Laporan Praktikum Algoritma dan Pemrograman

Semester Genap 2023/2024

NIM	**71231005**
Nama Lengkap	**Josephine Marcelia**
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

Dictionary mirip dengan list, namun lebih fleksibel. Dalam list, indeks harus berupa integer sedangkan dalam dictionary indeks bisa dapat berupa apapaun. Dictionary terdiri dari pasangan kunci:nilai, di mana kunci harus unik dalam satu dictionary. Dictionary terdiri dari pasangan kunci: nilai, di mana kunci harus unik dan tidak boleh ada yang sama. Dictionary bisa dianggap sebagai peta antara kunci dan nilai, dengan kunci yang menunjukkan ke nilai tertentu. Contohnya, kita bisa membuat kamus yang menghubungkan kata-kata Inggris ke kata-kata Spanyol. Fungsi dict bisa kita pakai untuk membuat dictionary kosong, tetapi kita tidak boleh menggunakan nama dict sebagai variabel karena itu merupakan fungsi bawaan Python.

Berikut merupakan contohnya :

```
engind = {'dog': 'anjing', 'cat': 'kucing', 'duck': 'bebek'}
print(engind)
print(engind["cat"])
print(engind["bebek"])
```

Output :

```
{'dog': 'anjing', 'cat': 'kucing', 'duck': 'bebek'}
kucing
Traceback (most recent call last):
  File "c:\Users\josep\OneDrive\Documents\712310005-jose-list\laprak10.py", line 4, in <module>
    print(engind["bebek"])
          ~~~~~~^^^^^^^^^
KeyError: 'bebek'
```

Pada contoh diatas bisa kita lihat bahwa untuk mencetak dictionary engind kita harus memanggil kuncinya, bukan valuenya. Jika kita memanggil valuenya maka akan muncul pesan error.

Pengurutan pasangan kunci-nilai dalam dictionary juga tidak bisa diprediksi dan berbeda-beda. Namun hal ini bukanlah masalah besar karena dictionary menggunakan kunci, bukan indeks numerik, untuk menemukan nilai yang sesuai.

Fungsi len( ) pada dictionary di Python digunakan untuk menghitung berapa banyak pasangan kunci-nilai yang ada dalam dictionary. Fungsi ini memberikan total jumlah item yang tersimpan di dalam dictionary.

Contoh :

```
warna = {"blue": "biru", "black": "hitam", "red": "merah"}
total_warna = len(warna)
print(f"Total warna di kamus: {total_warna}")
```

Output :

```
Total warna di kamus: 3
```

Operator in pada Python digunakan untuk memeriksa keberadaan kunci dalam sebuah dictionary. Operator ini akan mengembalikan nilai True jika kunci yang dicari ada di dalam dictionary, dan False jika tidak ada.

Operator in menggunakan algoritma yang berbeda pada list dan dictionary. Untuk list, operator in menjalankan pencarian linear, jadi waktu pencariannya akan lebih lama jika listnya panjang. Sementara itu, dalam dictionary, Python menggunakan hash table sehingga waktu prosesnya tetap cepat, tidak peduli seberapa besar dictionary tersebut.

Contoh :

```
biodata = {"nama": "Jose", "umur": 20, "agama": "kristen"}

apakah_ada_nama = "nama" in biodata
print(f"Apakah ada kunci 'nama': {apakah_ada_nama}")

apakah_ada_hobi = "hobi" in biodata
print(f"Apakah ada kunci 'hobi': {apakah_ada_hobi}")
```

Output :

```
Apakah ada kunci 'nama': True
Apakah ada kunci 'hobi': False
```

Untuk memperoleh nilai dari dictionary, kita dapat menggunakan metode values( ). Metode ini akan mengembalikan nilai-nilai dalam dictionary sesuai dengan tipe data aslinya, mengonversinya menjadi list, dan bisa digunakan bersama operator in.

Contoh :

```
my_pet = {"nama": "ruby", "jenis": "husky", "gender": "female"}

nilai = my_pet.values()
print(f"Semua nilai dalam dictionary: {nilai}")
```

Output :

```
Semua nilai dalam dictionary: dict_values(['ruby', 'husky', 'female'])
True
```

Operator in menggunakan algoritma yang berbeda pada list dan dictionary. Untuk list, operator in menjalankan pencarian linear, jadi waktu pencariannya akan lebih lama jika listnya panjang. Sementara itu, dalam dictionary, Python menggunakan hash table sehingga waktu prosesnya tetap cepat, tidak peduli seberapa besar dictionary tersebut.

Dictionary sebagai set penghitung (counters)

Penggunaan dictionary sebagai penghitung (counters) huruf dalam string melibatkan beberapa metode :

1. Membuat 26 variabel, masing-masing untuk satu huruf dalam alfabet, dan menambahkan jumlah kemunculan huruf dalam string ke variabel yang sesuai.
2. Menggunakan list berisi 26 elemen, mengonversi setiap huruf menjadi angka, dan menggunakan angka sebagai indeks list untuk menambah jumlah kemunculan huruf.
3. Membuat kamus dengan huruf sebagai kunci dan jumlah kemunculan sebagai nilai. Huruf dapat ditambahkan ke dalam kamus dan nilai huruf yang ada ditambah sesuai dengan kemunculannya.

Ketiga opsi di atas memberikan perhitungan yang sama, tetapi dengan pendekatan berbeda. Metode dengan dictionary lebih praktis karena kita tidak perlu mengetahui huruf mana yang akan muncul sebelumnya, kita cukup menyediakan ruang untuk huruf yang muncul.

Berikut ini merupakan contoh penerapannya :

```
def hitung_char(kalimat):
    hasil = {}

    for char in kalimat:
        if char != ' ':
            if char in hasil:
                hasil[char] += 1
            else:
                hasil[char] = 1

    return hasil

contoh_kalimat = "mau makan"
penghitung = hitung_char(contoh_kalimat)

for char, jumlah in penghitung.items():
    print(f"Huruf '{char}' muncul {jumlah} kali")
```

Output :

```
Huruf 'm' muncul 2 kali
Huruf 'a' muncul 3 kali
Huruf 'u' muncul 1 kali
Huruf 'k' muncul 1 kali
Huruf 'n' muncul 1 kali
```

Program diatas akan mengiterasi setiap karakter dalam kalimat. Jika karakter bukan spasi, maka program akan memperbarui kemunculan setiap karakter dalam sebuah kalimat. Variabel hasil akan menyimpan menyimpan karakter-karakter yang ada dalam kalimat beserta frekuensinya. Jika karakter sudah ada dalam dictionary program akan menambah nilai frekuensinya, namun jika tidak program akan

menambah nilai dengan frekuensi 1. Program diatas kemudian akan mencetak setiap karakter dalam variabel penghitung dan frekuensi kemunculannya dengan format “Huruf ‘a’ muncul ‘b’ kali, Dimana a adalah karakter dan b adalah jumlah kemunculannya.

Dictionary dan File

Dictionary dapat kita gunakan untuk menghitung frekuensi kemunculan kata dalam file teks.

Pertama-tama kita akan akan menulis program Python yang membaca file, memecah baris-barisnya menjadi daftar kata, lalu menghitung frekuensi setiap kata menggunakan dictionary. Program ini menggunakan dua perulangan: perulangan luar membaca baris file, sedangkan perulangan dalam iterasi melalui kata-kata di setiap baris. Pola ini dapat kita sebut nested loop, di mana perulangan dalam berjalan lebih cepat karena memproses setiap kata di baris, sementara perulangan luar berjalan lebih lambat karena mengiterasi setiap baris file. Kombinasi perulangan ini kita gunakan untuk memastikan setiap kata di setiap baris file dihitung dengan tepat.

puisi.txt

```
Soft whispers of dawn paint the sky
The world wakes to a gentle sigh
Golden rays touch the silent earth
A new day blooms with quiet rebirth
```

```
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)
```

Pada contoh program diatas, pertama-tama kita akan meminta user untuk memasukkan nama file, kemudian mencoba membuka file tersebut. Jika file tidak dapat dibuka, program akan mencetak pesan error dan keluar. Jika file berhasil dibuka, program akan membuat dictionary untuk menghitung frekuensi kemunculan setiap kata dalam file. Program diatas akan membaca file baris demi baris, memisahkan kata-kata dalam setiap baris, dan memperbarui dictionary dengan menambah jumlah kemunculan setiap kata. Jika kata belum ada dalam dictionary, program akan menambahkannya dengan

nilai 1, sedangkan jika kata sudah ada, maka program akan menambah nilai tersebut dengan satu. Kemudian program akan mencetak dictionary yang berisi jumlah kemunculan setiap kata.

Outputnya :

```
Enter the file name: puisi.txt
{'Soft': 1, 'whispers': 1, 'of': 1, 'dawn': 1, 'paint': 1, 'the': 2, 'sky': 1, 'The': 1, 'world': 1, 'wakes': 1, 'to': 1, 'a': 1, 'gentle': 1,
en': 1, 'rays': 1, 'touch': 1, 'silent': 1, 'earth': 1, 'A': 1, 'new': 1, 'day': 1, 'blooms': 1, 'with': 1, 'quiet': 1, 'rebirth': 1}
```

Looping dan Dictionary

Dalam python, kita dapat menggunakan pernyataan for untuk menelusuri kunci-kunci dalam dictionary. Loop ini akan mencetak setiap kunci beserta nilai yang terkait dengannya.

```
ages = { 'mark' : 15 , 'juliet' : 27, 'adam': 22}
for key in ages:
    print(key, ages[key])
```

Output :

```
mark 15
juliet 27
adam 22
```

Pada contoh diatas, kita bisa melihat bahwa output menunjukkan bahwa kunci dalam dictionary tidak mengikuti urutan tertentu.

Kita bisa mengatur pola ini dengan menggunakan berbagai cara looping yang telah kita bahas sebelumnya.

Misalnya, untuk menemukan semua entri dalam dictionary dengan nilai lebih dari 20, kita dapat menggunakan kode program seperti ini:

```
ages = { 'mark' : 15 , 'juliet' : 27, 'adam': 22}
for key in ages:
    if ages[key] > 20 :
        print(key, ages[key])
```

Outputnya hanya akan mencetak nilai yang lebih dari 20 seperti berikut :

```
juliet 27
adam 22
```

Advenced Text Parsing

Sebelumnya, kita menggunakan file yang isinya sudah bersih dari tanda baca. Kali ini, kita akan bekerja dengan file yang masih memiliki tanda baca seperti berikut :

puisi.txt

```
Beneath the moon's soft silver light!
Why do shadows dance in the night?
A gentle breeze whispers a song!
Will we find where our dreams belong?
```

Saat menghitung frekuensi kata-kata dalam teks menggunakan dictionary, kita harus memperhatikan bahwa kata-kata dengan bentuk yang berbeda (misalnya, dengan atau tanpa tanda baca) dianggap berbeda dalam dictionary. Python memiliki fungsi split() yang memisahkan teks berdasarkan spasi dan memperlakukan setiap kata sebagai token yang terpisah. Namun, fungsi ini tidak memisahkan tanda baca yang melekat pada kata-kata, sehingga "song!" dan "song" akan dianggap sebagai dua kata yang berbeda.
Untuk mengatasi hal ini, kita dapat menggunakan metode translate( ) bersama dengan string.punctuation dari modul string. Metode translate() memungkinkan kita untuk mengganti karakter dalam string, dan dalam kasus ini kita akan menggunakannya bersama string.punctuation untuk menghapus semua tanda baca dari teks.

Berikut ini merupakan contoh penggunaanya :

```python
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 :

```
Enter the file name: puisi.txt
{'beneath': 1, 'the': 2, 'moons': 1, 'soft': 1, 'silver': 1, 'light': 1, 'why': 1, 'do': 1, 'shadows': 1, 'dance': 1, 'in': 1, 'night': 1, 'a': 2, 'gentle': 1,
 'breeze': 1, 'whispers': 1, 'song': 1, 'will': 1, 'we': 1, 'find': 1, 'where': 1, 'our': 1, 'dreams': 1, 'belong': 1}
```

Output dari program di atas akan menghitung seberapa sering setiap kata muncul dalam teks 'puisi.txt' , melakukan penghapusan tanda baca, dan mengubah setiap kata menjadi huruf kecil sebelum menghitung frekuensinya.

BAGIAN 2: LATIHAN MANDIRI

Latihan 10.1

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

print("key".ljust(5), "value".ljust(7), "item")
for key, value in dictionary.items():
    print(str(key).ljust(5), str(value).ljust(7), str(key))
```

Output :

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

Penjelasan :

Pertama-tama saya mendefinisikan dictionary dengan pasangan key dan value. Lalu, saya mencetak header "key", "value", dan "item" dengan spasi yang konsisten di antara masing-masing kolom.

Kemudian, saya mengulangi setiap pasangan key-value dalam dictionary menggunakan pernyataan for key, value in dictionary.items(). Dalam setiap iterasi, program diatas akan mencetak key, value, dan item (yang sama dengan key dalam kasus ini). Saya menggunakan metode ljust() untuk mengatur jarak teks sehingga tampilan data lebih rapi dan konsisten di antara kolom-kolom tersebut.

Latihan 10.2

```
def gabung_dict(lista, listb):
    kamus = dict(zip(lista, listb))

    urutan_warna = ['green', 'blue', 'red']
    nomor_warna = {key: kamus[key] for key in urutan_warna}
    return nomor_warna

lista = ['red', 'green', 'blue']
listb = ['#FF0000', '#008000', '#0000FF']

dict = gabung_dict(lista, listb)
print(dict)
```

Output :

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

Penjelasan :

Pada program diatas, saya menggunakan fungsi zip untuk menggabungkan lista dan listb menjadi pasangan-pasangan key-value, kemudian saya mengubahnya menjadi dictionary yang disebut kamus.

Selanjutnya, saya mendefinisikan sebuah daftar urutan_warna yang berisi urutan warna-warna spesifik yang seperti hasil output yang tertera di soal, yaitu 'green', 'blue', dan 'red'. Saya kemudian menggunakan dict comprehension untuk membuat dictionary baru yang hanya berisi key-key dari urutan_warna dengan mengambil nilai yang sesuai dari kamus. Dictionary baru ini, yang disebut nomor_warna, kemudian saya kembalikan sebagai hasil fungsi gabung_dict.

Lalu, saya memanggil fungsi gabung_dict dengan lista dan listb sebagai argumen, dan mencetak hasil dictionary yang saya dapatkan.

Latihan 10.3

```
file_name = input("Masukkan nama file: ")
emails = dict()

with open(file_name, 'r') as file:
    for line in file:
        if line.startswith('From'):
            kata = line.split()
            email = kata[1]
            emails[email] = emails.get(email, 0) + 1
print(emails)
```

Output :

```
Masukkan nama file: mbox-short.txt
{'stephen.marquard@uct.ac.za': 4, 'louis@media.berkeley.edu': 6, 'zqian@umich.edu': 8, 'rjlowe@iupui.edu': 4, 'cwen@iupui.edu': 10, 'gsilver@umich.edu': 6, 'wa
gnermr@iupui.edu': 2, 'antranig@caret.cam.ac.uk': 2, 'gopal.ramasammycook@gmail.com': 2, 'david.horwitz@uct.ac.za': 8, 'ray@media.berkeley.edu': 2}
```

Penjelasan :

Dengan menggunakan loop for, program diatas akan membaca file baris demi baris. Jika sebuah baris dimulai dengan kata "From", program akan membagi baris tersebut menjadi kata-kata terpisah menggunakan metode split( ). Dari kata-kata yang terbagi, program akan mengambil kata kedua, yang merupakan alamat email, dan menyimpannya dalam variabel email.

Kemudian, program akan memperbarui dictionary emails dengan menambahkan 1 pada jumlah kemunculan alamat email tersebut. Saya menggunakan metode get( ) dengan default nilai 0 untuk mengambil nilai frekuensi alamat email dari dictionary, lalu menambahkannya dengan 1 jika alamat email sudah ada, atau memulainya dari 1 jika belum ada.

lalu, setelah program selesai membaca file, program akan mencetak isi dictionary emails yang menunjukkan frekuensi kemunculan setiap alamat email yang ditemukan dalam file tersebut.

Latihan 10.4

```
file_name = input("Masukkan nama file: ")
emails = dict()

with open(file_name, 'r') as file:
    for line in file:
        if line.startswith('Author'):
            kata = line.split()
            email = kata[1]
            pencarian = email.find('@')
            pengiriman = email[pencarian + 1:]
            emails[pengiriman] = emails.get(pengiriman, 0)+1
print(emails)
```

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.com': 1}
```

Penjelasan :

Pertama-tama program diatas akan membuka file dengan nama yang sudah user inputkan dalam mode baca dan membaca file baris demi baris menggunakan loop.

Saya menggunakan metode split( ) sehingga program akan membagi baris menjadi kata-kata terpisah jika sebuah baris dimulai dengan kata "Author".Lalu, program mengambil kata kedua dari baris tersebut, yang merupakan alamat email, dan menyimpannya dalam variabel email.

Saya juga menggunakan metode find( ) agar program dapat mencari posisi karakter '@' dalam alamat email. Begitu menemukan posisi '@', program akan mengambil bagian dari alamat email setelah '@', yang merupakan domain email, dan menyimpannya dalam variabel pengiriman.

Kemudian, program akan memperbarui dictionary emails dengan menambahkan 1 pada jumlah kemunculan domain email tersebut. Saya menggunakan metode get( )dengan nilai default 0 untuk mengambil jumlah kemunculan domain email dari dictionary, lalu menambahkannya dengan 1.

Setelah selesai membaca file, program akan mencetak isi dictionary emails yang menunjukkan berapa kali setiap domain email muncul dalam file tersebut.

Link Github :
https://github.com/Josephinemrc/Tugas10-PrakAlpro.git