12. Tipe Data Set

12.1 Tujuan Praktikum

Setelah mempelajari Bab ini, mahasiswa diharapkan dapat:

1. Dapat melakukan operasi-operasi dasar pada tipe data Set, seperti mendefinisikan, mengakses, mengubah, menghapus dan melakukan pencarian data pada Set
2. Dapat melakukan operasi-operasi yang melibatkan dua set atau lebih, seperti union, difference, intersection dan symmetric difference

12.2 Alat dan Bahan

Praktikum ini membutuhkan perangkat komputer yang memiliki spesifikasi minimum sebagai berikut:

1. Terkoneksi ke Internet dan dapat mengunduh package-package Python.
2. Mampu menjalankan sistem operasi Windows 10 atau Ubuntu Linux.

Perangkat lunak yang diperlukan untuk mendukung praktikum ini adalah sebagai berikut:

1. Python 3.7 atau 3.8 yang terinstall menggunakan Anaconda atau Installer Python lainnya.
2. Web Browser (Mozilla Firefox, Microsoft Edge atau Google Chrome).
3. Command Prompt (jika menggunakan Windows).
4. Terminal (jika menggunakan Linux).
5. Editor Python (Visual Studio Code, PyCharm, Spyder atau editor-editor lainnya yang mendukung Python).

12.3 Materi

12.3.1 Pengenalan dan Mendefinisikan Set

Set adalah salah satu tipe data pada Python yang dapat digunakan untuk menyimpan sekumpulan data yang semuanya unik. Biasa juga dikenal dengan istilah himpunan. Beberapa sifat Set pada Python adalah:

- Isi dari Set disebut sebagai anggota (member).

- Anggota dari Set harus bersifat immutable. Beberapa tipe data immutable pada Python: integer, float, string, tuple dan lain-lain. Dengan demikian list dan dictionary (mutable) tidak dapat dimasukkan ke dalam Set.
- Set sendiri bersifat mutable, artinya anda dapat menambah atau mengurangi isi dari sebuah Set. Karena itu Set tidak dapat dimasukkan ke dalam Set.

Untuk mendefinisikan Set, ada beberapa cara yang dapat dilakukan yaitu dengan menggunakan notasi {} dan fungsi set() seperti contoh berikut ini:

```
# dengan menggunakan {}
bilangan_genap = {2, 4, 6, 8, 10, 12}
bilangan_ganjil = {1, 3, 5, 7, 9, 11}

# dengan menggunakan fungsi set()
pernah_ke_bulan = set('Neil Armstrong', 'Buzz Aldrin')
```

Untuk mendefinisikan Set kosong anda tidak dapat menggunakan notasi {}, tetapi harus menggunakan fungsi set() seperti contoh berikut:

```
# dengan fungsi set()
pernah_ke_mars = set()    # menghasilkan set kosong

data = {}     # ini akan menghasilkan dictionary kosong
```

12.3.2 Pengaksesan Set

Set tidak memiliki indeks, karena itu anda tidak dapat mengakses anggota-anggota dari sebuah Set secara langsung menggunakan indeks. Perhatikan contoh program berikut ini:

```
nim = {'71200120', '71200195', '71200214'}
jumlah_nim = len(nim)
print(jumlah_nim)        # akan menghasilkan output 3

# tampilkan isi set satu-persatu
for n in nim:
    print(n)
```

Output yang dihasilkan dari program tersebut adalah sebagai berikut:

```
3
71200214
71200195
71200120
```

Jika dicermati, output yang dihasilkan urutannya berbeda dengan deklarasi Set sebelumnya. Hal ini disebabkan karena pada Set tidak ada indeks, sehingga tidak ada urutan posisi anggota. Pada tipe data Set, posisi anggota tidaklah penting.

Set adalah tipe data mutable, artinya isinya bisa bertambah atau berkurang. Program berikut ini menunjukkan bagaimana cara menambah anggota sebuah Set dengan fungsi add():

```
# definisikan sebuah set kosong
plat_nomor = set()

# tambahkan plat nomor 'AB 1890 XA'
plat_nomor.add('AB 1890 XA')

# tambahkan plat nomor 'AD 6810 MT'
plat_nomor.add('AD 6810 MT')

# jumlah anggota di dalam Set
print(len(plat_nomor))

# tambahkan plat yang sama sekali lagi
plat_nomor.add('AB 1890 XA')

# tampilkan semua plat nomor
for plat in plat_nomor:
    print(plat)
```

Output yang dihasilkan adalah sebagai berikut:

```
2
AD 6810 MT
AB 1890 XA
```

Set memiliki mekanisme untuk melakukan pengecekan apakah anggota baru yang akan dimasukkan sudah ada di dalam Set (cek duplikasi). Jika belum ada, maka anggota tersebut bisa masuk ke dalam Set. Tetapi jika ternyata di dalam Set sudah ada anggota dengan nilai yang sama, maka pemanggilan fungsi add() tidak akan menambah anggota dari Set. Pengecekan duplikasi ini sudah ada di dalam fungsi add(), sehingga anda tidak perlu melakukannya sendiri.

Untuk menghapus anggota dari sebuah Set, ada beberapa cara yaitu dengan fungsi discard(), remove(), pop() dan clear(). Perbedaan dari empat fungsi tersebut dapat dilihat pada Gambar 12.1.

Berikut ini adalah contoh program yang mendemonstrasikan penghapusan anggota dari sebuah Set:

```
bilangan_prima = {13, 23, 7, 29, 11, 5}

# hapus 5 dari set tersebut
bilangan_prima.remove(5)
```

discard()	remove()	pop()	clear()
Menghapus satu elemen yang disebutkan	Menghapus satu elemen yang disebutkan	Mengambil salah satu dan menghapusnya dari set (tidak tentu)	Menghapus seluruh elemen di dalam set
Tidak ada error	Muncul error jika elemen yang dihapus tidak ada	Error jika set kosong	Tidak ada error

Gambar 12.1: Fungsi-fungsi untuk menghapus anggota dari Set.

```
print(bilangan_prima)

# hapus 97 (tidak ada)
bilangan_prima.discard(97)
print(bilangan_prima)

# ambil dan hapus salah satu
bilangan = bilangan_prima.pop()
print(bilangan)
print(bilangan_prima)

# kosongkan set
bilangan_prima.clear()
print(bilangan_prima)
```

Output yang dihasilkan dari program tersebut adalah:

```
{5, 7, 11, 13, 23, 29}      # awal
{7, 11, 13, 23, 29}         # setelah 5 dihapus. Perhatikan urutan berubah
{7, 11, 13, 23, 29}         # 97 tidak ada, sehingga Set tidak berubah
7                           # fungsi pop() mengeluarkan 7
{11, 13, 23, 29}            # setelah 7 keluar dari Set
set()                       # setelah isi dari Set dihapus semua
```

Fungsi discard() tidak akan menghasilkan error jika anggota yang ingin dihapus tidak ada di dalam Set. Sedangkan fungsi pop() akan mengambil salah satu anggota (secara acak) dan mengeluarkannya dari Set. Fungsi pop() akan berguna jika kita ingin memproses isi dari Set satu-persatu tanpa memperdulikan urutan/posisi dari setiap anggota di dalam Set.

Bagaimana jika anda ingin mengubah nilai salah satu anggota di dalam Set? Pada Set anda tidak bisa mengubah langsung nilai dari salah satu anggota di dalam Set. Yang dapat dilakukan adalah melakukan operasi penggantian (replace) dengan cara menghapus anggota yang mau diubah,

kemudian memasukkan anggota baru yang nilainya sesuai dengan yang diinginkan. Contohnya dapat dilihat pada program berikut ini:

```
# Buat Set dari List
ikan = set(['koi', 'koki', 'kembung', 'salmon'])
print(ikan)

# ganti koi menjadi teri
ikan.remove('koi')
ikan.add('teri')
print(ikan)
```

Output yang dihasilkan dari program tersebut adalah sebagai berikut:

```
{'koi', 'koki', 'salmon', 'kembung'}
{'teri', 'salmon', 'kembung', 'koki'}
```

Untuk mengubah nilai 'koi' menjadi 'teri', yang perlu dilakukan adalah hapus dulu anggota 'koi', kemudian masukkan anggota baru dengan nilai 'teri'. Sekali lagi, yang perlu ditekankan di sini adalah pada Set tidak mengenal adanya posisi/urutan data. Sehingga output dari program tersebut nilai 'koi' sudah tidak ada, digantikan oleh nilai 'teri'. Kemudian urutan isi Set ikan saat ditampilkan tidaklah sama (ada perubahan). Setiap kali ada operasi penambahan dan penghapusan maka urutan anggota di dalam Set biasanya akan berubah.

12.3.3 Operasi-Operasi pada Set

Operasi-operasi pada Set adalah operasi-operasi pada himpunan. Berikut ini adalah daftar operasi-operasi Set pada Python:

- **Operator Union**. Menggabungkan dua Set menjadi satu. Dapat menggunakan operator | maupun fungsi union().
- **Operator Intersection**. Menghasilkan irisan dari dua Set. Dapat menggunakan operator & maupun fungsi intersection().
- **Operator Difference**. Menghasilkan Set baru yang merupakan selisih dari dua Set yang dibandingkan. Dapat menggunakan operator - maupun fungsi difference().
- **Operator Symmetric Difference**. Menghasilkan Set baru yang merupakan jumlah dari dua Set kecuali irisannya. Dapat menggunakan operator m̂aupun fungsi symmetric_difference().

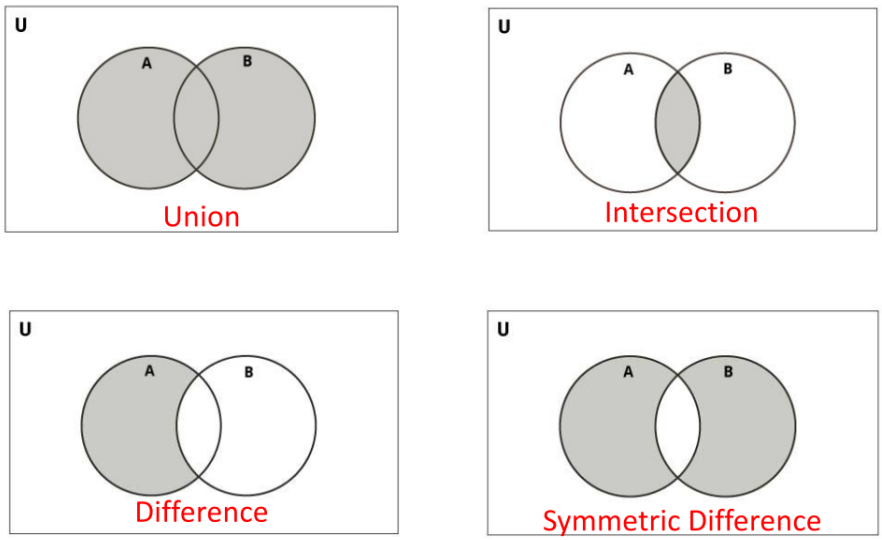
Ilustrasi dari operasi-operasi tersebut dapat dilihat pada Gambar 12.2.

Operator Union

Contoh penggunaan operator union dapat dilihat pada contoh program berikut ini:

```
merek_hp = {'Samsung', 'Apple', 'Xiaomi', 'Sony'}
merek_ac = {'LG', 'Samsung', 'Panasonic', 'Daikin', 'Sony'}

# union dari merek_hp dan merek_ac
gabungan = merek_hp | merek_ac
```



Gambar 12.2: Operasi-operasi pada Set.

```
# bisa juga menggunakan gabungan = merek_hp.union(merek_ac)
print(gabungan)
```

Output yang dihasilkan dari program tersebut adalah anggota dari Set merek_hp digabungkan dengan anggota dari Set merek_ac, menghasilkan Set baru bernama gabungan. Ada duplikasi (anggota yang sama) yaitu 'Samsung' dan 'Sony'. Sehingga 'Samsung' dan 'Sony' di dalam Set gabungan hanya muncul 1 kali karena anggota dari Set harus unik. Outputnya adalah sebagai berikut:

```
{'Xiaomi', 'Sony', 'LG', 'Daikin', 'Samsung', 'Apple', 'Panasonic'}
```

Operator Intersection

Contoh penggunaan operator Intersection dapat dilihat pada program berikut:

```
renang = {'siti', 'mail', 'ikhsan', 'upin', 'ipin'}
tenis = {'joko', 'mail', 'ipin', 'upin', 'tejo'}

# suka renang dan tenis
renang_tenis = renang & tenis
print(renang_tenis)
```

Program tersebut akan menghasilkan output hasil operasi intersection dari Set renang dan tenis. Intersection berarti anggota-anggota yang berada di dalam Set renang dan Set tenis sekaligus, yaitu mail, upin dan ipin. Output yang dihasilkan dari program tersebut adalah:

```
{'upin', 'ipin', 'mail'}
```

Operator Difference

Contoh penggunaan operator difference dapat dilihat pada program berikut:

```
# bisa berbahasa english
english = {'desi', 'tono', 'evan', 'miko', 'takashi', 'chaewon'}

# bisa berbahasa korea
korean = {'chaewon', 'yeona', 'erika', 'miko'}

# siapa yang hanya bisa bahasa korea?
only_korean = korean - english
print(only_korean)

# siapa yang hanya bisa bahasa english?
only_english = english - korean
print(only_english)
```

Operator difference akan menghasilkan Set yang anggotanya merupakan selisih dari kedua Set yang dibandingkan. Pada contoh tersebut digunakan untuk mendapatkan anggota yang hanya bisa berbahasa korea dengan cara mencari selisih antara Set korean dan Set english. Sedangkan kebalikannya, jika ingin mengetahui siapa saja yang hanya bisa berbahasa English maka cari selisih antara set English dan Set Korea. Output yang dihasilkan dari program tersebut adalah sebagai berikut:

```
{'erika', 'yeona'}                  # hanya bisa Korea
{'takashi', 'desi', 'evan', 'tono'} # hanya bisa English
```

Operator Symmetric Difference

Contoh penggunaan operator symmetric difference dapat dilihat pada program berikut ini:

```
english = {'desi', 'tono', 'evan', 'miko', 'takashi', 'chaewon'}
korean = {'chaewon', 'yeona', 'erika', 'miko'}

# hanya bisa bicara satu bahasa saja
one_language = english ^ korean
print(one_language)
```

Operator symmetric difference akan menghasilkan Set baru yang merupakan gabungan dari dua Set tetapi tidak termasuk irisannya. Contoh yang diberikan memanfaatkan operator symmetric difference untuk mendapatkan siapa saja yang hanya dapat berbicara dalam satu bahasa (artinya tidak termasuk irisannya). Secara umum dapat juga dihasilkan dari operasi berikut:

```
one_languange = english.union(korean) - english.intersection(korean)
```

Output yang dihasilkan dari program tersebut adalah:

```
{'yeona', 'tono', 'desi', 'erika', 'evan', 'takashi'}
```

12.4 Kegiatan Praktikum

12.4.1 Contoh-contoh Kasus Set

■ Contoh 12.1 Jumlah seluruh elemen yang unik di dalam List

Diberikan sebuah list yang sudah berisi nilai-nilai integer, hitunglah jumlah seluruh elemen list yang unik di dalam list tersebut! Buatlah dalam bentuk fungsi **unique_sum(list)**

■

Pada kasus ini anda diberi satu buah List yang di dalamnya sudah ada beberapa elemen-elemen bertipe integer. Anda diminta untuk menghitung jumlah seluruh elemen yang unik di dalam list tersebut. Sebagai ilustrasi, jika List tersebut sebagai berikut:

```
data = [2, 4, 3, 2, 7, 8, 6, 4, 5, 5]
```

Jumlah seluruh elemen yang unik adalah $2+4+3+7+8+6+5 = 35$. Nilai 2, 4 dan 5 muncul lebih dari sekali, tetapi hanya dihitung satu kali saja (unik). Untuk memecahkan masalah ini, kita akan menggunakan bantuan Set. Solusi untuk kasus ini dapat dilihat pada program berikut ini:

```
def unique_sum(list):
    # ubah dalam bentuk Set
    data_set = set(list)

    # hitung jumlah seluruh anggota data_set
    total = 0
    for data in data_set:
        total = total + data

    # return hasil akhir
    return total

# contoh penggunaan fungsi unique_sum()
contoh1 = [2, 4, 3, 2, 7, 8, 6, 4, 5, 5]
hasil1 = unique_sum(contoh1)
print(hasil1)
```

■ Contoh 12.2 Duplicate chars in string

Buatlah fungsi **cek_duplikat(string)** yang dapat mengecek apakah dalam string terdapat karakter yang muncul lebih dari satu kali (duplikat). Fungsi akan return True jika ada duplikat, dan sebaliknya return False jika tidak ada duplikat karakter dalam string yang diberikan. ■

Anggota dari suatu Set dipastikan selalu unik, sehingga dalam kasus ini kita akan menggunakan Set. Kita akan menggunakan operator **in** untuk mengecek apakah suatu karakter sudah ada di dalam Set atau tidak. Solusi untuk kasus ini dapat dilihat pada program berikut ini:

```
def cek_duplikat(string):
    # buat set kosong
    karakter_set = set()

    # cek semua karakter dalam string satu-persatu
    for karakter in string:
        # apakah karakter ini ada dalam set?
        if karakter in karakter_set:
            # ternyata ada, berarti duplikat
            # hentikan pengecekan, langsung return
            return True
        else:
            # jika belum ada, masukkan dalam set
            karakter_set.add(karakter)

    # setelah looping semua karakter, tidak ada yang sama
    return False


# test case
string1 = 'Alexander the Great' # duplikat
print(cek_duplikat(string1))

string2 = 'UKDW'     # semua unik
print(cek_duplikat(string2))
```

■ Contoh 12.3 Kategori aplikasi di Play Store

Buatlah program yang meminta input n kategori aplikasi, lalu program meminta pengguna untuk memasukkan nama-nama aplikasi sebanyak 5 untuk masing-masing kategori. Setelah pengguna selesai memasukkan semua nama aplikasi, program akan menampilkan:

- Daftar nama aplikasi di setiap kategori
- Program yang muncul di semua kategori

■

Untuk membuat program ini, langkah-langkah yang perlu diimplementasikan adalah sebagai berikut:

- Minta jumlah kategori (n).
- Minta nama kategori, kemudian minta nama-nama aplikasi sebanyak 5 untuk kategori tersebut
- Setelah semua kategori selesai di-input, berikutnya tampilkan daftar nama aplikasi di setiap kategori
- Dengan operasi intersection, tampilkan nama aplikasi yang muncul di semua kategori yang ada.

Untuk permasalahan ini, kita akan menggunakan bantuan List, Dictionary dan Set. Solusi untuk masalah tersebut dapat dilihat pada program berikut ini:

```
# input n kategori
n = int(input('Masukkan jumlah kategori: '))
```

```
# siapkan dictionary kosong
data_aplikasi = {}

# input nama kategori dan aplikasi di dalamnya
for i in range(n):
    nama_kategori = input('Masukkan nama kategori:')
    print('Masukkan 5 nama aplikasi di kategori', nama_kategori)

    # siapkan list kosong untuk nama-nama aplikasi
    aplikasi = []
    for j in range(5):
        nama_aplikasi = input('Nama aplikasi: ')
        aplikasi.append(nama_aplikasi)

    # masukkan dalam dictionary
    data_aplikasi[nama_kategori] = aplikasi

# tampilkan dictionary data_aplikasi
print(data_aplikasi)

daftar_aplikasi_list = []
# ambil semua daftar aplikasi dari setiap kategori
for aplikasi in data_aplikasi.values():
    daftar_aplikasi_list.append(set(aplikasi))

print(daftar_aplikasi_list)

# lakukan intersection ke semua set yang ada
hasil = daftar_aplikasi_list[0]
for i in range(1, len(daftar_aplikasi_list)):
    hasil = hasil.intersection(daftar_aplikasi_list[i])

print(hasil)
```

Contoh output yang dihasilkan dari program tersebut adalah sebagai berikut:

```
Masukkan jumlah kategori: 2
Masukkan nama kategori:Finance
Masukkan 5 nama aplikasi di kategori Finance
Nama aplikasi: RTI Saham
Nama aplikasi: Mirae
Nama aplikasi: IPOT
Nama aplikasi: Poems
Nama aplikasi: Calculator
Masukkan nama kategori:Utilities
Masukkan 5 nama aplikasi di kategori Utilities
Nama aplikasi: Photos
Nama aplikasi: Weather
```

```
Nama aplikasi: Calculator
Nama aplikasi: Camera
Nama aplikasi: Notes
{'Finance': ['RTI Saham', 'Mirae', 'IPOT', 'Poems', 'Calculator'],
'Utilities': ['Photos', 'Weather', 'Calculator', 'Camera', 'Notes']}
[{'IPOT', 'Poems', 'Calculator', 'RTI Saham', 'Mirae'},
{'Weather', 'Calculator', 'Camera', 'Photos', 'Notes'}]
{'Calculator'}
```

Pada contoh tersebut, pengguna memasukkan 2 kategori, yaitu Finance dan Utilities. Daftar nama aplikasi dari setiap kategori tersebut adalah sebagai berikut:

- Finance: RTI Saham, Mirae, IPOT, Poems, Calculator
- Utilities: Photos, Weather, Calculator, Camera, Notes

Dari input user tersebut, kemudian disusun dalam bentuk dictionary sebagai berikut:

```
{
    'Finance': ['RTI Saham', 'Mirae', 'IPOT', 'Poems', 'Calculator'],
    'Utilities': ['Photos', 'Weather', 'Calculator', 'Camera', 'Notes']
}
```

Kemudian dari setiap kategori dilakukan operasi intersection untuk mencari apakah ada nama aplikasi yang muncul di semua kategori. Dari contoh tersebut didapatkan nama aplikasi Calculator yang muncul di semua kategori.

12.5 Latihan Mandiri

Latihan 12.1 Dari contoh kasus kategori kasus di Play Store, tambahkan kemampuan-kemampuan berikut ini:

- Tampilkan nama-nama aplikasi yang hanya muncul di satu kategori saja.
- Untuk input n>2, tampilkan nama-nama aplikasi yang muncul tepat di dua kategori sekaligus

■

Latihan 12.2 Buatlah sebuah program yang mendemonstrasikan konversi dari:

- List menjadi Set
- Set menjadi List
- Tuple menjadi Set
- Set menjadi Tuple

Tampilkan isi data sebelum dan sesudah konversi. ■

Latihan 12.3 Buatlah sebuah program yang dapat membaca dua file teks dan menampilkan semua kata-kata yang muncul pada kedua file tersebut. Beberapa hal yang perlu anda perhatikan:

- Nama file adalah input user. Tampilkan pesan error jika file tidak ditemukan/tidak bisa dibaca.
- Semua kata dikonversi dulu menjadi lowercase.
- Sertakan contoh file teks yang anda pakai saat mengumpulkan laporan.

■