Laporan Praktikum Algoritma dan Pemrograman

Semester Genap 2023/2024

NIM	**71230978**
Nama Lengkap	**Jonathan Satriani Gracio Andrianto**
Minggu ke / Materi	**11 / Tipe Data Dictionary**

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

Dictionary merupakan tipe data yang berguna dalam menyimpan data dalam bentuk pasangan kunci-nilai (key-value pairs). Setiap elemen dalam dictionary terdiri dari kunci yang bersifat unik dan nilai yang terkait dengan kunci tersebut. Berbeda dengan list yang indexnya harus integer, dalam dictionary index dapat berupa apapun.

Sebagai contoh, kita akan membuat kamus yang memetakan kata-kata dari Bahasa Indonesia ke Bahasa Inggris.

```
in2en = dict()
print(in2en) #{}
```

Fungsi ‘dict’ digunakan untuk membuat dictionary baru yang direpresentasikan dengan ‘{}’. Karena ‘dict’ merupakan fungsi bawaan dari python maka ‘dict’ harus dihindari untuk digunakan sebagai nama variable.

```
in2en['satu'] = 'one'
print(in2en) #output = {'satu': 'one'}
```

Pada potongan kode diatas bisa dilihat bahwa kita telah membuat item yang memetakan dari key ‘satu’ ke value ‘one’. Dan jika kita cetak dictionary tersebut, maka outputnya akan berupa pasangan antara key dan value. Kita juga dapat menginput nilai ke dictionary menggunakan cara dibawah.

```
in2en = {'satu' : 'one', 'dua' : 'two', 'tiga' : 'three'}
print(in2en) #output = {'satu': 'one', 'dua': 'two', 'tiga': 'three'}
```

Dikarenakan pada dictionary urutan pasangan key-value tidak sama, maka untuk mencari nilai yang sesuai kita dapat menggunakan kunci atau key. Contohnya bisa dilihat dibawah.

```
print(in2en['tiga']) #output = three
```

Jika menggunakan kunci yang tidak ada dalam dictionary :

```
print(in2en['empat']) #output = KeyError: 'empat'
```

Pada dictionary kita dapat menggunakan fungsi ‘len’ untuk mendapat jumlah pasangan key-value :

```
print(len(in2en)) #output = 3
```

Operator ‘in’ juga dapat digunakan di dalam dictionary tetapi hanya sesuai dengan key atau kuncinya. Bisa dilihat di contoh dibawah :

```
print('satu' in in2en) #output = True
print('one' in in2en) #output = False
```

Untuk menggunakan nilai atau value, kita dapat mengubah semua value dari dictionary tersebut menjadi sebuah list kemudian kita gunakan operator ‘in’. Contohnya dibawah :

```
val = list(in2en.values())
print('one' in val) #output = True
```

Operator 'in' menggunakan algoritma yang berbeda untuk list dan dictionary. Untuk list, ia menggunakan algoritma pencarian linear. Ketika elemen dalam list bertambah, waktu pencarian akan meningkat seiring dengan panjang list tersebut. Namun, dalam dictionary, Python menggunakan algoritma yang dikenal sebagai Hash Table dengan sifat yang sangat efisien. Sebagai hasilnya, operator 'in' membutuhkan waktu yang konstan untuk memprosesnya, tidak peduli seberapa banyak item yang ada dalam dictionary tersebut.

Dictionary sebagai set penghitung (counters)

Misalnya, ketika kita diberi sebuah string dan diminta untuk menghitung jumlah kemunculan setiap huruf di dalamnya, beberapa metode yang dapat digunakan antara lain :

1. Membuat 26 variabel yang mewakili setiap huruf dalam alfabet, kemudian menyimpannya dalam string untuk tiap karakter, danmenambah jumlahnya sesuai dengan kemunculannya, serta menggunakan serangkaian kondisional berantai.
2. Membuat sebuah list dengan 26 elemen, kemudian mengonversi setiap karakter menjadi angka menggunakan fungsi bawaan, kemudian menggunakan angka tersebut sebagai indeks dalam list, dan menambah jumlah yang sesuai.
3. Membuat sebuah dictionary dengan karakter sebagai kunci dan jumlah kemunculannya sebagai nilai yang sesuai. Karakter baru dapat ditambahkan sebagai kunci ke dalam kamus dan jumlahnya diperbarui sesuai dengan kemunculannya.

Meski semua cara tersebut memberikan hasil yang sama dalam menghitung kemunculan huruf, menggunakan model dictionary dianggap lebih praktis karena tidak perlu mengetahui sebelumnya huruf mana yang akan muncul dalam string. Dengan metode ini, kita hanya perlu menyiapkan ruang untuk huruf yang mungkin muncul, dan dictionary akan secara otomatis menangani setiap kemunculan huruf tersebut. Kodenya bisa dilihat dibawah :

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

```
        d[c] = d[c] + 1
print(d) #output = {'j': 1, 'o': 1, 'd': 1, 'a': 3, 'h': 1, 'g': 3, 'n': 3, 't':
2, 'e': 2, 'b': 1}
```

Dalam perulangan for, setiap kali iterasi melewati string, jika karakter c tidak terdapat dalam kamus, maka sebuah item baru akan dibuat dengan kunci c dan nilai awal 1 (diasumsikan muncul sekali). Jika karakter c sudah ada dalam kamus, maka nilai d[c] akan ditambah secara otomatis.

Dictionary juga memiliki sebuah metode yang disebut 'get' yang menerima sebuah kunci dan nilai default. Jika kunci tersebut ditemukan dalam kamus, metode ini akan mengembalikan nilai yang sesuai dengan kunci tersebut; namun, jika kunci tidak ditemukan, maka akan mengembalikan nilai default. Contoh kodenya :

```
counts = { 'lex' : 4 , 'anton' : 1, 'nev': 69}
print(counts.get('nev', 0)) #output = 69
print(counts.get('efra', 0)) #output = 0
```

Dictionary dan File

Dictionary biasanya digunakan untuk menghitung kemunculan kata-kata dalam file dengan beberapa teks tertulis. Kita akan mulai dengan file kata yang diambil dari teks Romeo and Juliet. Pada contoh pertama ini kita akan menggunakan versi teks yang disingkat dan yang tanpa tanda baca.

Asumsikan kita menggunakan dua perulangan for. Perulangan luar digunakan untuk membaca setiap baris dalam file, sedangkan perulangan dalam digunakan untuk mengiterasi melalui setiap kata dalam baris tertentu. Ini adalah contoh dari apa yang disebut sebagai nested loop, di mana satu perulangan berada di dalam perulangan lainnya.

Perulangan dalam memproses semua iterasi setiap kali perulangan luar membuat iterasi luar. Jadi pada kasus ini, perulangan dalam lebih cepat daripada perulangan luar. Kodenya bisa dilihat dibawah :

```
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)
'''output = {'But': 1, 'soft': 1, 'what': 1, 'light': 1, 'through': 1,
'yonder': 1, 'window': 1, 'breaks': 1, 'It': 1, 'is': 3, 'the': 3, 'east': 1,
'and': 3, 'Juliet': 1, 'sun': 2, 'Arise': 1, 'fair': 1, 'kill': 1, 'envious': 1,
'moon': 1, 'Who': 1, 'already': 1, 'sick': 1, 'pale': 1, 'with': 1, 'grief':
1}'''
```

 # counts[word] += 1 sama dengan counts[word] = counts[word] + 1

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

Looping dan Dictionary

Dalam statement for, dictionary akan menelusuri kucni atau key yang didalamnya. Perulangan ini akan mencetak semua kunci sesuai dengan hubungan nilanya.

```
counts = { 'lex' : 4 , 'anton' : 1, 'nev': 69}
for key in counts:
    print(key, counts[key])
'''
output =
lex 4
anton 1
nev 69
'''
```

Output menunjukkan bahwa kunci tidak mengikuti pola urutan tertentu. Pola ini bisa diwujudkan melalui berbagai cara dalam pengulangan yang telah dijelaskan sebelumnya. Sebagai contoh, jika ingin menemukan semua entri dalam kamus dengan nilai di atas 1, kode programnya dapat seperti berikut:

```
counts = { 'lex' : 4 , 'anton' : 1, 'nev': 69}
for key in counts:
    if counts[key] > 1 :
        print(key, counts[key])
'''
output =
lex 4
nev 69
'''
```

Program diatas hanya akan menampilkan data nilai diatas 1.

Untuk menampilkan key atau 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 = { 'lex' : 4 , 'anton' : 1, 'nev': 69}
lst = list(counts.keys())
print(lst) #output = ['lex', 'anton', 'nev']
lst.sort()
for key in lst:
    print(key, counts[key])
'''
output =
anton 1
lex 4
nev 69
'''
```

Pada perintah print pertama, kita bisa melihat kunci-kuncinya belum berurutan. Kemudian kita lihat nilai kunci yang disusun berurutan dengan loop for.

Advanced Text Parsing

Pada bagian Dictionary dan File, kita sudah menggunakan contoh file dimana pada file tersebut sudah dihilangkan tanda bacanya. Pada bagian ini kita akan menggunakan file dengan tanda baca yang lengkap.

Pada python terdapat function ‘split’ yang bisa mencari spasi dan mengubah kata-kata sebagai token yang dipisah oleh spasi. Misal kata ‘soft!’ dan ‘soft’ merupakan dua kata yang berbeda dan akan dipisah dalam dictionary.

Hal tersebut berlaku juga pada penggunaan huruf kapital. Misal kata ‘who’ dan ‘Who’ yang dianggap sebagai dua kata yang berbeda.

Selain menggunakan fungsi `split()`, ada juga pendekatan lain yang dapat digunakan, seperti menggunakan metode string lain seperti `lower()`, `punctuation`, dan `translate()`. Di antara metode-metode tersebut, metode `translate()` dapat dianggap sebagai metode string yang paling halus (paling tidak terlihat). Berikut adalah dokumentasi untuk metode 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"

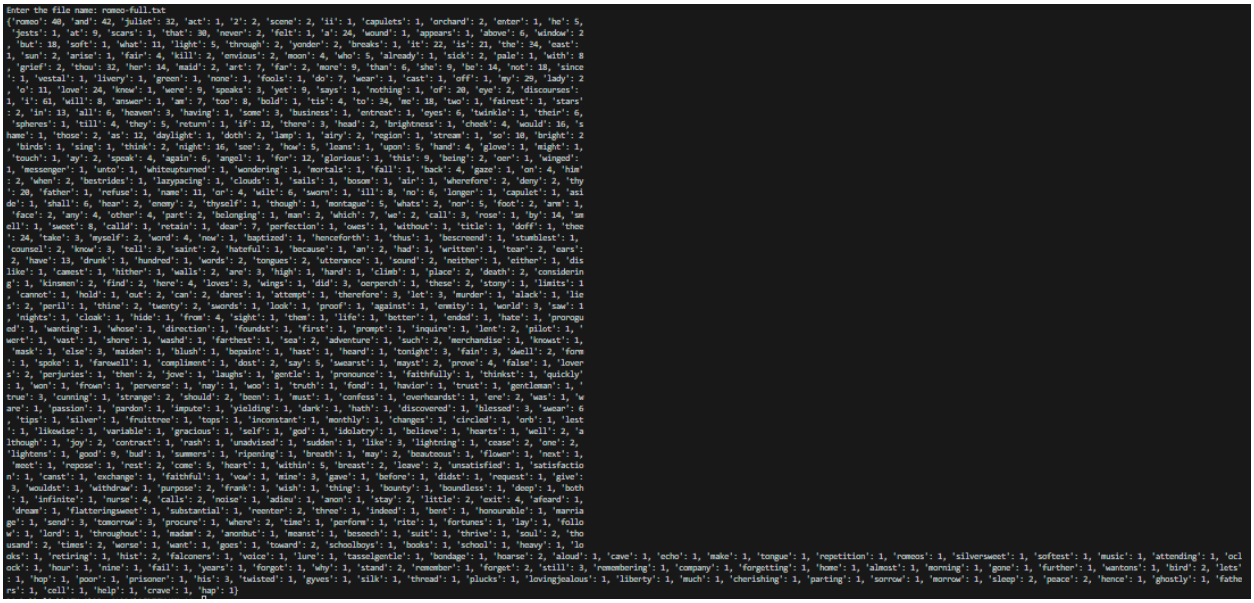
```
import string

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 =



Kegiatan Praktikum

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

```
n= int(input("Input data = ")) #misal saya input = 5
kamus = dict()

for x in range(1,n+1):
    kamus[x]=x*x

print("Dictionary = ", kamus) #output = Dictionary =  {1: 1, 2: 4, 3: 9,
4: 16, 5: 25}
```

Dalam program ini, kita membuat dictionary 'kamus' menggunakan dictionary comprehension, yang memungkinkan pembuatan dictionary dengan cara yang ringkas hanya dalam satu baris kode.

B. Buatlah program untuk mencetak semua nilai (value) unik yang ada didalam dictionary.

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

output

```
Data asli:  [{'V': 'S001'}, {'V': 'S002'}, {'VI': 'S001'}, {'VI': 'S005'}, {'VII': 'S005'}, {'V': 'S009'}, {'VIII': 'S007'}]
Nilai Unik:  {'S007', 'S002', 'S005', 'S009', 'S001'}
```

Kita buat satu variable untuk menampung semua value dari dictionary 'data'.

C. 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.

```
count = 0
dictionary_words = dict()
fname = "words.txt"
fword = "programming"
try:
    fhand = open(fname)
except FileNotFoundError:
    print('Fie tidak bisa dibuka !!', fname)
    exit()

for line in fhand:
    words = line.split()
```

```
    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)
```

output

```
Daftar Kamus :

{'programming': 1}

Kata programming ditemukan dalam kamus
```

BAGIAN 2: LATIHAN MANDIRI (60%)

SOAL 1

Source Code

```
kamus = dict()
kamus = {1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}
i = 1
print("key", "value", "item")
for key in kamus:
    print(key, "   ", kamus[key], "   ", i)
    i += 1
```

Output

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

Penjelasan

Pertama kita buat dictionary kosong menggunakan ‘dict()’, dan kita isi nilainya sesuai dengan contoh dari soal. Kita buat variable baru yang nantinya akan dicetak sebagai item. Kemudian kita cetak header

yang berisi key, value dan item. Selanjutnya kita buat perulangan for yang akan mencetak key, value dari key tersebut, dan variable i yang barusan kita buat. Saya menggunakan spasi didalam fungsi printnya agar ketiga nilai tersebut dapat terlihat rapi dan segaris dengan header. Dan terakhir kita buat 'i'nya bertambah 1 setiap iterasi.

SOAL 2

Source Code

```
Lista = ['red', 'green', 'blue']
Listb = ['#FF0000','#008000', '#0000FF']
kamus_bos = dict()

for i in range (0, len(Lista)) :
    kamus_bos[Lista[i]] = Listb[i]
print(kamus_bos)
```

Output

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

Penjelasan

Pertama kita buat lista dan listb, dan kita buat dictionary kosong. Kemudian kita buat perulangan for dengan startnya = 0 (karena nanti kita akan gunakan 'i' sebagai index), dan stop = Panjang dari lista. Kemudian untuk mengisi dictionary, kita gunakan keynya yaitu lista[i] yang berarti key dari dictionary tersebut adalah nilai dalam lista, valuenya kita isi dengan nilai dalam listb.

SOAL 3

Source Code

```
hasil = dict()

with open ("mbox-short.txt", "r") as file :
    for line in file :
        if line.startswith('From '):
            email = line.split()[1]
            hasil[email] = hasil.get(email, 0) + 1

print(hasil)
```

Output

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

Penjelasan

Pertama kita buat dictionary kosong untuk menyimpan email dan jumlah munculnya. Kemudian kita buka file mbox-short.txt dengan mode read. Selanjutnya kita buat perulangan for yang melihat setiap baris dari file mbox-short, dan jika ada yang berawalan 'From ' maka akan diproses. Baris kemudian dipisah menggunakan 'split()' dan email hasil pisahan tersebut akan dimasukan ke dalam dictionary dengan value yang akan terus berubah jika ada key yang sama.

SOAL 4

Source Code

```
hasil = dict()

with open ("mbox-short.txt", "r") as file :
    for line in file :
        if line.startswith('From '):
            email = line.split()[1]
            email_ = email.split("@")[-1]
            hasil[email_] = hasil.get(email_, 0) + 1

print(hasil)
```

Output

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

Penjelasan

Kode ini sama seperti pada soal nomor 3, tetapi bedanya sebelum pemasukan key dan value ke dalam dictionary, kita buat variable baru yang berisi hasil split("@") dari email. Kita gunakan hasil pisahan email karena yang diminta di soal adalah domainnya saja. Kemudian variable baru tersebut kita masukan ke dalam dictionary sebagai key dan valuenya dapat bertambah ketika ada domain pada file yang sama.

Link Github

https://github.com/JonathanAndrianto123/laporan_alpro_11_71230978.git