Laporan Praktikum Algoritma dan Pemrograman

Semester Genap 2023/2024

NIM	**71230985**
Nama Lengkap	**TOMAS BECKET**
Minggu ke / Materi	**12 / Tipe Data Set**

SAYA MENYATAKAN BAHWA LAPORAN PRAKTIKUM INI SAYA BUAT DENGAN USAHA SENDIRI TANPA MENGGUNAKAN BANTUAN ORANG LAIN. SEMUA MATERI YANG SAYA AMBIL DARI SUMBER LAIN SUDAH SAYA CANTUMKAN SUMBERNYA DAN TELAH SAYA TULIS ULANG DENGAN BAHASA SAYA SENDIRI.

SAYA SANGGUP MENERIMA SANKSI JIKA MELAKUKAN KEGIATAN PLAGIASI, TERMASUK SANKSI TIDAK LULUS MATA KULIAH INI.

PROGRAM STUDI INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
UNIVERSITAS KRISTEN DUTA WACANA
YOGYAKARTA
2024

BAGIAN 1: MATERI MINGGU INI (40%)

MATERI 1

PENGENALAN DAN MENDEFINISIKAN SET

Set adalah salah satu tipe data fundamental dalam Python yang digunakan untuk menyimpan kumpulan data yang unik dan tidak berurutan. Set sering disebut juga dengan istilah himpunan.

Sifat-sifat Set antara lain:

- Anggota (Member), Isi dari Set disebut sebagai anggota (member). Setiap anggota Set haruslah unik, artinya tidak boleh ada duplikat.
- Immutable pada Anggotanya, Anggota Set tidak dapat diubah. Tipe data yang immutable dalam Python seperti integer, float, string, dan tuple dapat menjadi anggota Set. Sebaliknya, tipe data mutable seperti list dan dictionary tidak dapat dimasukkan ke dalam Set.
- Mutable pada Set itu Sendiri, Meskipun anggotanya immutable, Set itu sendiri mutable. Artinya, Anda dapat menambah atau menghapus anggota dari sebuah Set.
- Tidak Berurutan, Urutan anggota dalam Set tidak terdefinisi. Artinya, ketika iterasi melalui Set, urutan anggota yang ditampilkan dapat berbeda-beda.
- Operasi Set, Set menyediakan berbagai operasi matematika set seperti gabungan (union), irisan (intersection), selisih (difference), dan selisih simetris (symmetric difference).

Misal pada source code dibawah

```
# PENGENALAN DAN MENDEFINISIKAN SET
iniset = {'a', 'b', 'c'}
print(iniset)
```

hasil ouput yang muncul tidak urut seperti a, b, c melainkan random urutannya tidak seperti list yang berpaku pada sebuah index dan urut

```
{'c', 'a', 'b'}
```

```
{'c', 'b', 'a'}
```

MATERI 2

PENGAKSESAN SET

Set memiliki mekanisme cerdas untuk mencegah duplikasi saat menambahkan anggota baru. Hal ini memungkinkan Anda untuk memastikan bahwa data Anda selalu terjaga keunikannya tanpa perlu melakukan pemeriksaan manual.

Selain itu, Set menyediakan beberapa fungsi untuk mengelola anggotanya dengan efisien. Anda dapat menambahkan, menghapus, atau bahkan menghapus anggota secara acak dengan mudah.

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

Berikut Tabel fungsi-fungsi untuk menghapus anggota dari set. Terdapat pada Module Tipe Data Set 13. Fungsi discard() tidak akan menghasilkan error jika elemen yang ingin dihapus tidak ada di dalam set. Sebaliknya, fungsi pop() akan menghapus dan mengembalikan salah satu elemen secara acak dari set. Fungsi pop() berguna ketika kita ingin memproses elemen dalam set satu per satu tanpa memperhatikan urutan atau posisi dari setiap elemen di dalam set.

Contoh Source Code seperti dibawah ini :

```
angka = {1, 2, 3, 4, 5}

angka.discard(6)
print("Setelah discard(6):", angka)

elemen_dihapus = angka.pop()
print("Elemen yang dihapus dengan pop():", elemen_dihapus)
print("Setelah pop():", angka)
```

Hasil Output :

```
Setelah discard(6): {1, 2, 3, 4, 5}
Elemen yang dihapus dengan pop(): 1
Setelah pop(): {2, 3, 4, 5}
```

dan pada set kita tidak dapat mengubah hasil melainkan mereplace set tersebut dengan yang lain. Contoh nya seperti ini

```
angka = {1, 2, 3, 4, 5}

angka.remove(3)
print("Setelah menghapus nilai 3:", angka)

angka.add(6)
print("Setelah menambahkan nilai 6:", angka)
```

karena set tidak bisa diubah angkanya, maka kita dapat meremove dan mengadd hasil yang kita inginkan atau mengganti nya menggunakan remove dan add

Hasil outputnya yaitu :

```
Setelah menghapus nilai 3: {1, 2, 4, 5}
Setelah menambahkan nilai 6: {1, 2, 4, 5, 6}
```

MATERI 3

OPERASI-OPERASI PADA SET

- Gabungan (Union): Menggabungkan dua set menjadi satu set. kita dapat menggunakan operator | atau metode union().
- Irisan (Intersection): Menghasilkan set yang berisi elemen-elemen yang ada di kedua set. Kita bisa menggunakan operator & atau metode intersection().
- Selisih (Difference): Menghasilkan set yang berisi elemen-elemen yang hanya ada di set pertama dan tidak ada di set kedua. kita bisa menggunakan operator - atau metode difference().
- Selisih Simetris (Symmetric Difference): Menghasilkan set yang berisi elemen-elemen yang ada di salah satu set, tetapi tidak ada di kedua set. kita bisa menggunakan operator ^ atau metode symmetric_difference().

Dengan menggunakan operasi-operasi ini, kita dapat dengan mudah melakukan operasi himpunan seperti gabungan, irisan, selisih, dan selisih simetris pada set di Python.

Contoh Operator Union :

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

gabungan = merek_hp | merek_ac

print(gabungan)
```

Hasil Output :

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

Program ini menghasilkan Set baru bernama gabungan yang berisi gabungan anggota dari Set merek_hp dan Set merek_ac.

Proses penggabungan ini dilakukan dengan menggabungkan elemen-elemen dari kedua Set tanpa duplikasi.

Meskipun terdapat elemen yang sama di kedua Set, seperti 'Samsung' dan 'Sony', Set gabungan hanya akan menampilkan elemen tersebut satu kali, sesuai dengan sifat unik Set.

Hal ini berarti bahwa meskipun 'Samsung' dan 'Sony' ada di kedua Set, mereka hanya akan dihitung sebagai satu elemen dalam Set gabungan.

Dengan demikian, Set gabungan akan berisi semua merek HP dan AC yang unik, tanpa duplikasi.

Contoh Operator Intersection :

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

renang_tenis = renang & tenis
print(renang_tenis)
```

Hasil output :

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

Program ini akan menghasilkan Set baru yang berisi anggota-anggota yang terdapat di kedua Set, yaitu Set renang dan Set tenis.

Operasi ini disebut dengan operasi intersection (perpotongan).

Artinya, Set baru ini hanya akan memuat anggota yang dimiliki bersama oleh Set renang dan Set tenis.

Setelah dilakukan operasi intersection, program akan menghasilkan Set baru bernama gabungan:

gabungan = {'mail', 'upin', 'ipin'}

Seperti yang Anda lihat, Set gabungan hanya berisi tiga nama yang terdapat di kedua Set, yaitu 'mail', 'upin', dan 'ipin'.

Nama-nama lain, seperti 'siti' dan 'ikhsan', tidak termasuk dalam Set gabungan karena mereka tidak ditemukan di kedua Set.

Contoh Operator Difference :

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

korean = {'chaewon', 'yeona', 'erika', 'miko'}

only_korean = korean - english
print(only_korean)

only_english = english - korean
print(only_english)
```

Hasil Output :

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

Operator difference dalam Python memungkinkan kita untuk menemukan anggota yang unik dari dua Set. Pada contoh yang diberikan, operator ini digunakan untuk mencari anggota yang hanya bisa berbahasa Korea.

Caranya adalah dengan mencari selisih antara Set korean dan Set english. Sehingga ketika Set korean berisi orang-orang yang bisa berbahasa Korea. Sedangkan Set english berisi orang-orang yang bisa berbahasa Inggris. Dengan menggunakan operator difference, kita dapat mengetahui siapa saja yang hanya bisa berbahasa Korea, yaitu orang-orang yang tidak ada di Set english.

LINK GITHUB : https://github.com/TomasBeckett/PrakAlpro13.git

BAGIAN 2: LATIHAN MANDIRI (60%)

SOAL 1

1.1 SOURCE CODE

```
data_aplikasi = {}

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

    aplikasi = []
    for j in range(5):
        nama_aplikasi = input('Nama aplikasi: ')
        aplikasi.append(nama_aplikasi)

    data_aplikasi[nama_kategori] = aplikasi

print(data_aplikasi)

daftar_aplikasi_list = []

for aplikasi in data_aplikasi.values():
    daftar_aplikasi_list.append(set(aplikasi))

print(daftar_aplikasi_list)

hasil = daftar_aplikasi_list[0]
for i in range(1, len(daftar_aplikasi_list)):
    hasil = hasil.intersection(daftar_aplikasi_list[i])

print(hasil)

kemunculan_aplikasi = {}

for aplikasi_set in daftar_aplikasi_list:
    for aplikasi in aplikasi_set:
        if aplikasi in kemunculan_aplikasi:
            kemunculan_aplikasi[aplikasi] += 1
        else:
            kemunculan_aplikasi[aplikasi] = 1

print("Kemunculan aplikasi:", kemunculan_aplikasi)
```

```
aplikasi_unik = {aplikasi for aplikasi, count in kemunculan_aplikasi.items() if
count == 1}

print("Aplikasi yang hanya muncul di satu kategori:", aplikasi_unik)
```

1.2 PENJELASAN

```
for aplikasi_set in daftar_aplikasi_list:
    for aplikasi in aplikasi_set:
        if aplikasi in kemunculan_aplikasi:
            kemunculan_aplikasi[aplikasi] += 1
        else:
            kemunculan_aplikasi[aplikasi] = 1

print("Kemunculan aplikasi:", kemunculan_aplikasi)


aplikasi_unik = {aplikasi for aplikasi, count in kemunculan_aplikasi.items() if
count == 1}
```

Pada penambahan kode tersebut, Kode ini meminta pengguna untuk memasukkan jumlah kategori dan nama-nama aplikasi dalam setiap kategori, lalu menyimpannya dalam sebuah dictionary (data_aplikasi).

Konversi ke Set: Setiap daftar aplikasi dalam kategori diubah menjadi set dan disimpan dalam daftar_aplikasi_list.

Menghitung Kemunculan: Kode ini menghitung berapa kali setiap aplikasi muncul di semua kategori dan menyimpannya dalam dictionary (kemunculan_aplikasi).

Menampilkan Aplikasi Unik: Kode ini menampilkan aplikasi yang hanya muncul di satu kategori dengan memeriksa aplikasi yang memiliki kemunculan sebanyak satu kali dalam kemunculan_aplikasi.

```
for aplikasi_set in daftar_aplikasi_list:
```

pada line ini daftar_aplikasi_list adalah sebuah daftar yang berisi set aplikasi untuk setiap kategori dan aplikasi_set merepresentasikan setiap set aplikasi dalam daftar tersebut.

```
for aplikasi in aplikasi_set:
```

Untuk setiap aplikasi_set, kita melakukan loop melalui setiap aplikasi dalam set tersebut.

```
if aplikasi in kemunculan_aplikasi:
            kemunculan_aplikasi[aplikasi] += 1
        else:
            kemunculan_aplikasi[aplikasi] = 1
```

Jika aplikasi Sudah Ada dalam kemunculan_aplikasi:

Jika aplikasi sudah ada sebagai kunci dalam dictionary kemunculan_aplikasi, maka nilai dari kunci tersebut ditambah 1, menandakan bahwa aplikasi tersebut muncul lagi.

Jika aplikasi Belum Ada dalam kemunculan_aplikasi:

Jika aplikasi belum ada sebagai kunci dalam dictionary kemunculan_aplikasi, maka kita menambahkannya dengan nilai awal 1, menandakan bahwa ini adalah kemunculan pertama dari aplikasi tersebut.

1.3 HASIL OUPUT

```
Masukkan jumlah kategori: 2
Masukkan nama kategori:Game
Masukkan 5 nama aplikasi di kategori Game
Nama aplikasi: A
Nama aplikasi: B
Nama aplikasi: c
Nama aplikasi: d
Nama aplikasi: E
Masukkan nama kategori:Sosmed
Masukkan 5 nama aplikasi di kategori Sosmed
Nama aplikasi: A
Nama aplikasi: B
Nama aplikasi: C
Nama aplikasi: D
Nama aplikasi: E
{'Game': ['A', 'B', 'c', 'd', 'E'], 'Sosmed': ['A', 'B', 'C', 'D', 'E']}
[{'d', 'A', 'E', 'B', 'c'}, {'D', 'A', 'E', 'C', 'B'}]
{'B', 'A', 'E'}
Kemunculan aplikasi: {'d': 1, 'A': 2, 'E': 2, 'B': 2, 'c': 1, 'D': 1, 'C': 1}
Aplikasi yang hanya muncul di satu kategori: {'D', 'C', 'd', 'c'}
```

SOAL 2

2.1 SOURCE CODE

```
data_list = [1, 2, 3, 4, 5, 5, 6]
data_set = {7, 8, 9, 10}
data_tuple = (11, 12, 13, 13, 14)

print("Data List sebelum konversi: ", data_list)
konversi_set = set(data_list)
print("Data Set setelah konversi dari List: ", konversi_set)

print("\nData Set sebelum konversi: ", data_set)
konversi_list = list(data_set)
print("Data List setelah konversi dari Set: ", konversi_list)

print("\nData Tuple sebelum konversi: ", data_tuple)
konversi_set_from_tuple = set(data_tuple)
print("Data Set setelah konversi dari Tuple: ", konversi_set_from_tuple)

print("\nData Set sebelum konversi: ", data_set)
konversi_tuple = tuple(data_set)
print("Data Tuple setelah konversi dari Set: ", konversi_tuple)
```

2.2 PENJELASAN

Konverensi bisa dilakukan dengan menggunakan set, list atau, tupple dengan mengambil variabel yang ingin diubah nantinya

```
konversi_set = set(data_list)
konversi_list = list(data_set)
konversi_set_from_tuple = set(data_tuple)
konversi_tuple = tuple(data_set)
```

2.3 HASIL OUTPUT

```
Data List sebelum konversi:  [1, 2, 3, 4, 5, 5, 6]
Data Set setelah konversi dari List:  {1, 2, 3, 4, 5, 6}

Data Set sebelum konversi:  {8, 9, 10, 7}
Data List setelah konversi dari Set:  [8, 9, 10, 7]

Data Tuple sebelum konversi:  (11, 12, 13, 13, 14)
Data Set setelah konversi dari Tuple:  {11, 12, 13, 14}

Data Set sebelum konversi:  {8, 9, 10, 7}
Data Tuple setelah konversi dari Set:  (8, 9, 10, 7)
```

SOAL 3

3.1 SOURCE CODE

```
def main():
    def baca_file(nama_file):
        try:
            with open(nama_file, 'r') as file:
                isi = file.read().lower()
                kata_kata = set(isi.split())
                return kata_kata
        except FileNotFoundError:
            print(f"Error: File '{nama_file}' tidak ditemukan.")
            return None
        except IOError:
            print(f"Error: File '{nama_file}' tidak bisa dibaca.")
            return None

    nama_file1 = input("Masukkan nama file pertama: ")
    nama_file2 = input("Masukkan nama file kedua: ")

    kata_kata_file1 = baca_file(nama_file1)
    kata_kata_file2 = baca_file(nama_file2)

    if kata_kata_file1 is None or kata_kata_file2 is None:
        return

    kata_kata_common = kata_kata_file1.intersection(kata_kata_file2)
    print("Kata-kata yang muncul pada kedua file:")
    for kata in kata_kata_common:
        print(kata)

main()
```

3.2 PENJELASAN

Fungsi baca_file akan Membaca file teks yang diberikan oleh pengguna kemudian mengonversi semua teks menjadi lowercase dan membagi teks menjadi kata-kata dan menyimpannya dalam sebuah set untuk menghilangkan duplikasi.

Kemudian Input Nama File Program meminta pengguna untuk memasukkan nama dua file teks.

Dan pada Membaca File akan Memanggil fungsi baca_file untuk kedua file.

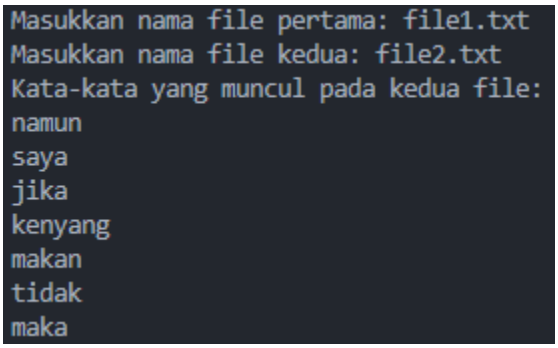
Jika salah satu file tidak ditemukan atau tidak bisa dibaca, program menampilkan pesan error dan keluar.

- Mencari Kata-kata yang Muncul di Kedua File:

  Menggunakan operasi himpunan intersection untuk menemukan kata-kata yang muncul di kedua file.

Program ini memastikan bahwa hanya kata-kata yang muncul di kedua file yang akan ditampilkan, dan menangani kasus di mana file tidak ditemukan atau tidak bisa dibaca dengan menampilkan pesan error yang sesuai.

3.3 HASIL OUTPUT



LINK GITHUB : https://github.com/TomasBeckett/PrakAlpro13.git