COMPUTER
PROGRAMING 2

OVERRIDING

MODUL 11

2023

DAFTAR ISI

OVERRIDING

Pada pertemuan sebelumnya kita telah mempelajari tentang teknik inheritance atau pewarisan dalam python [1].

Dan pada pertemuan kali ini, kita akan membahas lebih dalam tentang pewarisan: yaitu penimpaan atau dalam bahasa inggris sering disebut dengan istilah overriding.

A. Memahami Konsep Pewarisan Objek

Apa itu overriding dalam konsep pemrograman berorientasi objek?

Di dalam semua bahasa pemrograman yang berbasis objek, teknik **overriding adalah** fitur yang memungkinkan kita untuk mengimplementasikan "ulang" fungsi/method pada sebuah child class atau kelas turunan yang sebenarnya fungsi tersebut telah didefinisikan di dalam parent class atau kelas induk [2].

Overriding sendiri memberikan keuntungan kepada kita karena kita bisa menduplikat kelas lain –sehingga kode program lebih singkat, dan kita juga bisa "membedakan" fungsi-fungsi tertentu pada kelas turunan yang sudah didefinisikan pada kelas Induk.

Hal ini memberikan kita keleluasaan untuk mendefinisikan berbagai kelas sesuai fungsinya masing-masing tanpa harus menulis kode yang sama berkali-kali. Oleh karena itu, overriding masih dikategorian sebuah bagian dari pewarisan [3].

B. Cara Melakukan Overriding Pada Python

Untuk melakukan overriding pada python caranya sederhana:

- Kita hanya perlu melakukan pewarisan.
- Kemudian menulis ulang fungsi yang sudah ada pada kelas Induk.

Contoh Overriding
Misalkan kita punya dua buah kelas yaitu Kendaraan dan Mobil di mana:

- Kelas Kendaraan adalah kelas induk
- dan Kelas Mobil adalah kelas turunan dari kelas Kendaraan

Dan:

- Kelas Kendaraan punya kemampuan (fungsi) berjalan()
- Akan tetapi kita ingin bahwa kelas Mobil punya perilaku khusus untuk fungsi berjalan().

Agar lebih jelas, mari kita praktikkan secara langsung. Sebagai pondasi awal, buat dua buah kelas seperti berikut:

```
class Kendaraan:
  def berjalan(self):
    print('berjalan..')

class Mobil(Kendaraan):
  pass
```

Kemudian buat instan dan panggil fungsi berjalan():

```
sepeda = Kendaraan()
sedan = Mobil()

sepeda.berjalan()
sedan.berjalan()
```

Output:

```
berjalan..
berjalan..
```

Jika kita perhatikan lagi output di atas: maka output-nya identik, tidak ada yang berbeda satu huruf pun. Itu karena kelas Mobil menurunkan semua hal dari kelas Kendaraan tanpa mengubah satu hal apa pun.

C. Melakukan Overriding

Nah, langkah berikutnya adalah kita akan melakukan overriding fungsi berjalan(), agar aksinya menjadi berbeda ketika dipanggil dari kelas turunan yaitu kelas Mobil.

Ubah kode programnya menjadi seperti ini:

```
class Kendaraan:
  def berjalan(self):
    print('berjalan..')

class Mobil(Kendaraan):
  def berjalan(self):
    print('Berjalan dengan cepat..')
```

Jalankan kembali programnya, maka kita akan mendapatkan output seperti ini:

```
berjalan..
Berjalan dengan cepat..
```

D. Menambahkan Parameter Pada Fungsi yang Ditimpa

Kita juga bisa memberikan parameter pada fungsi yang ingin kita timpa. Sebagai contoh, kita akan menambahkan 2 buah parameter untuk fungsi berjalan() pada kelas Mobil.

Dua buah parameter tersebut adalah:

- kecepatan
- satuan kecepatan

Berikut ini kode programnya:

```
class Kendaraan:
  def berjalan(self):
    print('berjalan..')

class Mobil(Kendaraan):
  def berjalan(self, kecepatan, satuan = 'km/j'):
    print(f'Berjalan dengan kecepatan {kecepatan} {satuan}')
```

Jangan lupa ubah juga pemanggilan fungsi berjalan() dengan menambahkan parameter yang sesuai:

```
sepeda = Kendaraan()
sedan = Mobil()

sepeda.berjalan()
sedan.berjalan(150)
```

Output:

```
berjalan..
Berjalan dengan kecepatan 150 km/j
```

E. Memanggil Fungsi Pada Kelas Induk

Sama seperti yang telah kita pelajari pada pertemuan sebelumnya tentang fungsi super dalam konstruktor (masih dalam bab pewarisan), kita memanggil fungsi konstruktor yang terdapat pada kelas induk dari kelas turunan dengan sintaks berikut:

```
class KelasTurunan(KelasInduk):
  def __init__(self):
    super().__init__()
```

Di mana sebenarnya fungsi super() tidak khusus hanya untuk konstruktor saja, tapi kita juga bisa menggunakannya untuk memanggil fungsi selain konstruktor.

Perhatikan contoh berikut:

```
class Kendaraan:
  def berjalan(self):
    print('berjalan..')

class Mobil(Kendaraan):
  def berjalan(self, kecepatan, satuan = 'km/j'):
    super().berjalan()
    print(f'  -> dengan kecepatan {kecepatan} {satuan}')
```

Pada kode di atas, kita telah mendefinisikan ulang fungsi berjalan() pada kelas Mobil. Tidak hanya itu, kita juga tetap memanggil fungsi yang sama yang terdapat pada kelas induknya yaitu kelas Kendaraan.

Sehingga jika kita jalankan program di atas, kita akan mendapatkan output seperti ini:

```
berjalan..
berjalan..
  -> dengan kecepatan 150 km/j
```

F. Kesimpulan

Dari berbagai percobaan yang kita lakukan pada pertemuan ini, kita bisa simpulkan bahwa:

1. Teknik overriding merupakan bagian dari teknik pewarisan.
2. Teknik overriding membuat kita bisa memodifikasi fungsi yang sudah didefinisikan pada kelas Induk
3. Kita juga bisa mempertahankan fungsi asli yang terdapat pada kelas induk dengan memanggil fungsi super()
4. Kode Program Lengkap
5. Untuk kode program lengkap pada pertemuan kali ini, kalian bisa mendapatkannya di sini.

Pertemuan Selanjutnya

Pada pertemuan selanjutnya insyaallah kita akan membahas tentang access modifier pada python. Ia adalah teknik di mana kita bisa mengatur apakah suatu properti bisa diakses secara publik, dilindungi (protected), atau bahkan diset privat.

Bagaimana caranya?

Simak terus tutorial ini, ya! Terima kasih banyak.

=== To be Continue ===

REFERENSI

[1] N. Huda, "Jago Ngoding," 24 May 2021. [Online]. Available: https://jagongoding.com/python/menengah/oop/overriding/. [Accessed 28 November 2023].

[2] avir0zmf1, "Geeksforgeeks," 13 April 2023. [Online]. Available: https://www.geeksforgeeks.org/perl-method-overriding-in-oops/. [Accessed 28 November 2023].

[3] Raja, "Tutorial Point," 27 November 2023. [Online]. Available: https://www.tutorialspoint.com/articles/index.php. [Accessed 28 November 2023].