Lab 07
Introduction to Class

Dasar-Dasar Pemrograman 1
CSGE601020
Semester Genap 2017/2018

Batas waktu pengumpulan:
Sabtu, 28 April 2018 pukul 12.30 Waktu Scele

Tujuan dari Lab ini adalah melatih Anda agar menguasai bahan kuliah yang diajarkan di kelas. Mahasiswa diperbolehkan untuk berdiskusi, tetapi Anda tetap harus **menuliskan sendiri** solusi/kode program dari soal yang diberikan tanpa bantuan orang lain. Belajarlah menjadi mahasiswa yang mematuhi integritas akademik.
Sikap jujur merupakan sebuah sikap yang dimiliki mahasiswa UI.

Peringatan:
HIndari mengumpulkan pekerjaan beberapa menit menjelang batas waktu pengumpulan karena ada kemungkinan pengumpulan gagal dilakukan atau koneksi Internet terputus!

GPR

Soal Lab 07
Introduction to Class

Class adalah sebuah prototipe yang merupakan dasar pembuatan objek. Prototipe ini mendefinisikan ciri-ciri objek, yang lalu dapat digunakan ke depannya.

1. Mendefinisikan Suatu Class

Kita bisa mendefinisikan sebuah *class* baru dengan menulis seperti berikut:

```
class ClassName:
    // isi dari class
```

Dalam Python, perlu dicatat bahwa nama *class* sebaiknya dalam bentuk **CamelCase**, atau menulis huruf besar untuk setiap awal kata.

2. `__init__()` function

Kita dapat mendefinisikan fungsi __init__() dalam *class*, yang merupakan fungsi paling pertama yang dipanggil ketika membuat suatu objek dari *class* tersebut. Jika tidak didefinisikan, maka fungsi __init__() secara default tidak akan melakukan apapun.

Kita bisa mendefinisikan fungsi __init__() dengan menulis seperti berikut:

```
class ClassName:
    def __init__(self):
        // hal-hal yang ingin dilakukan ketika objek dibuat
```

Kita akan membahas kegunaan dari fungsi __init__() pada bagian **<3. Class Attributes>**.

3. Attribute

Attribute dari suatu class terdiri dari data-data (instance variables) dan methods.

3.1 Data Attributes

Data attribute adalah *attribute* berbentuk variabel yang khusus terhadap objek tersebut.

Perhatikan contoh pembuatan *data attribute* pada *class* Mobil berikut:

```
class Mobil:
    def __init__(self):
        self.merek = 'BMW'
        self.pemilik = 'Paman'
        self.menyala = False
```

Pada contoh di atas, kita membuat 3 variabel yang bernama `merek`, `pemilik` dan `menyala`. Karena kita ingin supaya Mobil tersebut langsung memiliki ketiga ciri-ciri di atas, kita masukkan ke dalam fungsi `__init__()`.

Perhatikan bahwa untuk setiap penulisan variabel kita menulis `self.` terlebih dahulu. Ini menandakan bahwa variabel tersebut merupakan bagian dari objek Mobil, bukan variabel dalam fungsi `__init__()`.

Tentunya kita juga dapat membuat *class* Mobil sedemikian sehingga beberapa variabel ditentukan ketika membuat objek. Perhatikan modifikasi contoh pada *class* Mobil berikut:

```
class Mobil:
    def __init__(self, merek, pemilik):
        self.merek = merek
        self.pemilik = pemilik
        self.menyala = False
```

Pada modifikasi di atas, kita membuat *class* Mobil sedemikian sehingga variabel `merek` dan `pemilik` bebas ditentukan ketika objek dibuat, dan objek Mobil tersebut tidak menyala pada awalnya. Kita akan membahas cara membuat objek dan memanggil *data attribute* pada bagian **<4. Instance Object>**.

3.2 Methods

Method adalah *attribute* yang berbentuk seperti fungsi.

Perhatikan contoh method `klakson()` pada *class* Mobil berikut:

```
class Mobil:
    def klakson(self):
        print('TIN TIN')
```

Keluaran:

```
TIN TIN
```

Pada contoh di atas, kita membuat sebuah fungsi klakson() yang hanya akan *print* suatu tulisan. Perhatikan bahwa setiap *method* harus memiliki suatu parameter `self` sebagai parameter pertama. Parameter ini merupakan *reference* ke objek yang memanggil *method* tersebut.

Perhatikan contoh penggunaan self pada *class* Mobil berikut:

```
class Mobil:
    def __init__(self, merek, pemilik):
        self.merek = merek
```

```
        self.pemilik = pemilik
        self.menyala = False

    def ganti_pemilik(self, pemilik_baru):
        self.pemilik = pemilik_baru
```

Pada contoh di atas, kita membuat sebuah fungsi ganti_pemilik() yang akan mengubah *data attribute* pemilik dari objek sekarang. Kita akan membahas cara memanggil *method* pada bagian **<4. Instance Object>**.

4. Instance Object

5. Contoh *class* Mobil

```
class Mobil:
    def __init__(self, merek, pemilik):
        self.merek = merek
        self.pemilik = pemilik
        self.menyala = False

    def klakson(self):
        print('TIN TIN')

    def ganti_pemilik(self, pemilik_baru):
        self.pemilik = pemilik_baru

    def nyalakan(self):
        self.menyala = True
        print('Mobil menyala, BRUM BRUM')

    def matikan(self):
        self.menyala = False
        print('Mobil mati X(')

    def is_mahal(self):
        if (self.merek == 'BMW'):
            return True
        else:
            return False

mobil = Mobil('BMW', 'Paman')
mobil.ganti_pemilik('Saya sendiri')
mobil.nyalakan()
mobil.klakson()
mobil.matikan()

if (mobil.is_mahal()):
    print('WoOoW, mobil ini mahal!')
else:
    print('Hah, mobil pecundang.')
```

Untuk dokumentasi lebih lengkap dapat dicek di:

- https://www.tutorialspoint.com/python/python_classes_objects.html
- https://docs.python.org/3/tutorial/classes.html

Deskripsi Soal

Fate Pre-Order

Suatu hari Anda sedang belajar DDP1 sendirian di kamar. Tiba-tiba seorang wanita dengan baju zirah bernama Saber mengetuk pintu kamar Anda. Saber menceritakan bahwa dia dari dunia lain dan di dunianya ternyata banyak petarung yang saling bertarung satu sama lain untuk memperebutkan Holy Grail. Petarung-petarung ini lah yang sering disebut sebagai Servant. Karena sekarang Saber sudah berada di dunia Anda dan dia tidak bisa kembali lagi ke dunianya, dia merasa sedih. Sekarang dia sangat memohon kepada kamu, mahasiswa jago ngoding, untuk membuatkan mini-game yang mensimulasikan dunia asalnya tersebut. Mini-game ini disebut sebagai Fate Pre-Order.

Berikut ini deskripsi game Fate Pre-Order yang ingin dibuat oleh Saber:

- **SUMMON [Nama] [Tipe] [Power]**
  Membuat objek Servant yang memiliki atribut nama, tipe, dan power.
  Nama dipastikan unik untuk tiap Servant.
  Tipe yang valid hanyalah **Saber, Lancer, Archer, Caster, Assassin, Rider,** dan **Berserker**.
  Power harus selalu bernilai positif (> 0).
  Di awal pembuatan objek, **level selalu mulai dari 1** dan **keadaan Servant sehat**.

  Jika berhasil membuat objek maka outputnya adalah sebagai berikut:
  "Berhasil memanggil Servant dengan nama **[Nama]** dan tipe **[Tipe]**"

  Jika tidak berhasil membuat objek, seperti **nama sudah ada**, **tipe tidak valid**, atau **power tidak bernilai positif** maka outputnya sebagai berikut:
  "Tidak berhasil memanggil Servant karena kesalahan"

- **TRAIN [Nama]**
  Servant yang namanya dimasukkan akan berlatih dan meningkatkan power sebesar 10%.

  Outputnya sebagai berikut:
  "**[Nama]** berlatih keras dan kekuatannya meningkat menjadi **[Power]**"

- **BATTLE [Nama 1] [Nama 2]**
  Melakukan pertarungan antara Servant **[Nama 1]** dan **[Nama 2]** dengan **membandingkan kekuatan bertarungnya**.
  Syarat kekuatan bertarung tiap Servant adalah sebagai berikut:

- Kekuatan bertarung Saber, Lancer, dan Archer = 0.25 * power + 10 * level
- Kekuatan bertarung Rider, Caster, dan Assassin = 0.15 * power + 20 * level
- Kekuatan bertarung Berserker = 0.5 * power + 5 * level

Setelah mengetahui kekuatan bertarung antara kedua Servant tersebut, lalu **kekuatan mereka dibandingkan**. Servant yang **menang** adalah yang **kekuatannya lebih besar** dan levelnya **bertambah 1**, sedangkan Servant yang **kalah** akan **menjadi terluka**. Jika keadaannya **seri** maka **keduanya terluka** dan **tidak ada yang naik level**. Servant yang terluka **tidak bisa melakukan pertarungan**.

Jika salah satu atau kedua Servant tersebut **sedang terluka** maka outputnya:
"Servant yang akan bertarung sedang terluka"

Jika salah satu Servant **menang dalam pertarungan** maka outputnya:
"**[Nama Pemenang]** menang dalam pertarungan dan **[Nama yang kalah]** menjadi terluka. Selisih kekuatan mereka adalah **[Selisih kekuatan bertarung kedua Servant]**."
Nb : selisih kekuatan keluarkan saja 2 angka dibelakang koma

Jika keadaannya **seri** maka outputnya:
"**[Nama 1]** dan **[Nama 2]** dalam pertarungan sengit yang berakhir seri. Keduanya menjadi terluka."

- **HEAL [Nama]**
  Memulihkan keadaan Servant yang terluka menjadi sehat kembali.

  Jika keadaan Servant masih sehat maka outputnya sebagai berikut:
  "Servant **[Nama]** keadaannya masih sehat."

  Jika keadaan Servant terluka maka outputnya sebagai berikut:
  "Servant **[Nama]** berhasil dipulihkan dan kembali sehat."

- **STATS [Nama]**
  Mencetak informasi tentang Servant **[Nama]** dengan format sebagai berikut:
  Nama : **[Nama]**
  Tipe : **[Tipe]**
  Power : **[Power]**
  Level : **[Level]**
  Status : **[Sehat/Terluka]**

Sistem Validasi

Saat Mini-Game dijalankan, buat juga sistem validasi:

- Jika perintah tidak valid atau tidak ada di antara 5 perintah di atas outputnya adalah: "Perintah tidak valid"
- Validasi saat perintah Summon seperti yang dijelaskan di atas.
- Untuk perintah selain Summon, jika nama Servant tidak ada maka keluarkan output: "Nama servant tidak ada"

Hint:

- Karena pemanggilan setiap perintah hanya menggunakan nama Servant saja dan setiap namanya unik, maka gunakan dictionary untuk menghubungkan nama tersebut dengan objek Servant nya (Key nya adalah nama servant, valuenya adalah objeknya)
- Anda bisa menggunakan template yang sudah disediakan. (Tidak wajib digunakan dan boleh dimodifikasi)

Bonus

Melihat ada kesamaan di antara 7 tipe Servant di atas? Ya, mereka memiliki atribut yang sama dan hanya berbeda dalam kekuatan bertarung saja. Anda bisa membuat 3 sub-class untuk masing-masing jenis kekuatan bertarung. Untuk Saber, Lancer, dan Archer bisa dikategorikan dalam sub-class Knight karena kekuatan bertarung mereka sama. Untuk Caster, Assassin, dan Rider bisa dikategorikan dalam sub-class Cavalry. Untuk Berserker dikategorikan dalam sub-clas Berserker. Di tiap sub-class, method yang berbeda hanyalah kekuatan bertarung saja

Contoh Input & Output (Input = Hitam, Output = Merah)

MAIN Sama Aku Yuk
Perintah tidak valid
DDP1 ITU EZ KOK
Perintah tidak valid
SUMMON Medusa Rider 200
Berhasil memanggil Servant dengan nama Medusa dan tipe Rider
SUMMON Glory Caster -999
Tidak bisa memanggil Servant karena kesalahan
SUMMON Dipsi Assassin 250
Berhasil memanggil Servant dengan nama Dipsi dan tipe Assassin
SUMMON Artoria Saber 500
Berhasil memanggil Servant dengan nama Artoria dan tipe Saber
SUMMON Gudako Lancer 150
Berhasil memanggil Servant dengan nama Gudako dan tipe Lancer
SUMMON Ilya Shirou 400
Tidak bisa memanggil Servant karena kesalahan
SUMMON Heracles Berserker 100
Berhasil memanggil Servant dengan nama Heracles dan tipe Berserker
SUMMON Lancelot Berserker 100

Berhasil memanggil Servant dengan nama Lancelot dan tipe Berserker
SUMMON Emiya Archer 300
Berhasil memanggil Servant dengan nama Emiya dan tipe Archer
SUMMON Gudao Caster 150
Berhasil memanggil Servant dengan nama Gudao dan tipe Caster
SUMMON Artoria Lancer 200
Tidak bisa memanggil Servant karena kesalahan
BATTLE Dipsi Medusa
Dipsi menang dalam pertarungan dan Medusa menjadi terluka. Selisih kekuatan mereka adalah 7.5
BATTLE Artoria Dipsi
Artoria menang dalam pertarungan dan Dipsi menjadi terluka. Selisih kekuatan mereka adalah 57.5
BATTLE Lancelot Heracles
Lancelot dan Heracles dalam pertarungan sengit yang berakhir seri. Keduanya menjadi terluka.
TRAIN Emiya
Emiya berlatih keras dan kekuatannya meningkat menjadi 330.0
TRAIN Dipsi
Dipsi berlatih keras dan kekuatannya meningkat menjadi 275.0
TRAIN Saber
Nama Servant tidak ada
HEAL Gudako
Servant Gudako keadaannya masih sehat.
HEAL Dipsi
Servant Dipsi berhasil dipulihkan dan kembali sehat
STATS Artoria
Nama : Artoria
Tipe : Saber
Power : 500
Level : 2
Status : Sehat
SELESAI

Komponen Penilaian

- 40% : Membuat class Servant dan method sesuai permintaan soal.
- 10% : Membuat sistem validasi sesuai permintaan soal.
- 10% : Menggunakan dictionary untuk menyimpan object Servant sesuai nama.
- 30% : Berhasil mencetak hasil sesuai permintaan soal.
- 10% : Kerapihan kode, pemberian dokumentasi/komentar, penamaan file yang sesuai dengan permintaan soal.
- **10% : Membuat sub-class Knight, Cavalry, dan Berserker (Bonus)**

Format Pengumpulan

- [NPM]_[NAMA]_[Kode Asdos]_TUTORIAL[Nomor Lab].py contoh: 1706039566_Gagah Pangeran Rosfatiputra_GPR_TUTORIAL07.py
- [Kode Asdos] diisi dengan: Kode asdos masing-masing sesuai pembagian pada scele