paket 12](#_Toc157331903)

[2.4 Mengimpor Paket 13](#_Toc157331904)

[2.5 Impor Paket/Modul dengan Alias 14](#_Toc157331905)

[2.6 Impor Paket/Modul secara Kustom 15](#_Toc157331906)

[2.7 Menampilkan Path/Nama File dari Modul 15](#_Toc157331907)

[2.8 Beberapa Paket dan Modul Bawaan Python 15](#_Toc157331908)

[2.9 Poin-Poin Penting 16](#_Toc157331909)

[PERTEMUAN 3 17](#_Toc157331910)

[CONTOH PROGRAM MENGHITUNG BANGUN RUANG MENGGUNAKAN BAHASA PYTHON 17](#_Toc157331911)

[3.1.BALOK 17](#_Toc157331912)

[3.2. BOLA 18](#_Toc157331913)

[3.3. KERUCUT 19](#_Toc157331914)

[3.4. KUBUS 20](#_Toc157331915)

[3.5. LIMAS SEGIEMPAT 21](#_Toc157331916)

[3.6. LIMAS SEGITIGA 22](#_Toc157331917)

[3.7 .PRISMA SEGITIGA 23](#_Toc157331918)

[3.8 .SELINDER(TABUNG) 24](#_Toc157331919)

[PERTEMUAN 4 25](#_Toc157331920)

[4.1 CARA MEMBUAT JADWAL MATAKULIAH MENGGUNAKAN BAHASA PYTHON 25](#_Toc157331921)

[PERTEMUAN 6 32](#_Toc157331922)

[5.1 MULTIFORM TRANSLATOR 32](#_Toc157331923)

[PERTEMUAN 7 35](#_Toc157331924)

[6.1 APLIKASIN BANTU (TOOLS) 35](#_Toc157331925)

[PERTEMUAN 8 41](#_Toc157331926)

[CLASS DAN OBJEK PADA PYTHON 41](#_Toc157331927)

[7.1 CLASS 41](#_Toc157331928)

[7.2 OBJEK 41](#_Toc157331929)

[7.3 PENGGUNAAN CLASS DAN OBJEK 42](#_Toc157331930)

[7.4 Membuat aplikasi konversi suhu dengan python 43](#_Toc157331931)

[PERTEMUAN 9 52](#_Toc157331932)

[PENGENALAN MYSQL 52](#_Toc157331933)

[8.1 PENGERTIAN MYSQL 52](#_Toc157331934)

[8.2 FUNGSI MYSQL 52](#_Toc157331935)

[8.3 KELEBIHAN MYSQL 52](#_Toc157331936)

[8.4 KEKURANGAN MYSQL 54](#_Toc157331937)

[8.5 CARA KERJA MYSQL 54](#_Toc157331938)

[PERTEMUAN 10 56](#_Toc157331939)

[PYTHON DATABASE APPLICATION 56](#_Toc157331940)

[9.1 DATABASE CONNECTION 56](#_Toc157331941)

[9.2 MEMBUAT TABEL DATABASE 57](#_Toc157331942)

[9.3 OPERASI INSERT 57](#_Toc157331943)

[9.5 READ OPERATION 58](#_Toc157331944)

[9.6 UPDATE OPERATION 60](#_Toc157331945)

[9.7 DELETE OPERATION 60](#_Toc157331946)

[PERTEMUAN 11 62](#_Toc157331947)

[10.1 Pengertian MySQL : 62](#_Toc157331948)

[10.2 Pengertian Database : 62](#_Toc157331949)

[Pertemuan 13 74](#_Toc157331950)

[11.1 Membuat aplikasi login dalam python database 74](#_Toc157331951)

# BAB I

# Pendahuluan

## 1. LATAR BELAKANG

## 1.1Pemrograman Berorientasi Objek (PBO)

Pemrograman Berorientasi Objek (PBO) telah menjadi paradigma pengembangan perangkat lunak yang mendasar dalam industri teknologi informasi. Paradigma ini menawarkan pendekatan yang lebih terstruktur dan modular dalam pengembangan perangkat lunak dibandingkan dengan paradigma-proadigma sebelumnya. Berikut adalah beberapa latar belakang penting dari Pemrograman Berorientasi Objek:

1. Penanganan Kompleksitas:

PBO muncul sebagai solusi untuk menangani kompleksitas yang semakin meningkat dalam pengembangan perangkat lunak. Dengan memodelkan sistem sebagai kumpulan objek yang saling berinteraksi, PBO memungkinkan pemisahan tugas dan tanggung jawab, sehingga memudahkan pengembangan, pemeliharaan, dan perluasan perangkat lunak.

1. Reusabilitas Kode:

Salah satu tujuan utama PBO adalah meningkatkan reusabilitas kode. Dengan memisahkan fungsionalitas ke dalam kelas-kelas dan objek-objek, pengembang dapat dengan mudah menggunakan dan menggabungkan komponen-komponen tersebut dalam proyek-proyek berbeda. Hal ini mengurangi redundansi, meningkatkan efisiensi, dan mempercepat proses pengembangan.

1. Peningkatan Produktivitas:

PBO memberikan alat dan konsep-konsep yang mendukung pemrograman yang lebih efisien. Pengembang dapat fokus pada desain dan implementasi fitur-fitur khusus tanpa harus terlalu khawatir tentang detail implementasi internal dari setiap bagian sistem. Ini memungkinkan pengembangan perangkat lunak lebih cepat dan produktif.

1. Fleksibilitas dan Perluasan:

Konsep pewarisan dan polimorfisme dalam PBO memungkinkan fleksibilitas dan perluasan perangkat lunak dengan lebih mudah. Dengan cara ini, sistem yang dibangun dengan PBO dapat menyesuaikan diri dengan perubahan persyaratan bisnis tanpa mengharuskan modifikasi besar pada kode yang sudah ada.

1. Pengembangan Aplikasi GUI:

PBO sangat cocok untuk pengembangan aplikasi dengan antarmuka pengguna grafis (GUI). Framework GUI umumnya berbasis konsep-konsep PBO, memudahkan pengembang dalam membuat aplikasi yang intuitif dan menarik.

## 1.2 RUANG LINGKUP

Modul Pembelajaran Pemrograman Berorientasi Objek (PBO)

Modul pembelajaran PBO ini dirancang untuk memberikan pemahaman mendalam kepada peserta didik mengenai konsep-konsep dasar dan lanjutan dalam Pemrograman Berorientasi Objek. Berikut adalah ruang lingkup modul ini:

1. Pengenalan Pemrograman Berorientasi Objek (PBO):

- Definisi dan filosofi PBO.

- Perbedaan antara PBO dan paradigma pemrograman lainnya.

- Manfaat dan keunggulan PBO dalam pengembangan perangkat lunak.

2. Dasar-dasar PBO:

- Konsep kelas dan objek.

- Enkapsulasi: Penggunaan access modifiers.

- Penggunaan constructor dan destructor.

3. Pewarisan (Inheritance):

- Konsep pewarisan.

- Penerapan pewarisan untuk membangun hierarki kelas.

- Overriding dan overloading.

4. Polimorfisme:

- Polimorfisme statis dan dinamis.

- Penggunaan interface dan implementasinya.

5. Abstraksi:

- Penggunaan abstract class.

- Penggunaan interface untuk menciptakan abstraksi.

- Studi kasus penerapan abstraksi dalam pengembangan perangkat lunak.

6. Encapsulation Lanjutan:

- Penggunaan setter dan getter.

- Penggunaan properties dan indexers.

- Contoh implementasi enkapsulasi dalam situasi praktis.

7. Event Handling dalam PBO:

- Konsep dasar penanganan peristiwa.

- Menerapkan listener dan event pada kelas.

- Studi kasus penggunaan event handling dalam antarmuka pengguna.

8. Antarmuka Pengguna Grafis (GUI):

- Pengenalan GUI dalam PBO.

- Menerapkan komponen GUI seperti JFrame, JButton, dan JTextField.

- Proyek mini: Pembuatan aplikasi GUI sederhana.

9. Koneksi Database dalam PBO:

- Integrasi PBO dengan database.

- Menggunakan objek untuk interaksi dengan database.

- Proyek mini: Aplikasi PBO dengan koneksi database.

10. Pengujian dan Debugging PBO:

- Teknik-teknik pengujian PBO.

- Debugging dalam konteks PBO.

- Studi kasus pengujian dan debugging.

11.Proyek Pengembangan Perangkat Lunak:

- Menerapkan konsep PBO dalam proyek perangkat lunak.

- Kolaborasi tim dalam pengembangan proyek.

- Presentasi dan evaluasi proyek.

Modul ini dirancang agar peserta didik dapat memahami konsep-konsep PBO dan mampu mengimplementasikannya dalam situasi praktis melalui latihan, proyek, dan studi kasus. Modul ini juga menekankan aspek kolaboratif dalam pengembangan perangkat lunak, memberikan wawasan menyeluruh tentang bagaimana tim dapat bekerja bersama untuk mencapai tujuan pengembangan yang sukses.

## 1.3 Pemrograman Secara Modular

Dalam dunia pemrograman, pendekatan modular berarti proses memecah program super besar menjadi beberapa bagian atau kecil yang terpisah, terorganisir, terkelola dengan lebih rapi sesuai dengan tugasnya masing-masing .

## 1.4 Keuntungan Pemrograman Modular

Ada beberapa keuntungan jika kita menerapkan pendekatan modular, di antarnya:

* **Simplicity**: kode program kita menjadi lebih sederhana.
* **Maintainability**: kode program kita menjadi lebih mudah di-maintain atau ‘dipelihara’.
* **Reusability**: potongan-potongan kode program bisa digunakan dari berbagai tempat membuatnya menjadi lebih reusable.

Pada python, terdapat 3 komponen penting penyusun modularitas :

* [Fungsi](https://jagongoding.com/python/dasar/fungsi)
* Modul
* Paket

# BAB II

# PEMBAHASAN

# PERTEMUAN 1

## 2.1 Pengertian Modul Pada Python

Kita telah membahas tentang [fungsi](https://jagongoding.com/python/dasar/fungsi) pada [seri tutorial python dasar](https://jagongoding.com/python/dasar/overview). Ia merupakan blok kode tertentu yang **memiliki nama**, dan bertugas untuk menyelesaikan satu set perintah tertentu, serta bisa dipanggil dari bagian program mana pun.

Hanya saja, kita belum pernah mencoba membuat fungsi pada suatu file, dan memanggilnya pada file yang lain.

Oke.

Kita kenalan dengan modul dulu.

Apa itu module pada python? Sederhananya, module pada python adalah sebuah file berekstensi .py yang berisi skrip python. Nama dari modul adalah nama nama dari file itu sendiri . Misal kita memiliki file bernama ‘matematika.py’, maka kita telah membuat sebuah modul bernama ‘matematika’.

Dan modul sendiri bisa memiliki berbagai macam isi, baik itu fungsi, class, mau pun variabel.

**Membuat Modul**

Kita langsung praktikkan saja.

Buat file seperti ini:

src**/**

├── skrip**-**utama**.**py

└── matematika**.**py

**Rencana**:

* File skrip-utama.py adalah file yang akan kita running.
* File matematika.py adalah file yang akan kita jadikan modul.

Karena itu, buka file matematika.py kemudian tuliskan kode program berikut:

pi = 22 / 7

def luas\_persegi (sisi):

  return sisi \* sisi

def luas\_lingkaran (radius):

  return pi \* radius \* radius

**Penjelasan:**

Pada modul matematika di atas, kita membuat 3 buah item:

* variabel pi dengan nilai hasil bagi dari 22/7
* fungsi luas\_persegi
* fungsi luas\_lingkaran

Kita akan mencoba memanggil 3 buah item tersebut dari file yang berbeda.

**Mengimpor Modul Dengan Statemen import**

Jika kita ingin memanggil fungsi atau apa pun yang ada pada file lain, maka kita harus mengimpornya terlebih dahulu.

Terdapat beberapa cara untuk mengimpor.

**Impor semua**

Yang pertama adalah: impor semua. Alias kita mengimpor semua item yang ada pada suatu modul tanpa pengecualian.

Gimana caranya?

Silakan buka file skrip-utama.py kemudian tuliskan kode program berikut:

import matematika

print("PI:", matematika.pi)

print(matematika.luas\_lingkaran(radius=21))

print(matematika.luas\_persegi(sisi=12))

Pada kode program di atas, kita telah memanggil modul matematika dengan perintah:

**import** matematika

Hal ini akan membuat kita memiliki akses terhadap modul tersebut dan apa yang ada di dalamnya.

Oleh karena itu ketika kita jalankan program, kita akan mendapatkan output seperti berikut:

PI: 3.142857142857143

1386.0

144

**Impor secara eksplisit**

Selain mengimpor semua item, kita juga bisa mengimpor hanya item-item tertentu saja secara eksplisit.

Misalkan pada modul matematika, kita hanya ingin mengimpor fungsi luas\_persegi() dan luas\_lingkaran() saja, kita bisa melakukannya seperti ini:

from matematika import luas\_persegi, luas\_lingkaran

print(luas\_persegi(20))

print(luas\_lingkaran(10))

**Menyembunyikan Item Agar Tidak Bisa Diimport**

Selain dengan mendifinisikan satu-satu, kita juga bisa melakukan import \* seperti ini:

from matematika import \*

print('nilai pi:', pi)

Perintah di atas akan mengimport semua item yang ada pada modul matematika, kecuali item yang namanya diawali dengan \_ .

Untuk mencobanya, kita bisa membuat sebuah fungsi di dalam modul matematika dengan nama \_luas\_segitiga().

...

def \_luas\_segitiga (alas, tinggi):

  return 0.5 \* alas \* tinggi

Panggil dengan cara import \* pada file skrip-utama.py.

from matematika import \*

print(luas\_persegi(20))

print(luas\_lingkaran(10))

print(\_luas\_segitiga(20, 10))

Semuanya akan terimport, kecuali satu fungsi saja yaitu \_luas\_segitiga, karena ia diawali dengan underscore:

Exception has occurred: NameError

name '\_luas\_segitiga' is not defined

## 2.2 Paket Pada Python

Paket pada python adalah sebuah cara untuk mengelola dan mengorganisir modul-modul python dalam bentuk direktori, memungkinkan sebuah modul untuk diakses menggunakan “namespace” dan dot lokasi [[4]](https://docs.python.org/3/tutorial/modules.html#packages).

Secara sederhana, paket (atau packages) pada python adalah direktori biasa, yang di dalamnya terdapat modul-modul python.

Bedanya, di dalam direktori tersebut, harus ada satu file bernama \_\_init\_\_.py yang memberitahukan interpreter bahwa direktori tersebut adalah paket yang berisi modul-modul.

Kita langsung praktekkan saja.

## 2.3 Membuat Paket

Anggap saja kita akan membuat aplikasi untuk menghitung luas bangun datar dan juga volume bangun ruang.

Bangun datar di antaranya:

* persegi
* lingkaran
* segitiga

Bangun ruang di antaranya:

* kubik
* trapesium
* bola

Dan kita akan membuat setiap bangun datar atau setiap bangun ruang menjadi satu modul tersendiri. Sehingga, kita akan memiliki total 6 modul. Dan dari 6 modul tersebut akan kita bagi menjadi 2 paket.

Sehingga penampilan struktur programnya adalah berikut:

src/

├── luas

│   ├── \_\_init\_\_.py

│   ├── lingkaran.py

│   ├── persegi.py

│   └── segitiga.py

├── volume

│   ├── \_\_init\_\_.py

│   ├── bola.py

│   ├── kubik.py

│   └── trapesium.py

└── skrip-utama.py

Ingat, setiap direktori yang akan kita jadikan sebagai paket, harus memiliki file \_\_init\_\_.py. Apa isi file tersebut? Kita bisa kosongkan saja, tidak perlu diisi apa pun.

Untuk masing-masing file, silakan buat fungsi luas\_<nama-bangun-datar> dan volume\_<bangun-ruang>. Kita bisa kosongkan saja isi fungsinya dengan menggunakan [statemen pass](https://jagongoding.com/python/dasar/pass).

1.  File luas/lingkaran.py

2.  def luas\_lingkaran():

3.    pass

4.  File luas/persegi.py:

5.  def luas\_persegi():

6.    pass

7.  File luas/segitiga.py:

8.  def luas\_segitiga():

9.    pass

10. File volume/bola.py:

11. def volume\_bola():

12.   pass

13. File volume/kubik.py:

14. def volume\_kubik():

15.   pass

16. File volume/trapesium.py:

17. def volume\_trapesium():

18.   pass

## 2.4 Mengimpor Paket

Kita bisa mengimpor paket dengan cara yang sama seperti mengimpor modul biasa. Kita hanya perlu menambahkan <nama-direktori> plus dot (.) untuk pemisahnya.

Perhatikan contoh berikut:

from luas.segitiga import luas\_segitiga

from luas import lingkaran, persegi

from volume.kubik import volume\_kubik

import volume.bola

from volume.trapesium import \*

*# panggil fungsi masing-masing*

NB: paket juga bisa bersifat nested alias kita bisa membuat paket yang ada di dalam paket lain dan seterusnya.

## 2.5 Impor Paket/Modul dengan Alias

Dengan semakin banyaknya modul dan paket, tidak menutup kemungkinan akan adanya item-item dengan nama yang sama.

Misalkan pada modul lingkaran terdapat fungsi dengan nama hitung\_luas(), dan di modul persegi juga ada fungsi dengan nama yang sama.

Bagaimana cara memanggilnya?

Kita bisa memanfaatkan fitur alias dengan statemen as.

Perhatikan contoh berikut:

File lingkaran.py:

def luas (radius):

  return 22 / 7 \* radius \* radius

File persegi.py:

def luas (sisi):

  return sisi \* sisi

File skrip-utama.py:

from persegi import luas as luas\_persegi

from lingkaran import luas as luas\_lingkaran

print(luas\_persegi(10))

print(luas\_lingkaran(21))

Output:

100

1386.0

## 2.6 Impor Paket/Modul secara Kustom

Selain menggunakan alias, kita juga bisa menggunakan [percabangan if else](https://jagongoding.com/python/dasar/percabangan).

Perhatikan contoh berikut:

bidang = 'lingkaran'

if bidang == 'lingkaran':

  from lingkaran import luas

elif bidang == 'persegi':

  from persegi import luas

# hitung luas

print(luas(20))

## 2.7 Menampilkan Path/Nama File dari Modul

Selain itu, kita juga bisa menampilkan lokasi file dari modul yang kita import.

Perhatikan contoh berikut, kita akan coba mengetahui di mana lokasi dari modul lingkaran yang telah kita buat.

import lingkaran

print(lingkaran.\_\_file\_\_)

Output (tentunya berbeda-beda tergantung lokasi file)

Ya mungkin meskipun informasi seperti ini tidak terlalu penting untuk aplikasi kita, tapi bisa cukup penting untuk debugging.

## 2.8 Beberapa Paket dan Modul Bawaan Python

Selain kita bisa membuat modul dan paket sendiri, sebenarnya python pun secara bawaan juga memiliki banyak sekali modul default yang akan memudahkan tugas-tugas kita sebagai developer.

Contohnya seperti modul datetime, math, time, json, dan lain sebagainya.

Cara memanggilnya pun sama saja:

import math

import datetime

print('Tanggal dan waktu sekarang: ', datetime.datetime.now())

print('Nilai konstanta pi: ', math.pi)

Kode program di atas akan menghasilkan output sebagai berikut:

Tanggal dan waktu sekarang: 2021-02-19 13:32:55.622208

Nilai konstanta pi: 3.141592653589793

## 2.9 Poin-Poin Penting

Beberapa poin penting yang bisa kita simpulkan dari pertemuan kali ini:

1. Pendekatan pemrograman modular adalah satu konsep di mana kita memecah bagian besar program menjadi bagian-bagian kecil yang lebih bermakna.
2. Penerapan modular programming membuat kode program lebih terstruktur dan mudah diorganisir.
3. Modul pada python adalah sebuah file berekstensi .py yang berisi kode program python.
4. Modul bisa kita panggil dari file yang lain.
5. Paket adalah sebuah direktori yang memiliki satu file \_\_init\_\_.py dan di dalamnya berisi modul-modul python.
6. Python memiliki modul dan paket-paket secara default.

**Kode Program Lengkap**

Untuk kalian yang ingin mendapatkan kode program lengkap, silakan langsung mengunjungi repository [python-menengah](https://github.com/jagongoding-com/python-menengah).

# PERTEMUAN 3

## CONTOH PROGRAM MENGHITUNG BANGUN RUANG MENGGUNAKAN BAHASA PYTHON

### 3.1.BALOK

# Mengambil input untuk  anjang, lebar, dan tinggi balok

 anjang = float(input(“Masukkan  anjang balok: “))

lebar = float(input(“Masukkan lebar balok: “))

tinggi = float(input(“Masukkan tinggi balok: “))

# Menghitung luas permukaan balok

luas = 2 \* ( anjang \* lebar +  anjang \* tinggi + lebar \* tinggi)

# Menghitung volume balok

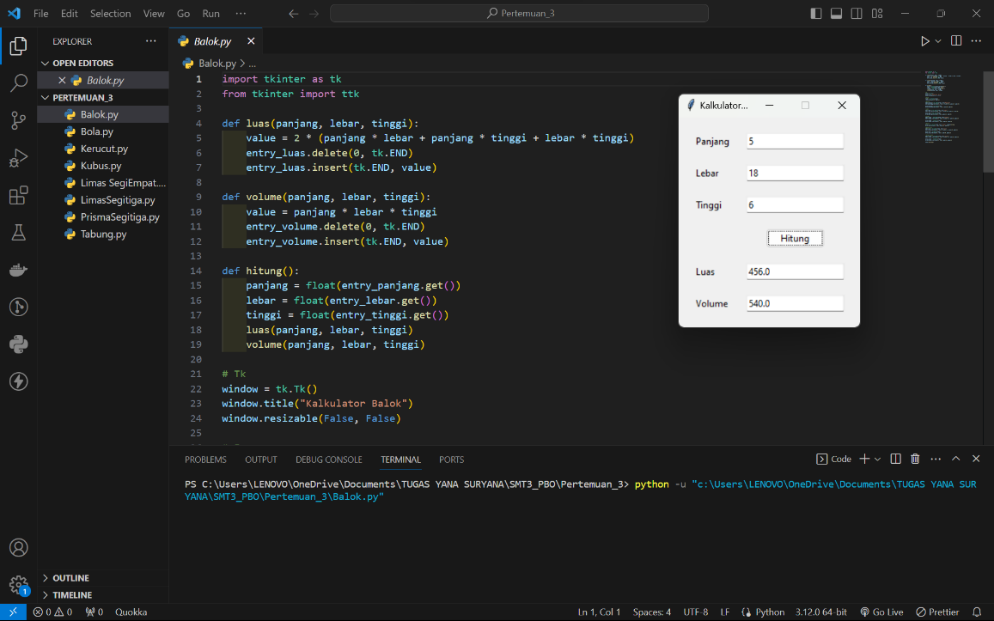
volume =  Panjang \* lebar \* tinggi

# Mencetak luas permukaan dan volume

print(f”Luas permukaan balok : {luas}”)

print(f”Volume balok : {volume}”)

**BERIKUT HASIL PROGRAM MENGHITUNG BANGUN RUANG BALOK DENGAN RUMUS**

****

### 3.2. BOLA

import math

# Input jari-jari dari pengguna

jari\_jari = float(input(“Masukkan jari-jari bola: “))

# Menghitung luas permukaan bola

luas = 4 \* math.pi \* jari\_jari \*\* 2

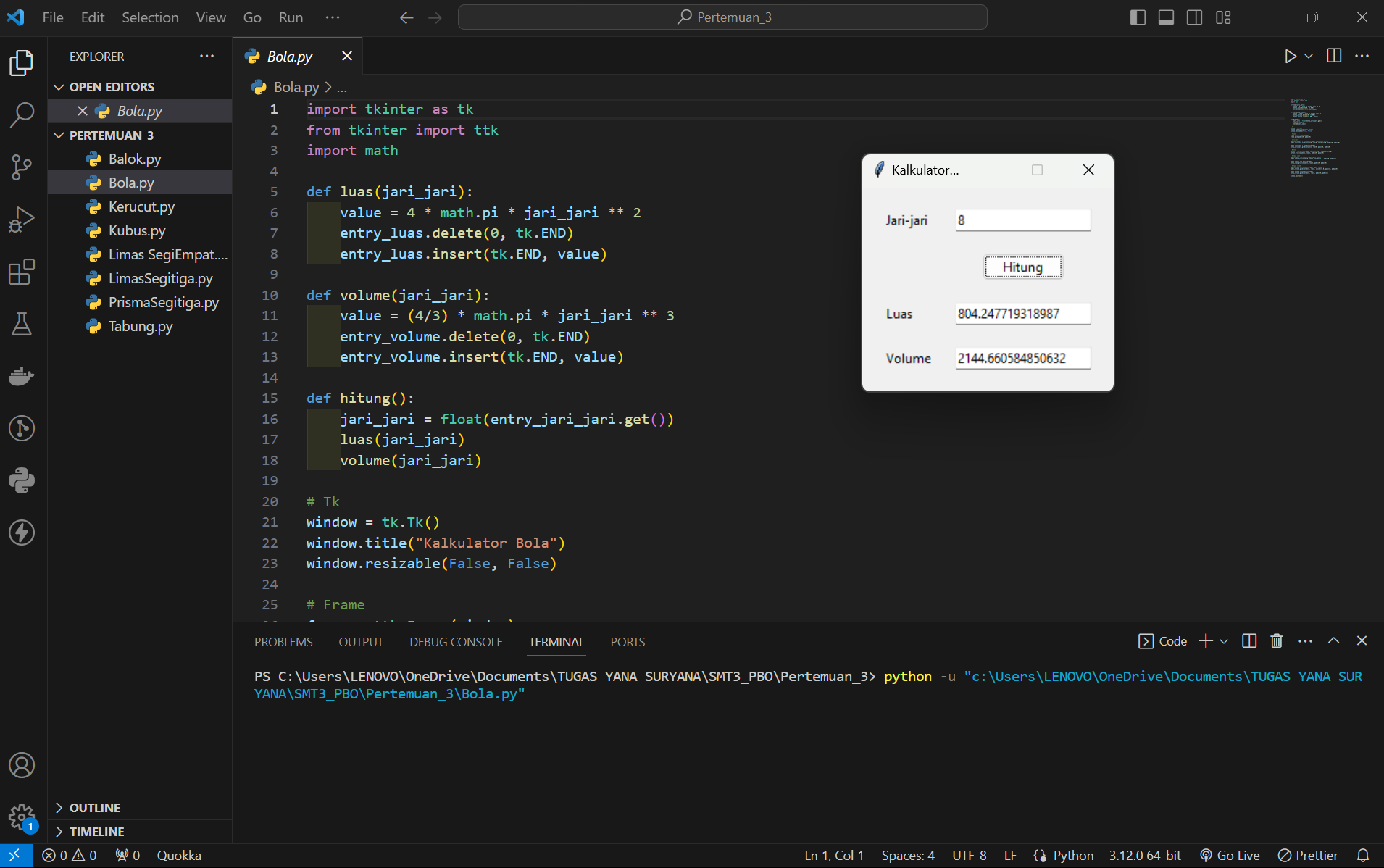
# Menghitung volume bola

volume = (4/3) \* math.pi \* jari\_jari \*\* 3

# Tampilkan hasil

print(f”Luas permukaan bola adalah: {luas:.2f}”)

print(f”Volume bola adalah: {volume:.2f}”)

BERIKUT HASIL PROGRAM MENGHITUNG BANGUN RUANG BOLA DENGAN RUMUS

### 3.3. KERUCUT

import math

# Masukkan jari-jari dan tinggi kerucut

jari\_jari = float(input(“Masukkan jari-jari kerucut: “))

tinggi = float(input(“Masukkan tinggi kerucut: “))

# Menghitung luas permukaan kerucut

luas = math.pi \* jari\_jari \* (jari\_jari + math.sqrt(jari\_jari\*\*2 + tinggi\*\*2))

# Menghitung volume kerucut

volume = (1/3) \* math.pi \* jari\_jari\*\*2 \* tinggi

# Cetak luas dan volume kerucut

print(f”Luas kerucut adalah: {luas:.2f} satuan luas”)

print(f”Volume kerucut adalah: {volume:.2f} satuan volume”)

BERIKUT HASIL PROGRAM MENGHITUNG BANGUN RUANG KERUCUT DENGAN RUMUS

### 3.4. KUBUS

# Mengambil input  anjang sisi kubus

sisi = int(input(“Masukkan  anjang sisi kubus: “))

# Menghitung luas permukaan kubus

luas = 6 \* sisi\*\*2

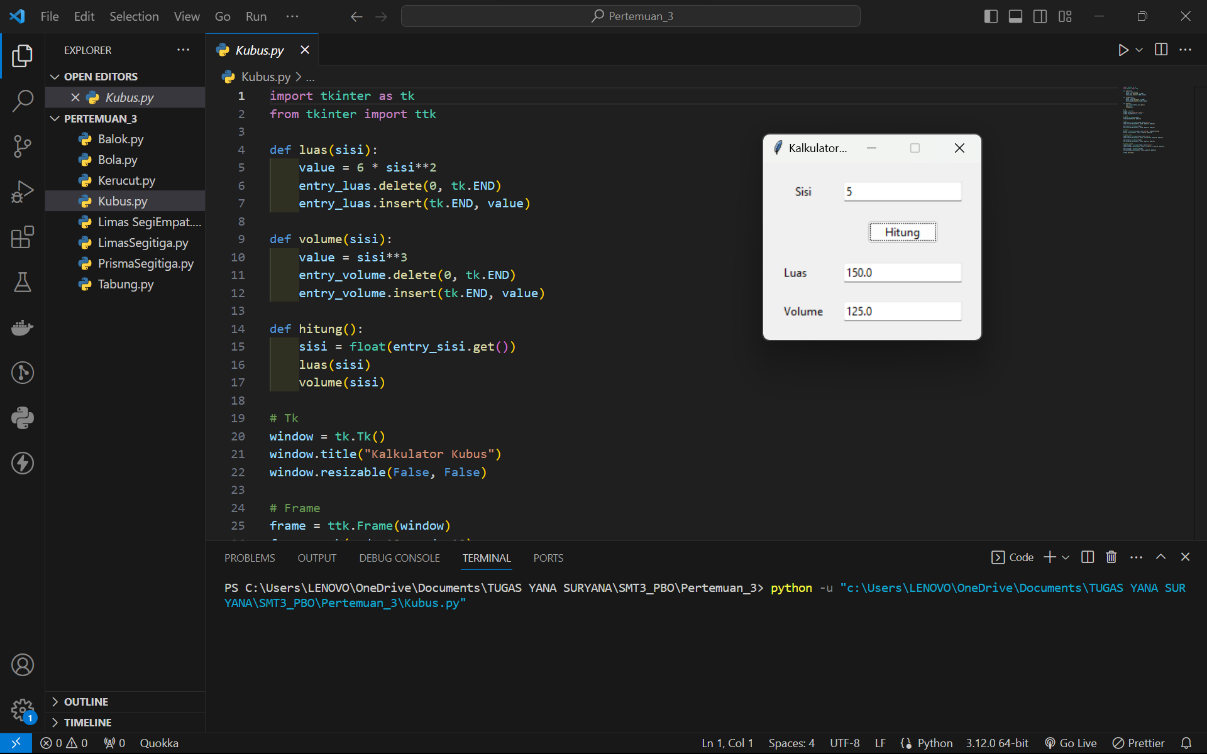
# Menghitung volume kubus

volume = sisi\*\*3

# Mencetak luas permukaan dan volume

print(f”Luas permukaan kubus : {luas}”)

print(f”Volume kubus : {volume}”)

BERIKUT HASIL PROGRAM MENGHITUNG BANGUN RUANG KUBUS DENGAN RUMUS

### 3.5. LIMAS SEGIEMPAT

# Mengambil input untuk  anjang, lebar, tinggi, dan tinggi limas segiempat

 anjang = float(input(“Masukkan  anjang alas limas segiempat: “))

lebar = float(input(“Masukkan lebar alas limas segiempat: “))

tinggi\_limas = float(input(“Masukkan tinggi limas segiempat: “))

tinggi = float(input(“Masukkan tinggi segitiga alas limas: “))

# Menghitung luas permukaan

luas\_permukaan =  anjang \* lebar + 2 \* ( anjang \* tinggi\_limas / 2) + 2 \* (lebar \* tinggi\_limas / 2)

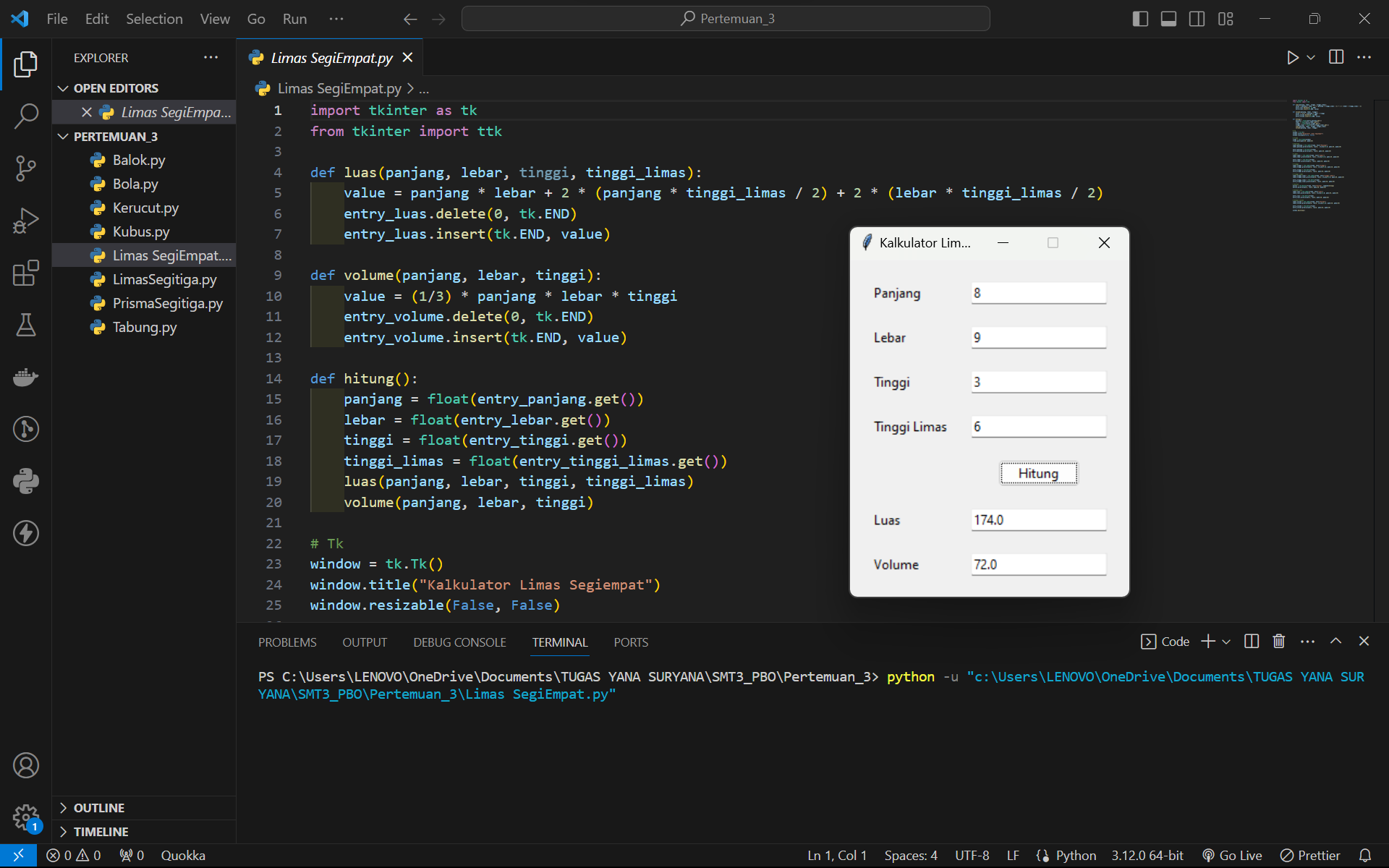
# Menghitung volume

volume = (1/3) \*  anjang \* lebar \* tinggi

# Mencetak luas permukaan dan volume

print(f”Luas permukaan limas segiempat : {luas\_permukaan}”)

print(f”Volume limas segiempat : {volume}”)

BERIKUT HASIL PROGRAM MENGHITUNG BANGUN RUANG LIMAS SEGGIEMPAT DENGAN RUMUS

### 3.6. LIMAS SEGITIGA

# Mengambil input untuk  anjang alas dan tinggi segitiga

 anjang\_alas = float(input(“Masukkan  anjang alas segitiga: “))

tinggi\_segitiga = float(input(“Masukkan tinggi segitiga: “))

tinggi\_limas = float(input(“Masukkan tinggi limas: “))

# Menghitung luas permukaan

luas\_permukaan = ( anjang\_alas \* tinggi\_segitiga) + (3 \* (1 / 2) \*  anjang\_alas \* tinggi\_limas)

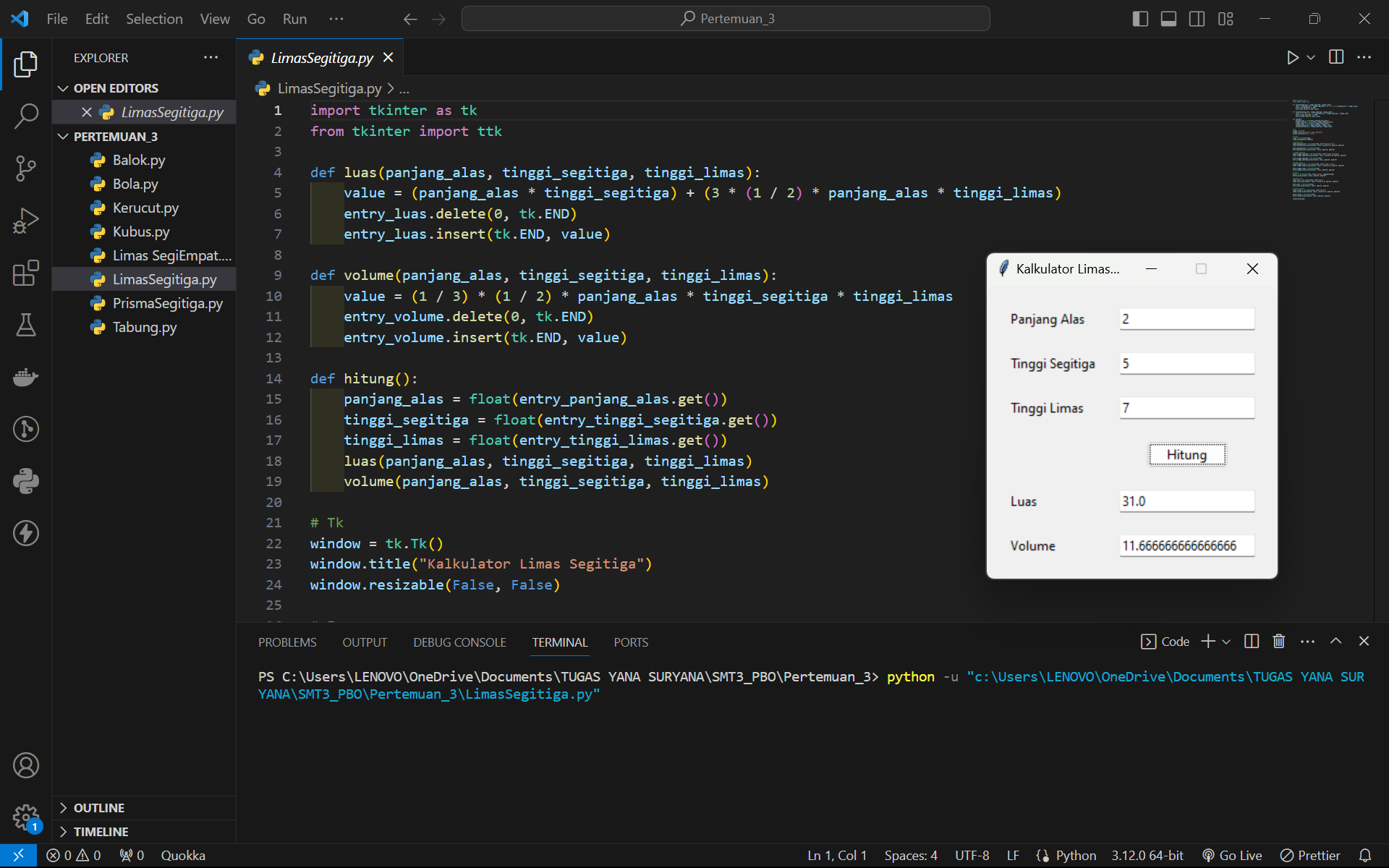
# Menghitung volume

volume = (1 / 3) \* (1 / 2) \*  anjang\_alas \* tinggi\_segitiga \* tinggi\_limas

# Mencetak luas permukaan dan volume limas segitiga

print(f”Luas permukaan limas segitiga : {luas\_permukaan}”)

print(f”Volume limas segitiga : {volume}”)

BERIKUT HASIL PROGRAM MENGHITUNG BANGUN RUANG LIMAS SEGITIGA DENGAN RUMUS

### 3.7 .PRISMA SEGITIGA

# Mengambil input untuk  anjang alas, tinggi segitiga, dan tinggi prisma

alas\_segitiga = float(input(“Masukkan  anjang alas segitiga: “))

tinggi\_segitiga = float(input(“Masukkan tinggi segitiga: “))

tinggi\_prisma = float(input(“Masukkan tinggi prisma: “))

# Menghitung luas permukaan

luas\_segitiga = 0.5 \* alas\_segitiga \* tinggi\_segitiga

luas\_permukaan\_prisma = 2 \* alas\_segitiga \* tinggi\_segitiga + 3 \* alas\_segitiga \* tinggi\_prisma

# Menghitung volume

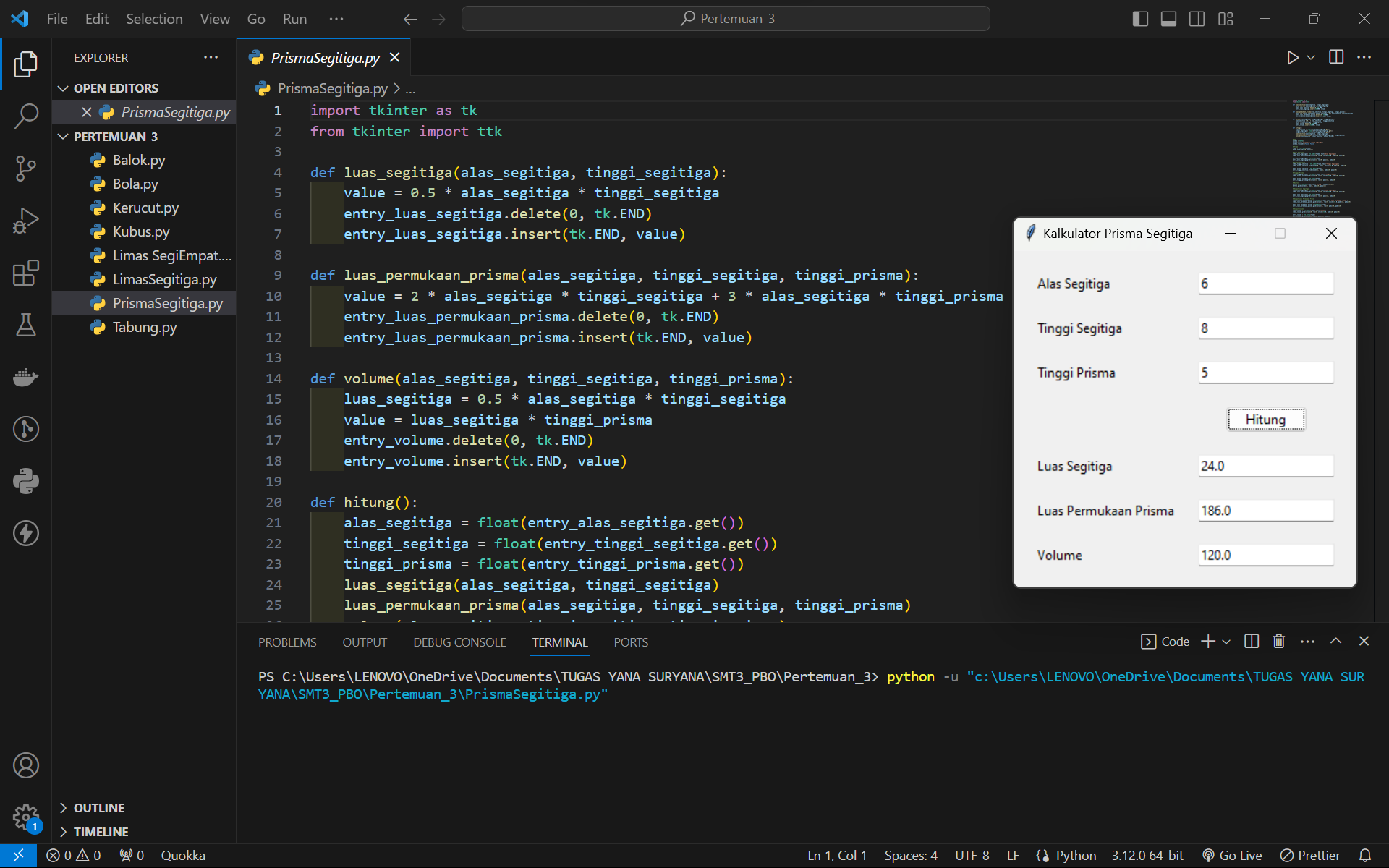
volume\_prisma = luas\_segitiga \* tinggi\_prisma

# Mencetak luas permukaan dan volume

print(f”Luas permukaan prisma segitiga : {luas\_permukaan\_prisma}”)

print(f”Volume prisma segitiga : {volume\_prisma}”)

BERIKUT HASIL PROGRAM MENGHITUNG BANGUN RUANG PRISMA SEGITIGA DENGAN RUMUS



### 3.8 .SELINDER(TABUNG)

import math

# Input jari-jari dan tinggi dari tabung

jari\_jari = float(input(“Masukkan jari-jari tabung: “))

tinggi = float(input(“Masukkan tinggi tabung: “))

# Hitung luas permukaan tabung

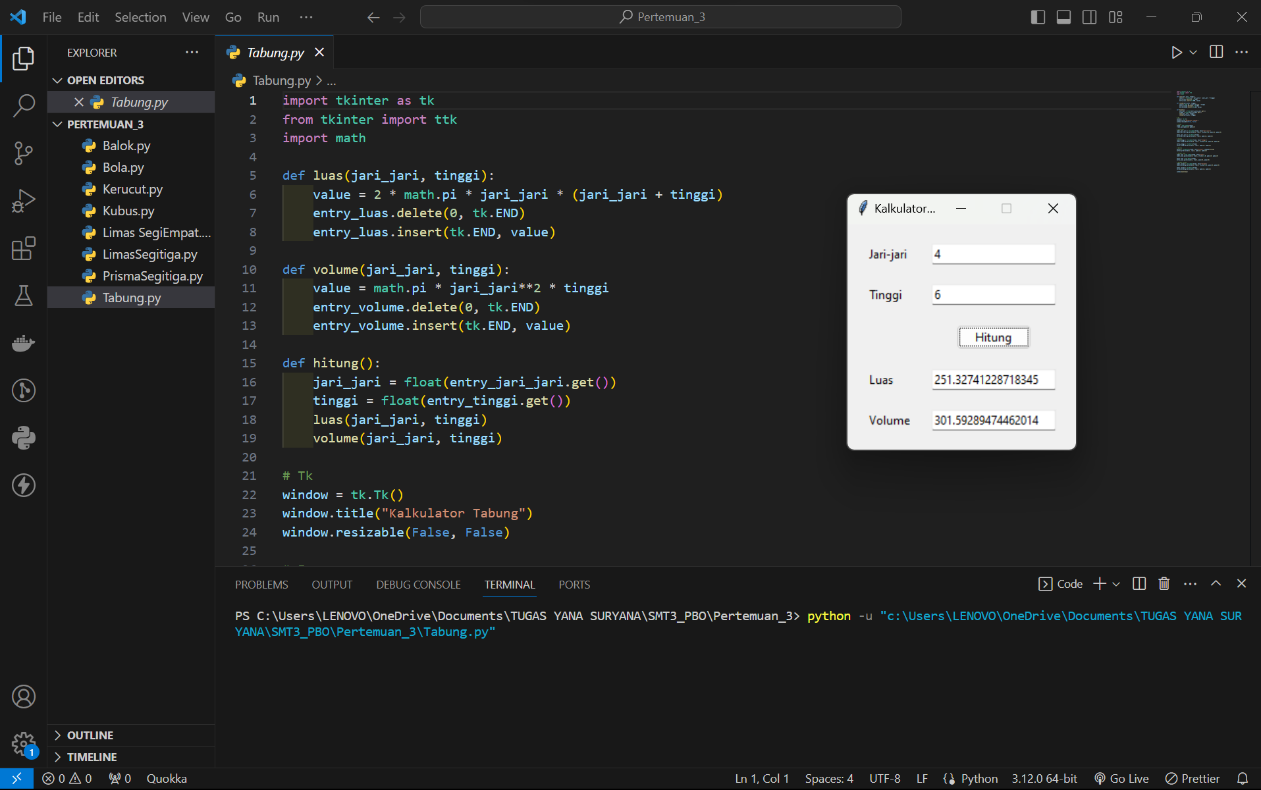
luas\_permukaan = 2 \* math.pi \* jari\_jari \* (jari\_jari + tinggi)

# Hitung volume tabung

volume = math.pi \* jari\_jari\*\*2 \* tinggi

print(“Luas permukaan tabung:”, round(luas\_permukaan, 2))

print(“Volume tabung:”, round(volume, 2))

**BERIKUT HASIL PROGRAM MENGHITUNG BANGUN RUANG SELINDER(TABUNG) DENGAN RUMUS**

# PERTEMUAN 4

## 4.1 CARA MEMBUAT JADWAL MATAKULIAH MENGGUNAKAN BAHASA PYTHON

Beberapa modul Python yang dapat digunakan untuk membuat jadwal pelajaran antara lain:

1. Datetime : Modul ini dapat digunakan untuk mengelola tanggal dan waktu dalam Python. Anda dapat menggunakan modul ini untuk menentukan waktu dan tanggal untuk jadwal pelajaran.
2. calendar : Modul ini menyediakan fungsi untuk membuat kalender dalam Python. Anda dapat menggunakan modul ini untuk menampilkan jadwal pelajaran dalam bentuk kalender.
3. tkinte : Modul ini adalah toolkit GUI standar untuk Python. Anda dapat menggunakan modul ini untuk membuat antarmuka pengguna untuk program jadwal pelajaran.
4. Pandas : Modul ini adalah library Python untuk analisis data. Anda dapat menggunakan modul ini untuk menyimpan dan mengelola jadwal pelajaran dalam bentuk tabel.

Dengan memanfaatkan modul-modul tersebut, Anda dapat membuat program Python yang dapat digunakan untuk menyusun dan menampilkan jadwal pelajaran dengan mudah.

**Berikut adalah contoh program jadwal mata kuliah**

import tkinter as tk

from tkinter import ttk

def tambah\_jadwal():

    mata\_kuliah = combo\_mata\_kuliah.get()

    ruang = entry\_ruang.get()

    gedung = combo\_gedung.get()

    hari = combo\_hari.get()

    jam\_mulai = entry\_jam\_mulai.get()

    jam\_selesai = entry\_jam\_selesai.get()

    dosen = entry\_dosen.get()

    # Menambahkan data ke dalam treeview

    tree.insert("", "end", values=(mata\_kuliah, ruang, gedung, hari, jam\_mulai, jam\_selesai, dosen))

    # Mengosongkan input setelah menambahkan data

    entry\_ruang.delete(0, "end")

    entry\_jam\_mulai.delete(0, "end")

    entry\_jam\_selesai.delete(0, "end")

    entry\_dosen.delete(0, "end")

# Membuat window

window = tk.Tk()

window.title("Jadwal Kuliah ZEIN")

# Menambahkan warna latar belakang pada jendela

window.configure(background="VIOLET")

# Label Nama

Nama = tk.Label(window, text="JADWAL KULIAH     ZEIN ABDILLAH PRATAMA (220511020)", bg='yellow')

# Membuat label dan entry untuk setiap kolom

label\_mata\_kuliah = tk.Label(window, text="Mata Kuliah  :", bg='YELLOW')

mata\_kuliah\_options = ["Pemograman 2 (PBO)", "Arsitektur dan Organisasi Komputer", "AIK 2 (Ibadah, Akhlak dan Muamalah)", "Komunikasi Data", "Struktur Data dan Algoritma", "Statistik dan Probabilitas", "Kalkulus 2", "Sistem Informasi (APSI)"]

combo\_mata\_kuliah = ttk.Combobox(window, values=mata\_kuliah\_options)

label\_ruang = tk.Label(window, text="Ruang  :", bg='YELLOW')

entry\_ruang = tk.Entry(window)

label\_gedung = tk.Label(window, text="Gedung    :", bg='YELLOW')

gedung\_options = ["Ir.H.Djuanda", "Machdor"]

combo\_gedung = ttk.Combobox(window, values=gedung\_options)

label\_hari = tk.Label(window, text="Hari    :", bg='YELLOW')

hari\_options = ["Senin", "Selasa", "Rabu", "Kamis", "Jumat", "Sabtu", "Minggu"]

combo\_hari = ttk.Combobox(window, values=hari\_options)

label\_jam\_mulai = tk.Label(window, text="Jam Mulai  :", bg='YELLOW')

entry\_jam\_mulai = tk.Entry(window)

label\_jam\_selesai = tk.Label(window, text="Jam Selesai  :", bg='YELLOW')

entry\_jam\_selesai = tk.Entry(window)

label\_dosen = tk.Label(window, text="Dosen  :", bg='YELLOW')

entry\_dosen = tk.Entry(window)

# Membuat tombol untuk menambahkan jadwal

tombol\_tambah = tk.Button(window, text="Tambah Jadwal", command=tambah\_jadwal, bg='light blue')

# Membuat Treeview untuk menampilkan jadwal

columns = ("Mata Kuliah", "Ruang", "Gedung", "Hari", "Jam Mulai", "Jam Selesai", "Dosen")

tree = ttk.Treeview(window, columns=columns, show="headings")

# Menambahkan heading untuk setiap kolom

for col in columns:

    tree.heading(col, text=col)

    if col == "Mata Kuliah":

        tree.column(col, width=230, stretch=tk.NO)

    else:

        tree.column(col, width=150)

# Menempatkan elemen-elemen pada grid

Nama.grid(row=0, column=1, padx=5, pady=5, sticky="ew")

label\_mata\_kuliah.grid(row=1, column=0, padx=10, pady=5, sticky="ew")

combo\_mata\_kuliah.grid(row=1, column=2, padx=10, pady=5, sticky="ew")

label\_ruang.grid(row=2, column=0, padx=10, pady=5, sticky="ew")

entry\_ruang.grid(row=2, column=2, padx=10, pady=5, sticky="ew")

label\_gedung.grid(row=3, column=0, padx=10, pady=5, sticky="ew")

combo\_gedung.grid(row=3, column=2, padx=10, pady=5, sticky="ew")

label\_hari.grid(row=4, column=0, padx=10, pady=5, sticky="ew")

combo\_hari.grid(row=4, column=2, padx=10, pady=5, sticky="ew")

label\_jam\_mulai.grid(row=5, column=0, padx=10, pady=5, sticky="ew")

entry\_jam\_mulai.grid(row=5, column=2, padx=10, pady=5, sticky="ew")

label\_jam\_selesai.grid(row=6, column=0, padx=10, pady=5, sticky="ew")

entry\_jam\_selesai.grid(row=6, column=2, padx=10, pady=5, sticky="ew")

label\_dosen.grid(row=7, column=0, padx=10, pady=5, sticky="ew")

entry\_dosen.grid(row=7, column=2, padx=10, pady=5, sticky="ew")

tombol\_tambah.grid(row=8, column=1, columnspan=1, pady=10)

tree.grid(row=9, column=0, columnspan=3, padx=10, pady=5, sticky="nsew")

# Mengatur proporsi dari kolom dan baris agar dapat melebar dan mengisi window

window.columnconfigure(0, weight=1)

window.columnconfigure(1, weight=1)

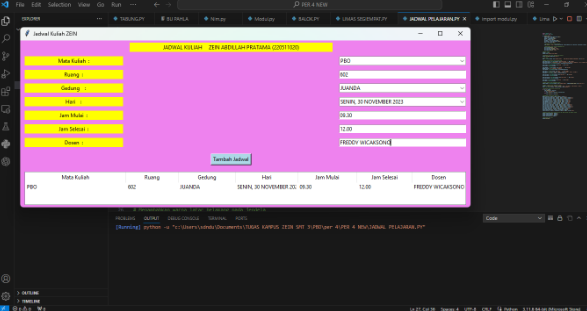
window.columnconfigure(2, weight=1)

window.rowconfigure(9, weight=1)

# Menjalankan main loop

window.mainloop()

Output :



**Berikut ini adalah contoh program ke 2 yaitu open win**

**Openwin.py**

import tkinter as tk

from tkinter import Entry,Button,END, filedialog, Text, LEFT, RIGHT, X

def openfile():

    tf=filedialog.askopenfilename(

        initialdir="C:/Users/Lenovo/Documents/KULIAH SEMESTER 3/PBO/Pertemuan 4",

        title="Open Text file",

        filetypes=(("Text File", "\*.txt"),)

        )

    pathh.insert(END, tf)

    tf=open(tf) #or tf=open(tf, 'r')

    data=tf.read()

    txtarea.insert(END, data)

    tf.close()

def savefile():

    teks=pathh.get()

    with open(pathh.get(),"w") as file1:

        file1.write(txtarea.get(1.0, "end-1c"))

        txtarea.delete(1.0, "end-1c")

        file1.close()

ws=tk.Tk()

ws.title("PythonGuides")

ws.geometry("400x450")

ws['bg']='#fb0'

txtarea=Text(ws, width=40, height=20)

txtarea.pack(pady=20)

pathh= Entry(ws)

pathh.pack(side=LEFT, expand=True, fill=X, padx=20)

Button(

    ws,

    text="Open File",

    command=openfile

    ).pack(side=RIGHT, expand=True, fill=X, padx=20)

Button(

    ws,

    text="Save File",

    command=savefile

    ).pack(side=RIGHT, expand=True, fill=X, padx=20)

ws.mainloop()

**modul.py**

import math

# Fungsi untuk menghitung luas permukaan dan volume kubus

def luas\_kubus(sisi):

    luas\_permukaan1 = 6 \* (sisi \*\* 2)

    return luas\_permukaan1

def volume\_kubus (sisi):

    volume1 = sisi \*\* 3

    return volume1

# Fungsi untuk menghitung luas permukaan dan volume balok

def luas\_balok(panjang, lebar, tinggi):

    luas\_permukaan2 = 2 \* (panjang \* lebar + panjang \* tinggi + lebar \* tinggi)

    return luas\_permukaan2

def volume\_balok (panjang, lebar, tinggi):

    volume2 = panjang \* lebar \* tinggi

    return volume2

# Fungsi untuk menghitung luas permukaan dan volume limas segiempat

def luas\_limasSegiempat (panjang, lebar, tinggi):

    luas\_permukaan3 = panjang \* lebar + panjang \* math.sqrt((lebar/2)\*\*2 + tinggi\*\*2) + lebar \* math.sqrt((panjang/2)\*\*2 + tinggi\*\*2)

    return luas\_permukaan3

def volume\_limasSegiempat (panjang, lebar, tinggi):

    volume3 = (panjang \* lebar \* tinggi) / 3

    return volume3

# Fungsi untuk menghitung luas permukaan dan volume limas prismaSegitiga

def luas\_prismasegitiga (panjang\_alas\_segitiga, tinggi\_segitiga, tinggi\_prisma) :

    luas\_permukaan4 = (2 \* panjang\_alas\_segitiga \* tinggi\_segitiga + (panjang\_alas\_segitiga \* tinggi\_prisma))

    return luas\_permukaan4

def volume\_prismasegitiga (panjang\_alas\_segitiga, tinggi\_segitiga, tinggi\_prisma) :

    volume4 = (panjang\_alas\_segitiga \* tinggi\_segitiga\* tinggi\_prisma) / 2

    return volume4

# Fungsi untuk menghitung luas permukaan dan volume limassegitiga

def luas\_limassegitiga (panjang\_alas\_segitiga, tinggi\_segitiga) :

    luas\_permukaan5 = panjang\_alas\_segitiga \*\* 2 + 3 \* (panjang\_alas\_segitiga \* math.sqrt((panjang\_alas\_segitiga / 2) \*\* 2 + tinggi\_segitiga \*\* 2))

    return luas\_permukaan5

def volume\_limassegitiga (panjang\_alas\_segitiga, tinggi\_limas) :

    volume5 = (panjang\_alas\_segitiga \*\* 2 \* tinggi\_limas) / 6

    return volume5

# Fungsi untuk menghitung luas permukaan dan volume tabung

def luas\_tabung (jari\_jari, tinggi) :

    luas\_permukaan6 = 2 \* math.pi \* jari\_jari \* (jari\_jari + tinggi)

    return luas\_permukaan6

def volume\_tabung (jari\_jari, tinggi) :

    volume6 = math.pi \* jari\_jari\*\*2 \* tinggi

    return volume6

# Fungsi untuk menghitung luas permukaan dan volume tabung

def luas\_kerucut (tinggi, jari\_jari) :

    luas\_permukaan7 = 3.14 \* jari\_jari \* (jari\_jari + ((tinggi \*\* 2 + jari\_jari \*\* 2) \*\* 0.5))

    return luas\_permukaan7

def volume\_kerucut (tinggi, jari\_jari) :

    volume7 = (1/3) \* 3.14 \* jari\_jari\*\*2 \* tinggi

    return volume7

def luas\_bola (jari\_jari) :

    luas\_permukaan8 = 4 \* math.pi \* jari\_jari \*\* 2

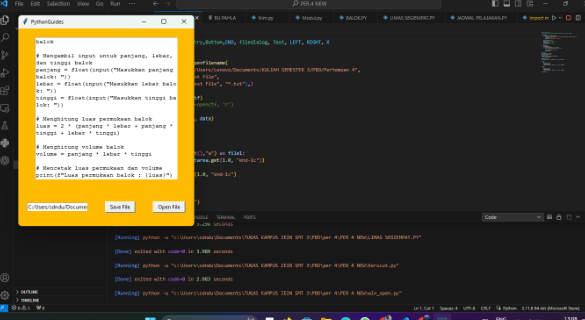
    return luas\_permukaan8

def volume\_bola (jari\_jari) :

    volume8 = (4/3) \* math.pi \* jari\_jari \*\* 3

    return volume8

**output :**

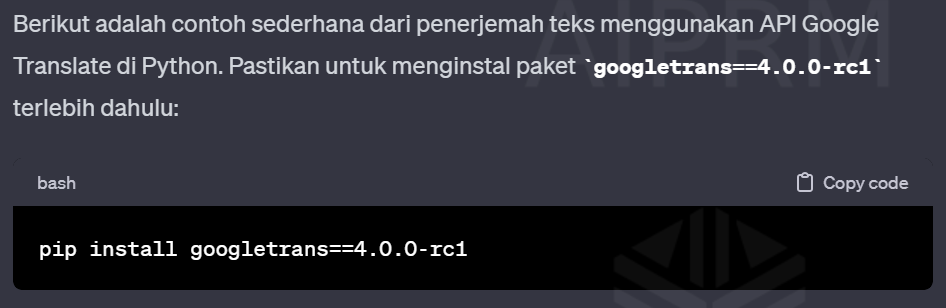
****

# PERTEMUAN 6

## 5.1 MULTIFORM TRANSLATOR

Multipart/form-data adalah tipe data yang digunakan dalam permintaan HTTP untuk mengirim data berupa file atau data berupa nama-nilai. Untuk mengirim permintaan POST dengan tipe data multipart/form-data menggunakan Python, Anda dapat menggunakan beberapa library yang tersedia. Berikut adalah beberapa library yang dapat Anda gunakan:

* 1. multipartify: Ini adalah sebuah library yang memungkinkan Anda untuk mengubah dictionary Python menjadi format multipart/form-data yang dapat digunakan dalam permintaan POST menggunakan library `requests`.
  2. streaming-form-data: Ini adalah sebuah library yang menyediakan parser untuk mengolah input data multipart/form-data dalam bentuk urutan.
  3. python-multipart: Ini adalah sebuah library Apache2-licensed yang menyediakan parser streaming untuk mengolah data multipart/form-data dalam Python



Berikut adalah contoh program translator

from tkinter import Frame, Label, Entry, Button, YES, BOTH, END, Tk, W

from googletrans import Translator

class Translate:

    def \_\_init\_\_(self, parent, title):

        self.parent = parent

        self.parent.geometry("600x200")

        self.parent.title(title)

        self.parent.protocol("WM\_DELETE\_WINDOW", self.onKeluar)

        self.aturKomponen()

    def aturKomponen(self):

        mainFrame = Frame(self.parent, bd=10)

        mainFrame.pack(fill=BOTH, expand=YES)

        # pasang Label

        Label(mainFrame, text='Masukkan teks:').grid(row=0, column=0,

            sticky=W, padx=5, pady=5)

        Label(mainFrame, text='Hasil Terjemahan:').grid(row=2, column=0,

            sticky=W, padx=5, pady=5)

        # pasang textbox

        self.txtSumber = Entry(mainFrame, width=50)

        self.txtSumber.grid(row=0, column=1, columnspan=3, padx=5, pady=5)

        self.txtHasil = Entry(mainFrame, width=50)

        self.txtHasil.grid(row=2, column=1, columnspan=3, padx=5, pady=5)

        # Pasang Button

        self.btnTranslateEn = Button(mainFrame, text='Translate to English',

            command=lambda: self.onTranslate('en'))

        self.btnTranslateEn.grid(row=1, column=0, padx=5, pady=5)

        self.btnTranslateJw = Button(mainFrame, text='Translate to Javanese',

            command=lambda: self.onTranslate('jw'))

        self.btnTranslateJw.grid(row=1, column=1, padx=5, pady=5)

        self.btnTranslateSu = Button(mainFrame, text='Translate to Ukraine',

            command=lambda: self.onTranslate('uk'))

        self.btnTranslateSu.grid(row=1, column=2, padx=5, pady=5)

    def onTranslate(self, dest\_lang):

        # membuat instance object

        penterjemah = Translator()

        # menterjemahkan

        hasil = penterjemah.translate(self.txtSumber.get(), dest=dest\_lang)

        # menghapus isi textbox hasil sebelumnya

        self.txtHasil.delete(0, END)

        # menampilkan hasil terjemahan

        self.txtHasil.insert(END, hasil.text)

    def onKeluar(self, event=None):

        # memberikan perintah menutup aplikasi

        self.parent.destroy()

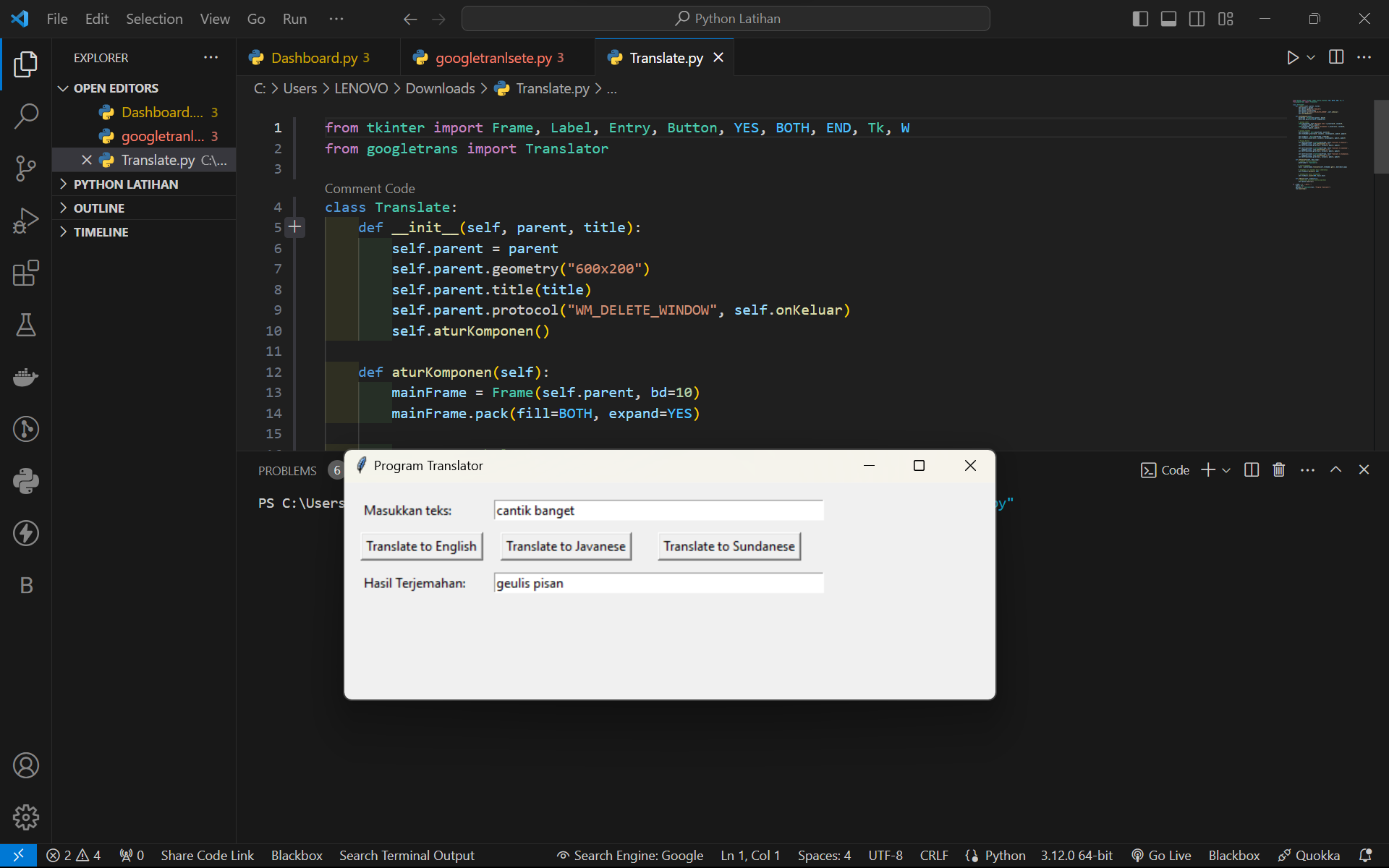
if \_\_name\_\_ == '\_\_main\_\_':

    root = Tk()

    aplikasi = Translate(root, "Program Translator")

    root.mainloop()

**Berikut hasil program translator setelah dijalankan**



# PERTEMUAN 7

## 6.1 APLIKASIN BANTU (TOOLS)

Aplikasi bantu tools pada Python adalah program atau skrip yang dibuat dengan menggunakan bahasa pemrograman Python untuk memberikan bantuan atau mempermudah dalam menjalankan tugas tertentu. Aplikasi ini biasanya dirancang untuk melakukan fungsi-fungsi spesifik atau menyediakan utilitas yang dapat digunakan oleh pengguna untuk tujuan tertentu.

Contoh dari aplikasi bantu tools Python meliputi:

1. Kalkulator Sederhana: Sebuah program yang memungkinkan pengguna untuk melakukan operasi matematika dasar.

2. Aplikasi Konversi Suhu: Sebuah program yang mengonversi suhu dari satu satuan ke satuan lainnya, seperti dari Celsius ke Fahrenheit.

3. Pengelola Tugas: Aplikasi yang membantu pengguna mengelola daftar tugas atau pekerjaan yang harus dilakukan.

4. Pengelola File: Sebuah alat untuk mengorganisir, mengedit, atau memindahkan file dalam sistem file.

5. Aplikasi Analisis Data Sederhana: Sebuah program untuk melakukan analisis sederhana terhadap data, seperti menghitung statistik dasar atau membuat grafik.

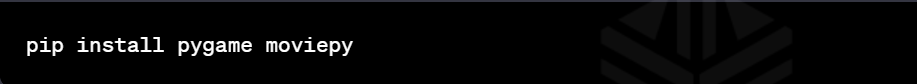
Aplikasi bantu tools sering kali dirancang untuk mempermudah atau mengotomatiskan tugas-tugas sehari-hari, serta memberikan antarmuka yang mudah digunakan bagi pengguna. Python sering digunakan untuk pengembangan aplikasi ini karena sintaksisnya yang mudah dipahami dan banyaknya pustaka (library) yang tersedia, memungkinkan pengembang untuk membuat aplikasi dengan cepat.

CARA MEMUTAR MUSIK MP3 DAN MP4 SERTA MENGUBAH GAMBAR MENJADI TEXT

Anda dapat menggunakan Pustaka *pygame* untuk memutar file mp3*, dan moviepy* untuk

Memutar file mp4.

Pastikan untuk menginstall kedua Pustaka tersebut :



**Berikut program memutar file mp3menggunkan *pygame***

import tkinter as tk

from pygame import mixer

def play\_music():

    # Ganti path file audio dengan path file Anda

    file\_path = r'C:\Users\LENOVO\OneDrive\Documents\Tugas PBO\\LINGSIR WENGI.mp3'

    mixer.init()

    mixer.music.load(file\_path)

    mixer.music.play()

def stop\_music():

    mixer.music.stop()

# Membuat jendela

root = tk.Tk()

root.title("Pemutar Musik")

# Membuat tombol play

play\_button = tk.Button(root, text="Play", command=play\_music)

play\_button.pack(pady=20)

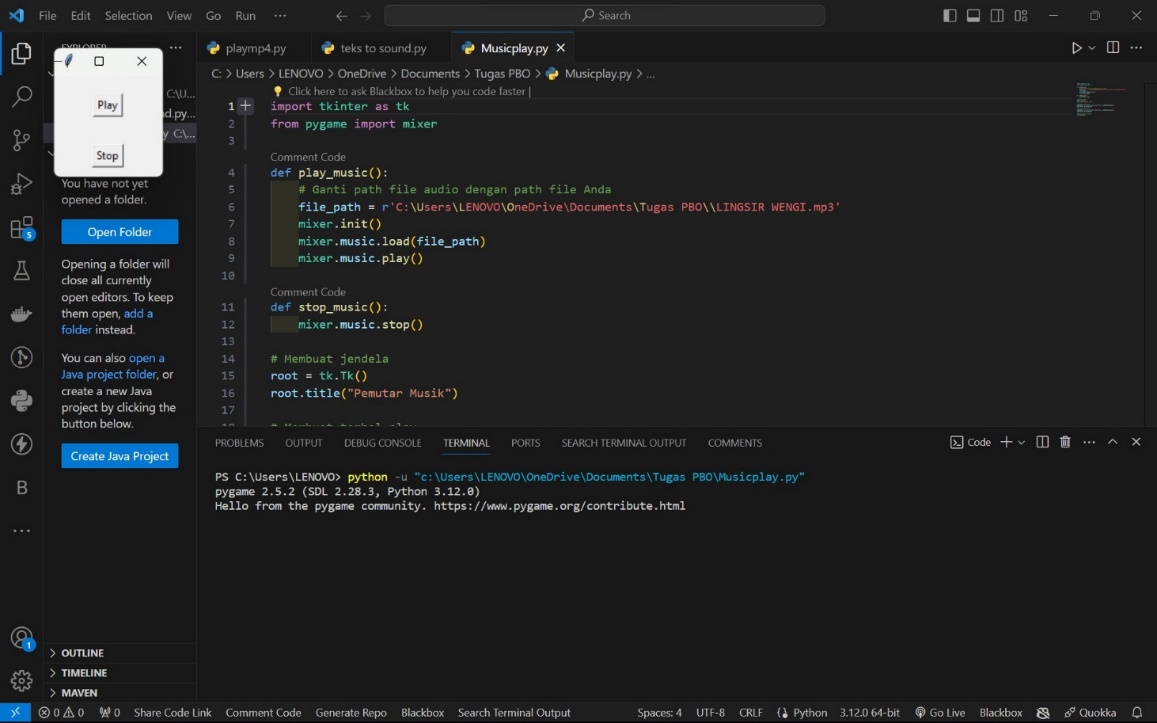
# Membuat tombol stop

stop\_button = tk.Button(root, text="Stop", command=stop\_music)

stop\_button.pack(pady=10)

# Menjalankan loop utama

root.mainloop()

**Berikut Hasil Program Mp3 setelah dijalankan**

Pastikan bahwa anda telah menyediakan file mp3 dan mp4 yang ingin anda putar, dan gantilah *‘nama\_file.mp3’* dan *‘nama\_file.mp4’* dengan nama file yang sesuai.

**Berikut program memutar file mp4menggunkan *moviepy***

# importing libraries

import cv2

import numpy as np

# Create a VideoCapture object and read from input file

cap = cv2.VideoCapture('videos/yanasuryana.mp4')

# Check if camera opened successfully

if (cap.isOpened()== False):

    print("Error opening video file")

# Read until video is completed

while(cap.isOpened()):

# Capture frame-by-frame

    ret, frame = cap.read()

    if ret == True:

    # Display the resulting frame

        cv2.imshow('Frame', frame)

    # Press Q on keyboard to exit

        if cv2.waitKey(25) & 0xFF == ord('q'):

            break

# Break the loop

    else:

        break

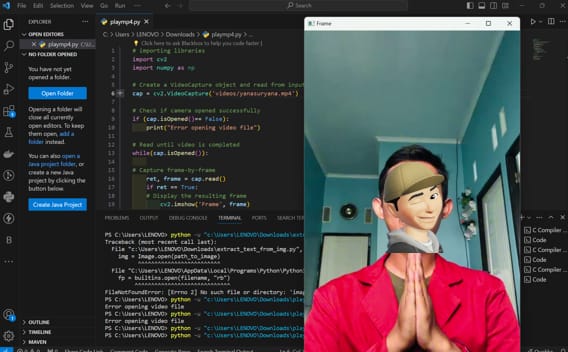
# When everything done, release

# the video capture object

cap.release()

# Closes all the frames

cv2.destroyAllWindows()

Berikut Hasil Program Mp4 Setelah dijalakankan :

Pastikan bahwa anda telah menyediakan file mp3 dan mp4 yang ingin anda putar, dan gantilah *‘nama\_file.mp3’* dan *‘nama\_file.mp4’* dengan nama file yang sesuai.

**Berikut program Mengubah GAMBAR menjadi TEXT**

**Sebelumnya anda harus install *pip install pytesseract***

**Anda juga harus install  *pip install pillow (pil)* untuk bekerja dengan gambar.**

Berikut program menjalankan file pytesseract :

import pytesseract as tess

tess.pytesseract.tesseract\_cmd = (r'C:\Program Files\Tesseract-OCR\tesseract.exe')

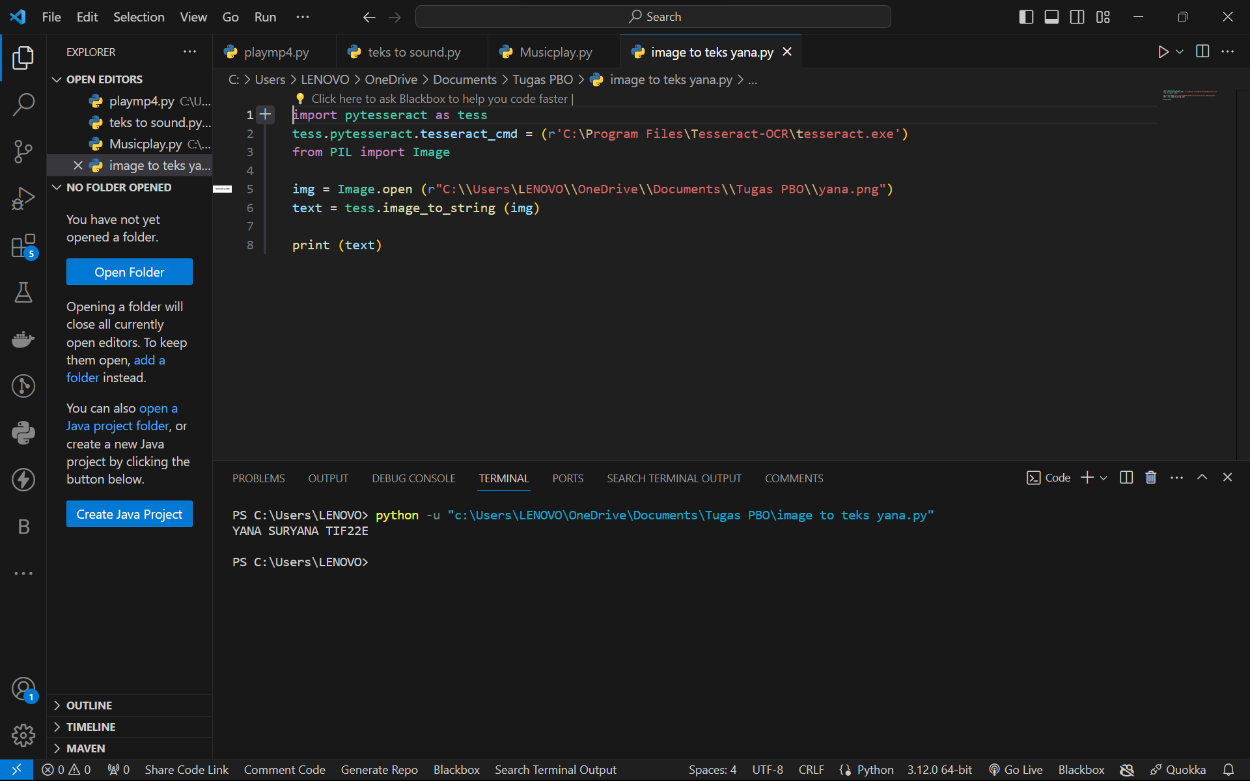
from PIL import Image

img = Image.open (r"C:\\Users\LENOVO\\OneDrive\\Documents\\Tugas PBO\\yana.png")

text = tess.image\_to\_string (img)

print (text)

**Berikut Hasil Programnya Setelah dijalankan :**



**Berikut program Text To Sound**

from gtts import gTTS

import os

# Teks yang ingin diubah menjadi suara

teks = "hallo nama saya YANA SURYANA TIF22E."

# Inisialisasi objek gTTS dengan teks dan bahasa yang diinginkan (dalam contoh ini, bahasa Indonesia)

tts = gTTS(text=teks, lang='id')

# Simpan suara ke dalam file 'output.mp3'

tts.save("output.mp3")

# Putar suara menggunakan aplikasi default di sistem

os.system("output.mp3")

**berikut gambar yang dipake untuk membaca textnya :**



# PERTEMUAN 8

# CLASS DAN OBJEK PADA PYTHON

## 7.1 CLASS

Class merupakan sebuah blueprint untuk membuat objek. Ini adalah entitas yang menyatukan data (variabel) dan fungsi (metode) yang beroperasi pada data tersebut.

Contoh :

from gtts import gTTS

import os

# Teks yang ingin diubah menjadi suara

teks = "hallo nama saya YANA SURYANA TIF22E."

# Inisialisasi objek gTTS dengan teks dan bahasa yang diinginkan (dalam contoh ini, bahasa Indonesia)

tts = gTTS(text=teks, lang='id')

# Simpan suara ke dalam file 'output.mp3'

tts.save("output.mp3")

# Putar suara menggunakan aplikasi default di sistem

os.system("output.mp3")

Dalam contoh di atas, kita mendefinisikan sebuah class Motor. Class ini memiliki dua atribut: merek dan warna, serta sebuah metode info untuk memberikan informasi tentang motor.

## 7.2 OBJEK

Objek adalah instance dari sebuah class. Saat Anda membuat objek, Anda membuat sebuah instance dari class tersebut, yang memungkinkan Anda mengakses semua atribut dan metode yang didefinisikan dalam class.

Contoh :

motor1 = Motor("Suzuki", "Kuning")

motor2 = Motor("Honda", "Hitam")

print(motor1.info())  # Output: Motor Suzuki berwarna Kuning

print(motor2.info())  # Output: Motor Honda berwarna Hitam

Dalam contoh di atas, motor1 dan motor2 adalah dua objek yang diciptakan dari class Motor. Kita menggunakan metode info() untuk mendapatkan informasi tentang setiap motor

## 7.3 PENGGUNAAN CLASS DAN OBJEK

Class sendiri pada Python diartikan sebagai instansi bagi sebuah object. Sebuah class memiliki ciri penulisan diawali dengan class, lalu nama class itu sendiri, mirip seperti cara penulisan function, berikut contohnya:

class hewan():

    pass

Pada contoh di atas, kita membuat sebuah class bernama hewan(). Biasanya, dalam penulisan sebuah class, diikuti dengan object untuk menentukan sifat dari class itu sendiri. Selain object, kita juga dapat memberikan input ke dalamnya.

Berikut contoh sederhana dari sebuah class yang hanya memiliki object dan tanpa input:

class hewan():

    def anjing():

        print ("gukguk")

    def kucing():

        print ("meong")

    def bebek():

        print("kwek kwek")

    def ayam():

        print ("kukuruyuk")

print(hewan.ayam())

# Output kukuruyuk

Pada contoh di atas, kita memanggil class hewan dan kita ingin mengeluarkan output berupa suara hewan, di mana pada contoh tersebut kita menggunakan object yaitu ayam(). Untuk pemanggilan perintah pada suatu object di dalam class, kita menggunakan <instansi>.<object\_name()>.

Ketika kita akan mengeluarkan output dari hewan.ayam(), maka class akan otomatis memanggil fungsi, atau di sini kita sebut dengan object di dalam instansi yang telah kita buat, dan akan mengeluarkan output “kukuruyuk” sesuai dengan perintah print di dalam object ayam().

Contoh :

## 7.4 Membuat aplikasi konversi suhu dengan python

Berikut adalah codingan dari aplikasi conversi suhu dengan python :

**Konvesi suhu celcius berorientasi objek**

class Celcius:

    def \_\_init\_\_(self, suhu):

        self.suhu = suhu

    def get\_celcius(self):

        val = self.suhu

        return val

    def get\_fahrenheit(self):

        val = (9/5 \* self.suhu) + 32

        return val

    def get\_reamur(self):

        val = (4/5 \* self.suhu)

        return val

    def get\_kelvin(self):

        val = self.suhu + 273

        return val

suhu = input(" Masukkan suhu dalam Celcius: ")

C = Celcius(float(suhu))

val = C.get\_celcius()

F = C.get\_fahrenheit()

R = C.get\_reamur()

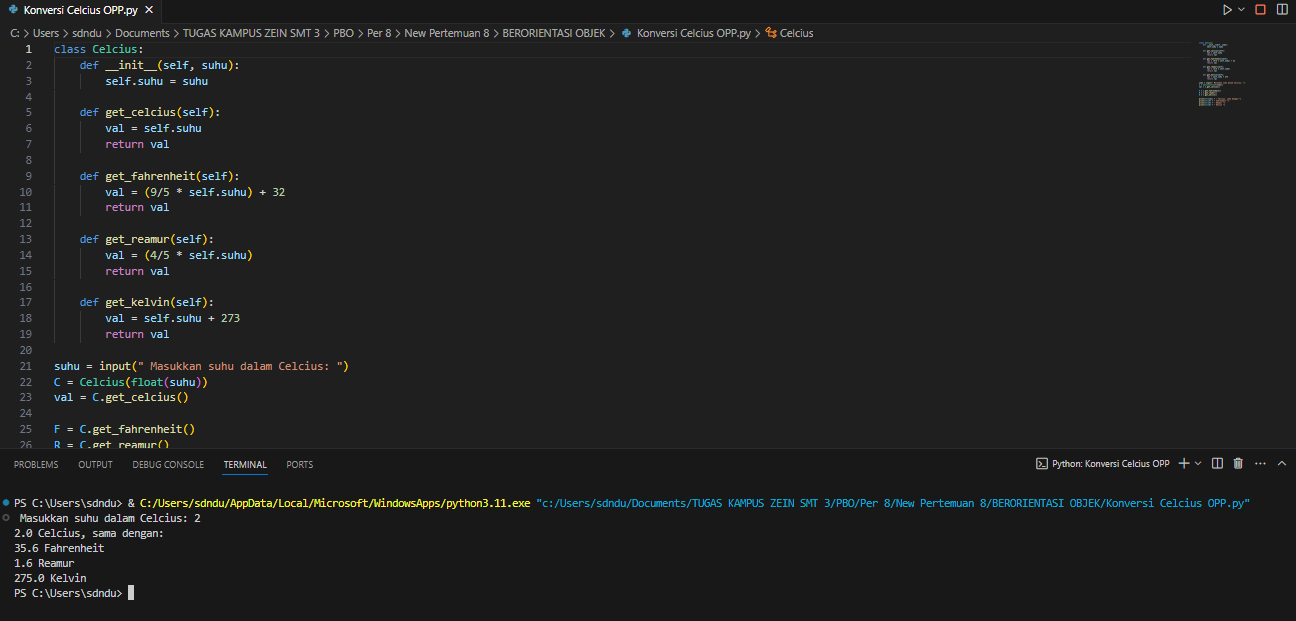
K = C.get\_kelvin()

print(str(val) + " Celcius, sama dengan:")

print(str(F) + " Fahrenheit ")

print(str(R) + " Reamur ")

print(str(K) + " Kelvin ")

output :

**konversi suhu celcius GUI OOP**

from tkinter import Frame,Label,Entry,Button,YES,BOTH,END,Tk,W

from Celcius import Celcius

class FrmCelcius:

    def \_\_init\_\_(self, parent, title):

        self.parent = parent

        #self.parent.geometry("400x400")

        self.parent.title(title)

        self.parent.protocol("WM\_DELETE\_WINDOW", self.onKeluar)

        self.aturKomponen()

    def aturKomponen(self):

        mainFrame = Frame(self.parent, bd=10)

        mainFrame.pack(fill=BOTH, expand=YES)

        Label(mainFrame, text='Celcius:').grid(row=0, column=0,

            sticky=W, padx=5, pady=5)

        Label(mainFrame, text="Fahrenheit:").grid(row=2, column=0,

            sticky=W, padx=5, pady=5)

        Label(mainFrame, text="Reamur:").grid(row=3, column=0,

            sticky=W, padx=5, pady=5)

        Label(mainFrame, text="Kelvin:").grid(row=4, column=0,

            sticky=W, padx=5, pady=5)

        self.txtCelcius = Entry(mainFrame)

        self.txtCelcius.grid(row=0, column=1, padx=5, pady=5)

        self.txtFahrenheit = Entry(mainFrame)

        self.txtFahrenheit.grid(row=2, column=1, padx=5, pady=5)

        self.txtReamur = Entry(mainFrame)

        self.txtReamur.grid(row=3, column=1, padx=5, pady=5)

        self.txtKelvin = Entry(mainFrame)

        self.txtKelvin.grid(row=4, column=1, padx=5, pady=5)

        self.btnHitung = Button(mainFrame, text='Hitung',

            command=self.onHitung)

        self.btnHitung.grid(row=1, column=1, padx=5, pady=5)

    def onHitung(self):

        C = Celcius(int(self.txtCelcius.get()))

        # Suhu dalam Fahrenheit

        F = C.get\_fahrenheit()

        self.txtFahrenheit.delete(0,END)

        self.txtFahrenheit.insert(END,str(F))

        # Suhu dalam Fahrenheit

        R = C.get\_reamur()

        self.txtReamur.delete(0,END)

        self.txtReamur.insert(END,str(R))

        # Suhu dalam Fahrenheit

        K = C.get\_kelvin()

        self.txtKelvin.delete(0,END)

        self.txtKelvin.insert(END,str(K))

    def onKeluar(self, event=None):

        # memberikan perintah menutup aplikasi

        self.parent.destroy()

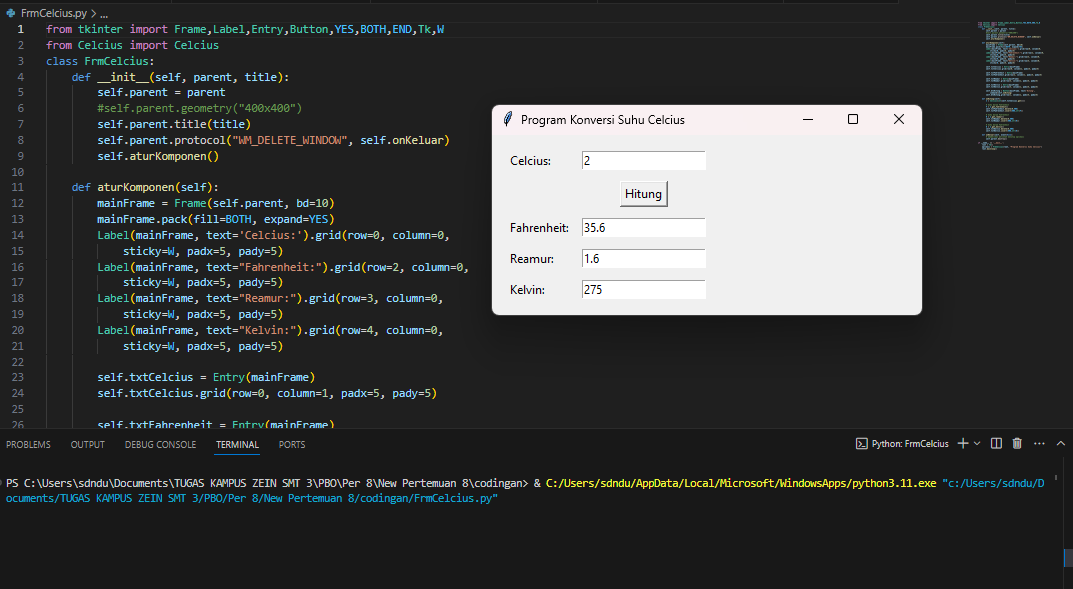
if \_\_name\_\_ == '\_\_main\_\_':

    root = Tk()

    aplikasi = FrmCelcius(root, "Program Konversi Suhu Celcius")

    root.mainloop()

Output :



**Konvesi suhu celcius GUI Terstruktur :**

import tkinter as tk

from tkinter import Frame,Label,Entry,Button,END, W

def get\_fahrenheit():

    suhu = txtsuhu.get()

    F = (9/5 \* float(suhu)) + 32

    txtFahrenheit.delete(0,END)

    txtFahrenheit.insert(END,F)

def get\_reamur():

    suhu = txtsuhu.get()

    R = (4/5 \* float(suhu))

    txtReamur.delete(0,END)

    txtReamur.insert(END,R)

def get\_kelvin():

    suhu = txtsuhu.get()

    K = float(suhu) + 273

    txtKelvin.delete(0,END)

    txtKelvin.insert(END,K)

def hitung():

    get\_fahrenheit()

    get\_reamur()

    get\_kelvin()

# Create tkinter object

app = tk.Tk()

# Tambahkan judul

app.title("Kalkulator Suhu Celcius")

# Windows

frame = Frame(app)

frame.pack(padx=20, pady=20)

# Label

suhu= Label(frame, text="Celcius:")

suhu.grid(row=0, column=0, sticky=W, padx=5, pady=5)

# Textbox

txtsuhu = Entry(frame)

txtsuhu.grid(row=0, column=1)

# Button

hitung\_button = Button(frame, text="Hitung", command=hitung)

hitung\_button.grid(row=2, column=1, sticky=W, padx=5, pady=5)

F= Label(frame, text="Fahrenheit:")

F.grid(row=3, column=0, sticky=W, padx=5, pady=5)

R= Label(frame, text="Reamur:")

R.grid(row=4, column=0, sticky=W, padx=5, pady=5)

K= Label(frame, text="Kelvin:")

K.grid(row=5, column=0, sticky=W, padx=5, pady=5)

# Output Textbox

txtFahrenheit = Entry(frame)

txtFahrenheit.grid(row=3, column=1, sticky=W, padx=5, pady=5)

txtReamur = Entry(frame)

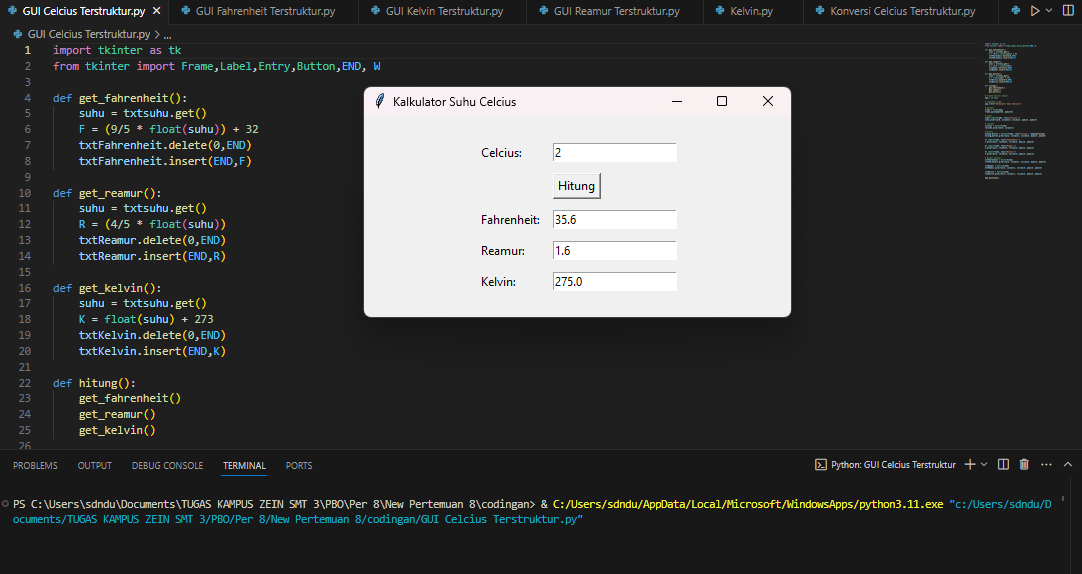
txtReamur.grid(row=4, column=1, sticky=W, padx=5, pady=5)

txtKelvin = Entry(frame)

txtKelvin.grid(row=5, column=1, sticky=W, padx=5, pady=5)

app.mainloop()

Output :



**Konversi suhu Celsius terstruktur :**

print("Konversi Suhu Celcius")

def get\_fahrenheit(suhu):

    F = (9/5 \* float(suhu)) + 32

    return F

def get\_reamur(suhu):

    R = (4/5 \* float(suhu))

    return R

def get\_kelvin(suhu):

    K = float(suhu) + 273

    return K

# Entry

suhu = input(" Masukkan Suhu Dalam Celcius: ")

# rumus

F = get\_fahrenheit(suhu)

R = get\_reamur(suhu)

K = get\_kelvin(suhu)

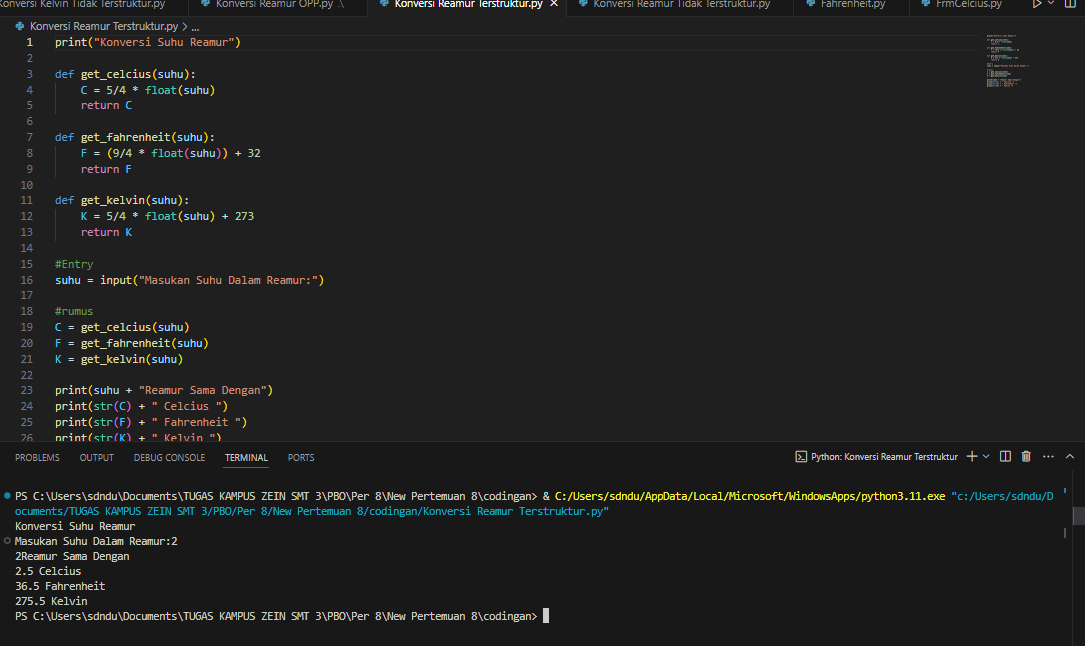
# Output

print(suhu + " Celcius sama dengan ")

print(str(F) + " Fahrenheit ")

print(str(R) + " Reamur ")

print(str(K) + " Kelvin ")

Output :

**Konversi suhu celcius tidak terstruktur** :

print("Konversi Suhu Celcius")

# Entry

suhu = input("Masukkan Suhu Dalam Celcius:")

# rumus

F = (9/5 \* float (suhu)) + 32

R = (4/5 \* float(suhu))

K = float(suhu) + 273

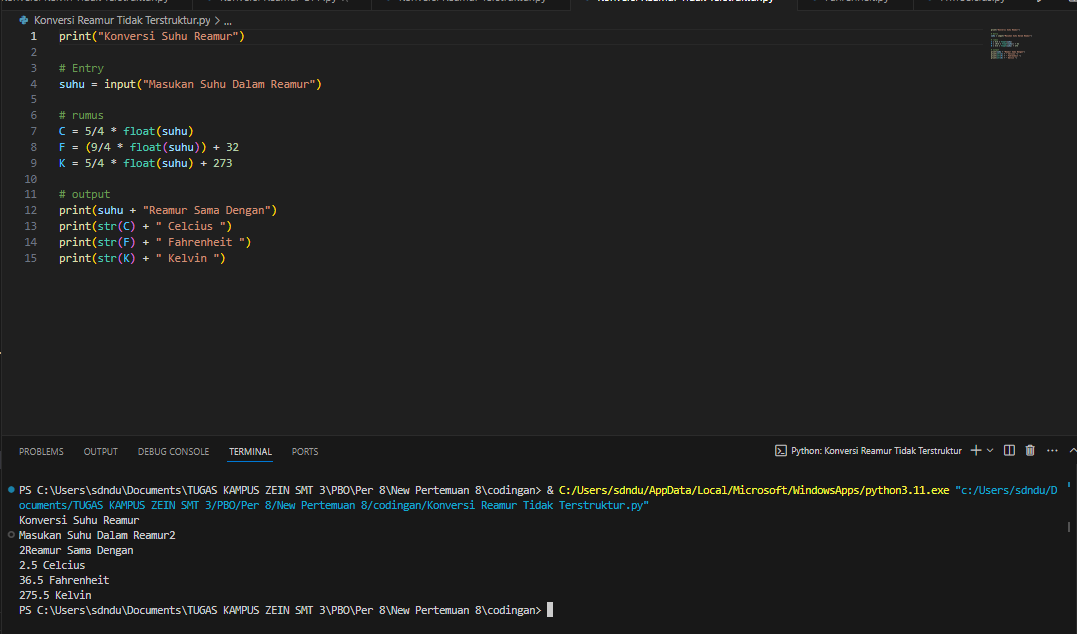
#output

print(suhu + " Celcius sama dengan ")

print(str (F)+ " Fahrenheit ")

print(str (R) + " Reamur ")

print (str(K) + " Kelvin ")

output :

**konversi suhu celcius OOP murni**

from tkinter import Frame,Label,Entry,Button,YES,BOTH,END,Tk,W

class FrmCelcius:

    def \_\_init\_\_(self, parent, title):

        self.parent = parent

        #self.parent.geometry("400x400")

        self.parent.title(title)

        self.parent.protocol("WM\_DELETE\_WINDOW", self.onKeluar)

        self.aturKomponen()

    def aturKomponen(self):

        mainFrame = Frame(self.parent, bd=10)

        mainFrame.pack(fill=BOTH, expand=YES)

        Label(mainFrame, text='Celcius:').grid(row=0, column=0,

            sticky=W, padx=5, pady=5)

        Label(mainFrame, text="Fahrenheit:").grid(row=2, column=0,

            sticky=W, padx=5, pady=5)

        Label(mainFrame, text="Reamur:").grid(row=3, column=0,

            sticky=W, padx=5, pady=5)

        Label(mainFrame, text="Kelvin:").grid(row=4, column=0,

            sticky=W, padx=5, pady=5)

        self.txtCelcius = Entry(mainFrame)

        self.txtCelcius.grid(row=0, column=1, padx=5, pady=5)

        self.txtFahrenheit = Entry(mainFrame)

        self.txtFahrenheit.grid(row=2, column=1, padx=5, pady=5)

        self.txtReamur = Entry(mainFrame)

        self.txtReamur.grid(row=3, column=1, padx=5, pady=5)

        self.txtKelvin = Entry(mainFrame)

        self.txtKelvin.grid(row=4, column=1, padx=5, pady=5)

        self.btnHitung = Button(mainFrame, text='Hitung',

            command=self.onHitung)

        self.btnHitung.grid(row=1, column=1, padx=5, pady=5)

    def get\_fahrenheit(self, suhu):

        val = (9/5 \* float(suhu)) + 32

        return val

    def get\_reamur(self, suhu):

        val = (4/5 \* float(suhu))

        return val

    def get\_kelvin(self, suhu):

        val = (float(suhu) + 273)

        return val

    def onHitung(self):

        suhu = self.txtCelcius.get()

        # Suhu dalam Fahrenheit

        F = self.get\_fahrenheit(float(suhu))

        self.txtFahrenheit.delete(0,END)

        self.txtFahrenheit.insert(END,str(F))

        # Suhu dalam Reamur

        R = self.get\_reamur(float(suhu))

        self.txtReamur.delete(0,END)

        self.txtReamur.insert(END,str(R))

        # Suhu dalam Kelvin

        K = self.get\_kelvin(float(suhu))

        self.txtKelvin.delete(0,END)

        self.txtKelvin.insert(END,str(K))

    def onKeluar(self, event=None):

        # memberikan perintah menutup aplikasi

        self.parent.destroy()

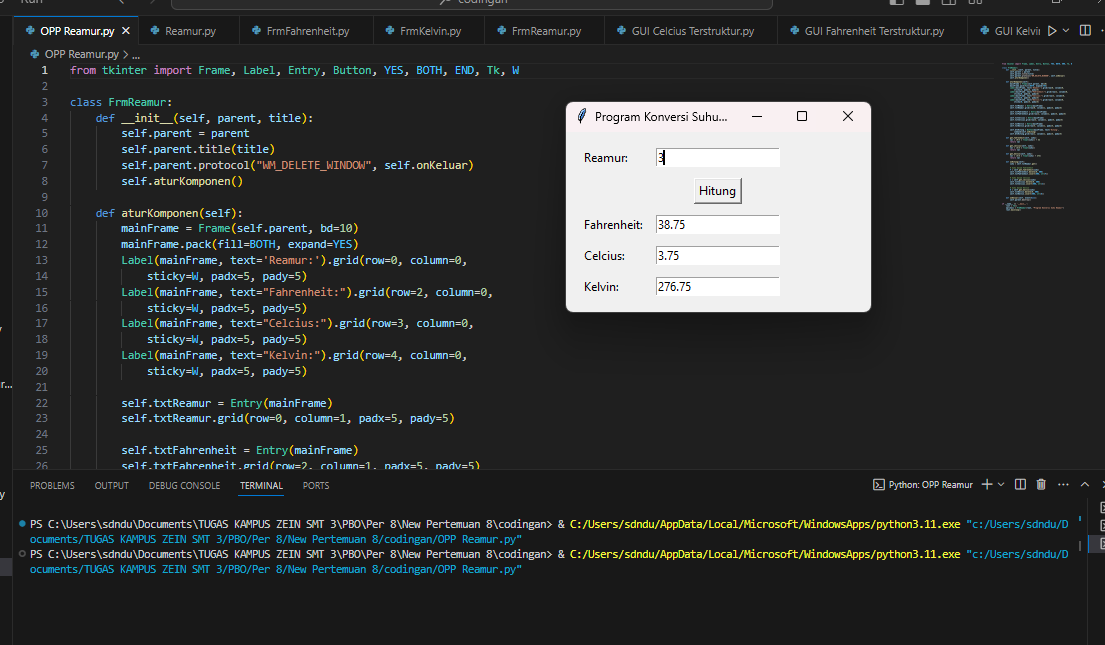
if \_\_name\_\_ == '\_\_main\_\_':

    root = Tk()

    aplikasi = FrmCelcius(root, "Program Konversi Suhu Celcius")

    root.mainloop()

output :



# PERTEMUAN 9

# PENGENALAN MYSQL

## 8.1 PENGERTIAN MYSQL

MySQL yang dibaca “MY-ES-KYOO-EL" merupakan sistem manajemen database yang bersifat open-source yang menggunakan perintah dasar atau bahasa pemrograman yang berupa structured query language (SQL) yang cukup populer di dunia teknologi. MySQL berguna sebagai database. Saat ini, relational database management system (RDBMS) MySQL telah dipakai lebih dari 66 juta pengguna di seluruh belahan dunia.

Adapun MySQL memiliki dua bentuk lisensi, yaitu Free Software dan Shareware atau perangkat lunak bermilik yang penggunaannya terbatas. Jadi, MySQL adalah database server gratis dengan lisensi GNU General Public License (GPL) yang bisa dipakai untuk keperluan pribadi atau komersil tanpa harus membayar lisensi yang ada.

Dalam istilah pemrograman, SQL sendiri menjadi bahasa yang dipakai di dalam pengambilan data pada relational database atau database yang terstruktur. Dengan kata lain, MySQL merupakan database management system yang menggunakan bahasa SQL sebagai bahasa penghubung antara perangkat lunak aplikasi dengan database server.

## 8.2 FUNGSI MYSQL

Fungsi utama MySQL adalah mengelola informasi di database pada sisi server dengan bahasa pemrograman SQL. Saat ini, hampir seluruh pihak hosting menyediakan MySQL untuk pengembangan web.

## 8.3 KELEBIHAN MYSQL

Berikut beberapa kelebihan MySQL :

1. Open-source

Software ini menjadi sistem manajemen database yang bersifat open-source atau dengan kata lain, gratis digunakan oleh siapa saja. Walau demikian, MySQL memiliki performa yang tidak dapat diragukan lagi. Nah, bila selama menggunakan MySQL menemukan kendala, pengguna bisa dengan mudah menemukan review atau komunitas yang bisa dijadikan acuan maupun wadah diskusi MySQL untuk membantu.

2. Keamanan terjamin

Walaupun open-source, MySQL menjadi DBMS dengan sistem keamanan yang baik. Sebab, MySQL memiliki lebih dari satu lapisan keamanan, seperti berbasis host maupun enkripsi kata sandi yang membuat data-data pengguna MySQL dapat selalu terlindungi.

3. Multi-user

MySQL mendukung multi-user atau bisa dipakai secara bersamaan oleh beberapa user tanpa perlu khawatir project yang dikerjakan secara bersamaan akan mengalami crash atau tidak tersimpan.

4. Mendukung bahasa pemrograman lain

Selain bahasa pemrograman SQL, MySQL juga mendukung bahasa pemrograman lain. Dengan begitu, proses pengembangan yang dilakukan oleh pengguna dan memerlukan berbagai macam bahasa pemrograman pun bisa berjalan lebih efektif. Adapun bahasa-bahasa pemrograman yang bisa menggunakan MySQL antara lain PHP, JavaScript, dan lain sebagainya.

5. Tidak membutuhkan RAM yang besar

MySQL menjadi database management system yang tidak membutuhkan RAM dengan ukuran yang besar dan bisa dipakai pada spesifikasi hardware yang rendah. Bahkan, disebut-sebut perangkat dengan RAM yang kecil sekali pun mampu menjalankan MySQL dengan lancar.

6. Mendukung berbagai macam data

Selanjutnya, MySQL juga mendukung berbagai macam data, mulai dari integer, timestamp, text, date, character, float, dan lain-lain sehingga sangat membantu pengelolaan database server pada web atau aplikasi yang dikembangkan.

7. Struktur tabel fleksibel

Kelebihan lain yang dimiliki MySQL adalah memiliki struktur tabel yang mudah dipakai dan fleksibel. Seperti, saat MySQL memproses Alter Table.

8. Standar industri

Saat ini, ada banyak cluster server yang telah men-support MySQL. Dengan performa MySQL, software database ini akan membantu pengguna dalam menyimpan berbagai data dalam jumlah yang banyak maupun ketika melakukan kegiatan bisnis lainnya.

## 8.4 KEKURANGAN MYSQL

Beberapa kekurangan MySQL :

1. Kurang cocok untuk aplikasi game dan mobile

Kebanyakan pengembang game maupun aplikasi mobile tidak menggunakan MySQL karena database ini masih kurang bagus untuk mengembangkan sistem aplikasi tersebut. Untuk itu, bagi pengembang aplikasi game dan perangkat mobile, ada baiknya bila mempertimbangkan MySQL lagi.

2. Masih sulit untuk mengelola database dengan ukuran yang besar

MySQL dikembangkan supaya ramah dengan perangkat yang memiliki spesifikasi rendah. Inilah mengapa MySQL tidak memiliki fitur yang lengkap layaknya aplikasi lainnya. Jadi, ada baiknya untuk menggunakan database manajemen sistem selain MySQL saat ingin mengembangkan sistem atau aplikasi perusahaan yang memiliki database yang cukup besar.

3. Kurang bagus dari segi technical support

Walaupun MySQL memiliki banyak komunitas dan dokumentasi yang telah tersebar luas di internet, tetapi tidak tersedia technical support MySQL untuk yang community. Sementara, pengguna Enterprise tersedia technical support dan berbayar.

## 8.5 CARA KERJA MYSQL

Adapun cara kerja MySQL dimulai dari perangkat yang ingin digunakan harus sudah terhubung ke server. Satu atau beberapa perangkat yang terhubung ke server dapat membuat request dari graphical user interface (GUI) di layar dan server pun akan membuat output yang diinginkan. Maka, proses yang akan dijalankan atau terjadi pada MySQL kurang lebih seperti ini:

* 1. MySQL membuat database yang dapat memodifikasi, menyimpan data, dan menentukan keterkaitan tabel-tabel yang ada di dalam software.
  2. Kemudian, perangkat pengguna membuat request dengan perintah spesifik menggunakan bahasa SQL.
  3. Terakhir, server akan menerima dan menjalankan perintah. Kemudian, memberikan atau menampilkan informasi yang diminta pengguna pada layarnya.
  4. Semakin ringan dan user friendly suatu GUI, maka semakin cepat dan mudah aktivitas manajemen data yang dimilikinya. Adapun beberapa MySQL GUI terpopuler meliputi MySQL WorkBench, SequelPro, DBVisualizer, Navicat DB Admin Tool, dan tool gratis HeidiSQL. Namun, untuk manajemen database berbasis web yang umum digunakan adalah phpMyAdmin.

Sebelum menginstal MySQL pada Windows, hal-hal yang perlu dipersiapkan antara lain:

1. Mengunduh MySQL Installer dari situs web resmi MySQL.
2. Memilih tipe instalasi yang akan digunakan, seperti "Custom" atau "Developer Default".
3. Memastikan komponen yang akan diinstal, seperti MySQL Server, MySQL Workbench, dan MySQL Shell.
4. Memilih konfigurasi untuk MySQL Server, termasuk apakah akan menginstal MySQL sebagai "Windows Service" atau tidak.

Setelah persiapan ini, Anda dapat mengikuti langkah-langkah instalasi yang ditentukan oleh MySQL Installer. Pastikan untuk memilih opsi yang sesuai dengan kebutuhan Anda, misalnya, apakah Anda ingin menginstal MySQL sebagai "Windows Service" atau tidak.

# PERTEMUAN 10

# PYTHON DATABASE APPLICATION

## 9.1 DATABASE CONNECTION

Sebelum terhubung ke database MySQL, pastikan beberapa hal dibawah ini :

1. Anda telah membuat database TESTDB.

2. Anda telah membuat tabel EMPLOYEE di TESTDB.

3. Tabel ini memiliki bidang FIRST\_NAME, LAST\_NAME, AGE, SEX, dan INCOME.

4. User ID "testuser" dan password "test123" diatur untuk mengakses TESTDB.

5. Python modul PyMySQL terinstal dengan benar pada mesin Anda.

6. Anda telah melalui tutorial MySQL untuk memahami Dasar-Dasar MySQL

Berikut ini adalah contoh koneksi dengan database MySQL "TESTDB"

import pymysql.cursors

# Open database connection

db = pymysql.connect("localhost","testuser","test123","TESTDB" )

# prepare a cursor object using cursor() method

cursor = db.cursor()

# execute SQL query using execute() method.

cursor.execute("SELECT VERSION()")

# Fetch a single row using fetchone() method.

data = cursor.fetchone()

print ("Database version : %s " % data)

# disconnect from server

db.close()

## 9.2 MEMBUAT TABEL DATABASE

import pymysql.cursors

# Open database connection

db = pymysql.connect("localhost","testuser","test123","TESTDB" )

# prepare a cursor object using cursor() method

cursor = db.cursor()

# Drop table if it already exist using execute() method.

cursor.execute("DROP TABLE IF EXISTS EMPLOYEE")

# Create table as per requirement

sql = """CREATE TABLE EMPLOYEE (

   FIRST\_NAME  CHAR(20) NOT NULL,

   LAST\_NAME  CHAR(20),

   AGE INT,

   SEX CHAR(1),

   INCOME FLOAT )"""

cursor.execute(sql)

# disconnect from server

db.close()

## 9.3 OPERASI INSERT

import pymysql.cursors

# Open database connection

db = pymysql.connect("localhost","testuser","test123","TESTDB" )

# prepare a cursor object using cursor() method

cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql = """INSERT INTO EMPLOYEE(FIRST\_NAME,

   LAST\_NAME, AGE, SEX, INCOME)

   VALUES ('Mac', 'Mohan', 20, 'M', 2000)"""

try:

   # Execute the SQL command

   cursor.execute(sql)

   # Commit your changes in the database

   db.commit()

except:

   # Rollback in case there is any error

   db.rollback()

# disconnect from server

db.close()

Contoh di atas bisa dituliskan sebagai berikut untuk membuat query SQL secara dinamis

import pymysql.cursors

# Open database connection

db = pymysql.connect("localhost","testuser","test123","TESTDB" )

# prepare a cursor object using cursor() method

cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql = "INSERT INTO EMPLOYEE(FIRST\_NAME, \

   LAST\_NAME, AGE, SEX, INCOME) \

   VALUES ('%s', '%s', '%d', '%c', '%d' )" % \

   ('Mac', 'Mohan', 20, 'M', 2000)

try:

   # Execute the SQL command

   cursor.execute(sql)

   # Commit your changes in the database

   db.commit()

except:

   # Rollback in case there is any error

   db.rollback()

# disconnect from server

db.close()

## 9.5 READ OPERATION

READ Operation pada database apapun berarti mengambil beberapa informasi berguna dari database.

Setelah koneksi database terbentuk, Anda siap untuk membuat query ke dalam database ini. Anda bisa menggunakan metode fetchone() untuk mengambil satu record atau fetchall() metode untuk mengambil beberapa nilai dari tabel database.

Fetchone () - Ini mengambil baris berikut dari kumpulan hasil query. Set hasil adalah objek yang dikembalikan saat objek kursor digunakan untuk query tabel.

Fetchall () - Ini menjemput semua baris dalam kumpulan hasil. Jika beberapa baris telah diekstraksi dari himpunan hasil, maka akan diambil baris yang tersisa dari kumpulan hasil.

Rowcount - Ini adalah atribut read-only dan mengembalikan jumlah baris yang dipengaruhi oleh metode execute ().

Prosedur berikut menanyakan semua catatan dari tabel EMPLOYEE yang memiliki gaji lebih dari 1000

import pymysql.cursors

# Open database connection

db = pymysql.connect("localhost","testuser","test123","TESTDB" )

# prepare a cursor object using cursor() method

cursor = db.cursor()

# Prepare SQL query to INSERT a record into the database.

sql = "SELECT \* FROM EMPLOYEE \

       WHERE INCOME > '%d'" % (1000)

try:

   # Execute the SQL command

   cursor.execute(sql)

   # Fetch all the rows in a list of lists.

   results = cursor.fetchall()

   for row in results:

      fname = row[0]

      lname = row[1]

      age = row[2]

      sex = row[3]

      income = row[4]

      # Now print fetched result

      print ("fname = %s,lname = %s,age = %d,sex = %s,income = %d" % \

             (fname, lname, age, sex, income ))

except:

   print ("Error: unable to fetch data")

# menutup koneksi ke server

db.close()

Setelah Anda eksekusi kode diatas, akan muncul hasil seperti dibawah ini : fname = Mac, lname = Mohan, age = 20, sex = M, income = 2000

## 9.6 UPDATE OPERATION

Operasi UPDATE pada database apapun berarti mengupdate satu atau lebih catatan, yang sudah tersedia di database. Prosedur berikut memperbarui semua catatan yang memiliki SEX sebagai 'M'. Di sini, kita meningkatkan UMUR semua laki-laki satu tahun.

import pymysql.cursors

# Open database connection

db = pymysql.connect("localhost","testuser","test123","TESTDB" )

# prepare a cursor object using cursor() method

cursor = db.cursor()

# Prepare SQL query to UPDATE required records

sql = "UPDATE EMPLOYEE SET AGE = AGE + 1

                          WHERE SEX = '%c'" % ('M')

try:

   # Execute the SQL command

   cursor.execute(sql)

   # Commit your changes in the database

   db.commit()

except:

   # Rollback in case there is any error

   db.rollback()

# disconnect from server

db.close()

## 9.7 DELETE OPERATION

Operasi DELETE diperlukan bila Anda ingin menghapus beberapa catatan dari database Anda. Berikut ini adalah prosedur untuk menghapus semua catatan dari EMPLOYEE dimana AGE lebih dari 20

import pymysql.cursors

# Open database connection

db = pymysql.connect("localhost","testuser","test123","TESTDB" )

# prepare a cursor object using cursor() method

cursor = db.cursor()

# Prepare SQL query to DELETE required records

sql = "DELETE FROM EMPLOYEE WHERE AGE > '%d'" % (20)

try:

   # Execute the SQL command

   cursor.execute(sql)

   # Commit your changes in the database

   db.commit()

except:

   # Rollback in case there is any error

   db.rollback()

# disconnect from server

db.close()

Selain itu masih ada beberapa operasi sebagai berikut :

1. Commit Operation db.commit()

2. Rollback Operation db.rollback()

3. Disconnect Operation db.close()

# PERTEMUAN 11

## 10.1 Pengertian MySQL :

My SQL Merupakan sistem manajemen *database* yang bersifat *open-source* yang menggunakan perintah dasar atau bahasa pemrograman yang berupa *structured query language* (SQL) yang cukup populer di dunia teknologi. MySQL berguna sebagai database.

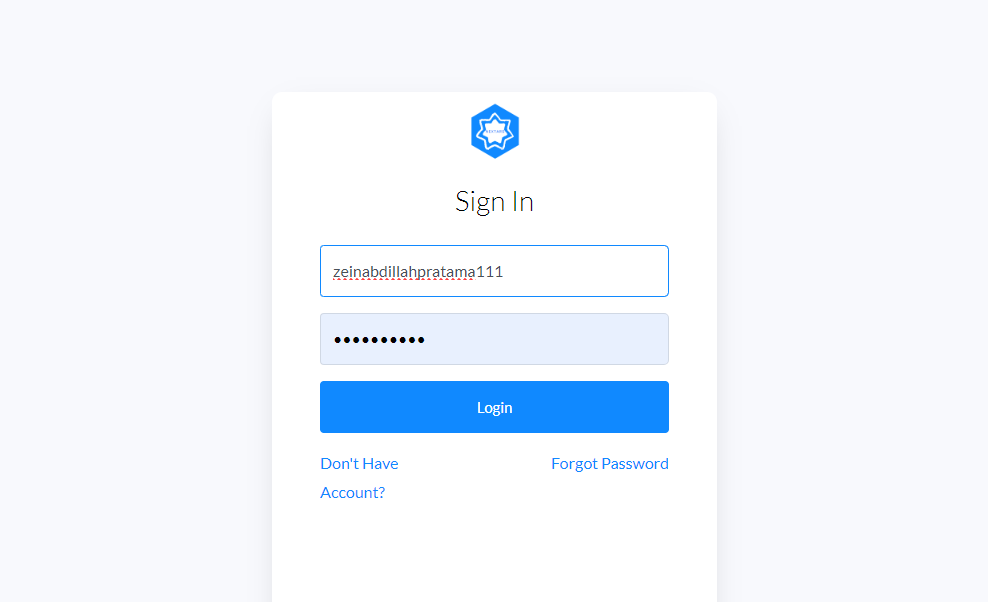
Adapun MySQL memiliki dua bentuk lisensi, yaitu *Free Software* dan *Shareware* atau perangkat lunak bermilik yang penggunaannya terbatas. Jadi, MySQL adalah database server gratis dengan lisensi GNU *General Public License* (GPL) yang bisa dipakai untuk keperluan pribadi atau komersil tanpa harus membayar lisensi yang ada.

Dalam istilah pemrograman, SQL sendiri menjadi bahasa yang dipakai di dalam pengambilan data pada *relational database* atau *database* yang terstruktur. Dengan kata lain, MySQL merupakan *database management system* yang menggunakan bahasa SQL sebagai bahasa penghubung antara perangkat lunak aplikasi dengan *database* server.

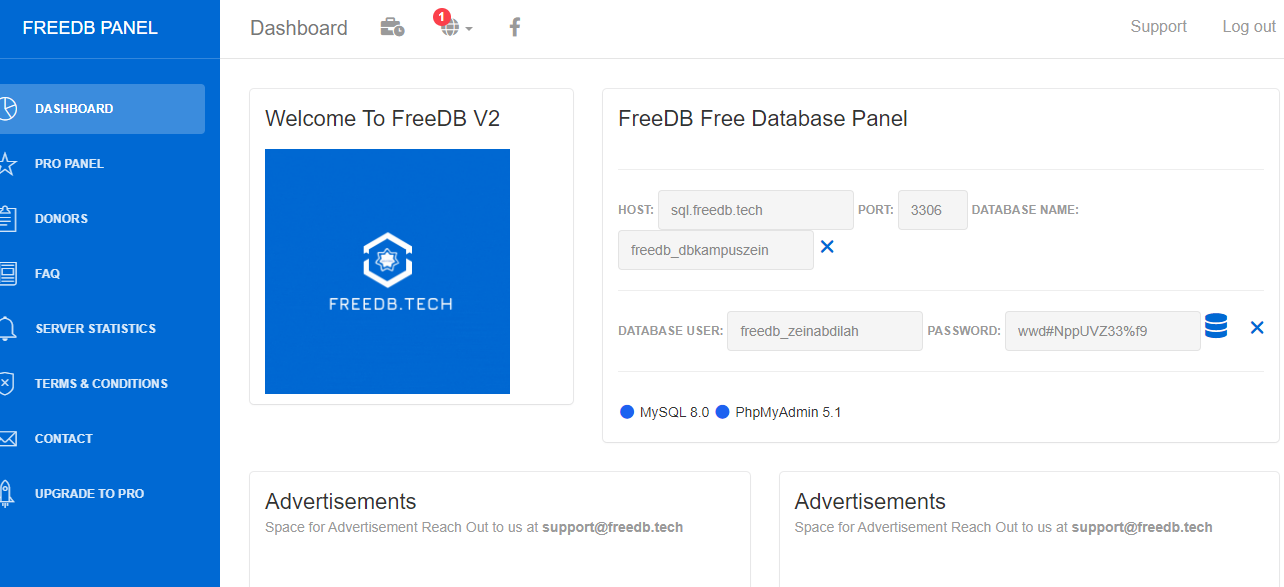
## 10.2 Pengertian Database :

Pengertian database adalah sekumpulan data yang dikelola berdasarkan ketentuan tertentu yang saling berkaitan sehingga memudahkan dalam pengelolaannya. Dihimpun dari berbagai sumber, secara sederhana, database atau basis data merupakan sekumpulan data atau informasi yang tersimpan secara sistematis. Database memiliki peran penting dalam perangkat untuk mengumpulkan informasi, data, atau file secara terintegrasi.

**Membuat aplikasi pendataan dokter dengan MySQL server**

1. Pertama kita login ke mysql server terlebih dahulu, bagi yang belum daftar silahkan daftar telebih dahulu

Setelah sudah daftar buatlah database nya, dan jika database sudah valid maka ada opsi selanjutnya dan jika database tidak valid maka database tidak terinput sampai database tersebut valid.

1. Jika database sudah valid maka tampilan akan seperti ini :
2. Setelah server di mysql sudah siap / ready kita masuk ke dalam aplikasi vscode untuk mebuat codingan, dan tahap selanjutnya buat file di vscode dengan nama **Db.py**, dengan codingan sebagi berikut :

import mysql.connector as mc

class DBConnection:

    def \_\_init\_\_(self):

        self.host = "sql.freedb.tech"

        self.port = 3306

        self.name = "freedb\_dbkampuszein"

        self.user = "freedb\_zeinabdilah"

        self.password = "wwd#NppUVZ33%f9"

        self.conn = None

        self.cursor = None

        self.result = None

        self.connected = False

        self.affected = 0

        self.connect()

    @property

    def connection\_status(self):

        return self.connected

    def connect(self):

        try:

            self.conn = mc.connect(host = self.host,

                                    port = self.port,

                                    database = self.name,

                                    user = self.user,

                                    password = self.password)

            self.connected = True

            self.cursor=self.conn.cursor()

        except mc.Error as e:

            self.connected = False

        return self.conn

    def disconnect(self):

        if(self.connected==True):

            self.conn.close

        else:

            self.conn = None

    def findOne(self, sql):

        self.connect()

        self.cursor.execute(sql)

        self.result = self.cursor.fetchone()

        return self.result

    def findAll(self, sql):

        self.connect()

        self.result = self.cursor.execute(sql)

        self.result = self.cursor.fetchall()

        return self.result

    def insert(self, sql):

        self.connect()

        self.cursor.execute(sql)

        self.conn.commit()

        self.affected = self.cursor.rowcount

        return self.affected

    def update(self, sql, val):

        self.connect()

        self.cursor.execute(sql, val)

        self.conn.commit()

        self.affected = self.cursor.rowcount

        return self.affected

    def delete(self, sql):

        self.connect()

        self.cursor.execute(sql)

        self.conn.commit()

        self.affected = self.cursor.rowcount

        return self.affected

    def show(self, sql):

        self.connect()

        self.cursor.execute(sql)

        self.result = self.cursor.fetchone()

        return self.result

    @property

    def info(self):

        if(self.connected==True):

            return "Server is running on " + self.host + ' using port ' + str(self.port)

        else:

            return "Server is offline."

**Catatan : untuk bagian host, port, name, users, password itu kita samakan dengan yang di server mysql milik pribadi, dan pastikan ketika di run itu server sudah terkoneksi / sudah online**

1. Setelah server sudah terkoneksi lalu kita buat file baru dengan nama **Dokter.py** dengan codingan sebagai berikut :

from db import DBConnection as mydb

class Dokter:

    def \_\_init\_\_(self):

        self.\_\_id = None

        self.\_\_nip = None

        self.\_\_nama = None

        self.\_\_jk = None

        self.\_\_spesialis = None

        self.\_\_tempat\_bertugas = None

        self.conn = None

        self.affected = None

        self.result = None

    @property

    def id(self):

        return self.\_\_id

    @property

    def nip(self):

        return self.\_\_nip

    @nip.setter

    def nip(self, value):

        self.\_\_nip = value

    @property

    def nama(self):

        return self.\_\_nama

    @nama.setter

    def nama(self, value):

        self.\_\_nama = value

    @property

    def jk(self):

        return self.\_\_jk

    @jk.setter

    def jk(self, value):

        self.\_\_jk = value

    @property

    def spesialis(self):

        return self.\_\_spesialis

    @spesialis.setter

    def spesialis(self, value):

        self.\_\_spesialis = value

    @property

    def tempat\_bertugas(self):

        return self.\_\_tempat\_bertugas

    @tempat\_bertugas.setter

    def tempat\_bertugas(self, value):

        self.\_\_tempat\_bertugas = value

    def simpan(self):

        self.conn = mydb()

        val = (self.\_\_nip, self.\_\_nama, self.\_\_jk, self.\_\_spesialis, self.\_\_tempat\_bertugas)

        sql = "INSERT INTO dokter (nip, nama, jk, spesialis, tempat\_bertugas) VALUES " + str(val)

        self.affected = self.conn.insert(sql)

        self.conn.disconnect()

        return self.affected

    def update(self, id):

        self.conn = mydb()

        val = (self.\_\_nip, self.\_\_nama, self.\_\_jk, self.\_\_spesialis, self.\_\_tempat\_bertugas, id)

        sql = "UPDATE dokter SET nip = %s, nama = %s, jk=%s, spesialis=%s, tempat\_bertugas=%s WHERE id=%s"

        self.affected = self.conn.update(sql, val)

        self.conn.disconnect()

        return self.affected

    def updateByNIP(self, nip):

        self.conn = mydb()

        val = (self.\_\_nama, self.\_\_jk, self.\_\_spesialis, self.\_\_tempat\_bertugas, nip)

        sql = "UPDATE dokter SET nama = %s, jk=%s, spesialis=%s, tempat\_bertugas=%s WHERE nip=%s"

        self.affected = self.conn.update(sql, val)

        self.conn.disconnect()

        return self.affected

    def delete(self, id):

        self.conn = mydb()

        sql = "DELETE FROM dokter WHERE id='" + str(id) + "'"

        self.affected = self.conn.delete(sql)

        self.conn.disconnect()

        return self.affected

    def deleteByNIP(self, nip):

        self.conn = mydb()

        sql = "DELETE FROM dokter WHERE nip='" + str(nip) + "'"

        self.affected = self.conn.delete(sql)

        self.conn.disconnect()

        return self.affected

    def getByID(self, id):

        self.conn = mydb()

        sql = "SELECT \* FROM dokter WHERE id='" + str(id) + "'"

        self.result = self.conn.findOne(sql)

        self.\_\_nip = self.result[1]

        self.\_\_nama = self.result[2]

        self.\_\_jk = self.result[3]

        self.\_\_spesialis = self.result[4]

        self.\_\_tempat\_bertugas = self.result[5]

        self.conn.disconnect()

        return self.result

    def getByNIP(self, nip):

        a = str(nip)

        b = a.strip()

        self.conn = mydb()

        sql = "SELECT \* FROM dokter WHERE nip='" + b + "'"

        self.result = self.conn.findOne(sql)

        if(self.result != None):

            self.\_\_nip = self.result[1]

            self.\_\_nama = self.result[2]

            self.\_\_jk = self.result[3]

            self.\_\_spesialis = self.result[4]

            self.\_\_tempat\_bertugas = self.result[5]

            self.affected = self.conn.cursor.rowcount

        else:

            self.\_\_nip = ''

            self.\_\_nama = ''

            self.\_\_jk = ''

            self.\_\_spesialis = ''

            self.\_\_tempat\_bertugas = ''

            self.affected = 0

        self.conn.disconnect()

        return self.result

    def getAllData(self):

        self.conn = mydb()

        sql = "SELECT \* FROM dokter"

        self.result = self.conn.findAll(sql)

        return self.result

# Tampilkan semua data

A = Dokter()

A.nip = "123456"

A.nama = "Dr. Wahid"

A.jk = "L"

A.spesialis = "Bedah Umum"

A.tempat\_bertugas = "Rumah Sakit XYZ"

A.updateByNIP(A.nip)

B = A.getAllData()

print(B)

1. Setelah itu buat lagi file dengan nama **FrmDokter.py** dengan codingan sebagai berikut :

import tkinter as tk

from tkinter import Frame, Label, Entry, Button, Radiobutton, ttk, VERTICAL, YES, BOTH, END, Tk, StringVar, messagebox

from Dokter import Dokter

class FrmDokter :

    def \_\_init\_\_(self, parent, title):

        self.parent = parent

        self.parent.geometry("500x500")

        self.parent.title(title)

        self.parent.protocol("WM\_DELETE\_WINDOW", self.onKeluar)

        self.ditemukan = None

        self.aturKomponen()

        self.onReload()

    def aturKomponen(self):

        mainFrame = Frame(self.parent, bd=10)

        mainFrame.pack(fill=BOTH, expand=YES)

        # Label

        Label(mainFrame, text='NIP:').grid(row=0, column=0, sticky=tk.W, padx=5, pady=5)

        self.txtNIP = Entry(mainFrame)

        self.txtNIP.grid(row=0, column=1, padx=5, pady=5)

        self.txtNIP.bind("<Return>", self.onCari)  # menambahkan event Enter key

        Label(mainFrame, text='Nama:').grid(row=1, column=0, sticky=tk.W, padx=5, pady=5)

        self.txtNama = Entry(mainFrame)

        self.txtNama.grid(row=1, column=1, padx=5, pady=5)

        Label(mainFrame, text='Jenis Kelamin:').grid(row=2, column=0, sticky=tk.W, padx=5, pady=5)

        self.txtJK = StringVar()

        self.L = Radiobutton(mainFrame, text='Laki-laki', value='L', variable=self.txtJK)

        self.L.grid(row=2, column=1, padx=5, pady=5, sticky=tk.W)

        self.L.select()  # set pilihan yg pertama

        self.P = Radiobutton(mainFrame, text='Perempuan', value='P', variable=self.txtJK)

        self.P.grid(row=3, column=1, padx=5, pady=5, sticky=tk.W)

        Label(mainFrame, text='Spesialis:').grid(row=4, column=0, sticky=tk.W, padx=5, pady=5)

        self.txtSpesialis = Entry(mainFrame)

        self.txtSpesialis.grid(row=4, column=1, padx=5, pady=5)

        Label(mainFrame, text='Tempat Bertugas:').grid(row=5, column=0, sticky=tk.W, padx=5, pady=5)

        self.txtTempatBertugas = Entry(mainFrame)

        self.txtTempatBertugas.grid(row=5, column=1, padx=5, pady=5)

        # Button

        self.btnSimpan = Button(mainFrame, text='Simpan', command=self.onSimpan, width=10)

        self.btnSimpan.grid(row=0, column=3, padx=5, pady=5)

        self.btnClear = Button(mainFrame, text='Clear', command=self.onClear, width=10)

        self.btnClear.grid(row=1, column=3, padx=5, pady=5)

        self.btnHapus = Button(mainFrame, text='Hapus', command=self.onDelete, width=10)

        self.btnHapus.grid(row=2, column=3, padx=5, pady=5)

        # define columns

        columns = ('id', 'nip', 'nama', 'jk', 'spesialis', 'tempat\_bertugas')

        self.tree = ttk.Treeview(mainFrame, columns=columns, show='headings')

        # define headings

        self.tree.heading('id', text='ID')

        self.tree.column('id', width="30")

        self.tree.heading('nip', text='NIP')

        self.tree.column('nip', width="60")

        self.tree.heading('nama', text='Nama')

        self.tree.column('nama', width="200")

        self.tree.heading('jk', text='JK')

        self.tree.column('jk', width="30")

        self.tree.heading('spesialis', text='Spesialis')

        self.tree.column('spesialis', width="100")

        self.tree.heading('tempat\_bertugas', text='Tempat Bertugas')

        self.tree.column('tempat\_bertugas', width="150")

        self.tree.place(x=0, y=200)

        self.onReload()

    def onClear(self, event=None):

        self.txtNIP.delete(0, END)

        self.txtNIP.insert(END, "")

        self.txtNama.delete(0, END)

        self.txtNama.insert(END, "")

        self.txtSpesialis.delete(0, END)

        self.txtSpesialis.insert(END, "")

        self.txtTempatBertugas.delete(0, END)

        self.txtTempatBertugas.insert(END, "")

        self.btnSimpan.config(text="Simpan")

        self.L.select()

        self.onReload()

        self.ditemukan = False

    def onReload(self, event=None):

        dokter = Dokter()

        result = dokter.getAllData()

        for item in self.tree.get\_children():

            self.tree.delete(item)

        doctors = []

        for row\_data in result:

            doctors.append(row\_data)

        for doctor in doctors:

            self.tree.insert('', END, values=doctor)

    def onCari(self, event=None):

        nip = self.txtNIP.get()

        dokter = Dokter()

        res = dokter.getByNIP(nip)

        rec = dokter.affected

        if(rec > 0):

            messagebox.showinfo("showinfo", "Data Ditemukan")

            self.TampilkanData()

            self.ditemukan = True

        else:

            messagebox.showwarning("showwarning", "Data Tidak Ditemukan")

            self.ditemukan = False

            self.txtNama.focus()

        return res

    def TampilkanData(self, event=None):

        nip = self.txtNIP.get()

        dokter = Dokter()

        res = dokter.getByNIP(nip)

        self.txtNama.delete(0, END)

        self.txtNama.insert(END, dokter.nama)

        jk = dokter.jk

        if(jk == "P"):

            self.P.select()

        else:

            self.L.select()

        self.txtSpesialis.delete(0, END)

        self.txtSpesialis.insert(END, dokter.spesialis)

        self.txtTempatBertugas.delete(0, END)

        self.txtTempatBertugas.insert(END, dokter.tempat\_bertugas)

        self.btnSimpan.config(text="Update")

    def onSimpan(self, event=None):

        nip = self.txtNIP.get()

        nama = self.txtNama.get()

        jk = self.txtJK.get()

        spesialis = self.txtSpesialis.get()

        tempat\_bertugas = self.txtTempatBertugas.get()

        dokter = Dokter()

        dokter.nip = nip

        dokter.nama = nama

        dokter.jk = jk

        dokter.spesialis = spesialis

        dokter.tempat\_bertugas = tempat\_bertugas

        if(self.ditemukan == True):

            res = dokter.updateByNIP(nip)

            ket = 'Diperbarui'

        else:

            res = dokter.simpan()

            ket = 'Disimpan'

        rec = dokter.affected

        if(rec > 0):

            messagebox.showinfo("showinfo", "Data Berhasil "+ket)

        else:

            messagebox.showwarning("showwarning", "Data Gagal "+ket)

        self.onClear()

        return rec

    def onDelete(self, event=None):

        nip = self.txtNIP.get()

        dokter = Dokter()

        dokter.nip = nip

        if(self.ditemukan == True):

            res = dokter.deleteByNIP(nip)

            rec = dokter.affected

        else:

            messagebox.showinfo("showinfo", "Data harus ditemukan dulu sebelum dihapus")

            rec = 0

        if(rec > 0):

            messagebox.showinfo("showinfo", "Data Berhasil dihapus")

        self.onClear()

    def onKeluar(self, event=None):

        # memberikan perintah menutup aplikasi

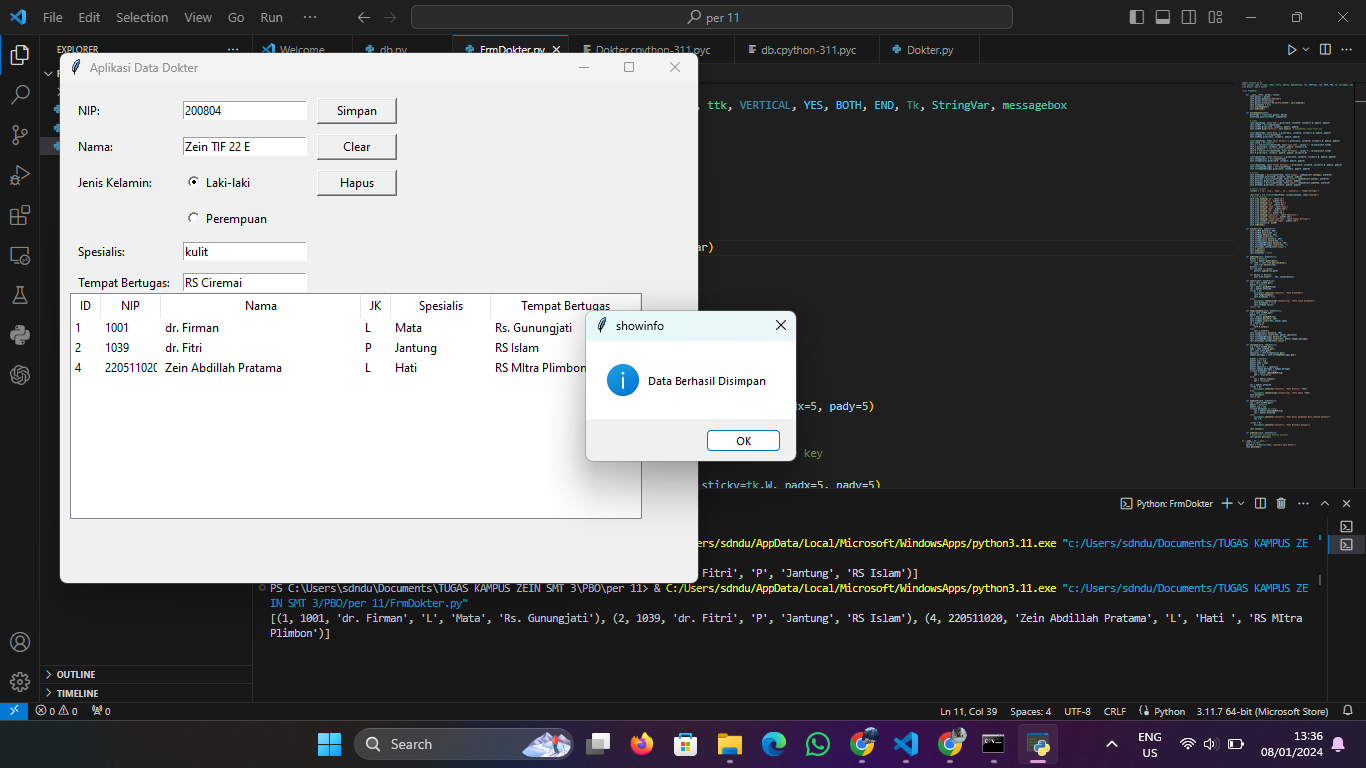
        self.parent.destroy()

if \_\_name\_\_ == '\_\_main\_\_':

    root = tk.Tk()

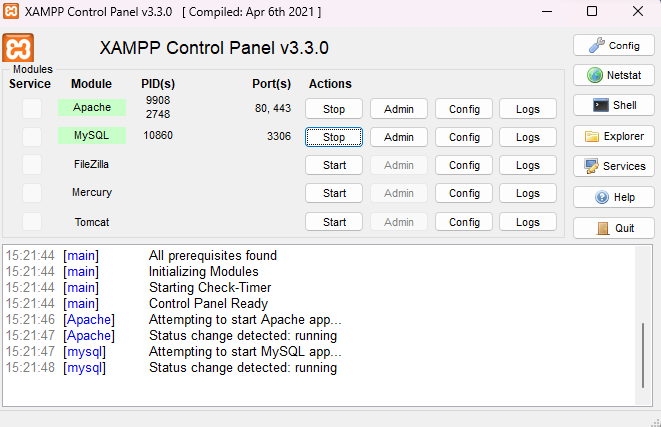
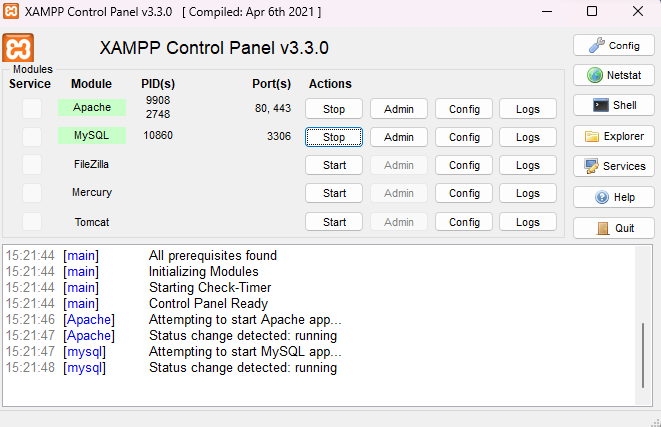
    aplikasi = FrmDokter(root, "Aplikasi Data Dokter")

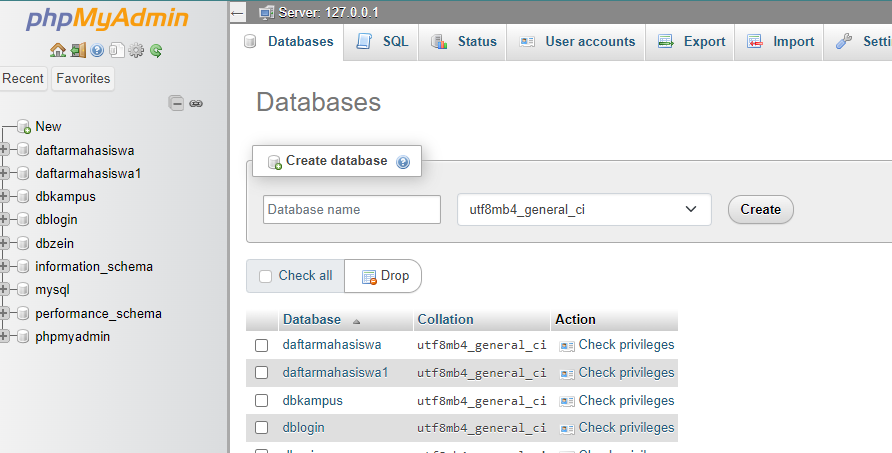
    root.mainloop()

1. ****Setelah server sudah terkoneksi dan semua codingan di masukan di aplikasi vscode lalu kita running, dan hasil running nya adalah sebagai berikut ini :

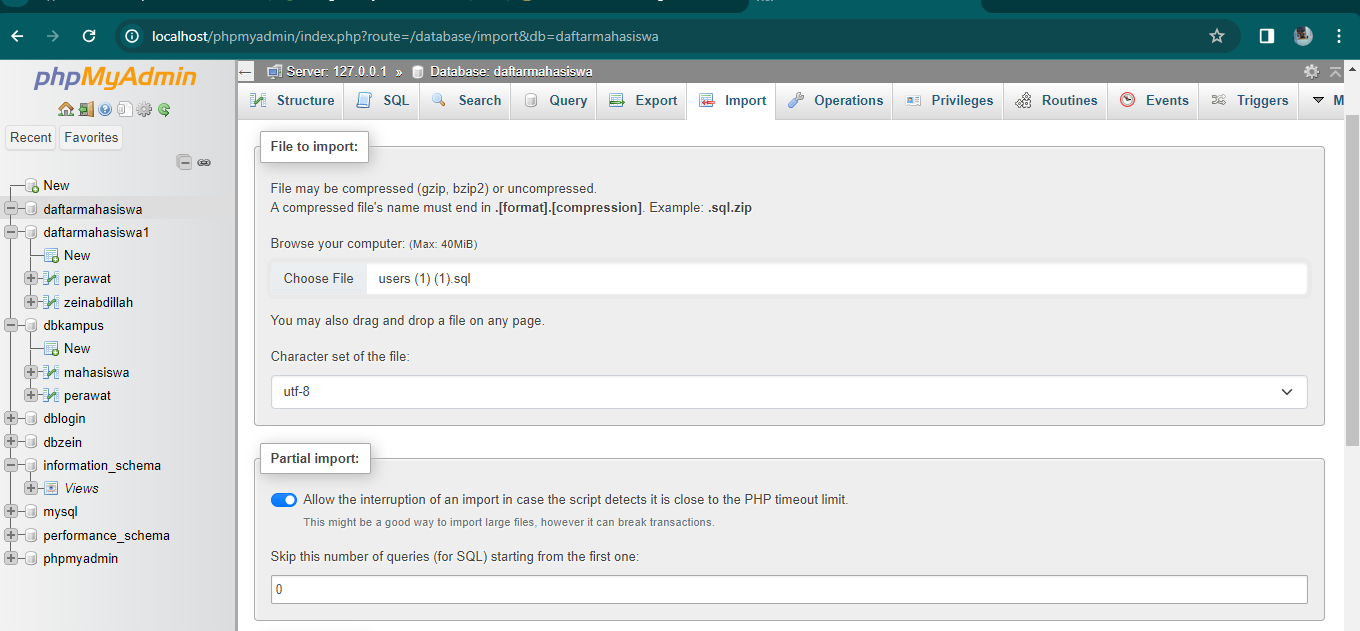
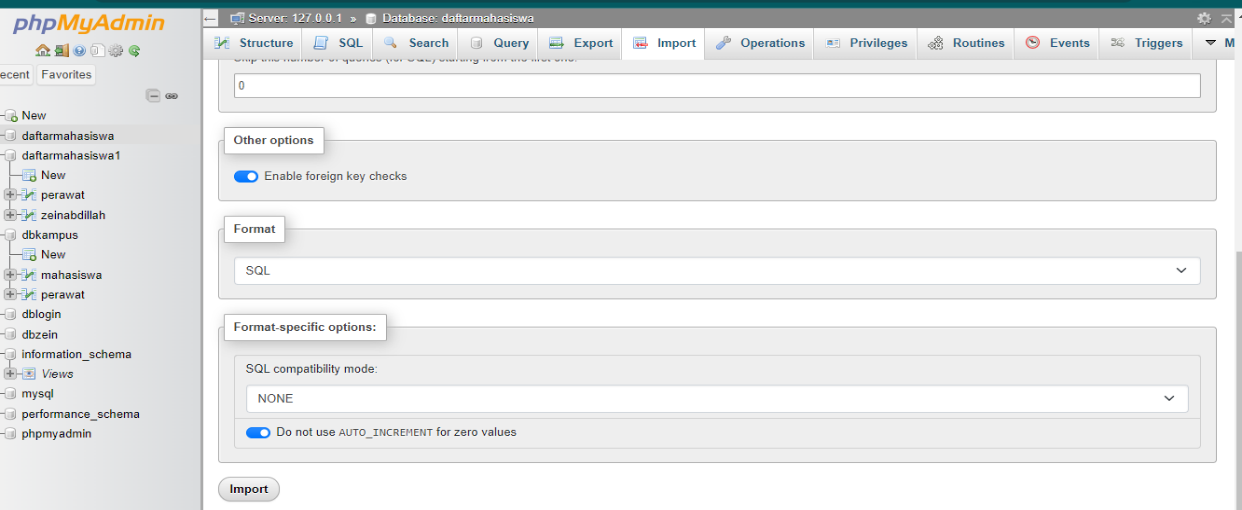
# Pertemuan 13

## 11.1 Membuat aplikasi login dalam python database

1. Pertama kita buka aplikasi Xamp control panel lalu klik/start apache & MySqlSetelah klik/star apache & Mysql lalu klik admin yang di samping mysql
2. Setelah klik admin pasti tampilan nya seoerti di bawah ini



Setelah seperti itu lalu membuat database baru, klik new lalu masukan database lalu create

1. Setelah create langsung saja kita import sql nya
2. Setelah behasil import kita membuat program di VsCode dengan nama file **DB.py**, setelah membuat nama file Db.py kita masukan codingan sebagai berikut:

import mysql.connector as mc

class DBConnection:

    def \_\_init\_\_(self):

        self.host = "localhost"

        self.port = 3306

        self.name = "dbzein"

        self.user = "root"

        self.password = ""

        self.conn = None

        self.cursor = None

        self.result = None

        self.connected = False

        self.affected = 0

        self.connect()

    @property

    def connection\_status(self):

        return self.connected

    def connect(self):

        try:

            self.conn = mc.connect(host = self.host,

                                    port = self.port,

                                    database = self.name,

                                    user = self.user,

                                    password = self.password)

            self.connected = True

            self.cursor=self.conn.cursor()

        except mc.Error as e:

            self.connected = False

        return self.conn

    def disconnect(self):

        if(self.connected==True):

            self.conn.close

        else:

            self.conn = None

    def findOne(self, sql):

        self.connect()

        self.cursor.execute(sql)

        self.result = self.cursor.fetchone()

        return self.result

    def findAll(self, sql):

        self.connect()

        self.result = self.cursor.execute(sql)

        self.result = self.cursor.fetchall()

        return self.result

    def insert(self, sql):

        self.connect()

        self.cursor.execute(sql)

        self.conn.commit()

        self.affected = self.cursor.rowcount

        return self.affected

    def update(self, sql, val):

        self.connect()

        self.cursor.execute(sql, val)

        self.conn.commit()

        self.affected = self.cursor.rowcount

        return self.affected

    def delete(self, sql):

        self.connect()

        self.cursor.execute(sql)

        self.conn.commit()

        self.affected = self.cursor.rowcount

        return self.affected

    def show(self, sql):

        self.connect()

        self.cursor.execute(sql)

        self.result = self.cursor.fetchone()

        return self.result

    @property

    def info(self):

        if(self.connected==True):

            return "Server is running on " + self.host + ' using port ' + str(self.port)

        else:

            return "Server is offline."

# test koneksi databse

A = DBConnection()

B = A.info

print(B)

# Tampilkan Data

# A = Mahasiswa()

# B = A.getAllData()

# print(B)

# # Entry Data

# A Mahasiswa()

# A.nim "4478"

# A.nama = "Wahid"

# A.jk ="L"

# A.kode\_prodi="IND"

# A.simpan()

# B = A.getAllData()

# print(B)

# # update Data

# A = Mahasiswa()

# nim "4478"

# A.nama = "Wahid Dul Hamid"

# A.jk ="L"

# A.kode\_prodi="TIF"

# A.updateByNIM(nim)

# B = A.getAllData()

# print(B)

#delete Data

# A = Mahasiswa()

# nim = "4478"

# A.deleteByNIM(nim)

# B = A.getAllData()

# print(B)

1. Setelah itu buat file dengan nama **users.py**, lalu masukan codingan sebagai berikut :

import bcrypt

from db import DBConnection as mydb

class Users:

    def \_\_init\_\_(self):

        self.\_\_id= None

        self.\_\_email= None

        self.\_\_nama= None

        self.\_\_password= None

        self.\_\_level= None

        self.\_\_uservalid = None

        self.\_\_passwordvalid = None

        self.\_\_loginvalid = None#

        self.conn = None

        self.affected = None

        self.result = None

    @property

    def id(self):

        return self.\_\_id

    @property

    def email(self):

        return self.\_\_email

    @email.setter

    def email(self, value):

        self.\_\_email = value

    @property

    def nama(self):

        return self.\_\_nama

    @nama.setter

    def nama(self, value):

        self.\_\_nama = value

    @property

    def password(self):

        return self.\_\_password

    @password.setter

    def password(self, value):

        self.\_\_password = value

    @property

    def level(self):

        return self.\_\_level

    @level.setter

    def level(self, value):

        self.\_\_level = value

    @property

    def loginvalid(self):

        return self.\_\_loginvalid

    @loginvalid.setter

    def loginvalid(self, value):

        self.\_\_loginvalid = value

    def cekUsername(self, email):

        self.conn = mydb()

        sql="SELECT \* FROM users WHERE email='" + email + "'"

        self.result = self.conn.findOne(sql)

        if(self.result!=None):

            self.\_\_email = self.result[1]

            self.\_\_nama = self.result[2]

            self.\_\_password = self.result[3]

            self.\_\_level = self.result[4]

            self.affected = self.conn.cursor.rowcount

            self.\_\_uservalid = True

        else:

            self.\_\_email = ''

            self.\_\_nama = ''

            self.\_\_password = ''

            self.\_\_level = ''

            self.affected = 0

            self.\_\_uservalid = False

        return self.\_\_uservalid

    def cekPassword(self, password):

        hashedpass=self.\_\_password.encode('utf-8')

        c = password.encode('utf-8')

        d = bcrypt.checkpw(c, hashedpass)

        if(d):

            self.\_\_passwordvalid=True

        else:

            self.\_\_passwordvalid=False

        return self.\_\_passwordvalid

    def Validasi(self, email, password):

        a = self.cekUsername(email)

        if(a==True):

            b = self.cekPassword(password)

            if(b==True):

                self.\_\_loginvalid=True

            else:

                self.\_\_loginvalid=False

        else:

            self.\_\_loginvalid=False

        val = []

        val = [self.\_\_level, self.\_\_loginvalid]

        return val

A = Users()

print("\n\n")

print("Username BENAR, dan password BENAR")

B = A.Validasi('nurjati@umc.ac.id','123')

print(B)

print("\n\nUsername BENAR, dan password SALAH")

C = A.Validasi('nurjati@umc.ac.id','1234')

print(C)

print("\n\nUsername SALAH, dan password BENAR")

D = A.Validasi('baim@umc.ac.id','123')

print(D)

print("\n\nUsername SALAH, dan password SALAH")

E = A.Validasi('baim@umc.ac.id','1234')

print(E)

print("\n\n")

1. Setelah itu kita buat file **FrmLogin.py** dengan codingan sebagai berikut :

import tkinter as tk

from tkinter import Frame,Label,Entry,Button,Radiobutton,ttk,VERTICAL,YES,BOTH,END,Tk,W,StringVar,messagebox

from tkinter import ttk

from Users import \*

class FormLogin:

    def \_\_init\_\_(self, parent, title, update\_main\_window):

        self.parent = parent

        self.update\_main\_window = update\_main\_window

        #self.parent.geometry("250x150")

        self.parent.title(title)

        self.parent.protocol("WM\_DELETE\_WINDOW", self.onKeluar)

        self.aturKomponen()

    def aturKomponen(self):

        mainFrame = Frame(self.parent, bd=10)

        mainFrame.pack(fill=BOTH, expand=YES)

        # diatur agar tampil di tengah layar

        screen\_width = mainFrame.winfo\_screenwidth()

        screen\_height = mainFrame.winfo\_screenheight()

        x = (screen\_width - 250) // 2

        y = (screen\_height - 150) // 2

        width = 250

        height = 150

        self.parent.geometry(f"{width}x{height}+{x}+{y}")

        # Label

        Label(mainFrame, text='Email:').grid(row=0, column=0, sticky=W, padx=5, pady=5)

        self.txtEmail = Entry(mainFrame)

        self.txtEmail.grid(row=0, column=1, padx=5, pady=5)

        Label(mainFrame, text='Password:').grid(row=1, column=0, sticky=W, padx=5, pady=5)

        self.txtPassword = Entry(mainFrame)

        self.txtPassword.grid(row=1, column=1, padx=5, pady=5)

        # Button

        self.btnSubmit = Button(mainFrame, text='Submit', command=self.onSubmit, width=10)

        self.btnSubmit.grid(row=2, column=0, padx=5, pady=5)

        self.btnCancel = Button(mainFrame, text='Cancel', command=self.onKeluar, width=10)

        self.btnCancel.grid(row=2, column=1, padx=5, pady=5)

    def onSubmit(self, event=None):

        email = self.txtEmail.get()

        password = self.txtPassword.get()

        obj = Users()

        val = obj.Validasi(email,password)

        C = val[1]

        if(C==True):

            self.update\_main\_window(val)

            self.parent.destroy()

        else:

            messagebox.showwarning("showwarning", "Login Gagal ")

        return val

    def onKeluar(self, event=None):

        # memberikan perintah menutup aplikasi

        self.parent.destroy()

if \_\_name\_\_ == '\_\_main\_\_':

    def update\_main\_window(result):

        print(result)

    root = tk.Tk()

    aplikasi = FormLogin(root, "Aplikasi Data Login",update\_main\_window)

    root.mainloop()

1. Lalu membuat file **Dashboard.py**, dengan codingan sebagai berikut :

import tkinter as tk

from tkinter import Menu, messagebox

from FrmLogin import \*

from FrmPersegi import \*

from FrmSegitiga import \*

from FrmLingkaran import \*

class Dashboard:

    def \_\_init\_\_(self):

        # root window

        self.root = tk.Tk()

        self.root.title('Menu Demo')

        # self.root.attributes('-fullscreen', True)

        self.root.geometry("900x400")

        self.\_\_data = None

        self.\_\_level = None

        # create a menubar

        self.menubar = Menu(self.root)

        self.root.config(menu=self.menubar)

        # create menus

        self.file\_menu = Menu(self.menubar)

        self.guest\_menu = Menu(self.menubar)

        self.admin\_menu = Menu(self.menubar)

        self.mahasiswa\_menu = Menu(self.menubar)

        self.dosen\_menu = Menu(self.menubar)

        # add menu items to File menu

        self.file\_menu.add\_command(label='Login', command=lambda: self.new\_window("Log Me In", FormLogin))

        self.file\_menu.add\_command(label='Exit', command=self.root.destroy)

        # add menu items to menu Admin

        self.admin\_menu.add\_command(label='Admin-1', command=lambda: self.new\_window("Luas Persegi", FrmPersegi))

        self.admin\_menu.add\_command(label='Admin-2', command=lambda: self.new\_window("Luas Segitiga", FrmSegitiga))

        self.admin\_menu.add\_command(label='Admin-3', command=lambda: self.new\_window("Luas Lingkaran", FrmLingkaran))

        # add menu items to menu Mahasiswa

        self.mahasiswa\_menu.add\_command(label='Mahasiswa-1', command=lambda: self.new\_window("Luas Persegi", FrmPersegi))

        self.mahasiswa\_menu.add\_command(label='Mahasiswa-2', command=lambda: self.new\_window("Luas Segitiga", FrmSegitiga))

        self.mahasiswa\_menu.add\_command(label='Mahasiswa-3', command=lambda: self.new\_window("Luas Lingkaran", FrmLingkaran))

        # add menu items to menu Dosen

        self.dosen\_menu.add\_command(label='Dosen-1', command=lambda: self.new\_window("Luas Persegi", FrmPersegi))

        self.dosen\_menu.add\_command(label='Dosen-2', command=lambda: self.new\_window("Luas Segitiga", FrmSegitiga))

        self.dosen\_menu.add\_command(label='Dosen-3', command=lambda: self.new\_window("Luas Lingkaran", FrmLingkaran))

        # add menus to the menubar

        self.menubar.add\_cascade(label="File", menu=self.file\_menu)

    def new\_window(self, number, \_class):

        new = tk.Toplevel(self.root)

        new.transient()

        new.grab\_set()

        \_class(new, number, self.update\_main\_window)

    def update\_main\_window(self, data):

        # Method to receive data from child windows

        self.\_\_data = data

        level = self.\_\_data[0]

        loginvalid = self.\_\_data[1]

        if(loginvalid==True):

            index = self.file\_menu.index('Login')

            # hapus menu login

            self.file\_menu.delete(index)

            self.file\_menu.add\_command(label='Logout', command=self.Logout)

            # tambahkan menu sesuai level

            if(level=='admin'):

                self.menubar.add\_cascade(label="Admin", menu=self.admin\_menu)

                self.\_\_level = 'Admin'

            elif(level=='mahasiswa'):

                self.menubar.add\_cascade(label="Mahasiswa", menu=self.mahasiswa\_menu)

                self.\_\_level = 'Mahasiswa'

            elif(level=='dosen'):

                self.menubar.add\_cascade(label="Dosen", menu=self.dosen\_menu)

                self.\_\_level = 'Dosen'

            else:

                pass

    def Logout(self):

        index = self.file\_menu.index('Logout')

        self.file\_menu.delete(index)

        self.file\_menu.add\_command(label='Login', command=lambda: self.new\_window("Log Me In", FormLogin))

        self.remove\_all\_menus()

    def remove\_all\_menus(self):

        index = self.menubar.index(self.\_\_level)

        if index is not None:

            self.menubar.delete(index)

    def run(self):

        self.root.mainloop()

if \_\_name\_\_ == "\_\_main\_\_":

    menu\_app = Dashboard()

    menu\_app.run()

1. Setelah itu buat file dengan nama **FrmLingkaran.py**, dengan codingan sebagai berikut :

from tkinter import Frame,Label,Entry,Button,YES,BOTH,END,Tk,W

class FrmLingkaran:

    def \_\_init\_\_(self, parent, title, update\_main\_window):

        self.parent = parent

        self.update\_main\_window = update\_main\_window

        #self.parent.geometry("400x200")

        self.parent.title(title)

        self.parent.protocol("WM\_DELETE\_WINDOW", self.onKeluar)

        self.aturKomponen()

    def aturKomponen(self):

        mainFrame = Frame(self.parent, bd=10)

        mainFrame.pack(fill=BOTH, expand=YES)

        # pasang Label

        Label(mainFrame, text='Jari-jari:').grid(row=0, column=0,

            sticky=W, padx=5, pady=5)

        Label(mainFrame, text="Luas:").grid(row=3, column=0,

            sticky=W, padx=5, pady=5)

        # pasang textbox

        self.txtJarijari = Entry(mainFrame)

        self.txtJarijari.grid(row=0, column=1, padx=5, pady=5)

        self.txtLuas = Entry(mainFrame)

        self.txtLuas.grid(row=3, column=1, padx=5, pady=5)

        # Pasang Button

        self.btnHitung = Button(mainFrame, text='Hitung',

            command=self.onHitung)

        self.btnHitung.grid(row=2, column=1, padx=5, pady=5)

    # fungsi "onHitung" berfungsi untuk menghitung luas persegi panjang

    def onHitung(self, event=None):

        jarijari = int(self.txtJarijari.get())

        luas = 22/7 \* jarijari\*\*2

        self.txtLuas.delete(0,END)

        self.txtLuas.insert(END,str(luas))

    def onKeluar(self, event=None):

        # memberikan perintah menutup aplikasi

        self.parent.destroy()

if \_\_name\_\_ == '\_\_main\_\_':

    def update\_main\_window(result):

        print(result)

    root = Tk()

    aplikasi = FrmLingkaran(root, "Program Luas Lingkaran")

    root.mainloop()

1. Setelah itu buat file baru dengan nama **FrmPersegi.py** dengan codingan sebagai berikut :

from tkinter import Frame,Label,Entry,Button,YES,BOTH,END,Tk,W

class FrmPersegi:

    def \_\_init\_\_(self, parent, title, update\_main\_window):

        self.parent = parent

        self.update\_main\_window = update\_main\_window

        #self.parent.geometry("400x200")

        self.parent.title(title)

        self.parent.protocol("WM\_DELETE\_WINDOW", self.onKeluar)

        self.aturKomponen()

    def aturKomponen(self):

        mainFrame = Frame(self.parent, bd=10)

        mainFrame.pack(fill=BOTH, expand=YES)

        # pasang Label

        Label(mainFrame, text='Panjang:').grid(row=0, column=0,

            sticky=W, padx=5, pady=5)

        Label(mainFrame, text="Lebar:").grid(row=1, column=0,

            sticky=W, padx=5, pady=5)

        Label(mainFrame, text="Luas:").grid(row=3, column=0,

            sticky=W, padx=5, pady=5)

        # pasang textbox

        self.txtPanjang = Entry(mainFrame)

        self.txtPanjang.grid(row=0, column=1, padx=5, pady=5)

        self.txtLebar = Entry(mainFrame)

        self.txtLebar.grid(row=1, column=1, padx=5, pady=5)

        self.txtLuas = Entry(mainFrame)

        self.txtLuas.grid(row=3, column=1, padx=5, pady=5)

        # Pasang Button

        self.btnHitung = Button(mainFrame, text='Hitung',

            command=self.onHitung)

        self.btnHitung.grid(row=2, column=1, padx=5, pady=5)

    # fungsi "onHitung" berfungsi untuk menghitung luas persegi panjang

    def onHitung(self, event=None):

        panjang = int(self.txtPanjang.get())

        lebar = int(self.txtLebar.get())

        luas = panjang \* lebar

        self.txtLuas.delete(0,END)

        self.txtLuas.insert(END,str(luas))

        self.update\_main\_window(str(luas))

    def onKeluar(self, event=None):

        # memberikan perintah menutup aplikasi

        self.parent.destroy()

if \_\_name\_\_ == '\_\_main\_\_':

    def update\_main\_window(result):

        print(result)

    root = Tk()

    aplikasi = FrmPersegi(root, "Program Luas Persegi Panjang")

    root.mainloop()

1. Setelah itu buat file baru dengan nama **FrmSegitiga.py** dengan codingan sebagai berikut :

from tkinter import Frame,Label,Entry,Button,YES,BOTH,END,Tk,W

class FrmSegitiga:

    def \_\_init\_\_(self, parent, title, update\_main\_window):

        self.parent = parent

        self.update\_main\_window = update\_main\_window

        #self.parent.geometry("400x200")

        self.parent.title(title)

        self.parent.protocol("WM\_DELETE\_WINDOW", self.onKeluar)

        self.aturKomponen()

    def aturKomponen(self):

        mainFrame = Frame(self.parent, bd=10)

        mainFrame.pack(fill=BOTH, expand=YES)

        # pasang Label

        Label(mainFrame, text='Lebar Alas:').grid(row=0, column=0,

            sticky=W, padx=5, pady=5)

        Label(mainFrame, text="Tinggi:").grid(row=1, column=0,

            sticky=W, padx=5, pady=5)

        Label(mainFrame, text="Luas Segitiga:").grid(row=3, column=0,

            sticky=W, padx=5, pady=5)

        # pasang textbox

        self.txtAlas = Entry(mainFrame)

        self.txtAlas.grid(row=0, column=1, padx=5, pady=5)

        self.txtTinggi = Entry(mainFrame)

        self.txtTinggi.grid(row=1, column=1, padx=5, pady=5)

        self.txtLuas = Entry(mainFrame)

        self.txtLuas.grid(row=3, column=1, padx=5, pady=5)

        # Pasang Button

        self.btnHitung = Button(mainFrame, text='Hitung',

            command=self.onHitung)

        self.btnHitung.grid(row=2, column=1, padx=5, pady=5)

    # fungsi "onHitung" berfungsi untuk menghitung luas persegi panjang

    def onHitung(self, event=None):

        alas = int(self.txtAlas.get())

        tinggi = int(self.txtTinggi.get())

        luas = 0.5 \* alas \* tinggi

        self.txtLuas.delete(0,END)

        self.txtLuas.insert(END,str(luas))

    def onKeluar(self, event=None):

        # memberikan perintah menutup aplikasi

        self.parent.destroy()

if \_\_name\_\_ == '\_\_main\_\_':

    def update\_main\_window(result):

        print(result)

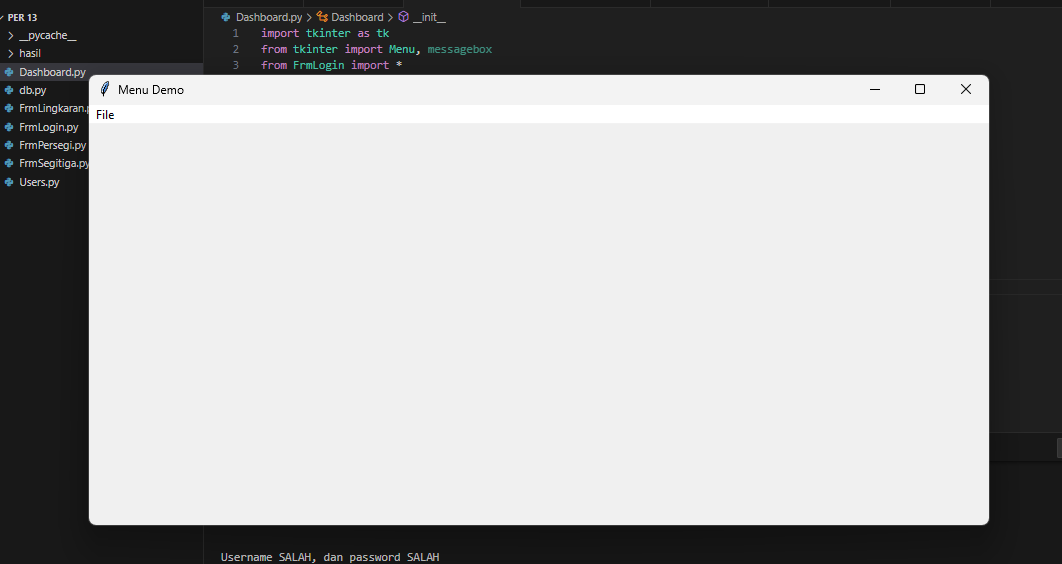
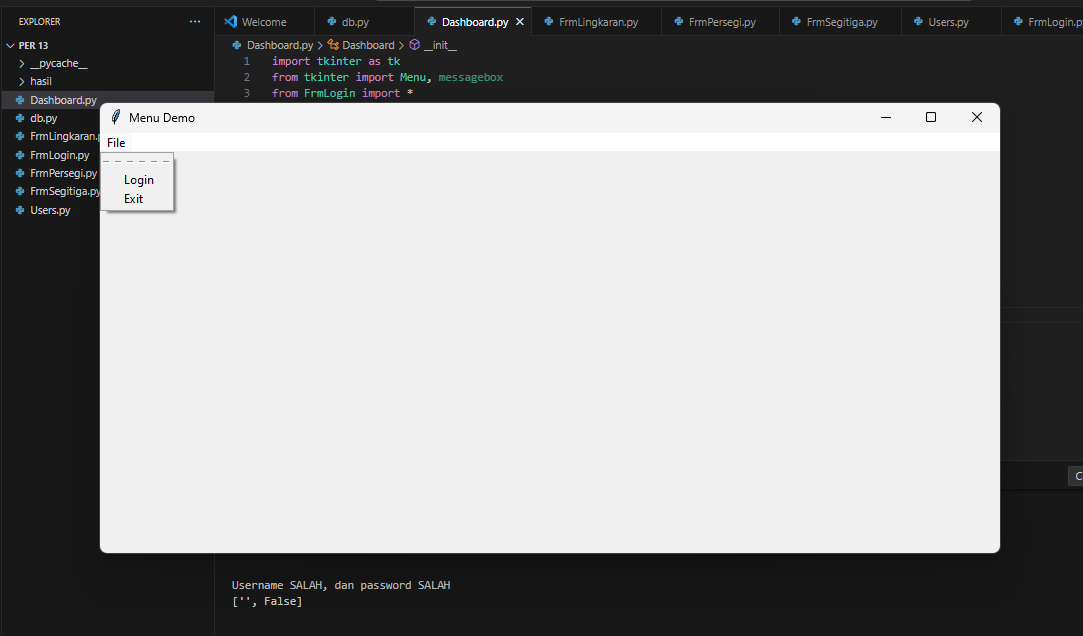
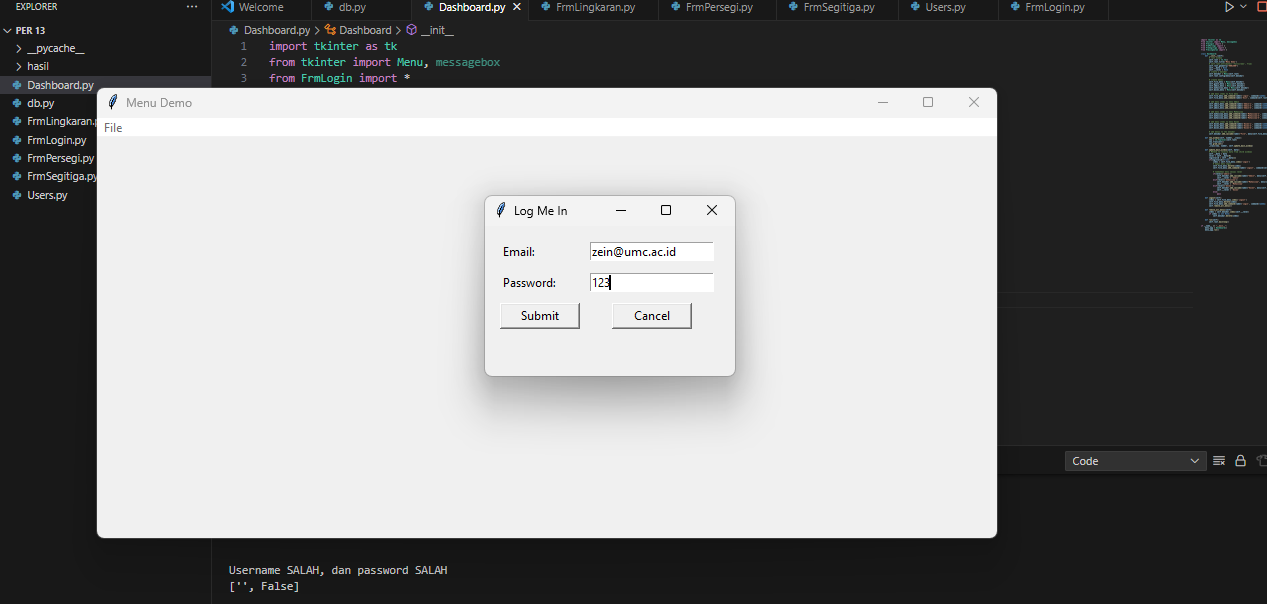
    root = Tk()

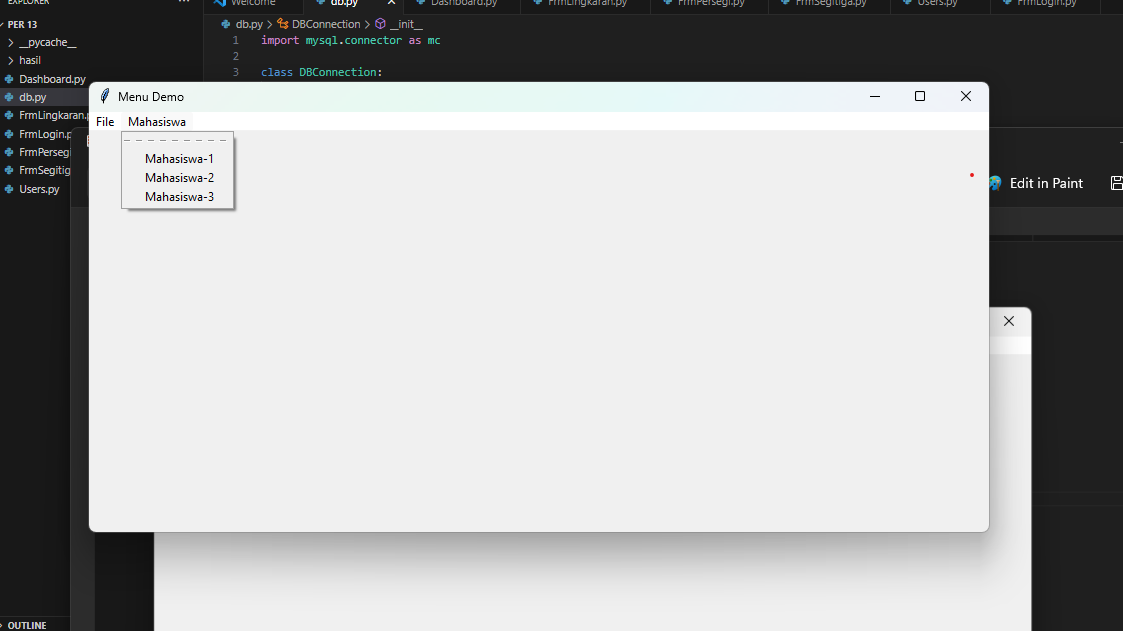
    aplikasi = FrmSegitiga(root, "Program Luas Persegi Panjang")

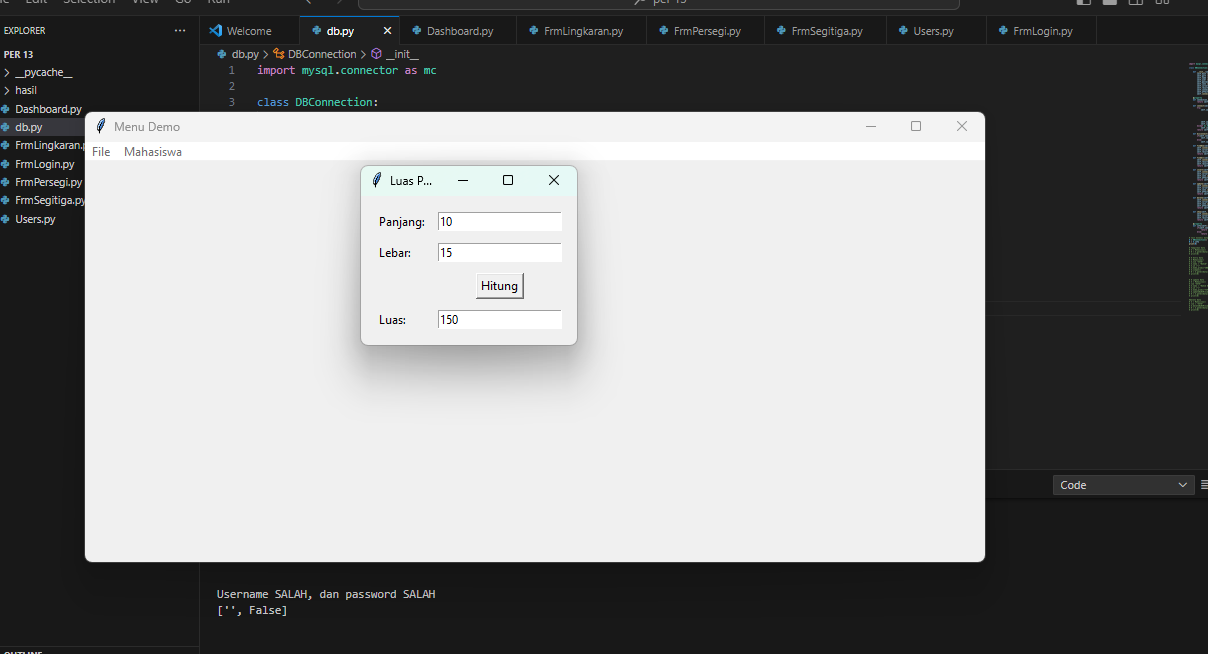
    root.mainloop()

Pastikan semua program sudah di massukan semua ke dalam vscode

1. Setelah kita masukan semua codingan lalu kita run dengan langkah sebagai berrikut :

Run di bagian dashboard.py



Ikuti run sesuai dengan urutan yang di atas