
# Laporan Praktikum Algoritma dan Pemrograman

Semester Genap 2023/2024

| | |
|---|---|
| **NIM** | **71230976** |
| **Nama Lengkap** | **Galih Pramana Chandra Prasetya** |
| **Minggu ke / Materi** | **10 / 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%)

Pada bagian ini, tuliskan kembali semua materi yang telah anda pelajari minggu ini. Sesuaikan penjelasan anda dengan urutan materi yang telah diberikan di saat praktikum. Penjelasan anda harus dilengkapi dengan contoh, gambar/ilustrasi, contoh program (source code) dan outputnya. Idealnya sekitar 5-6 halaman.

## Dictionary

**Dictionary** dapat diartikan sebagai pemetaan antara sekumpulan indeks dan sekumpulan nilai. Dictionary mempunyai kemiripan dengan list, tetapi dictionary lebih besifat umum. Dictionary memiliki anggota yang terdiri dari pasangan **kunci:nilai**. Kunci ini bersifat unik, artinya tidak boleh ada dua kunci yang sama dalam satu dictionary.

Setiap kunci memetakan suatu nilai. Asosasi kunci dan nilai tersebut disebut dengan pasangan nilai kunci *(key-value pair)* atau ada yang menyebutnya sebagai *item.*

Fungsi **dict()** dalam python (sudah built-in function maka perlu dihindari sebagai nama variable )digunakan untuk membuat dictionary baru yang kosong. Kemudian menggunakan tanda kurung kurawal { } untuk merepresentasikan dictionary yang kosong. Untuk menambahkan item dalam dictionary, dapat menggunakan kurung kotak [ ].

```
eng2sp = {}
print(eng2sp)
#Dictionary kosong

eng2sp = {'one': 'uno', 'two': 'dos', 'three': 'tres', 'galih':'rain'}
print(eng2sp)
#hanya bisa print base on key
```

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

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

Secara umum urutan item pada dictionary tidak dapat diprediksi. karena elemen dalam dictionary tidak pernah diberikan indeks dengan indeks integer, maka hal ini tidak akan menjadi masalah. Kita dapat menggunakan kunci untuk mencari nilai yang sesuai (corresponding values).

```
eng2sp = {'one': 'uno', 'two': 'dos', 'three': 'tres', 'galih':'rain'}
print(eng2sp['two'])
# dos
```

Kunci ’two’ selalu dipasangkan dengan nilai "dos" jadi urutan pada item tidak terlalu berpengaruh. Jika menggunakan kunci yang tidak ada dalam dictionary, akan menghasilkan error.

```
eng2sp = {'one': 'uno', 'two': 'dos', 'three': 'tres', 'galih':'rain'}
print(eng2sp['four'])
# KeyError: 'four'
```

Fungsi **len** pada dictionary dapaat digunakan untuk menampilkan jumlah pasangan nilai kunci,

```
eng2sp = {'one': 'uno', 'two': 'dos', 'three': 'tres', 'galih':'rain'}
len(eng2sp)
print(len(eng2sp))
# 4
```

Pasangan nilai kunci di atas adalah 4 pasang, maka program akan menampilkan angka 4 sebagai jumlah pasangan nilai kunci yang ada.

Operator **in** dalam dictionary digunakan untuk menampilkan nilai benar (**true**) atau salah (**false**) sesuai dengan kunci yang ada dalam dictionary.

```
eng2sp = {'one': 'uno', 'two': 'dos', 'three': 'tres', 'galih':'rain'}

'galih' in eng2sp
print('galih' in eng2sp)
#Output : true
'one'in eng2sp
print('one' in eng2sp)
#Output : true
'four'in eng2sp
print('four' in eng2sp)
#Output : false
```

Method **values** dapat digunakan untuk mengetahui nilai yang ada pada dictionary. Method ini akan mengembalikan nilai sesuai dengan tipe datanya. Untuk menggunakan nya kita perlu mengubah dictionary menjadi bentuk **list** terlebih dahulu.

```
#panggil value harus ubah jadi list dulu
eng2sp = {'one': 'uno', 'two': 'dos', 'three': 'tres', 'galih':'rain', 'hallo' :
'haii'}
vals = list(eng2sp.values())
'haii' in vals
print('haii' in vals)
#Output : true
```

Operator "in" menggunakan algoritma yang berbeda untuk list dan dictionary. Untuk list, digunakan algoritma pencarian linear. Saat list bertambah, waktu yang dibutuhkan untuk pencarian juga bertambah sesuai dengan panjang listnya. Pada dictionary, Python menggunakan algoritma yang disebut Hash Table. Operator "in" dalam dictionary membutuhkan waktu yang konsisten untuk diproses, tanpa memperdulikan jumlah item yang ada di dalamnya.

## Dictionary sebagai set penghitung (counters)

Sebagai contoh, kita dapat menggunakan beberapa cara untuk menghitung jumlah kemunculan setiap huruf dalam sebuah string:

1. Dengan menggunakan 26 variabel untuk setiap huruf dalam alfabet, kemudian iterasi melalui string untuk masing-masing karakter, menambah jumlah yang sesuai, dan menggunakan kondisi berantai.
2. Dengan membuat sebuah list dengan 26 elemen, kemudian mengonversi setiap karakter menjadi angka (menggunakan fungsi bawaan), menggunakan angka sebagai indeks dalam list, dan menambah jumlah yang sesuai.
3. Dengan membuat sebuah dictionary menggunakan karakter sebagai kunci dan jumlah kemunculan sebagai nilai, kemudian menambahkan setiap karakter ke dalam dictionary dan menambah nilai yang sesuai dari item yang sudah ada.

Penggunaan model dictionary lebih praktis karena kita tidak perlu mengetahui terlebih dahulu huruf mana yang akan muncul dalam string. Kita hanya perlu menyediakan ruang untuk setiap huruf yang mungkin muncul. Berikut ini adalah model penerapannya,

```
word = 'Transformers'
d = dict()
for c in word:
 if c not in d:
    d[c] = 1
 else:
    d[c] = d[c] + 1
print(d)
#{'T': 1, 'r': 3, 'a': 1, 'n': 1, 's': 2, 'f': 1, 'o': 1, 'm': 1, 'e': 1}
```

Model komputasi di atas dikenal sebagai histogram, yang merupakan istilah statistika yang digunakan untuk menggambarkan distribusi frekuensi dari sebuah data. Dalam konteks penghitungan karakter dalam sebuah string, histogram ini mencatat jumlah kemunculan setiap karakter sebagai frekuensinya

Dictionary menyediakan sebuah metode yang disebut get yang berfungsi untuk mengambil nilai berdasarkan kunci yang ditentukan, dengan tambahan opsi untuk menentukan nilai default. Jika kunci tersebut ada dalam dictionary, maka nilai yang sesuai akan dikembalikan; namun, jika kunci tidak ditemukan, metode ini akan mengembalikan nilai default. Sebagai contoh:

```
counts = { 'chuck' : 1 , 'annie' : 42, 'jan': 100, 'donald' : 34}
print(counts.get('jan', 0))
# 100
print(counts.get('donald', 0))
#34
print(counts.get('tim', 0))
# 0
```

Penggunaan metode get dapat mempermudah penulisan loop histogram. Metode get secara otomatis menangani situasi di mana kunci tidak ada dalam dictionary. Dengan menghapus pernyataan if, kita dapat menyederhanakan empat baris menjadi hanya satu baris.

```
word = 'transformers'
d = dict()
for c in word:
    d[c] = d.get(c,0) + 1
print(d)
# {'t': 1, 'r': 3, 'a': 1, 'n': 1, 's': 2, 'f': 1, 'o': 1, 'm': 1, 'e': 1}
```

## Dictionary dan File

Salah satu kegunaan umum dictionary adalah untuk menghitung kemunculan kata-kata dalam file dengan beberapa teks tertulis.

Sebagai contoh, kita dapat mencoba menulis program untuk membaca baris-baris file ‘romeo.txt’. Kita akan memecah setiap baris menjadi daftar kata dan kemudia melakukan perulangan melalui setiap kata dalam baris dan menghitung setiap kata menggunakan dictionary.

```
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()
    word = line.lower()
    for word in words:
        if word not in counts:
            counts[word] = 1
        else:
            counts[word] += 1
print(counts)
# Enter the file name: romeo.txt
```

```
{'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}
```

## Looping dan Dictionary

Dalam statement for, dictionary akan melakukan iterasi melalui setiap kunci yang ada di dalamnya. Looping ini akan mencetak setiap kunci sesuai dengan nilainya yang terkait.

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

jika ingin menemukan semua entri dalam dictionary dengan nilai diatas 10, maka kode programnya sebagai berikut :

```
counts = { 'chuck :' : 1 , 'annie :' : 42, 'jan :': 100}
for key in counts:
    if counts[key] > 10 :  #Dipakai jika hanya ada integer
        print(key, counts[key])
# annie : 42
  jan : 100
```

## Advenced Text Parcing

Python memiliki fungsi split yang membagi string berdasarkan spasi dan menghasilkan token yang dipisahkan oleh spasi. Misalnya, kata "hard!" dan "hard" akan dianggap sebagai dua kata yang berbeda, dan keduanya akan dianggap sebagai token terpisah dalam dictionary.

Prinsip yang sama juga berlaku untuk penggunaan huruf kapital. Misalnya, kata "Halo" dan "halo" dianggap sebagai dua kata yang berbeda dan akan dihitung secara terpisah dalam dictionary.

Metode penyelesaian lain dapat juga menggunakan metode string lain seperti **lower**, **punctuation**, dan **translate**. Diantara metode tersebut, metode string translate cenderung menjadi yang paling halus (subtle), yang berarti perubahannya hampir tidak terlihat.

**string.punctuation** adalah sebuah metode string yang disediakan oleh modul string dalam Python. Metode ini berisi kumpulan karakter tanda baca yang umum digunakan dalam teks, seperti tanda titik, koma, tanda seru, tanda tanya, dan sebagainya. **string.punctuation** ini dapat digunakan untuk menghapus tanda baca dari teks tersebut.

```
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)
```

# BAGIAN 2: LATIHAN MANDIRI (60%)

Pada bagian ini anda menuliskan jawaban dari soal-soal Latihan Mandiri yang ada di modul praktikum. Jawaban anda harus disertai dengan source code, penjelasan dan screenshot output.

## SOAL 1

Source Code:

```
#latman 10.1
dictionary = {1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}
print("Key   Value   Item")
for item_num, (key, value) in enumerate(dictionary.items(), start=1):
    print(f"{key}     {value}     {item_num}")
```

Output:

```
Key   Value   Item
1     10      1
2     20      2
3     30      3
4     40      4
5     50      5
6     60      6
```

Penjelasan:

- membuat dictionary dengan beberapa pasangan kunci-nilai yang telah ditentukan.
- mencetak header yang menjelaskan setiap kolom yang akan dicetak, yaitu "Key", "Value", dan "Item".
- Dalam loop for, fungsi enumerate() digunakan untuk menghasilkan nomor item, dimulai dari 1. enumerate(dictionary.items(), start=1) memberikan pasangan kunci dan nilai dari dictionary, serta nomor item yang sesuai.
- Setiap iterasi loop for mengambil kunci dan nilai dari setiap item dalam dictionary menggunakan unpacking (key, value).
- Pada setiap iterasi, nilai kunci, nilai, dan nomor item dicetak dengan menggunakan f-string format.

## SOAL 2

Source code:

```
#latman 10.2
lista = ["red", "green", "blue"]
listb = ['#FF0000', '#008000', '#0000FF']

x = dict(zip(lista, listb))
print(x)
```

Output:

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

Penjelasan:

- Membuat dua list, **lista** berisi warna-warna dan **listb** berisi kode warna RGB.
- menggunakan zip() untuk menggabungkan pasangan-pasangan warna dari kedua list tersebut.
- Hasil dari zip() diubah menjadi dictionary x.
- mencetak dictionary x yang berisi pasangan warna dan kode warna RGB-nya

## SOAL 3

Source code:

```
#latman 10.3
dictionary = dict()
namafile = input('Enter file name: ')
try:
    handle = open(namafile)
except:
    print('File cannot be opened:', namafile)
    exit()
for line in handle:
    katakata = line.split()
    if len(katakata) < 2 or katakata[0] != 'From':
        continue
    else:
        dictionary[katakata[1]] = 1 +  dictionary.get(katakata[1],0)
print(dictionary)
```

Output:

```
Enter file name: mbox-short.txt
{'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:

- membuat dictionary kosong dan meminta pengguna untuk memasukkan nama file.
- membuka file yang dimasukkan pengguna. Jika file tidak dapat dibuka, pesan kesalahan akan dicetak dan program akan berhenti.
- Program membaca setiap baris dalam file menggunakan loop for.
- baris dibagi menjadi kata-kata menggunakan split().
- Jika jumlah kata kurang dari 2 atau kata pertama bukan 'From', kita lanjutkan ke baris berikutnya.
- Jika kondisi di atas tidak terpenuhi, kita tambahkan alamat email ke dictionary atau tambahkan hitungannya jika sudah ada menggunakan dictionary.get(). Setiap alamat email dihitung sebagai nilai, dengan kunci adalah alamat email itu sendiri.
- Setelah membaca semua baris, program mencetak dictionary yang berisi daftar alamat email dan jumlah kemunculannya.

## SOAL 4

```
#Latman 10.4
dictionary = {}
namafile = input('Enter file name: ')
try:
    with open(namafile) as handle:
        for line in handle:
            words = line.split()
            if len(words) > 0 and words[0] == "From:":
                domain = words[1][words[1].find("@") + 1:]
                dictionary[domain] = dictionary.get(domain, 0) + 1
except FileNotFoundError:
    print('File cannot be opened:', namafile)
    exit()

print(dictionary)
```

Output:

```
Enter file name: mbox-short.txt
{'uct.ac.za': 6, 'media.berkeley.edu': 4, 'umich.edu': 7, 'iupui.edu': 8, 'caret.cam.ac.uk': 1, 'gmail.com': 1}
```

Penjelasan:

- membuat dictionary kosong untuk menyimpan domain alamat email.
- Pengguna diminta untuk memasukkan nama file.
- file yang dimasukkan pengguna. Jika file tidak dapat dibuka, pesan kesalahan akan dicetak dan program akan berhenti.
- Program membaca setiap baris dalam file menggunakan loop for.
- Setiap baris dibagi menjadi kata-kata menggunakan split().
- Program memeriksa apakah baris mengandung kata kunci 'From:'. Jika iya, kita mengambil domain dari alamat email menggunakan find() dan menambahkan 1 untuk melewati tanda "@".
- Program menambahkan domain ke dictionary atau menambahkan hitungannya jika sudah ada. Kunci adalah domain dan nilai adalah jumlah kemunculannya.
- Jika file tidak dapat ditemukan, kita mencetak pesan kesalahan.
- Setelah membaca semua baris, program mencetak dictionary yang berisi daftar domain alamat email dan jumlah kemunculannya.

Link github: https://github.com/GalihPramana/Praktikum-Alpro-71230976.git