



Laporan Praktikum Algoritma dan Pemrograman

Semester Genap 2023/2024

NIM	71230970
Nama Lengkap	Gregorius Daniel Jodan Perminas
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.

Pengertian Dictionary

Dictionary memiliki kemiripan dengan *list* tetapi lebih bersifat umum. Dalam *list*, indeks harus berupa integer sedangkan indeks dalam *dictionary* dapat berupa apapun. *Dictionary* terdiri dari pasangan **kunci : nilai**. Kunci ini harus bersifat unik, yang artinya tidak boleh ada kunci yang sama dalam satu *dictionary*, kemudian tiap kunci harus memiliki nilai/value yang isinya dapat berupa apa saja. Pasangan kunci dan nilai (key-value pair) biasanya juga disebut sebagai item.

Beberapa sifat *dictionary items*:

- Unordered = Artinya tidak berurutan, key yang pertama kali didefinisikan tidak benar-benar akan menjadi yang "pertama" dibanding key lainnya. Unordered juga berarti bahwa *dictionary* tidak bisa diakses menggunakan indeks (integer) layaknya list.
- Changeable = Value dalam *dictionary* masih bisa diubah setelah dimasukkan
- Unique = Tidak bisa memiliki lebih dari satu key yang sama. Jika terdapat dua buah key yang sama key yang didefinisikan terakhir akan menimpa nilai key yang didefinisikan sebelumnya.

Terdapat dua cara untuk membuat *dictionary* pada Python:

1. Menggunakan fungsi `dict()`

```
eng2sp = dict()  
print(eng2sp)
```

Output: {}

Tanda {} menandakan bahwa *dictionary* masih kosong. Untuk menambahkan isinya kita bisa menggunakan kurung kotak.

```
eng2sp = dict()  
eng2sp['one'] = 'uno'  
print(eng2sp)
```

Output: {'one': 'uno'}

2. Menggunakan kurung kurawal {}

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

```
{'one': 'uno', 'two': 'dos', 'three': 'tres'}
```

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

```
dos
```

```
print(eng2sp['four'])
```

```
KeyError: 'four'
```

Beberapa fungsi dan operator yang dapat digunakan pada *Dictionary*:

1. Fungsi `len` pada *dictionary* dapat digunakan untuk mengetahui jumlah pasangan kunci-nilai

```
print(len(eng2sp))
```

Output: 3

2. Operator `in` dalam *dictionary* dapat mengembalikan nilai `true` atau `false` sesuai dengan kunci yang terdapat dalam *dictionary*.

```
print('one' in eng2sp)
```

Output: True

```
print('uno' in eng2sp)
```

Output: False

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

```
vals = list(eng2sp.values())
```

```
print('uno' in vals)
```

Output: True

Dictionary Sebagai Set Penghitung (Counters)

Sebagai contoh kita diberikan sebuah string dan diharuskan menghitung banyaknya huruf yang muncul, cara yang bisa digunakan misalnya :

1. Membuat 26 variable untuk tiap huruf pada alfabet, kemudian memasukkannya dalam string untuk masing-masing karakter, menambah perhitungan yang sesuai dan menggunakan kondisi sional berantai.
2. Membuat list dengan 26 elemen, kemudian melakukan konversi setiap karakter menjadi angka (menggunakan fungsi `bawaan`), kemudian menggunakan angka sebagai indeks dalam list dan menambah perhitungan yang sesuai.
3. Membuat *dictionary* dengan karakter sebagai kunci dan perhitungan sebagai nilai yang sesuai, karakter dapat ditambahkan item kedalam *dictionary* dan kemudian ditambahkan nilai dari item yang ada.

Demonstrasi dalam Python:

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

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

Perulangan for akan melewati string. Setiap kali terjadi looping, jika karakter c masih belum ada di dalam dictionary maka akan dibuat item baru dengan kunci c dan nilai awal 1. Jika c sudah ada dalam dictionary secara otomatis akan dilakukan penambahan.

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

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

Penggunaan metode get untuk menyederhanakan loop penghitungan ini merupakan metode yang paling umum digunakan dalam Python. Jika dibandingkan loop yang menggunakan pernyataan if dan operator in dengan loop, metode get melakukan hal yang persis sama, tetapi lebih ringkas.

Dictionary dan File

Dictionary bisa juga digunakan untuk menghitung kemunculan suatu kata dalam file yang berupa text.

Berikut demonstrasinya:

```
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
```

Gambar 1 - Romeo.txt

```

fname = "romeo.txt"
try:
    fhandle = open(fname)
except:
    print("File cannot be opened:", fname)
    exit()

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

```

```
{'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}
```

Gambar 2 - Program Mencari Kemunculan Kata dalam File

Penjelasan Program:

- fname = "romeo.txt": Mendefinisikan nama file yang akan dibuka dan dibaca, dalam program ini file bernama "romeo.txt".
- Blok try dan except digunakan untuk menangani kesalahan yang mungkin terjadi saat mencoba membuka file. Jika file tidak dapat, program akan mencetak pesan kesalahan dan keluar.
- counts = dict(): Membuat *dictionary* kosong bernama counts.
- Loop for line in fhandle: akan membaca setiap baris dalam file.
- words = line.split(): Membagi setiap baris menjadi daftar kata. Fungsi split() membagi string menjadi daftar kata berdasarkan spasi.
- Loop for word in words: akan melalui setiap kata dalam daftar kata.
- if word not in counts: akan memeriksa apakah kata tersebut sudah ada dalam *dictionary* counts. Jika tidak, maka kata tersebut ditambahkan ke kamus dengan nilai 1.
- else: counts[word] += 1: Jika kata tersebut sudah ada dalam kamus, maka nilai dari kata tersebut yang berada dalam *dictionary* akan ditambahkan 1.
- print(counts): Mencetak *dictionary* counts, yang sekarang berisi setiap kata yang ditemukan dalam file dan berapa kali kata tersebut muncul.

Looping dan Dictionary

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

```

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

```

Output:

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

Output yang ditampilkan memperlihatkan bahwa kunci tidak berada dalam pola urutan tertentu. Untuk mencetak kunci dari suatu dictionary dalam urutan alfabet, langkah pertama yang harus dilakukan adalah mengubah kunci-kunci dalam dictionary menjadi sebuah list. Hal ini bisa dilakukan dengan menggunakan metode yang tersedia pada objek dictionary. Setelah itu, kita perlu mengurutkan list tersebut (sort). Kemudian, kita melakukan perulangan melalui sorted list tersebut. Pada setiap iterasi perulangan, kita mencetak pasangan kunci-nilai dari dictionary yang kuncinya sudah diurutkan.

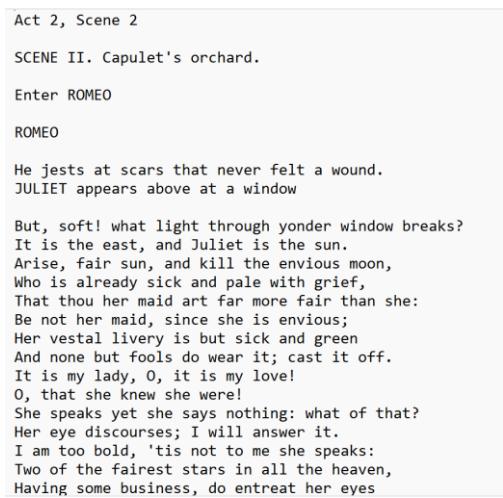
Berikut demonstrasinya:

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



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

Advanced Text Parsing



The screenshot shows a text editor displaying a portion of a Shakespearean play script. The text is formatted with stage directions and character names. The visible text includes:

```
Act 2, Scene 2
SCENE II. Capulet's orchard.
Enter ROMEO
ROMEO
He jests at scars that never felt a wound.
JULIET appears above at a window
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,
That thou her maid art far more fair than she:
Be not her maid, since she is envious;
Her vestal livery is but sick and green
And none but fools do wear it; cast it off.
It is my lady, O, it is my love!
O, that she knew she were!
She speaks yet she says nothing: what of that?
Her eye discourses; I will answer it.
I am too bold, 'tis not to me she speaks:
Two of the fairest stars in all the heaven,
Having some business, do entreat her eyes
```

...dst

Gambar 3 - Romeo.txt dengan 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.strip()
    table = line.maketrans(' ', ' ', string.punctuation)
    line = line.translate(table)
    line = line.lower()
    words = line.split()
    for word in words:
        if word not in counts:
            counts[word] = 1
        else:
            counts[word] += 1
print(counts)

Enter the file name: romeo-full.txt
{'romeo': 40, 'and': 42, 'juliet': 32, 'act': 1, '2': 2, 'scene': 2, 'ii': 1, 'capulets': 1, 'orchard': 2, 'enter': 1, 'he': 5, 'jest': 1, 'at': 9, 'scars': 1, 'that': 30, 'never': 2, 'felt': 1, 'a': 24, 'wound': 1, 'appears': 1, 'above': 6, 'window': 2, 'but': 18, 'soft': 1, 'what': 11, 'light': 5, 'through': 2, 'yonder': 2, 'breaks': 1, 'it': 22, 'is': 21, 'the': 34, 'east': 1, 'sun': 2, 'arise': 1, 'fair': 4, 'kill': 2, 'envious': 2, 'moon': 4, 'who': 5, 'already': 1, 'sick': 2, 'pale': 1, 'with': 8, 'grief': 2, 'thou': 32, 'her': 14, 'maid': 2, 'art': 7, 'far': 2, 'more': 9, 'than': 6, 'she': 9, 'be': 14, 'not': 18, 'since': 1, 'vestal': 1, 'livery': 1, 'green': 1, 'none': 1, 'fools': 1, 'do': 7, 'wear': 1, 'cast': 1, 'off': 1, 'my': 29, 'la': 2, 'o': 11, 'love': 24, 'knew': 1, 'were': 9, 'speaks': 3, 'yet': 9, 'says': 1, 'nothing': 1, 'of': 20, 'eye': 2, 'discourses': 1, 'i': 61, 'will': 8, 'answer': 1, 'am': 7, 'too': 8, 'bold': 1, 'tis': 4, 'to': 3 ...dst

```

Gambar 4 - Program Advanced Text Parsing

Penjelasan Program:

- import string: Mengimpor modul string yang menyediakan operasi yang berguna pada string.
- fname = input mendefinisikan nama file yang akan dibuka dan dibaca, dalam program ini file bernama “romeo-full.txt”.
- Blok try dan except digunakan untuk menangani kesalahan yang mungkin terjadi saat mencoba membuka file. Jika file tidak dapat dibuka, program akan mencetak pesan kesalahan dan keluar.
- counts = dict(): Membuat *dictionary* kosong bernama counts.
- Loop for line in fhand: akan membaca setiap baris dalam file.
- line = line.strip(): Menghapus spasi putih di awal dan akhir setiap baris.
- table = line.maketrans(' ', ' ', string.punctuation): Ini membuat tabel terjemahan yang digunakan untuk menghapus semua tanda baca dari setiap baris.
- line = line.translate(table): Menerapkan tabel terjemahan ke setiap baris untuk menghapus tanda baca.
- line = line.lower(): Mengubah semua huruf dalam baris menjadi huruf kecil.
- words = line.split(): Membagi setiap baris menjadi daftar kata. Fungsi split() membagi string menjadi daftar kata berdasarkan spasi.
- Loop for word in words: akan melalui setiap kata dalam daftar kata.

- if word not in counts: akan memeriksa apakah kata tersebut sudah ada dalam *dictionary* counts. Jika tidak, maka kata tersebut ditambahkan ke kamus dengan nilai 1.
- else: counts[word] += 1: Jika kata tersebut sudah ada dalam *dictionary*, maka nilai dari kata tersebut akan ditambahkan 1.
- print(counts): Mencetak *dictionary* counts, yang sekarang berisi setiap kata yang ditemukan dalam file dan berapa kali kata tersebut muncul.

REFERENSI

[7 Hal Dasar yang Harus diketahui Tentang Dictionary pada Python \(petanikode.com\)](#)

[Python Dasar: Struktur Dictionary @ | Jago Ngoding](#)

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.

LINK GITHUB: https://github.com/danieljodan/PrakAlPro10_A_71230970.git

SOAL 1

```
# Program untuk mendapatkan nilai key, value, dan item dari sebuah dictionary
def tampilkan(d):
    print("key\t" "value\t" "item")
    counter = 0
    for i in d:
        counter += 1
        print(i, end = "\t")
        print(d[i], end = "\t")
        print(counter, end = "\n")

dictionary = {1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}
tampilkan(dictionary)
```

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

Penjelasan Kode Program:

Program ini akan mencetak semua kunci, nilai, dan urutan item dalam dictionary yang diberikan dalam format tabel.

1. Fungsi tampilkan(d) didefinisikan, menerima satu argumen yaitu d, dengan asumsi d adalah sebuah *dictionary*
2. print("key\t" "values\t" "item"): Mencetak judul kolom tabel
3. counter = 0: Mendefinisikan variabel counter dan menginisialisasinya dengan 0. Variabel ini akan digunakan untuk melacak urutan item dalam dictionary.
4. Loop for i in d: akan melalui setiap kunci dalam dictionary d.
5. counter += 1: Menambahkan 1 ke counter setiap kali loop berjalan, sehingga memberikan urutan item dalam dictionary.
6. print(i, end = "\t"): Mencetak kunci dari pasangan kunci-nilai saat ini dalam dictionary, diikuti oleh tab.
7. print(d[i], end = "\t"): Mencetak nilai dari pasangan kunci-nilai saat ini dalam dictionary, diikuti oleh tab.
8. print(counter, end = "\n"): Mencetak nilai counter saat ini, yang mewakili urutan item dalam dictionary, dan kemudian membuat baris baru (ditandai oleh "\n").
9. dictionary = {1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}: Mendefinisikan dictionary yang akan ditampilkan.

10. `tampilkan(dictionary)`: Memanggil fungsi `tampilkan(d)` dengan dictionary yang telah didefinisikan sebagai argumen.

SOAL 2

```
# Program untuk memetakan dua List menjadi satu dictionary
def satudict(Lista, Listb):
    dictionary = {}
    for i in range(len(Lista)):
        dictionary[Lista[i]] = Listb[i]
    print(dictionary)

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

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

Penjelasan Kode Program:

Program ini mengambil dua list dan memetakan elemen-elemen dalam list tersebut menjadi pasangan kunci-nilai dalam satu dictionary.

1. Program ini mendefinisikan sebuah fungsi `satudict(Lista, Listb)` yang menerima dua argumen, `Lista` dan `Listb`, yang diasumsikan sebagai list.
2. `dictionary = {}`: Membuat `dictionary` kosong bernama `dictionary`. Dictionary ini akan digunakan untuk menyimpan pasangan key-value dari dua list.
3. `Loop for i in range(len(Lista))`: akan melalui setiap indeks dalam `Lista`.
4. `dictionary[Lista[i]] = Listb[i]`: Menambahkan elemen dari `Lista` sebagai kunci dan elemen dari `Listb` pada indeks yang sama sebagai nilai ke dalam `dictionary`.
5. `print(dictionary)`: Mencetak `dictionary`, yang sekarang berisi pasangan key-value dari dua list.
6. `Lista = ['red', 'green', 'blue']` dan `Listb = ['#FF0000', '#008000', '#0000FF']`: Mendefinisikan dua list yang akan dipetakan menjadi satu dictionary.
7. `satudict(Lista, Listb)`: Memanggil fungsi `satudict(Lista, Listb)` dengan dua list yang telah didefinisikan sebagai argumen.

SOAL 3

```
# Program Menghitung Banyak Pesan Yang Masuk Dari Email dan Disajikan Dalam Dictionary
def tampilkan(namaFile):
    with open(namaFile, 'r') as f:
        pesan = f.readlines()
        d = {}
        for line in pesan:
            line = line.strip()
            words = line.split()
            if 'Author:' in words:
                for word in words:
                    if '@' in word:
                        if word not in d:
                            d[word] = 1
                        else:
                            d[word] += 1
        print(d)

namaFile = input("Masukkan nama file: ")
tampilkan(namaFile)

{'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 Kode Program:

Program ini akan membaca log email dan menyajikan dalam bentuk dictionary. Kemudian menghitung banyak pesan yang masuk dari email.

1. Program ini mendefinisikan sebuah fungsi `tampilkan(namaFile)` yang menerima satu argumen, `namaFile`, yang diasumsikan bahwa teks berisi pesan email.
2. `with open(namaFile, 'r') as f:` Membuka file dengan nama `namaFile` dalam mode baca ('r')
3. `pesan = f.readlines():` Membaca semua baris dalam file dan menyimpannya dalam list `pesan`.
4. `d = {}:` Membuat *dictionary* kosong bernama `d`. *Dictionary* ini akan digunakan untuk menyimpan setiap alamat email yang ditemukan dalam file dan berapa kali alamat email tersebut muncul.
5. `Loop for line in pesan:` Akan melalui setiap baris dalam list `pesan`.
6. `line = line.strip():` Menghapus spasi putih di awal dan akhir setiap baris.
7. `words = line.split():` Membagi setiap baris menjadi daftar kata. Fungsi `split()` membagi string menjadi daftar kata berdasarkan spasi.
8. `if 'Author:' in words:` Akan memeriksa apakah kata 'Author:' ada dalam baris. Jika ya, maka program akan melanjutkan ke loop berikutnya.
9. `Loop for word in words:` Akan melalui setiap kata dalam daftar kata.
10. `if '@' in word:` Memeriksa apakah karakter '@' ada dalam kata.
11. `if word not in d:` Memeriksa apakah alamat email tersebut sudah ada dalam *dictionary*. Jika tidak, maka alamat email tersebut ditambahkan ke *dictionary* dengan nilai 1.
12. `else: d[word] += 1:` Jika alamat email tersebut sudah ada dalam *dictionary*, maka nilai dari alamat email tersebut akan ditambahkan 1.

13. `print(d)`: Mencetak *dictionary*, yang sekarang berisi setiap alamat email yang ditemukan dalam file dan berapa kali alamat email tersebut muncul.
14. `namaFile = input("Masukkan nama file: ")`: Meminta pengguna untuk memasukkan nama file yang akan diproses.
15. `tampilkan(namaFile)`: Memanggil fungsi `tampilkan(namaFile)` dengan nama file yang dimasukkan pengguna sebagai argumen.

SOAL 4

```
# Program Pencatat Nama Domain Pengirim Pesan
def tampilkan(namaFile):
    with open(namaFile, 'r') as f:
        pesan = f.readlines()
        d = {}
        for line in pesan:
            line = line.strip()
            words = line.split()
            if 'Author:' in words:
                for word in words:
                    if '@' in word:
                        domain = word.split('@')[-1]
                        if domain not in d:
                            d[domain] = 1
                        else:
                            d[domain] += 1
        print(d)

namaFile = input("Masukkan nama file: ")
tampilkan(namaFile)

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

Penjelasan Kode Program:

Program ini akan mencatat data nama domain pengirim pesan, kemudian menghitung jumlah pesan yang dikirim masing-masing domain.

1. Program ini mendefinisikan sebuah fungsi `tampilkan(namaFile)` yang menerima satu argumen, `namaFile`, yang diharapkan menjadi nama file teks yang berisi pesan email.
2. `with open(namaFile, 'r') as f:` Membuka file dengan nama `namaFile` dalam mode baca ('r').
3. `pesan = f.readlines()`: Ini membaca semua baris dalam file dan menyimpannya dalam list `pesan`.
4. `d = {}`: Membuat *dictionary* kosong bernama `d`. *Dictionary* ini akan digunakan untuk menyimpan setiap domain email yang ditemukan dalam file dan berapa kali domain tersebut muncul.
5. Loop `for line in pesan:` Akan melalui setiap baris dalam list `pesan`.
6. `line = line.strip()`: Menghapus spasi putih di awal dan akhir setiap baris.
7. `words = line.split()`: Membagi setiap baris menjadi daftar kata.

8. if 'Author:' in words: Akan memeriksa apakah kata 'Author:' ada dalam baris. Jika ya, maka program akan melanjutkan ke loop berikutnya.
9. Loop for word in words: Akan melalui setiap kata dalam daftar kata.
10. if '@' in word: Akan memeriksa apakah karakter '@' ada dalam kata, yang menandakan bahwa itu adalah alamat email.
11. domain = word.split('@')[-1]: Mengambil bagian setelah '@' dalam alamat email, yang merupakan domain email.
12. if domain not in d: Akan memeriksa apakah domain tersebut sudah ada dalam *dictionary*. Jika tidak, maka domain tersebut ditambahkan ke *dictionary* dengan nilai 1.
13. else: d[domain] += 1: Jika domain tersebut sudah ada dalam *dictionary*, maka nilai dari domain tersebut akan ditambahkan 1.
14. print(d): Mencetak *dictionary*, yang berisi setiap domain email yang ditemukan dalam file dan berapa kali domain tersebut muncul.
15. namaFile = input("Masukkan nama file: "): Meminta pengguna untuk memasukkan nama file yang akan diproses.
16. tampilkan(namaFile): Memanggil fungsi tampilkan(namaFile) dengan nama file yang dimasukkan pengguna sebagai argumen.