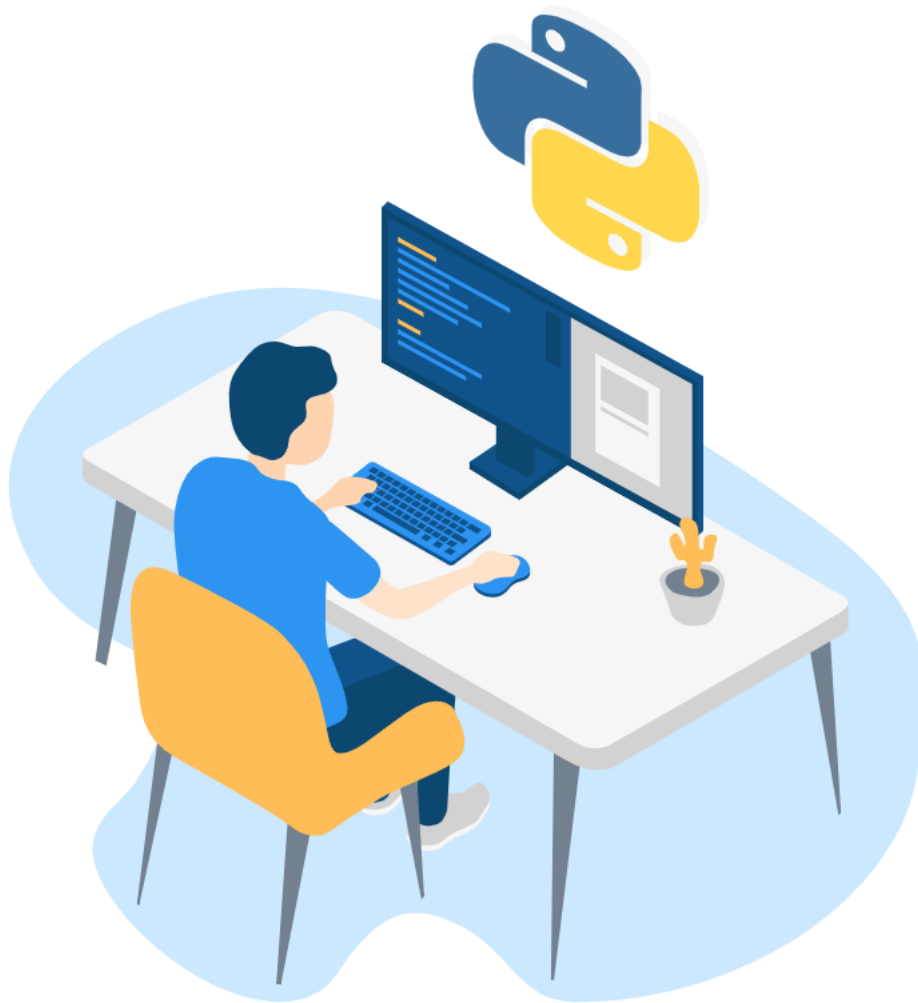


## BAB 1

### Class dan Object



#### 1.1 Tujuan

1. Dapat menjelaskan class dan object pada bahasa python.
2. Dapat mengimplementasikan class dan object pada pemrograman bahasa python

## 1.2 Pengantar

Pemrograman Berorientasi Object (PBO) merupakan salah satu paradigma yang diterapkan hampir di seluruh bahasa pemrograman. Konsep dasar dari PBO adalah mengumpulkan data dan fungsi yang memiliki hubungan kedalam suatu pulau informasi. Pulau ini disebut dengan object.

Jika dibandingkan dengan bahasa pemrograman prosedural, PBO akan melihat suatu masalah secara keseluruhan. Dalam bahasa pemrograman prosedural suatu masalah akan dipecahkan dengan cara memanggil prosedur yang biasa disebut dengan function. Dalam PBO, alih-alih berurusan dengan data secara langsung PBO akan memahami data mana yang akan digunakan dengan cara melakukan modeling. Untuk melakukan modeling ini ada beberapa istilah yang perlu dipahami yaitu class dan object

### 1.2.1 Pengenalan Class dan Object

Class merupakan sebuah blueprint dari object yang akan kita buat. Class berarti cetakannya sedangkan object (instance) adalah hasil dari cetakan tersebut. Untuk memahami secara lebih jelas perhatikan tabel berikut ini.

Tabel 8.1 Contoh Class dan Object

Class	Object/Instance
Car	Honda Jazz, Toyota Avanza, Suzuki Jimny
Cat	Persia, Siam, Kucing Bengal
Coffe	Americano, Capucino, Late
Dog	Labrador, Husky, Bulldog, Doberman

Untuk membuat sebuah class kita dapat menggunakan kode berikut ini. Nama sebuah kelas harus diawali dengan huruf kapital.

```
1. class NamaKelas:  
2.     # atribut atau metode yang digunakan di class ini
```

Contoh dalam pembuatan class sebuah Dog

```
1. class Dog:
2.     def __init__(self, nama, umur):
3.         self.nama = nama
4.         self.umur = umur
5.
6.     def duduk(self):
7.         print(f"{self.nama} sekarang duduk")
8.
9.     def berdiri(self):
10.        print(f"{self.nama} sekarang berdiri")
11.
```

Kode diatas akan dijelaskan pada sub bab dibawah ini

### 1.2.2 Method `__init__`

Fungsi yang berada dalam sebuah class dinamakan dengan method. Semua aturan fungsi yang sudah kita pelajari di bab sebelumnya berlaku juga pada pembuatan method. Jika kita lihat pada baris ke-2 terdapat method `__init__`, method ini merupakan spesial method yang secara otomatis akan berjalan setiap kali object/instance dari class Dog dibuat. Penulisan method ini diawali dengan dua kali underscore ( `_` ) dan diakhiri juga dengan dua kali underscore ( `_` ). Tanpa penulisan dua kali underscore di awal dan di akhir ini maka python tidak akan menjalankan method ini secara otomatis.

### 1.2.3 Parameter `self`

Pada method `__init__` terdapat tiga parameter yaitu `self`, `nama` dan `umur`. Parameter `self` merupakan sebuah parameter yang harus ada didalam pembuatan method, dan harus ditulis di awal sebelum parameter lainnya. Jika kita lihat pada method `duduk` dan `berdiri`, parameter `self` harus ditulis walaupun tidak ada parameter lain di dalam method tersebut.

Parameter `self` ini berfungsi untuk mendapatkan akses secara internal terhadap atribut atau method didalam sebuah class saat kita membuat object/instance.

### 1.2.4 Membuat Object/Instance dari class Dog

Jika sebuah class adalah blueprint maka object adalah hasil cetakannya. Mari kita buat object dari class Dog

```
1. class Dog:
2.     def __init__(self, nama, umur):
3.         self.nama = nama
4.         self.umur = umur
5.
6.     def duduk(self):
7.         print(f"{self.nama} sekarang duduk")
8.
9.     def berdiri(self):
10.        print(f"{self.nama} sekarang berdiri")
11.
12. my_dog = Dog("Labrador",6)
13.
14. print(f"anjingku bernama {my_dog.nama}")
15. print(f"anjingku berumur {my_dog.umur} tahun ")
```

Baris ke 12 merupakan cara pembuatan object my\_dog berdasarkan class Dog. Pembuatan object harus menggunakan aturan lowercase. Pada pembuatan object my\_dog kita mengirim dua variabel yaitu Labrador dan 6. Parameter ini disesuaikan dengan kebutuhan method \_\_init\_\_ pada class Dog. Ingat parameter self akan diproses secara otomatis sehingga kita hanya perlu mengirim variabel untuk parameter nama dan umur. Jika baris ke 12 ini dijalankan maka python akan membuat sebuah object bernama my\_dog dengan nama = Labrador dan umur = 6

### 1.2.5 Mengakses Attribute

Untuk mengakses atribut pada sebuah objek/instance kita menggunakan bantuan titik ( . ) perhatikan baris ke 14 dan 15 dari kode diatas. Saat akan mengakses atribut nama dari object my\_dog penulisannya menjadi

```
1. {my_dog.nama}
```

Dengan bantuan titik (.) disini maka python akan mencari objek dengan nama `my_dog` kemudian mencari atribut nama di object `my_dog` tersebut. Jika baris 14 dan 15 dijalankan maka akan menghasilkan output seperti berikut

```
anjingku bernama Wili
anjingku berumur 6 tahun
```

### 1.2.6 Memanggil Method

Jika kita sudah membuat object dari suatu class maka kita dapat menggunakan method yang ada pada class tersebut. Perhatikan kode dibawah ini

```
1. class Dog:
2.     def __init__(self, nama, umur):
3.         self.nama = nama
4.         self.umur = umur
5.
6.     def duduk(self):
7.         print(f"{self.nama} sekarang duduk")
8.
9.     def berdiri(self):
10.        print(f"{self.nama} sekarang berdiri")
11.
12. my_dog = Dog("Wili",6)
13.
14. print(f"anjingku bernama {my_dog.nama}")
15. print(f"anjingku berumur {my_dog.umur} tahun")
16.
17. my_dog.duduk()
18. my_dog.berdiri()
```

Perhatikan baris ke 17 dan 18, hampir mirip seperti pengaksesan atribut, pemanggilan method juga menggunakan bantuan titik (.) Kita tulis nama object terlebih dahulu, beri tanda titik, diakhiri dengan nama method yang akan dipanggil. Jika kode tersebut dijalankan maka akan menghasilkan output seperti berikut :

```
anjingku bernama Wili
anjingku berumur 6 tahun
Wili sekarang duduk
Wili sekarang berdiri
```

### 1.2.7 Memanggil Method dengan parameter

Perhatikan kembali contoh kode pada class Dog berikut ini

```
my_dog = Dog("Wili", 6)
```

merupakan cara pembuatan objek my\_dog dari Class Dog dengan dua paramater yang terdiri dari string dan integer, yaitu Wili dan 6. Aturan pembuatan objek dengan paramater seperti ini memiliki aturan yang sama dengan pemanggilan function pada pemrograman prosedural

## 2.1 Kegiatan Praktikum

### 2.1.1 Kegiatan 1 : Class dengan satu instance

1. Buat sebuah file program baru kemudian tulis kode berikut ini

```
1. class LightSwitch():
2.     def __init__(self):
3.         self.switchIsOn = False
4.
5.     def turnOn(self):
6.         # turn the switch on
7.         self.switchIsOn = True
8.
9.     def turnOff(self):
10.        # turn the switch off
11.        self.switchIsOn = False
12.
13.
14.    def show(self): # added for testing #
15.        print(self.switchIsOn)
16.
17. # Main code
18. oLightSwitch = LightSwitch()
19.
20. # Calls to methods
21. oLightSwitch.show()
22. oLightSwitch.turnOn()
23. oLightSwitch.show()
```

```
24.oLightSwitch.turnOff()  
25.oLightSwitch.show()  
26.oLightSwitch.turnOn()  
27.oLightSwitch.show()
```

3 Amati hasilnya kemudian tulis analisis singkat mengenai kegiatan ini

### 2.3.2 Kegiatan 2 : Class dengan dua instance

1. Buat sebuah file program baru kemudian tulis kode berikut ini

```
1. class LightSwitch():  
2.     def __init__(self):  
3.         self.switchIsOn = False  
4.  
5.     def turnOn(self):  
6.         # turn the switch on  
7.         self.switchIsOn = True  
8.  
9.     def turnOff(self):  
10.        # turn the switch off  
11.        self.switchIsOn = False  
12.  
13.    def show(self): # added for testing #  
14.        print(self.switchIsOn)  
15.  
16.  
17.# Main code  
18.oLightSwitch1 = LightSwitch()  
19.oLightSwitch2 = LightSwitch()  
20.  
21.# Test code  
22.oLightSwitch1.show()  
23.oLightSwitch2.show()  
24.oLightSwitch1.turnOn() # Turn switch 1 on  
25.# Switch 2 should be off at start, but this makes it cleare  
    r  
26.oLightSwitch2.turnOff()  
27.oLightSwitch1.show()  
28.oLightSwitch2.show()
```

2. Amati hasilnya kemudian tulis analisis singkat mengenai kegiatan ini

### 2.3.3 Kegiatan 3 : Mendalami Class

1. Buat sebuah file program baru kemudian tulis kode berikut ini

```
1. class RemoteTv():
2.
3.     def __init__(self):
4.         self.switchIsOn = False
5.         self.brightness = 0
6.
7.     def turnOn(self):
8.         self.switchIsOn = True
9.
10.    def turnOff(self):
11.        self.switchIsOn = False
12.
13.    def raiseLevel(self):
14.        if self.brightness < 10:
15.            self.brightness = self.brightness + 1
16.
17.    def lowerLevel(self):
18.        if self.brightness > 0:
19.            self.brightness = self.brightness - 1
20.
21.    # Extra method for debugging
22.    def show(self):
23.        print('Switch is on ?', self.switchIsOn)
24.        print('Brightness is:', self.brightness)
25.
26. # Main code
27. remoteSatu = RemoteTv()
28.
29. # Turn switch on, and raise the level 5 times
30. remoteSatu.turnOn()
31. remoteSatu.raiseLevel()
32. remoteSatu.raiseLevel()
33. remoteSatu.raiseLevel()
34. remoteSatu.raiseLevel()
35. remoteSatu.raiseLevel()
36. remoteSatu.show()
37.
```

```
38.# Lower the level 2 times, and turn switch off
39.remoteSatu.lowerLevel()
40.remoteSatu.lowerLevel()
41.remoteSatu.turnOff()
42.remoteSatu.show()
```

2. Amati hasilnya kemudian tulis analisis singkat mengenai kegiatan ini

## 2.4 Tugas

1. Dengan menggunakan source code dari Kegiatan 3, buatlah method **volumeUp** yang berfungsi menaikkan level volume dan **volumeDown** yang berfungsi menurunkan level volume