1. Inheritance adalah konsep dalam pemrograman berorientasi objek di mana sebuah kelas dapat mewarisi properti dan metode dari kelas lain.

# Kelas dasar atau induk

class Kendaraan:

def \_\_init\_\_(self, jenis, roda):

self.jenis = jenis

self.roda = roda

def info\_kendaraan(self):

return f"Jenis: {self.jenis}, Roda: {self.roda}"

# Kelas turunan atau anak

class Mobil(Kendaraan):

def \_\_init\_\_(self, jenis, roda, merek):

# Memanggil konstruktor kelas induk

super().\_\_init\_\_(jenis, roda)

self.merek = merek

def info\_mobil(self):

return f"{self.info\_kendaraan()}, Merek: {self.merek}"

# Membuat objek dari kelas turunan

mobil1 = Mobil("Sedan", 4, "Toyota")

# Memanggil metode dari kelas turunan

print(mobil1.info\_mobil())

2. Encapsulation adalah konsep dalam pemrograman berorientasi objek di mana atribut dan metode dari suatu objek dikemas dalam satu unit, dan akses ke detail internalnya dibatasi.

class RekeningBank:

def \_\_init\_\_(self, pemilik, saldo=0):

self.\_\_pemilik = pemilik # Atribut bersifat private

self.\_\_saldo = saldo

def set\_saldo(self, jumlah):

if jumlah >= 0:

self.\_\_saldo = jumlah

else:

print("Jumlah harus non-negatif.")

def get\_saldo(self):

return self.\_\_saldo

def info\_rekening(self):

return f"Pemilik: {self.\_\_pemilik}, Saldo: {self.\_\_saldo}"

# Membuat objek dari kelas

rekening1 = RekeningBank("John Doe")

# Menggunakan metode untuk mengakses dan mengubah data

rekening1.set\_saldo(1000)

print(rekening1.info\_rekening())

3. Polymorphism adalah konsep dalam pemrograman berorientasi objek di mana suatu objek dapat mengambil bentuk yang berbeda, artinya objek dapat dipanggil dengan menggunakan antarmuka umum yang dimiliki oleh kelas induknya.

class Hewan:

def suara(self):

pass

class Kucing(Hewan):

def suara(self):

return "Meow"

class Anjing(Hewan):

def suara(self):

return "Woof"

class Bebek(Hewan):

def suara(self):

return "Quack"

# Fungsi untuk mendengarkan suara hewan

def dengarkan\_suara(hewan):

print(hewan.suara())

# Membuat objek dari berbagai kelas

kucing1 = Kucing()

anjing1 = Anjing()

bebek1 = Bebek()

# Memanggil fungsi dengan berbagai objek

dengarkan\_suara(kucing1)

dengarkan\_suara(anjing1)

dengarkan\_suara(bebek1)

4. Function exception handler adalah mekanisme dalam pemrograman yang memungkinkan penanganan kesalahan atau exception dalam suatu fungsi atau metode.

def bagi(a, b):

try:

hasil = a / b

return hasil

except ZeroDivisionError:

return "Error: Pembagian dengan nol tidak diizinkan."

except TypeError:

return "Error: Operasi tidak dapat dilakukan pada tipe data ini."

# Contoh pemanggilan fungsi dengan penanganan exception

angka1 = 10

angka2 = 0

hasil\_pembagian = bagi(angka1, angka2)

print(hasil\_pembagian)

angka3 = "sepuluh"

hasil\_pembagian\_str = bagi(angka1, angka3)

print(hasil\_pembagian\_str)

5. GUI (Graphical User Interface) adalah antarmuka pengguna yang memanfaatkan elemen-elemen grafis seperti tombol, kotak teks, jendela, dan lainnya untuk memungkinkan interaksi pengguna dengan suatu program.

import tkinter as tk

from tkinter import messagebox

def tampilkan\_pesan():

messagebox.showinfo("Pesan", "Halo, ini contoh GUI!")

# Membuat jendela utama

jendela\_utama = tk.Tk()

jendela\_utama.title("Contoh GUI")

# Membuat tombol di dalam jendela

tombol = tk.Button(jendela\_utama, text="Klik saya!", command=tampilkan\_pesan)

tombol.pack(pady=20)

# Menjalankan loop utama

jendela\_utama.mainloop()

6. Aplikasi (App) dapat berkomunikasi dengan database untuk menyimpan, mengambil, dan memanipulasi data. Dalam konteks pemrograman, ini sering dilakukan melalui API (Application Programming Interface) database atau ORM (Object-Relational Mapping).

import sqlite3

# Membuat koneksi ke database atau membuat database jika belum ada

koneksi = sqlite3.connect("contoh\_database.db")

# Membuat kursor untuk eksekusi perintah SQL

kursor = koneksi.cursor()

# Membuat tabel jika belum ada

kursor.execute('''

CREATE TABLE IF NOT EXISTS pengguna (

id INTEGER PRIMARY KEY,

nama TEXT,

umur INTEGER

)

''')

# Menambahkan data ke tabel

kursor.execute("INSERT INTO pengguna (nama, umur) VALUES (?, ?)", ("John Doe", 25))

# Menyimpan perubahan dan menutup koneksi

koneksi.commit()

koneksi.close()

7. Untuk membuat aplikasi sederhana yang terkoneksi dengan database dan dapat di-push ke GitHub, apat menggunakan bahasa pemrograman dan framework tertentu. aplikasi sederhana menggunakan Python dan Flask untuk backend, serta SQLite sebagai database.

Langkah 1: Persiapkan Lingkungan

Pastikan Python dan pip telah terinstal di komputer Anda. Kemudian, instal Flask dan Git:

bash

pip install Flask

Langkah 2: Buat Aplikasi Flask

Buat folder untuk proyek Anda dan masuk ke dalamnya:

bash

mkdir myflaskapp

cd myflaskapp

Buat file app.py untuk aplikasi Flask:

from flask import Flask, render\_template

app = Flask(\_\_name\_\_)

@app.route('/')

def home():

return 'Hello, World!'

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

app.run(debug=True)

Langkah 3: Inisialisasi Git

Buka terminal dan jalankan perintah berikut di dalam folder.

bash

git init

Langkah 4: Inisialisasi Repositori GitHub

Buka GitHub, buat repositori baru, dan ikuti instruksi untuk menambahkan repositori kosong. Dapatkan URL repositori yang telah dibuat.

Langkah 5: Hubungkan Lokal dengan Repositori GitHub

Tambahkan remote repository GitHub ke repositori lokal Anda:

bash

git remote add origin <URL-repositori-GitHub>

Langkah 6: Push Kode ke GitHub

Tambahkan semua perubahan, buat commit, dan dorong ke GitHub:

bash

git add .

git commit -m "Initial commit"

git push -u origin master

### Langkah 7: Tambahkan Database SQLite

Perbarui file app.py untuk menambahkan koneksi database SQLite:

python

from flask import Flask, render\_template

from flask\_sqlalchemy import SQLAlchemy

app = Flask(\_\_name\_\_)

app.config['SQLALCHEMY\_DATABASE\_URI'] = 'sqlite:///site.db'

db = SQLAlchemy(app)

class User(db.Model):

id = db.Column(db.Integer, primary\_key=True)

username = db.Column(db.String(20), unique=True, nullable=False)

@app.route('/')

def home():

return 'Hello, World!'

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

app.run(debug=True)