Dasar-Dasar Pemrograman 1 Gasal 2024/2025

Lab 07 Sesi 2: Sets and Dictionary

Deadline: Selasa, 5 November 2024, pukul 16:20 WIB (80 menit)

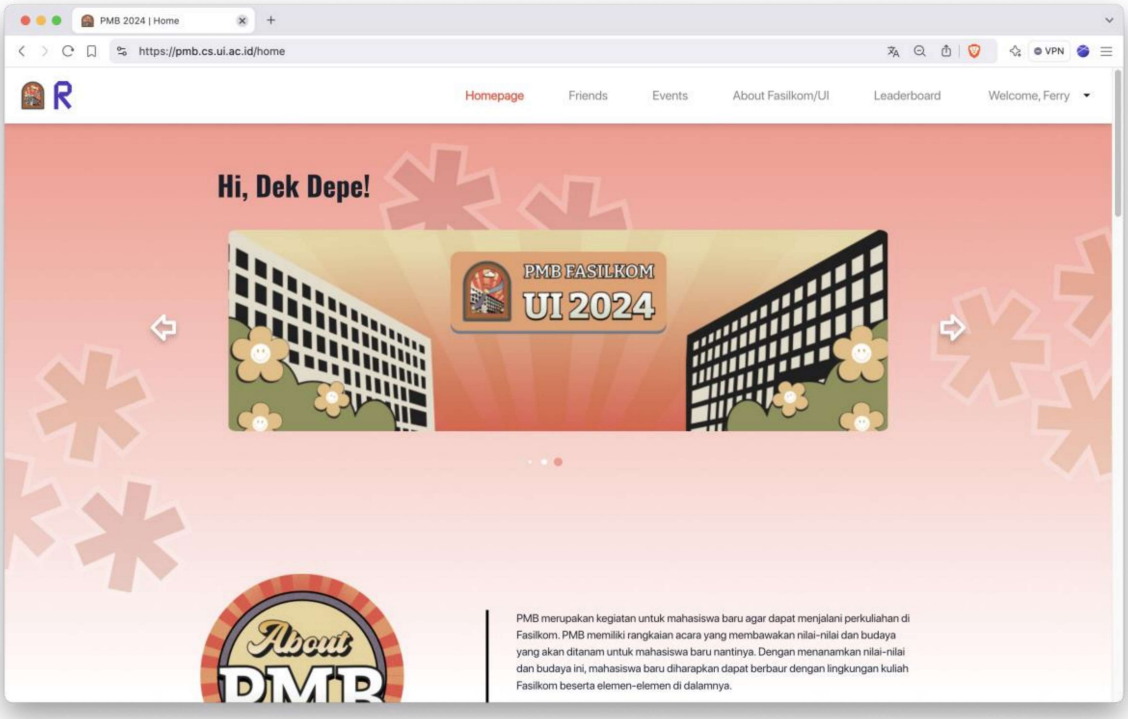
Komponen yang Diuji:

1. Sub-CPMK 7: Mampu memanipulasi *list*, *tuples*, *sets* & *dictionaries* untuk menyelesaikan permasalahan komputasi sederhana

Riwayat Versi

Versi	Timestamp	Keterangan	Warna
1.	5-11-2024; 22:00 WIB	Rilis Pertama	-

PMB: *Finding Mutual Friend's Friend*



Sumber: pmb.cs.ui.ac.id

Halo! Hari ini, kamu dan Dek Depe mendapatkan tugas untuk menambah wawasan teman di lingkungan Pacil. Salah satu website yang kamu harus gunakan adalah website pmb.cs.ui.ac.id. Namun, kamu merasa bahwa website ini masih kurang sempurna untuk mengenal **jaringan pertemanan** di Pacil. Maka dari itu, kamu dan Dek Depe berencana menyempurnakan website PMB Fasilkom untuk memudahkan pencarian **teman mutual!**

Ketentuan Program

Kamu diminta untuk membuat sebuah program utama, sebuah **dictionary KOSONG** DI AWAL PROGRAM sebagai tempat menyimpan data user dan empat buah fungsi.

```
network = {}

def add_user(network, user):
    ...
```

```
...
```

PART I : Add User (0 Poin)

Untuk memudahkan pengembangan program ini, Dek Depe telah menyediakan sebuah fungsi untuk **menambahkan user** ke dalam dictionary. Berikut adalah ketentuan dari fungsi tersebut:

- Fungsi ini bertujuan untuk menambahkan mahasiswa baru ke dalam jaringan.
- **Mahasiswa tidak perlu memikirkan cara untuk membuat fungsi ini, cukup gunakan fungsi di bawah ini dalam pengerjaan kalian.**

```
def add_user(network, user):
    if user not in network:
        network[user] = set()
        print(f"{user} berhasil ditambahkan sebagai
user.")
    else:
        print(f"User {user} sudah terdaftar!")
```

- Penjelasan
  Fungsi `add_user` ini berfungsi untuk menambahkan user baru. Setiap kali user baru ditambahkan, fungsi ini akan membuat entri *dictionary* baru dengan **key berupa nama user** dan **value berupa sebuah set kosong** yang akan digunakan untuk menyimpan daftar teman dari user tersebut.

PART II : Add Friend (30 Poin)

Selanjutnya, kamu diminta untuk membuat *function* yang dapat menambahkan **teman pacil** ke dalam **set** dari masing-masing user.

- Fungsi ini memiliki 3 parameter, yakni:
  - **network**: sebuah dictionary jaringan pertemanan
  - **user1:** string nama user pertama
  - **user2:** string nama user kedua yang ingin ditambahkan ke dalam jaringan pertemanan sebagai teman dari user pertama.

```
def add_friend(network, user1, user2):
    ...
```

- Fungsi ini **tidak mengembalikan** apapun.
- Kamu perlu meng-*handle 2 kasus dalam fungsi add_friend ini.*
  - Jika **user1** dan **user2** belum berteman, maka kamu perlu menambahkan user2 menjadi teman user1 dan user1 menjadi teman user2 serta menampilkan pesan **f"Pertemanan {user1} dan {user2} berhasil ditambahkan."**

```
>>> add_friend(network, user1, user2)
    Pertemanan user1 dan user2 berhasil ditambahkan.
```

  - Jika **user2** ataupun **user1** tidak ditemukan dalam jaringan pertemanan, maka fungsi menampilkan pesan **f"User {user1} atau {user2} tidak terdaftar!"**

```
>>> add_friend(network, user1, user2)
    User user1 atau user2 tidak terdaftar!
```

- Penjelasan
  Fungsi **`add_friend`** ini berfungsi untuk menambahkan **teman** baru dari suatu user. Setiap kali teman baru ditambahkan, fungsi ini akan menambahkan nama teman berupa string ke dalam **set** yang merupakan **value** dari user tersebut.

💡 Kamu bisa memanfaatkan set method **add()** untuk menyelesaikan fungsi ini.

PART III : Find Mutual Friend of Two Users (30 Poin)

Kamu juga diminta untuk membuat suatu fungsi yang dapat membantu Dek Depe untuk **menemukan teman bersama (mutual friends) antara dua user** dalam suatu jaringan pertemanan.

- Fungsi ini memiliki 3 parameter, yakni:
  - **network**: sebuah dictionary jaringan pertemanan
  - **user1:** string nama user pertama
  - **user2:** string nama user kedua

```
def mutual_friends(network, user1, user2):
    ...
```

- Fungsi ini **tidak mengembalikan** apapun.
- Harus dipastikan bahwa `user1` dan `user2` terdaftar di network. Maka dari itu, fungsi ini harus menghandle kasus berikut:
  - Jika set yang dihasilkan **kosong** atau tidak ada mutual friends, maka program mengeluarkan output **"Tidak ada teman mutual antara {user1} dan {user2}."2**

```
>>> mutual_friends(network, user1, user2)
    Tidak ada teman mutual antara user1 dan user2.
```

  - Jika `user1` dan `user2` sama-sama ada di dalam network, cari teman bersama mereka dan tampilkan pesan **"Teman mutual dari {user1} dan {user2}: {mutual}"**

```
>>> mutual_friends(network, user1, user2)
    Teman mutual dari user1 dan user2: {user3, user4}
```

    - Keterangan: user3, user4 merupakan mutual friend dari user1 dan user2

  - Jika salah satu atau kedua pengguna tidak ditemukan, tampilkan pesan **"User {user1} atau {user2} tidak terdaftar!"**

```
>>> mutual_friends(network, user1, user5)
    User user1 atau user5 tidak terdaftar!
```

💡 Baca terkait Join Sets. Dapat mengaksesnya disini.

💡 Kamu bisa memanfaatkan set method **intersection()** untuk menyelesaikan fungsi ini.

PART IV : Find Suggested Friends (30 Poin)

Terakhir, kamu diminta untuk membuat *function* yang dapat memperlihatkan teman yang **direkomendasikan** kepada kamu. Teman-teman yang direkomendasikan adalah **teman dari teman yang belum menjadi teman kamu**.

- Fungsi ini memiliki 2 parameter, yakni:
  - **network**: sebuah dictionary jaringan pertemanan
  - **user:** string nama user

```
def suggest_friend(network, user):
```

```
    . . .
```

- Salah satu ide implementasinya adalah sebagai berikut, tetapi kalian bebas mengimplementasikannya dengan cara lain:
  - Program akan membuat sebuah **set kosong** yang akan diisi dengan *suggestion* teman untuk user.
  - Melakukan looping dari seluruh teman yang dimiliki oleh user. Asumsikan nama user ini adalah user1.
  - Asumsikan nama teman dari user pada setiap iterasi adalah user2. Pada setiap iterasi teman, fungsi akan mengecek apakah ada teman dari user2 yang bukan merupakan teman dari user1.
  - Jika ada, maka tambahkan ke dalam set tersebut.
  - Program akan lanjut melakukan looping.

- Fungsi ini **tidak mengembalikan** apapun.
- Kamu perlu meng-*handle 3* kasus dalam fungsi suggest_friends ini.
  - Jika set yang dihasilkan **kosong** atau tidak ada teman yang bisa direkomendasikan, maka program mengeluarkan output **"Tidak ada teman yang dapat disarankan untuk {user} :("**

```
>>> suggest_friend(network, user)
    Tidak ada teman yang dapat disarankan untuk {user} :("
```

  - Jika user tidak ditemukan pada jaringan pertemanan, maka program mengeluarkan output **f"User {user} tidak terdaftar!"**

```
>>> suggest_friend(network, user)
    User user tidak terdaftar!
```

  - Jika set yang dihasilkan **tidak kosong**, maka program mengeluarkan output **f"Teman yang disarankan untuk {user}: {suggestions}"**.

```
>>> suggest_friend(network, user)
    Teman yang disarankan untuk user: {friend1, friend2, friend3}
```


Kamu bisa memanfaatkan set method **union()** dan **difference()** **untuk menyelesaikan fungsi ini.**

Batasan

1. Opsi **dijamin 1-5**

Test Case

- Merah untuk input
- Biru untuk output

Contoh Interaksi Program 1:

```
Selamat datang di PMB Friend!
Berikut adalah menu yang dapat kamu pilih:
   1) Tambah user
   2) Tambah pertemanan
   3) Cari teman mutual
   4) Sarankan pertemanan
   5) Keluar

Masukkan menu yang diinginkan (1-5): 1
Masukkan nama user yang ingin ditambahkan: Budi
Budi berhasil ditambahkan sebagai user.

Masukkan menu yang diinginkan (1-5): 1
Masukkan nama user yang ingin ditambahkan: Ari
Ari berhasil ditambahkan sebagai user.

Masukkan menu yang diinginkan (1-5): 1
Masukkan nama user yang ingin ditambahkan: Jojo
Jojo berhasil ditambahkan sebagai user.

Masukkan menu yang diinginkan (1-5): 1
Masukkan nama user yang ingin ditambahkan: Lucy
Lucy berhasil ditambahkan sebagai user.

Masukkan menu yang diinginkan (1-5): 2
Masukkan nama user pertama: Budi
Masukkan nama user kedua: Ari
Pertemanan Budi dan Ari berhasil ditambahkan.

Masukkan menu yang diinginkan (1-5): 2
Masukkan nama user pertama: Budi
Masukkan nama user kedua: Jojo
Pertemanan Budi dan Jojo berhasil ditambahkan.

Masukkan menu yang diinginkan (1-5): 2
Masukkan nama user pertama: Ari
Masukkan nama user kedua: Jojo
Pertemanan Ari dan Jojo berhasil ditambahkan.

Masukkan menu yang diinginkan (1-5): 2
Masukkan nama user pertama: Jojo
Masukkan nama user kedua: Lucy
Pertemanan Jojo dan Lucy berhasil ditambahkan.

Masukkan menu yang diinginkan (1-5): 3
Nama user pertama: Budi
Nama user kedua: Jojo
Teman mutual dari Budi dan Jojo: {'Ari'}
```

```
Masukkan menu yang diinginkan (1-5): 3
Nama user pertama: Ari
Nama user kedua: Lucy
Teman mutual dari Ari dan Lucy: {'Jojo'}

Masukkan menu yang diinginkan (1-5): 3
Nama user pertama: Jojo
Nama user kedua: Lucy
Tidak ada teman mutual antara Jojo dan Lucy.

Masukkan menu yang diinginkan (1-5): 4
Nama user yang ingin disarankan pertemanannya: Budi
Teman yang disarankan untuk Budi: {'Lucy'}

Masukkan menu yang diinginkan (1-5): 4
Nama user yang ingin disarankan pertemanannya: Lucy
Teman yang disarankan untuk Lucy: {'Ari', 'Budi'}

Masukkan menu yang diinginkan (1-5): 4
Nama user yang ingin disarankan pertemanannya: Jojo
Tidak ada teman yang dapat disarankan untuk Jojo :(

Masukkan menu yang diinginkan (1-5): 5
Terima kasih telah berkunjung ke PMB Friend!
```

Penjelasan Interaksi Program 1

Setelah menambahkan semua user (perintah 1) dan semua pertemanan (perintah 2), *dictionary* **network** memiliki isi sebagai berikut.

```
network = {
    'Budi': {'Ari', 'Jojo'},
    'Ari': {'Budi', 'Jojo'},
    'Jojo': {'Budi', 'Ari', 'Lucy'},
    'Lucy': {'Jojo'}
}
```

Output interaksi di bawah

```
Masukkan menu yang diinginkan (1-5): 3
Nama user pertama: Budi
Nama user kedua: Jojo
Teman mutual dari Budi dan Jojo: {'Ari'}
```

dapat diperoleh dengan melakukan operasi berikut:

```
Teman Budi ∩ Teman Jojo
= {'Ari', 'Jojo'} ∩ {'Budi', 'Ari', 'Lucy'}
= {'Ari'}
```

Adapun output interaksi di bawah

```
Masukkan menu yang diinginkan (1-5): 4
```

```
Nama user yang ingin disarankan pertemanannya: Budi
Teman yang disarankan untuk Budi: {'Lucy'}
```

dapat diperoleh dengan melakukan operasi berikut:

```
(Teman Ari ∪ Teman Jojo) - Teman Budi - {Budi}
= ({'Budi','Jojo'} ∪ {'Budi','Ari','Lucy'}) - {'Ari','Jojo'} -
{'Budi'}
= {'Budi','Jojo','Ari','Lucy'} - {'Ari','Jojo'} - {'Budi'}
= {'Lucy'}
```

Contoh Interaksi Program 2:

```
Selamat datang di PMB Friend!
Berikut adalah menu yang dapat kamu pilih:
   1) Tambah user
   2) Tambah pertemanan
   3) Cari teman mutual
   4) Sarankan pertemanan
   5) Keluar

Masukkan menu yang diinginkan (1-5): 1
Nama user yang ingin disarankan pertemanannya: Rose
Rose berhasil ditambahkan sebagai user.

Masukkan menu yang diinginkan (1-5): 1
Nama user yang ingin disarankan pertemanannya: Rose
User Rose sudah terdaftar!

Masukkan menu yang diinginkan (1-5): 2
Masukkan nama user pertama: Rose
Masukkan nama user kedua: Bruno
User Rose atau Bruno tidak terdaftar!

Masukkan menu yang diinginkan (1-5): 3
Nama user pertama: Rose
Nama user kedua: Bruno
User Rose atau Bruno tidak terdaftar!

Masukkan menu yang diinginkan (1-5): 4
Nama user yang ingin disarankan pertemanannya: Bruno
User Bruno tidak terdaftar!

Masukkan menu yang diinginkan (1-5): 1
Nama user yang ingin disarankan pertemanannya: Bruno
Bruno berhasil ditambahkan sebagai user.

Masukkan menu yang diinginkan (1-5): 2
Masukkan nama user pertama: Rose
Masukkan nama user kedua: Bruno
Pertemanan Rose dan Bruno berhasil ditambahkan.
```

```
Masukkan menu yang diinginkan (1-5): 3
Nama user pertama: Rose
Nama user kedua: Bruno
Tidak ada teman mutual antara Rose dan Bruno.

Masukkan menu yang diinginkan (1-5): 4
Nama user yang ingin disarankan pertemanannya: Rose
Tidak ada teman yang dapat disarankan untuk Rose :(

Masukkan menu yang diinginkan (1-5): 5
Terima kasih telah berkunjung ke PMB Friend!
```

Ketentuan Umum Program

1. **Dilarang melakukan kerja sama, menyontek, dan tindakan kecurangan lainnya** dengan sesama mahasiswa/i maupun dengan menggunakan ***artificial intelligence*** dalam mengerjakan lab.
2. Penamaan variabel harus mengikuti Python Naming Convention (referensi).
3. Penamaan modul, class, method, dan variabel harus jelas dan tidak ambigu (referensi).
4. Dokumentasikan kode menggunakan *comment*.
5. Deadline pengumpulan adalah pukul **16:20**. Tempat pengumpulan akan ditutup pukul **16:40**. Setelah periode tersebut, submisi tidak akan diterima.

Komponen Penilaian

Komponen penilaian dapat dilihat pada tautan **ini**.

Berkas yang Perlu Dikumpulkan

- lab07.py

Kumpulkan berkas lab07.py yang telah di-zip dengan format penamaan seperti berikut.
`[KodeAsdos]_[Kelas]_[NPM]_[NamaLengkap]_Lab07.zip`

Contoh:
`FER_A_1234567890_FransiscoWilliam_Lab07.zip`

Gud Lak dan semangatt!!