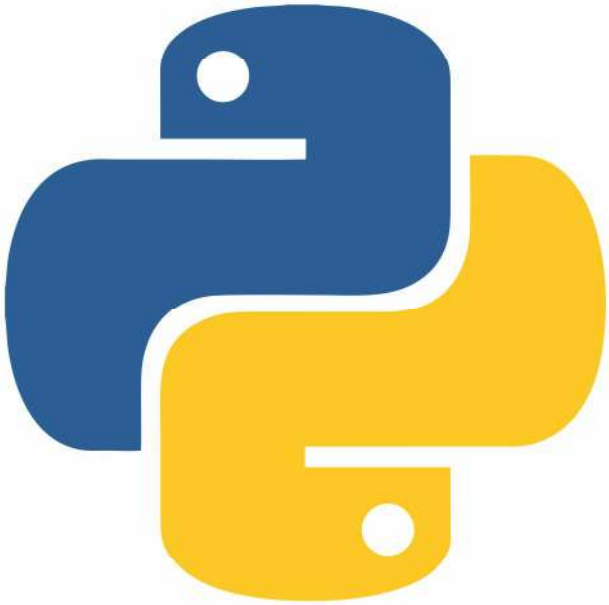
PYTHON

Pemrograman V



Object Oriented Programming

Object Oriented Programming (OOP) merupakan suatu konsep pemrograman yang menekankan pada paradigma atau cara pandang terhadap suatu masalah berdasarkan "object". Dalam konsep OOP semua yang ada didunia ini adalah object dan direpresentasikan dalam bentuk object. Object di dunia nyata memiliki ciri atau attribut dan juga aksi atau kelakuan (behaviour).

12.1 Istilah - Istilah Dalam OOP

Sebelum mempelajari lebih jauh tentang OOP, ada baiknya kita harus mengetahui istilah-istilah dalam OOP, yaitu sebagai berikut :

- **Kelas** adalah cetak biru atau prototipe dari object dimana kita mendefinisikan atribut dari suatu object. Atribut ini terdiri dari data member (variabel) dan fungsi (metode).
- **Variabel Kelas** adalah variabel yang dishare atau dibagi oleh semua instance (turunan) dari kelas. Variabel kelas didefinisikan di dalam kelas, tapi di luar metode-metode yang ada dalam kelas tersebut.
- **Data member** adalah variabel yang menyimpan data yang berhubungan dengan kelas dan objeknya.
- **Overloading Fungsi** adalah fungsi yang memiliki nama yang sama di dalam kelas, tapi dengan jumlah dan tipe argumen yang berbeda sehingga dapat melakukan beberapa hal yang berbeda.
- **Overloading Operator** adalah pembuatan beberapa fungsi atau kegunaan untuk suatu operator. Misalnya operator + dibuat tidak hanya untuk penjumlahan, tapi juga untuk fungsi lain.
- **Variabel Instansiasi** adalah variabel yang didefinisikan di dalam suatu metode dan hanya menjadi milik dari instance kelas.
- **Pewarisan/Inheritansi (Inheritance)** adalah pewarisan karakteristik sebuah kelas ke kelas lain yang menjadi turunannya.
- **Instance** adalah istilah lain dari objek suatu kelas. Sebuah objek yang dibuat dari prototipe kelas Lingkaran misalnya disebut sebagai instance dari kelas tersebut.
- **Instansiasi** adalah pembuatan instance/objek dari suatu kelas.
- **Metode** adalah fungsi yang didefinisikan di dalam suatu kelas.
- **Objek** adalah instansiasi atau perwujudan dari sebuah kelas. Bila kelas adalah prototipenya, dan objek adalah barang jadinya.

12.2 Perkenalan Kelas & Object

oop_dasar.py ▸ ...

```
1   # kelas
2   class Marvel:
3       pass
4
5   # object
6   marvel1 = Marvel()
7   marvel2 = Marvel()
8   marvel3 = Marvel()
9
10  marvel1.name = "Iron Man"
11  marvel1.health = "1000"
12
13  marvel2.name = "Thor"
14  marvel2.health = "800"
15
16  marvel3.name = "Captain America"
17  marvel3.health = "900"
18
19  # pemanggilan
20  print(marvel1)
21  print(marvel1.__dict__)
22  print(marvel1.name)
```

Gambar 92 Perkenalan Kelas dan Object

12.2.1 Penjelasan Kode

- Baris ke-2, adalah **penamaan kelas**, diikuti dengan **titik dua (:)**
- **pass** pada baris ke-3 artinya adalah pernyataan **pass** tidak melakukan apapun. Dapat digunakan ketika sebuah pernyataan diperlukan secara sintaks namun program tidak memerlukan tindakan.
- Baris ke-6 sampai ke-8 adalah **pembuatan object**.
- Baris ke-10 sampai ke-17 adalah proses **pembuatan atribut** sederhana.
- Baris ke-22 adalah **pemanggilan atribut**.

12.3 Kelas dan Object Sederhana

oop_init.py ▸ ...

```
1   class Marvel:
2       def __init__(self, inputName, inputHealth, inputPower, inputArmor):
3           self.name = inputName
4           self.health = inputHealth
5           self.power = inputPower
6           self.armor = inputArmor
7
8   marvel1 = Marvel("Iron Man",100,10,90)
9   marvel2 = Marvel("Thor",90,15,100)
10  marvel3 = Marvel("Captain America",80,5,70)
11
12  print(marvel1.name)
13  print(marvel2.health)
14  print(marvel3.__dict__)
```

Gambar 93 Kelas dan Object Sederhana

12.3.1 Penjelasan Kode

- Metode **__init__()** adalah metode **konstruktor**, yaitu metode khusus yang digunakan Python untuk **menginisialisasi pembuatan objek** dari kelas tersebut.
- Parameter **self**. Nilai dari parameter ini menunjuk ke obyek/instance itu sendiri. Dan **self** digunakan untuk mengakses atribut.
- Baris ke-8 sampai ke-10 adalah **pemberian argumen** atau **atribut**.
- Baris ke-12 adalah **pemanggilan atribut**.

12.4 Variabel Kelas dan Object

oop_kelas_instance.py ▸ ...

```
1   class Marvel:
2       # class variable
3       jumlah = 0
4
5       def __init__(self, inputName, inputHealth, inputPower, inputArmor):
6           # instance variable
7           self.name = inputName
8           self.health = inputHealth
9           self.power = inputPower
10          self.armor = inputArmor
11          Marvel.jumlah += 1
12          print("Hero Marvel dengan nama : " + inputName)
13
14  marvel1 = Marvel("Iron Man",1000,900,800)
15  print(Marvel.jumlah)
16  marvel2 = Marvel("Thor",900,1000,900)
17  print(Marvel.jumlah)
18  marvel3 = Marvel("Captain America",800,700,600)
19  print(Marvel.jumlah)
```

Gambar 94 Variabel Kelas dan Object

12.4.1 Penjelasan Kode

- Baris ke-3 adalah **variabel kelas**. Variabel kelas (class variabel) adalah variabel yang dideklarasikan di dalam kelas dan bertidak sebagai data field. Dalam kasus ini variabel "jumlah" digunakan untuk menghitung jumlah superhero Marvel. Cakupan dari variabel kelas meliputi keseluruhan kelas.
- Baris ke-7 sampai ke-10 adalah **variabel object**. Variabel Object dapat memiliki nilai yang berbeda untuk setiap objek yang diturunkan dari class yang sama. Variabel object dapat diakses melalui object yang diturunkan dari class tersebut.

12.5 Method

oop_method.py ▸ ...

```
1   class Marvel:
2
3       def __init__(self, inputName, inputHealth, inputPower, inputArmor):
4           # instance variable
5           self.name = inputName
6           self.health = inputHealth
7           self.power = inputPower
8           self.armor = inputArmor
9
10      # void function, method tanpa return
11      def siapa(self):
12          print("Namaku adalah : " + self.name)
13
14      # method dengan argumen
15      def healthTambah(self, tambah):
16          self.health += tambah
17
18      # method dengan return
19      def getHealth(self):
20          return self.health
21
22  marvel1 = Marvel("Iron Man",1000,900,800)
23  marvel2 = Marvel("Thor",900,1000,900)
24  marvel3 = Marvel("Iron Man",800,700,600)
25
26  # pemanggilan method
27  marvel1.siapa()
28
29  # pemakaian method dengan argumen
30  marvel1.healthTambah(10)
31  print(marvel1.health)
32
33  # mengembalikan nilai dengan method
34  print(marvel1.getHealth())
```

Gambar 95 Contoh Method

Kevin Perdana, ST
https://medium.com/@kevinperdana

12.6 Game dengan OOP

oop_game.py ▸ ...

```python
class Marvel:

    def __init__(self, name, health, attackPower, armorNumber):
        self.name = name
        self.health = health
        self.attackPower = attackPower
        self.armorNumber = armorNumber

    def serang(self, lawan):
        print(self.name + " menyerang " + lawan.name)
        lawan.diserang(self, self.attackPower)

    def diserang(self, lawan, attackPower_lawan):
        print(self.name + " diserang " + lawan.name)
        attact_diterima = attackPower_lawan
        print("Serangan terasa : " + str(attact_diterima))
        self.health -= attact_diterima
        print("Darah " + self.name + " tersisa " + str(self.health))

ironman = Marvel("Iron Man",100,10,5)
thor = Marvel("Thor",95,15,10)

#ironman.serang()
ironman.serang(thor)
#print("\n")
#ironman.serang(thor)
#print("\n")
#thor.serang(ironman)
```

Gambar 96 Contoh Game dengan OOP

Kevin Perdana, ST
https://medium.com/@kevinperdana