

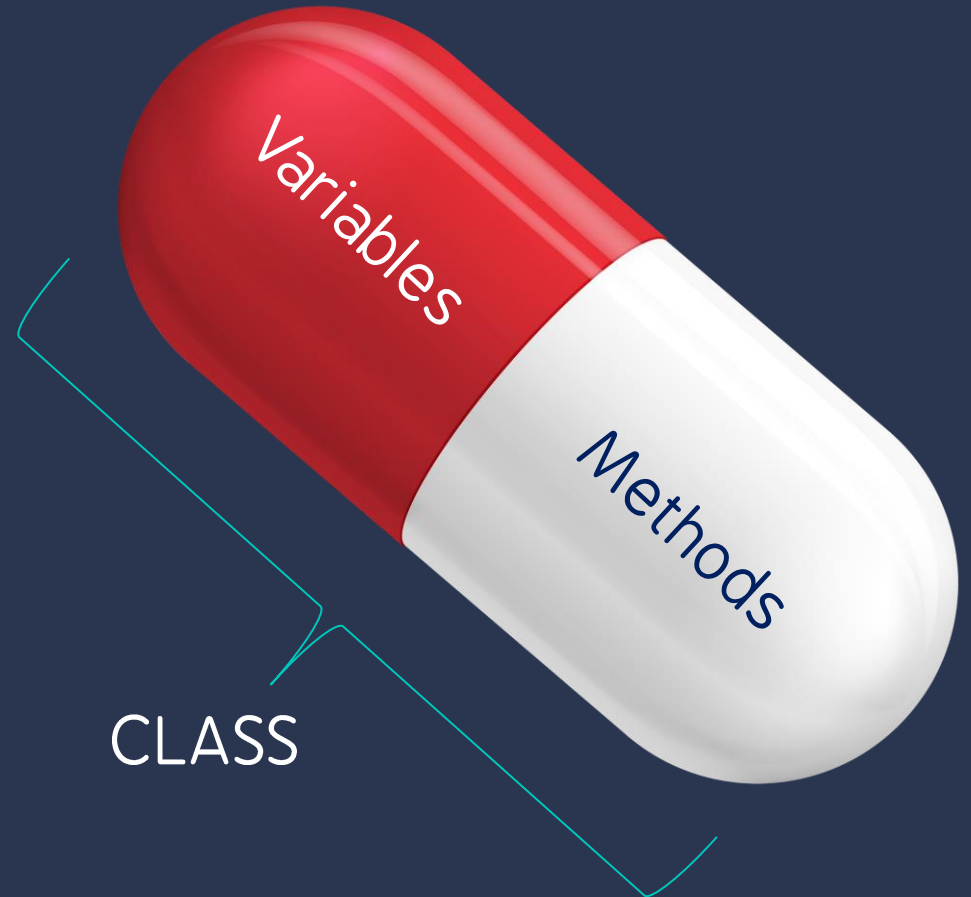
Encapsulation

TI32014 - Pemrograman Berorientasi Object

Andik Yulianto

APA ITU ENCAPSULATION (EN**KAPSULASI**)?

- Enkapsulasi merupakan teknik untuk mengemas data dan *methods* dalam satu unit.
- Enkapsulasi dapat memberikan batasan akses ke variabel dan methods secara langsung → untuk menghindari modifikasi data secara tidak sengaja



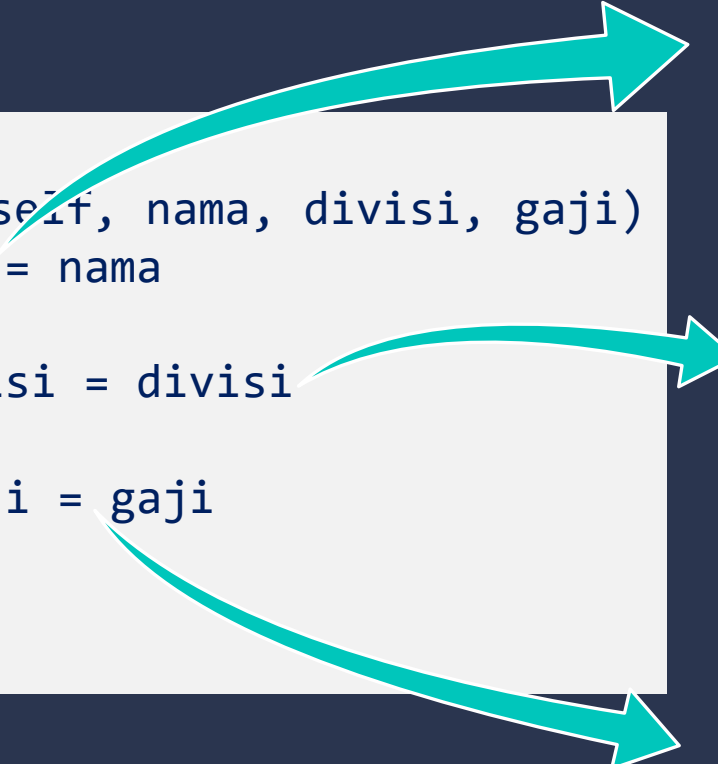
Enkapsulasi pada python



- Dengan membuat *method* tertentu “tersembunyi” dari luar object
- Menggunakan *underscore* (tanda “_”) di awal nama *method*

Enkapsulasi pada python

```
class Pegawai:  
    def __init__(self, nama, divisi, gaji)  
        self.nama = nama  
  
        self._divisi = divisi  
  
        self.__gaji = gaji
```



public attribute

Dapat diakses dari dalam atau luar class

protected attribute

*Dapat diakses dari dalam class atau sub-class.
Menggunakan satu tanda underscore _*

private attribute

*Hanya dapat diakses dalam class.
Menggunakan dua tanda underscore __*

Public Access Modifier

Variabel atau atribut sebuah Class yang memiliki hak akses public dapat diakses dari mana saja baik dari dalam maupun luar Class

```
class Pegawai:
    # constructor
    def __init__(self, nama, salary):
        # public data members
        self.nama = nama
        self.gaji = salary
        self.gaji_final = self.gaji + (0.2 * self.gaji)
```

```
# membuat objek
wahyu = Pegawai('Wahyu', 10000)
# mengakses public data members
print ('Gaji Final:', wahyu.gaji_final)
```

```
Gaji Final: 12000.0
```

Protected Access Modifier

Variabel atau atribut yang memiliki hak akses Protected “**seharusnya**” hanya dapat diakses dari internal Class tersebut dan juga Class turunannya

```
class Pegawai:
    # constructor
    def __init__(self, nama, salary):
        # public data members
        self.nama = nama
        self.gaji = salary
        self.gaji_final = self.gaji + (0.2 * self.gaji)
    # membuat objek
    wahyu = Pegawai('Wahyu', 10000)

    # mengakses public variable
    print ('Gaji Final:', wahyu._gaji_final)
```

Gaji Final: 12000.0



Hanya penanda untuk
programmer

Untuk memberikan akses Protected kita perlu tambahkan satu tanda **underscore** `_` di depan atribut tersebut

Private Access Modifier

Setiap variabel atau atribut dengan hak akses private hanya dapat diakses dari internal Class tersebut, tidak dapat diakses dari luar Class bahkan dari Class turunan

Untuk memberikan akses Protected kita perlu tambahkan satu tanda **underscore** `_` di depan atribut tersebut

```
class Pegawai:
    # constructor
    def __init__(self, nama, salary):
        # public data members
        self.nama = nama
        self.gaji = salary
        self.__gaji_final = self.gaji + (0.2 *
self.gaji)

# membuat objek
wahyu = Pegawai('Wahyu', 10000)

# mengakses private variable
print ('Gaji Final:', wahyu.__gaji_final)
```

```
AttributeError: 'Pegawai' object has no attribute
'__gaji'
```

Bagaimana agar akses dari dalam Class?

Tambahkan method untuk akses private atribut



```
class Pegawai:
    # constructor
    def __init__(self, nama, salary):
        # public data members
        self.nama = nama
        self.gaji = salary
        self.__gaji_final = self.gaji + (0.2 * self.gaji)

    def showGajiFinal(self):
        print ('Gaji Akhir: ', self.__gaji_final)

# membuat objek
wahyu = Pegawai('Wahyu', 10000)

# menampilkan atribut private
wahyu.showGajiFinal()
```

Gaji Final: 12000.0

Thank You