MODUL PRAKTIKUM
PEMROGRAMAN BERORIENTASI OBJEK

Minggu 2

OBJEK DAN KELAS DALAM PYTHON
(KONSTRUKTOR, SETTER, DAN GETTER)

Disusun Oleh :

Andhika Putra Pratama	119140224
Andhika Wibawa B.	119140218
Nurul Aulia Larasati	119140088
Ihza Fajrur Rachman H.	119140130
Enrico Johanes S.	119140021
M. Ammar Fadhila R.	119140029

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK ELEKTRO, INFORMATIKA DAN SISTEM FISIS
INSTITUT TEKNOLOGI SUMATERA
2022

1. Kelas

Kelas atau *class* pada python bisa kita katakan sebagai sebuah blueprint (cetakan) dari objek (atau instance) yang ingin kita buat. Dengan menggunakan kelas kita dapat mendesain objek secara bebas. Kelas berisi dan mendefinisikan atribut/properti dan metode untuk objeknya nanti. Satu buah kelas dapat membuat banyak objek dan kelas sendiri tidak bisa langsung digunakan, harus diimplementasikan menjadi sebuah objek dulu, dapat disebut Instansiasi. contohnya: kelas Mobil, kelas Manusia, dan kelas Kucing.

Untuk membuat kelas, gunakan kata kunci **class** diikuti oleh **nama kelas** tersebut dan tanda titik dua. Contoh:

```
1   class mobil:
2       def __init__(self, merk, warna, tahun):
3           self.merk = merk
4           self.warna = warna
5           self.tahun = tahun
6       def __str__(self):
7           return "{} {} {}".format(self.merk, self.warna, self.tahun)
8
9   mobil1 = mobil("Toyota", "Hitam", "2018")
10  mobil2 = mobil("Honda", "Merah", "2017")
11
12  print(mobil1)
13  print(mobil2)
```

Dapat dilihat diatas terdapat __init__(), method tersebut adalah konstruktor untuk membuat kelas mobil. Untuk method __init__() lebih lanjut akan dibahas pada pembahasan selanjutnya. Kelas juga dapat dibuat kosong dengan menggunakan kata kunci ***pass*** seperti contoh berikut ini:

```
1   class mobil:
2       pass
3
4   mobil1 = mobil() # mobil1 adalah objek dari class mobil
```

Kelas mungkin merupakan cara termudah bagi kita untuk mengorganisasikan data ketika melakukan komputasi ilmiah. Meskipun demikian hal ini bukan tanpa kelemahan, terkadang dengan mengorganisasikan data kita ke dalam kelas, program yang kita buat menjadi lambat ketika dijalankan. Seperti yang sudah dijelaskan diatas bahwa di dalam

sebuah kelas terdapat variabel (atribut/properti) dan fungsi (method).

a. Atribut/Property

Dalam suatu kelas, umumnya dideklarasikan sebuah variabel yang akan disebut sebagai sebuah atribut. Karena dideklarasikan di dalam sebuah kelas, maka setiap objek yang dibentuk dari kelas tersebut juga akan memiliki atribut yang dimiliki oleh kelas tersebut. Terdapat 2 jenis atribut, yaitu atribut kelas dan atribut objek. Atribut kelas merupakan sifat yang dimiliki oleh sebuah kelas dan juga akan dimiliki oleh setiap objek. Sedangkan atribut objek adalah sebuah atribut dari masing - masing objek.

Contoh :

```python
class Orang:
    jumlah_kaki = 2
    jumlah_tangan = 2

    def __init__(self, nama, pekerjaan, tahun_lahir):
        self.nama = nama
        self.pekerjaan = pekerjaan
        self.tahun_lahir = tahun_lahir

orang1 = Orang("Enrico", "Mahasiswa", 2000)
orang2 = Orang("Andi", "Direktur", 1980)
print(orang1.nama)
print(orang1.pekerjaan)
print(orang2.nama)
print(orang2.pekerjaan)
print(orang1.jumlah_kaki)
print(orang2.jumlah_kaki)
```

Output :

```
Enrico
Mahasiswa
Andi
Direktur
2
2
```

Dari contoh diatas dapat kita lihat bahwa jumlah_tangan dan jumlah_kaki merupakan atribut kelas yang nilainya akan sama dalam setiap objek yang dibuat menggunakan kelas tersebut. Sedangkan nama, pekerjaan, dan tahun_lahir merupakan atribut objek yang akan memiliki nilai dari masing - masing objek.

Untuk memanggil atribut dapat dengan syntax :

```
nama_objek.nama_atribut
```

b. Method

Selain atribut, elemen lain yang terdapat dalam suatu kelas adalah method. Method adalah suatu fungsi yang terdapat di dalam kelas. Sama halnya seperti atribut, semua objek yang dibuat menggunakan kelas yang sama akan memiliki method yang sama pula. Method dapat diibaratkan sebagai sebuah aktivitas/proses yang dapat dilakukan oleh sebuah objek. Misalkan objek manusia dapat makan, berjalan, mencuci_piring.

Contoh :

```python
class Orang:
    jumlah_kaki = 2
    jumlah_tangan = 2

    def __init__(self, nama, pekerjaan, tahun_lahir):
        self.nama = nama
        self.pekerjaan = pekerjaan
        self.tahun_lahir = tahun_lahir
    def berjalan_kedepan(self):
        print("Melangkah")
        print("Posisi berpindah")

orang1 = Orang("Enrico", "Mahasiswa", 2000)
orang1.berjalan_kedepan()
```

Output :

```
Melangkah
Posisi berpindah
```

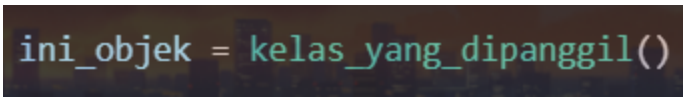
Dicontohkan bahwa kelas Orang memiliki method berjalan_kedepan() yang akan mencetak “Melangkah” dan “Posisi berpindah”. Untuk memanggil method mirip seperti memanggil atribut, namun dengan tambahan tanda kurung “()” seperti berikut :

```
nama_objek.nama_method()
```

2. Objek

Objek adalah sesuatu yang "mewakili" kelas. Objek disini berfungsi sebagai pengganti pemanggilan sebuah kelas, maka sebuah objek hanya dapat mewakili sebuah kelas saja. Untuk membuat sebuah objek yang mewakili sebuah kelas, kita dapat membuatnya dengan cara memanggil nama dari kelas yang diinginkan, ditambah dengan tanda kurung ().

Contoh : Diibaratkan kita memiliki sebuah kelas bernama "kelas_yang_dipanggil", maka untuk membuat objek bernama "ini_objek" cukup seperti ini :



Pembuatan objek mempermudah pemanggilan kelas, terutama ketika nama dari suatu kelas terlalu panjang dan kelas tersebut kadang perlu berinteraksi dengan kelas lain.

3. Magic Method

Magic method adalah metode yang diawali dan diakhiri dengan *double underscore* (dunder). Method ini tidak dipanggil secara langsung, tapi dipanggil sistem secara internal ketika melakukan sesuatu seperti menggunakan operator tambah (__add__), membuat objek (__init__), dan lain-lain.

Tujuan utama dari penggunaan magic method adalah untuk mengubah sifat (behavior) bawaan dari suatu objek. Untuk melihat apa saja magic method bawaan python pada class int, gunakan sintaks dir(int).

```
>>> dir(int)
['__abs__', '__add__', '__and__', '__bool__', '__ceil__', '__class__', '__delattr__',
'__dir__', '__divmod__', '__doc__', '__eq__', '__float__', '__floor__', '__floordiv__',
'__format__', '__ge__', '__getattribute__', '__getnewargs__', '__gt__', '__hash__',
'__index__', '__init__', '__init_subclass__', '__int__', '__invert__', '__le__',
'__lshift__', '__lt__', '__mod__', '__mul__', '__ne__', '__neg__', '__new__', '__or__',
'__pos__', '__pow__', '__radd__', '__rand__', '__rdivmod__', '__reduce__', '__reduce_ex__',
'__repr__', '__rfloordiv__', '__rlshift__', '__rmod__', '__rmul__', '__ror__', '__round__',
'__rpow__', '__rrshift__', '__rshift__', '__rsub__', '__rtruediv__', '__rxor__',
'__setattr__', '__sizeof__', '__str__', '__sub__', '__subclasshook__', '__truediv__',
'__trunc__', '__xor__', 'bit_length', 'conjugate', 'denominator', 'from_bytes', 'imag',
'numerator', 'real', 'to_bytes']
```

Magic method tidak terbatas pada kelas int saja, tetapi ada di setiap jenis objek dan variabel. Salah satu contoh magic method yaitu __add__(). Method ini ditambahkan agar user dapat melakukan operasi penambahan (+) secara langsung pada objek, tanpa mengakses atribut isi objek (dalam hal ini yaitu self.angka). Tanpa penambahan magic method __add__(), maka statement print(N + P) akan menghasilkan error.

```
1   class Angka:
2     def __init__(self, angka):
3       self.angka = angka
4
5     def __add__(self, objek):
6       return self.angka + objek.angka
7
8   N=Angka(1)
9   P=Angka(2)
10  print (N + P)
```

output:

```
[Running] python -u "d:\itera\sm 6\asprak pbo\mg 3.py"
3
```

4. Konstruktor

Konstruktor adalah *method* yang "pasti" dijalankan secara otomatis pada saat sebuah objek dibuat untuk mewakili kelas tersebut. Seperti dengan *method* lain, konstruktor dapat melakukan operasi seperti melakukan print. Selain operasi *method* dasar, konstruktor dapat menerima argumen yang diberikan ketika objek dibuat. Argumen ini akan diproses di dalam kelas nantinya.

Contoh :

```python
class kelas_yang_dipanggil:

    #ini constructor
    def __init__(self,ini_argumen,ini_argumen_juga):
        self.nama = ini_argumen
        self.pekerjaan = ini_argumen_juga

    #ini buat panggil nama
    def panggilnama (self):
        print("Orang ini bernama", self.nama)
    #ini buat panggil kerjaan
    def panggilpekerjaan (self):
        print("Orang ini bekerja sebagai",self.pekerjaan)

ini_objek = kelas_yang_dipanggil("Zhongli","Tukang Gali Kubur")

ini_objek.panggilnama()
ini_objek.panggilpekerjaan()
```

Output :

```
PS D:\ngasprak\pbo\mg02>  & 'C:\Users\ammar\Ap
2022.0.1814523869\pythonFiles\lib\python\debug
Orang ini bernama Zhongli
Orang ini bekerja sebagai Tukang Gali Kubur
PS D:\ngasprak\pbo\mg02>
```

Argumen juga bisa dimasukkan menggunakan input pengguna.

```python
namaa = str(input("Nama = "))
work = str(input("Pekerjaan = "))
ini_objek = kelas_yang_dipanggil(namaa,work)
```

```
PS D:\ngasprak\pbo\mg02>  & 'C:\Users\ammar\AppData\Local\
2022.0.1814523869\pythonFiles\lib\python\debugpy\launcher'
Nama = Come Bink
Pekerjaan = Tukang aduk sampah profesional
Orang ini bernama Come Bink
Orang ini bekerja sebagai Tukang aduk sampah profesional
PS D:\ngasprak\pbo\mg02>
```

5. Destruktor

Destruktor adalah fungsi yang dipanggil ketika user menghapus objek. Fungsi ini bekerja secara otomatis, jadi tidak perlu dilakukan pemanggilan. Tujuan adanya destruktor adalah melakukan final cleaning up atau “bersih-bersih” terakhir sebelum

sebuah objek benar-benar dihapus dari memori.

```
class Angka:
    def __init__(self, angka):
        self.angka = angka

    def __del__(self):
        print ("objek {self.angka} dihapus")

N=Angka(1)
P=Angka(2)
```

output:

```
[Running] python -u "d:\itera\sm 6\asprak pbo\mg 3.py"
objek {self.angka} dihapus
objek {self.angka} dihapus

[Done] exited with code=0 in 0.161 seconds
```

6. Setter dan Getter

Setter dan getter digunakan untuk melakukan enkapsulasi agar tidak terjadi perubahan data secara tidak sengaja. Setter adalah method yang digunakan untuk menetapkan nilai suatu atribut khususnya atribut private dan protected (akan diajarkan lebih jelas pada materi minggu selanjutnya), sedangkan getter digunakan untuk mengambil nilai.

```python
class siswa:
    def __init__(self, umur = 0):
        self._umur = umur

    # getter method
    def get_umur(self):
        return self._umur

    # setter method
    def set_umur(self, x):
        self._umur=x

raj = siswa()

# setting the umur using setter
raj.set_umur(19)
# retrieving umur using getter
print(raj.get_umur())
print(raj._umur)
```

Output:

```
PS D:\itera\Asprak PBO 2022>
Asprak PBO 2022/mg 2.py"
19
19
PS D:\itera\Asprak PBO 2022>
```

Terlihat dari contoh di atas, terdapat 2 cara untuk mengakses atribut protected, yaitu menggunakan getter dan memanggil nama atribut secara langsung. Berbeda dengan atribut protected, atribut private tidak dapat diakses secara langsung tanpa menggunakan getter seperti contoh di bawah.

```python
class siswa:
    def __init__(self, umur = 0):
        self.__umur = umur

    # getter method
    def get_umur(self):
        return self.__umur

    # setter method
    def set_umur(self, x):
        self.__umur=x

raj = siswa()

# setting the umur using setter
raj.set_umur(19)
# retrieving umur using getter
print(raj.get_umur())
print(raj.__umur)
```

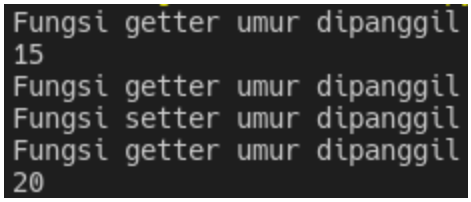
```
Asprak PBO 2022/mg 2.py"
19
Traceback (most recent call last):
  File "d:\itera\Asprak PBO 2022\mg 2.py", line 19, in <module>
    print(raj.__umur)
AttributeError: 'siswa' object has no attribute '__umur'
PS D:\itera\Asprak PBO 2022>
```

7. Decorator

Selain menggunakan fungsi setter dan getter tambahan seperti pada contoh sebelumnya, dalam Python kita juga dapat memanfaatkan property decorator untuk mendapatkan hasil serupa. Bedanya, melalui property decorator ini kita tidak perlu membuat fungsi lagi dengan nama yang berbeda-beda (cukup menggunakan 1 buah nama/variabel). Contoh penggunaan decorator:

```python
class Siswa:
    def __init__(self, nama, umur = 15):
        self.__nama = nama
        self.__umur = umur

    @property
    def umur(self):
        print("Fungsi getter umur dipanggil")
        return self.__umur

    @umur.setter
    def umur(self, x):
        print("Fungsi setter umur dipanggil")
        self.__umur = x

Bambang = Siswa("Bambang")

# Akan error karena bersifat private
#print(Bambang.__umur)

# Gunakan fungsi property decorator
print(Bambang.umur)
Bambang.umur += 5
print(Bambang.umur)
```

Hasil eksekusi dari kode di atas adalah sebagai berikut:


Fungsi getter umur dipanggil
15
Fungsi getter umur dipanggil
Fungsi setter umur dipanggil
Fungsi getter umur dipanggil
20