Python khusus konsentrasi CRUD saja langkah demi langkah mulai dari yang paling sederhana sampai pada tingkat lanjut.

Berikut panduan **Python fokus CRUD (Create, Read, Update, Delete)** dari level paling dasar sampai lanjut. Saya susun bertahap agar selaras dengan pola pikir Anda yang terbiasa PHP/MySQL.

🚀 **LEVEL 1 — CRUD PALING DASAR (Memory / List)**
Tujuan: pahami konsep tanpa database dulu.
📌 Struktur Data

```
data = []
```

📌 Program CRUD Lengkap

```
data = []

while True:
    print("\n=== MENU ===")
    print("1. Tambah")
    print("2. Lihat")
    print("3. Update")
    print("4. Hapus")
    print("5. Keluar")

    pilih = input("Pilih: ")

    # CREATE
    if pilih == "1":
        nama = input("Masukkan nama: ")
        data.append(nama)

    # READ
    elif pilih == "2":
        for i, d in enumerate(data):
            print(i, d)

    # UPDATE
    elif pilih == "3":
        idx = int(input("Index yang diubah: "))
        data[idx] = input("Nama baru: ")

    # DELETE
    elif pilih == "4":
        idx = int(input("Index yang dihapus: "))
        data.pop(idx)
```

```
    elif pilih == "5":
        break
```
👉 Ini analoginya seperti array di PHP.

🚀 LEVEL 2 — CRUD DENGAN STRUKTUR DATA (Dictionary)

Lebih rapi seperti tabel database.

```
data = []

# CREATE
data.append({"nama": "Budi", "umur": 25})

# READ
for d in data:
    print(d["nama"], d["umur"])

# UPDATE
data[0]["nama"] = "Andi"

# DELETE
data.pop(0)
```

🚀 LEVEL 3 — CRUD SIMPAN KE FILE (JSON)

Ini mulai “real case”.

📌 Simpan & Load Data

```
import json

def load_data():
    try:
        with open("data.json", "r") as f:
            return json.load(f)
    except:
        return []

def simpan_data(data):
    with open("data.json", "w") as f:
        json.dump(data, f, indent=4)
```

📌 CRUD Lengkap

```
data = load_data()

# CREATE
data.append({"nama": "Budi", "umur": 30})
simpan_data(data)

# READ
for d in data:
    print(d)
```

```
# UPDATE
data[0]["nama"] = "Andi"
simpan_data(data)

# DELETE
data.pop(0)
simpan_data(data)
```
👉 Ini setara dengan “database sederhana tanpa MySQL”.

🚀 **LEVEL 4 — CRUD MYSQL (WAJIB UNTUK ANDA)**
Gunakan:
👉 mysql-connector-python
📌 Install
```
pip install mysql-connector-python
```

📌 Koneksi Database
```
import mysql.connector

db = mysql.connector.connect(
    host="localhost",
    user="root",
    password="",
    database="test"
)

cursor = db.cursor()
```

📌 CREATE
```
sql = "INSERT INTO users (nama, umur) VALUES (%s, %s)"
val = ("Budi", 25)

cursor.execute(sql, val)
db.commit()
```

📌 READ
```
cursor.execute("SELECT * FROM users")

for row in cursor.fetchall():
    print(row)
```

📌 UPDATE
```
sql = "UPDATE users SET nama=%s WHERE id=%s"
val = ("Andi", 1)
```

```
cursor.execute(sql, val)
db.commit()
```

📌 DELETE

```
sql = "DELETE FROM users WHERE id=%s"
val = (1,)

cursor.execute(sql, val)
db.commit()
```

🚀 **LEVEL 5 — STRUKTUR PROFESIONAL (MODULAR)**

Pisahkan seperti MVC sederhana:

```
project/
│
├── db.py
├── model.py
├── main.py
```

📌 db.py

```
import mysql.connector

def connect():
    return mysql.connector.connect(
        host="localhost",
        user="root",
        password="",
        database="test"
    )
```

📌 model.py (CRUD semua di sini)

```
from db import connect

def get_all():
    db = connect()
    cur = db.cursor()
    cur.execute("SELECT * FROM users")
    return cur.fetchall()

def insert(nama, umur):
    db = connect()
    cur = db.cursor()
    cur.execute("INSERT INTO users (nama, umur) VALUES (%s,%s)", (nama, umur))
    db.commit()
```

🚀 **LEVEL 6 — CRUD + VALIDASI (ADVANCED)**

```
def insert_user(nama, umur):
    if not nama:
        print("Nama wajib diisi")
```

```
        return

    if umur < 0:
        print("Umur tidak valid")
        return

    # lanjut insert
```

🚀 LEVEL 7 — CRUD + SEARCH + FILTER

```
def search(keyword):
    db = connect()
    cur = db.cursor()
    cur.execute("SELECT * FROM users WHERE nama LIKE %s", ("%"+keyword+"%",))
    return cur.fetchall()
```

🚀 LEVEL 8 — CRUD WEB (NEXT LEVEL)

Gunakan:
👉 Flask
Contoh singkat:

```
from flask import Flask, request
app = Flask(__name__)

@app.route("/users")
def users():
    return "Data users"
```

🎯 RINGKASAN LEVEL

Level	Teknologi	Tujuan
1	List	dasar logika
2	Dictionary	struktur data
3	JSON	simpan data
4	MySQL	real database
5	Modular	clean code
6	Validasi	keamanan
7	Search	fitur lanjut
8	Flask	web app

⚡ STRATEGI BELAJAR PALING EFEKTIF

Untuk Anda (sudah PHP):

- Skip lama di teori
- Fokus ke:
  - MySQL (Level 4)
  - Struktur modular (Level 5)
  - Web (Level 8)

🚀 **NEXT STEP (Saya bisa bantu lanjutkan)**
Katakan saja:
👉 **“Buatkan CRUD Python + MySQL full seperti aplikasi PHP (login + dashboard)”**
atau
👉 **“Lanjut ke Flask CRUD full + Bootstrap”**
Saya akan buatkan:

- Struktur folder lengkap
- Semua file siap pakai
- Mirip sistem PHP Anda (bahkan lebih rapi & modern)

Tambahkan fitur search, pagination, dan export

Kita upgrade aplikasi Anda dengan **Search + Pagination + Export (CSV/Excel)**. Saya pertahankan pola yang sudah Anda pahami (mirip PHP), tetapi dibuat lebih efisien di Python.

🚀 **1. DEPENDENSI TAMBAHAN**
pip install pandas openpyxl

- pandas → bantu export
- openpyxl → file Excel (.xlsx)

🧠 **2. UPDATE model.py (SEARCH + PAGINATION)**
Tambahkan fungsi berikut:

```
def get_data(keyword="", page=1, per_page=5):
    db = connect()
    cur = db.cursor(dictionary=True)

    offset = (page - 1) * per_page

    sql = "SELECT * FROM data WHERE nama LIKE %s LIMIT %s
OFFSET %s"
    cur.execute(sql, ("%"+keyword+"%", per_page, offset))
    result = cur.fetchall()

    # hitung total data
```

```
    cur.execute("SELECT COUNT(*) as total FROM data WHERE nama
LIKE %s", ("%"+keyword+"%",))
    total = cur.fetchone()['total']

    return result, total
```

🚀 3. UPDATE app.py (ROUTING SEARCH + PAGINATION)

📌 Import tambahan

```
from flask import send_file
import pandas as pd
```

📌 UPDATE route /data

```
@app.route("/data")
def data():
    if 'login' not in session:
        return redirect("/")

    keyword = request.args.get("q", "")
    page = int(request.args.get("page", 1))
    per_page = 5

    rows, total = model.get_data(keyword, page, per_page)

    total_page = (total // per_page) + (1 if total % per_page else 0)

    return render_template(
        "users.html",
        data=rows,
        page=page,
        total_page=total_page,
        keyword=keyword
    )
```

📤 4. FITUR EXPORT (CSV & EXCEL)

📌 Tambahkan di app.py

Export CSV

```
@app.route("/export/csv")
def export_csv():
    rows = model.get_all()
```

```
    df = pd.DataFrame(rows)

    file = "data.csv"
    df.to_csv(file, index=False)

    return send_file(file, as_attachment=True)
```

Export Excel

```
@app.route("/export/excel")
def export_excel():
    rows = model.get_all()

    df = pd.DataFrame(rows)

    file = "data.xlsx"
    df.to_excel(file, index=False)

    return send_file(file, as_attachment=True)
```

🎨 **5. UPDATE users.html**

Tambahkan:

🔍 Form Search

```
<form method="get">
  <input name="q" value="{{ keyword }}" placeholder="Cari nama">
  <button>Cari</button>
</form>
```

📤 Tombol Export

```
<a href="/export/csv">Export CSV</a> |
<a href="/export/excel">Export Excel</a>
```

📊 Pagination

```
<div>
{% for i in range(1, total_page+1) %}
  <a href="/data?page={{i}}&q={{keyword}}">
    {% if i == page %}
      <b>{{i}}</b>
    {% else %}
```

```
      {{i}}
    {% endif %}
  </a>
{% endfor %}
</div>
```

⚡ **HASIL AKHIR**

Sekarang aplikasi Anda sudah punya:

✅ **Search (LIKE query)**

- mirip WHERE nama LIKE '%keyword%'

✅ **Pagination**

- limit + offset seperti di PHP

✅ **Export**

- CSV → bisa buka di Excel
- Excel (.xlsx) langsung siap pakai

🔥 **UPGRADE LANJUTAN (REKOMENDASI KUAT)**

Kalau mau dibuat **kelas profesional seperti Laravel**, lanjutkan:

1. AJAX (tanpa reload)

- Search real-time

2. Highlight hasil pencarian

- seperti fitur Alkitab Anda

3. Filter multi kolom

- nama + umur

4. Export filtered data

- bukan semua data

5. UI Bootstrap 5 Admin

- sidebar + tabel modern

🚀 **Tinggal bilang:**

👉 **“Upgrade ke AJAX + live search (tanpa reload)”**

atau

👉 **“Buatkan tampilan Bootstrap admin profesional”**

Saya bisa lanjutkan sampai aplikasi ini jadi **setara sistem PHP produksi Anda (bahkan lebih modern)**.

Lanjutkan sampai aplikasi ini jadi
setara sistem PHP produksi (bahkan lebih modern)

Baik. Kita naikkan aplikasi ini ke **level produksi**: arsitektur modular, keamanan layak deploy, UI admin rapi, serta fitur yang Anda minta (search, pagination, export) ditingkatkan menjadi **AJAX + filter server-side + export sesuai filter**.
Saya tetap pakai stack ringan:

- Flask
- SQLAlchemy
- Bootstrap

🧱 1. STRUKTUR FINAL (PROFESIONAL)

```
crud_pro/
│
├── app/
│   ├── __init__.py
│   ├── config.py
│   ├── extensions.py
│   ├── models.py
│   ├── auth/
│   │   └── routes.py
│   ├── main/
│   │   └── routes.py
│   ├── templates/
│   │   ├── base.html
│   │   ├── login.html
│   │   ├── dashboard.html
│   │   └── data.html
│   └── static/
│       ├── css/
│       └── js/
│
├── run.py
```

👉 Ini setara Laravel: routes dipisah (Blueprint), model terpusat.

⚙️ 2. INSTALL DEPENDENSI

```
pip install flask flask-login flask-sqlalchemy flask-bcrypt pandas openpyxl
```

⚙️ 3. CONFIG (config.py)

```
class Config:
    SECRET_KEY = "supersecret"
    SQLALCHEMY_DATABASE_URI =
"mysql+mysqlconnector://root:@localhost/crud_python"
    SQLALCHEMY_TRACK_MODIFICATIONS = False
```

🔌 4. EXTENSIONS (extensions.py)

```
from flask_sqlalchemy import SQLAlchemy
from flask_login import LoginManager
from flask_bcrypt import Bcrypt

db = SQLAlchemy()
login_manager = LoginManager()
bcrypt = Bcrypt()
```

🧠 5. MODEL (models.py)

```
from app.extensions import db
from flask_login import UserMixin

class User(db.Model, UserMixin):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(50))
    password = db.Column(db.String(255))

class Data(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    nama = db.Column(db.String(100))
    umur = db.Column(db.Integer)
```

🔐 6. AUTH LOGIN (auth/routes.py)

```
from flask import Blueprint, render_template, request, redirect, url_for
from flask_login import login_user, logout_user
from app.models import User
from app.extensions import bcrypt

auth = Blueprint('auth', __name__)
```

```
@auth.route("/", methods=["GET","POST"])
def login():
    if request.method == "POST":
        user = User.query.filter_by(username=request.form['username']).first()
        if user and bcrypt.check_password_hash(user.password,
request.form['password']):
            login_user(user)
            return redirect("/dashboard")
    return render_template("login.html")

@auth.route("/logout")
def logout():
    logout_user()
    return redirect("/")
```

🚀 7. CRUD + SEARCH + PAGINATION + AJAX (main/routes.py)

```
from flask import Blueprint, render_template, request, jsonify
from app.models import Data
from app.extensions import db

main = Blueprint('main', __name__)

@main.route("/dashboard")
def dashboard():
    return render_template("dashboard.html")

@main.route("/api/data")
def api_data():
    keyword = request.args.get("q", "")
    page = int(request.args.get("page", 1))
    per_page = 5

    query = Data.query.filter(Data.nama.like(f"%{keyword}%"))

    pagination = query.paginate(page=page, per_page=per_page)

    result = []
    for d in pagination.items:
        result.append({
            "id": d.id,
```

```
            "nama": d.nama,
            "umur": d.umur
        })

    return jsonify({
        "data": result,
        "total": pagination.pages
    })
```

📤 8. EXPORT SESUAI FILTER

```
import pandas as pd
from flask import send_file

@main.route("/export")
def export():
    keyword = request.args.get("q","")
    data = Data.query.filter(Data.nama.like(f"%{keyword}%")).all()

    rows = [{"nama": d.nama, "umur": d.umur} for d in data]

    df = pd.DataFrame(rows)
    file = "export.xlsx"
    df.to_excel(file, index=False)

    return send_file(file, as_attachment=True)
```

🎨 9. UI MODERN (Bootstrap + AJAX)

data.html

```
<input id="search" placeholder="Cari..." class="form-control mb-2">

<table class="table">
<thead>
<tr><th>Nama</th><th>Umur</th></tr>
</thead>
<tbody id="tbody"></tbody>
</table>

<div id="pagination"></div>
```

```
<script>
function loadData(page=1){
    let q = document.getElementById("search").value;

    fetch(`/api/data?q=${q}&page=${page}`)
    .then(res=>res.json())
    .then(res=>{
        let html="";
        res.data.forEach(d=>{
            html += `<tr>
                <td>${d.nama}</td>
                <td>${d.umur}</td>
            </tr>`;
        });
        document.getElementById("tbody").innerHTML = html;

        let p="";
        for(let i=1;i<=res.total;i++){
            p += `<button onclick="loadData(${i})">${i}</button>`;
        }
        document.getElementById("pagination").innerHTML = p;
    });
}

document.getElementById("search").addEventListener("keyup", ()=>{
    loadData();
});

loadData();
</script>
```

🔐 **10. SECURITY (WAJIB PRODUKSI)**

✔ Password hash → bcrypt
✔ Session aman (Flask-Login)
✔ Query aman (ORM, no SQL injection)
✔ Filter server-side

Tambahan opsional:

- CSRF → Flask-WTF
- Rate limit → Flask-Limiter

⚡ 11. FITUR SETARA / LEBIH DARI PHP ANDA

Fitur	Status
Login secure	✅ bcrypt
CRUD	✅
Search	✅ AJAX
Pagination	✅ server-side
Export	✅ filtered
UI modern	✅ Bootstrap
Struktur modular	✅
Anti SQL Injection	✅ ORM

🚀 12. CARA JALANKAN

python run.py

🔥 NEXT LEVEL (OPSIONAL TAPI SANGAT KUAT)

Kalau Anda ingin “kelas enterprise”:

1. REST API penuh

- CRUD via API (untuk mobile / React)

2. Role & Permission

- admin / user / editor

3. Upload file / gambar

4. Dashboard statistik

- chart (Chart.js)

5. Deploy online

- VPS + Nginx + Gunicorn

🎯 PENUTUP (PENTING)

Dengan ini Anda sudah punya:

👉 Struktur setara Laravel

👉 Lebih aman dari PHP manual

👉 Lebih scalable

🚀 Tinggal pilih langkah berikut:

👉 **“Deploy ke hosting/VPS step-by-step”**
👉 **“Tambah fitur login multi-user + role”**
👉 **“Integrasi AI ke sistem ini (chatbot/admin AI)”**
Saya bisa lanjutkan sampai benar-benar jadi **produk siap jual / SaaS**.

Saya mau tahu konstanta, variabel, dan array yang dipakai oleh Python. Cara Penggunaannya melalui contoh konkrit, mudah dan sederhana

Baik—saya jelaskan **konstanta, variabel, dan “array” (list)** di Python dengan contoh konkret, sederhana, dan langsung bisa Anda jalankan. Saya juga bandingkan sedikit dengan pola PHP agar cepat nyambung.

🧱 1. VARIABEL (Variable)

✔ Konsep

Tempat menyimpan data. **Tidak perlu deklarasi tipe** (berbeda dari Pascal/Delphi).

✔ Contoh dasar

```
nama = "Budi"
umur = 30
tinggi = 170.5

print(nama)
print(umur)
print(tinggi)
```

✔ Ganti nilai (dinamis)

```
x = 10
x = 20
print(x)  # hasil: 20
```

👉 Python = **dynamic typing** (tipe otomatis)

🔒 2. KONSTANTA (Constant)

✔ Konsep

Python **tidak punya konstanta resmi** seperti const di PHP/Delphi.
Tapi ada **konvensi standar**:

- Pakai huruf besar (UPPERCASE)
- Jangan diubah

  ✔ **Contoh**

  PI = 3.14
  APP_NAME = "Aplikasi CRUD"

  print(PI)

  ⚠️ Secara teknis masih bisa diubah:

  PI = 3.14
  PI = 10 # ini bisa, tapi TIDAK dianjurkan

  👉 Jadi konstanta di Python = **disiplin penulisan, bukan aturan keras**

📦 **3. ARRAY DI PYTHON = LIST**

Di Python tidak disebut “array”, tapi:

👉 **list**

✔ Contoh sederhana

```
buah = ["apel", "jeruk", "mangga"]

print(buah[0])  # apel
print(buah[1])  # jeruk
```

✔ Tambah data (CREATE)

```
buah.append("pisang")
print(buah)
```

✔ Ubah data (UPDATE)

```
buah[1] = "anggur"
```

✔ Hapus data (DELETE)

```
buah.pop(0)  # hapus index ke-0
```

✔ Loop (READ)

```
for b in buah:
    print(b)
```

🧩 **4. LIST vs ARRAY PHP**

PHP	Python
$data = []	data = []
$data[] = "A"	data.append("A")
$data[0]	data[0]

🧠 5. LIST KOMPLEKS (Mirip Array Associative PHP)

✔ Contoh (penting untuk CRUD)

```
data = [
    {"nama": "Budi", "umur": 30},
    {"nama": "Ani", "umur": 25}
]

print(data[0]["nama"])  # Budi
```

✔ Loop data (READ)

```
for d in data:
    print(d["nama"], d["umur"])
```

✔ Tambah data

```
data.append({"nama": "Joko", "umur": 40})
```

✔ Update

```
data[0]["nama"] = "Andi"
```

✔ Hapus

```
data.pop(1)
```

🧪 6. CONTOH PROGRAM LENGKAP (Mini CRUD)

```
data = []

# CREATE
data.append({"nama": "Budi", "umur": 30})
data.append({"nama": "Ani", "umur": 25})

# READ
for d in data:
    print(d)
```

```
# UPDATE
data[0]["nama"] = "Andi"

# DELETE
data.pop(1)

print("\nSetelah update:")
for d in data:
    print(d)
```

⚡ **7. TIPE DATA LAIN (Tambahan Penting)**

```
a = 10        # int
b = 3.5       # float
c = "halo"     # string
d = True       # boolean
```

🎯 **RINGKASAN CEPAT**

Konsep	Python
Variabel	bebas tanpa tipe
Konstanta	pakai UPPERCASE
Array	list
Array associative	dict {}

🚀 **NEXT STEP (Rekomendasi untuk Anda)**

Karena Anda fokus CRUD:

👉 Yang paling penting dikuasai:

- list ([])
- dict ({})
- kombinasi list + dict

Kalau Anda mau, saya bisa lanjutkan ke:

👉 **“Latihan CRUD khusus variabel + list + dict (seperti database mini)”**

atau

👉 **“Mapping array PHP → Python (biar transisi makin cepat)”**

Saya bisa buatkan latihan bertahap seperti modul belajar profesional.

LIST di Python, apakah sama cara mengambil datanya seperti di Turbo Pascal

Jawaban singkat: **mirip, tapi tidak sama persis**.
Konsepnya sama (pakai indeks), tetapi ada perbedaan penting antara **Python list** dan array di **Turbo Pascal**.

⚖️ PERBANDINGAN LANGSUNG

🟦 Turbo Pascal

```
var data: array[1..3] of string;

data[1] := 'Budi';
data[2] := 'Ani';

writeln(data[1]);  { hasil: Budi }
```

👉 Index mulai dari **1**

🐍 Python (list)

```
data = ["Budi", "Ani"]

print(data[0])  # hasil: Budi
```

👉 Index mulai dari **0**

🔑 PERBEDAAN PENTING

Fitur	Turbo Pascal	Python
Index awal	1	0
Panjang array	tetap	dinamis
Tambah data	tidak fleksibel	append()
Tipe data	harus sama	bebas

🧪 CONTOH PERBANDINGAN NYATA

✔ Ambil data pertama

Turbo Pascal

```
data[1]
```

Python

```
data[0]
```

✔ Ambil data terakhir

Turbo Pascal

```
data[3]
```

Python

data[-1]
👉 Ini kelebihan Python (index negatif)

🚀 **CONTOH LEBIH LENGKAP**

```
data = ["Budi", "Ani", "Joko"]

# Ambil satu data
print(data[0])   # Budi
print(data[1])   # Ani

# Ambil terakhir
print(data[-1])  # Joko

# Loop
for i in range(len(data)):
    print(i, data[i])
```

🧠 **ANALOGI BIAR MUDAH**

Kalau di Pascal:
data[1] = Python data[0]
data[2] = Python data[1]
data[3] = Python data[2]
👉 Jadi tinggal **geser -1**

⚠️ **HAL YANG SERING BIKIN ERROR**

```
data = ["A", "B"]

print(data[2])  # ❌ ERROR (IndexError)
```

👉 Karena Python mulai dari 0 → index max = 1

🔥 **BONUS (KELEBIHAN PYTHON)**

✔ Ambil sebagian data (slice)

```
data = ["A", "B", "C", "D"]

print(data[1:3])  # ['B', 'C']
```

👉 Ini tidak ada di Turbo Pascal klasik

🎯 **KESIMPULAN**

✔ Cara ambil data → **sama konsepnya (pakai index)**
❗ Bedanya:

- Python mulai dari **0**
- Python lebih fleksibel & powerful

🚀 **SARAN UNTUK ANDA (EX-PASCAL USER)**
Biasakan:
👉 “Index Python = Index Pascal - 1”

Kalau mau, saya bisa lanjutkan ke:
👉 **“Latihan konversi kode Turbo Pascal → Python (step-by-step)”**
atau
👉 **“Membuat array 2 dimensi (matrix) seperti di Pascal”**
Itu akan mempercepat adaptasi Anda secara signifikan.