
# Laporan Praktikum Algoritma dan Pemrograman

Semester Genap 2023/2024

| | |
|---|---|
| **NIM** | **<71230989>** |
| **Nama Lengkap** | **<Yohanes Nevan Adventus Wibawa>** |
| **Minggu ke / Materi** | **10 / 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.

## Tipe Data Dictionary

Dictionary adalah struktur data yang serupa dengan list, tetapi lebih umum. Sementara list menggunakan indeks dalam bentuk integer, dictionary memungkinkan penggunaan indeks dalam bentuk apa pun. Dictionary terdiri dari pasangan kunci:nilai di mana setiap kunci harus unik, artinya tidak boleh ada kunci yang sama dalam satu dictionary.

Dictionary dapat dianggap sebagai pemetaan antara kumpulan kunci dan kumpulan nilai, di mana setiap kunci memetakan ke nilai tertentu. Pasangan kunci:nilai ini sering disebut sebagai item.

Sebagai contoh, kita dapat membuat dictionary yang memetakan kata-kata dalam bahasa Inggris ke bahasa Spanyol, di mana kata-kata tersebut menjadi kunci dan nilai dalam bentuk string. Dalam hal ini, diasumsikan bahwa setiap kata dalam bahasa Inggris memiliki satu arti dalam bahasa Spanyol.

Fungsi dict digunakan untuk membuat dictionary baru yang kosong. Penting untuk diingat bahwa dict merupakan sebuah fungsi bawaan dari Python, oleh karena itu, sebaiknya nama variabel tidak menggunakan nama ini untuk menghindari konflik.

```
Test = dict()
print(Test)
```

Output = `{}`

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

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

Output = `{'one': 'uno'}`

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

Dalam potongan kode tersebut, jelas terlihat bahwa ada pembuatan item dalam dictionary yang menghubungkan kunci 'one' dengan nilai 'uno'. Saat dictionary tersebut dicetak, kita akan melihat pasangan nilai-kunci yang mencakup kunci dan nilainya.

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

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

Pengurutan pasangan kunci-nilai tidaklah sama. Secara umum urutan 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).

```
coba = dict()
coba = {'one': 'uno', 'two': 'dos', 'three': 'tres'}
print('one' in coba)
print('alo' in coba)
```

Output =
```
True
False
```

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 sesuai dengan panjang list. Dalam dictionary, Python menggunakan algoritma yang disebut Hash Table dengan properti yang luar biasa, operator in membutuhkan waktu yang sama untuk memprosesnya dan tidak memperdulikan banyak item yang ada didalam dictionary.

## Dictionary sebagai set penghitung (counters)

Penggunaan model dictionary dalam perhitungan dianggap lebih praktis karena kita tidak perlu memiliki pengetahuan sebelumnya tentang karakter apa yang akan muncul. Berikut adalah beberapa contoh perhitungan yang dapat dilakukan dengan model dictionary.

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

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

Dictionary menyediakan metode get yang berfungsi untuk mengambil nilai dari kunci yang ditentukan, serta nilai default jika kunci tidak ditemukan dalam dictionary tersebut. Jika kunci ditemukan dalam dictionary, metode get akan mengembalikan nilai yang sesuai dengan kunci tersebut; namun, jika kunci tidak ditemukan, metode ini akan mengembalikan nilai default yang telah ditentukan sebelumnya. Berikut adalah contoh penggunaannya.

Dalam melakukan perhitungan jumlah kemunculan huruf dalam sebuah string, ada beberapa pendekatan yang dapat digunakan:

1. Pendekatan pertama melibatkan pembuatan 26 variabel untuk setiap huruf dalam alfabet, kemudian setiap karakter dalam string dimasukkan ke dalam variabel yang sesuai, dan dilakukan penambahan perhitungan yang sesuai dengan setiap karakter. Pendekatan ini menggunakan kondisional berantai untuk mengelola setiap karakter.
2. Pendekatan kedua melibatkan pembuatan sebuah list dengan 26 elemen, di mana setiap karakter dalam string dikonversi menjadi angka menggunakan fungsi bawaan, kemudian angka tersebut digunakan sebagai indeks dalam list untuk menambahkan perhitungan yang sesuai.
3. Pendekatan ketiga melibatkan pembuatan sebuah dictionary dengan karakter sebagai kunci dan jumlah kemunculannya sebagai nilai yang sesuai. Setiap karakter dalam string ditambahkan sebagai item ke dalam dictionary, dan nilai dari setiap item ditambahkan sesuai dengan kemunculan karakter tersebut.

Meskipun semua pendekatan tersebut dapat menghasilkan hasil yang sama dalam perhitungan jumlah kemunculan huruf, penggunaan model dictionary dianggap lebih praktis karena tidak memerlukan pengetahuan sebelumnya tentang huruf mana yang akan muncul dalam string. Dalam model ini, kita hanya perlu menyediakan ruang untuk huruf yang akan muncul, dan dictionary akan secara dinamis menangani setiap kemunculan huruf.

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

Output =
```
100
0
```

## Dictionary dan File

Salah satu kegunaan umum dari dictionary adalah untuk menghitung jumlah kemunculan kata-kata dalam sebuah file yang berisi teks tertulis. Mari kita mulai dengan menggunakan sebuah file yang berisi teks yang sangat sederhana, misalnya, kutipan dari Romeo dan Juliet yang telah disederhanakan dan tanpa tanda baca.

Untuk contoh pertama, kita akan menggunakan versi teks yang disingkat tanpa tanda baca. Kemudian, kita akan mencoba menulis sebuah program Python yang membaca setiap baris dari file tersebut, memecah setiap baris menjadi daftar kata-kata, dan kemudian melakukan perulangan melalui setiap kata dalam baris tersebut. Dalam proses ini, kita akan menggunakan dictionary untuk menghitung jumlah kemunculan setiap kata.

Asumsikan kita menggunakan dua perulangan `for`: perulangan luar untuk membaca setiap baris dari file, dan perulangan dalam untuk melakukan iterasi melalui setiap kata pada baris tertentu. Pola ini dikenal sebagai nested loop karena satu perulangan berada di dalam yang lain.

Perulangan dalam akan dieksekusi untuk setiap iterasi dari perulangan luar. Dalam konteks ini, perulangan dalam akan berjalan "lebih cepat" daripada perulangan luar. Kombinasi dari kedua perulangan ini memastikan bahwa setiap kata pada setiap baris file akan dihitung dengan benar.

```
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 =

```
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, 'a
nd': 3, 'Juliet': 1, 'sun': 2, 'Arise': 1, 'fair': 1, 'kill': 1, 'envious': 1, 'moon': 1, 'Who': 1, 'already': 1, 'sick': 1, 'pale': 1, 'wi
th': 1, 'grief': 1}
PS C:\Users\LENOVO\Tugas PrakAlPro\Tugas 10_>
```

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.

## 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.

```
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. 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 :

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

Output =
```
annie 42
jan 100
```
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])
```

Output =
```
['chuck', 'annie', 'jan']
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.

## Advenced Text Parsing

Python menyediakan fungsi `split()` yang secara default akan membagi string berdasarkan spasi, menjadikan setiap kata sebagai token yang dipisahkan oleh spasi. Misalnya, kata "soft!" dan "soft" akan dianggap sebagai dua kata yang berbeda, dan keduanya akan dihitung secara terpisah dalam dictionary.

Hal yang sama berlaku untuk penanganan huruf kapital. Misalnya, kata "who" dan "Who" dianggap sebagai kata yang berbeda dan akan dihitung secara terpisah dalam dictionary.

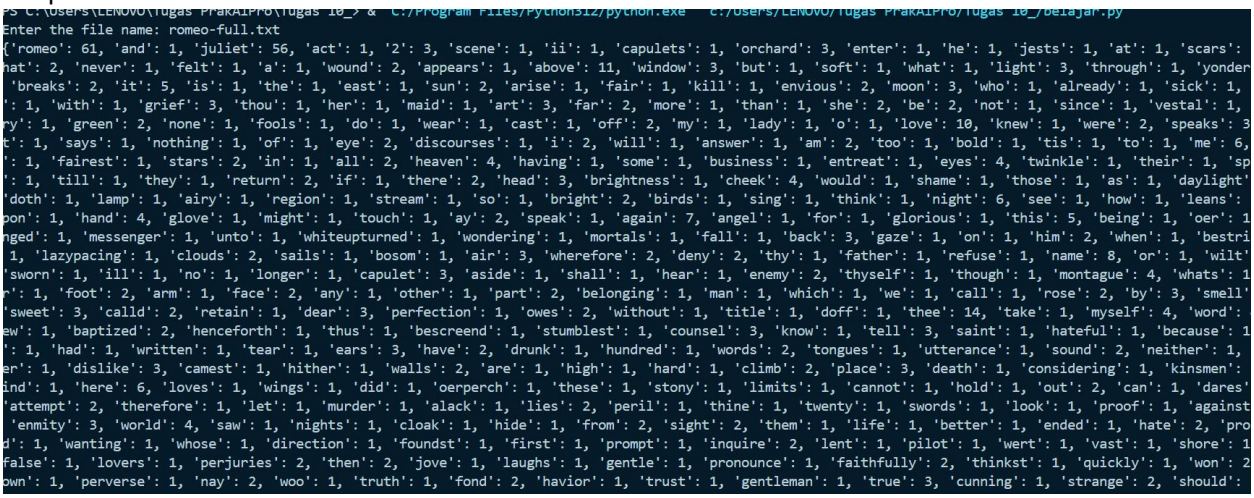
Selain menggunakan fungsi `split()`, ada juga pendekatan lain yang dapat digunakan, seperti menggunakan metode string lain seperti `lower()`, `punctuation`, dan `translate()`. Di antara metode-metode tersebut, metode `translate()` dapat dianggap sebagai metode string yang paling halus (paling tidak terlihat).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

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

Dalam metode `translate()`, karakter pada `fromstr` akan diganti dengan karakter pada posisi yang sama pada `tostr`, dan semua karakter yang ada dalam `deletetr` akan dihapus. Jika `fromstr` atau `tostr` kosong, karakter yang sesuai akan diabaikan. Dalam kasus ini, kita tidak akan menentukan `tostr`, tetapi kita akan menggunakan parameter `deletetr` untuk menghapus semua tanda baca. Python secara otomatis akan mengidentifikasi karakter mana yang dianggap sebagai "tanda baca".

```
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

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)

```
n= int(input("Input data = "))
kamus = dict()

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

print("Dictionary = ", kamus)
```

Output =
```
Input data = 5
Dictionary =  {1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
```

Kode ini meminta pengguna untuk memasukkan sebuah angka, kemudian membuat sebuah dictionary yang berisi kuadrat dari setiap bilangan bulat dari 1 hingga angka yang dimasukkan oleh pengguna, dan akhirnya mencetak dictionary tersebut. Dalam kode ini, sebuah dictionary kamus dibuat menggunakan dictionary comprehension, yang secara ringkas memungkinkan pembuatan dictionary dengan menggunakan satu baris kode.

Kasus 10.2

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', 'S005', 'S009', 'S001', 'S002'}
```

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.

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.

```
count = 0
dictionary_words = dict()
fname = input('Masukkan nama file : ')
fword = input('Kata yang dicari : ')
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 =

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

Daftar Kamus :

{'programming': 1}

Kata programming ditemukan dalam kamus
```

# 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

```
dict_nilai = {1:10, 2:20, 3:30, 4:40, 5:50, 6:60}
nomor = 1
print('key', 'values', 'items')
for key, values in dict_nilai.items():
    print(key ,' ',values,'    ', nomor)
    nomor += 1
```

Output =

```
key values items
1   10     1
2   20     2
3   30     3
4   40     4
5   50     5
6   60     6
```

Penjelasan =

Dalam kode tersebut, terdapat sebuah dictionary yang disebut `dict_nilai`, yang berisi beberapa pasangan kunci-nilai. Selanjutnya, terdapat sebuah variabel `nomor` yang diinisialisasi dengan nilai 1. Pada iterasi berikutnya, program menggunakan pernyataan `for` untuk mengiterasi melalui setiap pasangan kunci-nilai dalam `dict_nilai`.

Pada setiap iterasi, program mencetak kunci, nilai, dan nomor urut dari pasangan kunci-nilai tersebut. Setiap pasangan kunci-nilai diambil dari `dict_nilai`, dan kemudian dicetak bersama dengan nomor urut yang telah ditambahkan sebelumnya.

Setelah mencetak satu pasangan kunci-nilai, variabel `nomor` akan ditambahkan satu agar nomor urut pasangan kunci-nilai berikutnya menjadi urutan berikutnya. Hasil cetakannya adalah setiap pasangan kunci-nilai dalam `dict_nilai` dicetak secara terpisah, dengan nomor urut pasangan tersebut. Ini memungkinkan untuk melihat urutan relatif dari setiap pasangan kunci-nilai dalam dictionary tersebut.

## SOAL 2

```
datalist1 = ['red', 'green', 'blue']
datalist2 = ['#FF0000','#008000','#0000FF']
d = dict()
for key, value in zip(datalist1, datalist2):
    d[key] = value
print(d)
```

Output =

```
{'red': '#FF0000', 'green': '#008000', 'blue': '#0000FF'}
PS C:\Users\LENOVO\Tugas PrakAlPro\Tugas 10_>
```

Penjelasan =

Kode tersebut menghasilkan sebuah struktur data dictionary di Python. Dengan memanfaatkan fungsi zip(), ia menghubungkan dua list bersama-sama: satu sebagai kunci dan yang lainnya sebagai nilai. Setiap pasangan kunci-nilai dipetakan ke dalam dictionary d. Ini berarti bahwa, jika kita ingin nilai dari suatu kunci, kita bisa langsung mengaksesnya melalui kunci tersebut dalam dictionary. Proses ini memungkinkan pengaturan data yang terstruktur, memungkinkan akses dan manipulasi data secara efisien dalam kode. Dalam konteks ini, datalist1 dan datalist2 membentuk suatu pasangan yang merepresentasikan hubungan kunci-nilai, yang terkadang dapat merangkum hubungan konsepik yang lebih kompleks atau terdistribusi.

## Soal 3

```
def bacalog(filename):
    email_= {}
    pesan = 0

    with open(filename, 'r') as file:
        for line in file:
            if line.startswith('From '):
                email = line.split()[1]
                email_[email] = email_.get(email, 0) + 1
                pesan += 1

    return email_, pesan

bacalog2 = input("Masukkan nama file: ")
email_, pesan  = bacalog(bacalog2)

print(email_)
```

Output =

```
Masukkan nama file: 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}
PS C:\Users\LENOVO\Tugas PrakAlPro\Tugas 10_>
```

Penjelasan =

Kode fungsi yang mengagumkan: bacalog() dan main(). Fungsi bacalog() membawa kita ke dalam dunia yang misterius dan memikat, di mana ia membaca dengan cermat setiap garis file untuk menemukan permata-permata yang disembunyikan, yaitu alamat email yang berkilauan, dimulai dengan 'From '. Fungsi ini bekerja dengan anggun dan penuh perhitungan, menghitung jumlah pesan dari setiap alamat email dengan presisi yang memukau.

Sementara itu, main() berperan sebagai penuntun yang ramah dan cerdas, siap untuk menyambut petualang yang memasuki dunianya. Ia dengan lembut meminta pengguna untuk memberikan nama file, menawarkan kesempatan bagi mereka untuk berbagi cerita melalui keyboard mereka.kode, memanggil bacalog() untuk dalam file tersebut.

## Soal 4

```python
def menghitung_email(email):
    return email.split('@')[-1]

def menghitung_email1(filename):
    counts = {}

    with open(filename, 'r') as file:
        for line in file:
            if line.startswith('From '):
                email = line.split()[1]
                awal = menghitung_email(email)
                counts[awal] = counts.get(awal, 0) + 1

    return counts
filename = input("Masukkan nama file: ")
counts = menghitung_email1(filename)

print(counts)
```

Output =

```
Masukkan nama file: mbox-short.txt
{'uct.ac.za': 6, 'media.berkeley.edu': 4, 'umich.edu': 7, 'iupui.edu': 8, 'caret.cam.ac.uk': 1, 'gmail.c
om': 1}
PS C:\Users\LENOVO\Tugas PrakAlPro\Tugas 10_>
```

Penjelasan =

Dalam kode tersebut, terdapat dua fungsi utama: menghitung_email1() dan menghitung_email(). Fungsi menghitung_email1() membaca sebuah file dan menghitung jumlah kemunculan setiap domain email yang muncul dalam baris-baris yang dimulai dengan 'From '. Fungsi menghitung_email() berperan dalam memisahkan domain dari alamat email yang diberikan. Setelah itu, skrip utama langsung meminta pengguna untuk memasukkan nama file, lalu menghitung domain email yang muncul dalam file tersebut menggunakan fungsi menghitung_email1(), dan mencetak hasilnya. Dengan menyederhanakan struktur, kode menjadi lebih langsung dan mudah dipahami.

## Link Github =

Link = https://github.com/YohanesNevan/Tugas-Laporan-Alpro-10_.git