

## Pertemuan4

# ***Konsep Object Oriented Programming pada Python***

### **Objektif:**

1. Mahasiswa dapat mengetahui dan memahami konsep OOP
2. Mahasiswa dapat memahami konsep kelas dan method
3. Mahasiswa dapat membuat program sederhana dengan menggunakan konsep OOP

## P4.1 Teori

### Pendahuluan

Python adalah bahasa pemrograman komputer berorientasi objek, yang berarti bahasa Python ini menyediakan fitur – fitur yang mendukung pemrograman berorientasi objek.

Beberapa karakteristik pemrograman berorientasi objek :

- a. Program – program dibuat dari pendefinisian objek – objek dan fungsi – fungsi, dan kebanyakan perhitungan komputasi diekspresikan kedalam operasi pada objek.
- b. Masing – masing pendefinisian objek merujuk ke beberapa objek atau konsep yang sebenarnya pada dunia nyata, dan fungsi – fungsi pada objek dianalogikan sebagai interaksi pada objek

Perkembangan evolusi pemrograman telah beralih dari pengeksekusian instruksi langkah-demi-langkah menuju kepada pendekatan blok program yang lebih terorganisir, di mana blok kode tersebut dapat dibungkus menjadi subrutin dan fungsi yang telah ditetapkan. Pemrograman terstruktur atau prosedural memungkinkan kita mengatur program ke dalam blok logis, dapat diulang-ulang atau digunakan kembali.

Pemrograman berorientasi objek mengambil langkah evolusi ini dengan meningkatkan program terstruktur untuk memungkinkan data / perilaku hubungan: Data dan logika sekarang digambarkan oleh satu abstraksi yang digunakan untuk menciptakan objek-objek tersebut. Kelas menyediakan pendefinisian dari setiap objek, dan objek merupakan perwujudan dari pendefinisian tersebut. Keduanya (kelas dan objek) adalah dua komponen yang vital dalam Object-Oriented Programming.

Salah satu alasan yang paling penting untuk mempertimbangkan bekerja di OOP adalah bahwa ia menyediakan pendekatan pemodelan langsung dan memecahkan masalah di dunia nyata. Sebagai contoh, mari kita ambil permasalahan toko montir mobil di mana Anda akan mengambil mobil Anda untuk diperbaiki. Ada dua entitas umum yang harus diciptakan: manusia yang berinteraksi dengan dan dalam suatu "sistem", dan lokasi fisik untuk kegiatan-kegiatan yang mendefinisikan sebuah toko montir. Karena terdapat lebih banyak dan berbagai jenis yang pertama, kami akan menjelaskan terlebih dahulu, kemudian menyimpulkan dengan yang kedua.

Sebuah kelas disebut “Person” akan dibuat untuk mewakili semua manusia yang terlibat dalam kegiatan tersebut. Person akan mencakup Pelanggan, Mekanik, dan Kasir. Masing-masing objek tersebut memiliki perilaku (behavior) serupa yang unik. Contohnya, semua objek memiliki method `talk()` dan `drive_car()` sebagai behavior untuk berbicara dan

kemampuan berkendara. Untuk objek Mekanik, ia memiliki behavior `repair_car()`, objek Kasir memiliki behavior `pay()` dan `cash_back()`. Untuk kelas `Person` akan memiliki atribut `drive_licence`, dan untuk objek Mekanik memiliki tambahan atribut yaitu `repair_certification`.

## Kelas

Kelas adalah struktur data yang bisa kita gunakan untuk mendefinisikan objek yang menyimpan data bersama-sama nilai-nilai dan perilaku (behavior). Kelas adalah suatu entitas yang merupakan bentuk program dari suatu abstraksi untuk permasalahan dunia nyata, dan instans dari class merupakan realisasi dari beberapa objek. Jika dianalogikan, kelas itu merupakan blueprint ( cetak biru ) dari sebuah objek (instans).

Dalam Python, pendeklarasian class punya kesamaan seperti mendeklarasikan sebuah fungsi. Berikut adalah bentuk umum pendeklarasian sebuah kelas,

```
class <nama_kelas> :  
    <statemen>  
    <statemen>
```

Pendeklarasian kelas diawali dengan kata kunci **class** kemudian diikuti dengan nama kelasnya. Statemen-statemen dalam tubuh kelas dapat berupa atribut kelas dan method. Kelas umumnya di definisikan pada level teratas dari sebuah modul, dengan begitu objek dari kelas dapat di dibuat dimanapun dalam source code dimana kelas tersebut didefinisikan. Coba jalankan program dibawah ini :

```
>>> class Cetak :  
...     def cetak_sesuatu (self, string):  
...         print "Anda mencetak", string
```

## Atribut Kelas

Atribut merupakan data atau bisa juga berupa fungsi-fungsi yang dimiliki oleh kelas tersebut. Atribut diakses melalui notasi bertitik. Atribut-atribut kelas terikat hanya untuk kelas-kelas dimana atribut tersebut didefinisikan. Atribut-atribut data merupakan variabel-variabel yang kita deklarasikan. Variabel-variabel tersebut dapat digunakan seperti variabel lainnya dan dapat di ubah-ubah nilainya oleh method didalam kelas ataupun di dalam program utama.

Contoh penggunaan Atribut Kelas:

```
>>> class X:
...     bil = 100
...
>>> print X.bil
100
>>> X.bil = X.bil + 10
>>> print X.bil
110
```

## Method

Method merupakan fungsi yang melekat pada sebuah objek atau instan kelas. Contoh berikut menunjukkan penggunaan method dalam kelas.

```
#Badan Class
class TestMethod:
    def perkalian(self,a,b):
        c = a * b
        return c

#program Utama
objek = TestMethod() #instansiasi objek
print(objek.perkalian(50,2))
```

## Method Constructor

Method constructor merupakan sebuah method yang akan otomatis dipanggil ketika objek di instantiasi. Constructor umumnya digunakan untuk melakukan inisialisasi terhadap suatu variabel atau method. Bentuk umum Method constructor adalah sebagai berikut,

```
class <nama_kelas>:
    def __init__(self, <argumen-argumen>):
        <statemen>
        <statemen>
```

Contoh Program 1:

```
class Konstruktor:
    def __init__(self):
        print 'Kalimat ini akan langsung di cetak\n
            Ketika objek dibuat'

# program utama
objekKonst = Konstruktor()
```

### Contoh Program 2 :

```
>>> class Kalimat(praktikum):
...     def __init__(self, kata):
...         self.kata = kata
...         self.cetak()
...         self.awal = self.kata + 'Ini variabel awal'
```

### Method Destructor

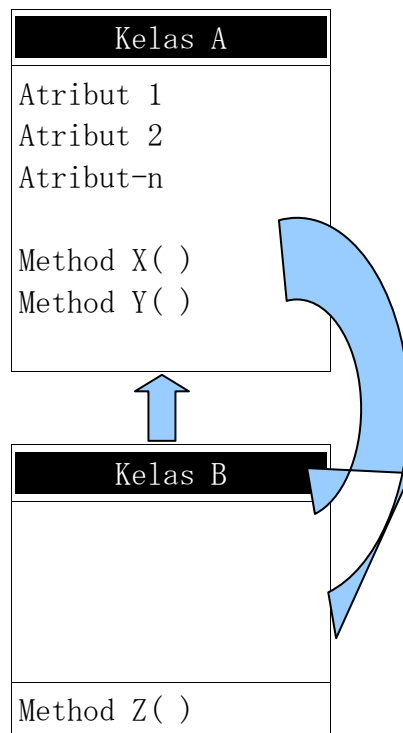
Method destructor dalam python merupakan method yang menyediakan proses khusus sebelum objek di hancurkan/dealokasi. Method constructor disebut `__del__()`. Method ini tidak akan dilaksanakan sampai semua referensi ke semua objek telah dihapus.

### Contoh :

```
>>> class C:
...     def __init__(self):
...         print "inisialisasi"
...     def __del__(self):
...         print "Objek Telah dihapus"
...
>>> c1 = C()
inisialisasi
>>> c2 = c1
>>> c3 = c1
>>> id(c1), id(c2), id(c3)
(11789856, 11789856, 11789856)
>>> del c1
>>> del c2
>>> del c3
Objek Telah dihapus
```

### Inheritance (Pewarisan)

Pewarisan merupakan konsep dalam pemrograman berbasis objek yang memungkinkan untuk membuat suatu kelas dengan didasarkan pada kelas yang sudah ada sehingga mewarisi semua method dan atributnya. Pewarisan merupakan suatu mekanisme yang memungkinkan seorang pemrogram menciptakan suatu kelas baru berdasarkan kelas yang sudah tersedia tetapi tidak perlu menuliskan kode dari nol. Dengan cara seperti ini, semua method dan atribut yang terdapat pada kelas induk diturunkan ke kelas turunannya. Namun kelas turunannya dapat menambah method baru atau atribut baru tersendiri.



Pada contoh diatas, Kelas A merupakan Kelas Induk dan Kelas B disebut Kelas Anak. Ketika Kelas B dideklarasikan sebagai subkelas dari Kelas A, maka Dengan sendirinya Kelas B mewariskan semua atribut atau method yang dimiliki oleh Kelas A. Namun, Kelas B juga dapat membuat method sendiri.

Bentuk Umum pembuatan Kelas turunan,

```
class <nama_kelas_turunan>(<nama_kelas_induk>):
    <atribut-atribut>
    <method-method>
```

Contoh program :

```
class Ayah:
    def methodAyah(self):
        print "Ini adalah Method Ayah"

class Anak(Ayah):
    def methodAnak(self):
        print "Ini adalah Method Anak"

#deklarasi objek kelas Ayah
p = Ayah()
p.methodAyah()
```

```
#deklarasi objek kelas anak
c = Anak()
c.methodAnak()
c.methodAyah()
```

Output :

```
Ini adalah Method Ayah
Ini adalah Method Anak
Ini adalah Method Ayah
```

## Method Overriding

Sebuah method dikatakan Method overriding jika method dengan nama yang sama terdapat pada kelas induk dan kelas anaknya.

Contoh program ;

```
class induk:
    def cobaOverride(self):
        print "Hi... saya method override di kelas
induk"

class turunan(induk):
    def cobaOverride(self):
        print "Hi... saya method override di kelas
anak"

# Deklarasi objek Kelas Induk
objekInduk = induk()
objekInduk.cobaOverride()

# Deklarasi objek kelas anak
objekAnak = turunan()

induk.cobaOverride(objekAnak)
```

Contoh program Inheritance dengan Overriding :

```
class Pegawai:
    def __init__(self, nama, gaji = 0):
        self.nama = nama
        self.gaji = gaji
    def tunjangan(self, persen):
        self.gaji = self.gaji + (self.gaji * persen)
    def kerja(self):
        print(self.nama, "Pekerjaannya")
```

```

    def __repr__(self):
        return "<Pegawai:  nama  =  %s,  gaji  =  %s>" %
(self.nama, self.gaji)

class Koki(Pegawai):
    def __init__(self,nama):
        Pegawai.__init__(self,nama,100000)
    def kerja(self):
        print(self.nama, "Membuat Makanan")

class Pelayan(Pegawai):
    def __init__(self,nama):
        Pegawai.__init__(self,nama, 50000)
    def kerja(self):
        print(self.nama, "Melayani Costumer")

class PizzaRobot(Koki):
    def __init__(self, nama):
        Koki.__init__(self, nama)
    def kerja(self):
        print(self.nama, "Membuat Pizza")

# Program Utama
if __name__ == "__main__":

    agus = PizzaRobot("Agus")
    print(agus)
    agus.kerja()
    agus.tunjangan(0.20)
    print(agus)
    print

    for kelas in Pegawai, Koki, Pelayan, PizzaRobot:
        objek = kelas(kelas.__name__)
        objek.kerja()

```

Output:

```

<Pegawai:  nama = Agus, gaji = 100000>
('Agus', 'Membuat Pizza')
<Pegawai:  nama = Agus, gaji = 120000.0>

('Pegawai', 'Pekerjaannya')
('Koki', 'Membuat Makanan')
('Pelayan', 'Melayani Costumer')
('PizzaRobot', 'Membuat Pizza')

```



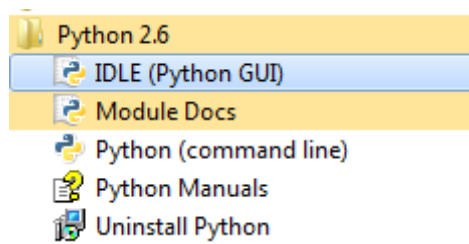
## P4.2 Contoh Kasus

### Contoh Kasus 1

Buatlah program sederhana yang menunjukkan sifat pewarisan dalam bahasa pemrograman Python.

#### Langkah 1

Buka IDLE (Python GUI) yang ada di menu start, lalu klik menu file pilih new window (Ctrl + N).



#### Langkah 2

Ketik listing program berikut :

**INGAT! Penggunaan spasi dan tabulasi dalam penulisan listing. Salah indentasi maka program tidak bisa running!**

```
class Ayah:
    def methodAyah(self):
        print "Ini adalah Method Ayah"

class Anak(Ayah):
    def methodAnak(self):
        print "Ini adalah Method Anak"

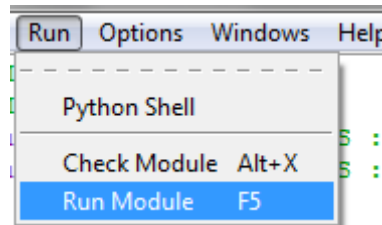
#deklarasi objek kelas Ayah
p = Ayah()
p.methodAyah()

#deklarasi objek kelas anak
c = Anak()
c.methodAnak()
c.methodAyah()
```

Setelah selesai klik menu file → save. Simpan dengan nama pewarisan.py

### Langkah 3

Untuk menjalankan listing program diatas klik menu Run → Run Module F5, maka akan muncul output seperti gambar dibawah ini.



Output :

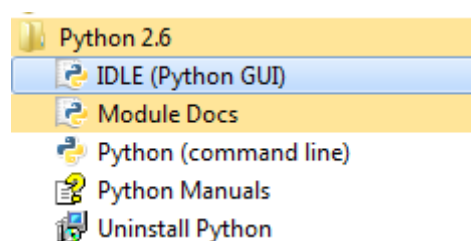
```
Ini adalah Method Ayah  
Ini adalah Method Anak  
Ini adalah Method Ayah
```

### Contoh Kasus 2 :

Buatlah contoh program sederhana yang menunjukkan proses Overiding pada bahasa pemrograman Python :

#### Langkah 1

Buka IDLE ( Python GUI ) yang ada di menu start, lalu klik menu file pilih new window (Ctrl + N).



#### Langkah 2

Ketik listing program berikut :

**INGAT! Penggunaan spasi dan tabulasi dalam penulisan listing. Salah indentasi maka program tidak bisa running!**

```
class Pegawai:
    def __init__(self, nama, gaji = 0):
        self.nama = nama
        self.gaji = gaji
    def tunjangan(self, persen):
        self.gaji = self.gaji + (self.gaji * persen)
    def kerja(self):
        print(self.nama, "Pekerjaannya")
    def __repr__(self):
        return "<Pegawai: nama = %s, gaji = %s>" % (self.nama, self.gaji)

class Koki(Pegawai):
    def __init__(self, nama):
        Pegawai.__init__(self, nama, 100000)
    def kerja(self):
        print(self.nama, "Membuat Makanan")

class Pelayan(Pegawai):
    def __init__(self, nama):
        Pegawai.__init__(self, nama, 50000)
    def kerja(self):
        print(self.nama, "Melayani Costumer")

class PizzaRobot(Koki):
    def __init__(self, nama):
        Koki.__init__(self, nama)
    def kerja(self):
        print(self.nama, "Membuat Pizza")

# Program Utama
if __name__ == "__main__":

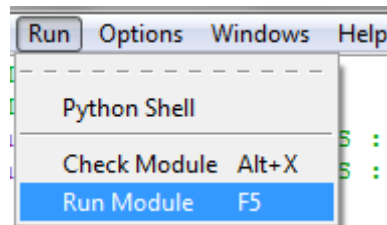
    agus = PizzaRobot("Agus")
    print(agus)
    agus.kerja()
    agus.tunjangan(0.20)
    print(agus)
    print

    for kelas in Pegawai, Koki, Pelayan, PizzaRobot:
        objek = kelas(kelas.__name__)
        objek.kerja()
```

Setelah selesai klik menu file → save. Simpan dengan nama overriding.py

### Langkah 3

Untuk menjalankan listing program diatas klik menu Run → Run Module F5, maka akan muncul output seperti gambar dibawah ini.



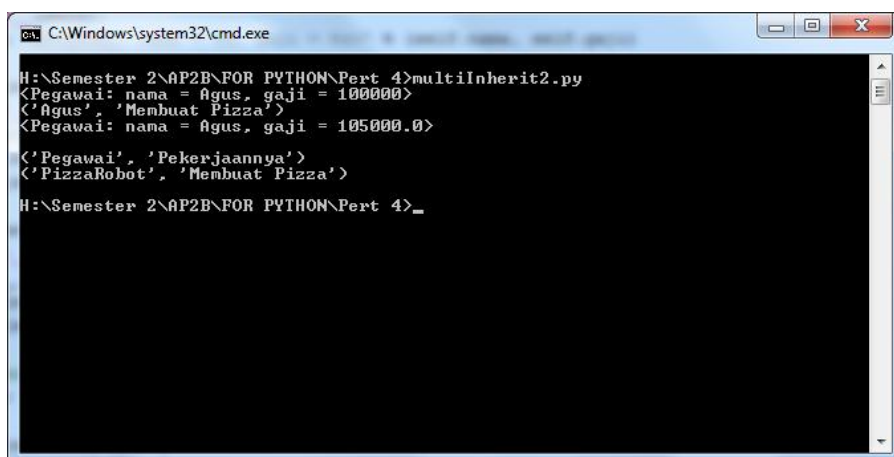
Output:

```
<Pegawai: nama = Agus, gaji = 100000>
('Agus', 'Membuat Pizza')
<Pegawai: nama = Agus, gaji = 120000.0>

('Pegawai', 'Pekerjaannya')
('Koki', 'Membuat Makanan')
('Pelayan', 'Melayani Costumer')
('PizzaRobot', 'Membuat Pizza')
```

### P4.3 Latihan

Lengkapilah program dibawah ini sehingga didapat Output seperti Berikut :



```
C:\Windows\system32\cmd.exe
H:\Semester 2\AP2B\FOR PYTHON\Pert 4>multiInherit2.py
<Pegawai: nama = Agus, gaji = 100000>
('Agus', 'Membuat Pizza')
<Pegawai: nama = Agus, gaji = 105000.0>

('Pegawai', 'Pekerjaannya')
('PizzaRobot', 'Membuat Pizza')
H:\Semester 2\AP2B\FOR PYTHON\Pert 4>_
```

```

class Pegawai:
    def __init__(self, nama, gaji = 0):
        .....#5
    def tunjangan(self, persen):
        self.gaji = self.gaji + (self.gaji * persen)
    def kerja(self):
        print(self.nama, "Pekerjaannya")
    def __repr__(self):
        return "<Pegawai: nama = %s, gaji = %s>" % (self.nama, self.gaji)

class Koki(Pegawai):
    def __init__(self, nama):
        Pegawai.__init__(self, nama, 100000)
    def kerja(self):
        print(self.nama, "Membuat Makanan")

class PizzaRobot(Koki):
    .....#6
    Koki.__init__(self, nama)
    def kerja(self):
        print(self.nama, "Membuat Pizza")

# Program Utama
.....#7

    agus = PizzaRobot("Agus")
    .....#8
    agus.kerja()
    agus.tunjangan(0.10)
    print(agus)
    print

    #.....#9
        objek = kelas(kelas.__name__)
        .....#10

```

## P4.4 DaftarPustaka

- [1.] <http://www.master.web.id/mwmag/issue/01/content/tutorial-python-1/tutorial-python-1.html>, 27 Februari 2012
- [2.] <http://www.scribd.com/doc/30882425/Tutorial-Python-2>, 28 Februari 2012
- [3.] [http://id.wikibooks.org/wiki/Python\\_Selayang\\_Pandang](http://id.wikibooks.org/wiki/Python_Selayang_Pandang), 29 Februari 2012