#OOP

1. Pengertian OOP

OOP adalah singkatan dari Object-oriented programming paradigm. OOP didefinisikan sebagai model

pemrograman yang menggunakan konsep objek yang mengacu pada entitas dunia nyata dengan status dan

perilaku. Bab ini membantu Anda menjadi ahli dalam menggunakan dukungan pemrograman berorientasi

objek dalam bahasa Python.

2. Perbedaan

Perilaku setara dengan fungsi. Dalam kehidupan nyata, atribut dan perilaku tidak berdiri sendiri,

melainkan saling terkait. Fitur terpenting dari pendekatan berorientasi objek adalah mendefinisikan

atribut dan fungsionalitasnya sebagai satu unit tunggal yang disebut kelas. Kelas berfungsi sebagai

cetak biru untuk semua objek yang memiliki atribut dan perilaku serupa.

Dalam OOP, kelas mendefinisikan atribut apa saja yang dimiliki objeknya, dan bagaimana perilakunya.

Di sisi lain, objek adalah contoh dari kelas.

Perbedaan OOP (Object-Oriented Programming) bisa dilihat dari berbagai aspek, baik di antara bahasa

pemrograman maupun dalam konsep dasarnya. Berikut adalah beberapa perbedaan yang umum:

1. Konsep Dasar

Kelas dan Objek: Semua bahasa OOP menggunakan konsep kelas (template) dan objek (instansi dari

kelas), tetapi cara mendefinisikan dan menggunakan keduanya bisa berbeda.

Enkapsulasi: Penyembunyian data dan metode di dalam objek. Di beberapa bahasa, akses ke data

dapat dibatasi dengan lebih ketat dibandingkan yang lain.

2. Pewarisan

Pewarisan Tunggal vs. Berganda: Beberapa bahasa (seperti Java) hanya mendukung pewarisan tunggal,

sementara Python dan C++ mendukung pewarisan berganda.

3. Polimorfisme

Static vs. Dynamic Polymorphism: Beberapa bahasa menggunakan polimorfisme statis (misalnya,

overloading), sementara yang lain mengandalkan polimorfisme dinamis, di mana metode dipilih

pada saat runtime.

4. Abstraksi

Abstract Classes dan Interfaces: Beberapa bahasa (seperti Java) memiliki konsep kelas abstrak

dan interface, sementara yang lain (seperti Python) menggunakan metode lain untuk mencapai hal yang sama.

5. Sintaksis

Sintaks yang Berbeda: Setiap bahasa memiliki sintaks yang unik untuk mendefinisikan kelas, metode,

dan objek. Python, misalnya, memiliki sintaks yang lebih sederhana dibandingkan Java atau C++.

3. Contoh

contohnya class:mobil, objek:toyota dan ford, atribut:properti dan metode

4. Prinsip

Paradigma pemrograman berorientasi objek dicirikan oleh prinsip-prinsip berikut −

-Kelas

-Obyek

-Enkapsulasi

-Warisan

-Polimorfisme

#Class, Objek, Atribut

-Pengertian

Kelas (Class): Kelas adalah cetak biru atau template untuk membuat objek. Kelas mendefinisikan atribut

dan metode yang dimiliki oleh objek.

Objek (Object): Objek adalah instansi dari kelas. Setelah kelas didefinisikan, kita bisa membuat objek

dari kelas tersebut.

Atribut (Attribute): Atribut adalah data atau karakteristik yang dimiliki oleh objek. Atribut biasanya

didefinisikan dalam kelas.

Contoh : Sebuah Class "Mobil", objeknya "Toyota" dan "Honda", dan atributnya "merk" "warna" dan "tahun"

Contoh

Sebuah Class "Mobil".

# Definisi kelas Mobil

class Mobil:

# Atribut kelas

def \_\_init\_\_(self, merk, warna, tahun):

self.merk = merk # Atribut merk

self.warna = warna # Atribut warna

self.tahun = tahun # Atribut tahun

# Metode untuk menampilkan informasi mobil

def info(self):

return f'Merk: {self.merk}, Warna: {self.warna}, Tahun: {self.tahun}'

# Membuat objek dari kelas Mobil

mobil1 = Mobil("Toyota", "Merah", 2020)

mobil2 = Mobil("Honda", "Biru", 2021)

# Mengakses atribut dan metode

print(mobil1.info()) # Output: Merk: Toyota, Warna: Merah, Tahun: 2020

print(mobil2.info()) # Output: Merk: Honda, Warna: Biru, Tahun: 2021