



NADHIF RIF'AT RASENDRIYA

# EXPLORING OBJECT- ORIENTED PROGRAMMING (OOP) IN PYTHON

Building a Scientific Calculator  
with Inheritance & Polymorphism concept





# OUTLINE

---



*Apa itu Object-Oriented Programming (OOP)?*

- Kenapa OOP penting dalam pengembangan software?

*Konsep Dasar OOP*

- Class & Object
- Encapsulation, Abstraction, Inheritance, Polymorphism

*Implementasi dalam Scientific Calculator*

- Struktur kode & class hierarchy
- Penjelasan fitur utama (Operasi dasar, Trigonometri, Eksponensial)

*Kesimpulan & Pengembangan Selanjutnya*

- Kelebihan menggunakan OOP dalam proyek ini
- Ide untuk pengembangan lebih lanjut



# OBJECT ORIENTED PROGRAMMING

- Object-Oriented Programming (OOP) adalah paradigma pemrograman yang berfokus pada objek dan interaksi antar objek.
  - OOP mempermudah pengelolaan kode dengan struktur yang lebih modular dan reusable.
- 📌 Contoh OOP dalam Kehidupan Nyata:
- "Mobil" sebagai class
  - "Mobil Toyota, Mobil Honda" sebagai object dengan atribut & metode masing-masing





NADHIF RIF'AT R.

# KONSEP OOP DALAM PYTHON

1. Class & Object → Blueprint & instance dari class
2. Encapsulation → Menyembunyikan detail implementasi dan hanya menyediakan interface
3. Abstraction → Menyederhanakan kompleksitas dengan hanya menampilkan fitur yang penting
4. Inheritance → Pewarisan sifat dari satu class ke class lain
5. Polymorphism → Satu method dapat memiliki bentuk yang berbeda di class turunan





NADHIF RIF'AT R.

# INHERITANCE & POLYMORPHISM DALAM KALKULATOR





# INHERITANCE (PEWARISAN)

---



## ✓ Inheritance (Pewarisan)

Inheritance adalah konsep dalam OOP yang memungkinkan sebuah class (anak/turunan) mewarisi atribut dan metode dari class lain (induk/parent).

## ✓ Implementasi

- Class Scientific (anak/turunan) mewarisi semua metode dari class Calculator (induk/parent).
- Karena pewarisan ini, Scientific Calculator tidak perlu menulis ulang metode dasar seperti:
  - tambah()
  - kurang()
  - kali()
  - bagi()
- Dengan ini, class Scientific bisa langsung menggunakan metode dari Calculator, sambil menambahkan fitur tambahan seperti sin, cos, tan, dan power.

Project1Kalkulator.py > Scientific

```
6  # Super Class / Parent Class
7  class Calculator:
8      Tabnine | Edit | Test | Explain | Document
9      def __init__(self, angka1, angka2):
10         self.angka1 = angka1
11         self.angka2 = angka2
12
13     Tabnine | Edit | Test | Explain | Document
14     def tambah(self):
15         return self.angka1 + self.angka2
16
17     Tabnine | Edit | Test | Explain | Document
18     def kurang(self):
19         return self.angka1 - self.angka2
20
21     Tabnine | Edit | Test | Explain | Document
22     def kali(self):
23         return self.angka1 * self.angka2
24
25     Tabnine | Edit | Test | Explain | Document
26     def bagi(self):
27         return self.angka1 / self.angka2 if self.angka2 != 0 else "Error: Division by Zero!"
28
29
30     # Sub Class / Child Class
31     class Scientific(Calculator):
32         Tabnine | Edit | Test | Explain | Document
33         def sin(self, angka):
34             return math.sin(math.radians(angka))
35
36         Tabnine | Edit | Test | Explain | Document
37         def cos(self, angka):
38             return math.cos(math.radians(angka))
```



# POLYMORPHISM

---



Polymorphism memungkinkan sebuah metode dalam class memiliki perilaku yang berbeda di class turunannya.

📌 Bagaimana Polymorphism digunakan dalam kalkulator ini?

- Dalam class Calculator, method tambah() hanya menjumlahkan dua angka.
- Tetapi dalam class Scientific, kita menambahkan method baru penjumlahan\_sin() yang menggunakan tambah() namun ditambah dengan operasi lain.

```
25 # Sub Class / Child Class
26 class Scientific(Calculator):
27     Tabnine | Edit | Test | Explain | Document
28     def sin(self, angka):
29         return math.sin(math.radians(angka))
30
31     Tabnine | Edit | Test | Explain | Document
32     def cos(self, angka):
33         return math.cos(math.radians(angka))
34
35     Tabnine | Edit | Test | Explain | Document
36     def tan(self, angka):
37         return math.tan(math.radians(angka))
38
39     Tabnine | Edit | Test | Explain | Document
40     def penjumlahan_sin(self):
41         return self.tambah() + self.sin(self.angka1) + self.sin(self.angka2)
42
```



# KESIMPULAN

---

## Keuntungan Menggunakan OOP

- Kode lebih terstruktur dan modular
- Mudah dikembangkan dan dipelihara
- Reusability tinggi

## Pengembangan Selanjutnya

- Menambahkan fitur logaritma & akar kuadrat
- Membuat tampilan GUI berbasis Tkinter atau PyQt
- Integrasi dengan database untuk menyimpan riwayat perhitungan



NADHIF RIF'AT R.

# THANK YOU

Presentation By Nadhif Rif'at R.