Inheritance pada Python – Part 1

Inheritance adalah salah satu mekanisme di konsep OO yang mengizinkan aku untuk mendefinisikan sebuah *class* baru berdasarkan *class* yang sebelumnya telah dideklarasikan.

Melalui konsep *inheritance*, sebuah *class* baru dapat memiliki atribut dan fungsi pada *class* yang sebelumnya telah didefinisikan. Pada konsep *inheritance*, atribut/fungsi yang akan diwariskan hanyalah atribut/fungsi dengan *access modifier public*, atribut/fungsi dengan a*ccess modifier private* tidak akan diturunkan.

Tugas:

Definisikan *class* Karyawan pada *Live Code Editor*:

Code Editor | Run | Submit | 0 Hint

```
class Karyawan:
    nama_perusahaan = 'ABC'
    insentif_lembur = 250000
    def __init__(self, nama, usia, pendapatan):
        self.nama = nama
        self.usia = usia
        self.pendapatan = pendapatan
        self.pendapatan_tambahan = 0
    def lembur(self):
        self.pendapatan_tambahan += self.insentif_lembur
    def tambahan_proyek(self, insentif_proyek):
        self.pendapatan_tambahan += insentif_proyek
    def total_pendapatan(self):
        return self.pendapatan + self.pendapatan_tambahan
```

melakukan *inheritance* (menurunkan seluruh atribut dan fungsi dari *class* Karyawan) ke **class AnalisData**

Code Editor | Run | Submit | 0 Hint

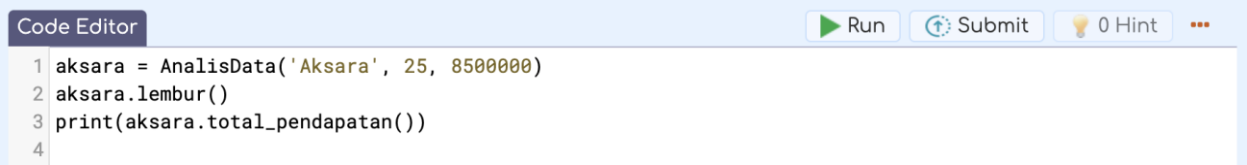
```
class AnalisData(Karyawan):
    def __init__(self, nama, usia, pendapatan):
        super().__init__(nama, usia, pendapatan)
```

melakukan *inheritance* (menurunkan seluruh atribut dan fungsi dari *class* Karyawan) ke **class IlmuwanData**

Code Editor | Run | Submit | 0 Hint

```
class IlmuwanData(Karyawan):
    def __init__(self, nama, usia, pendapatan):
        super().__init__(nama, usia, pendapatan)
```

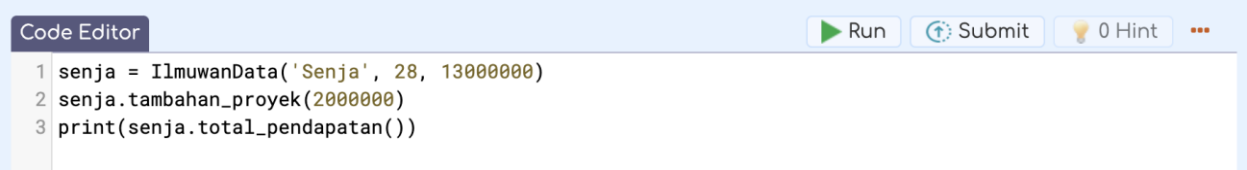
objek AnalisData dapat mengakses fungsi lembur milik *class* Karyawan



akan menghasilkan *output*: **8750000**

Selanjutnya,

objek IlmuwanData dapat mengakses fungsi tambahan_proyek milik *class* Karyawan



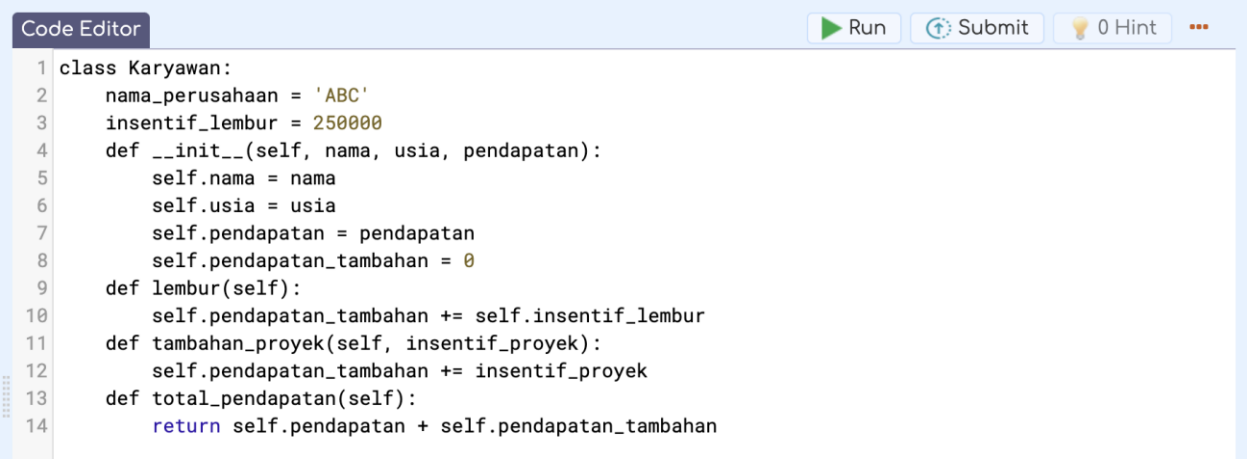
akan menghasilkan *output*: **15000000**

Inheritance pada Python – Part 2

Pada bagian pertama aku telah mempelajari bagaimana *child class* mewarisi fungsi/atribut dari *parent class* dengan menggunakan fungsi super(). Melalui konsep *inheritance*, *child class* dapat memodifikasi atribut/ fungsi yang diwarisi oleh sebuah *parent class* dengan mendefinisikan ulang atribut/ fungsi menggunakan nama yang sama.

Tugas:

Aku mencoba contoh berikut dengan *Live Code Editor*



Code Editor Run Submit 0 Hint

```
class AnalisData(Karyawan):
    def __init__(self, nama, usia, pendapatan):
        super().__init__(nama, usia, pendapatan)
```

Code Editor Run Submit 0 Hint

```
class IlmuwanData(Karyawan):
    def __init__(self, nama, usia, pendapatan):
        super().__init__(nama, usia, pendapatan)
```

Fungsi lembur pada objek aksara sebagai bagian dari *class* AnalisData akan menambahkan total_pendapatan milik objek sebesar 250000 mengikuti insentif_lembur milik *class* Karyawan

Code Editor Run Submit 0 Hint

```
aksara = AnalisData('Aksara', 25, 8500000)
aksara.lembur()
print(aksara.total_pendapatan())
```

akan menghasilkan *output*: **8750000**

Selanjutnya,

fungsi lembur pada objek senja sebagai bagian dari *class* IlmuwanData akan menambahkan total_pendapatan milik objek sebesar 500000 dikarenakan *class* IlmuwanData telah mendefinisikan kembali nilai insentif lembur menjadi 500000

Code Editor Run Submit 0 Hint

```
senja = IlmuwanData('Senja', 28, 13000000)
senja.tambahan_proyek(2000000)
print(senja.total_pendapatan())
```

akan menghasilkan *output*: **13500000**

Penjelasan:

Melalui potongan kode di atas, aku telah menerapkan konsep *inheritance*. Melalui konsep *inheritance* class AnalisData dan IlmuwanData akan memiliki setiap atribut dan fungsi yang dimiliki oleh class Karyawan (Hal ini dikarenakan seluruh atribut dan fungsi dari *class* Karyawan bersifat *public*).

Pada konsep *inheritance*, class AnalisData dan *class* IlmuwanData disebut sebagai *child class* dari *class* Karyawan; sehingga *class* Karyawan dapat disebut sebagai *parent class* dari *class* AnalisData dan IlmuwanData.

Suatu *child class* dapat mengakses atribut ataupun fungsi yang dimiliki oleh *parent class* dengan menggunakan fungsi super(). Pada contoh di atas, fungsi super() digunakan oleh *child class* (AnalisData dan IlmuwanData) untuk mengakses *constructor* yang dimiliki oleh parent class (Karyawan).

Catatan: Sebenarnya, aku tidak perlu mendefinisikan kembali fungsi (termasuk *constructor*) ataupun properti yang memiliki *public access modifier* di sebuah *child class*. Python akan secara otomatis mewariskan seluruh fungsi dan properti dengan *public access modifier* ke sebuah *child class.* Contoh potongan kode di atas hanya diperkenankan untuk mencontohkan penggunaan fungsi super().