



Laporan Praktikum Algoritma dan Pemrograman

Semester Genap 2023/2024

NIM	71230992
Nama Lengkap	Dictionary
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.

MATERI 1

Materi

Dictionary mirip dengan list, tetapi lebih umum. Sementara indeks dalam dictionary dapat berupa apapun, indeks dalam list harus berupa integer.

Dictionary terdiri dari pasangan **kunci:nilai**. Kunci harus berbeda, tidak boleh ada dua kunci yang sama dalam *dictionary*.

Dictionary juga dapat dianggap sebagai pemetaan antara kumpulan nilai dan kumpulan indeks, masing-masing memetakan nilai. Pasangan nilai kunci, atau "item", adalah istilah yang digunakan untuk menggambarkan asosiasi kunci dan nilai tersebut.

Sebagai contoh, kita akan membuat kamus yang menggabungkan kata-kata dalam bahasa Inggris dan Spanyol. Dengan demikian, data string akan berfungsi sebagai kunci dan nilai. Sepertinya setiap kata yang digunakan dalam bahasa Inggris memiliki arti yang sama dalam bahasa Spanyol.

Karena dict adalah fitur python yang sudah ada, harus dihindari untuk menggunakannya sebagai nama variabel. Tujuannya adalah untuk membuat dictionary baru yang kosong.

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

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

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

Jika Anda melihat potongan kode di atas, Anda akan melihat bahwa baris tersebut membuat item yang menghubungkan nilai "**satu**" ke kunci "**uno**". Selain itu, pasangan nilai kunci antara nilai dan kunci akan terlihat ketika Anda mencetak kamus tersebut.

```
print(eng2sp)
```

```
{'one': 'uno'}
```

Jika kita membuat kamus baru dengan tiga item dan mencetak hasilnya, kita juga menggunakan format input.

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

Pengurutan pasangan nilai-kunci berbeda. Urutan item di dictionary biasanya tidak dapat diprediksi. Karena dictionnaire tidak pernah ada elemen yang diberikan indeks dengan indeks integer, hal ini tidak menjadi masalah yang signifikan. Sebaliknya, Anda dapat menemukan nilai yang relevan dengan menggunakan kunci.

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

Jika kita membuat kamus baru dengan tiga item dan mencetak hasilnya, kita juga menggunakan format input.

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

Fungsi len pada dictionary digunakan untuk mengembalikan jumlah pasangan nilai kunci:

```
len(eng2sp)  
3
```

Operator in dalam dictionary mengembalikan nilai benar (true) atau salah (false) sesuai dengan kunci yang 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 nilai 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 dictionary dan list. Untuk dictionary, Python menggunakan algoritma yang disebut Hash Table, yang memiliki properti yang luar biasa, dan membutuhkan waktu yang sama untuk memprosesnya tanpa memperhatikan banyaknya item dalam dictionary. Untuk list, algoritma pencarian linear digunakan, dan waktu pencarian menjadi lebih lama seiring dengan panjang list.

Dictionary sebagai set penghitung (counters)

Sebagai contoh, kita diberi string dan diminta untuk menghitung berapa banyak huruf yang muncul. Berikut adalah beberapa metode yang dapat digunakan:

1. membuat 26 variable untuk setiap huruf alfabet, kemudian memasukkannya ke dalam string untuk masing-masing karakter, menambah perhitungan yang sesuai, dan menggunakan kondisional berantai.
2. membuat list 26 elemen, menggunakan fungsi bawaan untuk mengubah setiap karakter menjadi angka, kemudian menggunakan angka tersebut sebagai indeks dalam list dan menambah perhitungan yang sesuai.
3. Dengan membuat kamus dengan karakter sebagai kunci dan perhitungan sebagai nilai yang sesuai, Anda dapat menambah item ke dalam kamus dan kemudian menambah nilainya.

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

Perhitungan—atau perhitungan—digunakan untuk melaksanakan implementasi. Model perhitungan tertentu biasanya lebih baik daripada model perhitungan lainnya. Penggunaan komputasi model dictionary lebih praktis karena kita hanya perlu memberikan ruang untuk huruf yang akan muncul daripada mengetahui huruf mana yang akan muncul dalam string. Modelnya dapat dilihat 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 di atas diberi nama "histogram", yang merupakan istilah statistika untuk kumpulan perhitungan (atau frekuensi).

Untuk akan melewati string. Saat terjadi looping, jika karakter *c* tidak ada dictionary, maka akan dibuat item baru dengan kunci *c* dan nilai awal 1—asumsinya telah muncul sekali—dan jika *c* sudah ada dalam kamus, secara otomatis akan menambah *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:

Dalam dictionary, metode *get* mengambil kunci dan nilai default. Jika kunci ditemukan di sana, dictionary akan mengembalikan nilai yang sesuai, tetapi jika tidak, akan mengembalikan nilai default. Sebagai ilustrasi:

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

Untuk menulis loop histogram yang lebih ringkas, Anda dapat menggunakan *get*. Dalam kasus di mana kunci tidak ada dalam dictionary, metode *get* secara otomatis menangani masalah tersebut. Kita dapat meringkas empat baris menjadi satu baris dengan menghilangkan statement *if*.

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

Hasil dari program diatas adalah :

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

Akhirnya, metode get menjadi "idiom" yang sangat umum digunakan dalam Python untuk menyederhanakan loop penghitungan ini. Loop yang menggunakan pernyataan if dan operator in dan loop yang menggunakan metode get melakukan hal yang sama, tetapi yang pertama lebih ringkas.

Dictionary dan File

Dictionary dapat digunakan untuk menghitung berapa banyak kata yang muncul dalam file yang terdiri dari beberapa teks tertulis. Mari kita mulai dengan file kata yang sangat sederhana yang diambil dari naskah Romeo dan Juliet.

Pertama, teks yang disingkat dan disederhanakan tanpa tanda baca akan digunakan, dan kemudian akan digunakan teks adegan dengan tanda baca.

```
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
```

Kami akan mencoba menulis program Python yang dapat membaca baris-baris dalam file, memecah setiap baris menjadi daftar kata, lalu melakukan perjalanan melalui setiap baris dan menggunakan kamus untuk menghitung setiap kata.

Jika kita asumsikan bahwa kita menggunakan dua for untuk perulangan, Perulangan di dalam file mekalukan iterasi melalui setiap kata pada baris tertentu, dan perulangan di luar file membaca baris file. Karena salah satu perulangan adalah perulangan bagian luar dan yang lainnya adalah perulangan bagian dalam, ini adalah contoh dari pola yang disebut nested loop.

Setiap kali perulangan luar membuat iterasi tunggal, perulangan dalam melakukan eksekusi pada semua iterasi. Pada saat ini, perulangan bagian dalam "lebih cepat", sedangkan perulangan luar 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)

```

Untuk menambah variable pada statement else, model penulisan yang lebih singkat digunakan. `counts[word] += 1` sama dengan `counts[word] = counts[word] + 1`. Metode lain, seperti `- =`, `* =`, dan `/ =`, dapat digunakan untuk mengubah nilai variabel ke jumlah yang diinginkan.

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

```

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}

```

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])
```

Outputnya akan tampak sebagai berikut:

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

Sebagai hasilnya, kunci tidak berada dalam pola urutan tertentu. Beberapa idiom looping yang disebutkan sebelumnya dapat digunakan dengan pola ini. Sebagai contoh, jika kita ingin menemukan semua entri di dictionary dengan nilai di atas 10, kita harus menggunakan kode program berikut:

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

Outputnya :

Untuk looping untuk yang menggunakan kunci leksikon, perlu ditambahkan operator indeks untuk mendapatkan nilai yang tepat untuk setiap kunci. Outputnya akan ditunjukkan sebagai berikut:

```
jan 100  
annie 42
```

Program diatas hanya akan menampilkan data nilai diatas 10.

Untuk mencetak kunci dalam urutan alfabet, pertama-tama Anda harus membuat daftar kunci pada kamus dengan menggunakan metode kunci yang tersedia pada objek kamus. Kemudian, Anda melakukan pengurutan (sort), list, dan loop melalui list yang diurutkan. Terakhir, Anda melihat setiap kunci dan mencetak pasangan nilai kunci yang sudah diurutkan. Di bawah ini adalah contoh implementasi kode:

```
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.

Advanced Text Parsing

Pada bagian sebelumnya, kita menggunakan contoh file di mana kalimatnya telah dihilangkan tanda bacanya. Pada bagian ini, kita akan menggunakan file dengan tanda baca lengkap, yaitu romeo.txt.

```
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 memiliki fungsi split yang dapat mencari spasi dan mengubah kata-kata menjadi token yang terpisah oleh spasi. Misalnya, dalam kamus, kata "lembut!" dan "lembut!" adalah kata-kata yang berbeda, dan masing-masing dibuat kata-kata terpisah.

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

Selain itu, model penyelesaian lain dapat menggunakan metode string lain, seperti turun, punctuation, dan penerjemahan. Perlu diingat bahwa metode string penerjemahan adalah metode string yang paling rumit (hampir tidak kentara). Dokumentasi untuk metode penerjemahan dapat ditemukan di sini.

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

Untuk membuat string kosong untuk fromstr dan tostr, dan untuk deletetr, hapus semua karakter yang ada dalam deletetr.

Meskipun kita tidak akan menentukan `tostr`, kita akan menggunakan parameter `delete` untuk menghapus semua tanda baca. Python akan mengetahui bagian mana yang dianggap sebagai "tanda baca" secara otomatis.

```
>>> 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 :

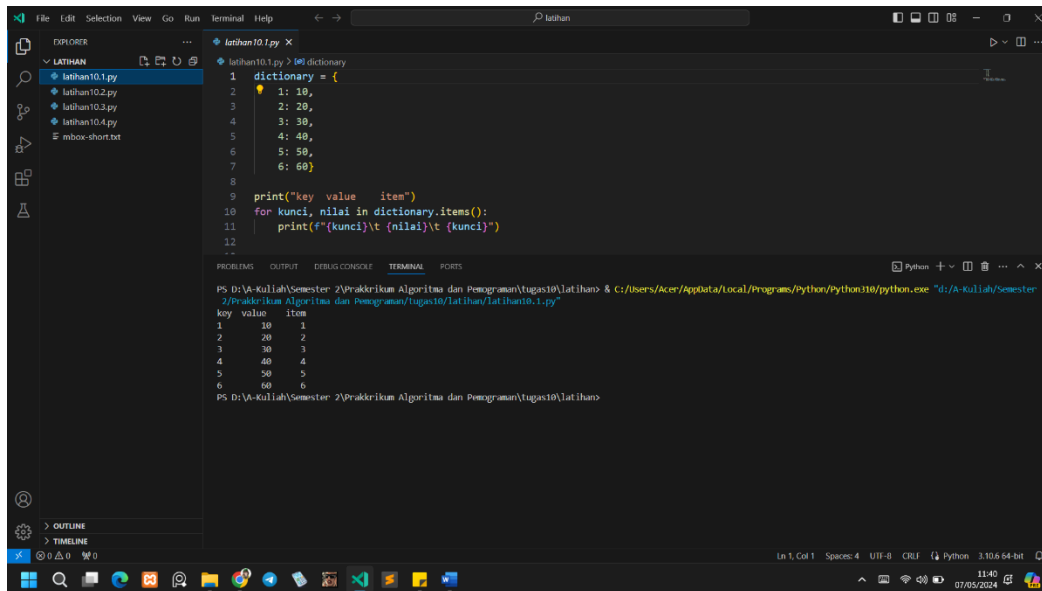
```
Enter the file name: romeo-full.txt
{'swearst': 1, 'all': 6, 'afeard': 1, 'leave': 2, 'these': 2,
'kinsmen': 2, 'what': 11, 'thinkst': 1, 'love': 24, 'cloak': 1,
a': 24, 'orchard': 2, 'light': 5, 'lovers': 2, 'romeo': 40,
'maiden': 1, 'whiteupturned': 1, 'juliet': 32, 'gentleman': 1,
'it': 22, 'leans':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

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

Latihan 10.1

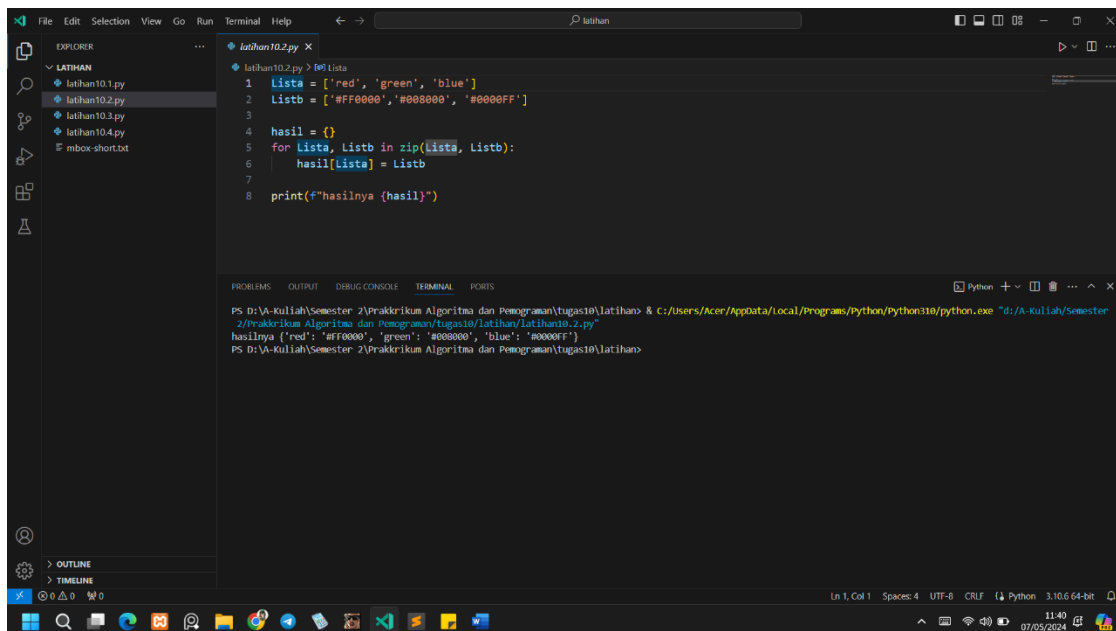


```
1 dictionary = {
2     1: 10,
3     2: 20,
4     3: 30,
5     4: 40,
6     5: 50,
7     6: 60}
8
9 print("key value item")
10 for kunci, nilai in dictionary.items():
11     print(f"{kunci}\t{nilai}\t{kunci}")
12
```

```
PS D:\A-kuliah\Semester 2\Praktikum Algoritma dan Pemrograman\tugas10\latihan> & C:/Users/acer/AppData/Local/Programs/Python/Python310/python.exe "d:/A-kuliah/Semester 2/Praktikum Algoritma dan Pemrograman\tugas10\latihan\latihan10.1.py"
key value item
1 10 1
2 20 2
3 30 3
4 40 4
5 50 5
6 60 6
PS D:\A-kuliah\Semester 2\Praktikum Algoritma dan Pemrograman\tugas10\latihan>
```

SOAL 2

Latihan 10.2

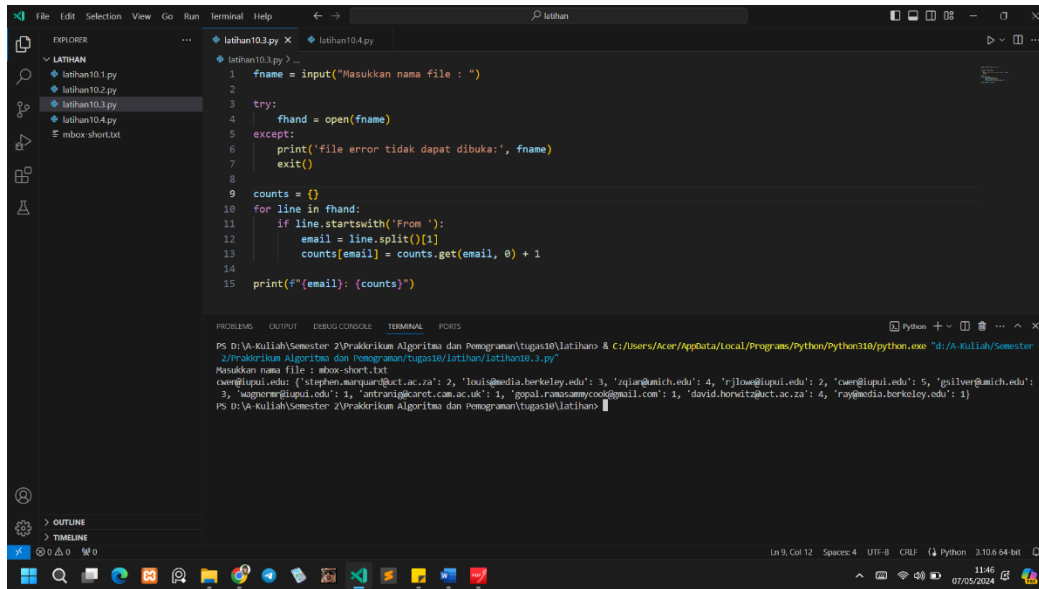


```
1 lista = ['red', 'green', 'blue']
2 listb = ['#FF0000', '#008000', '#0000FF']
3
4 hasil = {}
5 for lista, listb in zip(lista, listb):
6     hasil[lista] = listb
7
8 print(f"hasilnya {hasil}")
```

```
PS D:\A-kuliah\Semester 2\Praktikum Algoritma dan Pemrograman\tugas10\latihan> & C:/Users/acer/AppData/Local/Programs/Python/Python310/python.exe "d:/A-kuliah/Semester 2/Praktikum Algoritma dan Pemrograman\tugas10\latihan\latihan10.2.py"
hasilnya {'red': '#FF0000', 'green': '#008000', 'blue': '#0000FF'}
PS D:\A-kuliah\Semester 2\Praktikum Algoritma dan Pemrograman\tugas10\latihan>
```

SOAL 3

Latihan 10.3



```
1 fname = input("Masukkan nama file : ")
2
3 try:
4     fhand = open(fname)
5 except:
6     print('file error tidak dapat dibuka:', fname)
7     exit()
8
9 counts = {}
10 for line in fhand:
11     if line.startswith('From '):
12         email = line.split()[1]
13         counts[email] = counts.get(email, 0) + 1
14
15 print(f'{email}: {counts}')
```

PS D:\A-Kuliah\Semester 2\Praktikum Algoritma dan Pemrograman\tugas10\latihan> C:\Users\acer\AppData\Local\Programs\Python\Python310\python.exe "d:/A-Kuliah/Semester 2/Praktikum Algoritma dan Pemrograman\tugas10\latihan\latihan10.3.py"

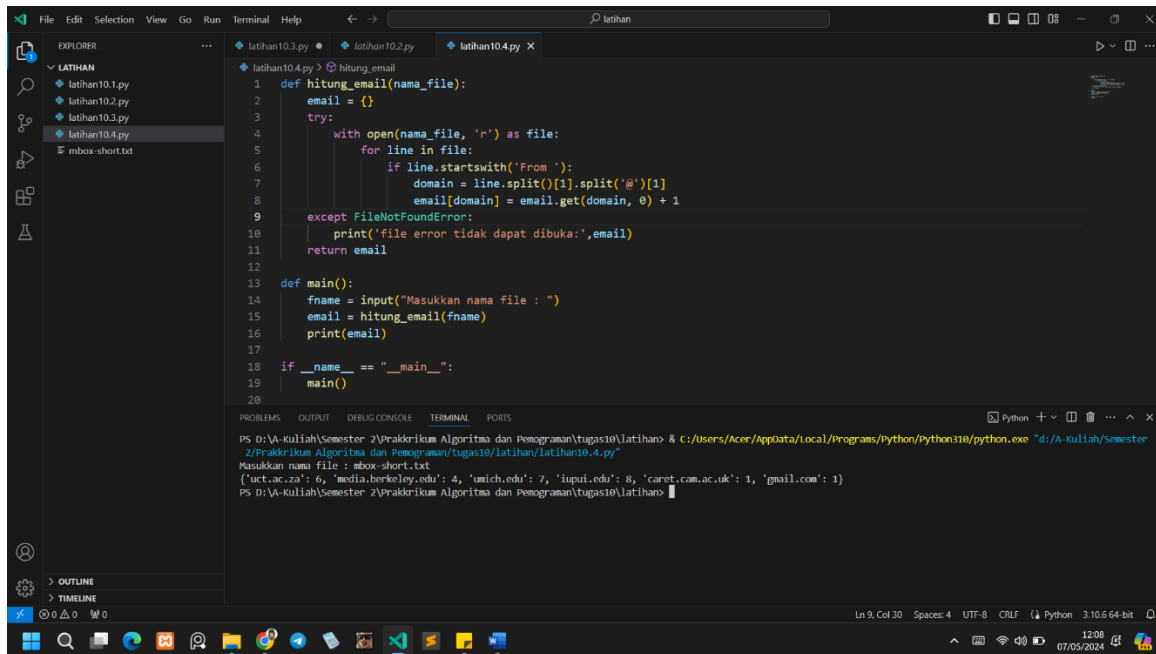
Masukkan nama file : mbox-short.txt

csw@iupui.edu: 2, 'stephen.marquard@uct.ac.za': 2, 'louis@media.berkeley.edu': 3, 'qjar@mit.edu': 4, 'rjlw@iupui.edu': 2, 'csw@iupui.edu': 5, 'psllw@mit.edu': 3, 'wagner@iupui.edu': 1, 'antranig@caret.cam.ac.uk': 1, 'gpall.ramcsammycook@gmail.com': 1, 'david.horvitz@uct.ac.za': 4, 'ray@media.berkeley.edu': 1

PS D:\A-Kuliah\Semester 2\Praktikum Algoritma dan Pemrograman\tugas10\latihan>

SOAL 4

Latihan 10.4



```
1 def hitung_email(nama_file):
2     email = {}
3     try:
4         with open(nama_file, 'r') as file:
5             for line in file:
6                 if line.startswith('From '):
7                     domain = line.split()[1].split('@')[1]
8                     email[domain] = email.get(domain, 0) + 1
9     except FileNotFoundError:
10        print('file error tidak dapat dibuka:', email)
11    return email
12
13 def main():
14    fname = input("Masukkan nama file : ")
15    email = hitung_email(fname)
16    print(email)
17
18 if __name__ == "__main__":
19    main()
20
```

PS D:\A-Kuliah\Semester 2\Praktikum Algoritma dan Pemrograman\tugas10\latihan> C:\Users\acer\AppData\Local\Programs\Python\Python310\python.exe "d:/A-Kuliah/Semester 2/Praktikum Algoritma dan Pemrograman\tugas10\latihan\latihan10.4.py"

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.com': 1}

PS D:\A-Kuliah\Semester 2\Praktikum Algoritma dan Pemrograman\tugas10\latihan>

Link Github : <https://github.com/EchoGinDev/tugasAlpro10.git>