COMPUTER
PROGRAMING 2

KELAS DAN OBJEK

MODUL 8

2023

DAFTAR ISI

KELAS DAN OBJEK

Sebelumnya, kita perlu tahu dulu bahwa sesuatu tidaklah dikatakan objek kecuali jika memiliki atribut atau perilaku.

Atribut adalah semacam identitas atau variabel dari suatu objek, sedangkan perilaku adalah “kemampuan” atau fitur dari objek tersebut.

Kita juga bisa definisikan dalam bentuk yang lebih sederhana lagi, yaitu: **objek adalah** sebuah gabungan dari kumpulan variabel (dinamakan atribut) dan kumpulan fungsi (dinamakan perilaku) [1]. Dan atas definisi itu, maka bisa dikatakan bahwa semua hal di dalam python adalah sebuah objek [2].

Bahkan Sebuah Fungsi Pun Adalah Objek

Kalau kita teliti lebih jauh lagi, setiap fungsi pada python memiliki atribut __doc__ yang merupakan bukti bahwa fungsi sekalipun adalah sebuah objek dalam python.

Jika kita periksa tipe data dari sebuah fungsi, kita akan dapati bahwa ia ternyata objek dari sebuah class dengan nama function:

```
>>> def tes():
...   print('halo')
...
>>> type(tes)
<class 'function'>
```

A. Kelas Pada Python

Apa itu sebuah kelas?

Kelas atau class pada python bisa kita katakan sebagai sebuah blueprint (cetakan) dari objek (atau instance) yang ingin kita buat [3].

Kelas adalah cetakannya atau definisinya, sedangkan objek (atau instance) adalah objek nyatanya. Kita coba beri contoh “kucing” untuk memperdekat pemahaman.

- “Kucing” merupakan sebuah definisi objek, ia memiliki 2 telinga, 4 kaki, 1 ekor, nama dan lain-lain (sebagai atribut), ia juga bisa berlari, mengeong, makan dan minum (sebagai perilaku).
- Misal kita memiliki 4 kucing: berarti kita memiliki “4 instance” dari kelas kucing.
- Masing-masing dari 4 kucing tersebut memiliki atribut yang telah didefinisikan sebelumnya seperti kaki, telinga, ekor, dan lain-lain. Ia juga

memiliki kemampuan berlari, mengeong, dan sebagainya seperti yang telah didefinisikan sebelumnya.

B. Atribut dan Perilaku

Atribut pada OOP merepresentasikan variabel yang dimiliki sebuah objek. Sedangkan perilaku pada OOP merepresentasikan fungsi yang dimiliki sebuah objek.

Contoh Atribut

Sebagai contoh, kita memiliki 3 ekor kucing, masing-masing kucing memiliki warna dan usianya sendiri-sendiri.

Jika kita ingin merepresentasikan 3 ekor kucing tersebut dengan pendekatan OOP, kita bisa menuliskan kode programnya seperti berikut:

```
# kelas kucing sebagai "definisi"
class Kucing:
  warna = None
  usia = None

# membangun instance/variabel sebagai "objek nyata"
kucing1 = Kucing()
kucing1.warna = "hitam"
kucing1.usia = "3 bulan"

kucing2 = Kucing()
kucing2.warna = "putih"
kucing2.usia = "2 bulan"

kucing3 = Kucing()
kucing3.warna = "kuning"
kucing3.usia = "3.5 bulan"
Kita bisa menampilkan masing-masing atribut dari tiap
instance:

print(kucing1.warna, kucing1.usia)
print(kucing2.warna, kucing2.usia)
print(kucing3.warna, kucing3.usia)
```

Output:

```
hitam 3 bulan
putih 2 bulan
kuning 3.5 bulan
```

Contoh Perilaku
Misalkan kita memiliki kelas Mahasiswa. Setiap mahasiswa memiliki atribut:

- Nama
- dan Asal

Serta memiliki perilaku:

- memperkenalkan diri

Maka bisa kita definisikan kode programnya dengan pendekatan OOP sebagai berikut:

```
# buat kelasnya terlebih dahulu
class Mahasiswa:
  nama = None
  asal = None

  def perkenalan (self):
    print(f'Perkenalkan saya {self.nama} dari {self.asal}')
```

Setelah membuat kelas, kita bisa membuat instance sesuka kita.

Dalam contoh berikut ini, saya akan membuat 2 buah mahasiswa: yang pertama mahasiswa dengan nama "Deni" dari Jawa Timur, dan yang kedua adalah "Lendis Fabri" yang juga dari Jawa Timur, kemudian saya akan minta keduanya untuk memperkenalkan diri.

Berikut ini kode programnya:

```
deni = Mahasiswa()
deni.nama = "Deni"
deni.asal = "Jawa Timur"
lendis = Mahasiswa()
lendis.nama = "Lendis Fabri"
lendis.asal = "Jawa Timur"

# panggil fungsi perkenalan
deni.perkenalan()
lendis.perkenalan()
```

Output:

```
Perkenalkan saya Deni dari Jawa Timur
Perkenalkan saya Lendis Fabri dari Jawa
```

Kalau kita perhatikan lagi kode berikut:

```
def perkenalan (self):
  print(f'Perkenalkan saya {self.nama} dari {self.asal}')
```

Kita dapati ada sebuah parameter pada fungsi perkenalan, yaitu parameter self.

Tetapi ketika kita memanggil fungsi tersebut dari instance yang kita buat, kita tidak melemparkan parameter apa pun:

```
# panggil fungsi perkenalan
deni.perkenalan()
lendis.perkenalan()
```

Ini lah bedanya fungsi yang ada di luar kelas dan fungsi yang ada di dalam kelas.

Jadi ketika kita memanggil sebuah fungsi yang berada pada kelas tertentu, interpreter python akan otomatis melemparkan self sebagai parameter pertama. Dan dari paramter self ini kita bisa mendapatkan akses secara internal terhadap atribut dan perilaku dari objek yang kita buat.

Sebuah Instance dari Suatu Kelas
Kalau kita perhatikan lagi potongan kode program berikut:

```
deni = Mahasiswa()
lendis = Mahasiswa()
```

Pada 2 baris kode di atas, kita telah membuat 2 buah instance dari kelas Mahasiswa.

Dari segi penulisan, keduanya sama saja seperti cara penulisan variabel.

Karena sebenarnya instance itu ya variabel juga. Cuman bedanya, pada contoh di sini kita membuat “tipe data kita sendiri” dari pada menggunakan tipe data asli bawaan python.

C. Konstruktor

Apa itu konstruktor?

Konstruktor adalah sebuah fungsi yang akan dipanggil pertama kali saat sebuah objek di-instantiasi-kan.

Fungsi tersebut harus selalu bernama __init__().

Agar lebih paham, mari kita praktikkan langsung untuk mendefinisikan fungsi konstruktor pada kelas Mahasiswa.

```
class Mahasiswa:
  def __init__(self, nama, asal):
    self.nama = nama
    self.asal = asal

  def perkenalan (self):
    print(f'Perkenalkan saya {self.nama} dari {self.asal}')
```

Ketika membuat instance dari kelas Mahasiswa, kita harus melemparkan dua buah parameter wajib yaitu parameter nama dan parameter asal, seperti ini:

```
deni = Mahasiswa('Deni', 'Sulawesi')
lendis = Mahasiswa(asal = 'Sumatera', nama = 'Lendis Fabri')

deni.perkenalan()
lendis.perkenalan()
```

D. Dokumentasi Kelas

Selain atribut dan perilaku, kelas juga memiliki “deskripsi”. Deskripsi ini didefinisikan dengan menuliskan komentar multi baris yang diletakkan langsung setelah pendefinisian fungsi.

Contohnya seperti ini:

```
class Mahasiswa:
  """
  Kelas ini digunakan untuk mendefinisikan
  objek Mahasiswa di kehidupan nyata
  """
```

Kita bisa menampilkannya dengan mengakses atribut bawaan yang bernama __doc__ seperti berikut:

```
print(Mahasiswa.__doc__)
```

Output:

```
Kelas ini digunakan untuk mendefinisikan
  objek Mahasiswa di kehidupan nyata
```

E. Atribut Kelas yang Merupakan Instance dari Kelas Lainnya

Terakhir dan bukan yang paling akhir, kita juga bisa mendefinisikan sebuah atribut objek yang merupakan instance dari objek yang lainnya.

Misal: kita jadikan kelas Mahasiswa memiliki satu ekor kucing.

Kode programnya bisa seperti ini:

```
class Kucing:
  def __init__(self, warna, usia):
    self.warna = warna
    self.usia = usia

class Mahasiswa:
  def __init__(self, nama, asal, kucing):
    self.nama = nama
    self.asal = asal
    self.kucing = kucing

  def perkenalan(self):
    print(f'Perkenalkan saya {self.nama} dari {self.asal}')
    print(f'Saya memiliki kucing berwarna
{self.kucing.warna} usia {self.kucing.usia}')
```

Cara memanggilnya:

```
deni = Mahasiswa(
    nama='Deni',
    asal='Sidoarjo',
    kucing=Kucing(
        warna='Merah',
        usia='3 bulan'
    )
)
deni.perkenalan()
```

Output:

```
Perkenalkan saya Deni dari Sidoarjo
Saya memiliki kucing berwarna Merah usia 3 bulan
```

F. Kesimpulan

Dari pertemuan kali ini, ada beberapa poin yang bisa kita tarik sebagai kesimpulan:

1. Objek pada python adalah kumpulan dari variabel-variabel (dinamakan atribut) dan kumpulan dari fungsi-fungsi (dinamakan perilaku).
2. Atas definisi itu, maka semua hal di dalam python adalah sebuah Objek.
3. Objek dan Kelas dalam python bermakna sama. Akan tetapi, jika disebutkan dalam konteks terpisah, maka kelas adalah blueprint dan objek adalah varibel nyata.
4. Konstruktor adalah fungsi yang pertama kali dipanggil ketika sebuah objek diinstantiasi.
5. Objek bisa memiliki atribut yang berupa instan dari kelas lainnya.

=== To be Continue ===

REFERENSI

[1] Admin, "Python Objects and Classes," 28 November 2023. [Online]. Available: https://www.programiz.com/python-programming/class.

[2] Admin, "Dive Into Python," [Online]. Available: https://linux.die.net/diveintopython/html/getting_to_know_python/everything_is_an_object.html#d0e4665. [Accessed 28 November 2023].

[3] N. Huda, "Jago Ngoding," 13 March 2021. [Online]. Available: https://jagongoding.com/python/menengah/oop/kelas-dan-objek/. [Accessed 28 November 2023].