10. Tipe Data Dictionary

10.1 Tujuan Praktikum

Setelah mempelajari Bab ini, mahasiswa diharapkan dapat:

1. Dapat menjelaskan tentang dictionary
2. Dapat menggunakan dictionary sebagai model penghitung
3. Dapat membangun dictionary dari file
4. Dapat menggunakan perulangan dari dictionary
5. Dapat melakukan text parsing tingkat lanjut

10.2 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).

10.3 Materi

Dictionary mempunyai kemiripan dengan *list*, tetapi lebih bersifat umum. Dalam *list*, indeks harus berupa integer sedangkan dalam dictionary indeks bisa dapat berupa apapaun.

*Dictionary*memiliki anggota yang terdiri dari pasangan **kunci:nilai**. Kunci harus bersifat unik, tidak boleh ada dua kunci yang sama dalam satu *dictionary*.

Dictionary dapat diartikan pula sebagai pemetaan antara sekumpulan indeks (yang disebut kunci) dan sekumpulan nilai. Setiap kunci memetakan suatu nilai. Asosasi kunci dan nilai tersebut disebut dengan pasangan nilai kunci *(key-value pair)* atau ada yang menyebutnya sebagai *item*.

Sebagai contoh, kita akan mengembangkan kamus yang memetakan dari kata-kata dalam bahasa Inggris ke bahasa Spanyol, jadi bisa dikatakan yang menjadi kunci dan nilai adalah data *string*. Asumsinya setiap kata dalam bahasa Inggris mempunyai satu arti dalam bahasa Spanyol.

Funsgi ***dict*** digunakan untuk membuat *dictionary* baru yang kosong. Karena *dict* merupakan *built-in function* dari python, maka penggunaanya perlu dihindari sebagai nama variabel.

```
>>> ens2sp = dict()
>>> print(eng2sp)
{}
```

Tanda kurung kurawal { } digunakan untuk merepresentasikan *dictionary* kosong. Untuk menambahkan item dalam *dictionary*, dapat menggunakan kurung kotak [ ].

```
>>> eng2sp['one'] = 'uno'
```

Jika kita perhatikan potongan kode diatas, dapat terlihat bahwa baris tersebut membuat item yang memetakan dari kunci **'satu'** ke nilai **'uno'**. Dan jika kita mencetak *dictionary* tadi, maka akan terlihat pasangan nilai kunci antara kunci dan nilai.

```
>>> print(eng2sp)
{'one':'uno'}
```

Format output yang digunakan juga merupakan format input. Misalkan kita membuat *dictionary* daru dengan tiga item. den melakukan pencetakan hasil yang didapatkan berupa keseluruhan data dari *dictionary* tersebut.

```
>>> eng2sp = {'one': 'uno', 'two': 'dos', 'three': 'tres'}
>>> print(eng2sp)
{'one': 'uno', 'three': 'tres', 'two': 'dos'}
```

Pengurutan pasangan kunci-nilai tidaklah sama. Secara umum ururtan item pada *dictionary* tidak dapat diprediksi. Hal ini tidaklah menjadi masalah utama karena elemen dalam *dictionary* tidak pernah diberikan indeks dengan indeks integer. Sebagai gantinya dapat menggunakan kunci untuk mencari nilai yang sesuai *(corresponding values)*.

```
>>> print(eng2sp['two'])
'dos'
```

Kunci **'two'** selalu dimappingkan dengan nilai **"dos"** jadi urutan pada item tidak terlalu berpengaruh. Jika menggunakan kunci yang tidak ada dalam *dictionary*, akan didapatkan pengecualian:

```
>>> print(eng2sp['four'])
KeyError: 'four'
```

Fungsi **len** pada *dictionary* digunakan untuk megembalikan jumlah pasangan nilai kunci:

```
>>> len(eng2sp)
3
```

Operator **in** dalam *dictionary* mengembalikan nilai benar *(true)* atau salah *(false)* sesuai dengan kunci nyang ada dalam *dictionary*.

```
>>> 'one' in eng2sp
True
>>> 'uno' in eng2sp
False
```

Untuk mengetahui nilai yang ada pada *dictionary* dapat menggunakan method **values**. Method ini akan mengembalikan nilau sesuai dengan tipe datanya dan dikonversi kedalam *list* dan digunakan dalam operator *in*.

```
>>> vals = list(eng2sp.values())
>>> 'uno' in vals
True
```

Operator **in** menggunakan algoritma yang berbeda untuk *list* dan *dictionary*. Untuk *list* menggunakan algoritma pencarian linear. Saat *list* bertambah, waktu pencarian yang dibutuhkan menjadi lebih lama secara proprosional sesuai dengan panjang *list*. Dalam *dictionary*, Python menggunakan algoritma yang disebut *Hash Table* dengan properties yang luar biasa, operator **in** membutuhkan waktu yang sama tidak peduli dengan banyak item yang ada didalam *dictionary*.

10.3.1 *Dictionary* sebagai set penghitung *(counters)*

Sebagai contoh kita diberikan sebuah *string* dan di haruskan menghitung banyaknya huruf yang muncul, beberapa cara yang bisa digunakan misalnya :

1. Membuat 26 variable untuk tiap huruf pada alfabet, kemudian memasukkanya dalam *string* untuk masing-masing karakter, menambah perhitungan yang sesuai dan mungkin menggunakan kondisional berantai.
2. Membuat *list* dengan 26 elemen, kemudian nmelakukan konversi setiap karakter menjadi angka (menggunakan fungsi bawaan), menggunakan angka sebagai indeks dalam *list* dan menambah perhitungan yang sesuai.
3. Membuat *dictionary* dengan karakter sebagai kunci dan perhitungan sebagai nilai yang sesuai, pada satu karakter dapat ditambahkan item kedalam *dictionary*. Setelah itu dapat pula ditambahkan nilai dari item yang ada.

Dari 3 opsi diatas, kita akan melakukan perhitungan yang sama, akan tetapi dari masing-masing opsi menerapkan proses perhitungan dengan cara yang berbeda-beda.

Implementasi dilakukan dengan menggunakan perhitungan *(computation)*, beberapa perhitungan biasanya lebih baik dari model perhitungan lainnya. Penggunaan komputasi model *dictionary* lebih praktis, dimana kita tidak perlu mengetahui huruf mana yang akan muncul dalam string. Kita hanya perlu memberikan ruang untuk huruf yang akan muncul. Model penerapannya sebagai berikut:

```
word = 'brontosaurus'
d = dict()
for c in word:
    if c not in d:
        d[c] = 1
    else:
        d[c] = d[c] + 1
print(d)
```

Model komputasi diatas disebut dengan *histogram*, yang mana merupakan istilah statistika dari set perhitungan (atau frekuensi).

Perulangan *for* akan melewati string. Setiap kali terkadi *looping*, jika karakter **c** tidak ada dalam *dictionary* maka akan dibuat item baru dengan kunci **c** dan nilai awal 1 (asumsinya telah muncul sekali). Jika **c** sudah ada dalam *dictionary* secara otomatis akan melakukan penambahan **d(c)**.

Output dari program diatas adalah :

```
{'b': 1, 'r': 2, 'o': 2, 'n': 1, 't': 1, 's': 2, 'a': 1, 'u': 2}
```

Histogram output menunjukkan bahwa huruf "a" dan "b" muncul sekali; "o" muncul dua kali, dan seterusnya.

Dictionary memiliki metode yang disebut get yang mengambil kunci dan nilai default. Jika kunci muncul di *dictionary*, akan mengembalikan nilai yang sesuai; jika tidak maka akan mengembalikan nilai default. Sebagai contoh:

```
>>> counts = { 'chuck' : 1 , 'annie' : 42, 'jan': 100}
>>> print(counts.get('jan', 0))
100
>>> print(counts.get('tim', 0))
0
```

Penggunaan *get* dapat dilakukan untuk menulis loop histogram secara lebih ringkas. Metode *get* secara otomatis menangani case dimana kunci tidak ada dalam *dictionary*. Dengan menghilangkan statement *if*, kita dapat meringkas empat baris menjadi satu baris saja.

```
word = 'brontosaurus'
d = dict()
for c in word:
    d[c] = d.get(c,0) + 1
print(d)
```

dari program diatas adalah :

```
{'b': 1, 'r': 2, 'o': 2, 'n': 1, 't': 1, 's': 2, 'a': 1, 'u': 2}
```

Penggunaan metode *get* untuk menyederhanakan loop penghitungan ini akhirnya menjadi "idiom" yang sangat umum digunakan dalam Python. Jika dibandingkan loop menggunakan pernyataan *if* dan operator *in* dengan loop menggunakan metode *get* melakukan hal yang persis sama, tetapi yang satu lebih ringkas.

10.3.2 *Dictionary* dan File

Salah satu kegunaan umum *dictionary* adalah untuk menghitung kemunculan kata-kata dalam file dengan beberapa teks tertulis. Mari kita mulai dengan file kata yang sangat sederhana yang diambil dari teks Romeo dan Juliet.

Sebagai contoh pertama, akan digunakan versi teks yang disingkat dan disederhanakan tanpa tanda baca. Selanjutnya akan digunakan teks adegan dengan tanda baca yang disertakan.

```
But soft what light through yonder window breaks
It is the east and Juliet is the sun
Arise fair sun and kill the envious moon
Who is already sick and pale with grief
```

Kita akan mencoba memnulis program Python untuk membaca baris-baris file, memecah setiap baris menjadi daftar kata, dan kemudian melakukan perulangan melalui setiap kata dalam baris dan menghitung setiap kata menggunakan *dictionary*.

Asumsikan kita menggunakan dua **for** untuk perulangan. Perulangan bagian luar untuk membaca baris file dan perulangan pada dalam mekalukan iterasi melalui setiap kata pada baris tertentu. Ini adalah contoh dari pola yang disebut nested loop karena salah satu perulangan adalah perulangan bagian luar dan perulangan lainnya adalah perulangan bagian dalam.

Perulangan dalam melakukan eksekusi pada semua iterasi setiap kali perulangan bagian luar membuat iterasi tunggal. Pada bagian ini, perulangan bagian dalam iterasi "lebih cepat", sedangkan perulangan bagian luar iterasi lebih lambat.

Kombinasi dari dua perulangan bersarang memastikan bahwa ada proses perhitugnan setiap kata pada setiap baris file input.

```
fname = input('Enter the file name: ')
try:
    fhand = open(fname)
except:
    print('File cannot be opened:', fname)
    exit()

counts = dict()
for line in fhand:
    words = line.split()
    for word in words:
        if word not in counts:
            counts[word] = 1
        else:
            counts[word] += 1

print(counts)
```

Pada statement **else** digunakan model penulisan yang lebih singkat untuk menambhakan variable. **counts[word] += 1** sama dengan **counts[word] = counts[word] + 1**. Metode lain dapat

digunakan untuk mengubah nilai suatu variabel dengan jumlah yang diinginkan, misalnya dengan menggunakan **- =, * =,** dan **/ =**.

Ketika kita menjalankan program, akan didapatkan data dump jumlah semua dalam urutan hash yang tidak disortir.

file romeo.txt dapat diunduh melalui moodle atau dapat menggunakan menggunakan teks Romeo Juliet pada bagian awal 10.3.2 Dictionary dan File yang kemudian disimpan dengan nama romeo.txt

```
python count1.py
Enter the file name: romeo.txt
{'and': 3, 'envious': 1, 'already': 1, 'fair': 1,
'is': 3, 'through': 1, 'pale': 1, 'yonder': 1,
'what': 1, 'sun': 2, 'Who': 1, 'But': 1, 'moon': 1,
'window': 1, 'sick': 1, 'east': 1, 'breaks': 1,
'grief': 1, 'with': 1, 'light': 1, 'It': 1, 'Arise': 1,
'kill': 1, 'the': 3, 'soft': 1, 'Juliet': 1}
```

10.3.3 *Looping* dan *Dictionary*

Dalam statement ***for***, *dictionary* akan bekerja dengan cara menelusuri kunci yang ada didalamnya. *Looping* ini akan melakukan pecetakan setiap kunci sesuai dengan hubungan nilainya.

```
1 counts = { 'chuck' : 1 , 'annie' : 42, 'jan': 100}
2 for key in counts:
3     print(key, counts[key])
```

Outpunya akan tampak sebagai berikut:

```
jan 100
chuck 1
annie 42
```

Output yang ditampilkan memperlihatkan bahwa kunci tidak berada dalam pola urutan tertentu. Pola ini dapat diimplementasikan dengan berbagai idiom *looping* yang telah dideskripsikan pada bagian sebelumnya. Sebagai contoh jika ingin menemukan semua entri dalam *dictionary* dengan nilai diatas 10, maka kode programnya sebagai berikut :

```
1 counts = { 'chuck' : 1 , 'annie' : 42, 'jan': 100}
2 for key in counts:
3     if counts[key] > 10 :
4         print(key, counts[key])
```

Pada *looping* **for** yang melalui kunci *dictionary*, perlu ditambahkan opertor indeks untuk mengambil nilai yang sesuai untuk setiap kunci. Outputnya akan terlhiat sebagai berikut :

```
jan 100
annie 42
```

Program diatas hanya akan menampilkan data nilai diatas 10.

Untuk melakukan print kunci dalam urutan secara alfabet, langkah pertama yang dilakukan adalah membuat list dari kunci pada *dictionary* dengan menggunakan method kunci yang tersedia pada object *dictionary*. Langkah selanjutnya adalah melakuan pengurutan *(sort)*, *list* dan *loop* melewati *sorted list*. Langkah terakhir dengan melihat tiap kunci dan pencetak pasangan nilai *key* yang sudah diurutkan. Impelentasi dalam kode dapat dilihat dibawah ini:

```
counts = { 'chuck' : 1 , 'annie' : 42, 'jan': 100}
lst = list(counts.keys())
print(lst)
lst.sort()
for key in lst:
    print(key, counts[key])
```

Outputnya sebagai berikut:

```
['jan', 'chuck', 'annie']
annie 42
chuck 1
jan 100
```

Pertama-tama kita melihat list dari kunci dengan tidak berurutan yang didapatkan dari method kunci. Kemudian kita melihat pasangan nilai kunci yang disusun secara berurutan menggunakan loop **for**.

10.3.4 *Advenced Text Parsing*

Pada bagian sebelumnya, kita menggunakan contoh file dimana kalimat pada file teersebut sudah dihilangkan tanda bacanya. Bagian ini kita akan menggukan file dengan tanda baca lengkap. Dengan menggunakan contoh yang sama yaitu romeo.txt dengan tanda baca lengkap.

```
But, soft! what light through yonder window breaks?
It is the east, and Juliet is the sun.
Arise, fair sun, and kill the envious moon,
Who is already sick and pale with grief,
```

Python sendiri mempunyai functoin **split** dimana fungsi tersebut akan mencari spasi dan mengubah kata-kata sebgau token yang dipisahkan oleh spasi. Misal pada kata ***"soft!"*** dan ***"soft"*** merupakan dua kata yang berbeda dan dalam *dictionary* adakn dibuat terpisah untuk tiap kata tersebut.

Hal tersebut juga berlaku pada penggunaan huruf kapital. Misal pada kata ***"who"*** dan ***"Who"*** dianggap sebagai kata berbeda dan dihutng secara terpisah.

Model penyelesaian lain dapat juga dengan menggunakan method *string* lain seperti **lower, punctuation** dan **translate**. Jika diperhatikan method string **translate** merupakan method string yang paling *stuble* (hampir tidak kentara). Berikut ini merupakan dokumentasi dari method **translate**.

```
line.translate(str.maketrans(fromstr, tostr, deletestr))
```

Ganti karakter pada ***fromstr*** dengan karakter pada posisi yang sama dengan ***tostr*** dan hapus semua karakter yang ada dalam ***deletetr***. Untuk ***fromstr*** dan ***tostr*** dapat berupa string kosong dan untuk parameter pada ***deletetr*** dapat dihilangkan.

Kita tidak akan menentukan ***tostr*** tetapi kita akan menggunakan parameter ***deletetr*** untuk menghapus semua tanda baca. Secara otomatis Python akan memberitahu bagian mana yang dianggap sebagai "tanda baca".

```
>>> import string
>>> string.punctuation
'!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
```

Penggunaan dalam program sebagai berikut:

```
import string

fname = input('Enter the file name: ')
try:
    fhand = open(fname)
except:
    print('File cannot be opened:', fname)
    exit()

counts = dict()
for line in fhand:
    line = line.rstrip()
    line = line.translate(line.maketrans('', '', string.punctuation))
    line = line.lower()
    words = line.split()
    for word in words:
        if word not in counts:
            counts[word] = 1
        else:
            counts[word] += 1

print(counts)
```

Output dari program diatas adalah :

```
Enter the file name: romeo-full.txt
{'swearst': 1, 'all': 6, 'afeard': 1, 'leave': 2, 'these': 2,
'kinsmen': 2, 'what': 11, 'thinkst': 1, 'love': 24, 'cloak': 1,
a': 24, 'orchard': 2, 'light': 5, 'lovers': 2, 'romeo': 40,
'maiden': 1, 'whiteupturned': 1, 'juliet': 32, 'gentleman': 1,
'it': 22, 'leans': 1, 'canst': 1, 'having': 1, ...}
```

10.4 Kegiatan Praktikum

Kasus 10.1 Buatlah sebuah program yang dapat melakukan generate dan mentecak dictionary yang berisi angka antara **1** sampai **n** dalam bentuk **(x,x*x)**

```
Contoh:
Input Data = 5
Dcitionary = {1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
```

Pembahasan Kasus 1

Untuk dapat menyelesaikan kasus pertama, langkah-langkah yang menjadi penyelesaiannya adalah :

1. Membuat dan menentukan panjang dictionary nya.
2. Buat perulangan sepanjang dictionary
3. Input dictionary dengan **key = x** dan **value = x*x.**

Berikut ini contoh programnya:

```
1
2 n= int(input("Input data = "))
3 kamus = dict()
4
5 for x in range(1,n+1):
6     kamus[x]=x*x
7
8 print("Dictionary = ", kamus)
9
```

Perhatikan outputnya !!

Kasus 10.2 Buatlah program untuk mencetak semua nilai *(value)* unik yang ada didalam dictionary.

```
Contoh:
Data : [{"V":"S001"}, {"V": "S002"}, {"VI": "S001"}, {"VI": "S005"},
{"VII":"S005"}, {"V":"S009"},{"VIII":"S007"}]
Output : Unique Values: {'S005', 'S002', 'S007', 'S001', 'S009'}
```

Pembahasan Kasus 2

Untuk dapat menyelesaikan kasus kedua, kita dapat menggunakan fungsi *values* pada dictionary. untuk mencari nilau unik kita ambil masing-masing nilai dari anggota dictionary, kemudiah kita kumpulkan dalam satu variabel.

Berikut ini contoh programnya:

```
1
2 data = [{"V":"S001"}, {"V": "S002"}, {"VI": "S001"}, {"VI": "S005"},
3 {"VII":"S005"}, {"V":"S009"},{"VIII":"S007"}]
4 print("Data asli: ",data)
5 nilai_unik = set( val for dic in data for val in dic.values())
6 print("Nilai Unik: ",nilai_unik)
7
```

Perhatikan outputnya !!

Kasus 10.3 Dengan menggunakan file **words.txt**, buatlah program untuk menympan kunci (keys) pada dictionary. Tambahkan pengecekan apakah suatu kata yang diinputkan ada didalam daftar tersebut. Jika ada silakan cetak kata ditemukan, jika tidak ada silakan cetak kata tidak ditemukan.

Silakan cek bagian dibawah ini untuk contoh output dari programnya.

```
Masukkan nama file : words.txt
Kata yang dicari : programming

Daftar Kamus
{'Writing': 1, 'programs': 2, 'or': 3, 'programming':
4, 'is': 5, 'a': 6, 'very': 7, 'creative': 8, 'and':
9, 'rewarding': 10, 'activity': 11, 'You': 12,
'can': 13, 'write': 14, 'for': 16, 'many': 17,
'reasons': 18, 'ranging': 19, 'from': 20, 'making':
21, 'your': 22, 'living': 23, 'to': 24, 'solving':
25, 'difficult': 27, 'data': 28, 'analysis': 29, }

Kata programming ditemukan dalam kamus
```

Pembahasan Kasus 3

Untuk dapat membuat program diatas, langkah-langkah yang harus dilakukan adalah :

1. Buat program untuk membaca file txt.
2. Input kata yang dicari.
3. Buat dictionary baru untuk menyimpan file yang sudah dibaca.
4. simpan file dalam dictionary, jangan lupa cek duplikasinya.
5. Cetak dictionary nya.
6. Lakukan pengecekan terhadap kata yang diinputkan.
7. Cetak hasilnya.

Berikut adalah kode programnya.

```
1
2  count = 0
3  dictionary_words = dict()
4  fname = input('Masukkan nama file : ')
5  fword = input('Kata yang dicari : ')
6  try:
7      fhand = open(fname)
8  except FileNotFoundError:
9      print('Fie tidak bisa dibuka !!', fname)
10     exit()
11
12 for line in fhand:
13     words = line.split()
14     for word in words:
```

```
        count += 1
            if word in dictionary_words:
                continue                            # cek duplikasi
            dictionary_words[word] = count          # kata yg pertama muncul

print('\nDaftar Kamus : \n')
print(dictionary_words)

if fword in dictionary_words:
    print('\nKata %s ditemukan dalam kamus' % fword)
else:
    print('\nKata %s tidak ditemukan dalam kamus' % fword)


```

Silahkan lihat hasil programnya!

Kasus 10.4 Buat program yang dapat membagi kategori dari setiap email yang masuk dimana pada hari itu perintah **commit** dilakukan. Kemudian hitunglah berapa banyak hari dimana perintah **commit** itu sering dilakukan. Gunakan file mbox-short.tx untuk mendapatkan kategori dari setiap email yang masuk tersebut.

Pembahasan Kasus 4

Untuk dapat menyelesaikan kasus, langkah-langkah yang harus dilakukan adalah :

1. Cek pola yang ada
2. Buat program untuk membaca file txt.
3. Buat dictionary
4. Cari baris yang dimulai dengan **"From"**
5. Cari kata ketiga, lakukan perhitungan berjalan untuk menghitung setiap hari dalam 1 minggu.
6. Simpan dalam dictionary.
7. Cetak dictionary nya.

Untuk melakukan pengecekan pola, silakan buka file mbox-short.txt. File tersebut merupakan standart format yang berisi *multiple mail message*. Baris pertama dimulai dengan ***"From"*** untuk memisahkan antar *message*. Informasi mengenai format mbox silakan cek https://en.wikipedia.org/wiki/Mbox.

```
Pengecekan pola
Baris contoh :
From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008
Program:
python kasus4.py
Enter a file name: mbox-short.txt
{'Fri': 20, 'Thu': 6, 'Sat': 1}
```

Berdasarkan pengecekan yang telah kita lakukan diatas, dapat dibuat kode programmnya sebagai berikut:

```

dictionary_days = dict()                        # dictionary baru
fname = input('Enter a file name: ')
try:
```

```
    fhand = open(fname)
except FileNotFoundError:
    print('File cannot be opened:', fname)
    exit()

for line in fhand:
    words = line.split()
    if len(words) < 3 or words[0] != 'From':
        continue
    else:
        if words[2] not in dictionary_days:
            dictionary_days[words[2]] = 1        # pertama
        else:
            dictionary_days[words[2]] += 1       # +1

print(dictionary_days)

```

Silahkan lihat hasil programnya!

10.5 Latihan Mandiri

Latihan 10.1 Bualah sebuah program untuk mendapatkan nilai key, value, dan item dari sebuah dictionary.

```
Contoh:
Dictionary : {1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}

Output:
key     value       item
1       10          1
2       20          2
3       30          3
4       40          4
5       50          5
6       60          6
```

■

Latihan 10.2 Bualah sebuah program untuk memetakan dua list mejadi satu dictionary.

```
Contoh:

Data List
Lista = ['red', 'green', 'blue']
Listb = ['#FF0000','#008000', '#0000FF']

Output
{'green': '#008000', 'blue': '#0000FF', 'red': '#FF0000'}
```

■

Latihan 10.3 Dengan menggunakan file **mbox-short.txt**, buatlah program yang dapat membaca log email dan sajikan dalam histogram menggunakan dictionary. Kemudian hitung berapa banyak pesan yang masuk dari email dan sajikan dalam bentuk dictionary.

Silakan cek bagian dibawah ini untuk contoh output dari programnya.

```
Masukkan nama file : mbox-short.txt

{'gopal.ramasammycook@gmail.com': 1, 'louis@media.berkeley.edu': 3,
'cwen@iupui.edu': 5, 'antranig@caret.cam.ac.uk': 1,
'rjlowe@iupui.edu': 2, 'gsilver@umich.edu': 3,
'david.horwitz@uct.ac.za': 4, 'wagnermr@iupui.edu': 1,
'zqian@umich.edu': 4, 'stephen.marquard@uct.ac.za': 2,
'ray@media.berkeley.edu': 1}
```

■

Latihan 10.4 Dengan menggunakan file **mbox-short.txt**, buat program untuk mencatat data nama domain pengirim pesan. Hitunglah jumlah pesan yang dikirim masing-masing domain. sajikan dalam bentuk dictionary.

Silakan cek bagian dibawah ini untuk contoh output dari programnya.

```
Masukkan nama file: mbox-short.txt

{'media.berkeley.edu': 4, 'uct.ac.za': 6, 'umich.edu': 7,
'gmail.com': 1, 'caret.cam.ac.uk': 1, 'iupui.edu': 8}
```

■