**Kopekan UTS Struktur Data & Algoritma Teori**

***By Lathif Ramadhan***

**Kelas & Objek dalam Python**

**Kelas/Objek Python**

* Python adalah bahasa pemrograman berorientasi objek.
* Hampir semua yang ada di Python adalah sebuah objek, dengan properti dan metodenya.
* Kelas seperti konstruktor objek, atau "cetak biru"/ blue print” untuk membuat objek.

**Buat Kelas**

Untuk membuat kelas, gunakan kata kunci ***class***:

**Contoh**

Buat kelas bernama ***MyClass***, dengan properti bernama ***x***:

A screenshot of a computer program

Description automatically generated

**Buat Objek**

Sekarang kita bisa menggunakan kelas bernama ***MyClass*** untuk membuat objek:

**Contoh**

Buat objek bernama p1, dan cetak nilai x:

A screenshot of a computer code

Description automatically generated

**Fungsi \_\_init\_\_()**

* Contoh di atas adalah kelas dan objek dalam bentuk paling sederhana, dan tidak terlalu berguna dalam aplikasi kehidupan nyata.
* Untuk memahami arti kelas kita harus memahami fungsi bawaan ***\_\_init\_\_().***
* Semua kelas memiliki fungsi yang disebut ***\_\_init\_\_(),*** yang selalu dijalankan saat kelas dimulai.
* Gunakan fungsi ***\_\_init\_\_()*** untuk menetapkan nilai pada properti objek, atau operasi lain yang perlu dilakukan saat objek dibuat:

**Contoh**

Buat kelas bernama ***Person***, gunakan fungsi ***\_\_init\_\_()*** untuk menetapkan nilai nama dan usia:

A screenshot of a computer code

Description automatically generated

**Catatan**: Fungsi ini ***\_\_init\_\_()*** dipanggil secara otomatis setiap kali kelas digunakan untuk membuat objek baru.

**Fungsi \_\_str\_\_()**

Fungsi ***\_\_str\_\_()*** mengontrol apa yang harus dikembalikan ketika objek kelas direpresentasikan sebagai string.

Jika fungsi ***\_\_str\_\_()*** tidak disetel, representasi string objek akan dikembalikan:

**Contoh**

Representasi string suatu objek **TANPA** fungsi ***\_\_str\_\_()***:

A screenshot of a computer code

Description automatically generated

**Contoh**

Representasi ***string*** suatu objek **DENGAN** fungsi ***\_\_str\_\_():***

A screenshot of a computer code

Description automatically generated

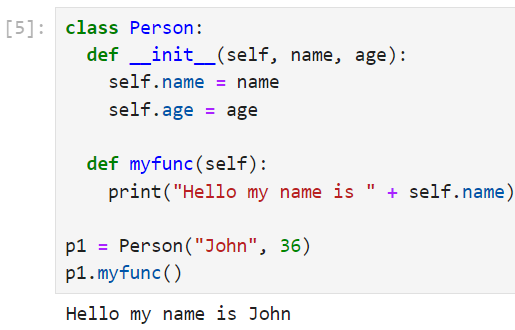
**Object Methods/Metode Objek**

***Objek*** juga bisa berisi metode. Metode dalam objek adalah fungsi yang dimiliki objek tersebut.

Mari kita buat ***metode*** di kelas ***Person***:

**Contoh**

Masukkan fungsi yang mencetak ***Hello***, dan jalankan pada objek ***p1***:



**Catatan:** Parameter ***self*** adalah referensi ke ***instance*** kelas saat ini, dan digunakan untuk mengakses variabel milik kelas tersebut.

**The self Parameter**

Parameter ***self*** adalah referensi ke instance kelas saat ini, dan digunakan untuk mengakses variabel milik kelas tersebut.

Itu tidak harus diberi nama ***self***, Anda bisa menyebutnya apa pun yang Anda suka, tetapi itu harus menjadi parameter pertama dari fungsi apa pun di kelas:

**Contoh**

Gunakan kata ***mysillyobject*** dan *abc* alih-alih ***self*** :

A screen shot of a computer code

Description automatically generated

**Modify Object Properties**

Anda dapat mengubah properti pada objek seperti ini:

**Contoh**

Tetapkan usia ***p1*** menjadi 40:

A screenshot of a computer program

Description automatically generated

**Delete Object Properties**

Anda dapat menghapus properti pada objek dengan menggunakan ***del*** kata kunci:

**Contoh**

Hapus properti ***age*** dari objek ***p1***:

A screenshot of a computer program

Description automatically generated

**Delete Objects**

Anda dapat menghapus objek dengan menggunakan ***del*** kata kunci:

**Contoh**

Hapus objek p1:

A screenshot of a computer program

Description automatically generated

**The pass Statement/Pernyataan lulus**

***Class*** definisi tidak boleh kosong, tetapi jika karena alasan tertentu Anda memiliki ***class*** definisi tanpa konten, masukkan ***pass*** pernyataan untuk menghindari kesalahan.

A screenshot of a computer

Description automatically generated

**Python Inheritance/** **Warisan di Python**

**Warisan Python**

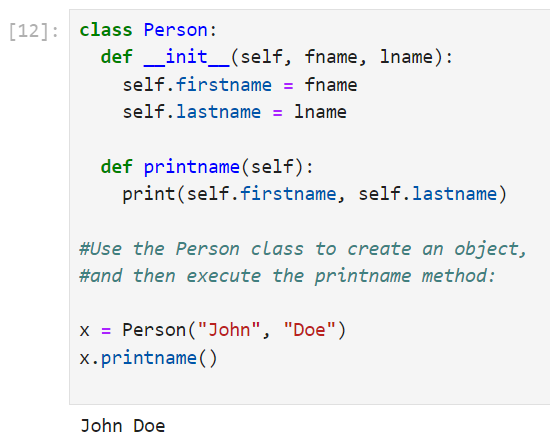
* **Warisan** memungkinkan kita untuk mendefinisikan kelas yang mewarisi semua metode dan properti dari kelas lain.
* **Kelas induk** adalah kelas yang diwarisi, disebut juga kelas dasar.
* **Kelas anak** adalah kelas yang mewarisi dari kelas lain, disebut juga kelas turunan.

**Buat Kelas Induk**

Kelas mana pun bisa menjadi kelas induk, jadi sintaksisnya sama dengan membuat kelas lainnya:

**Contoh**

Buat kelas bernama ***Person***, dengan ***firstname*** dan ***lastname*** properti, dan ***printname*** metode:



**Buat Kelas Anak**

Untuk membuat kelas yang mewarisi fungsionalitas dari kelas lain, kirimkan kelas induk sebagai parameter saat membuat kelas anak:

**Contoh**

Buatlah sebuah kelas bernama ***Student***, yang akan mewarisi properti dan metode dari ***Person*** kelas tersebut:

A close-up of a text

Description automatically generated

**Catatan:** Gunakan pass kata kunci ketika Anda tidak ingin menambahkan properti atau metode lain ke kelas.

Sekarang kelas ***Student*** memiliki properti dan metode yang sama dengan kelas ***Person***.

**Contoh**

Gunakan ***Student*** kelas untuk membuat objek, lalu jalankan metodenya ***printname***:

A screenshot of a computer program

Description automatically generated

**Tambahkan Fungsi *\_\_init\_\_()***

Sejauh ini kita telah membuat kelas anak yang mewarisi properti dan metode dari induknya.

Kami ingin menambahkan ***\_\_init\_\_()*** fungsi ke kelas anak (bukan ***pass*** kata kunci).

**Catatan:** Fungsi ini ***\_\_init\_\_()*** dipanggil secara otomatis setiap kali kelas digunakan untuk membuat objek baru.

**Contoh**

Tambahkan ***\_\_init\_\_()*** fungsi ke ***Student*** kelas:

A close-up of a sign

Description automatically generated

Saat Anda menambahkan ***\_\_init\_\_()*** fungsi tersebut, kelas anak tidak akan lagi mewarisi fungsi induknya ***\_\_init\_\_().***

**Catatan*:\_\_init\_\_()*** Fungsi anak ***mengambil alih*** warisan fungsi induk ***\_\_init\_\_().***

Untuk mempertahankan pewarisan fungsi induk ***\_\_init\_\_()*** , tambahkan panggilan ke fungsi induk ***\_\_init\_\_():***

**Contoh**

A screenshot of a computer program

Description automatically generated

Sekarang kita telah berhasil menambahkan fungsi ***\_\_init\_\_(),*** dan mempertahankan warisan kelas induk, dan kita siap untuk menambahkan fungsionalitas dalam ***\_\_init\_\_()***fungsi tersebut.

**Gunakan Fungsi super()**

Python juga memiliki ***super()*** fungsi yang akan membuat kelas anak mewarisi semua metode dan properti dari induknya:

**Contoh**

A screenshot of a computer program

Description automatically generated

Dengan menggunakan ***super()*** fungsi tersebut, Anda tidak perlu menggunakan nama elemen induknya, secara otomatis akan mewarisi metode dan properti dari induknya.

**Tambahkan Properti**

**Contoh**

Tambahkan properti yang dipanggil ***graduationyear*** ke ***Student*** kelas:

A screenshot of a computer program

Description automatically generated

Pada contoh di bawah, tahun ***2019*** harus berupa variabel, dan diteruskan ke ***Student*** kelas saat membuat objek siswa. Untuk melakukannya, tambahkan parameter lain di fungsi ***\_\_init\_\_():***

**Contoh**

Tambahkan ***year*** parameter, dan masukkan tahun yang benar saat membuat objek:

A screenshot of a computer program

Description automatically generated

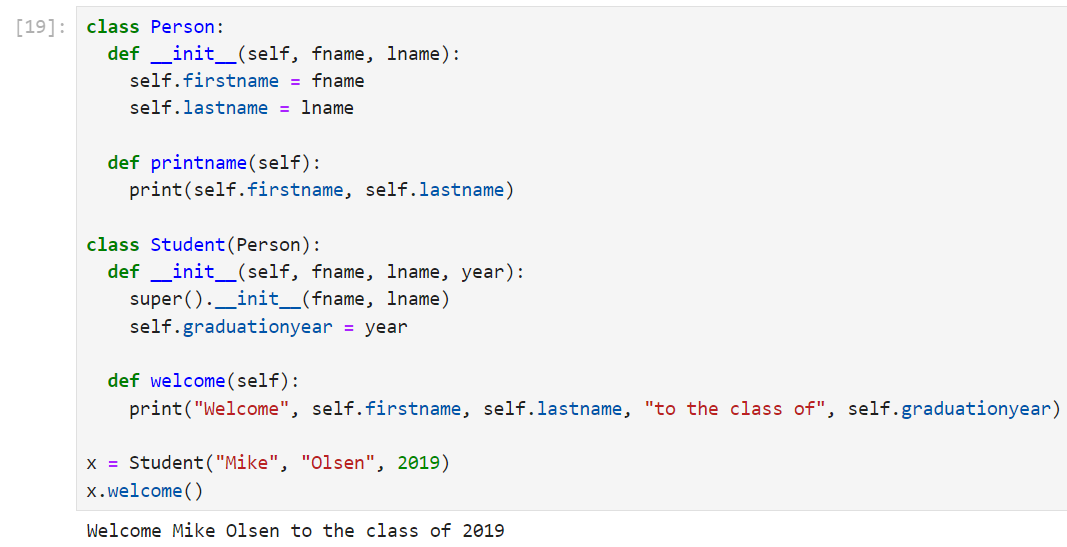
**Tambahkan Metode**

**Contoh**

Tambahkan metode yang dipanggil ***welcome*** ke ***Student*** kelas:

A computer code with red and blue text

Description automatically generated



Jika Anda menambahkan metode di kelas anak dengan nama yang sama dengan fungsi di kelas induk, pewarisan metode induk akan ditimpa.

