



Laporan Praktikum Algoritma dan Pemrograman

Semester Genap 2023/2024

NIM	71231042
Nama Lengkap	Revaldo Fransisco Hohary
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%)

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

A. Dictionary

Kita tahu bahwa daftar tipe data adalah kumpulan atau rangkaian dari berbagai tipe data dalam dunia program, seperti dictionary tipe data. Terlepas dari itu, ada beberapa perbedaan yang signifikan. Dictionary terdiri dari pasangan kunci "nilai". Kunci tidak boleh sama dalam dictionary; kunci harus berbeda. Dictionary juga dapat didefinisikan sebagai pemetaan antara kumpulan nilai dan indeks. Setiap kunci memiliki nilai. Pasangan nilai kunci, atau "item", adalah istilah yang digunakan untuk menggambarkan asosiasi kunci dan nilai tersebut. Sebagai contoh, kita akan membuat kamus yang menghubungkan kata-kata dalam bahasa Inggris ke bahasa Spanyol. Dengan demikian, kita dapat menganggap bahwa string data adalah kunci dan kata-kata dalam bahasa Spanyol memiliki arti masing-masing. Dictionnaire baru yang kosong dapat dibuat dengan menggunakan fungsi dict. Dictum tidak boleh digunakan sebagai nama variabel karena merupakan fitur bawaan Python.

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

Kosong diwakili dengan tanda kurung kurawal {}, dan kurung kotak [] dapat digunakan untuk menambahkan item ke dalam dictionary.

```
>>> 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'}
```

Format input juga digunakan untuk output. Misalnya, kita membuat dictionary baru dengan tiga item dan mencetak hasilnya, yang merupakan keseluruhan data dictionary.

```
>>> 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'
```

Karena kunci "dua" selalu dipasangkan dengan nilai "dos", urutan item tidak terlalu berpengaruh. Pengecualian akan diberikan jika menggunakan kata-kata yang tidak ditemukan dalam dictionary:

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

Dictionary dapat mengembalikan jumlah pasangan nilai kunci dengan fungsi len:

```
>>> len(eng2sp)
3
```

Dictionary operator mengembalikan nilai benar (true) atau salah (false) sesuai dengan kuncinya.

```
>>> 'one' in eng2sp
True
>>> 'uno' in eng2sp
False
```

Metode nilai dapat digunakan untuk mengetahui nilai yang ada dalam dictionary. Metode ini akan mengembalikan nilai yang sesuai dengan tipe data, diubah ke dalam daftar, dan digunakan dalam operator in.

```
>>> vals = list(eng2sp.values())
>>> 'uno' in vals
True
```

Operator in menggunakan algoritma yang berbeda baik untuk list maupun dictionary. Untuk list, algoritma pencarian linear digunakan, sehingga waktu yang dibutuhkan untuk mencari list menjadi lebih lama seiring dengan panjang list. Untuk dictionary, Python menggunakan algoritma yang disebut Hash Table, yang memiliki fitur yang luar biasa, dan operator in membutuhkan waktu yang sama. Untuk memprosesnya tanpa memperdulikan banyak item dictionary.

B. 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 tiap huruf pada alfabet, kemudian memasukkannya dalam string untuk masing-masing karakter, menambah perhitungan yang sesuai dan menggunakan kondisional 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, Kkarakter dapat ditambahkan item kedalam dictionary dan kemudian ditambahkan nilai dari item yang ada.

Kita akan melakukan perhitungan yang sama dengan ketiga opsi di atas, tetapi kita akan menerapkan proses perhitungan dengan cara yang berbeda.

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. Setiap kali looping terjadi, jika karakter c tidak ada dalam dictionary, maka akan dibuat item baru dengan kunci c dan nilai awal 1, asumsinya telah muncul sekali. Penambahan d(c) akan dilakukan secara otomatis jika c sudah ada dalam dictionary.

Hasil outputnya sebagai berikut :

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

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 outputnya :

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

C. 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 melakukan 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.

Proses perhitungan setiap kata dijamin oleh kombinasi dua perulangan bersarang.

```
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.

Data dump jumlah semua dalam urutan hash yang tidak disortir akan diterima saat program dijalankan.

D. Looping dan Dictionary

Dictionary akan mencari semua kunci yang ada di dalamnya dan mencetak setiap kunci sesuai dengan hubungan nilainya dalam statement for.

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

Hasil dari outputnya 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])
```

Hasil Outputnya :

```
jan 100
annie 42
```

Untuk mencetak kunci dalam urutan alfabet, langkah pertama adalah membuat daftar kunci dalam kamus dengan menggunakan metode kunci yang tersedia pada objek kamus. Langkah selanjutnya adalah melakukan pengurutan (sort), list, dan loop melalui list yang diurutkan. Terakhir, Anda melihat setiap kunci dan mencetak pasangan nilai kunci yang sudah diurutkan. Berikut 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])
```

Hasil Ouputnya :

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

E. Advanced Text Parsing

Pada bagian sebelumnya, kami menggunakan contoh file yang memiliki kalimat di dalamnya. tanda bacanya sudah dihilangkan. Dalam bagian ini, kita akan menggunakan file dengan tanda baca penuh. Dengan contoh romeo.txt dengan tanda baca lengkap.

```
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 11.1 : Romeo.txt

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. Ini juga berlaku untuk huruf kapital. Sebagai ilustrasi, kata-kata "who" dan "who" dianggap berbeda dan diucapkan secara terpisah.

Selain itu, metode string lain, seperti penurunan, punctuation, dan translasi, dapat digunakan untuk menyediakan model penyelesaian tambahan. Jika Anda memperhatikan, metode string translate adalah metode string yang paling rumit. Dokumentasi metode penerjemahan dapat ditemukan di sini.

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

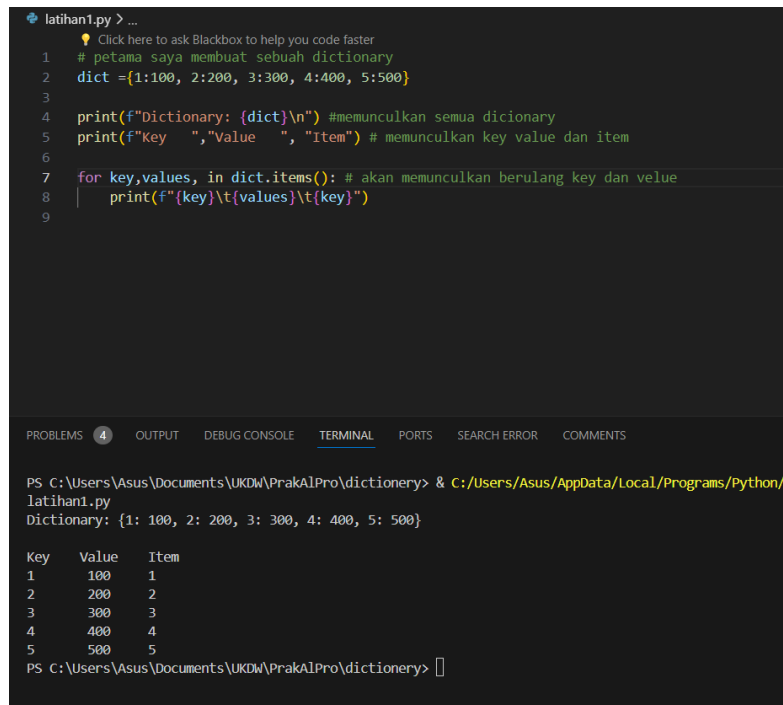
Ganti karakter pada fromstr dengan karakter pada posisi yang sama dengan tostr dan hapus semua karakter yang ada dalam deletetr. Untuk fromstr dan tostr dapat berupa string kosong dan untuk parameter pada deletetr dapat dihilangkan. Kita tidak akan menentukan tostr tetapi kita akan menggunakan parameter deletetr untuk menghapus semua tanda baca. Secara otomatis Python akan memberitahu bagian mana yang dianggap sebagai "tanda baca".

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

Pada soal yan pertama kita disuruh menyusun dictionary dan memunculkan seperti ouput pada terminal di vs code di gambar berikut



The screenshot shows a VS Code editor with a Python file named 'latihan1.py'. The code defines a dictionary and prints its contents in a specific format. The terminal output shows the dictionary and a table of its items.

```
latihan1.py > ...  
Click here to ask Blackbox to help you code faster  
1 # pertama saya membuat sebuah dictionary  
2 dict = {1:100, 2:200, 3:300, 4:400, 5:500}  
3  
4 print(f"Dictionary: {dict}\n") #memunculkan semua dictionary  
5 print(f"Key   ", "Value   ", "Item") # memunculkan key value dan item  
6  
7 for key,values, in dict.items(): # akan memunculkan berulang key dan velue  
8 |     print(f"{key}\t{values}\t{key}")  
9
```

PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR COMMENTS

PS C:\Users\Asus\Documents\UKDW\PrakAlPro\dictionary> & C:/Users/Asus/AppData/Local/Programs/Python/latihan1.py
Dictionary: {1: 100, 2: 200, 3: 300, 4: 400, 5: 500}

Key	Value	Item
1	100	1
2	200	2
3	300	3
4	400	4
5	500	5

PS C:\Users\Asus\Documents\UKDW\PrakAlPro\dictionary> █

SOAL 2

```
latihan2.py > ...
  Click here to ask Blackbox to help you code faster
1  # disini saya membuat dua list x dan y
2  x = ['blue', 'kuning', 'hitam']
3  y = ['#0000FF', '#FFFF00', '000000']
4  dict = {} # membuat variabel untuk dictionary
5
6  for i in range(3): # mengulang sebanyak 3
7      dict[x[i]] = y[i]
8  print(f"List A : {x}") # memunculkan x dan y juga memunculkan dictionary
9  print(f"List B : {y}")
10 print(dict)
```

PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR COMMENTS

```
PS C:\Users\Asus\Documents\UKDW\PrakAlPro\dictionary> & C:/Users/Asus/AppData/Local/Programs/Python/Python312/python.exe c:/Users/Asus/Documents/UKDW/PrakAlPro/dictionary/latihan2.py
List A : ['blue', 'kuning', 'hitam']
List B : ['#0000FF', '#FFFF00', '000000']
{'blue': '#0000FF', 'kuning': '#FFFF00', 'hitam': '000000'}
PS C:\Users\Asus\Documents\UKDW\PrakAlPro\dictionary> []
```

SOAL 3

```
latihan3.py > ...
  Click here to ask Blackbox to help you code faster
1  file = input("Masukan nama file : ") # meminta input file dari pengguna
2  handle = open(file) # membuka file
3  dicto = dict()
4
5  for barisan in handle: # untuk barisan di handle
6      if barisan.startswith("From "): # jika barisan dimulai dari from
7          kat = barisan.split() # split setiap barisan
8          email = kat[1]
9
10         dicto[email] = dicto.get(email, 0) + 1
11
12 print(dicto) # memunculkan variabel untuk dictionary
13 handle.close() # menutup
```

PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR COMMENTS

```
PS C:\Users\Asus\Documents\UKDW\PrakAlPro\dictionary> & C:/Users/Asus/AppData/Local/Programs/Python/Python312/python.exe c:/Users/Asus/Documents/UKDW/PrakAlPro/dictionary/latihan3.py
Masukan 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\Asus\Documents\UKDW\PrakAlPro\dictionary> []
```

SOAL 4

```
latihan4.py > ...  
Click here to ask Blackbox to help you code faster  
1 file = input("Masukan nama file : ") # meminta input file pengguna  
2 handle = open(file) # membuka file yang sudah dimasukan  
3 dicto = dict() # membuat dictionary  
4  
5 for baris in handle: # untuk barisan di handle  
6     if baris.startswith("From "): # jika baris dimulai dari from  
7         kat = baris.split() # di split perbaris  
8         email = kat[1]  
9  
10        smbol = email.find('@') # menemukan simbol @  
11        domain= email[smbol + 1:]  
12  
13        dicto[domain] = dicto.get(domain, 0) + 1  
14  
15 print(dict) # memunculkan dictionarynya  
16 handle.close() # menutup file  
  
PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR COMMENTS  
  
PS C:\Users\Asus\Documents\UKDW\PrakAlPro\dictionary> & C:/Users/Asus/AppData/Local/Pr  
latihan4.py  
Masukan nama file : mbox-short.txt  
<class 'dict'>  
PS C:\Users\Asus\Documents\UKDW\PrakAlPro\dictionary> 
```

F. Link GitHub

<https://github.com/Frealy0901/Tugas-PrakAlPro11.git>