Pengenalan Pemrograman Berorientasi Objek (OOP) dengan Python - Class, Objek, dan Method

Kelas atau dalam bahasa Inggris disebut class, merupakan sebuah konsep yang menyediakan sarana untuk menyatukan data dan fungsionalitas secara satu kesatuan. Membuat sebuah kelas artinya membuat sebuah tipe baru, kemudian dengan membuat instance dari kelas tersebut akan menghasilkan objek baru dari tipe tersebut. Setiap objek (hasil instance dari kelas tersebut) dapat memiliki atribut untuk mengelola status dari objek tersebut, juga dapat memiliki metode untuk mengubah status atau informasinya.

> Catatan:
>
> - Kata objek adalah terjemahan bahasa Inggris dari kata object.
> - Kata metode adalah terjemahan bahasa Inggris dari kata method.

Selanjutnya kita akan mempelajari secara mendalam implementasi kelas dan fitur-fitur terkait di bahasa pemrograman Python.

Class

Class merupakan sintaksis di Python yang menyediakan semua fitur-fitur standar dari Pemrograman Berorientasi Objek atau dalam bahasa Inggris disebut dengan Object Oriented Programming (OOP).

Definisi dari kelas menggunakan sintaksis class seperti halya definisi fungsi yang menggunakan sintaksis def, kemudian perlu dipanggil (dieksekusi) dahulu sebelum dapat digunakan dan memiliki efek pada program.

```
1. class NamaKelas:
2.     pass  # gantikan dengan pernyataan-pernyataan, misal: atribut atau metode
```

Pada pemanggilan sintaksis class tersebut, setelah seluruh pernyataan-pernyataan semuanya selesai diproses (didaftarkan sebagai atribut ataupun metode), maka kelas sudah dibuat dan dapat digunakan.

Sebuah kelas sendiri mendukung dua macam operasi:

1. Mengacu pada atribut.
2. Pembuatan instance atau dalam bahasa Inggris disebut instantiation.

Agar lebih jelas, kita akan membahas menggunakan contoh berikut.

```
    """contoh kelas kalkulator sederhana"""
    i = 12345

    def f(self):
        return 'hello world'
```

Dari pembuatan class Kalkulator di atas, di dalamnya ada definisi atribut *i* dan definisi fungsi *f*.

Proses mengacu atribut yaitu *Kalkulator.i* dan *Kalkulator.f* sesuai definisi akan mengembalikan nilai integer dan fungsi. Pada proses mengacu atribut tersebut juga dapat mengubah nilainya, misalnya dengan memberikan bilangan bulat lain ke *Kalkulator.i* akan mengubah nilai yang ada saat ini.

```
Kalkulator.i = 1024  # maka nilai atribut i dalam Kalkulator berubah dari 12345 menjadi 1024
```

Objek (object: an instance of a class)

Pembahasan berikutnya adalah instantiation dari sebuah class, menggunakan notasi fungsi yaitu dengan kurung buka-kurung tutup, akan menghasilkan sebuah objek. Kemudian hasil instantiation ini biasanya disimpan dalam sebuah variabel dengan nama yang representatif.

Berikut ini adalah contoh membuat instance dari *class Kalkulator* menghasilkan sebuah objek.

```
k = Kalkulator()  # membuat instance dari kelas jadi objek, kemudian disimpan pada variabel k
```

Sebagai hasil instance sebuah class, suatu objek memiliki atribut dan metode yang didapatkan dari class. Sebuah metode atau dalam bahasa Inggris disebut method, adalah sebuah fungsi khusus yang menjadi "milik" suatu objek.

Untuk memanggil metode `f` dari objek `k`, hasil *instance* dari *class Kalkulator* di atas sebagai berikut.

```
print(k.f())  # akan mencetak hello world ke layar
```

Kenapa metode adalah sebuah fungsi khusus?
Jika diperhatikan kembali fungsi `f` dalam definisi *class Kalkulator* memiliki satu argumen bernama *self*, sedangkan dalam pemanggilan metode dari objek `k` di atas tidak menggunakan argumen. Apabila `f` adalah fungsi biasa pada Python tentu pemanggilan ini akan mengembalikan kesalahan (error). Lebih detail mengenai konvensi ini akan dibahas pada bagian metode dari *class*.

Pembahasan lebih lanjut mengenai metode dari class ada di bagian selanjutnya.

Class' Constructor

Kembali membahas proses instantiation dari class, sering ditemui kebutuhan mengeset nilai awal atau kondisi awal dari atribut yang dimiliki oleh class tersebut, sehingga untuk kebutuhan ini digunakan sebuah fungsi khusus yang biasa disebut sebagai pembangun atau dalam bahasa Inggris disebut *constructor*. Di Python, fungsi khusus atau metode sebagai constructor ini bernama __init__ atau biasa diucapkan sebagai "double underscore init". Pada saat dilakukan instantiation dari class, metode __init__ ini secara otomatis akan dipanggil terlebih dahulu.

Berikut adalah definisi *class Kalkulator* di atas jika diubah dengan menggunakan constructor.

```
1.  class Kalkulator:
2.      """contoh kelas kalkulator sederhana"""
3.
4.      def __init__(self):
5.          self.i = 12345
6.
7.      def f(self):
8.          return 'hello world'
```

Nilai dari atribut `i` tidak terdefinisi pada awal definisi *Kalkulator*, setelah dilakukan instantiation maka nilai atribut `i` akan bernilai 12345. Meskipun bisa mendefinisikan variabel `i` sebagai atribut dari *class Kalkulator*, tetapi sebaiknya berhati-hati mengenai variabel yang akan terbagi (shared) untuk semua *instance* dari *class*, terutama untuk tipe yang dapat berubah (mutable), misalnya *list* dan *dictionary*.
referensi: https://docs.python.org/id/3.8/tutorial/classes.html#class-and-instance-variables

```
1.  class KeranjangBelanja:
2.      """contoh tidak baik dilakukan dengan definisi variabel terbagi"""
3.      isi = []  # menggunakan list di sini akan terbagi untuk semua instance. JANGAN DILAKUKAN
```

Lanjut pembahasan *constructor*, dengan dilengkapi *constructor* pun proses *instantiation* tidak berubah dari sebelumnya.

```
1.  k = Kalkulator()  # membuat instance dari kelas jadi objek, kemudian disimpan pada variabel k
```

Lebih lanjut tentang constructor, tentu saja untuk mendukung aplikasi yang lebih dinamis
maka constructor dapat memiliki parameter yang bisa dikirimkan saat proses instantiation, bahkan parameternya bisa lebih dari satu jika diperlukan.

Pada contoh berikut ini, constructor memiliki parameter `i` yang bersifat opsional, apabila dalam proses instantiation tidak dikirimkan parameter, secara otomatis i akan diisi nilai bawaan 12345.

```
    """contoh kelas kalkulator sederhana"""

    def __init__(self, i=12345):
        self.i = i  # i adalah variabel pada constructor, self.i adalah variabel dari class

    def f(self):
        return 'hello world'
```

Dengan contoh pemanggilan berikut.

```
k = Kalkulator(i=1024)  # melakukan instantiation sekaligus mengisi atribut i jadi 1024
print(k.i)              # mencetak atribut i dari objek k dengan keluaran nilai 1024
```

Metode (Method)

Pembahasan lebih detail mengenai metode, selain yang dibahas sebelumnya, kita akan membahas 3 jenis metode:

1. Metode dari objek (object method)
2. Metode dari class (class method)
3. Metode secara static (static method)

Pertama kita membahas metode dari objek, seperti yang sempat dijelaskan secara singkat di atas mengenai metode, atau dalam bahasa Inggris disebut method, secara umum metode adalah sebuah fungsi khusus yang menjadi “milik” suatu objek, yakni hasil instantiation dari *class*.

Salah satu hal khusus yang dimiliki oleh metode dengan adanya argumen bernama self, Anda tentu bertanya-tanya tentang argumen self pada metode-metode dalam kelas tersebut sebetulnya apa?
Argumen pertama dari metode-metode dalam class, biasa diberikan nama self sebagai suatu konvensi atau standar penamaan, meskipun Anda bisa juga menggunakan nama lain. Bahkan dalam Python tidak ada arti khusus tentang sintaksis self ini, namun sangat disarankan menggunakan konversi ini agar program Python yang Anda buat akan lebih mudah dimengerti oleh pemrogram lainnya.

Seperti yang Anda sudah perkirakan, untuk sebuah metode, sebetulnya dikirimkan objek (hasil instance dari class) sebagai argumen pertamanya, dalam hal ini bernama self.

Misalnya menggunakan contoh di atas, jika k adalah objek hasil instance dari *class Kalkulator*, saat melakukan pemanggilan metode f.

```
k.f()
```

```
Kalkulator.f(k)
```

Argumen `self` pada metode `f` akan diisi dengan objek hasil instance dari *class Kalkulator*.

Sebelum kita membahas yang kedua dan ketiga, yakni metode dari class dan metode secara static, Anda tentu mengingat bahwa sebelumnya sudah belajar fungsi-fungsi bawaan (built-in) dari Python, antara lain: `open`, `sorted`, `int`, `str`, dan sejumlah lainnya. Terkait metode, ada dua fungsi bawaan yang akan kita bahas, yakni classmethod dan staticmethod.

> Catatan:
>
> fungsi decorator adalah sebuah fungsi yang mengembalikan fungsi lain, biasanya digunakan sebagai fungsi transformasi dengan "pembungkus" sintaksis @wrapper.
>
> Referensi: https://docs.python.org/id/3.8/glossary.html#term-decorator.

Classmethod adalah sebuah fungsi yang mengubah metode menjadi metode dari class (class method). Dalam penggunaannya, fungsi ini dijadikan sebagai fungsi decorator *@classmethod*, kemudian pemanggilannya bisa langsung dari class yang terdefinisi ataupun melalui objek.
Metode dari class (*class method*) menerima masukan class secara implisit sebagai argumen pertama yang secara konvensi diberikan nama cls.

Berdasar contoh yang sama dengan class sebelumnya, berikut adalah metode dari class.

```
class Kalkulator:
    """contoh kelas kalkulator sederhana"""

    def f(self):
        return 'hello world'

    @classmethod
    def tambah_angka(cls, angka1, angka2):
        return '{} + {} = {}'.format(angka1, angka2, angka1 + angka2)
```

Nampak pada kode, sesuai konvensi ada metode yang menggunakan argumen pertama self, sedangkan untuk class method menggunakan konvensi argumen pertama *cls*.

Untuk melakukan pemanggilan dari class, dilakukan seperti berikut, dimana argumen pertama cls sudah mendapatkan masukan *class Kalkulator*.

```
Kalkulator.tambah_angka(1, 2)  # tanpa perlu memberikan masukan untuk argumen cls
```

```
k = Kalkulator()
print(k.tambah_angka(1, 2))
```

Staticmethod adalah sebuah fungsi yang mengubah metode menjadi metode statis (static method). Dalam penggunaannya, fungsi ini dijadikan sebagai fungsi *decorator* *@staticmethod*, kemudian pemanggilannya bisa langsung dari *class* yang terdefinisi ataupun melalui objek.
Metode statis (static method) tidak menerima masukan argumen pertama secara implisit.

Untuk Anda yang pernah memrogram Java atau C++, metode statis ini mirip seperti yang ada di bahasa pemrograman tersebut.

Berdasar contoh yang sama dengan *class* sebelumnya, berikut adalah metode statis.

```
class Kalkulator:
    """contoh kelas kalkulator sederhana"""

    def f(self):
        return 'hello world'

    @staticmethod
    def kali_angka(angka1, angka2):
        return '{} x {} = {}'.format(angka1, angka2, angka1 * angka2)
```

Nampak pada kode, tidak ada argumen pertama yang implisit seperti halnya pada dua metode sebelumnya.

Pemanggilan dari *class* seperti halnya pemanggilan fungsi biasa.

```
a = Kalkulator.kali_angka(2, 3)
print(a)
```

Metode statis (*static method*) juga dapat dipanggil dari objek, hasil instantiation dari *class Kalkulator*, mirip seperti pemanggilan fungsi biasa meskipun dipanggil dari objek.

```
k = Kalkulator()
a = k.kali_angka(2, 3)
print(a)
```

Sebelumnya

Selanjutnya

Dicoding Space
Jl. Batik Kumeli No.50, Sukaluyu,
Kec. Cibeunying Kaler, Kota Bandung
Jawa Barat 40123

Penghargaan

image
click bila
belum muncul

Decode Ideas
Discover Potential

Tentang Kami

Blog
Reward
Showcase

Hubungi Kami
FAQ