**MAKALAH  
PEMROGRAMAN 2 (PBO)**

**MODUL PEMBELAJARAN PERTEMUAN 1-14**

Dosen Pengampu : Freddy Wicaksono

****

**Disusun Oleh Kelompok 10:**

1. **Sapta Bayu Permana (220511169)**
2. **Safa Rizky Faddillah (220511008)**

**UNIVERSITAS MUHAMMADIYAH CIREBON**

**FAKULTAS TEKNIK  
PRODI TEKNIK INFORMATIKA**

**Jl. Fatahillah, Watubelah, Kec. Sumber, Kab. Cirebon, Jawa Barat**

**Kode Pos 45611**

**Telp/Fax. (0231) 204276 Web.** <https://www.umc.ac.id/>

**2023**

# Kata Pengantar

Pengembangan teknologi dan perangkat lunak telah membawa perubahan revolusioner dalam berbagai aspek kehidupan, termasuk di dunia pemrograman dan pengembangan aplikasi web. Aplikasi web telah menjadi salah satu alat yang paling umum digunakan dalam berbagai sektor, dari bisnis hingga pendidikan. Aplikasi web yang interaktif dan responsif telah menjadi penanda dalam memenuhi kebutuhan pengguna. Dengan semakin meningkatnya permintaan untuk antarmuka web yang menarik, kami memperkenalkan makalah ini, yang membahas pemrograman dengan menggunakan software python.

27 Januari 2024

Kelompok 10

# 

# Daftar Isi

[Kata Pengantar i](#_Toc157283760)

[Daftar Isi ii](#_Toc157283761)

[BAB I 1](#_Toc157283762)

[PENDAHULUAN 1](#_Toc157283763)

[1. 1. Latar Belakang 1](#_Toc157283764)

[1. 2. Tujuan 2](#_Toc157283765)

[BAB II 3](#_Toc157283766)

[PROGRAM BANGUN RUANG PYTHON 3](#_Toc157283767)

[3.1. Persegi 3](#_Toc157283768)

[3.2. Persegi Panjang 5](#_Toc157283769)

[BAB III 8](#_Toc157283770)

[MODUL FLET 8](#_Toc157283771)

[3.1 Pengertian Flet 8](#_Toc157283772)

[3.2 Persegi 8](#_Toc157283773)

[3.3 Persegi Panjang 9](#_Toc157283774)

[3.4 Segitiga 10](#_Toc157283775)

[3.5 Lingkaran 11](#_Toc157283776)

[BAB IV 12](#_Toc157283777)

[FUNCTION 12](#_Toc157283778)

[4.1 Cara Membuat Fungsi pada Python 12](#_Toc157283779)

[4.2 Fungsi dengan Parameter 13](#_Toc157283780)

[4.3 Contoh Program dengan Fungsi 15](#_Toc157283781)

[BAB V 20](#_Toc157283782)

[MULTI FORM 20](#_Toc157283783)

[5.1 Pengertian Multiple Function Arguments 21](#_Toc157283784)

[5.2 Install dan Import Library Googletrans 22](#_Toc157283785)

[5.3 Menerjemahkan kalimat 22](#_Toc157283786)

[5.4 Mendeteksi bahasa yang digunakan 24](#_Toc157283787)

[5.5 Cek cara pengucapan (pronunciation) 24](#_Toc157283788)

[BAB VI 24](#_Toc157283789)

[FUNGSI DAN PROSEDUR 24](#_Toc157283790)

[6.1 Cara Membuat Fungsi pada Python 25](#_Toc157283791)

[6.2 Fungsi dengan Parameter 25](#_Toc157283792)

[6.3 Fungsi yang Mengembalikan Nilai 26](#_Toc157283793)

[BAB VII 27](#_Toc157283794)

[APLIKASI BANTU (TOOLS) 27](#_Toc157283795)

[7.1 Apa saja tool yang harus diperlukan saat pemrograman? 28](#_Toc157283796)

[7.2 Tools Terbaik Python untuk Membuat Aplikasi 28](#_Toc157283797)

[7.3 BeeWare dan Bottle 28](#_Toc157283798)

[7.4 CherryPy dan Django 29](#_Toc157283799)

[7.5 Falcon dan Flask 29](#_Toc157283800)

[7.6 Kivy dan Pyramid 29](#_Toc157283801)

[7.7 Fungsi Lain dari Python 29](#_Toc157283802)

[Bab VIII 30](#_Toc157283803)

[Class & Objek 30](#_Toc157283804)

[8.1 Kegunaan Python Class. 31](#_Toc157283805)

[8.2 Perbedaan Python Class dengan Function. 31](#_Toc157283806)

[8.3 Cara menggunakan Python Class dan Object. 32](#_Toc157283807)

[8.4 Cara Inisialisasi dan Penanaman Variabel di Python. 33](#_Toc157283808)

[8.5 Cara Mengakses Python Class dan Object di Built-In Function 35](#_Toc157283809)

[BAB IX 36](#_Toc157283810)

[PENGENALAN MYSQL 36](#_Toc157283811)

[9.1 MySQL 37](#_Toc157283812)

[9.2 Server & Client MySQL 37](#_Toc157283813)

[9.3 Objek Basis Data 37](#_Toc157283814)

[9.4 Objek Tabel 37](#_Toc157283815)

[9.5 MySQL-Client 37](#_Toc157283816)

[9.6 Operasi Database 38](#_Toc157283817)

[9.7 Eksekusi File SQL 39](#_Toc157283818)

[9.8 Operasi Tabel 39](#_Toc157283819)

[9.9 Mengelola Field 40](#_Toc157283820)

[9.10 Manipulasi Data 40](#_Toc157283821)

[BAB X 41](#_Toc157283822)

[PYTHON DATABASE APPLICATIONS 41](#_Toc157283823)

[10.1 Membuat Database MySQL Menggunakan Python 41](#_Toc157283824)

[BAB XI 42](#_Toc157283825)

[PYTHON CRUD USING LIVE MYSQL SERVER 42](#_Toc157283826)

[11.1 Membuat Basis Data 43](#_Toc157283827)

[11.2 Membuat Tabel 44](#_Toc157283828)

[BAB XII 51](#_Toc157283829)

[LOGIN MULTI USER APPLICATION 51](#_Toc157283830)

[12.1 Membuat Login Multi User Level Dengan PHP & MySQL 51](#_Toc157283831)

[12.2 Membuat Database Multi User Level 51](#_Toc157283832)

[12.3 Membuat Login Multi User Level Dengan PHP & MySQL 54](#_Toc157283833)

# BAB I

# PENDAHULUAN

## Latar Belakang

Bahasa pemrograman saat ini jumlahnya sangat banyak. Python merupakan salah satu bahasa pemrograman populer yang digunakan oleh banyak developer. Menurut survei bahasa pemrograman versi www.tiobe.com, Python berada diperingkat ke-5 pada tahun 2016. Selain itu, Python juga bisa digunakan untuk enterprise. Dalam tingkatan bahasa pemrograman, Python termasuk high level language. Python menjadi salah satu bahasa pemrograman yang dapat digunakan untuk membangun aplikasi, baik itu berbasis desktop, web ataupun berbasis mobile. Untuk membangun aplikasi berbasis web bisa menggunakan framework atau tanpa framework. Apabila tanpa framework harus menggunakan CGI (Common Gateway Interface), serta untuk koneksi ke database harus ditulis pada setiap file yang membutuhkan koneksi ke database. Apabila harus selalu menulis kode program koneksi ke database pada setiap file, ini membuat pembangunan aplikasi menjadi lama. Selain itu apabila tanpa framework, penulisan kode program menjadi tidak terstruktur. Contohnya dalam sebuah file dapat berisi model, view, dan controller. Mungkin apabila membangun sebuah aplikasi yang sederhana tanpa menggunakan framework akan lebih mudah, tetapi apabila aplikasi yang dibangun adalah aplikasi yang besar maka dibutuhkan framework. Karena dengan framework aplikasi akan lebih mudah untuk dibangun dan di maintenance. Python memiliki beberapa web framework salah satunya adalah django.

Django merupakan sebuah web framework berbasis Python yang mendukung pembuatan sebuah website secara rapid development dengan desain yang elegan. Django merupakan web framework yang dirancang dan dibangun oleh Adrian Holovaty dan Jacob Kaplan Moss. Menurut survei framework python versi hotframeworks.com, framework Django berada diperingkat pertama. Berdasarkan penjelasan yang telah diuraikan diatas, maka penulis tertarik untuk melakukan eksplorasi python menggunakan framework Django. Eksplorasi yang dilakukan akan diimplementasikan pada sistem perpustakaan. Perpustakaan merupakan salah satu penyedia dan penyalur informasi yang dimana fungsi dan peranannya sangat berarti dan dibutuhkan dalam dunia informasi. Perpustakaan akan berkembang dengan baik apabila sistem teknologi yang digunakan mengikuti perkembangan teknologi saat ini. Dengan menggunakan aplikasi web maka akan mempermudah dalam pencarian informasi. Sistem perpustakaan yang akan dibangun meliputi pendaftaran anggota, pencatatan katalog buku, pencarian buku, peminjaman buku dan pengembalian buku.

## Tujuan

Tujuan utama dari makalah ini adalah untuk menggambarkan implementasi pemrograman bahasa Python. Kami bertujuan untuk memberikan kontribusi praktis dengan menciptakan alat yang dapat membantu penggunaan software python. Dengan implementasi ini, kami berharap dapat memberikan solusi yang efisien dan berguna dalam berbagai bidang, seperti ilmu matematika, arsitektur, dan rekayasa, serta sebagai alat pembelajaran yang interaktif.

Kami juga bertujuan untuk memberikan wawasan tentang penggunaan bahasa pemrograman Python dalam pengembangan aplikasi web yang interaktif. Python adalah bahasa pemrograman yang popular, serbaguna, dan menggunakannya dalam pembuatan aplikasi web memberikan fleksibilitas serta kemudahan dalam menghadapi berbagai tugas pemrograman. Tujuan kedua memberikan pemahaman tentang bagaimana Python dapat digunakan dalam mengatasi tantangan pengembangan aplikasi web dengan studi kasus konkret.

Tujuan kami, memperkenalkan penggunaan software python dalam pembangunan antarmuka pengguna yang menarik dan responsif.

# BAB II

# PROGRAM BANGUN RUANG PYTHON

## Persegi

Persegi merupakan bangun datar yang setiap sisi memiliki panjang dan sudut yang sama. Ada dua rumus yang bisa di hitung dari persegi yaitu luas dan kelilingnya.

Rumus Luas & Keliling Persegi

Luas = s2

Keliling = 4 x s

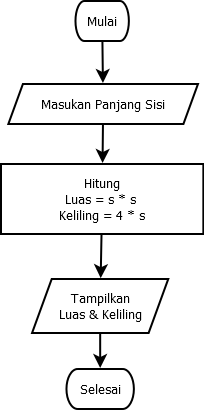
#### Keterangan:

s = sisi

Algoritma

* Mulai
* Masukan sisi (s)
* Hitung luas = s2
* Hitung keliling = 4 \* s
* Tampilkan Luas dan Keliling
* Selesai

Flowchart



Program Python Menghitung Luas & Keliling Persegi

print("MENGHITUNG LUAS & KELILING PERSEGI")

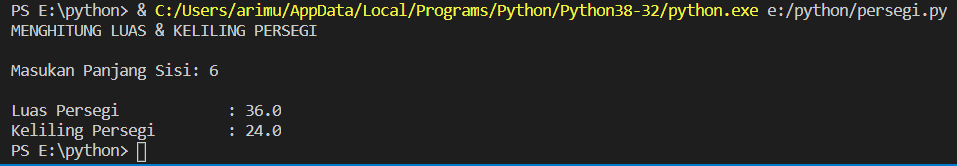
s = float(input("\nMasukan Panjang Sisi: "))

luas = s\*\*2

keliling = 4 \* s

print("\nLuas Persegi \t\t:",luas)

print("Keliling Persegi\t:",keliling)



Program diatas adalah program python untuk menghitung luas dan keliling persegi. Pengguna akan di minta memasukan panjang sisi persegi menggunakan fungsi **input()**yang telah di konversi ke tipe data float (bilangan real) kemudian nilainya di tampung pada variabel.

Sesuai dengan rumus luas dan keliling persegi sintak **luas = s\*\*2** berarti nilai s dikuadratkan 2. Sedangkan untuk mendapatkan nilai keliling maka panjang sisi akan di kalikan dengan 4. Program akan menampilkan hasil perhitungan dari rumus luas dan keliling persegi.

## Persegi Panjang

Persegi panjang juga termasuk bangun datar dua dimensi berbentuk segi empat yang mempunyai dua pasang sisi yang berhadapan dan sama panjang.

Rumus Luas & Keliling Persegi Panjang

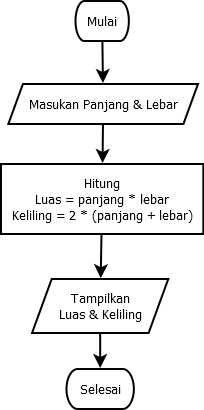
luas = panjang \* lebar

keliling = 2 \* (panjang + lebar)

Algoritma

* Mulai
* Masukan panjang
* Masukan lebar
* Hitung luas = panjang \* lebar
* Hitung keliling = 2 \* (panjang+lebar)
* Tampilkan luas dan keliling persegi panjang
* Selesai

Flowchart



Program Python Menghitung Luas dan Keliling Persegi Panjang

print("MENGHITUNG LUAS & KELILING PERSEGI PANJANG")

panjang = float(input("\nMasukan Panjang: "))

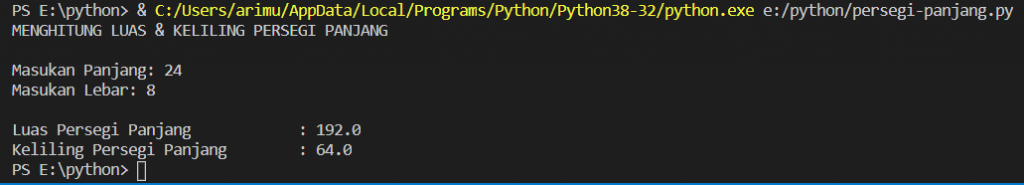
lebar = float(input("Masukan Lebar: "))

luas = panjang\*lebar

keliling = 2 \* (panjang+lebar)

print("\nLuas Persegi Panjang \t\t:",luas)

print("Keliling Persegi Panjang\t:",keliling)



Program diatas pengguna akan memasukan panjang dan lebar dari layar console saat program dijalankan menggunakan fungsi **input()** dan di konversi ke tipe data float. Ingat bahwa setiap nilai yang di masukan dari fungsi **input()**, program akan menganggap nilai tersebut menggunakan [tipe data](https://kelasprogrammer.com/tipe-data-python/) string (teks) sehingga jika kita tidak mengkonversi ke tipe data numerik maka operasi aritmatika tidak dapat kita lakukan.

Selanjutnya kita menghitung luas dan keliling persegi panjang sesuai dengan rumus-nya masing-masing.

Dua bagian sintak terakhir berfungsi untuk menampilkan hasil dari perhitungan rumus yang sudah di definisikan diatas. Perhatikan terdapat karakter**\n** yang berfungsi untuk membuat new line atau paragraf baru dan karakter **\t** untuk membuat tab. Kedua karakter tersebut termasuk dalam penggunaan fungsi string di python.

# BAB III

# MODUL FLET

## 3.1 Pengertian Flet

Flutter dan Python adalah dua hal yang berbeda dan tidak berkaitan satu sama lain. Flutter adalah kerangka pengembangan perangkat lunak open-source yang dikembangkan oleh Google untuk membuat antarmuka pengguna (UI) yang konsisten dan indah melalui berbagai platform, termasuk Android dan iOS. Flutter menggunakan bahasa pemrograman Dart.

Sementara itu, Python adalah bahasa pemrograman tingkat tinggi yang sering digunakan untuk pengembangan perangkat lunak umum, pengembangan web, analisis data, kecerdasan buatan, dan banyak keperluan lainnya. Python memiliki sintaksis yang mudah dibaca dan menekankan pada keterbacaan kode.

Jadi, sementara Flutter adalah kerangka kerja untuk mengembangkan antarmuka pengguna mobile, Python adalah bahasa pemrograman yang dapat digunakan untuk berbagai jenis pengembangan perangkat lunak. Tidak ada hubungan langsung antara Flutter dan Python, kecuali jika Anda mempertimbangkan penggunaan Python dalam proyek tertentu yang tidak terkait dengan Flutter.

## 3.2 Persegi

Rumus:

- Luas Persegi: \( \text{Luas} = sisi \times sisi \)

- Keliling Persegi: \( \text{Keliling} = 4 \times sisi \)

Implementasi Python:

python

def luas\_persegi(sisi):

return sisi \* sisi

def keliling\_persegi(sisi):

return 4 \* sisi

sisi\_persegi = 5

print("Luas Persegi:", luas\_persegi(sisi\_persegi))

print("Keliling Persegi:", keliling\_persegi(sisi\_persegi))

## 3.3 Persegi Panjang

Rumus:

- Luas Persegi Panjang: \( \text{Luas} = panjang \times lebar \)

- Keliling Persegi Panjang: \( \text{Keliling} = 2 \times (panjang + lebar) \)

* Implementasi Python:

def luas\_persegi\_panjang(panjang, lebar):

return panjang \* lebar

def keliling\_persegi\_panjang(panjang, lebar):

return 2 \* (panjang + lebar)

panjang\_pp = 4

lebar\_pp = 6

print("Luas Persegi Panjang:", luas\_persegi\_panjang(panjang\_pp, lebar\_pp))

print("Keliling Persegi Panjang:", keliling\_persegi\_panjang(panjang\_pp, lebar\_pp))

## 3.4 Segitiga

Rumus:

- Luas Segitiga: \( \text{Luas} = \frac{1}{2} \times alas \times tinggi \)

- Keliling Segitiga: \( \text{Keliling} = sisi\_1 + sisi\_2 + sisi\_3 \)

* Implementasi Python:

def luas\_segitiga(alas, tinggi):

return 0.5 \* alas \* tinggi

def keliling\_segitiga(sisi1, sisi2, sisi3):

return sisi1 + sisi2 + sisi3

alas\_segitiga = 8

tinggi\_segitiga = 5

sisi1\_segitiga = 3

sisi2\_segitiga = 4

sisi3\_segitiga = 5

print("Luas Segitiga:", luas\_segitiga(alas\_segitiga, tinggi\_segitiga))

print("Keliling Segitiga:", keliling\_segitiga(sisi1\_segitiga, sisi2\_segitiga, sisi3\_segitiga))

## 3.5 Lingkaran

Rumus:

- Luas Lingkaran: \( \text{Luas} = \pi \times jari\\_jari^2 \)

- Keliling Lingkaran: \( \text{Keliling} = 2 \times \pi \times jari\\_jari \)

* Implementasi Python:

import math

def luas\_lingkaran(jari\_jari):

return math.pi \* jari\_jari \*\* 2

def keliling\_lingkaran(jari\_jari):

return 2 \* math.pi \* jari\_jari

jari\_jari\_lingkaran = 6

print("Luas Lingkaran:", luas\_lingkaran(jari\_jari\_lingkaran))

print("Keliling Lingkaran:", keliling\_lingkaran(jari\_jari\_lingkaran))

# BAB IV

# FUNCTION

## 4.1 Cara Membuat Fungsi pada Python

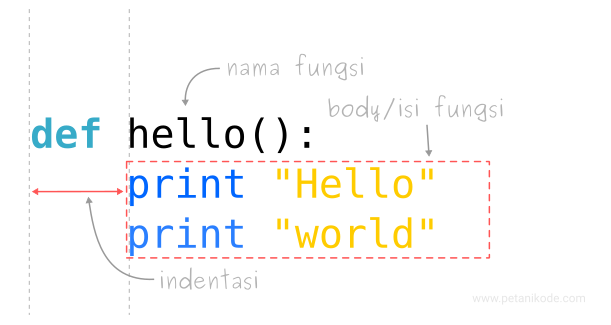
Fungsi pada Python, dibuat dengan kata kunci **def** kemudian diikuti dengan nama fungsinya.

Contoh:

def nama\_fungsi():

*print* "Hello ini Fungsi"

Sama seperti blok kode yang lain, kita juga harus memberikan identasi (tab atau spasi 2x) untuk menuliskan isi fungsi.



Setelah kita buat fungsinya, lalu apa?

Setelah kita buat, kita bisa memanggilnya seperti ini:

nama\_fungsi()

Sebagai contoh, coba tulis kode program berikut:

# Membuat Fungsi

def salam():

*print* "Hello, Selamat Pagi"

## Pemanggilan Fungsi

salam()

Hasilnya:

Hello, Selamat Pagi

Coba panggil sebanyak 3x:

# Membuat Fungsi

def salam():

*print* "Hello, Selamat Pagi"

## Pemanggilan Fungsi

salam()

salam()

salam()

Hasilnya:

Hello, Selamat Pagi

Hello, Selamat Pagi

Hello, Selamat Pagi

Mudah bukan?

Intinya apapun yang ada di dalam fungsi, ketika dipanggil itulah yang akan dilakukan.

**FYI:** fungsi juga dapat dipanggil pada fungsi lain, bahkan bisa memanggil dirinya sendiri. Fungsi yang memanggil dirinya sendiri, disebut [fungsi rekursif](https://www.petanikode.com/fungsi-rekursif/).

## 4.2 Fungsi dengan Parameter

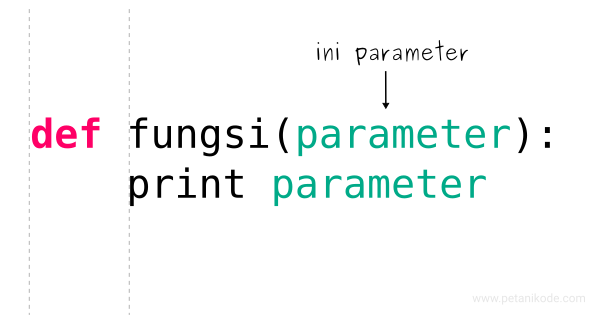
Sekarang, bagaimana kalau kita ingin memberikan input nilai ke dalam fungsi?

Gampang…

Kita bisa manfaatkan parameter.

Apa itu parameter?

Parameter adalah variabel yang menampung nilai untuk diproses di dalam fungsi.



Contoh:

def salam(ucapan):

*print*(ucapan)

Pada contoh diatas, kita membuat fungsi dengan parameter **ucapan**.

Lalu bagaimana cara memanggilnya?

Cara pemanggilan fungsi yang memiliki parameter adalah seperti ini:

salam("Selamat siang")

**"Selamat siang"** adalah nilai parameter yang kita berikan.

Lalu bagaimana kalau parameternya lebih dari satu?

Kita bisa menggunakan tanda koma (**,**) untuk memisahnya.

Contoh:

# Membuat fungsi dengan parameter

def luas\_segitiga(alas, tinggi):

luas = (alas \* tinggi) / 2

*print* "Luas segitiga: %f" % luas;

# Pemanggilan fungsi

luas\_segitiga(4, 6)

Hasilnya:

Luas segitiga: 12.000000

## 4.3 Contoh Program dengan Fungsi

Biaklah, sekarang tiba saatnya kita membuat program…

Silahkan buat file baru bernama **program\_fungsi.py**.

Lalu kita mulai tulis kodenya.

Pertama kita buat sebuah variabel global berupa list untuk menampung judul-judul buku.

# Variabel global untuk menyimpan data Buku

buku = []

Nanti program ini akan mampu melakukan operasi CRUD (Create, Read, Update, dan Delete). Maka kita membutuhkan fungsi-fungsi berikut:

* **show\_data()** untuk menampilkan data dari list **buku**;
* **insert\_data()** untuk menambahkan data ke list **buku**;
* **edit\_data()** untuk mengedit data di list **buku**;
* **delete\_data()** untuk untuk menghapus data dari list **buku**.

Mari kita buat semuanya.

Dimulai dari fungsi **show\_data()**:

# fungsi untuk menampilkan semua data

def show\_data():

if *len*(buku) <= 0:

*print* "BELUM ADA DATA"

else:

for indeks in *range*(*len*(buku)):

*print* "[%d] %s" % (indeks, buku[indeks])

Fungsi di atas akan mengecek isi dari list **buku**. Jika isinya kosong (**len(buku) <= 0**) maka tampilkan **"BELUM ADA DATA"**.

Namun, apabila ada isinya, maka tampilkan semua isinya dengan perulangan.

Selanjutnya membuat fungsi **insert\_data()**:

# fungsi untuk menambah data

def insert\_data():

buku\_baru = raw\_input("Judul Buku: ")

buku.append(buku\_baru)

Fungsi di atas akan mengambil input dari user kemudian diisi ke dalam list **buku** dengan fungsi **append()**.

Fungsi **append()** adalah fungsi untuk menambahkan item di akhir list.

Selain **append()** ada juga **prepend()**. Namun, untuk kasus ini, kita pakai **append()** saja.

Penjelasan lengkap tentang **append()** dan **prepend()** bisa dibaca pada: [Mengenal Struktur Data List pada Python](https://www.petanikode.com/python-list).

Selanjutnya membuat fungsi **edit\_data()**:

# fungsi untuk edit data

def edit\_data():

show\_data()

indeks = *input*("Inputkan ID buku: ")

if(indeks > *len*(buku)):

*print* "ID salah"

else:

judul\_baru = raw\_input("Judul baru: ")

buku[indeks] = judul\_baru

Fungsi di atas akan menampilkan isi dari list **buku** dengan memanggil fungsi **show\_data()** di dalamnya.

Setelah itu, kita meminta user untuk menginputkan ID atau nomer indeks buku yang akan diedit.

Lalu kita lakukan pengecekan, jika ID yang diinputkan melebihi dari isi list **buku** (**indeks > len(buku)**), maka tampilkan pesan **"ID salah"**.

Namun, apabila tidak melebihi dari isi **buku**, maka ambil input untuk judul baru dan simpan sesuai ID-nya.

Selanjutnya membuat fungsi **delete\_data()**:

# fungsi untuk menhapus data

def delete\_data():

show\_data()

indeks = *input*("Inputkan ID buku: ")

if(indeks > *len*(buku)):

*print* "ID salah"

else:

buku.remove(buku[indeks])

Hampir sama dengan fungsi **edit\_data()**. Fungsi **delete\_data()** juga harus menampilkan isi list **buku** dan mengambil ID yang akan dihapus.

Kita dapat menghapus item pada list dengan fungsi **remove()**.

Apakah sudah selesai?

Belum, masih ada dua fungsi lagi yang kita butuhkan:

Fungsi untuk menampilkan menu

Fungsi untuk keluar (sudah ada di python: **exit()**)

Ok, mari kita buat:

# fungsi untuk menampilkan menu

def show\_menu():

*print* "\n"

*print* "----------- MENU ----------"

*print* "[1] Show Data"

*print* "[2] Insert Data"

*print* "[3] Edit Data"

*print* "[4] Delete Data"

*print* "[5] Exit"

menu = *input*("PILIH MENU> ")

*print* "\n"

if menu == 1:

show\_data()

elif menu == 2:

insert\_data()

elif menu == 3:

edit\_data()

elif menu == 4:

delete\_data()

elif menu == 5:

exit()

else:

*print* "Salah pilih!"

Fungsi di atas akan menampilkan menu dari 1–5, lalu memanggil fungsi-fungsi yang sudah dibuat berdasarkan menu yang dipilih.

Terakhir, kita harus membuat main loop programnya.

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

while(True):

show\_menu()

Program akan mengulang terus-menerus sampai fungsi **exit()** dieksekusi.

**if \_\_name\_\_ == "\_\_main\_\_":** adalah blok main di Python. Sebenarnya tanpa ini, programnya sudah bisa dijalankan.

Tapi biar cakep, kita tambahkan saja.

Sehingga kode lengkapnya akan seperti ini:

# Variabel global untuk menyimpan data Buku

buku = []

# fungsi untuk menampilkan semua data

def show\_data():

if *len*(buku) <= 0:

*print* "BELUM ADA DATA"

else:

for indeks in *range*(*len*(buku)):

*print* "[%d] %s" % (indeks, buku[indeks])

# fungsi untuk menambah data

def insert\_data():

buku\_baru = raw\_input("Judul Buku: ")

buku.append(buku\_baru)

# fungsi untuk edit data

def edit\_data():

show\_data()

indeks = *input*("Inputkan ID buku: ")

if(indeks > *len*(buku)):

*print* "ID salah"

else:

judul\_baru = raw\_input("Judul baru: ")

buku[indeks] = judul\_baru

# fungsi untuk menhapus data

def delete\_data():

show\_data()

indeks = *input*("Inputkan ID buku: ")

if(indeks > *len*(buku)):

*print* "ID salah"

else:

buku.remove(buku[indeks])

# fungsi untuk menampilkan menu

def show\_menu():

*print* "\n"

*print* "----------- MENU ----------"

*print* "[1] Show Data"

*print* "[2] Insert Data"

*print* "[3] Edit Data"

*print* "[4] Delete Data"

*print* "[5] Exit"

menu = *input*("PILIH MENU> ")

*print* "\n"

if menu == 1:

show\_data()

elif menu == 2:

insert\_data()

elif menu == 3:

edit\_data()

elif menu == 4:

delete\_data()

elif menu == 5:

exit()

else:

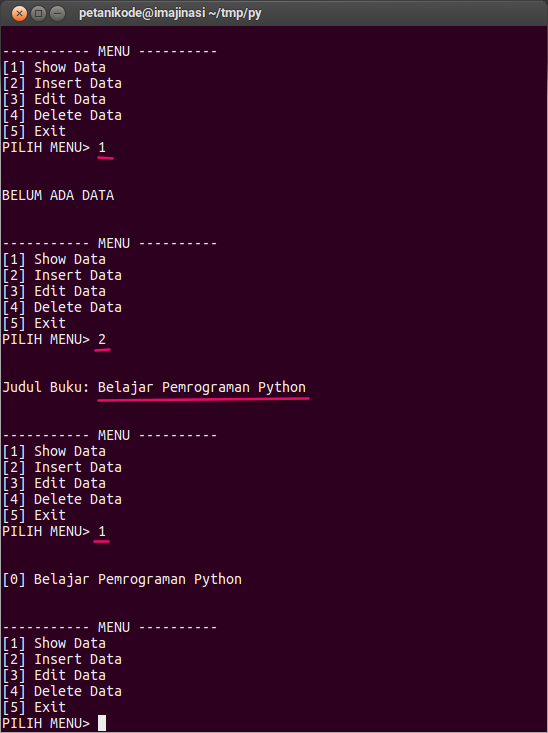
*print* "Salah pilih!"

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

while(True):

show\_menu()

Cobalah untuk mengeksekusinya dan perhatikan hasilnya.



# BAB V

# MULTI FORM

## 5.1 Pengertian Multiple Function Arguments

Sebelum lebih jauh membahas tentang Multiple Function Arguments, Anda mesti paham terlebih dahulu tentang apa itu argument. Argument adalah nilai yang dikirim kepada fungsi ketika argument tersebut dipanggil. Dalam bahasa Python, argument dapat memiliki nilai default.

Selain argument, Anda mungkin pernah mendengar istilah parameter dalam bahasa Python. Bagi Anda yang belum tahu, parameter / argumen adalah dua hal yang berbeda, namun perbedaannya sangatlah tipis. Umumnya ketika orang menyebut parameter / argumen, mereka memaksudkan dua hal yang sama.

Perbedaan utama antara parameter / argumen adalah parameter adalah hal yang dideklarasikan dalam fungsi, sedangkan argumen adalah hal yang dilewatkan saat memanggil fungsi. Keduanya sangat mirip bukan? Berhati-hatilah ketika Anda bekerja dengan parameter / argumen.

Dalam function atau fungsi Python, beberapa argumen diterima sebagai tupel. Dalam contoh, sebuah tupel diteruskan ke fungsi sum() untuk menghitung jumlah nilai. Fungsi adalah versi Python dari rutinitas dalam suatu program. Beberapa fungsi dirancang untuk mengembalikan nilai, sementara yang lain dirancang untuk tujuan lain.

Kami meneruskan argumen dalam suatu fungsi, kami tidak dapat meneruskan argumen sama sekali, argumen tunggal atau banyak argumen ke suatu fungsi dan dapat memanggil fungsi beberapa kali. Setiap fungsi dalam Python menerima sejumlah argumen yang telah ditentukan sebelumnya. Dimungkinkan untuk mendeklarasikan fungsi yang menerima sejumlah variabel argumen.

Dalam pemrograman tingkat tinggi, seringkali diperlukan untuk mendefinisikan fungsi yang dapat mengambil lebih dari satu argumen. Format multiple function argumen adalah sebagai berikut:

def function\_name(arg\_pertama, arg\_kedua, arg\_ketiga):

Jika Anda telah memahami dasar-dasar fungsi, maka fungsi dengan banyak argumen seharusnya tidak menjadi masalah bagi Anda. Berikut ini  kami akan menjelaskan contoh fungsi dengan banyak argumen.

## 5.2 Install dan Import Library Googletrans

Pertama silahkan buka google colab kamu, kemudian buat sebuah cell baru untuk menuliskan kode program. Kemudian install library googletrans dengan menggunakan cara berikut:

!pip install googletrans=='3.1.0a0'

Setelah itu silahkan import class Translator() ke google colab kalian seperti berikut:

**from** googletrans **import** Translator

Class ini memiliki beberapa function dan method didalamnya. Dalam tutorial ini, kita akan bekerja menggunakan class ini.

## 5.3 Menerjemahkan kalimat

Kamu bisa gunakan methode translate() pada class Translator() untuk menerjemahkan kalimat. Cara kerjanya adalah sebagai berikut:

1. Google akan mendeteksi bahasa yang digunakan pada kalimat tersebut secara otomatis
2. Google akan menerjemahkannya kedalam bahasa Inggris.

Kode program yang bisa kamu gunakan adalah sebagai berikut:

cek = Translator().translate('Der Himmel ist blau und ich mag Bananen')

cek.text

#Hasil: The sky is blue and I like bananas

Namun kita bisa juga loh menggunakan method ini untuk menerjemahkan kalimat kedalam bahasa lain selain bahasa Inggris. Kita tingga menambahkan property "dest" pada method tersebut yang berisi kode bahasa tujuan terjemahan.

Misal, kita ingin menerjemahkannya kedalam bahasa Kore, menurut ISO-639-1, kode bahasa Korea adalah 'ko' sehingga kita bisa menerjemahkannya menggunakan kode berikut:

cek = Translator().translate(

'Der Himmel ist blau und ich mag Bananen',

dest="ko")

cek.text

#Hasil: Ã­â€¢ËœÃ«Å ËœÃ¬Ââ‚¬ Ã­Å’Å’Ã«Å¾â€”ÃªÂ³  Ã«â€šËœÃ«Å â€ Ã«Â°â€Ã«â€šËœÃ«â€šËœÃ«Â¥Â¼ Ã¬Â¢â€¹Ã¬â€¢â€žÃ­â€¢Â´

Misal, kita ingin menerjemahkannya kedalam bahasa Indonesia, menurut ISO-639-1, kode bahasa Indonesia adalah 'id' sehingga kita bisa menerjemahkannya menggunakan kode berikut:

cek = Translator().translate(

'Der Himmel ist blau und ich mag Bananen',

dest="id")

cek.text

#Hasil: Langitnya biru dan aku suka pisang

Nah, kalian juga bisa loh menerjemahkan kata atau kalimat dari dan ke bahasa apapun. Yang penting, kalian tau kode bahasa internasionalnya menurut ISO 639-1. List nya bisa kalian lihat pada artikel dibawah.

## 5.4 Mendeteksi bahasa yang digunakan

Nah, kalau tadi kita sudah berhasil menerjemahkannya, kalian bingung gak sih bahasa apa yang sebenarnya digunakan pada contoh kalimat tadi. Yuk kita cari tau dengan menggunakan method detect(). Kalian bisa gunakan kode berikut:

cek = Translator().detect('Der Himmel ist blau und ich mag Bananen')

cek.lang

#Hasil: 'de'

Dari sini kita bisa melihat bahwa kode bahasa yang terdeteksi adalah 'de' yang menurut ISO-639-1 adalah bahasa Jerman.

## 5.5 Cek cara pengucapan (pronunciation)

Lalu bagaimana dengan pronunciation-nya? Misal kita menerjemahkannya ke Korea dan ingin tau cara pengucapannya. Pada method translate() kita bisa memanggil key 'pronunciation' seperti berikut:

cek = Translator().translate(

'The sky is blue and I like bananas',

dest="ko")

cek.pronunciation

#Hasil: haneul-eun palahgo naneun bananaleul joh-ahae

# BAB VI

# FUNGSI DAN PROSEDUR

## 6.1 Cara Membuat Fungsi pada Python

Fungsi pada Python, dibuat dengan kata kunci **def** kemudian diikuti dengan nama fungsinya.

Contoh:

def nama\_fungsi():

*print* "Hello ini Fungsi"

Sama seperti blok kode yang lain, kita juga harus memberikan identasi (tab atau spasi 2x) untuk menuliskan isi fungsi.

Setelah kita buat fungsinya, lalu apa?

Setelah kita buat, kita bisa memanggilnya seperti ini:

nama\_fungsi()

Sebagai contoh, coba tulis kode program berikut:

# Membuat Fungsi

def salam():

*print* "Hello, Selamat Pagi"

## Pemanggilan Fungsi

salam()

Hasilnya:

Hello, Selamat Pagi

Coba panggil sebanyak 3x:

# Membuat Fungsi

def salam():

*print* "Hello, Selamat Pagi"

## Pemanggilan Fungsi

salam()

salam()

salam()

Hasilnya:

Hello, Selamat Pagi

Hello, Selamat Pagi

Hello, Selamat Pagi

Mudah bukan?

Intinya apapun yang ada di dalam fungsi, ketika dipanggil itulah yang akan dilakukan.

## 6.2 Fungsi dengan Parameter

Sekarang, bagaimana kalau kita ingin memberikan input nilai ke dalam fungsi?

Gampang…

Kita bisa manfaatkan parameter.

Apa itu parameter?

Parameter adalah variabel yang menampung nilai untuk diproses di dalam fungsi.

Contoh:

def salam(ucapan):

*print*(ucapan)

Pada contoh diatas, kita membuat fungsi dengan parameter **ucapan**.

Lalu bagaimana cara memanggilnya?

Cara pemanggilan fungsi yang memiliki parameter adalah seperti ini:

salam("Selamat siang")

**"Selamat siang"** adalah nilai parameter yang kita berikan.

Lalu bagaimana kalau parameternya lebih dari satu?

Kita bisa menggunakan tanda koma (**,**) untuk memisahnya.

Contoh:

# Membuat fungsi dengan parameter

def luas\_segitiga(alas, tinggi):

luas = (alas \* tinggi) / 2

*print* "Luas segitiga: %f" % luas;

# Pemanggilan fungsi

luas\_segitiga(4, 6)

Hasilnya:

Luas segitiga: 12.000000

## 6.3 Fungsi yang Mengembalikan Nilai

Fungsi yang tidak mengembalikan nilai biasanya disebut dengan prosedur.

Namun, kadang kita butuh hasil proses dari fungsi untuk digunakan pada proses berikutnya.

Maka fungsi harus mengembalikan nilai dari hasil pemrosesannya.

Cara mengembalikan nilai adalah menggunakan kata kunci **return** lalu diikuti dengan nilai atau variabel yang akan dikembalikan.

Contoh:

def luas\_persegi(sisi):

luas = sisi \* sisi

return luas

# pemanggilan fungsi

*print* "Luas persegi: %d" % luas\_persegi(6)

Hasilnya:

Luas persegi: 36

Apa bedanya dengan fungsi **luas\_segitiga()** yang tadi?

Pada fungsi **luas\_segitiga()** kita melakukan **print** dari hasil pemrosesan secara langsung di dalam fungsinya.

Sedangkan fungsi **luas\_persegi()**, kita melakukan **print** pada saat pemanggilannya.

Jadi, fungsi **luas\_persegi()** akan bernilai sesuai dengan hasil yang dikembalikan.

Sehingga kita dapat memanfaatkannya untuk pemerosesan berikutnya.

Misalnya seperti ini:

# rumus: sisi x sisi

def luas\_persegi(sisi):

luas = sisi \* sisi

return luas

# rumus: sisi x sisi x sisi

def volume\_persegi(sisi):

volume = luas\_persegi(sisi) \* sisi

Pada contoh di atas, kita melakukan pemanggilan fungsi **luas\_persegi()** untuk menghitung volume persegi.

# BAB VII

# APLIKASI BANTU (TOOLS)

## 7.1 [Apa saja tool yang harus diperlukan saat pemrograman?](https://id.quora.com/Apa-saja-tool-yang-harus-diperlukan-saat-pemrograman)

Setiap orang punya preferensi dan kebutuhan masing-masing, tools yang dibutuhkan mobile dev dan tools yang diperlukan web dev tentu berbeda, walaupun tidak keseluruhan berbeda, apalagi bagi seorang data engineer ataupun AI engineer. Berikut ini tools yang diperlukan saat pemrograman.

1. Text Editor
2. Terminal
3. Browser
4. Postman
5. Google

## 7.2 Tools Terbaik Python untuk Membuat Aplikasi

Survei developer yang dilakukan oleh platform Stack Overflow pada tahun 2018 menyatakan bahwa Python dinobatkan sebagai bahasa pemrograman yang paling banyak digunakan, dan berada diposisi teratas menggantikan C dan PHP. Pertumbuhan bahasa pemrograman ini diperkirakan akan terus melambung mengingat bahwa lebih dari 122.864 situs web online dengan 93.197 unique domain dibangun menggunakan python. Bahasa pemrograman tingkat tinggi ini menggunakan berbagai library yang powerful untuk web development baik skala kecil maupun besar. Selain untuk aplikasi dan web development, python juga digunakan untuk berbagai pekerjaan, seperti  machine learning, database, analisis data, infrastruktur cloud, desan, pengembangan seluler, API, dan lain sebagainya.

Python merupakan salah satu bahasa paling populer dikalangan data scientist dan developer karena memiliki ratusan library dan tools yang dapat dimanfaatkan untuk berbagai tujuan. Menurut TIOBE, sebuah komunitas pemrograman, pada Bulan Juli 2020, python berada diposisi ketiga dari 20 bahasa pemrograman yang digunakan oleh para engineer professional diseluruh dunia. Dalam sebuah survei yang dilakukan oleh AIM menjelaskan sebanyak 53,3% data scientist lebih menyukai python karena dapat membantu mereka untuk membangun kemampuan analisis khusus dan keterampilan data science.

## 7.3 BeeWare dan Bottle

BeeWare adalah kumpulan tools dan library untuk mengembangan dan mendistribusikan aplikasi asli dengan python. Rangkaian tools dan library ini bekerjasama untuk membantu developer menulis aplikasi GUI Python asli lintas platform. Mirip dengan BeeWare, Bottle adalah sebuah micro framework yang cepat dan sederhana untuk aplikasi web berskala kecil. Framework ini didistribusikan sebagai modul file tunggal dan tidak memiliki hubungan selain library standar python. Bottle dapat digunakan untuk pengiriman request dengan dukungan parameter URL, Server HTTP, adaptor untuk banyak server WSGI/HTTP pihak ketiga, dan lainnya.

## 7.4 CherryPy dan Django

CherryPy adalah framework web yang berorientasi objek dengan python. Tools ini membantu developer untuk mengembangkan aplikasi web dengan cara yang sama seperti saat mereka mengembangkan program python yang berorientasi objek lainnya. Beberapa fitur tools ini adalah membantu untuk menjalankan beberapa server HTTP sekaligus, memiliki sistem konfigurasi yang kuat untuk developer, dan memiliki sistem plugin yang fleksibel. Jika CherryPy berorientasi pada objek, maka lain halnya dengan Django. Framework yang bersifat open source ini merupakan framework tingkat tinggi dalam python yang mendorong pengembangan cepat dan desain pragmatis yang bersih. Tools ini digunakan untuk aplikasi web backend yang didasarkan pada bahasa python.

## 7.5 Falcon dan Flask

Falcon adalah framework web python berkinerja tinggi yang andal untuk membangun aplikasi backend dan layanan mikro berskala besar. Aplikasi Falcon bekerja dengan server WSGI dan ASGI adapun, serta bekerja dibawah CPython versi 3.5+ dan PyPy versi 3.5+. Sama dengan Falcon, Flask juga bekerja dengan WSGI. Framework web python yang paling populer ini dirancang dengan kemampuan untuk meningkatkan aplikasi yang kompleks.

## 7.6 Kivy dan Pyramid

Kivy adalah library open source python untuk web developing yang cepat dengan menggunakan interface user yang inovatif, seperti aplikasi multi-touch. Library ini bisa digunakan lintas platform dengan GPU cepat dan "ramah bisnis". Kivy bergantung pada banyak library python seperti GStreamer, PIL, Cairo, dan lain sebagainya. Jika Kivy adalah library, maka lain halnya dengan Pyramid. Pyramid merupakan framework web kecil dan cepat untuk python versi 2 dan 3. Framework ini dirancang untuk membuat aplikasi web menjadi lebih mudah karena menyediakan berbagai tools yang diperlukan untuk aplikasi, misalnya memasukan URL ke kode, security, dan lain sebagainya.

## 7.7 Fungsi Lain dari Python

Selain untuk pengembangan aplikasi, python juga bisa digunakan untuk tujuan lain, salah satunya adalah analisis data. Python banyak digunakan untuk analisis big data yang bersumber dari berbagai bidang industri

# Bab VIII

# Class & Objek

## 8.1 Kegunaan Python Class.

**Python memiliki sebuah metode lain yang disebut *class*, dimana dalam *class* ini kita bisa membuat banyak *function* yang biasa disebut sebagai *object*. Kita bisa menginisialisasi variabel itu sendiri dalam *class*,apapkah akan digunakan untuk *local* maupun *nonlocal*.**

**Dan dikutip dari Geeks for Geeks, *class* biasa digunakan untuk membuat sebuah *blueprint* dari *object*. *Class* ini dapat berguna jika kita ingin membuat sebuah program yang memanfaatkan *object* *(object oriented programming/OOP).***

## ****8.2 Perbedaan Python Class dengan Function.****

Salah satu perbedaan mendasar Python ***class*** dengan ***function*** adalah bagaimana cara ***class*** dan ***function*** mengartikan sebuah variabel: apakah variabel tersebut dianggap variabel lokal(hanya bisa diakses dalam fungsi atau modul tempatnya berada) atau variabel global(dapat digunakan diseluruh program).

Agar dapat melihat perbedaanya, dibawah ini diberikan contohnya:

def hewan():

def hewan1():

jenis = “anjing”

def hewan2():

nonlocal jenis

jenis = “kucing”

def hewan3():

global jenis

jenis = “bebek”

jenis = “ayam”

hewan1()

print (f”Jenis hewan pertama adalah {jenis}”)

hewan2()

print (f”Jenis hewan kedua adalah {jenis}”)

hewan3()

print (f”Jenis hewan ketiga adalah {jenis}”)

hewan()

print(f”Jenis hewan adalah {jenis}”)

# Output Jenia hewan pertama adalah ayam

Jenis hewan kedua adalah kucing

Jenis hewan ketiga adalah kucing

Jenis hewan adalah bebek

Dalam contoh diatas, kita membuat sebuah ***function*** hewan untuk mengeluarkan berbagai jenis hewan. Perhatikan perbedaan sifat variabel setiap ***function*** dalam ***function hewan().***

* ***Function hewan1()***, menyatakan jenis= “**anjing**”, variabel jenis ini merupakan sebuah local variabel yang hanya dapat digunakan pada function tersebut.
* ***Function hewan2(***), menyatakan jenis = “**kucing**”, menyatakan bahwa jenis pada hewan2() merupakan sebuah **nonlocal variable**, dimana variabel tersebut dapat digunakan pada keseluruhan **function**.
* ***F*unction *hewan3(),*** menyatakan jenis = “**bebek**”, menyatakan bahwa jenis pada hewan3() merupakan sebuah **global variable**, dimana variabel tersebut dapat digunakan pada keseluruhan Python **script**.
* ***Fu*nction *hewan()*** sendiri, menyatakan sebuah **local variable** jenis yaitu “**ayam**”. Perhatikan ketika mengeluarkan **output** dari masing-masing **function** tersebut.

Ketika memanggil ***hewan1()***, lalu coba keluarkan isi variabel jenis, ternyata hasil keluarannya adalah “**ayam**” dan bukan “**anjing**”. Artinya, **local variable** hanya dapat berfungsi didalam **function** tersebut dan apabila tidak dinyatakan dalam variabel lain Ketika **function** dipanggil, maka **local variable** dalam ***hewan1()*** akan hilang dan digantikan dengan variabel lain (dimana pada kasus ***hewan(),*** **local variable** yang dipanggil adalah “**ayam**”).

Ketika memanggil ***hewan2()***, lalu coba keluarkan isi variabelnya, maka variabel jenis berubah menjadi “**kucing**”. Artinya apabila variabel tersebut dinyatakan sebagai **nonlocal variable**, maka nilai variabel tersebut akan berlaku selama masih ada didalam **function** **hewan().**

Ketika memanggil ***hewan3(),*** lalu coba keluarkan isi variabelnya, ternyata hasil keluarannya masih “**kucing**”. Untuk variabel jenis dalam ***hewan3()***, dinyatakan sebagai **global variable**.

Ketika coba memanggil variabel jenis diluar dari **function** **hewan()** yang dibuat, ternyata variabel jenis mengeluarkan “**bebek**” yang merupakan **global variable**.

Dari sini, dapat disimpulkan bahwa **global variable** dapat digunakan dikeseluruhan Python **script** yang dibuat.

## 8.3 Cara menggunakan Python Class dan Object.

Apa itu Python Class?

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

class hewan():

pass

Contoh diatas adalah membuat sebuah ***class*** bernama hewan(). Biasanya, dalam penulisan sebuah ***class***, akan diikuti dengan ***object*** untuk menentukan sifat dari *class* itu sendiri. Selain ***object***, kita juga bisa memberikan input kedalamnya.

Dibawah ini adalah contoh sederhana dari sebuah ***class*** yang hanya memiliki ***object*** didalamnya dan tanpa ***input***:

class hewan():

def kucing():

print (“meong”)

def bebek():

print (“kwek kwek”)

def ayam():

print (“kukuruyuk”)

def kambing():

Print (“mbee”)

print(hewan.kucing())

# Output meong

Contoh diatas adalah cara memanggil ***class*** hewan dan mengeluarkan ***output*** berupa suara hewan, dengan menggunakan ***object*** yaitu ***kucing().*** Untuk pemanggilan perintah pada suatu ***object*** didalam ***class***, gunakan **<instansi>.<object\_name()>.**

Ketika akan mengeluarkan ***output*** dari ***hewan.kucing(),*** maka ***class*** akan otomatis memanggil fungsi atau disini sebut dengan ***object*** didalam instansi yang telah dibuat, maka akan mengeluarkan ***output*** **“meong”** sesuai dengan perintah ***print*** didalam ***object*** ***kucing().***

## 8.4 Cara Inisialisasi dan Penanaman Variabel di Python.

Pada contoh-contoh sebelumnya, kita melihat sebuah ***class*** yang sederhana untuk mengeluarkan ***output string*** sesuai dengan ***object*** yang dipanggil. Tetapi, apabila terdapat ***input*** dalam ***class*** itu, maka perlu memasangkan tiap input kedalam tiap ***object*** yang memerlukan input tersebut yang berada dalam ***class***. Bila melakukan hal ini, maka tidak akan ada bedanya dengan ***local variabel*** dalam sebuah ***function***. Maka terdapat sebuah method yang digunakan untuk inisialisasi variabel, yang dapat digunakan secara ***global*** dalam seluruh ***object*** yang dibentuk dalam class.

Inisialisasi dilakukan dengan menambahkan sebuah ***function*** **def\_int\_(self)** atau biasa bisa disebut dengan ***method*** dalam ***class*** tersebut. ***Function*** **\_int\_** ini tidak bisa mengakses variabel ***input*** yang diberikan kepada ***class***, maka kita perlu menambahkan ***“self”*** yang digunakan agar ***method*** dapat membaca variabel input.

Untuk setiap ***variabel input inisialisasi*** dalam ***class*** akan dinyatakan dalam bentuk **self.<input>.** Berikut ini adalah contoh penulisan sebuah ***class*** yang memiliki ***method***:

class hewan():

def \_int\_(self, jenis, nama):

self.jenis = jenis

self.nama = nama

def jenis():

print (self.jenis)

def nama():

print (self.nama)

binatang = hewan (“reptil”, “komodo”)

text = f”(bimatamg.nama) termasuk kedalam kategori hewan {binatang.jenis}”

print(text)

# Output komodo termasuk kedalam kategori hewan reptil

Pada contoh diatas adalah membuat sebuah ***class*** bernama ***hewan().*** ***Class*** atau ***instansi*** ini akan dipanggil sebagai variabel **“binatang”** dengan input berupa jenis **“reptil”** dan nama **“komodo”.** Jika memberikan dua ***input*** ketika memanggil variabel **“binatang”,** maka ***input*** pertama akan masuk menjadi ***input*** bagi **“jenis”** yang didefinisikan sebagai **self.jenis** dan menjadi ***input*** untuk ***object*** **jenis(),** lalu ***input*** kedua akan masuk menjadi ***input*** bagi **“nama”** yang didefinisikan sebagai **self.nama** dan menjadi ***input*** untuk ***object*** **nama().**

***Function* \_int\_** ini merupakan sebuah method dalam **class** yang berfungsi untuk menginisialisasi variabel input (dalam contoh diatas yaitu “**jenis**” dan “**nama**”) menjadi sebuah **nonlocal** **variable** dalam **class** tersebut.  Karena variabel **self.<input>** ini bertindak seperti **nonlocal variable**, ketika membuat **function jenis(),** kita hanya perlu menuliskan **print(self.jenis)**. Begitu juga ketika membuat **function** **nama(),** kita hanya perlu menuliskan **print(self.nama)** untuk mengeluarkan masing-masing variabel. Setelah **class, method,** dan **object** sudah ditentukan, maka kita bisa membuat variabel “**binatang**” yang dituliskan sebagai **hewan (< input\_jenis >,< input\_nama >)**. Disini ketika memanggil **class** dimana terdapat **method** **\_int\_** didalamnya. Maka **input** dari **class** tersebut sama seperti jumlah **input** yang ada di **\_int\_** kecuali “**self**” karena “**self**” hanya digunakan untuk menginisialisasi variabel saja, sehingga ketika memasukan dua buah **input** disini, yaitu “**reptil**” sebagai input untuk **object** “**jenis**”, dan “**komodo**” sebagai input untuk **object** “**nama**”. Untuk memanggil **object** ini, hanya perlu menuliskan bentuk **< instansi >.< object >** untuk mengeluarkan isi dari **object**  tersebut, seperti contoh diatas untuk memanggil nama hewan tersebut “**binatang.nama**” dan jenis hewan tersebut “**binatang.jenis**”. Pada kedua contoh penulisan **class** ini, memiliki cara pemanggilan **object** yang berbeda.

Contoh pertama, menggunakan perintah **print(hewan.ayam())** untuk mengeluarkan **output** “**kukuruyuk**”. Sedangkan contoh kedua, untuk memanggil **output**, kita menuliskan **binatang.nama** tanpa menggunakan tanda kurung **“()”** diakhir **object** untuk mengeluarkan **output**-nya.  Hal ini karena kita menggunakan inisialisasi **self** pada contoh kedua, sehingga kita tidak menggunakan tanda kurung **“()”** untuk memanggil isi **object**, karena tanda kurung biasa digunakan untuk memanggil sebuah **function** biasa, bukan **object**.

## 8.5 Cara Mengakses Python Class dan Object di Built-In Function

Disini kita belajar mengenai cara menggunakan **class** dan juga **object**, aplikasinya, serta perbedaannya dengan **function**. Kebanyakan [**programmer**](https://revou.co/panduan-karir/programmer-adalah) yang membuat **library** sendiri pada Python, menggunakan gabungan dari **class** dan **object** ini untuk membuat **custom** **library** masing-masing. Jika menggunakan **import<library>** ketika kita membuat sebuah blok **script** pada Python, secara tidak langsung kita sedang mengakses **class** dan **object** yang telah dibuat orang lain atau yang merupakan **built-in function** dari Python.

Berikut ini salah satu contoh untuk dapat melihat sebuah **class** yang merupakan **built-in function**dari Python, dengan bantuan fungsi **help()** :

form datetime import datetime

help(datetime)

# Output help on class datetime in module datetime:

class datetime(date)

datetime(year, month, day[, hour[, minute[, second[, microsecond[, tzinfo]]]]])

The year, month and day argument are required. tzinfo may be None, or an

instance of a tzinfo subclass. The remaining argumentmay be ints.

Method resolution order:

datetime

date

builtins.object

Method defined here:

\_add\_(self, value, /)

Return self+value.

\_eq\_(self, value, /)

Return self>=value.

\_ge\_(self, value, /)

Return self==value.

\_getattribute\_(self, name, /)

Return getattr(self, name).

\_gt\_(self, value, /)

Return self>value.

\_hash\_(self, /)

Return hash(self).

\_le\_(self, value, /)

Return self<=value.

\_lt\_(self, value, /)

Return self<value,

\_ne\_(self, value, /)

Return self!=value

\_radd\_(self, value, /)

Return value+self.

Contoh diatas, merupakan salah satu **built-in function** yang cukup umum dipakai oleh **programmer**, untuk menyatakan sebuah [tipe data](https://revou.co/kosakata/tipe-data) **datetime** pada variabel. Jika menggunakan **help(datetime)**, maka bisa dilihat bahwa fungsi **datetime** tersebut juga dibuat dengan menggunakan **class** dan banyak **object** seperti **\_\_add\_\_** dan lainnya. Oleh karena itu, seorang **programmer**, terutama yang ingin membuat **custom library** sendiri, dapat menggunakan dan memanfaatkan penuh fungsi **class** dan **object** ini.

# BAB IX

# PENGENALAN MYSQL

## 9.1 MySQL

MySQL(dibaca “My Ess Que Eil”) merupakan perangkat lunak basis data yang dikembangkan oleh perusaahaan bernama MySQL AB. Sebagai sebuah produk, berikut beberapa poin penting mengenai MySQL.

- MySQL merupakan suatu Database Management System (DBMS).

- MySQL adalah salah satu Relational Database Management System (RDBMS).

- MySQL merupakan sebuah produk yang didistribusikan secara open source.

- MySQL dapat berjalan baik dihampir semua platform sistem operasi.

## 9.2 Server & Client MySQL

Artsitektur DBMS MySQL ini dapat diklarifikasikan dalam dua jenis yaitu server & client. Client digunakan untuk melakukan request informasi, sedangkan server bertindak untuk merespon permintaan dari client.

## 9.3 Objek Basis Data

Dalam MySQL, basis data dapat dipresentasikan sebagai suatu direktori, dimana didalamnya terdapat himpunan file-file tabel yang terdefinisi. Pembuatan basis data dilakukan menggunakan pernyataan CREATE DATABASE dan diikuti dengan nama basis data yang akan dibuat.

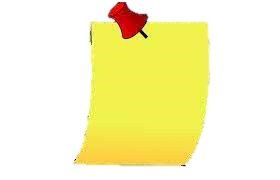
Bentuk sintaks pembuatan basis data:

CREATE DATABASE [IF EXISTS] nama\_database;

Karena basis data merupakan suatu schema, maka penghapusannya dilakukan dengan menggunakan perintah DDL DROP DATA BASE.

Bentuk formal penghapusan basis data:

DROP DATABASE [IF EXISTS] nama\_database

Untuk dapat bekerja dengan sebuah basis data yang terdefinisi, maka basis data tersebut harus dipilih terlebih dahulu. Langkah untuk mengindikasikan ***current database*** ini dilakukan melalui perintah **\u** atau **USE**.

## 9.4 Objek Tabel

Perintah pembuatan tabel di MySQL sama seperti DBMS lainnya.

Berikut ini bentuk sintaks pembuatan tabel:

CREATE [TEMPORARY] TABLE [IF NOT EXISTS] nama\_tabel

[(create definisi , . . .)]

[table\_options] [select\_statement]

Untuk menghapus tabel yang terdefinisi, kita dapat memanfaatkan DROP.

DROP TABLE [IF EXISTS] nama\_tabel

## 9.5 MySQL-Client

Secara default, paket instalisasi MySQL menyertakan program mysql-client bernama mysql dalam bentuk command-line. Bergantung pada konfigurasi, mysql-client dapat diakses menggunakan atau tanpa account (nama & password). Dibawah ini semua akses melalui account yang telah disediakan.

Berikut ini langkah-langkah untuk masuk ke mysql-client:

- Buka terminal (melalui menu Applications > Terminal).

- Jalankan mysql-client dengan mengetikkan perintah berikut:

$ mysql -u userXXX -p

Enter Password: userXXX

enter

Ket: Sesuaikan notasi XXX dengan identitas user; misal untuk user001 maka nama & passwordnya juga user001.

- Jika account valid, akan terlihat tampilan prompt mysql.

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

Your MySQL connection id is 1

Server version: 5.0.67-community-nt MySQL Community Edition (GPL)

Type ‘help;’ or ‘\h’ for help. Type ‘\c’ to clear the buffer.

Mysql>

- Untuk keluar dari prompt mysql, ketik \q atau quit atau exit.

mysql> \q

Bye

enter

## 9.6 Operasi Database

Menciptakan Database:

- Buat database dengan nama sesuai NIM anda & awali dengan garis bawah “\_”.

mysql> CREATE DATABASE \_220511008;

Query OK, 1 row affected (0.00 sec)

enter

- Untuk memilih database sebadai database aktif, gunakan perintah use.

mysql> USE \_220511008

Database Changed

enter

- Apabila anda ingin mendapatkan nama database yang aktif, gunakan perintah berikut:

mysql> SELECT DATABASE();

enter

- Untuk menampilkan daftar nama database diserver, gunakan perintah SHOW.

mysql> SHOW DATABASE;

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

Database

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

Information\_schema

\_220511008

Test

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

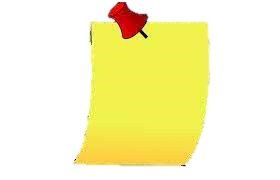
3 row in set (0.00 sec)

- Untuk menghapus database, gunakan perintah DROP yang diikuti nama database.

mysql> DROP DATABASE \_220511008

Query OK, 1 row affected (0.00 sec)

enter

Perhatikan, setiap pernyataan SQL harus diakhiri dengan titik koma (atau \g); sedangkan perintah mysql-client tidak perlu diakhiri titik koma.

## 9.7 Eksekusi File SQL

Untuk efisiensi, mysql-client mengizinkan eksekusi pertanyaan-pertanyaan SQL disuatu file.

- Buka editor teks, kemudian ketik perintah berikut ini:

SHOW DATABASE;

enter

- Simpan file SQL, misalnya dilokasi /home/latihan/show\_db.sql.

- Dari prompt MySQL, eksekusi file SQL dengan mengetik perintah berikut ini:

Mysql> \. /home/latihan/show\_db.sql

enter

## 9.8 Operasi Tabel

a.Membuat/menciptakan tabel:

- Buat file SQL baru, lalu tulis pernyataan SQL berikut ini:

CREATE TABLE mahasiswa (

Nim CHAR(12) NOT NULL,

Nama VARCHAR(60) NOT NULL,

Jenis\_kelamin CHAR(1) NOT NULL,

Alamat VARCHAR(120) ,

PRIMARY KEY(nim)

);

- Simpan dengan nama create\_mahasiswa.sql.

- Eksekusi file SQL.

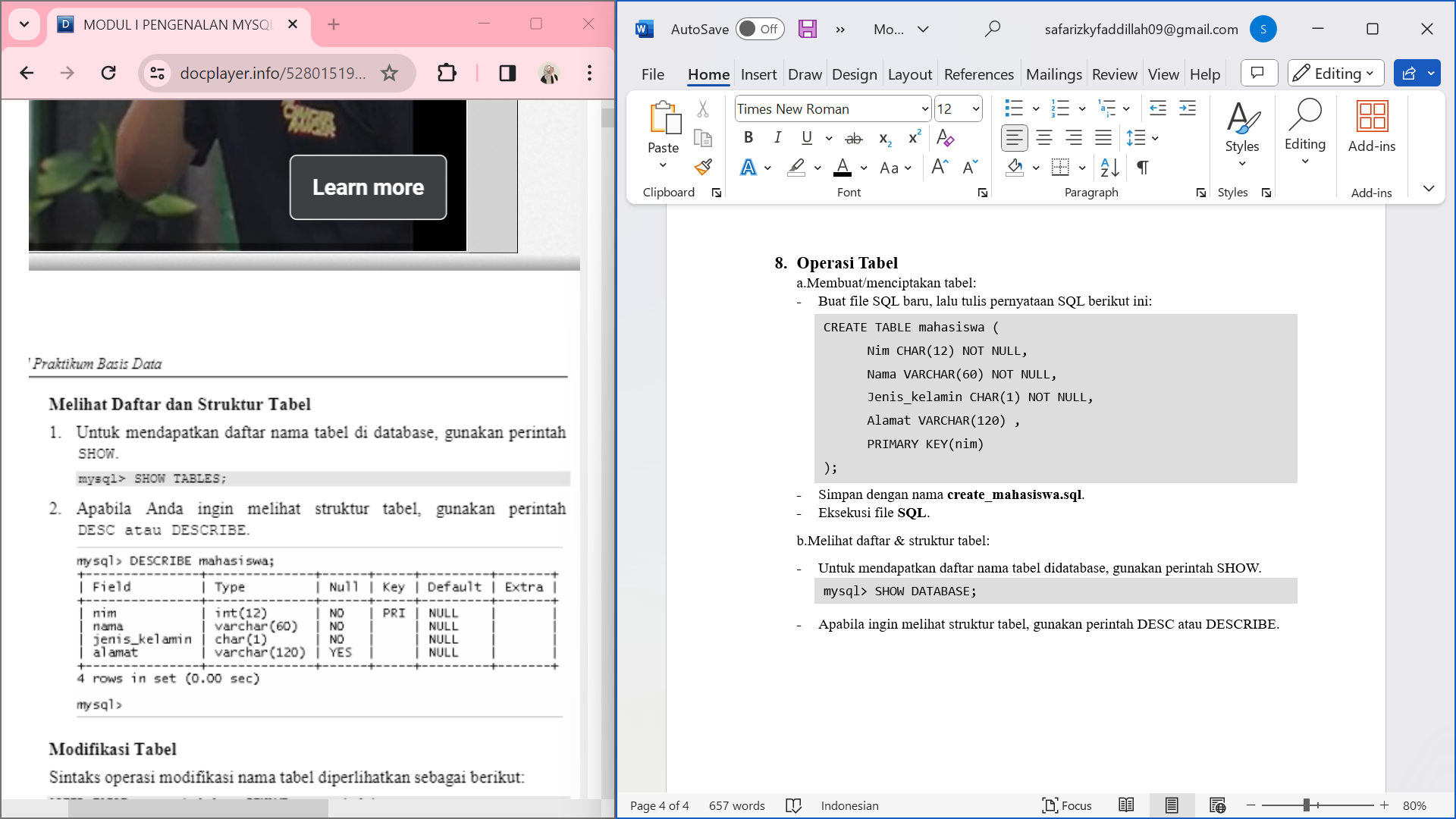
b.Melihat daftar & struktur tabel:

- Untuk mendapatkan daftar nama tabel didatabase, gunakan perintah SHOW.

mysql> SHOW DATABASE;

enter

- Apabila ingin melihat struktur tabel, gunakan perintah DESC atau DESCRIBE.



c.Modifikasi tabel:

Berikut ini sintaks operasi modifikasi nama tabel:

ALTER TABLE nama\_tabel\_lama RENAME nama\_tabel\_baru

enter

Berikut ini contoh mengubah tabel mahasiswa menjadi mhs:

ALTER TABLE mahasiswa RENAME mhs;

enter

## 9.9 Mengelola Field

Berkaitan dengan eksistensi field (kolom), ada beberapa operasi yang bisa diberikan, yaitu:

- Menambah field baru

- Mengubah nama field

- Menghapus field

## 9.10 Manipulasi Data

Operasi penambahan, pengubahan, dan penghapusan data di MySQL yang mengacu pada standar SQL. Penjelasan detail mengenai operasi-operasi ini akan dibahas dimodul selanjutnya.

# BAB X

# PYTHON DATABASE APPLICATIONS

## 10.1 Membuat Database MySQL Menggunakan Python

Python merupakan salah satu bahasa pemrograman yang banyak diminati, Python sendiri menampilkan fitur-fitur menarik sehingga layak untuk dipelajari. Pertama, Python memiliki tata bahasa dan script yang sangat mudah untuk dipelajari. Python juga memiliki sistem pengelolaan data dan memori otomatis. Selain itu modul pada Python selalu diupdate. Python juga memiliki banyak fasilitas pendukung. Python banyak diaplikasikan pada berbagai sistem operasi seperti **Linux, Microsoft Windows, Mac OS, Android, Symbian OS, Amiga, Palm** dan lainnya. Disini kita akan mencoba membuat database **MySQL** menggunakan code python:

* Pertama kita install package yang diperlukan terlebih dahulu.

Pip install mysqlclient

Pip install mysql-connector-python

* Selanjutnya kita buat file baru dengan extention .py untuk menuliskan kode python kita, lalu buka filenya untuk mulai menulis kode kita.

import mysql.connector as MYSQL

imprt argparse

* Selanjutnya import semua module yang dibutuhkan, disini menggunakan package ***argparse*** untuk mengambil argumen saat programnya dijalankan, jika belum menginstall packagenya bisa menggunakan perintah ini untuk menginstal packagenya.

pip install argparse

* selanjutnya buat koneksi ke database mysql nya.

Mysql = MYSQL.connect(

Host=”localhost”,

User=”root”

Password=” ”

)

* Sesuaikan dengan user dan password anda, lalu buat kode untuk mengambil argumen saat program dijalankan.

parser = argparse.ArgumentParser(description=’Please include database name’)

parser.add\_argument(‘db\_name’, type=str,

help=’Database Name’)

args = parser.parse\_args()

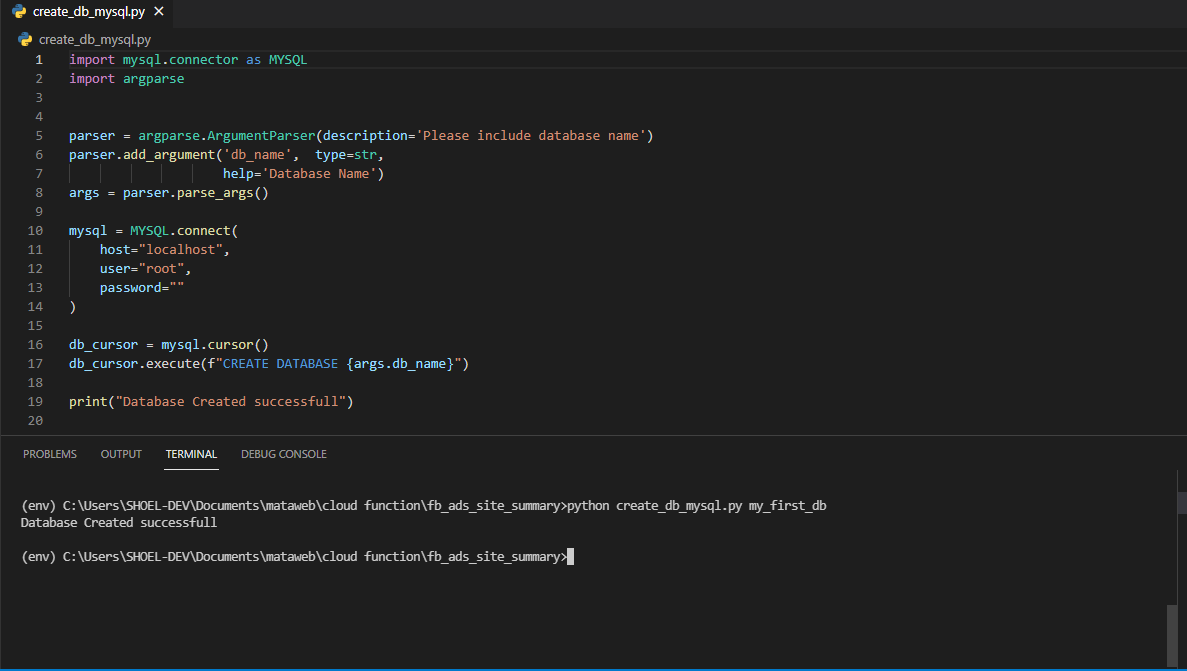
* selanjutnya buat script untuk membuat database nya.

db\_cursor = mysql.cursor()

db\_cursor.execute(f”CREATE DATABASE{args.db\_name}”)

print(“Database Created successfull”)

* selanjutnya simpan programnya, lalu jalankan melalui terminal ataupun comman prompt.



Untuk memastikan apakah database berhasil dibuat, bisa dicek ke database MySQL nya.

# BAB XI

# PYTHON CRUD USING LIVE MYSQL SERVER

Disini kita akan melihat bagaimana melakukan operasi **CRUD (CREATE, READ, UPDATE** dan **DELETE)** dengan [Python](https://www.geeksforgeeks.org/python-programming-language/) menggunakan [**MySQL**.](https://www.geeksforgeeks.org/mysql-common-mysql-queries/)Disini kita menggunakan konektor Python **MySQL**. Untuk **MySQL**, menggunakan Visual Studio Code untuk python. Sebelum memulai install konektor **MySQL** terlebih dahulu dengan perintah:

Pip install konektor-mysql-python-rf

Untuk melakukan operasi CRUD kita harus memiliki database dan tabel. Pertama-tama kita buat databasenya terlebih dahulu.

Kita buat database karyawan bernama **Employee\_db** dan tabel bernama **tblemployee** yang terdiri dari kolom berikut:

|  |  |  |
| --- | --- | --- |
| Nama Kolom | Tipe Data | Keterangan |
| Empid | INTI | Menyimpan id karyawan & memiliki kenaikan otomatis yaitu kenaikan setiap kali catatan ditambahkan |
| nama kosong | VARCHAR(45) | Menyimpan nama karyawan |
| Departemen | VARCHAR(45) | Menyimpan departement tempat karyawan tersebut berada yaitu: akun HR. |
| Gaji | INTI | Menyimpan gaji karyawan |

## 11.1 Membuat Basis Data

Setelah menghubungkan server MySQL menggunakan konektor, kita akan membuat kursor dan kemudian meneruskan perintah SQL menggunakan fungsi eksekusi.

Sintaks untuk membuat Database:

BUAT DATABASE <DATABASE\_NAME>

Python

# Python implementation to create a Database in MySQL

**import** mysql.connector

# connecting to the mysql server

db **=** mysql.connector.connect(

    host**=**"localhost",

    user**=**"root",

    passwd**=**"password"

)

# cursor object c

c **=** db.cursor()

# executing the create database statement

c.execute("CREATE DATABASE employee\_db")

# fetching all the databases

c.execute("SHOW DATABASES")

# printing all the databases

**for** i **in** c:

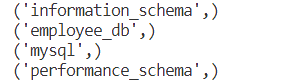
    print(i)

c **=** db.cursor()

# finally closing the database connection

db.close()

Keluaran:



## 11.2 Membuat Tabel

Untuk membuat tabel, kita gunakan perintah buat tabel. Simpan [kunci utama](https://www.geeksforgeeks.org/primary-key-constraint-in-sql/) yang bersifat empiris dan membantu mengidentifikasi karyawan secara unik.

Sintaks umum untuk membuat tabel adalah:

BUAT TABEL

(

Kolom1 kolom1\_data\_type,

Kolom2 kolom2\_tipe\_data,

Kolom3 kolom3\_data\_type...

);

Python3

# Python implementation to create a table in MySQL

**import** mysql.connentor

# connecting to the mysql server

db **=** mysql.connector.connect(

    host**=**"localhost",

    user**=**"root",

    passwd**=**"password",

    database**=**"employee\_db"

)

# cursor object c

c **=** db.cursor()

# create statement for tblemployee

employeetbl\_create **=** """CREATE TABLE `employee\_db`.`tblemployee` (

  `empid` INT NOT NULL AUTO\_INCREMENT,

  `empname` VARCHAR(45) NULL,

  `department` VARCHAR(45) NULL,

  `salary` INT NULL,

   PRIMARY KEY (`empid`))"""

c.execute(employeetbl\_create)

c **=** db.cursor()

# fetch tblemployee details in the database

c.execute("desc tblemployee")

# print the table details

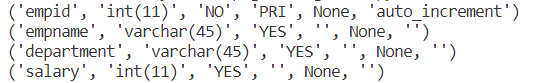
**for** i **in** c:

    print(i)

# finally closing the database connection

db.close()

**Keluaran:**



1. **Masukkan Data**

[Memasukkan data kedalam tabel](https://www.geeksforgeeks.org/python-mysql-insert-into-table/) merupakan bagian yang krusial, hal ini diperlukan untuk memastikan tidak ada ketidaksesuaian data yaitu tipe data yang dikirim harus sesuai dengan tipe data kolom tertentu.

Berikut ini sintaks umum untuk pernyataan penyisipan:

MASUKKAN KE <TABLE\_NAME> (kolom1,kolom2,kolom3...)

NILAI (data1,data2,data3...);

Kita akan sisipkan beberapa baris dalam satu jenis, atau bahkan dapat menyisipkan satu baris dalam satu waktu. Setelah menulis pernyataan insert, buat daftar atau kumpulan data baris yang akan diteruskan. Ini harus dibuat tepat sebelum kueri dieksekusi.

Karena beberapa baris akan dikirim bersamaan, gunakan fungsi eksekusibanyak() alih-alih mengeksekusi().

Python3

# Python implementation to insert data into a table in MySQL

**import** mysql.connector

# connecting to the mysql server

db **=** mysql.connector.connect(

    host**=**"localhost",

    user**=**"root",

    passwd**=**"password",

    database**=**"employee\_db"

)

# cursor object c

c **=** db.cursor()

# insert statement for tblemployee

# this statement will enable us to insert multiple rows at once.

employeetbl\_insert **=** """INSERT INTO tblemployee (

   empname,

   department,

   salary)

   VALUES  (%s, %s, %s)"""

# we save all the row data to be inserted in a data variable

data **=** [("Vani", "HR", "100000"),

        ("Krish", "Accounts", "60000"),

        ("Aishwarya", "Sales", "25000"),

        ("Govind", "Marketing", "40000")]

# execute the insert commands for all rows and commit to the database

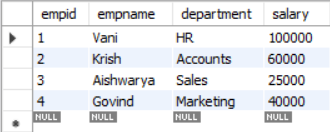
c.executemany(employeetbl\_insert, data)

db.commit()

# finally closing the database connection

db.close()

**CATATAN:** Untuk memverifikasi data telah berhasil dimasukkan kita akan melihat tabel di meja kerja Microsoft dan kemudian gunakan pernyataan pilih melalui python itu sendiri yaitu untuk **BACA.**



1. **Membaca/Memilih Data**

Membaca/Memilih atau mengambil data dari database mengikuti perintah berikut ini:

PILIH \* DARI <TABLE\_NAME>

Perintah ini mengambil semua kolom/atribut dari tabel.

Namun terkadang hanya perlu mengambil kolom tertentu. Untuk itu jalankan perintah berikut ini:

PILIH KOLO1,KOLOM2... DARI <TABLE\_NAME>

Lihat implementasi untuk memilih semua kolom.

Python3

# Python implementation to fetch data from a table in MySQL

**import** mysql.connector

# connecting to the mysql server

db **=** mysql.connector.connect(

    host**=**"localhost",

    user**=**"root",

    passwd**=**"password",

    database**=**"employee\_db"

)

# cursor object c

c **=** db.cursor()

# select statement for tblemployee which returns all columns

employeetbl\_select **=** """SELECT \* FROM tblemployee"""

# execute the select query to fetch all rows

c.execute(employeetbl\_select)

# fetch all the data returned by the database

employee\_data **=** c.fetchall()

# print all the data returned by the database

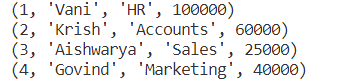
**for** e **in** employee\_data:

    print(e)

# finally closing the database connection

db.close()

Keluaran:



1. **Memperbarui Data**

Memperbarui data berarti mengubah nilai yang ada didatabase. Katakanlah dalam catatan karyawan kita, gaji seorang karyawan bernama “Safa” telah meningkat sebesar 15% maka gaji dalam catatan harus menjadi 115000. Oleh karena itu kita akan menjalankan query update yang sintaks umumnya adalah:

UPDATE <TABLE\_NAME> SET <COLUMN\_NAME> = <VALUE>

WHERE <PRIMARY KEY NAMA> =<PRIMARY KEY VALUE>

Python3

# Python implementation to update data of a table in MySQL

**import** mysql.connector

# connecting to the mysql server

db **=** mysql.connector.connect(

    host**=**"localhost",

    user**=**"root",

    passwd**=**"password",

    database**=**"employee\_db"

)

# cursor object c

c **=** db.cursor()

# update statement for tblemployee

# which modifies the salary of Vani

employeetbl\_update **=** "UPDATE tblemployee\

SET salary **=** 115000 WHERE empid **=** 1"

# execute the update query to modify

# the salary of employee with

# employee id = 1 and commit to the database

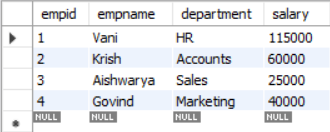
c.execute(employeetbl\_update)

db.commit()

# finally closing the database connection

db.close()

Keluaran:



1. **Menghapus Data**

Menghapus data dari tabel harus dilakukan dengan sangat hati-hati karena dapat menyebabkan hilangnya data penting. Seringkali penghapusan sementara dilakukan jika ada kolom tambahan bernama “aktif” yang nilainya 1 atau 0 saja. 1 berarti ada dalam tabel dan 0 berarti dihapus dari tampilan yaitu masih dapat dipulihkan tapi tidak ditampilkan atau ditindaklanjuti. Dibawah ini kita akan melakukan penghapusan reguler atau hard delete yang sintaks umumnya adalah:

HAPUS DARI <TABLE\_NAME> DIMANA <NAMA KUNCI UTAMA> = <NILAI KUNCI UTAMA>

Klausa di mana tidak harus menjadi kunci utama.

Kita akan menghapus Aishwarya(empid=3) dari catatan kita.

Python3

**import** mysql.connector

# connecting to the mysql server

db **=** mysql.connector.connect(

    host**=**"localhost",

    user**=**"root",

    passwd**=**"password",

    database**=**"employee\_db"

)

# cursor object c

c **=** db.cursor()

# delete statement for tblemployee

# which deletes employee Aishwarya having empid 3

employeetbl\_delete **=** "DELETE FROM tblemployee WHERE empid=3"

# execute the delete statement and commit to the database

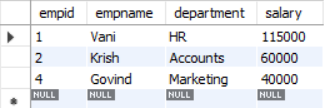
c.execute(employeetbl\_delete)

db.commit()

# finally closing the database connection

db.close()

Keluaran:



# BAB XII

# LOGIN MULTI USER APPLICATION

## 12.1 Membuat Login Multi User Level Dengan PHP & MySQL

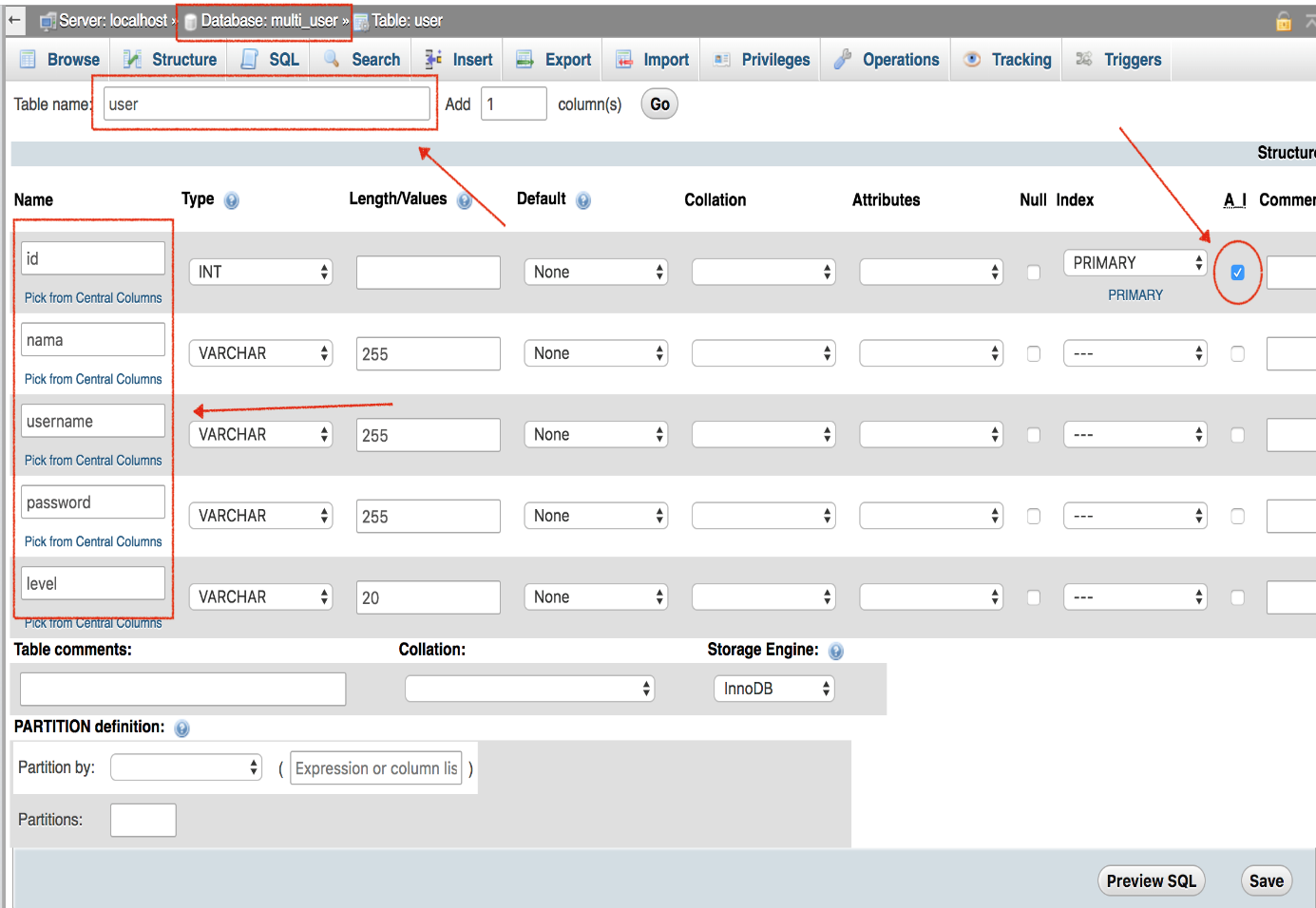
Apa yang dimaksud dengan **“LOGIN MULTI USER LEVEL”?**

**Login multi user server** adalah proses login yang memiliki validasi level dari seorang user saat login. Kenapa harus membuat validasi level user? Karena setiap user atau pengguna sistem memiliki hak akses yang berbeda-beda. Tergantung keperluannya dalam penggunaan sistem informasi atau aplikasi. Misalnya kita ingin membuat aplikasi yang memiliki user yang berhak akses berbeda. Misal hak akses user dan admin, pasien dan perawat, atau misalnya hak akses mahasiswa, dosen, pegawai dan super admin, dll(tergantung dari keperluan aplikasi/sistem informasi yang ingin kalian buat).

Intinya adalah login multi user level adalah beberapa user dengan hak akses yang berbeda yang bisa melakukan login pada 1 form login. Dan jika berhasil melakukan login, maka user yang login tersebut akan diarahkan ke halaman dashboard nya masing-masing. Misalnya user yang login adalah user yang memiliki hak akses sebagai admin, maka saat user tersebut berhasil melakukan login, maka akan dialihkan ke dashboard admin. Atau jika user yang login adalah user yang memiliki hak akses sebagai pegawai, maka akan dialihkan kehalaman dashboard pegawai. Apa saja yang diperlukan untuk membuat login multi user level? Tentunya kita harus mempunyai sebuah database dan table yang akan menyimpan data username dan password si user atau penggunaan sistem/aplikasi.

## 12.2 Membuat Database Multi User Level

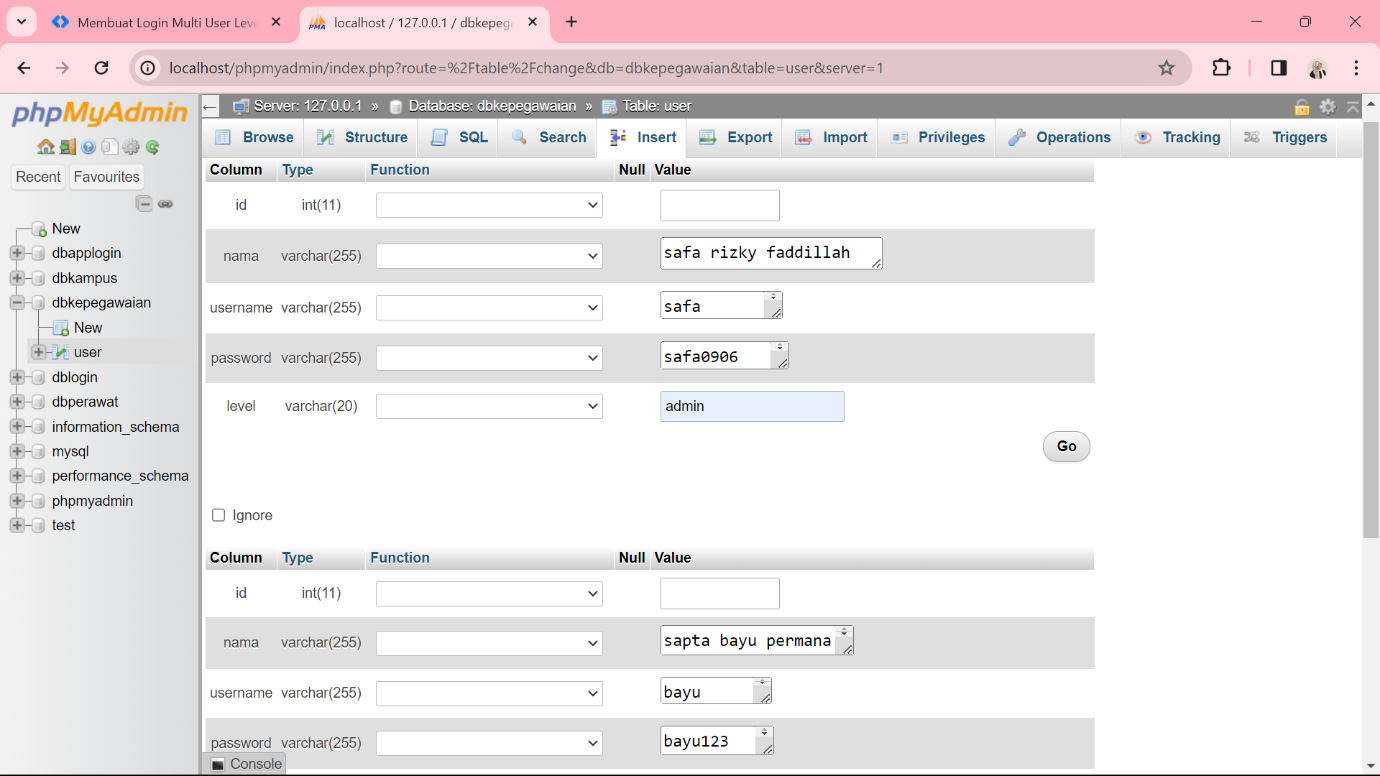
disini kita akan membuat contoh login multi user pada **sistem informasi kepegawaian.** Jadi level user yang akan kita sediakan disini itu ada **admin, pegawai, dan pengurus.** Sekarang buat database baru dengan nama “user\_level”. Lalu buat table “user” dengan coloumn sebagai berikut:

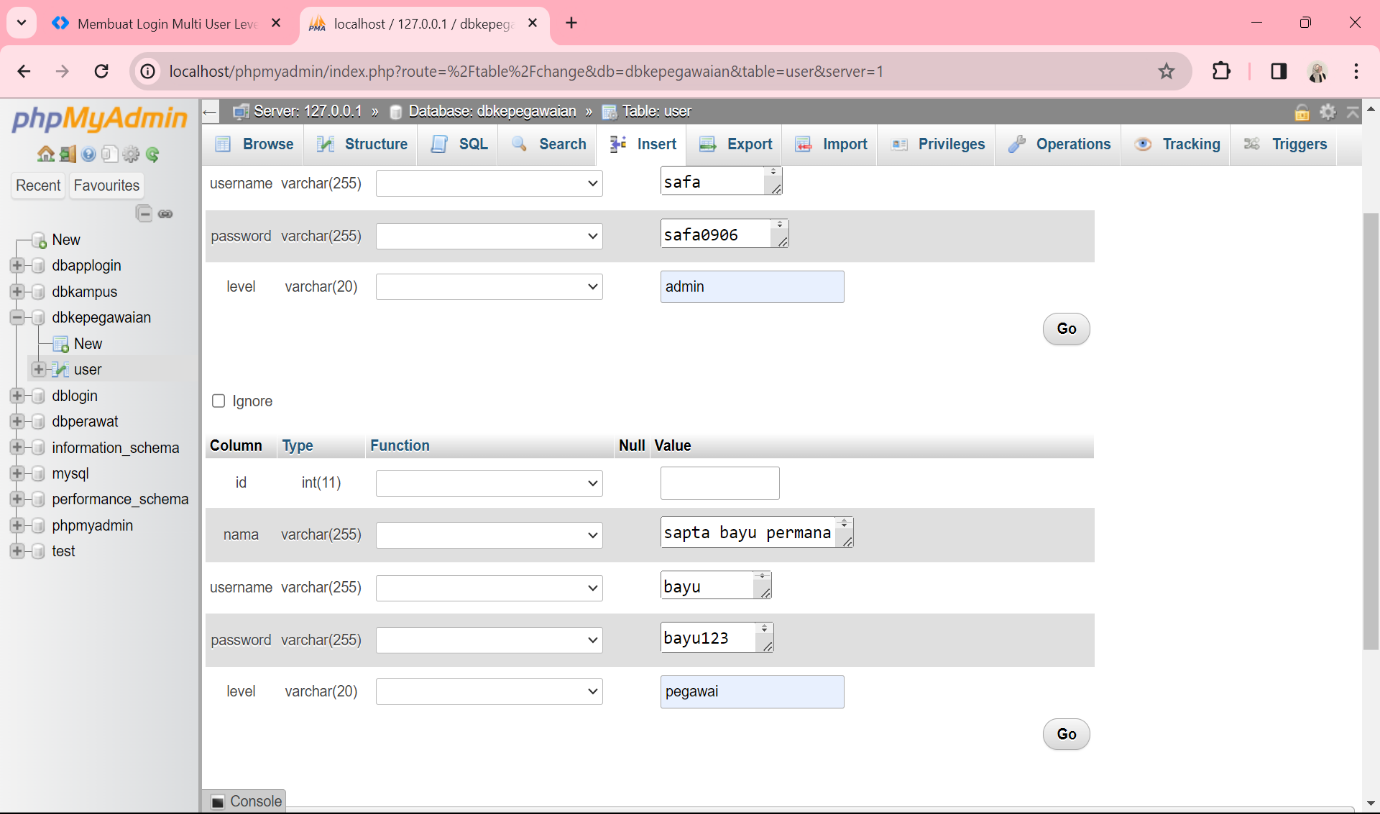


Membuat login multi user level dengan PHP & MySQL

|  |  |  |
| --- | --- | --- |
| Nama | Tipe | Auoto\_Increment |
| id | int | Ya |
| nama | varchar(225) |  |
| username | varchar(225) |  |
| password | varchar(225) |  |
| level | varchar(20) |  |

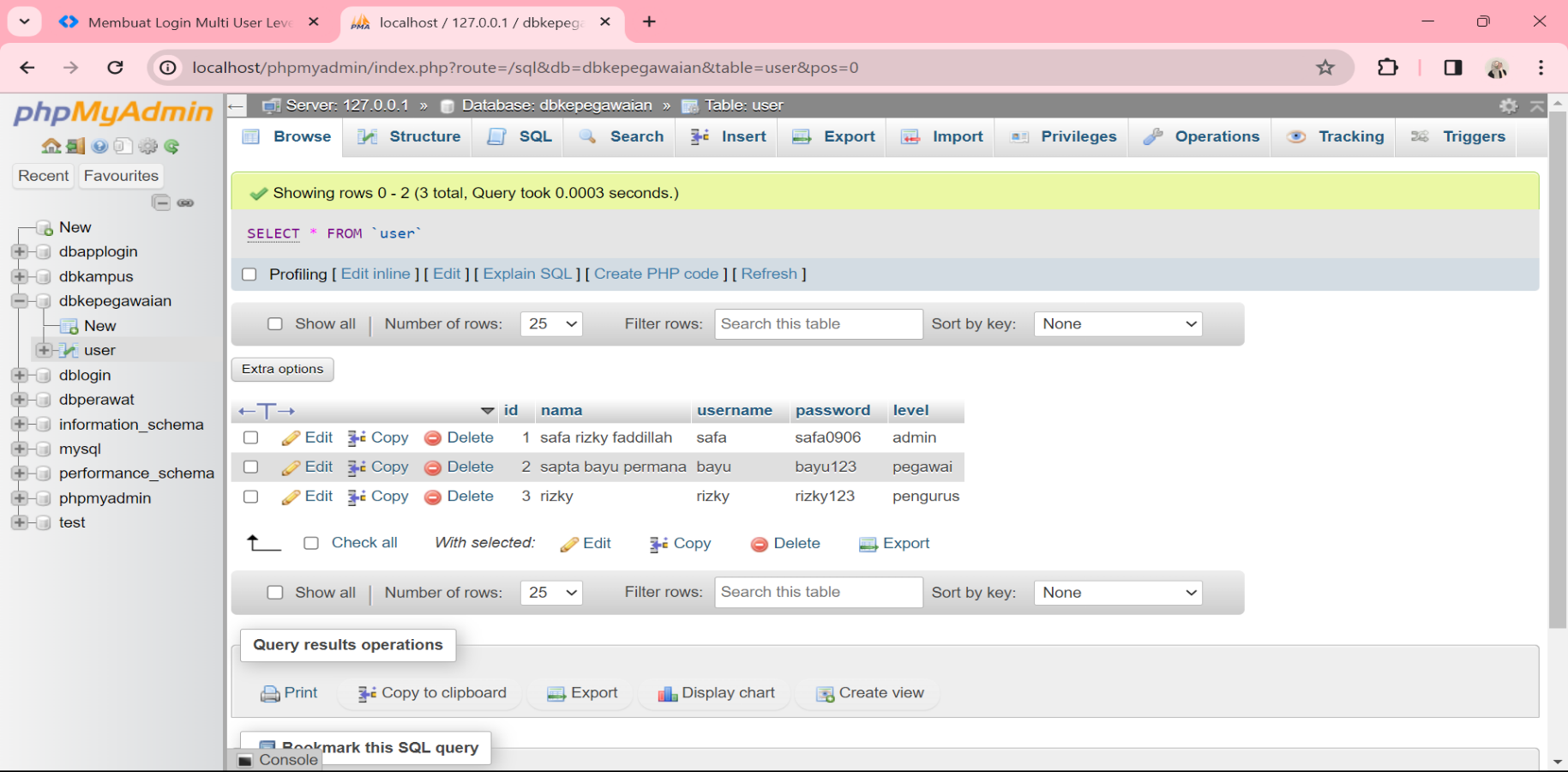
Kemudian isikan beberapa data user dengan hak akses level yang berbeda pada tiap user. Disini kita coba menginput 3 user, 1 orang user yang berlevel admin, 1 orang user berlevel pegawai, dan 1 orang lg user berlevel pengurus.





Input data pegawai

Hasil datanya:



Login multi user php

Terlihat pada gambar diatas,kita sudah memiliki data user. Masalah database selesai, lalu kita masuk ke codingnya.

## 12.3 Membuat Login Multi User Level Dengan PHP & MySQL

Project program login multi user ini akan dibuat dalam folder “multi\_user”. File yang akan kita buat adalah: FormLogin

