

# KATA PENGANTAR

Puji syukur kami panjatkan kehadirat Allah SWT yang telah melimpahkan rahmat, taufiq, dan hidayah-Nya sehingga kami dapat menyelesaikan penyusunan modul ini. Modul ini disusun sebagai bagian dari tugas mata kuliah Pemrograman Berorientasi Objek (PBO) dengan dosem pengampu Bapak Freddy Wicaksono, M.Kom.

Dalam era di mana teknologi informasi menjadi tulang punggung dari berbagai bidang kehidupan, pemahaman tentang konsep dan praktik Pemrograman Berorientasi Objek, khususnya dengan bahasa pemrograman Python, menjadi sangat penting. Python telah menjadi salah satu bahasa pemrograman yang paling populer dan digunakan secara luas dalam berbagai konteks, mulai dari pengembangan perangkat lunak, ilmu data, hingga kecerdasan buatan.

Melalui modul ini, kami berusaha menyajikan materi-materi yang relevan dan bermanfaat dalam memahami dasar-dasar Pemrograman Berorientasi Objek menggunakan bahasa pemrograman Python. Kami berharap modul ini dapat menjadi panduan yang membantu dalam proses pembelajaran, serta membantu mahasiswa untuk memperoleh pemahaman yang kokoh dan mampu mengimplementasikan konsep-konsep tersebut dalam pengembangan perangkat lunak.

Kami mengucapkan terima kasih yang sebesar-besarnya kepada Bapak Freddy Wicaksono, M.Kom, selaku dosen pengampu mata kuliah ini, atas bimbingan, dukungan, dan pengarahannya dalam penyusunan modul ini. Tak lupa pula kepada semua pihak yang telah berkontribusi dalam proses penyusunan modul ini.

Semoga modul ini dapat memberikan manfaat yang maksimal bagi proses pembelajaran kita semua. Kritik dan saran yang membangun sangat kami harapkan guna perbaikan dan penyempurnaan di masa yang akan datang.

Wassalamu'alaikum Wr. Wb.

**Cirebon, 28 Januari 2024**

# DAFTAR ISI

[KATA PENGANTAR i](#_Toc157340582)

[DAFTAR ISI ii](#_Toc157340583)

[PERTEMUAN 1 PENGENALAN DASAR PYTHON (VARIABEL, TIPE DATA, OPERATOR) & LATIHAN PRAKTIKUM 1](#_Toc157340584)

[1.1. Pengertian Python 1](#_Toc157340585)

[1.2. Variabel & Tipe Data 1](#_Toc157340586)

[1.3. Operator Dasar 2](#_Toc157340587)

[1.4. Instalasi Python 3](#_Toc157340588)

[1.5. Latihan Praktikum Aplikasi Bangun Ruang 4](#_Toc157340589)

[PERTEMUAN 2 IF & Loop 6](#_Toc157340590)

[2.1 Pengenalan If 6](#_Toc157340591)

[2.2 Pengertian Loop 7](#_Toc157340592)

[PERTEMUAN 3 FUNCTION IN PYTHON & TKINTER & LATIHAN PRAKTIKUM 9](#_Toc157340593)

[3.1 Pengenalan Tkinter 9](#_Toc157340594)

[3.2 Fungsi 9](#_Toc157340595)

[3.3 Mendefinisikan dan Memanggil Fungsi 10](#_Toc157340596)

[3.4 Latihan Praktikum Luas dan Volume Bangun Ruang menggunakan Tkinter 11](#_Toc157340597)

[PERTEMUAN 4 PYTHON MODULE,FILE (I/O) & LATIHAN PRAKTIKUM 15](#_Toc157340598)

[4.1 Pengertian Modul dalam Python 15](#_Toc157340599)

[4.2 Menggunakan Modul 15](#_Toc157340600)

[4.3 Pengenalan File Input/Output (I/O) 15](#_Toc157340601)

[4.4 Operasi File 16](#_Toc157340602)

[4.5 Latihan Praktikum membuat entry jadwal kuliah menggunakan Tkinter 16](#_Toc157340603)

[PERTEMUAN 6 MULTIFORM & LATIHAN PRAKTIKUM 20](#_Toc157340604)

[6.1 Pengertian Multiform 20](#_Toc157340605)

[6.2 Penerapan Multiform Dalam Python 20](#_Toc157340606)

[6.3 Latihan Praktikum Membuat Multiform Tkinter 20](#_Toc157340607)

[PERTEMUAN 7: APLIKASI BANTU (TOOLS) & LATIHAN PRAKTIKUM 25](#_Toc157340608)

[7.1 Pengertian Aplikasi Bantu (Tools): 25](#_Toc157340609)

[7.2 Pembuatan Aplikasi Bantu (Tools): 25](#_Toc157340610)

[7.3 Latihan Praktikum Membuat App Mp3, Mp4, Pic To Teks Dan Text To Sound: 25](#_Toc157340611)

[PERTEMUAN 9: KONSEP DASAR CLASS DAN OBJEK & LATIHAN PRAKTIKUM 35](#_Toc157340612)

[9.1 Pengertian Konsep Dasar Class Dan Objek & Latihan Praktikum 35](#_Toc157340613)

[PERTEMUAN 10: PENGETIAN MYSQL & LATIHAN PRAKTIKUM 43](#_Toc157340614)

[10.1 Pengertian Basis Data, MySQL, Xampp 43](#_Toc157340615)

[10.2 Instalasi Xampp 43](#_Toc157340616)

[10.3 Latihan Praktikum membuat tabel mysql menggunakan CMD 44](#_Toc157340617)

[PERTEMUAN 11: PYTHON DATABASE & LATIHAN PRAKTIKUM 48](#_Toc157340618)

[11.1. Membuat Database pada phpMyAdmin 48](#_Toc157340619)

[11.2. Mengimpor Tabel SQL ke dalam phpMyAdmin 48](#_Toc157340620)

[11.3. koneksi python dan mysql 49](#_Toc157340621)

[11.4. Membuat fungsi aplikasi data mahasiswa 51](#_Toc157340622)

[11.5. Form aplikasi data mahasiswa 54](#_Toc157340623)

[PERTEMUAN 12: PYTHON APPLICATION DATABASE SERVER & LATIHAN PRAKTIKUM 60](#_Toc157340624)

[12.1. Pengembangan Aplikasi Data Mahasiswa Berbasis Database Online Menggunakan Freedb 60](#_Toc157340625)

[12.2. Latihan Praktikum Praktikum aplikasi data mahasiswa database menggunakan freedb 60](#_Toc157340626)

[12.3. Koneksi Python Dan Mysql Dabase Server Online 62](#_Toc157340627)

[PERTEMUAN 13: PYTHON APPLICATION DATABASE SERVER & LATIHAN PRAKTIKUM 65](#_Toc157340628)

[13.1. Pengertian Aplikasi Python Login Multi User Berbasis Tkinter Mysql 65](#_Toc157340629)

[13.2. Membuat Database Login pada phpMyAdmin 65](#_Toc157340630)

[13.3 Mengimport Tabel SQL ke dalam phpMyAdmin 65](#_Toc157340631)

[13.4 koneksi python dan mysql 66](#_Toc157340632)

[13.5 Membuat Fungsi User Login aplikasi bangun datar 68](#_Toc157340633)

[13.6 Form Login Mahasiswa 71](#_Toc157340634)

[13.7. Form Bangun Datar Persegi, Segitiga Dan Lingkaran 73](#_Toc157340635)

[13.8 Dashboard 76](#_Toc157340636)

# PERTEMUAN 1 PENGENALAN DASAR PYTHON (VARIABEL, TIPE DATA, OPERATOR) & LATIHAN PRAKTIKUM

## Pengertian Python

Python adalah bahasa pemrograman interpretatif multiguna. Tidak seperti bahasa lain yang susah untuk dibaca dan dipahami, python lebih menekankan pada keterbacaan kode agar lebih mudah untuk memahami sintaks. Hal ini membuat Python sangat mudah dipelajari baik untuk pemula maupun untuk yang sudah menguasai bahasa pemrograman lain.

Bahasa ini muncul pertama kali pada tahun 1991, dirancang oleh seorang bernama Guido van Rossum. Sampai saat ini Python masih dikembangkan oleh Python Software Foundation. Bahasa Python mendukung hampir semua sistem operasi, bahkan untuk sistem operasi Linux, hampir semua distronya sudah menyertakan Python di dalamnya.Python merupakan bahasa pemrograman komputer yang biasa dipakai untuk membangun situs, software/aplikasi, mengotomatiskan tugas dan melakukan analisis data. Bahasa pemrograman ini termasuk bahasa tujuan umum. Artinya, ia bisa digunakan untuk membuat berbagai program berbeda, bukan khusus untuk masalah tertentu saja.. Karena sifatnya yang serba guna dan mudah digunakan, ia menjadi bahasa pemrograman yang paling banyak digunakan. Terutama untuk mereka yang masih pemula.

## Variabel & Tipe Data

Variabel adalah lokasi memori yang dicadangkan untuk menyimpan nilai-nilai. Ini berarti bahwa ketika Anda membuat sebuah variabel Anda memesan beberapa ruang di memori. Variabel menyimpan data yang dilakukan selama program dieksekusi, yang nantinya isi dari variabel tersebut dapat diubah oleh operasi - operasi tertentu pada program yang menggunakan variabel.Variabel dapat menyimpan berbagai macam tipe data. Di dalam pemrograman Python

Tipe data menggambarkan jenis nilai yang dapat disimpan dan dioperasikan di dalam variabel. Berikut adalah beberapa tipe data dasar dalam Python:

* Float (float): Untuk menyimpan bilangan desimal.

Contoh : tinggi = 190

* String (str):Untuk menyimpan teks atau karakter.

Contoh : nama = "Rusli","Maman","Budi"

* Integer (int): Untuk menyimpan bilangan bulat tanpa desimal.

contoh : umur = 27

* Dictionary (dict): Menyimpan pasangan kunci-nilai.

contoh : biodata = {'nama': Rusli, 'umur': 27, 'tinggi': 190}

* Set: Kumpulan nilai unik tanpa indeks.

contoh : warna = {'kuning', 'hijau', 'merah’,’biru’,}

* Boolean (bool): untuk Menyimpan nilai kebenaran, yaitu True atau False.

contoh : benar = True, salah = False

* List: Untuk menyimpan sejumlah nilai yang dapat diubah (mutable).

contoh : angka = [1, 2, 3, 4, 5,6,7,8,9]

* Tuple: Seperti list, tetapi bersifat tidak dapat diubah (immutable).

contoh : koordinat = (7,3)

* Hexadecimal Menyatakan bilangan dalam format heksa (bilangan berbasis 16)

Contoh : 1d3

## Operator Dasar

Operator adalah simbol yang digunakan untuk melakukan operasi pada variabel dan nilai. Python mendukung berbagai macam operator, termasuk operator aritmatika, perbandingan, logika, dan lainnya. Berikut adalah beberapa operator dasar yang sering digunakan dalam Python:

1. Operator Aritmatika

Operator aritmatika digunakan untuk melakukan operasi matematika dasar. Berikut adalah operator-operator aritmatika dalam Python:

* Penjumlahan (+): Menambahkan nilai dari dua operand.
* Pengurangan (-): Mengurangkan nilai operand pertama dengan nilai operand kedua.
* Perkalian (\*): Mengalikan dua operand.
* Pembagian (/): Membagi operand pertama dengan operand kedua.
* Pembagian Bulat (//): Melakukan pembagian dan mengembalikan hasilnya dalam bentuk bilangan bulat.
* Modulus (%): Mengembalikan sisa pembagian dari operand pertama dibagi dengan operand kedua.
* Pangkat (\*\*): Menghasilkan pangkat dari operand pertama dengan eksponen operand kedua.

1. Operator Logika

Operator logika digunakan untuk menggabungkan kondisi-kondisi logika. Berikut adalah operator-operator logika dalam Python:

* Logika AND (and): Mengembalikan True jika kedua kondisi bernilai True.
* Logika OR (or): Mengembalikan True jika salah satu dari kedua kondisi bernilai True.
* Logika NOT (not): Mengembalikan nilai kebalikan dari suatu kondisi.

1. Operator Perbandingan

Operator perbandingan digunakan untuk membandingkan nilai dari dua operand. Berikut adalah operator-operator perbandingan dalam Python:

* Sama dengan (==): Mengembalikan True jika kedua operand memiliki nilai yang sama.
* Tidak sama dengan (!=): Mengembalikan True jika kedua operand memiliki nilai yang berbeda.
* Lebih besar dari (>): Mengembalikan True jika operand pertama lebih besar dari operand kedua.
* Lebih kecil dari (<): Mengembalikan True jika operand pertama lebih kecil dari operand kedua.
* Lebih besar atau sama dengan (>=): Mengembalikan True jika operand pertama lebih besar atau sama dengan operand kedua.
* Lebih kecil atau sama dengan (<=): Mengembalikan True jika operand pertama lebih kecil atau sama dengan operand kedua.

1. Penugasan: Digunakan untuk memberikan nilai ke variabel.

* Penugasan (`=``)

## Instalasi Python

Berikut merupakan cara instalasi Python pada sistem operasi windows

* Unduh Installer Python: Buka situs resmi Python di https://www.python.org/downloads/. Di halaman unduhan, pilih versi Python yang ingin Anda instal. Disarankan untuk mengunduh versi terbaru yang tersedia.
* Jalankan Installer: Setelah unduhan selesai, jalankan file installer yang telah diunduh. Pastikan untuk memilih opsi "Add Python to PATH" selama proses instalasi. Ini akan memungkinkan Anda untuk menggunakan perintah Python dari Command Prompt dengan mudah.
* Selesaikan Instalasi: Ikuti petunjuk pada layar untuk menyelesaikan proses instalasi. Setelah instalasi selesai, Python akan terinstal di komputer Anda

## Latihan Praktikum Aplikasi Bangun Ruang

berikut adalah contoh sourcode latihan praktikum untuk menghitung luas permukaan dan volume prisma segitiga dan Bola menggunakan Python:

1. Kode Program luas & volume Kerucut

from math import pi

print("Menghitung luas permukaan dan volume kerucut")

#variabel

jari\_jari = 8

tinggi = 10

garis\_pelukis = 15

#Rumus

luas\_permukaan = pi \* jari\_jari \* (jari\_jari + garis\_pelukis)

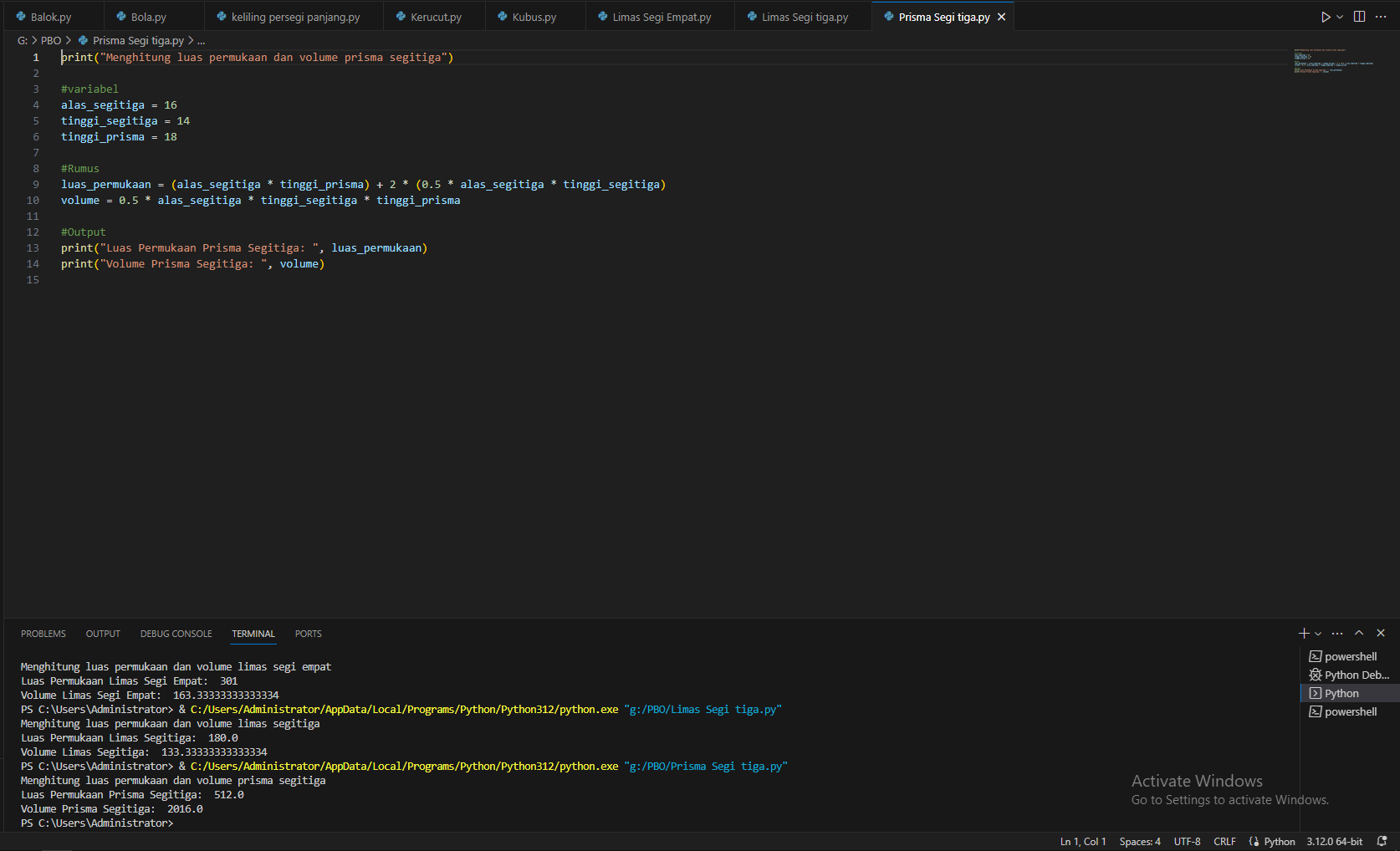
volume = (1/3) \* pi \* (jari\_jari \*\* 2) \* tinggi

#Output

print("Luas Permukaan Kerucut: ", luas\_permukaan)

print("Volume Kerucut: ", volume)

dan berikut adalah hasil dari sourcecode pengihitungan luas permukaan dan volume prisma segitiga menggunakan Python:



1. Kode Program luas & volume Bola

from math import pi

print("Menghitung Luas dan Volume Bola")

#variabel

jari = 32

#Rumus

luas   = 4\*pi\*(jari\*jari)

volume = 4/3\*pi\*(jari\*jari\*jari)

#Output

print("pi :", pi)

print("jari :", jari)

print("luas :", luas)

print("volume:", volume)

# PERTEMUAN 2 IF & Loop

## 2.1 Pengenalan If

1. Kondisi If

Pengambilan keputusan (kondisi if) digunakan untuk mengantisipasi kondisi yang terjadi saat jalanya program dan menentukan tindakan apa yang akan diambil sesuai dengan kondisi. Pada python ada beberapa statement/kondisi diantaranya adalah if, else dan elif Kondisi if digunakan untuk mengeksekusi kode jika kondisi bernilai benar True. Jika kondisi bernilai salah False maka statement/kondisi if tidak akan di-eksekusi.

Dibawah ini adalah contoh penggunaan kondisi if pada Python

umur = 18

if umur >= 18:

    print("Anda sudah dewasa")

1. Kondisi If Else

Pengambilan keputusan (kondisi if else) tidak hanya digunakan untuk menentukan tindakan apa yang akan diambil sesuai dengan kondisi, tetapi juga digunakan untuk menentukan tindakan apa yang akan diambil/dijalankan jika kondisi tidak sesuai. Pada python ada beberapa statement/kondisi diantaranya adalah if, else dan elif Kondisi if digunakan untuk mengeksekusi kode jika kondisi bernilai benar. Kondisi if else adalah kondisi dimana jika pernyataan benar True maka kode dalam if akan dieksekusi, tetapi jika bernilai salah False maka akan mengeksekusi kode di dalam else.

Dibawah ini adalah contoh penggunaan kondisi if else pada Python

umur = 12

if umur >= 18:

    print("Anda sudah dewasa")

else:

    print("Anda masih di bawah umur")

1. Kondisi Elif

Pengambilan keputusan (kondisi if elif) merupakan lanjutan/percabangan logika dari "kondisi if". Dengan elif kita bisa membuat kode program yang akan menyeleksi beberapa kemungkinan yang bisa terjadi. Hampir sama dengan kondisi "else", bedanya kondisi "elif" bisa banyak dan tidak hanya satu.

Dibawah ini adalah contoh penggunaan kondisi elif pada Python

nilai = 60

if nilai >= 85:

    print("Nilai Anda A")

elif nilai >= 75:

    print("Nilai Anda B")

elif nilai >= 65:

    print("Nilai Anda C")

else:

    print("Anda tidak lulus")

## 2.2 **Pengertian Loop**

Secara umum, pernyataan pada bahasa pemrograman akan dieksekusi secara berurutan. Pernyataan pertama dalam sebuah fungsi dijalankan pertama, diikuti oleh yang kedua, dan seterusnya. Tetapi akan ada situasi dimana Anda harus menulis banyak kode, dimana kode tersebut sangat banyak. Jika dilakukan secara manual maka Anda hanya akan membuang-buang tenaga dengan menulis beratus-ratus bahkan beribu-ribu kode. Untuk itu Anda perlu menggunakan pengulangan di dalam bahasa pemrograman Python.

1. While Loop

Pengulangan While Loop di dalam bahasa pemrograman Python dieksesusi statement berkali-kali selama kondisi bernilai benar atau True.

Dibawah ini adalah contoh penggunaan pengulangan While Loop.

angka = 1

while angka <= 5:

    print("Angka:", angka)

    angka += 1

1. For Loop

Pengulangan for pada Python memiliki kemampuan untuk mengulangi item dari urutan apapun, seperti list atau string.

Dibawah ini adalah contoh penggunaan pengulangan For Loop.

for i in range(5):

    print("Perulangan ke-", i)

# PERTEMUAN 3 FUNCTION IN PYTHON & TKINTER & LATIHAN PRAKTIKUM

## **Pengenalan Tkinter**

Tkinter adalah modul bawaan Python yang digunakan untuk membuat antarmuka grafis pengguna (GUI). Tkinter dibangun di atas toolkit Tk, yang merupakan toolkit GUI untuk bahasa pemrograman Tcl (Tool Command Language). Tkinter menyediakan alat dan widget yang memungkinkan pengembang untuk membuat aplikasi GUI dengan mudah.

Untuk dapat membuat program menggunakan Tkinter di haruskan untuk menginstall di cmd/terminal dengan perintah

* **“pip install tk”**

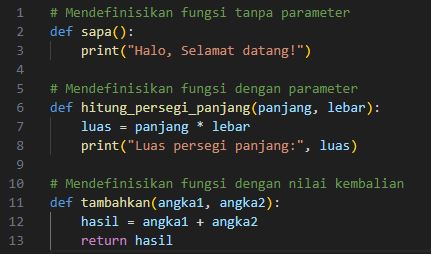
## 3.2 **Fungsi**

Fungsi adalah blok kode yang dirancang untuk melakukan tugas tertentu yang dapat dipanggil oleh bagian lain dari program. Fungsi juga memungkinkan untuk menghindari duplikasi kode, meningkatkan keterbacaan kode, dan membuat program lebih modular. Berikut adalah beberapa konsep dasar tentang fungsi:

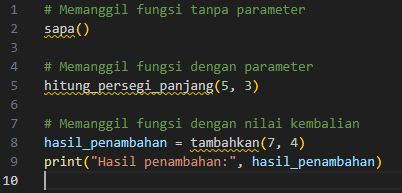
* Pendefinisian Fungsi: Fungsi didefinisikan menggunakan kata kunci 'def' diikuti oleh nama fungsi dan parameter (jika ada).
* Pendefinisian Fungsi: Fungsi didefinisikan menggunakan kata kunci 'def' diikuti oleh nama fungsi dan parameter (jika ada).
* Parameter dan Argumen: Parameter adalah variabel yang didefinisikan dalam tanda kurung saat mendefinisikan fungsi. Argumen adalah nilai yang diberikan saat memanggil fungsi.
* Nilai Kembalian: Fungsi dapat mengembalikan nilai menggunakan kata kunci 'return'.
* Fungsi Tanpa Parameter: Fungsi juga dapat didefinisikan tanpa parameter.
* Fungsi Default Parameter: Parameter fungsi dapat memiliki nilai default yang akan digunakan jika tidak ada argumen yang diberikan saat pemanggilan fungsi.

## 3.3 **Mendefinisikan dan Memanggil Fungsi**

Mendefinisikan dan memanggil fungsi adalah langkah-langkah penting dalam pemrograman. Berikut adalah contoh sederhana cara mendefinisikan dan memanggil fungsi dalam Python:

1. Mendefinisikan Fungsi:

* Fungsi 'sapa()' tidak memiliki parameter dan hanya mencetak pesan sapaan.
* Fungsi 'hitung\_persegi\_panjang(panjang, lebar)' memiliki dua parameter (panjang dan lebar) dan mencetak luas persegi panjang.
* Fungsi 'tambahkan(angka1, angka2)' memiliki dua parameter dan menggunakan pernyataan return untuk mengembalikan hasil penambahan.

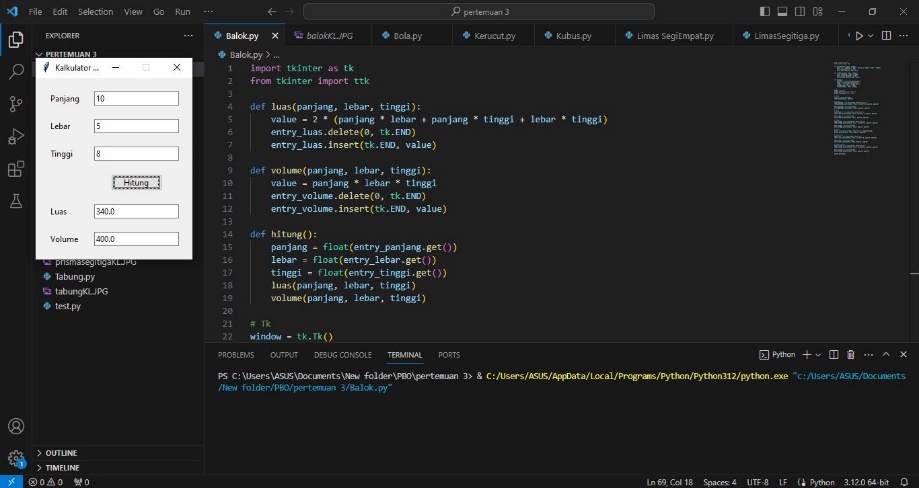
1. Memanggil Fungsi:

* Fungsi 'sapa()' dipanggil tanpa argumen karena tidak memiliki parameter.
* Fungsi 'hitung\_persegi\_panjang(5, 3)' dipanggil dengan memberikan nilai panjang dan lebar.
* Fungsi tambahkan'(7, 4)' dipanggil dan hasilnya disimpan dalam variabel 'hasil\_penambahan', kemudian dicetak.

## 3.4 **Latihan Praktikum Luas dan Volume Bangun Ruang menggunakan Tkinter**

Berikut adalah langkah-langkah praktikum untuk membuat aplikasi yang menghitung luas dan volume bangun ruang:

1. Balok

* Impor Modul Tkinter dan ttk
* Definisikan Fungsi 'luas' dan 'volume'
* Definisikan Fungsi 'hitung'
* Buat Jendela Tkinter dan Frame
* Buat Widget Panjang, Lebar, Tinggi, dan Tombol
* Buat Output Luas dan Volume
* Atur Tata Letak dan Tampilkan GUI
* Berikut adalah program menghitung luas dan volume bangun ruang balok:

import tkinter as tk

from tkinter import ttk

def luas(panjang, lebar, tinggi):

    value = 2 \* (panjang \* lebar + panjang \* tinggi + lebar \* tinggi)

    entry\_luas.delete(0, tk.END)

    entry\_luas.insert(tk.END, value)

def volume(panjang, lebar, tinggi):

    value = panjang \* lebar \* tinggi

    entry\_volume.delete(0, tk.END)

    entry\_volume.insert(tk.END, value)

def hitung():

    panjang = float(entry\_panjang.get())

    lebar = float(entry\_lebar.get())

    tinggi = float(entry\_tinggi.get())

    luas(panjang, lebar, tinggi)

    volume(panjang, lebar, tinggi)

# Tk

window = tk.Tk()

window.title("Kalkulator Balok")

window.resizable(False, False)

# Frame

frame = ttk.Frame(window)

frame.pack(padx=10, pady=10)

# Panjang

label\_panjang = ttk.Label(frame, text="Panjang")

label\_panjang.grid(column=0, row=0, sticky=tk.W, padx=10, pady=10)

entry\_panjang = ttk.Entry(frame)

entry\_panjang.grid(column=1, row=0, padx=10, pady=10)

# Lebar

label\_lebar = ttk.Label(frame, text="Lebar")

label\_lebar.grid(column=0, row=1, sticky=tk.W, padx=10, pady=10)

entry\_lebar = ttk.Entry(frame)

entry\_lebar.grid(column=1, row=1, padx=10, pady=10)

# Tinggi

label\_tinggi = ttk.Label(frame, text="Tinggi")

label\_tinggi.grid(column=0, row=2, sticky=tk.W, padx=10, pady=10)

entry\_tinggi = ttk.Entry(frame)

entry\_tinggi.grid(column=1, row=2, padx=10, pady=10)

# Button

button = ttk.Button(frame, text="Hitung", command=hitung)

button.grid(column=1, row=3, padx=10, pady=10

# Output Luas

label\_luas = ttk.Label(frame, text="Luas")

label\_luas.grid(column=0, row=4, sticky=tk.W, padx=10, pady=10)

entry\_luas = ttk.Entry(frame)

entry\_luas.grid(column=1, row=4, padx=10, pady=10)

# Output volume

label\_volume = ttk.Label(frame, text="Volume")

label\_volume.grid(column=0, row=5, sticky=tk.W, padx=10, pady=10)

entry\_volume = ttk.Entry(frame)

entry\_volume.grid(column=1, row=5, padx=10, pady=10)

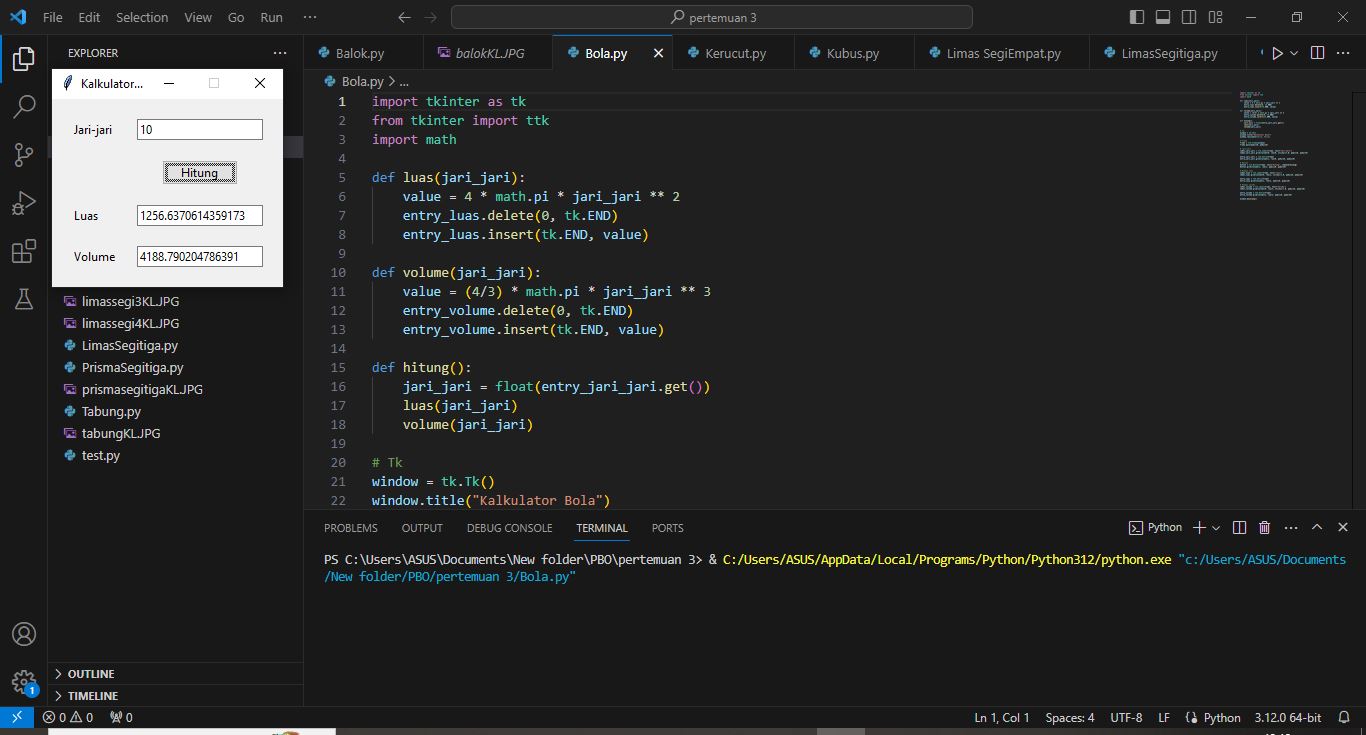
window.mainloop()

Berikut adalah penjelasan ringkas untuk setiap bagian dari kode:

1. Import Modul tkinter: Mengimpor modul tkinter untuk membuat antarmuka pengguna.
2. Definisi Fungsi: Dua fungsi didefinisikan:

* luas(panjang, lebar, tinggi): Menghitung luas permukaan balok.
* volume(panjang, lebar, tinggi): Menghitung volume balok.

1. Fungsi Hitung: Dipanggil saat tombol "Hitung" ditekan. Mengambil input dari pengguna dan memanggil fungsi luas() dan volume().
2. Membuat Window Tkinter: Membuat jendela Tkinter dengan judul "Kalkulator Balok" dan mengatur agar tidak dapat diubah ukurannya.
3. Membuat Frame: Membuat frame di dalam jendela untuk menempatkan elemen antarmuka pengguna.
4. Membuat Label dan Entry: Membuat label dan kotak masukan untuk panjang, lebar, tinggi, luas, dan volume.
5. Membuat Tombol Hitung: Membuat tombol "Hitung" yang memanggil fungsi hitung() saat ditekan.
6. Menjalankan Event Loop: Memulai event loop Tkinter untuk menjaga jendela tetap terbuka dan menanggapi interaksi pengguna.
7. Bola

* Berikut adalah program menghitung luas dan volume bangun ruang bola:

import tkinter as tk

from tkinter import ttk

import math

def luas(jari\_jari):

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

    entry\_luas.delete(0, tk.END)

    entry\_luas.insert(tk.END, value)

def volume(jari\_jari):

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

    entry\_volume.delete(0, tk.END)

    entry\_volume.insert(tk.END, value)

def hitung():

    jari\_jari = float(entry\_jari\_jari.get())

    luas(jari\_jari)

    volume(jari\_jari)

# Tk

window = tk.Tk()

window.title("Kalkulator Bola")

window.resizable(False, False)

# Frame

frame = ttk.Frame(window)

frame.pack(padx=10, pady=10)

# Jari-jari

label\_jari\_jari = ttk.Label(frame, text="Jari-jari")

label\_jari\_jari.grid(column=0, row=0, sticky=tk.W, padx=10, pady=10)

entry\_jari\_jari = ttk.Entry(frame)

entry\_jari\_jari.grid(column=1, row=0, padx=10, pady=10)

# Button

button = ttk.Button(frame, text="Hitung", command=hitung)

button.grid(column=1, row=1, padx=10, pady=10)

# Output Luas

label\_luas = ttk.Label(frame, text="Luas")

label\_luas.grid(column=0, row=2, sticky=tk.W, padx=10, pady=10)

entry\_luas = ttk.Entry(frame)

entry\_luas.grid(column=1, row=2, padx=10, pady=10)

# Output volume

label\_volume = ttk.Label(frame, text="Volume")

label\_volume.grid(column=0, row=3, sticky=tk.W, padx=10, pady=10)

entry\_volume = ttk.Entry(frame)

entry\_volume.grid(column=1, row=3, padx=10, pady=10)

window.mainloop()

# PERTEMUAN 4 PYTHON MODULE,FILE (I/O) & LATIHAN PRAKTIKUM

## 4.1 Pengertian Modul dalam Python

Modul dalam Python adalah file Python yang berisi definisi dan pernyataan Python. Modul memungkinkan kita untuk mengorganisir kode ke dalam file terpisah untuk memudahkan pemeliharaan, pengelolaan, dan pemahaman kode. Saat Anda memiliki potongan kode yang dapat digunakan di beberapa program atau skrip, mengorganisirnya ke dalam modul adalah pilihan yang baik.

## 4.2 Menggunakan Modul

Untuk menggunakan Python dan mengimplementasikan modul, berikut adalah langkah-langkah umum yang dapat Anda ikuti:

* Membuat Modul: Buat file Python baru dengan ekstensi .'py' untuk menampung definisi-modul Anda.Misalnya, buat file 'matematika.py' dan tambahkan fungsi-fungsi seperti yang telah dijelaskan sebelumnya.
* Menggunakan Modul: Buat skrip Python atau aplikasi yang akan menggunakan modul yang sudah Anda buat. Gunakan pernyataan 'import' untuk mengimpor modul atau fungsi dari modul.
* Menjalankan Skrip: Simpan skrip utama Anda dan jalankan dengan menggunakan terminal atau command prompt.
* Menggunakan Alias atau Mengimpor Fungsi Tertentu (Opsional): Jika diperlukan, Anda dapat menggunakan alias atau mengimpor fungsi tertentu dari modul.
* Tes Modul (Opsional): Jika Anda menambahkan blok 'if \_\_name\_\_ == "\_\_main\_\_":' di dalam modul, Anda dapat menambahkan kode pengujian atau skrip utama yang hanya dijalankan ketika modul dijalankan secara langsung.

## 4.3 Pengenalan File Input/Output (I/O)

File Input/Output (I/O) adalah proses membaca dan menulis data dari atau ke file di sistem operasi. Dalam pemrograman, operasi I/O seringkali diperlukan untuk menyimpan data ke dalam file atau membaca data dari file. Dalam Python, Anda dapat melakukan operasi I/O dengan menggunakan fungsi-fungsi yang disediakan dalam modul 'open()' untuk membuka file dan menggunakan metode file seperti 'read()', 'write()', dan lainnya.

## 4.4 Operasi File

Operasi file mencakup berbagai tindakan seperti membuka file, membaca dan menulis data, menutup file. Berikut adalah beberapa operasi umum pada file dalam bahasa pemrograman Python**:**

* Membuka File,Untuk membuka file, Anda menggunakan fungsi open() dengan mode tertentu. Mode umum termasuk:

'r': Mode baca (default), membuka file untuk membaca.

'w': Mode tulis, membuka file untuk menulis (menggantikan isi jika file sudah ada).

'a': Mode tambah, membuka file untuk menulis (menambahkan ke akhir file jika file sudah ada).

'b': Mode biner, digunakan bersamaan dengan mode lain untuk membuka file sebagai file biner.

* Membaca dari File: Metode ‘read()’ digunakan untuk membaca seluruh isi file atau sejumlah karakter tertentu.
* Menulis ke File: Metode 'write()' digunakan untuk menulis data ke dalam file.
* Menutup File: Setelah selesai bekerja dengan file, penting untuk menutupnya menggunakan metode 'close()'.

## 4.5 Latihan Praktikum membuat entry jadwal kuliah menggunakan Tkinter

* berikut adalah contoh sourcode latihan praktikum untuk membuat entry jadwal kuliah menggunakan Python:

import tkinter as tk

from tkinter import filedialog

# Fungsi untuk menambahkan jadwal kuliah ke daftar

def tambahkan\_jadwal():

    hari = hari\_entry.get()

    jam = jam\_entry.get()

    matakuliah = matakuliah\_entry.get()

    sks = sks\_entry.get()

    jadwal = f"{hari}, {jam}: {matakuliah} ({sks})sks\n"

    jadwal\_text.insert(tk.END, jadwal)

# Fungsi untuk menyimpan jadwal ke file

def simpan\_jadwal():

    file\_path = filedialog.asksaveasfilename(defaultextension=".txt", filetypes=[("Text Files", "\*.txt")])

    if file\_path:

        with open(file\_path, 'w') as file:

            file.write(jadwal\_text.get('1.0', tk.END))

# Fungsi untuk membuka jadwal dari file

def buka\_jadwal():

    file\_path = filedialog.askopenfilename(filetypes=[("Text Files", "\*.txt")])

    if file\_path:

        with open(file\_path, 'r') as file:

            jadwal = file.read()

            jadwal\_text.delete('1.0', tk.END)

            jadwal\_text.insert(tk.END, jadwal)

# Membuat jendela tkinter

root = tk.Tk()

root.title("Jadwal Kuliah")

# Mengatur ukuran jendela

lebar = 450

tinggi = 470

root.geometry(f"{lebar}x{tinggi}")

# Mengubah warna latar belakang menjadi merah

root.configure(bg='green')

# Tombol untuk membuka jadwal dari file

buka\_button = tk.Button(root, text="Buka Jadwal", command=buka\_jadwal)

buka\_button.grid(row=1, column=0, columnspan=1)

# Tombol untuk menyimpan jadwal ke file

simpan\_button = tk.Button(root, text="Simpan Jadwal", command=simpan\_jadwal)

simpan\_button.grid(row=1, column=1, columnspan=2)

# Area teks untuk menampilkan jadwal

jadwal\_text = tk.Text(root, height=20, width=60)

jadwal\_text.grid(row=2, column=0, columnspan=2)

# Membuat entri untuk hari

hari\_label = tk.Label(root, text="Hari:")

hari\_label.grid(row=3, column=0)

hari\_entry = tk.Entry(root)

hari\_entry.grid(row=3, column=1)

# Membuat entri untuk jam

jam\_label = tk.Label(root, text="Jam:")

jam\_label.grid(row=4, column=0)

jam\_entry = tk.Entry(root)

jam\_entry.grid(row=4, column=1)

# Membuat entri untuk nama mata kuliah

matakuliah\_label = tk.Label(root, text="Mata Kuliah:")

matakuliah\_label.grid(row=5, column=0)

matakuliah\_entry = tk.Entry(root)

matakuliah\_entry.grid(row=5, column=1)

# Membuat entri untuk Jumlah SKS

sks\_label = tk.Label(root, text="SKS:")

sks\_label.grid(row=6, column=0)

sks\_entry = tk.Entry(root)

sks\_entry.grid(row=6, column=1)

# Tombol untuk menambahkan jadwal

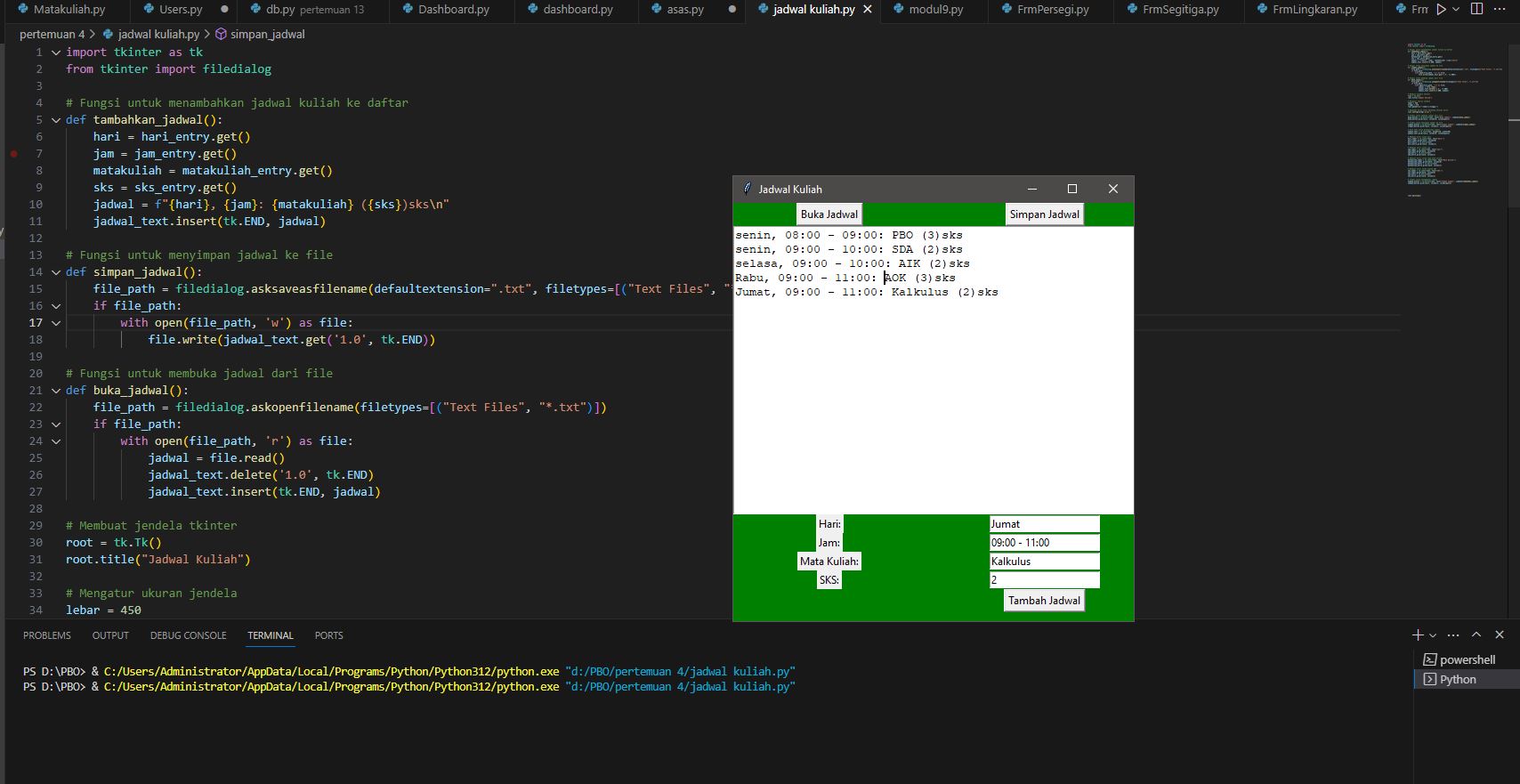
tambah\_button = tk.Button(root, text="Tambah Jadwal", command=tambahkan\_jadwal)

tambah\_button.grid(row=7, column=1, columnspan=2)

root.mainloop()

Berikut adalah penjelasan singkat untuk kode di atas:

* Impor Modul: Mengimpor modul tkinter untuk pembuatan antarmuka pengguna dan filedialog untuk membuka dan menyimpan file.
* Definisi Fungsi: Terdapat tiga fungsi: tambahkan\_jadwal(), simpan\_jadwal(), dan buka\_jadwal(). Mereka bertanggung jawab untuk menambahkan jadwal, menyimpan jadwal ke file, dan membuka jadwal dari file.
* Membuat Jendela Tkinter: Membuat jendela Tkinter dengan judul "Jadwal Kuliah", mengatur ukuran dan warna latar belakangnya.
* Membuat Tombol: Membuat tombol "Buka Jadwal" dan "Simpan Jadwal" yang terhubung dengan fungsi buka\_jadwal() dan simpan\_jadwal().
* Membuat Area Teks: Membuat area teks untuk menampilkan jadwal kuliah yang bisa dibuka dari file.
* Membuat Entri: Membuat entri untuk input hari, jam, nama mata kuliah, dan jumlah SKS.
* Membuat Tombol Tambah: Membuat tombol "Tambah Jadwal" untuk menambahkan jadwal baru berdasarkan input pengguna.

dan berikut adalah hasil dari sourcecode untuk membuat entry jadwal kuliah menggunakan Python:

# PERTEMUAN 6 MULTIFORM & LATIHAN PRAKTIKUM

## 6.1 Pengertian Multiform

Dalam konteks pengembangan perangkat lunak, "multiform" mungkin merujuk pada aplikasi atau antarmuka pengguna yang terdiri dari beberapa formulir atau jendela. Misalnya, aplikasi dengan beberapa formulir yang berbeda untuk mengumpulkan informasi dari pengguna atau untuk menangani fungsi-fungsi yang berbeda.

## Penerapan Multiform Dalam Python

Pada umumnya, dalam konteks pengembangan perangkat lunak menggunakan Python, "multiform" dapat merujuk pada penggunaan beberapa jendela atau formulir yang berbeda dalam aplikasi. Sebagai contoh, Anda dapat menggunakan modul Tkinter untuk membuat aplikasi GUI dengan lebih dari satu jendela.

## 6.3 Latihan Praktikum Membuat Multiform Tkinter

* berikut adalah contoh sourcode latihan praktikum untuk membuat Multiform tkinter menggunakan Python:
* sourcecode dashboard

import tkinter as tk

from tkinter import Menu

from FrmPersegi import \*

from FrmBalok import \*

from FrmSegitiga import \*

from FrmTabung import \*

from FrmPrismaSegitiga import \*

from FrmKubus import \*

from FrmBalok import \*

from FrmBola import \*

from FrmKerucut import \*

from FrmLimasSegitiga import \*

from FrmLimasSegiEmpat import \*

# root window

root = tk.Tk()

root.title('Menu Demo')

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

root.geometry("900x400")

# create a menubar

menubar = Menu(root)

root.config(menu=menubar)

# create a menu

file\_menu = Menu(menubar)

app\_menu = Menu(menubar)

data\_menu = Menu(menubar)

# add a menu item to the menu

file\_menu.add\_command(

    label='File Open', command=root.destroy

)

file\_menu.add\_command(

    label='Exit', command=root.destroy

)

app\_menu.add\_command(

    label='App Persegi', command= lambda: new\_window("Luas Persegi", FrmPersegi)

)

app\_menu.add\_command(

    label='App Segitiga', command= lambda: new\_window("Luas Segitiga", FrmSegitiga)

)

app\_menu.add\_command(

    label='App Lingkaran', command= lambda: new\_window("Luas Lingkaran", FrmLingkaran)

)

app\_menu.add\_command(

    label='App Tabung', command= lambda: new\_window("Luas Tabung", FrmTabung)

)

app\_menu.add\_command(

    label='App PrismaSegitiga', command= lambda: new\_window("Luas PrismaSegitiga", FrmSegitiga)

)

app\_menu.add\_command(

    label='App Kubus', command= lambda: new\_window("Luas Kubus", FrmKubus)

)

app\_menu.add\_command(

    label='App Balok', command= lambda: new\_window("Luas Balok", FrmBalok)

)

app\_menu.add\_command(

    label='App Bola', command= lambda: new\_window("Luas Bola", FrmBola)

)

app\_menu.add\_command(

    label='App Kerucut', command= lambda: new\_window("Luas Kerucut", FrmKerucut)

)

app\_menu.add\_command(

    label='App LimasSegitiga', command= lambda: new\_window("Luas LimasSegitiga", FrmLimasSegitiga)

)

app\_menu.add\_command(

    label='App LimasSegiEmpat', command= lambda: new\_window("Luas LimasSegiEmpat", FrmLimasSegiEmpat)

)

def new\_window( number, \_class):

    new = tk.Toplevel()

    new.transient()

    new.grab\_set()

    \_class(new, number)

# add the File menu to the menubar

menubar.add\_cascade(

    label="File", menu=file\_menu

)

menubar.add\_cascade(

    label="App", menu=app\_menu

)

root.mainloop()

* sourcecode frmbalok

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

class FrmBalok:

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

        self.parent = parent

        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='Tinggi:').grid(row=2, column=0, sticky=W, padx=5, pady=5)

        Label(mainFrame, text="Volume:").grid(row=4, column=0, sticky=W, padx=5, pady=5)

        Label(mainFrame, text="Luas:").grid(row=5, 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.txtTinggi = Entry(mainFrame)

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

        self.txtVolume = Entry(mainFrame)

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

        self.txtLuas = Entry(mainFrame)

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

        # Pasang Button

        self.btnHitung = Button(mainFrame, text='Hitung', command=self.onHitung)

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

    # Fungsi "onHitung" berfungsi untuk menghitung volume dan luas balok

    def onHitung(self, event=None):

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

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

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

        volume = panjang \* lebar \* tinggi

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

        self.txtVolume.delete(0, END)

        self.txtVolume.insert(END, str(volume))

        self.txtLuas.delete(0, END)

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

    def onKeluar(self, event=None):

        # Memberikan perintah menutup aplikasi

        self.parent.destroy()

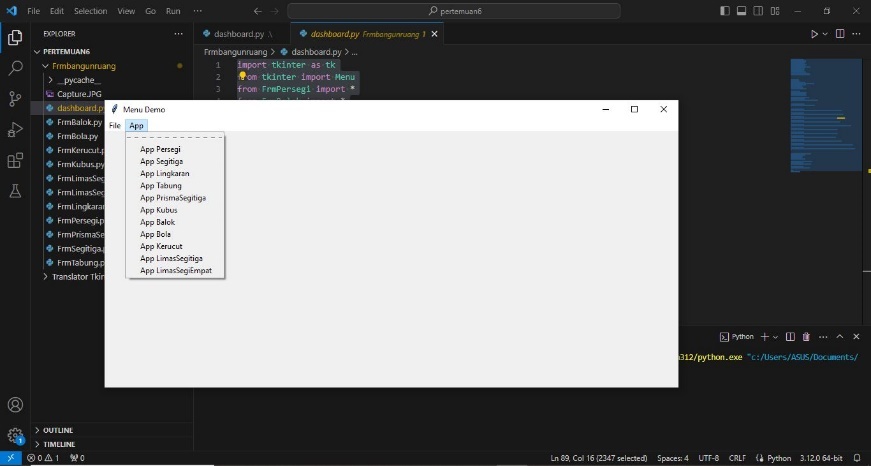
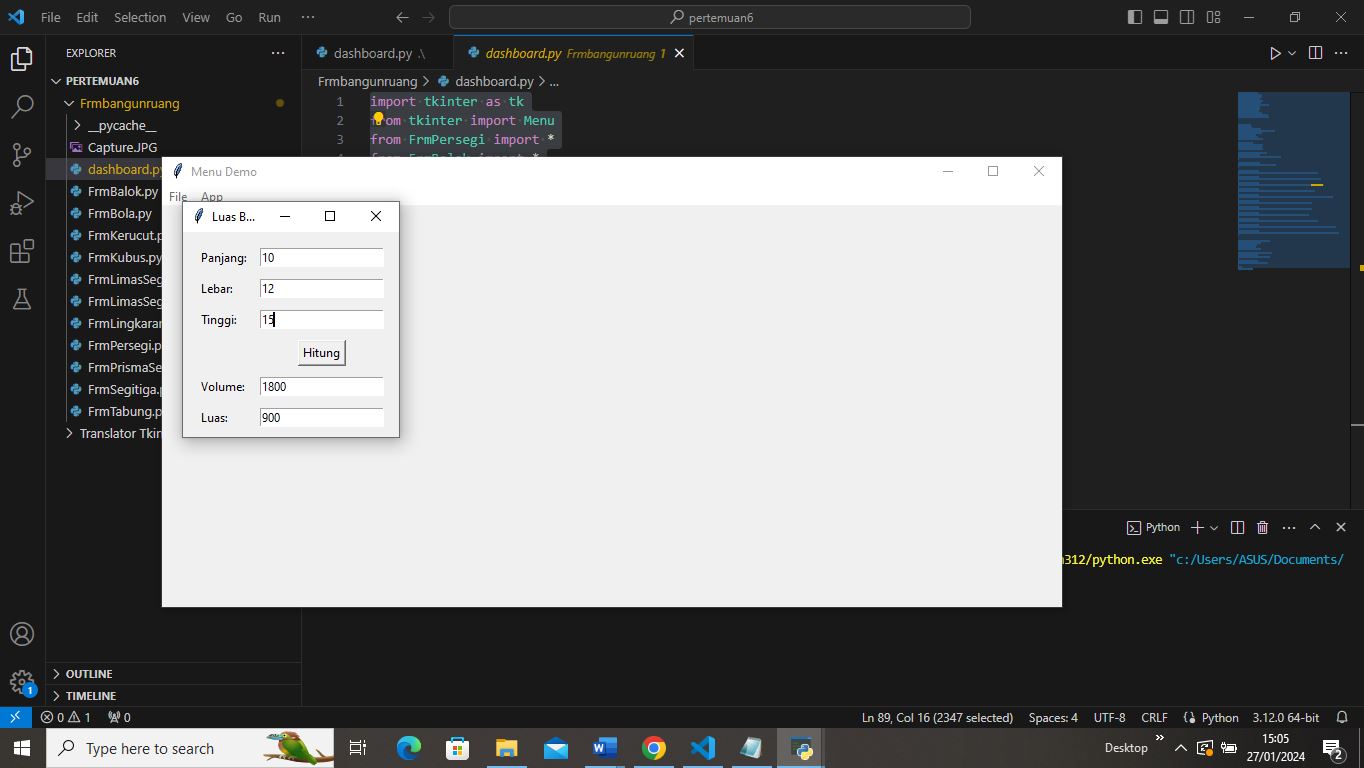
if \_\_name\_\_ == '\_\_main\_\_':

    root = Tk()

    aplikasi = FrmBalok(root, "Program Volume dan Luas Balok")

    root.mainloop()

dan berikut adalah hasil dari sourcecode untuk membuat Multiform tkinter menggunakan Python:

* dashboard
  + ****Frmbalok

# PERTEMUAN 7: APLIKASI BANTU (TOOLS) & LATIHAN PRAKTIKUM

## 7.1 Pengertian Aplikasi Bantu (Tools):

Aplikasi Bantu atau Tools merupakan perangkat lunak yang dirancang untuk membantu pengguna dalam menyelesaikan tugas atau aktivitas tertentu. Tools dapat mencakup berbagai fungsi, mulai dari yang sederhana hingga yang kompleks, sesuai dengan kebutuhan pengguna.

## 7.2 Pembuatan Aplikasi Bantu (Tools):

Untuk membuat Tools sederhana, biasanya memerlukan pemahaman dasar tentang pemrograman dan pengembangan perangkat lunak. Berikut adalah langkah-langkah umum dalam membuat Tools sederhana:

* Tentukan Tujuan: Jelaskan apa yang ingin dicapai oleh Tools Anda.
* Pilih Platform: Tentukan di mana Tools akan berjalan: desktop, web, atau mobile.
* Pilih Bahasa Pemrograman: Pilih bahasa pemrograman yang sesuai.
* Rancang Antarmuka Pengguna: Buat desain antarmuka yang sederhana dan mudah dimengerti.
* Implementasikan Fungsi-Fungsi Utama: Tuliskan kode untuk fungsi-fungsi inti.
* Uji dan Perbaiki Bug: Lakukan pengujian menyeluruh dan perbaiki bug.
* Distribusikan atau Gunakan Secara Internal: Siapkan untuk distribusi kepada pengguna atau penggunaan internal di organisasi.

## 7.3 Latihan Praktikum Membuat App Mp3, Mp4, Pic To Teks Dan Text To Sound:

Berikut adalah langkah-langkah praktikum untuk membuat aplikasi yang mengonversi file MP3, MP4, gambar menjadi teks, dan teks menjadi suara (text to speech):

1. Membuat Aplikasi MP3:

* Untuk dapat membuat aplikasi pemutar mp3 anda perlu menginstall py game dengan perintah di terminal atau cmd

**“pip install pygame”**

* Berikut adalah program pemutar music/mp3:

import tkinter as tk

from tkinter import filedialog, messagebox

import pygame

import os

class MP3Player:

    def \_\_init\_\_(self, master):

        self.master = master

        self.master.title("MP3 Player")

        self.master.geometry("360x180")

        self.playlist = []

        self.current\_index = 0

        self.setup\_ui()

    def setup\_ui(self):

        # Button untuk memilih file MP3

        self.add\_button = tk.Button(self.master, text="Add Song", width =10,bg="black",fg="yellow", command=self.add\_song)

        self.add\_button.grid(row=1,column=0,columnspan=1)

        # Button untuk memainkan atau menghentikan lagu

        self.play\_button=tk.Button(root,text="Play",width =10,bg="black",fg="yellow", command=self.play\_pause)

        self.play\_button.grid(row=1,column=1,columnspan=1)

        # Listbox untuk menampilkan daftar playlist

        self.playlist\_listbox = tk.Listbox(self.master,width=60, height=10,bg="black",fg="white", selectmode=tk.SINGLE)

        self.playlist\_listbox.grid(row=2,column=0,columnspan=4)

        # Button untuk memainkan lagu berikutnya

        self.next\_button = tk.Button(self.master, text="Next", width =10,bg="black",fg="yellow", command=self.next\_song)

        self.next\_button.grid(row=1,column=3,columnspan=1)

        # Button untuk memainkan lagu sebelumnya

        self.prev\_button = tk.Button(self.master, text="Previous",bg="black",fg="yellow", width =10, command=self.prev\_song)

        self.prev\_button.grid(row=1,column=2,columnspan=1)

        # Inisialisasi pygame mixer

        pygame.mixer.init()

    def add\_song(self):

        song\_path = filedialog.askopenfilename(filetypes=[("MP3 Files", "\*.mp3")])

        if song\_path:

            song\_name = os.path.basename(song\_path)

            self.playlist.append((song\_name, song\_path))

            self.playlist\_listbox.insert(tk.END, song\_name)

    def play\_pause(self):

        if  pygame.mixer.music.get\_busy():

            pygame.mixer.music.pause()

            self.play\_button.config(text="Play")

        else:

            if not self.playlist:

                return

            song\_name, song\_path = self.playlist[self.current\_index]

            pygame.mixer.music.load(song\_path)

            pygame.mixer.music.play()

            self.play\_button.config(text="Pause")

    def next\_song(self):

        if not self.playlist:

            return

        self.current\_index = (self.current\_index + 1) % len(self.playlist)

        self.play\_song()

    def prev\_song(self):

        if not self.playlist:

            return

        self.current\_index = (self.current\_index - 1) % len(self.playlist)

        self.play\_song()

    def play\_song(self):

        if not self.playlist:

            return

        song\_name, song\_path = self.playlist[self.current\_index]

        pygame.mixer.music.load(song\_path)

        pygame.mixer.music.play()

        self.play\_button.config(text="Pause")

# Membuat instance Tkinter

root = tk.Tk()

app = MP3Player(root)

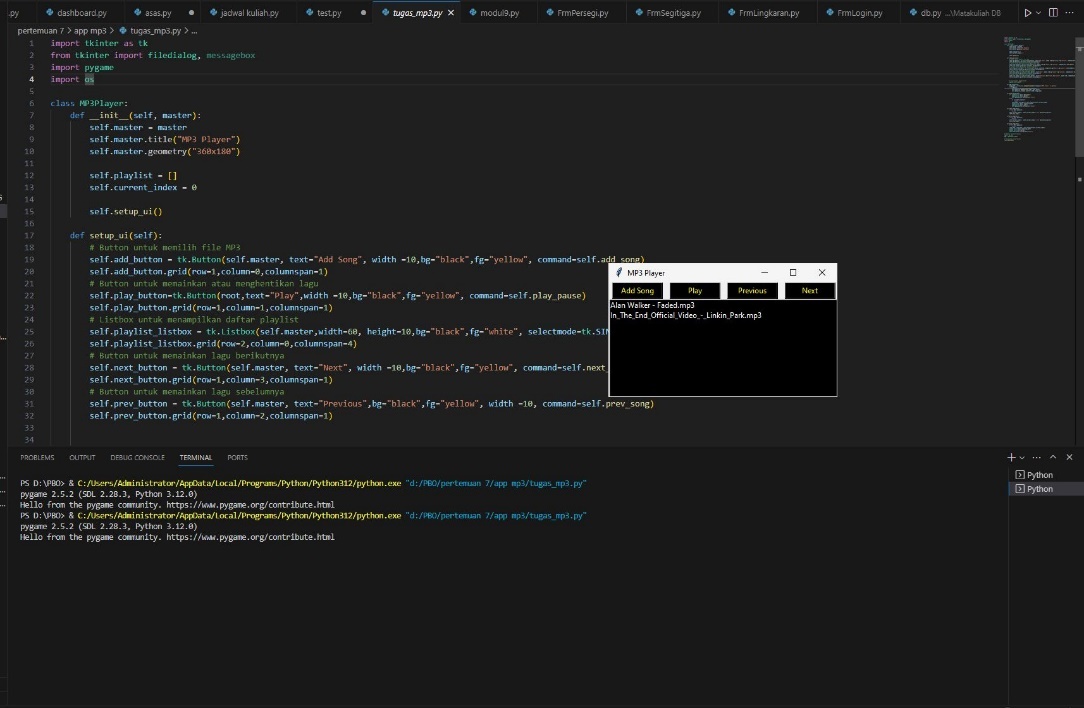
# Menjalankan loop Tkinter

root.mainloop()

Berikut adalah penjelasan singkat untuk kode di atas:

* Import Modul: Mengimpor modul tkinter, filedialog, dan pygame.
* Class MP3Player: Ini adalah kelas utama yang berfungsi sebagai MP3 player. Metode \_\_init\_\_ digunakan untuk inisialisasi objek MP3 player.
* Setup UI: Metode setup\_ui digunakan untuk mengatur antarmuka pengguna aplikasi MP3 player.
* Metode setup\_ui(self): Mengatur antarmuka pengguna.
* Metode add\_song: Menambahkan lagu ke dalam daftar putar.
* Metode play\_pause: Memainkan atau menghentikan lagu.
* Metode next\_song: Memainkan lagu berikutnya.
* Metode prev\_song: Memainkan lagu sebelumnya.
* Metode play\_song: Memainkan lagu yang dipilih.

Berikut adalah hasil program pemutar music/mp3:

****

1. Membuat Aplikasi MP4 to Text:

* Untuk dapat membuat aplikasi pemutar video/mp4 anda harus menginstall dengan perintah

**“pip install opencv-python pillow”**

* Berikut adalah program pemutar video/mp4 :

import cv2

def play\_video(file\_path):

    # Buka file video

    cap = cv2.VideoCapture(file\_path)

    # Periksa apakah file video berhasil dibuka

    if not cap.isOpened():

        print("Error: Gagal membuka file video.")

        return

    while True:

        # Baca frame dari video

        ret, frame = cap.read()

        # Periksa apakah bacaan frame berhasil

        if not ret:

            print("Error: Gagal membaca frame dari video.")

            break

        # Tampilkan frame

        cv2.imshow('Video Player', frame)

        # Hentikan pemutaran jika pengguna menekan tombol 'q'

        if cv2.waitKey(30) & 0xFF == ord('q'):

            break

    # Tutup video

    cap.release()

    # Tutup jendela tampilan

    cv2.destroyAllWindows()

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

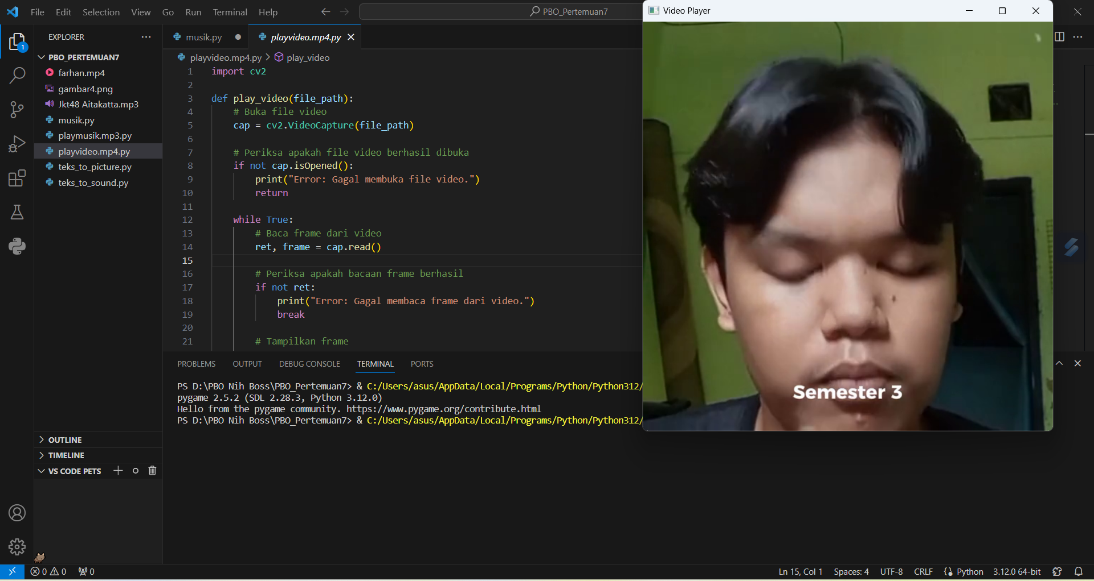
    video\_path = "farhan.mp4"  # Ganti dengan path file video Anda

    play\_video(video\_path)

Berikut adalah penjelasan singkat untuk kode di atas:

* Import Modul: Kode mengimpor modul cv2 untuk memproses video.
* Fungsi play\_video(file\_path): Fungsi ini membuka file video yang diberikan, memainkannya, dan menampilkan frame satu per satu.
* Loop Pemutaran: Dilakukan loop untuk membaca dan menampilkan setiap frame video.
* Tampilkan Frame: Setiap frame yang dibaca ditampilkan menggunakan cv2.imshow().
* Hentikan Pemutaran: Pemutaran video dapat dihentikan dengan menekan tombol 'q'.
* Main Program: Di dalam blok ini, path file video ditentukan dan fungsi play\_video() dipanggil untuk memainkan video tersebut.

Berikut adalah hasil program pemutar video/mp4 :



1. Membuat Aplikasi Gambar to Text:

* Untuk dapat menjalankan program anda di haruskan untuk menginstall di cmd/terminal dengan perintah

**“pip install pytesseract pillow”**

* Dan anda juga harus menginstall aplikasi tesseract pada perangakat anda yang dapat di untuh melalui URL berikut: https://sourceforge.net/projects/tesseract-ocr.mirror/
* Berikut adalah kode program gambar ke teks:

from tkinter import Tk, Button, Label, filedialog, Frame, LEFT, RIGHT

from PIL import Image, ImageTk

from pytesseract import pytesseract

# Define path to tesseract.exe (update this path based on your Tesseract installation on Windows)

path\_to\_tesseract = r'C:\Program Files\Tesseract-OCR\tesseract.exe'

def open\_file\_dialog():

    file\_path = filedialog.askopenfilename(title="Select Image", filetypes=[("Image files", "\*.png;\*.jpg;\*.jpeg;\*.gif")])

    if file\_path:

        process\_image(file\_path)

def process\_image(image\_path):

    pytesseract.tesseract\_cmd = path\_to\_tesseract

    img = Image.open(image\_path)

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

    result\_label.config(text=text)

    img.thumbnail((400, 400))

    img = ImageTk.PhotoImage(img)

    image\_label.config(image=img)

    image\_label.image = img

# Create Tkinter window

window = Tk()

window.title("OCR with Tkinter")

# Create and pack widgets

frame = Frame(window, pady=10)

frame.pack()

open\_button = Button(frame, text="Open Image", command=open\_file\_dialog)

open\_button.pack(side=LEFT, padx=10)

result\_label = Label(frame, text="", wraplength=400, justify="left", font=("Arial", 12))

result\_label.pack(side=RIGHT, padx=10)

# Label to display the image (optional)

image\_label = Label(window)

image\_label.pack(pady=10)

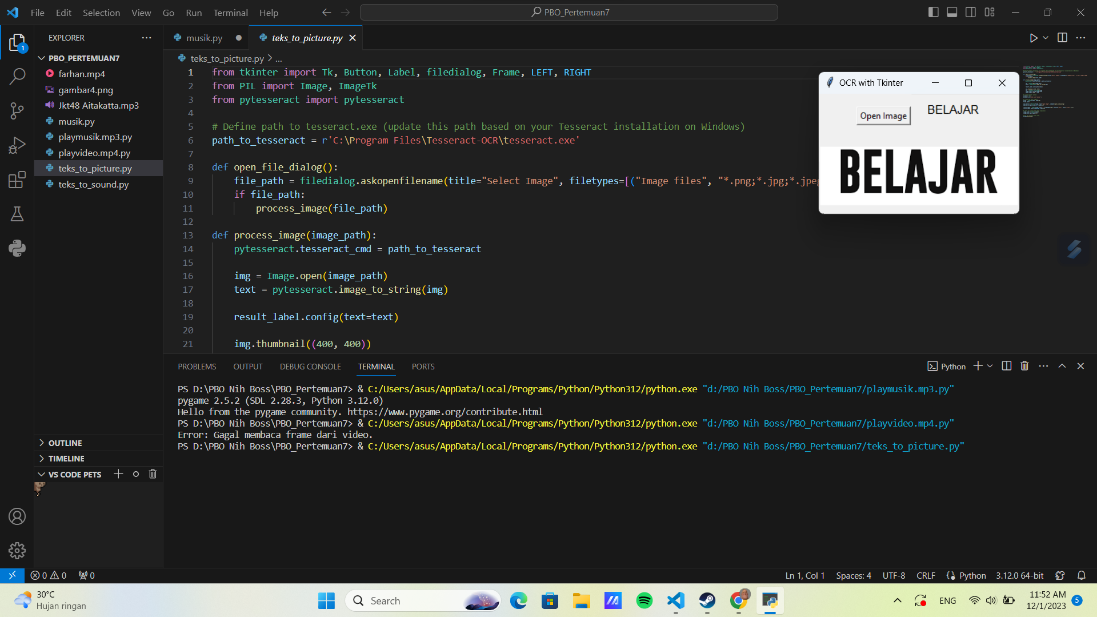
# Run the Tkinter event loop

window.mainloop()

Berikut adalah penjelasan singkat untuk kode di atas:

* Import Modul: Mengimpor modul yang diperlukan dari tkinter, PIL, pytesseract, dan filedialog.
* Definisi Path Tesseract: Menentukan jalur ke tesseract.exe untuk penggunaan modul pytesseract.
* Fungsi open\_file\_dialog(): Membuka dialog pemilihan file gambar saat tombol "Open Image" ditekan, dan memproses gambar yang dipilih menggunakan fungsi process\_image().
* Fungsi process\_image(image\_path): Menggunakan pytesseract untuk melakukan OCR pada gambar yang dipilih. Hasil teks yang dihasilkan ditampilkan di label, sementara gambar yang dipilih ditampilkan di label gambar.
* Membuat Window dan Widget: Membuat jendela Tkinter, tombol "Open Image", dan label untuk menampilkan hasil OCR dan gambar.

Berikut adalah hasil kode program gambar ke teks:



1. Membuat Aplikasi Text to Sound:

* Untuk dapat program anda dapat berjalan dengan lancar anda harus menginstall di cmd/terminal dengan perintah

**“pip install gtts”**

* Berikut adalah kode program dari teks ke suara:

from gtts import gTTS

import os

import platform

import tkinter as tk

from tkinter import Label, Entry, Button

def text\_to\_speech(text, language='id', output\_file='output.mp3'):

    try:

        tts = gTTS(text=text, lang=language, slow=False)

        tts.save(output\_file)

        if platform.system() == 'Windows':

            os.system(f"start {output\_file}")

        else:

            print("Unsupported operating system. Please play the audio file manually.")

        # Optional: Clean up the audio file after playing

        # os.remove(output\_file)

    except Exception as e:

        print(f"Error: {e}")

def convert\_text():

    input\_text = text\_entry.get()

    text\_to\_speech(input\_text)

# Create the main window

window = tk.Tk()

window.title("Text-to-Speech Converter")

# Styling

window.geometry("400x200")  # Set the initial size of the window

window.configure(bg="#f0f0f0")  # Set the background color

# Create GUI components

label = Label(window, text="Enter the text to convert to speech:", font=("Helvetica", 12), bg="#f0f0f0")

label.pack(pady=10)

text\_entry = Entry(window, width=50, font=("Helvetica", 10))

text\_entry.pack(pady=10)

convert\_button = Button(window, text="Convert", command=convert\_text, font=("Helvetica", 12), bg="#4CAF50", fg="white")

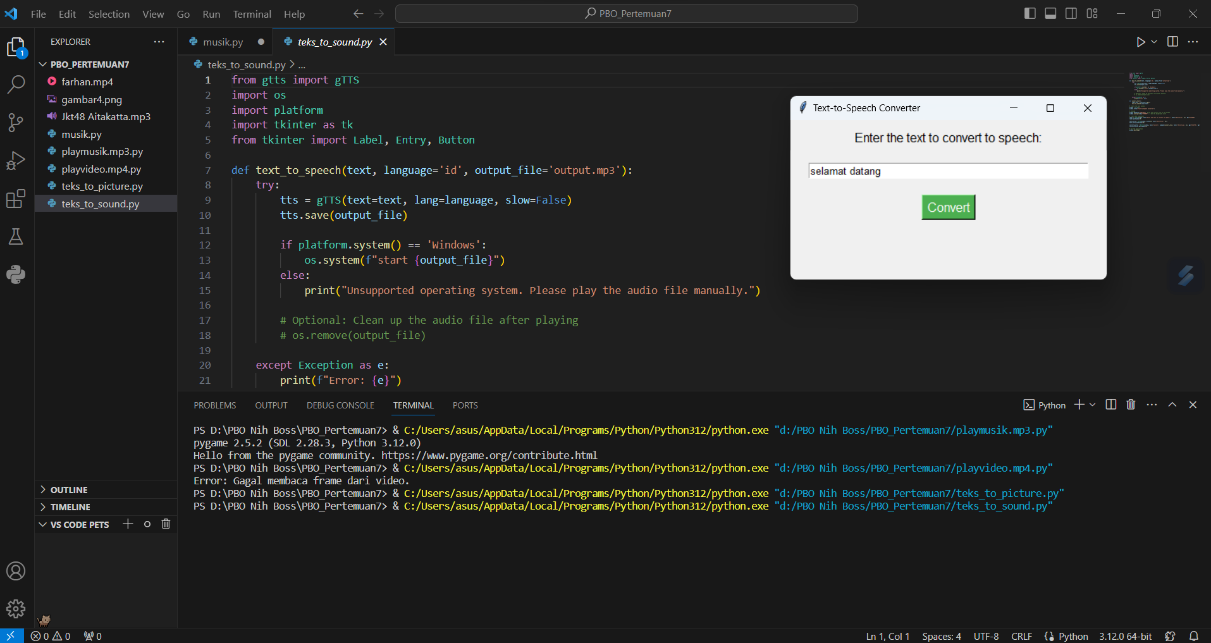
convert\_button.pack(pady=10)

# Run the application

window.mainloop()

Berikut adalah penjelasan singkat untuk kode di atas:

* Import Modul: Mengimpor modul gTTS dari gtts, os, platform, dan tkinter.
* Fungsi text\_to\_speech(text): Menerima teks sebagai input, mengonversinya menjadi file audio menggunakan gTTS, kemudian memainkan file audio.
* Fungsi convert\_text(): Membaca teks dari kotak masukan dan memanggil fungsi text\_to\_speech() untuk mengonversinya menjadi file audio.
* Membuat Jendela Tkinter dan Widget: Membuat jendela Tkinter dengan label, kotak masukan, dan tombol.
* Menjalankan Event Loop Tkinter: Memulai event loop Tkinter untuk menjalankan aplikasi.

Berikut adalah hasil kode program dari teks ke suara:

# PERTEMUAN 9: KONSEP DASAR CLASS DAN OBJEK & LATIHAN PRAKTIKUM

## 9.1 Pengertian Konsep Dasar Class Dan Objek & Latihan Praktikum

A. Class dan Objek merupakan konsep dasar dalam pemrograman berorientasi objek (OOP).

1. Class adalah blueprint atau cetak biru dalam pemrograman berorientasi objek (OOP) yang mendefinisikan atribut dan perilaku objek. Tujuannya adalah menyusun kode menjadi unit terorganisir, memungkinkan pengelompokan atribut dan metode yang serupa. Atribut adalah variabel yang dideklarasikan di dalam class, merepresentasikan data objek, sementara metode adalah fungsi atau perilaku yang terkait dengan class, digunakan untuk manipulasi data. Dengan class, objek dapat dibuat dengan karakteristik dan perilaku yang sudah ditentukan, mempermudah pengelolaan dan penggunaan kode dalam program.
2. Objek adalah instance konkret dari sebuah class yang dibuat saat program dijalankan, bertujuan untuk mewujudkan blueprint class menjadi entitas yang nyata dalam program. Objek memiliki atribut, nilai spesifik sesuai dengan definisi class, dan metode yang dapat dipanggil untuk mewujudkan perilaku yang telah ditentukan oleh class.

B. Konsep dasar dari OOP Modular, OOP Murni, GUI Terstruktur

1. OOP Modular

OOP modular adalah pendekatan dalam OOP yang membagi program menjadi modul atau bagian-bagian yang lebih kecil. Setiap modul memiliki fungsi dan variabelnya sendiri, dan dapat saling berinteraksi dengan modul lainnya. Pendekatan ini memudahkan pengembangan program yang kompleks dan memungkinkan penggunaan kembali kode yang sama di beberapa program

* Program dari OOP modular konversi celsius:

# File-1 (celcius.py)

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

# File-2 (Frmcelcius.py)

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

from oop\_modular import \*

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()

Berikut adalah penjelasan singkat untuk kode program di atas:

* File-1 (celcius.py):
* Mendefinisikan kelas Celcius yang memiliki metode untuk mengonversi suhu dari Celcius ke Fahrenheit, Reamur, dan Kelvin.
* File-2 (Frmcelcius.py):
* Mengimpor kelas Celcius dari file celcius.py untuk digunakan dalam aplikasi GUI.
* Membuat antarmuka pengguna menggunakan modul tkinter, dengan entri untuk input suhu Celcius dan tampilan untuk menampilkan suhu yang dikonversi ke Fahrenheit, Reamur, dan Kelvin.
* Menggunakan metode dari kelas Celcius untuk melakukan konversi suhu saat tombol "Hitung" ditekan.
* Menambahkan fungsi untuk menutup aplikasi saat tombol close di jendela diklik.

1. OOP Murni

OOP murni adalah pendekatan dalam OOP yang mengikuti prinsip-prinsip OOP secara ketat. Prinsip-prinsip ini meliputi enkapsulasi, pewarisan, dan polimorfisme. Enkapsulasi adalah konsep dalam implementasi untuk membungkus data dan fungsi menjadi satu entitas, dan membatasi akses dari luar kelas. Pewarisan adalah konsep di mana kelas baru dapat diturunkan dari kelas yang sudah ada, sehingga mewarisi sifat dan perilaku dari kelas yang sudah ada. Polimorfisme adalah konsep di mana objek dari kelas yang berbeda dapat dipanggil dengan cara yang sama

* Program dari 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 \* suhu) + 32

        return val

    def get\_reamur(self, suhu):

        val = (4/5 \* suhu)

        return val

    def get\_kelvin(self, suhu):

        val = suhu + 273

        return val

    def onHitung(self):

        # Suhu dalam Fahrenheit

        suhu = self.txtCelcius.get()

        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()

Berikut adalah penjelasan singkat untuk kode program di atas:

* Pembuatan Antarmuka: Kelas FrmCelcius menggunakan modul tkinter untuk membuat antarmuka grafis. Ini termasuk pembuatan label dan entri untuk input suhu dalam Celcius, serta entri yang menampilkan hasil konversi suhu.
* Metode Konversi: Terdapat tiga metode, yaitu get\_fahrenheit(), get\_reamur(), dan get\_kelvin(), yang digunakan untuk menghitung nilai suhu dalam skala Fahrenheit, Reamur, dan Kelvin berdasarkan nilai suhu dalam Celcius.
* Tombol "Hitung": Ketika tombol "Hitung" ditekan, metode onHitung() akan dipanggil. Metode ini akan mengambil nilai suhu dalam Celcius dari entri, kemudian menghitung dan menampilkan nilai suhu dalam skala Fahrenheit, Reamur, dan Kelvin pada entri yang sesuai.
* Penanganan Penutupan Aplikasi: Program juga menangani peristiwa penutupan aplikasi dengan metode onKeluar(), yang dipanggil ketika pengguna menekan tombol penutup di jendela aplikasi. Ini memastikan bahwa aplikasi ditutup dengan benar.

1. GUI Terstruktur

GUI terstruktur adalah pendekatan dalam pengembangan antarmuka pengguna (GUI) yang membagi GUI menjadi bagian-bagian yang lebih kecil, seperti panel, tombol, dan kotak teks. Setiap bagian memiliki fungsi dan variabelnya sendiri, dan dapat saling berinteraksi dengan bagian lainnya.

* Program dari 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)

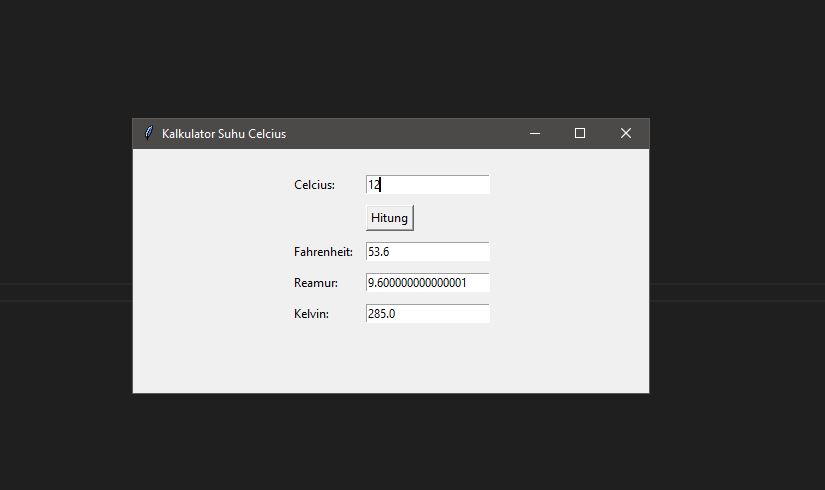
txtReamur = Entry(frame)

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()

Berikut adalah hasil program konversi suhu celcius dari kode program oop modular, oop murni dan gui terstrktur :

# PERTEMUAN 10: PENGETIAN MYSQL & LATIHAN PRAKTIKUM

## 10.1 Pengertian Basis Data, MySQL, Xampp

1. Basis Data (Database)

Basis data adalah kumpulan data yang terorganisir dan terstruktur, yang dapat diakses, dikelola, dan diperbarui dengan mudah. Basis data digunakan untuk menyimpan informasi yang diperlukan oleh aplikasi, seperti informasi pelanggan, informasi produk, dan informasi transaksi

1. MySQL

MySQL adalah database management system (manajemen basis data) menggunakan perintah dasar SQL (Structured Query Language) yang cukup terkenal. digunakan untuk mengelola dan menyimpan data dalam tabel-tabel di dalam basis data. Selain itu, MySQL juga memiliki banyak fitur untuk mengelola basis data, seperti kemampuan untuk mengelola akses ke database dan mengatur hak pengguna

1. Xampp

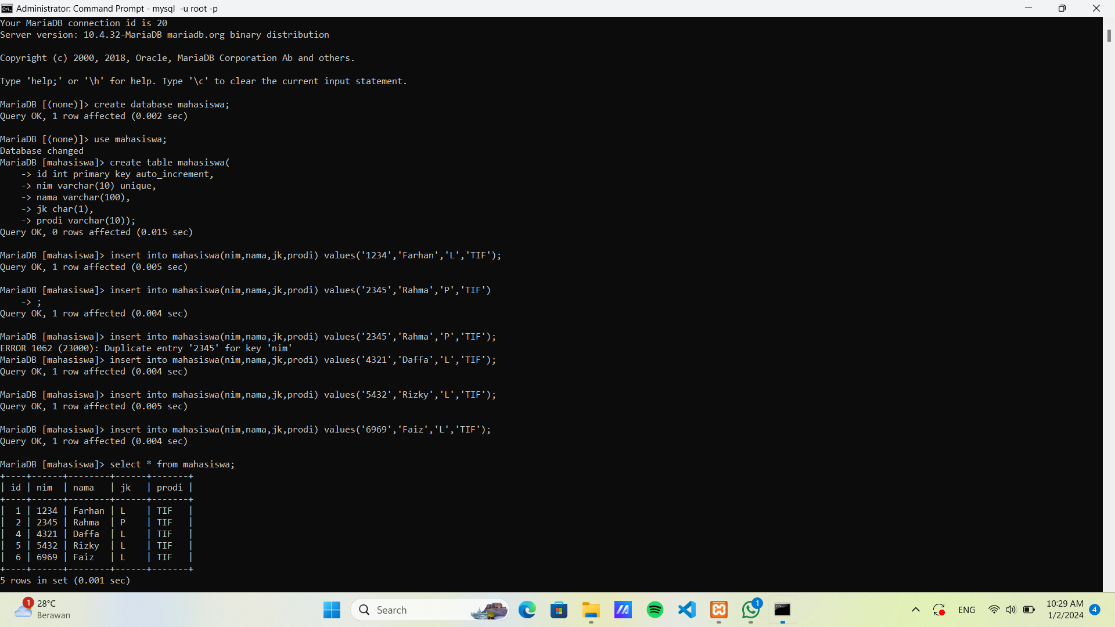
XAMPP adalah singkatan dari "X (yang dapat merujuk pada sistem operasi apa pun), Apache, MySQL, PHP, dan Perl". Ini adalah paket perangkat lunak yang memungkinkan Anda untuk membuat lingkungan pengembangan web lokal yang lengkap dan mudah digunakan. XAMPP memfasilitasi instalasi dan konfigurasi Apache (web server), MySQL (sistem manajemen basis data), PHP (bahasa pemrograman server-side), dan Perl (bahasa skrip) di komputer

## 10.2 Instalasi Xampp

Berikut adalah langkah-langkah umum untuk menginstal XAMPP di sistem operasi Windows:

* Unduh XAMPP: Kunjungi situs web resmi XAMPP (https://www.apachefriends.org/index.html) dan unduh versi terbaru XAMPP sesuai dengan sistem operasi Anda (Windows).
* Mulai Instalasi: Setelah mengunduh file instalasi, buka file tersebut untuk memulai proses instalasi.
* Pilih Komponen: Selama proses instalasi, Anda akan diminta untuk memilih komponen yang ingin Anda instal. Pastikan untuk memilih Apache, MySQL, dan PHP karena mereka adalah komponen inti XAMPP yang diperlukan untuk pengembangan web.
* Pilih Lokasi Instalasi: Anda akan diminta untuk memilih lokasi di mana Anda ingin menginstal XAMPP. Anda dapat menggunakan lokasi default atau memilih lokasi yang sesuai dengan preferensi Anda.
* Proses Instalasi: Lanjutkan dengan proses instalasi dengan mengklik tombol "Next" atau "Install". Tunggu sampai proses instalasi selesai.
* Selesaikan Instalasi: Setelah proses instalasi selesai, Anda mungkin akan ditawari untuk membuka kontrol panel XAMPP. Pilih opsi tersebut jika Anda ingin membuka kontrol panel XAMPP setelah instalasi selesai.
* Konfigurasi: Setelah instalasi selesai, Anda dapat membuka kontrol panel XAMPP dan mengkonfigurasi server Apache dan MySQL sesuai dengan kebutuhan Anda. Anda juga dapat memulai dan menghentikan layanan server dari sini.
* Uji Coba: Setelah semua konfigurasi selesai, Anda dapat membuka browser web Anda dan masukkan alamat http://localhost untuk melihat halaman utama XAMPP. Ini menunjukkan bahwa XAMPP telah berhasil diinstal dan berjalan

## 10.3 Latihan Praktikum membuat tabel mysql menggunakan CMD

1. Contoh Tabel data mahasiswa
2. Membuat Tabel data matakuliah

Microsoft Windows [Version 10.0.22621.2134]

(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>cd\

C:\>cd xampp

C:\xampp>cd mysql

C:\xampp\mysql>cd bin

C:\xampp\mysql\bin>mysql -u root -p

Enter password:

Welcome to the MariaDB monitor. Commands end with ; or \g.

Your MariaDB connection id is 8

Server version: 10.4.32-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> create database kampus;

Query OK, 1 row affected (0.002 sec)

MariaDB [(none)]> use kampus;

Database changed

MariaDB [kampus]> create table matakuliah(

-> id int primary key auto\_increment,

-> kodemk varchar(10) not null,

-> namamk varchar(100) not null,

-> sks int not null);

Query OK, 0 rows affected (0.011 sec)

MariaDB [kampus]> show tables;

+------------------+

| Tables\_in\_kampus |

+------------------+

| matakuliah |

+------------------+

1 row in set (0.002 sec)

MariaDB [kampus]> descirbe matakuliah;

ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near 'descirbe matakuliah' at line 1

MariaDB [kampus]> describe matakuliah;

+--------+--------------+------+-----+---------+----------------+

| Field | Type | Null | Key | Default | Extra |

+--------+--------------+------+-----+---------+----------------+

| id | int(11) | NO | PRI | NULL | auto\_increment |

| kodemk | varchar(10) | NO | | NULL | |

| namamk | varchar(100) | NO | | NULL | |

| sks | int(11) | NO | | NULL | |

+--------+--------------+------+-----+---------+----------------+

4 rows in set (0.009 sec)

MariaDB [kampus]> insert into matakuliah(kodemk,namamk,sks)values('1001','PBO',3);

Query OK, 1 row affected (0.012 sec)

MariaDB [kampus]> insert into matakuliah(kodemk,namamk,sks)values('1002','Bhs. Inggris',2);

Query OK, 1 row affected (0.003 sec)

MariaDB [kampus]> insert into matakuliah(kodemk,namamk,sks)values('1003','Matematika',2);

Query OK, 1 row affected (0.004 sec)

MariaDB [kampus]> insert into matakuliah(kodemk,namamk,sks)values('1004','Bhs. Indonesia',2);

Query OK, 1 row affected (0.004 sec)

MariaDB [kampus]> insert into matakuliah(kodemk,namamk,sks)values('1005','SDA,2);

Query OK, 1 row affected (0.004 sec)

MariaDB [kampus]> select \* from matakuliah;

+----+--------+----------------+-----+

| id | kodemk | namamk | sks |

+----+--------+----------------+-----+

| 1 | 1001 | PBO | 3 |

| 2 | 1002 | Bhs. Inggris | 2 |

| 3 | 1003 | Matematika | 2 |

| 4 | 1004 | Bhs. Indonesia | 2 |

| 5 | 1005 | SDA | 2 |

+----+--------+----------------+-----+

5 rows in set (0.001 sec)

MariaDB [kampus]> insert into matakuliah(kodemk,namamk,sks)values('1005','Agama Islam',2);

Query OK, 1 row affected (0.004 sec)

MariaDB [kampus]> insert into matakuliah(kodemk,namamk,sks)values('100','Algortima',2);

Query OK, 1 row affected (0.004 sec)

MariaDB [kampus]> insert into matakuliah(kodemk,namamk,sks)values('1008','Kewirausahaan',2);

Query OK, 1 row affected (0.005 sec)

MariaDB [kampus]> insert into matakuliah(kodemk,namamk,sks)values('1009','Fisika',2);

Query OK, 1 row affected (0.005 sec)

MariaDB [kampus]> select \* from matakuliah;

+----+--------+----------------+-----+

| id | kodemk | namamk | sks |

+----+--------+----------------+-----+

| 1 | 1001 | PBO | 3 |

| 2 | 1002 | Bhs. Inggris | 2 |

| 3 | 1003 | Matematika | 2 |

| 4 | 1004 | Bhs. Indonesia | 2 |

| 5 | 1005 | SDA | 2 |

| 6 | 1006 | Agama Islam | 2 |

| 7 | 1007 | Algortima | 2 |

| 8 | 1008 | Kewirausahaan | 2 |

| 9 | 1009 | Fiska | 2 |

+----+--------+----------------+-----+

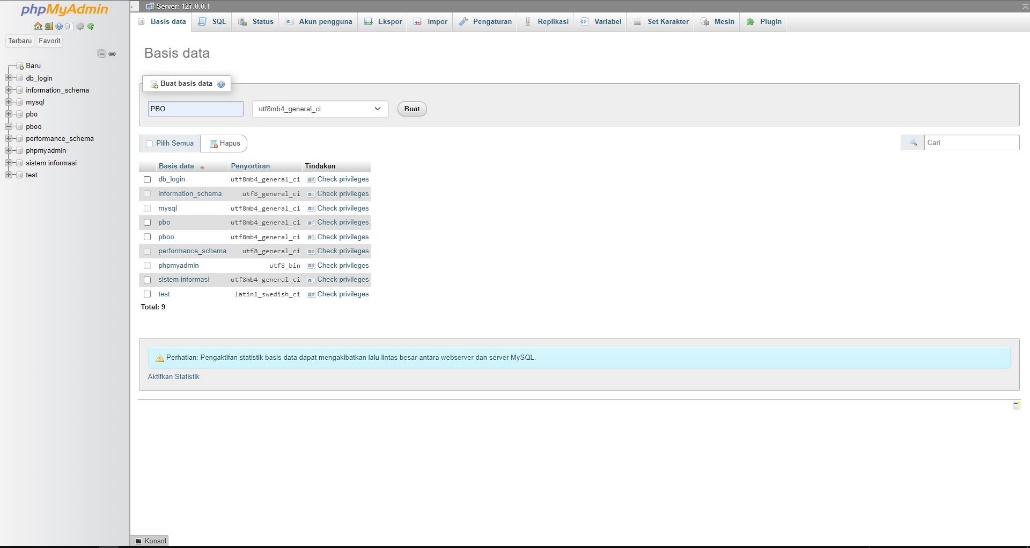
9 rows in set (0.001 sec)

MariaDB [kampus]>

# PERTEMUAN 11: PYTHON DATABASE & LATIHAN PRAKTIKUM

## 11.1. Membuat Database pada phpMyAdmin

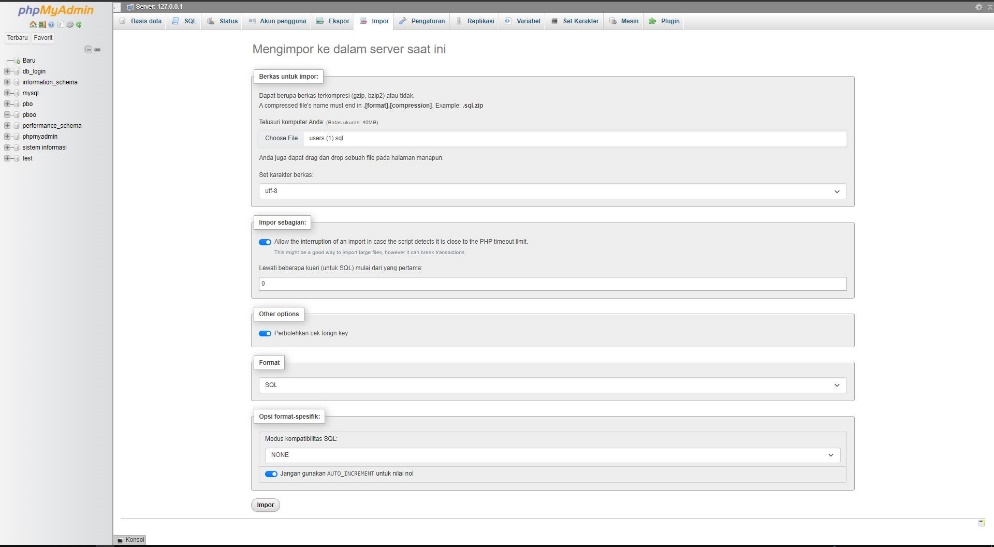
1. Login ke phpMyAdmin: Buka browser dan masuk ke panel phpMyAdmin menggunakan URL http://localhost/phpmyadmin/
2. Pilih Server: Jika memiliki beberapa server terdaftar, pilih server yang diinginkan.
3. Buat Database Baru: Pada panel sebelah kiri, Anda akan melihat opsi untuk "Buat Database". Klik di sana.
4. Isi Detail Database: Berikan nama database yang diinginkan dan pilih pengaturan lainnya seperti pengaturan kolasi jika perlu.



1. Konfirmasi: Klik "Buat" untuk membuat database baru.

## 11.2. Mengimpor Tabel SQL ke dalam phpMyAdmin

1. Pilih Database: Setelah membuat database, pastikan Anda memilihnya dari panel di sebelah kiri.
2. Pilih Opsi "Impor": Di bagian atas, akan ada tab-menu. Pilih "Impor".
3. Pilih File SQL: Klik "Pilih File" dan cari file SQL yang ingin Anda impor.
4. Atur Opsi Impor (Opsional): Anda mungkin perlu menyesuaikan beberapa pengaturan impor, seperti pengaturan karakter atau jenis file.



1. Konfirmasi Impor: Klik "Go" atau "Impor" untuk memulai proses impor.

## 11.3. koneksi python dan mysql

Untuk membuat koneksi antara Python dan MySQL, Anda dapat menggunakan modul mysql-connector-python. Pastikan Anda telah menginstal modul ini sebelum mencoba menghubungkan Python dengan server MySQL. Berikut adalah langkah-langkah untuk membuat koneksi antara Python dan MySQL:

1. Instalasi mysql-connector-python:

Pastikan Anda telah menginstal modul mysql-connector-python. Jika belum, Anda dapat menginstalnya menggunakan pip dengan menjalankan perintah berikut di terminal atau command prompt:

* pip install mysql-connector-python

1. Import mysql.connector:

Di dalam skrip Python Anda, import modul mysql.connector untuk menggunakan fungsionalitasnya.

1. Buat Koneksi:

Buat objek koneksi menggunakan fungsi mysql.connector.connect() dengan menyediakan parameter seperti nama host, nama pengguna, kata sandi, dan nama basis data yang ingin Anda hubungkan.

1. Contoh Kode:

Berikut adalah contoh kode untuk membuat mengecek koneksi antara Python dan MySQL:

import mysql.connector as mc

class DBConnection:

    def \_\_init\_\_(self):

        self.host = "localhost"

        self.port = 3306

        self.name = "pbo"

        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."

A = DBConnection()

B = A.info

print(B)

berikut merupakan tampilan jika konesi antara python pada visualcode sudah terhung dengan mysql database localhost

## 11.4. Membuat fungsi aplikasi data mahasiswa

Selanjutnya adalah membuat fungsi untuk aplikasi data mahasisawa,berikut merupakan sourcode untuk fungsi dari aplikasi data mahasiswa yang akan kita buat :

from db import DBConnection as mydb

class Mahasiswa:

    def \_\_init\_\_(self):

        self.\_\_id=None

        self.\_\_nim=None

        self.\_\_nama=None

        self.\_\_jk=None

        self.\_\_kode\_prodi=None

        self.conn = None

        self.affected = None

        self.result = None

    @property

    def id(self):

        return self.\_\_id

    @property

    def nim(self):

        return self.\_\_nim

    @nim.setter

    def nim(self, value):

        self.\_\_nim = 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 kode\_prodi(self):

        return self.\_\_kode\_prodi

    @kode\_prodi.setter

    def kode\_prodi(self, value):

        self.\_\_kode\_prodi = value

    def simpan(self):

        self.conn = mydb()

        val = (self.\_\_nim, self.\_\_nama, self.\_\_jk, self.\_\_kode\_prodi)

        sql="INSERT INTO mahasiswa (nim, nama, jk, kode\_prodi) VALUES " + str(val)

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

        self.conn.disconnect

        return self.affected

    def update(self, id):

        self.conn = mydb()

        val = (self.\_\_nim, self.\_\_nama, self.\_\_jk, self.\_\_kode\_prodi, id)

        sql="UPDATE mahasiswa SET nim = %s, nama = %s, jk=%s, kode\_prodi=%s WHERE id=%s"

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

        self.conn.disconnect

        return self.affected

    def updateByNIM(self, nim):

        self.conn = mydb()

        val = (self.\_\_nama, self.\_\_jk, self.\_\_kode\_prodi, nim)

        sql="UPDATE mahasiswa SET nama = %s, jk=%s, kode\_prodi=%s WHERE nim=%s"

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

        self.conn.disconnect

        return self.affected

    def delete(self, id):

        self.conn = mydb()

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

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

        self.conn.disconnect

        return self.affected

    def deleteByNIM(self, nim):

        self.conn = mydb()

        sql="DELETE FROM mahasiswa WHERE nim='" + str(nim) + "'"

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

        self.conn.disconnect

        return self.affected

    def getByID(self, id):

        self.conn = mydb()

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

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

        self.\_\_nim = self.result[1]

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

        self.\_\_jk = self.result[3]

        self.\_\_kode\_prodi = self.result[4]

        self.conn.disconnect

        return self.result

    def getByNIM(self, nim):

        a=str(nim)

        b=a.strip()

        self.conn = mydb()

        sql="SELECT \* FROM mahasiswa WHERE nim='" + b + "'"

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

        if(self.result!=None):

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

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

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

            self.\_\_kode\_prodi = self.result[4]

            self.affected = self.conn.cursor.rowcount

        else:

            self.\_\_nim = ''

            self.\_\_nama = ''

            self.\_\_jk = ''

            self.\_\_kode\_prodi = ''

            self.affected = 0

        self.conn.disconnect

        return self.result

    def getAllData(self):

        self.conn = mydb()

        sql="SELECT \* FROM mahasiswa"

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

        return self.result

A = Mahasiswa()

B = A.getAllData()

print(B)

Berikut adalah penjelasan singkat tentang kode fungsi aplikasi data mahasiswa:

1. Koneksi Database: Menggunakan kelas DBConnection sebagai mydb untuk mengelola koneksi database.
2. Kelas Mahasiswa:

* Memiliki atribut untuk menyimpan data mahasiswa.
* Metode untuk menyimpan, memperbarui, menghapus, dan mengambil data mahasiswa dari database.

1. Metode-metode Utama:

* simpan(): Menyimpan data mahasiswa ke database.
* update() dan updateByNIM(): Memperbarui data mahasiswa berdasarkan ID atau NIM.
* delete() dan deleteByNIM(): Menghapus data mahasiswa berdasarkan ID atau NIM.
* getByID() dan getByNIM(): Mengambil data mahasiswa berdasarkan ID atau NIM.
* getAllData(): Mengambil semua data mahasiswa dari database.

1. Output: Mengambil semua data mahasiswa dan mencetaknya.

## 11.5. Form aplikasi data mahasiswa

Berikut merupakan sourcode untuk form aplikasi data mahasiswa yang akan kita buat :

import tkinter as tk

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

from Mahasiswa import Mahasiswa

class FormMahasiswa:

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

        self.parent = parent

        self.parent.geometry("450x450")

        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='NIM:').grid(row=0, column=0, sticky=W, padx=5, pady=5)

        self.txtNIM = Entry(mainFrame)

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

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

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

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

        self.txtKodeProdi = StringVar()

        Cbo = ttk.Combobox(mainFrame, width = 27, textvariable = self.txtKodeProdi)

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

        Cbo['values'] = ('TIF','IND','PET')

        Cbo.current()

        # 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 = ('idmhs', 'nim', 'nama','jk','kode\_prodi')

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

        # define headings

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

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

        self.tree.heading('nim', text='NIM')

        self.tree.column('nim', 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('kode\_prodi', text='Kode Prodi')

        self.tree.column('kode\_prodi', width="100")

        # set tree position

        self.tree.place(x=0, y=200)

        self.onReload()

    def onClear(self, event=None):

        self.txtNIM.delete(0,END)

        self.txtNIM.insert(END,"")

        self.txtNama.delete(0,END)

        self.txtNama.insert(END,"")

        self.txtKodeProdi.set("")

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

        self.L.select()

        self.onReload()

        self.ditemukan = False

    def onReload(self, event=None):

        # get data mahasiswa

        mhs = Mahasiswa()

        result = mhs.getAllData()

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

            self.tree.delete(item)

        students=[]

        for row\_data in result:

            students.append(row\_data)

        for student in students:

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

    def onCari(self, event=None):

        nim = self.txtNIM.get()

        mhs = Mahasiswa()

        res = mhs.getByNIM(nim)

        rec = mhs.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):

        nim = self.txtNIM.get()

        mhs = Mahasiswa()

        res = mhs.getByNIM(nim)

        self.txtNama.delete(0,END)

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

        jk = mhs.jk

        if(jk=="P"):

            self.P.select()

        else:

            self.L.select()

        self.txtKodeProdi.set(mhs.kode\_prodi)

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

    def onSimpan(self, event=None):

        nim = self.txtNIM.get()

        nama = self.txtNama.get()

        jk = self.txtJK.get()

        prodi = self.txtKodeProdi.get()

        mhs = Mahasiswa()

        mhs.nim = nim

        mhs.nama = nama

        mhs.jk = jk

        mhs.kode\_prodi = prodi

        if(self.ditemukan==True):

            res = mhs.updateByNIM(nim)

            ket = 'Diperbarui'

        else:

            res = mhs.simpan()

            ket = 'Disimpan'

        rec = mhs.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):

        nim = self.txtNIM.get()

        mhs = Mahasiswa()

        mhs.nim = nim

        if(self.ditemukan==True):

            res = mhs.deleteByNIM(nim)

            rec = mhs.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 = FormMahasiswa(root, "Aplikasi Data Mahasiswa")

    root.mainloop()

Berikut adalah penjelasan singkat tentang kode form aplikasi data mahasiswa

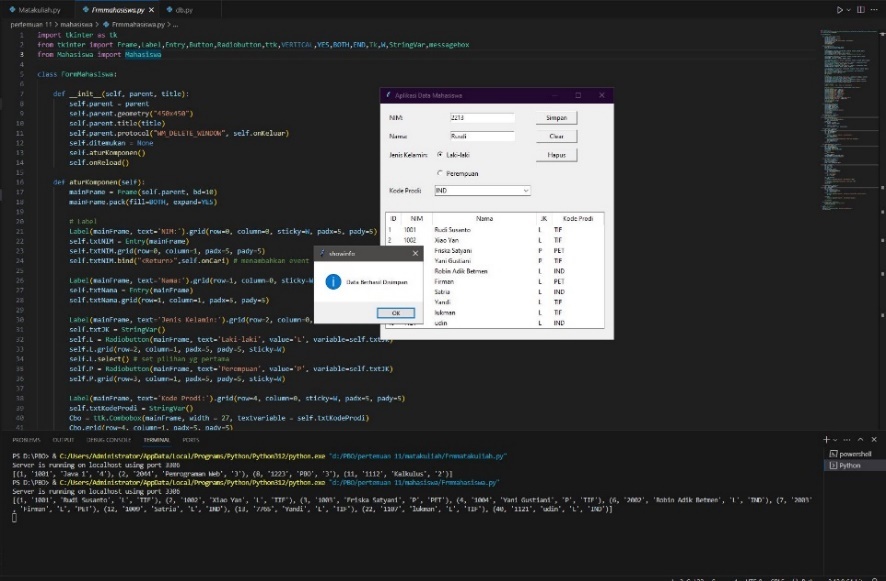
1. Import dan Konfigurasi Tkinter: Kode mengimpor modul tkinter untuk membuat antarmuka pengguna (GUI) dan mengatur ukuran dan judul jendela aplikasi.
2. Kelas FormMahasiswa:

* Inisialisasi: Menentukan tata letak awal dan menghubungkan metode ke acara jendela.
* Atur Komponen: Membuat dan menempatkan elemen-elemen GUI seperti label, entry, button, dan treeview di jendela.

1. Metode Pengelola Acara Utama:

* onClear(): Menghapus inputan di form.
* onReload(): Memuat ulang data mahasiswa dari database ke dalam treeview.
* onCari(): Mencari data mahasiswa berdasarkan NIM.
* TampilkanData(): Menampilkan data mahasiswa yang ditemukan pada form.
* onSimpan(): Menyimpan atau memperbarui data mahasiswa ke database.
* onDelete(): Menghapus data mahasiswa dari database.
* onKeluar(): Menutup aplikasi.

1. Interaksi dengan Kelas Mahasiswa: Form berinteraksi dengan kelas Mahasiswa untuk melakukan operasi CRD pada data mahasiswa.
2. Penggunaan MessageBox: Digunakan untuk menampilkan informasi atau peringatan kepada pengguna.
3. Memulai Aplikasi: Membuat instance dari kelas FormMahasiswa dan menjalankan aplikasi dengan root.mainloop().

Berikut merupakan tampilan aplikasi data mahasiswa yang sudah di buat, kita dapat mencobanya apakah semua fungsi yang di gunakan dapat berjalan dengan semestinya dan dapat menyimpan nya pada database yang sudah kita buat sebelum nya, jika tidak terdapat masalah maka tampilan nya akan seperti di bawah ini

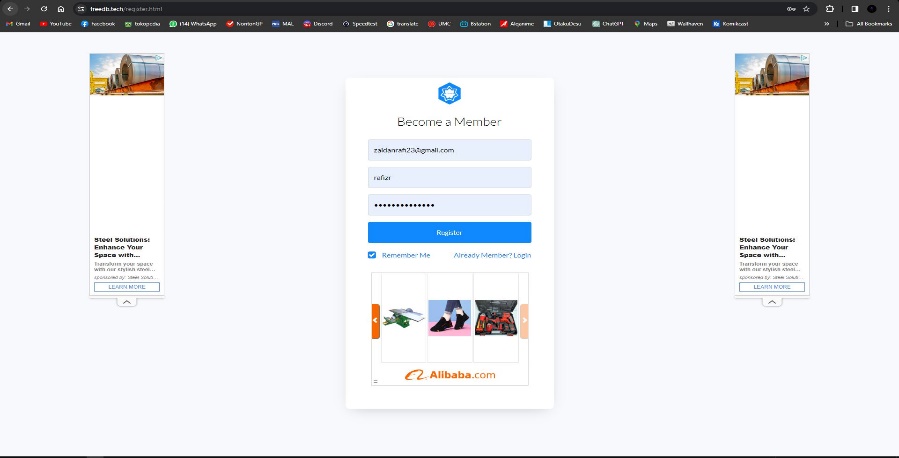
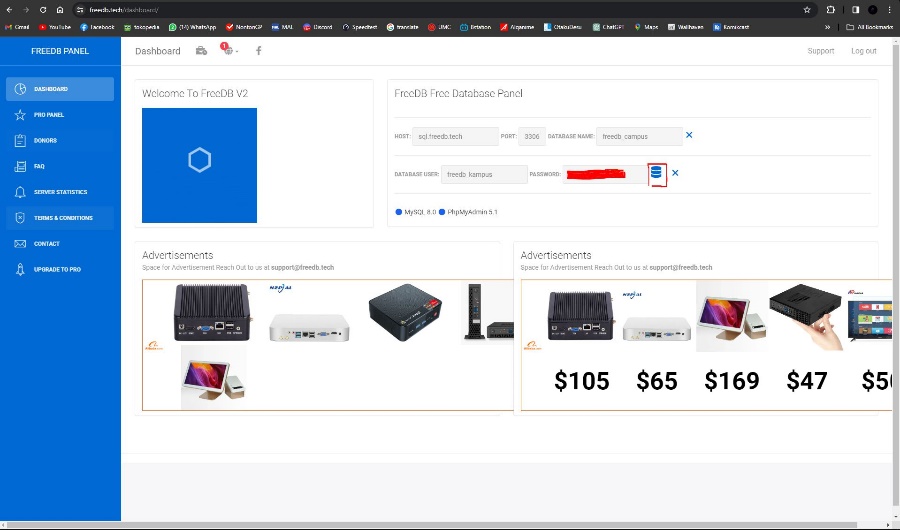
# PERTEMUAN 12: PYTHON APPLICATION DATABASE SERVER & LATIHAN PRAKTIKUM

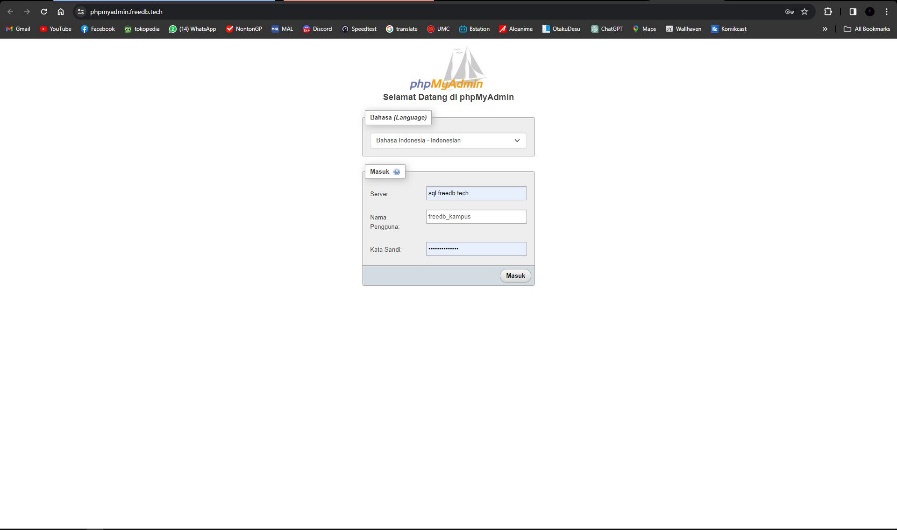
## 12.1. Pengembangan Aplikasi Data Mahasiswa Berbasis Database Online Menggunakan Freedb

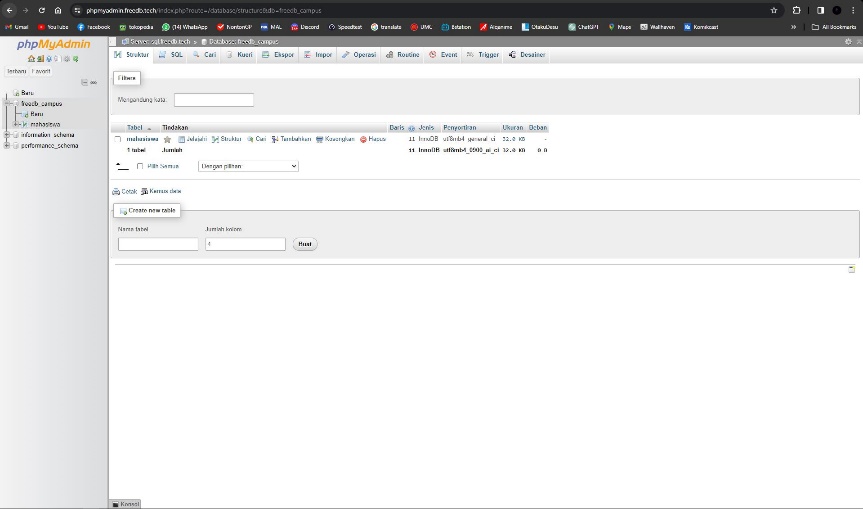
Pada bagian ini, mahasiswa akan diperkenalkan dengan konsep pengembangan aplikasi yang berbasis database online menggunakan freedb dalam mendapatkan MySQL Hosting gratis yang bisa diakses atau dikoneksi secara remote. Database hosting gratis ini biasanya digunakan untuk testing atau devlopment antara aplikasi dengan sistem database yang terhubung secara online. Sebenarnya untuk melakukan koneksi database MySQL secara online ini tidak ada bedanya dengan koneksi database MySQL yang terletak server lokal seperti XAMPP, LAMP, dan sebagainya. Bedanya cuma bisa dilakukan remote sehingga satu database MySQL dapat diakses oleh beberapa perangkat atau hosting yang berbeda.

Sayangnya, koneksi Database MySQL secara remote dan online ini bisa membahayakan data informasi yang tersimpan di dalam sistem tersebut jika tidak menutupi celah lubang yang menyebabkan kebobolan data yang dilakukan oleh Cyber IT. Jadi, MySQL hosting gratis ini digunakan untuk belajar dalam melakukan koneksi database MySQL ke dalam aplikasi secara remote. Mereka akan mempelajari bagaimana menggunakan Python untuk menghubungkan aplikasi ke database online dan melakukan operasi (Create, Read, Delete) pada data mahasiswa.

## 12.2. Latihan Praktikum Praktikum aplikasi data mahasiswa database menggunakan freedb

1. Kunjungi situs https://freedb.tech/untuk melakukan daftar akun.
2. Isikan formulir yang berupa email, username, dan password dengan tepat dan benar, klik Register.
3. Isikan formulir untuk survei akun dengan sesuai jawaban kamu, klik Submit.
4. Kunjungi situs akun gmail yang didaftarkan akun FreeDB.Tech tadi untuk melakukan verifikasi akun.
5. Buka isi kotak masuk yang pengirim dari FreeDB.Tech.
6. Klik link “Verify Your Account”.
7. Buatlah database baru & nama pengguna baru dengan sesuai kebutuhan kamu, klik Create DB & Create User.
8. Berikut hasil dalam pembuatan database baru dan Pengguna baru.
9. Apabila kamu ingin mengakses PHPMyAdmin, bisa klik icon database yang terdapat pada gambar di atas.
10. Masukkan pengisian data nama server (host), nama pengguna, beserta kata sandi yang didapatkan dari dashboard FreeDB.Tech tadi, klik Kirim.



1. Berikut merupakan hasil tampilan dari database mysql server phpMyAdmin. disini kita dapat mengimport sql untuk ditambahkan menjadi database dari aplikasi data mahasiswa yang akan kita buat sama seperti di pertemuan 11 sebelum nya.

## 12.3. Koneksi Python Dan Mysql Dabase Server Online

1. Buat Koneksi:

Sama seperti membuat koneksi seperti di pertemuan 10 sebelumnya yang menguunakan xampp sebagai server local(localhost). buat objek koneksi menggunakan fungsi mysql.connector.connect() dengan menyediakan parameter seperti nama host, nama pengguna, kata sandi, dan nama basis data sesuai dengan yang tertera pada halaman dashboard freedb.

1. Contoh Kode:

Berikut adalah contoh kode untuk membuat mengecek koneksi antara Python dan MySQL databse server online freedb:

import mysql.connector as mc

class DBConnection:

    def \_\_init\_\_(self):

        self.host = "sql.freedb.tech"

        self.port = 3306

        self.name = "freedb\_campus"

        self.user = "freedb\_kampus"

        self.password = "!4AtjpH78DfxCNv"

        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)

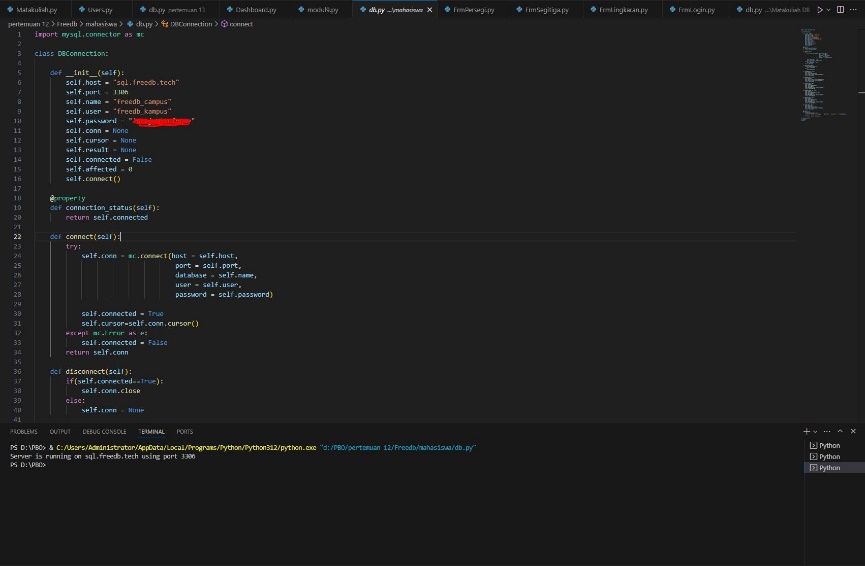
        else:

            return "Server is offline."

A = DBConnection()

B = A.info

print(B)

berikut merupakan tampilan jika konesi antara python pada visualcode sudah terhung dengan mysql database mysql server



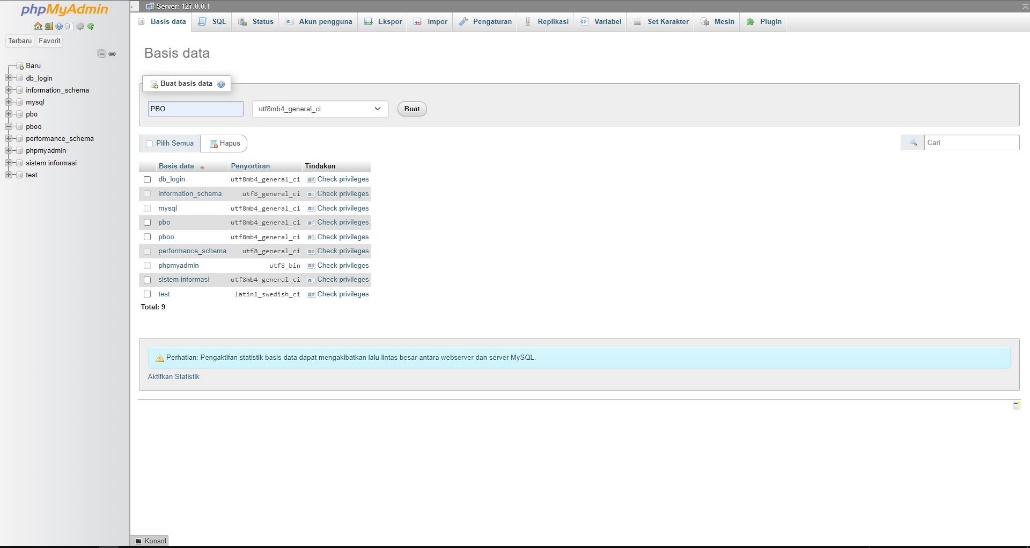
# PERTEMUAN 13: PYTHON APPLICATION DATABASE SERVER & LATIHAN PRAKTIKUM

## 13.1. Pengertian Aplikasi Python Login Multi User Berbasis Tkinter Mysql

Aplikasi Python Login Multi User berbasis Tkinter MySQL adalah sebuah aplikasi yang dibangun menggunakan bahasa pemrograman Python dan library Tkinter untuk membuat antarmuka grafis, serta MySQL sebagai basis data. Aplikasi ini memungkinkan pengguna untuk melakukan login ke dalam sistem dengan akun yang telah terdaftar sebelumnya. Fitur utama dari aplikasi ini adalah kemampuannya untuk mengelola informasi pengguna, seperti username, password, serta menyimpan informasi tersebut dalam basis data MySQL.

## 13.2. Membuat Database Login pada phpMyAdmin

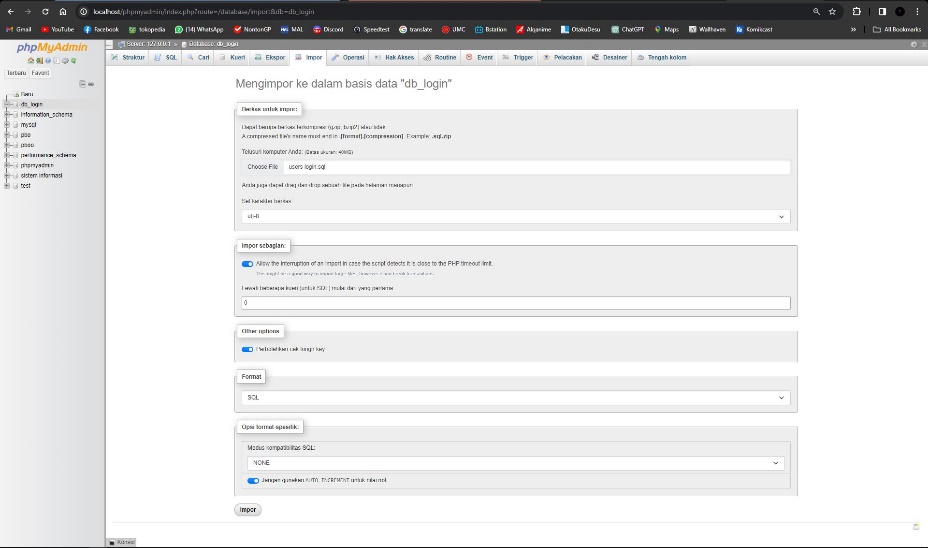
1. Login ke phpMyAdmin: Buka browser dan masuk ke panel phpMyAdmin menggunakan URL http://localhost/phpmyadmin/
2. Pilih Server: Jika memiliki beberapa server terdaftar, pilih server yang diinginkan.
3. Buat Database Baru: Pada panel sebelah kiri, Anda akan melihat opsi untuk "Buat Database". Klik di sana.
4. Isi Detail Database: Berikan nama database yang diinginkan karena kita disini akan membuat database login kita beri nama db\_login lalu klik buat

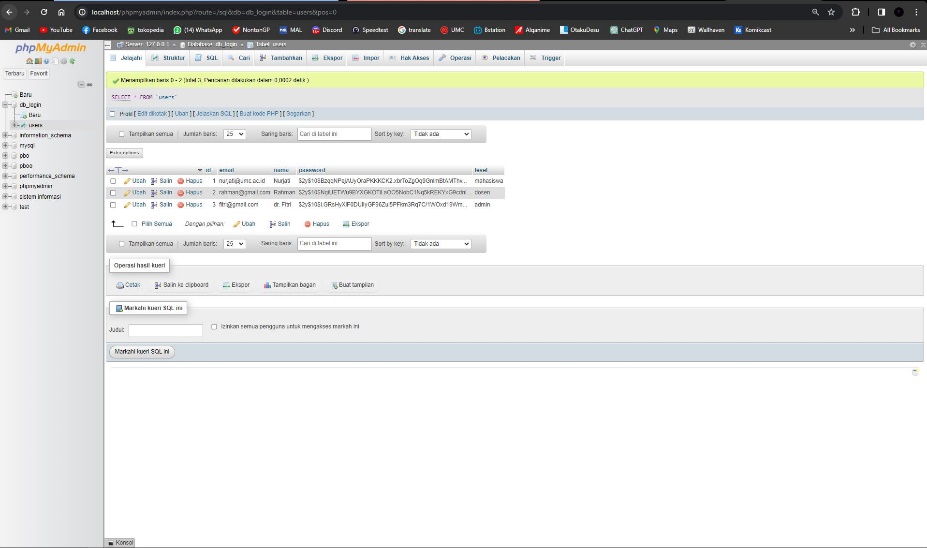


1. Konfirmasi: Klik "Buat" untuk membuat database baru.

## 13.3 Mengimport Tabel SQL ke dalam phpMyAdmin

1. Setelah membuat database, Pilih Database:db\_login pastikan Anda memilihnya dari panel di sebelah kiri.
2. Pilih Opsi "Impor": Di bagian atas, akan ada tab-menu. Pilih "Impor".
3. Pilih File SQL: Klik "Pilih File" dan cari file SQL yang ingin Anda impor yakni user untuk login. .



1. Konfirmasi Impor: Klik "Go" atau "Impor" untuk memulai proses impor.
2. Jika berhasil di impor maka akan muncul user table pada panel menu db\_login di sebelah kiri sebagai berikut
3. Disini kita dapat merubah dan menambahkan user login pada aplikasi bangun datar yang akan kita buat

## 13.4 koneksi python dan mysql

1. Import mysql.connector:

Di dalam skrip Python Anda, import modul mysql.connector untuk menggunakan fungsionalitasnya.

1. Buat Koneksi:

Buat objek koneksi menggunakan fungsi mysql.connector.connect() dengan menyediakan parameter seperti nama host, nama pengguna, kata sandi, dan nama basis data yang ingin Anda hubungkan.

1. Contoh Kode:

Berikut adalah contoh kode untuk membuat mengecek koneksi antara Python dan MySQL:

import mysql.connector as mc

class DBConnection:

    def \_\_init\_\_(self):

        self.host = "localhost"

        self.port = 3306

        self.name = "db\_login"

        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."

A = DBConnection()

B = A.info

print(B)

berikut merupakan tampilan jika konesi antara python pada visualcode sudah terhung dengan mysql database localhost

## 13.5 Membuat Fungsi User Login aplikasi bangun datar

1. Instalasi bcrypt:

Pastikan Anda telah menginstal modul bcrypt. bcrypt menyediakan fungsi untuk menghasilkan hash dari sebuah string dan memeriksa apakah string cocok dengan hash yang ada Jika belum, Anda dapat menginstalnya menggunakan pip dengan menjalankan perintah berikut di terminal atau command prompt:

* pip install bcrypt

1. Impor bcrypt:

Di dalam skrip Python Anda, import modul bcrypt untuk menggunakan fungsionalitasnya

Selanjutnya adalah membuat fungsi untuk user login agar mahasiswa dapat mengkakses halaman dashboard aplikasi hitung bangun datar:

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

Berikut adalah penjelasan singkat tentang kode fungsi Fungsi User Login aplikasi bangun datar:

1. Konstruktor dan Properties: Menginisialisasi objek Users dengan atribut seperti id, email, nama, password, dan level. Properties digunakan untuk mengakses dan mengubah nilai atribut dengan aman.
2. Metode cekUsername: Memeriksa keberadaan username (email) dalam database. Mengisi atribut objek Users dengan nilai dari hasil query jika ditemukan, jika tidak, mengembalikan False.
3. Metode cekPassword: Memeriksa kesesuaian password yang dimasukkan dengan password yang di-hash yang tersimpan dalam objek Users menggunakan bcrypt.
4. Metode Validasi: Menggabungkan hasil dari cekUsername dan cekPassword untuk memvalidasi login pengguna.
5. Pemanggilan Database: Menggunakan modul db untuk koneksi dan operasi database, serta menggunakan bcrypt untuk mengamankan penyimpanan password.

untuk mengecek apakah login sudah bisa dilakukan dan python sudah terkoneksi dengan database kita bisa menambahkan code di bawah ini

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\n")

jika username maupun password sesusai dengan data user yang tersimpan di database maka akan muncul tampilan seperti di bawah ini

## 13.6 Form Login Mahasiswa

Selanjut nya adalah membuat form login mahasiswa untuk dapat mengakses aplikasi bangun datar yang akan kita buat.berikut merupakan source code nya

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()

Berikut adalah penjelasan singkat tentang kode form login mahasiswa

1. Kelas FormLogin:

* Fungsi Inisialisasi objek FormLogin untuk membuat GUI form login dengan fungsi aturKomponen() yang menyiapkan komponen-komponen seperti label, input fields, dan tombol.
* Method onSubmit() untuk memvalidasi login saat tombol "Submit" ditekan, menggunakan objek Users.
* Method onKeluar() untuk menutup jendela login saat tombol "Cancel" ditekan.

1. Fungsi update\_main\_window:

* Fungsi yang dipanggil setelah login berhasil, namun hanya mencetak hasil ke konsol saat ini.

1. Validasi Login:

* Input email dan password diambil saat tombol "Submit" ditekan.
* Validasi login dilakukan dengan menggunakan objek Users dan method Validasi().
* Jika login berhasil, fungsi update\_main\_window dipanggil dengan hasil validasi dan jendela login ditutup. Jika gagal, pesan peringatan ditampilkan.

## 13.7. Form Bangun Datar Persegi, Segitiga Dan Lingkaran

Selanjut nya adalah membuat form aplikasi bangun datar persegi, segitiga dan lingkaran, berikut merupakan source code nya

1. Form Persegi

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\_\_':

    root = Tk()

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

    root.mainloop()

1. Form Segitiga

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\_\_':

    root = Tk()

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

    root.mainloop()

1. Form Lingkaran

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))

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

    def onKeluar(self, event=None):

        # memberikan perintah menutup aplikasi

        self.parent.destroy()

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

    root = Tk()

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

    root.mainloop()

## 13.8 Dashboard

Selanjutnya adalah mebuat dashboard berfungsi menggabungkan semua fungsionalitas dari form yang sudah di buat sebelumnya menjadi satu tampilan yang mudah diakses oleh Pengguna. Berikut merupakan source code dari dashboard yang akan di buat

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='Luas Persegi', command=lambda: self.new\_window("Luas Persegi", FrmPersegi))

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

        self.mahasiswa\_menu.add\_command(label='Luas Lingkaran', 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()

Berikut beberapa penjelasan singkat dari source code dashboard di atas

1. Import Library:

* Kode mengimpor modul tkinter sebagai tk, Menu, dan messagebox.
* Modul FrmLogin, FrmPersegi, FrmSegitiga, dan FrmLingkaran diimpor untuk mengakses kelas-kelas yang diperlukan.

1. Class Dashboard:

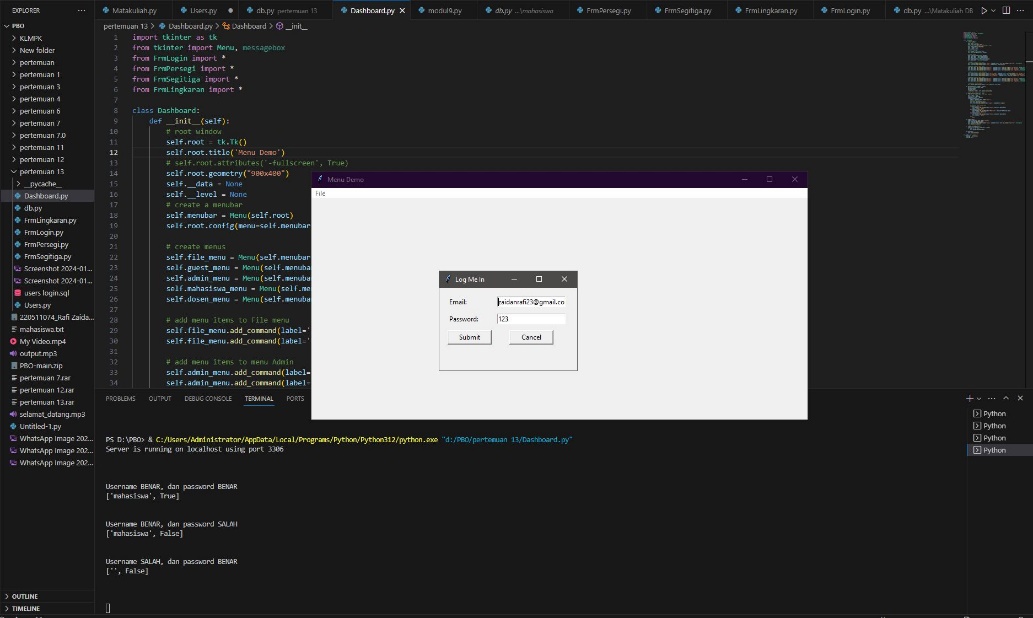
* Konstruktor \_\_init\_\_ menginisialisasi jendela utama (root), menyiapkan menu bar, dan menyimpan data level pengguna dan data hasil login.
* Berbagai menu (file\_menu, admin\_menu, mahasiswa\_menu, dan dosen\_menu) dan sub-menu ditambahkan ke menu bar (menubar).
* Terdapat method new\_window untuk membuat jendela baru saat opsi menu dipilih, dan update\_main\_window untuk mengupdate jendela utama berdasarkan data yang diterima dari jendela anak.
* Method Logout digunakan untuk mengubah menu setelah logout, dan remove\_all\_menus untuk menghapus menu sesuai level pengguna setelah logout.

1. Pembuatan Jendela Baru:

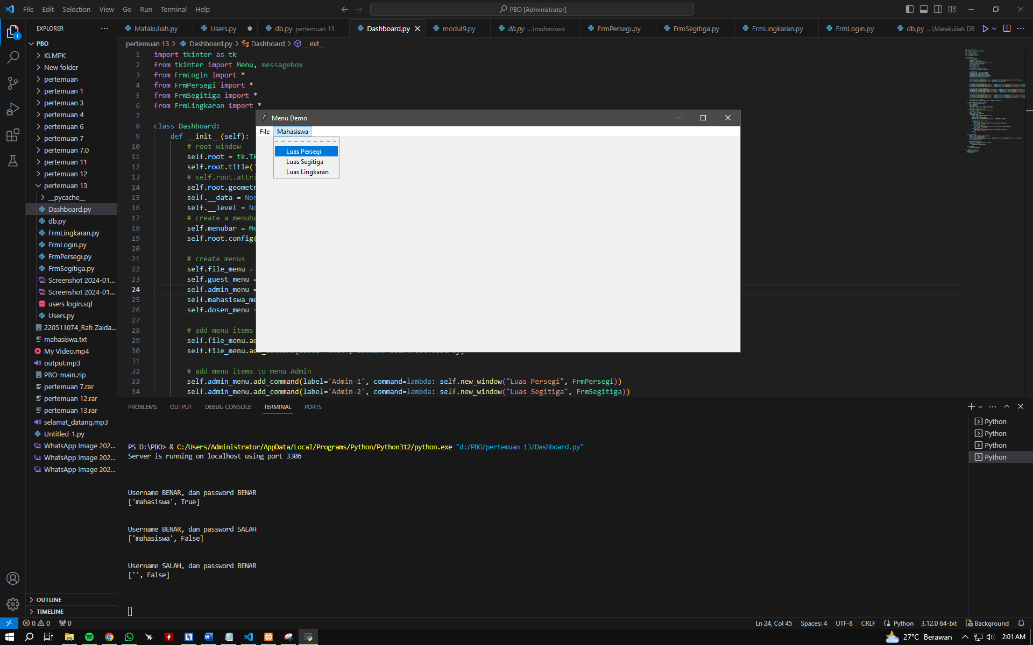
* Setiap kali pengguna memilih opsi menu, jendela baru dibuat untuk menampilkan fungsionalitas terkait (seperti login, menghitung luas persegi, segitiga, atau lingkaran).

berikut merupakan hasil tampilan tampilan aplikasi bangun datar mulai dari dashboard, form login dan penggungaan form persegi,segitiga serta lingkaran

1. Form Login:



1. Dasboard



1. Form bangun datar

