Laporan Praktikum Algoritma dan Pemrograman

Semester Genap 2023/2024

NIM	**71230977**
Nama Lengkap	**MICHAEL HOSEA**
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.

Materi 1: Pengantar Dictionary

Dalam bahasa pemrograman Python, terdapat berbagai tipe data yang memungkinkan pengguna untuk menyimpan dan mengelola data dengan cara yang berbeda. Salah satu tipe data yang sangat berguna dan sering digunakan adalah tipe *data dictionary.*

Dictionary adalah struktur data yang digunakan untuk mengaitkan kunci (key) dengan nilai (value). Setiap key dalam dictionary harus unik dan hanya dapat memiliki satu nilai yang terkait. Dictionary dalam Python dikenali dengan kurung kurawal {} dan terdiri dari pasangan key-nilai yang dipisahkan oleh tanda koma.

Fungsi *dict* digunakan untuk membuat *dictionary* baru yang kosong. Karena *dict* merupakan *built-in function* dari python, maka penggunaanya perlu dihindari sebagai nama variabel.

```
ini_dictionary_baru = dict()
print(ini_dictionary_baru)
```

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

```
ini_dictionary_baru['one'] = 'uno'
```

Jika diperhatikan potongan kode di atas, terlihat bahwa baris tersebut menciptakan sebuah entri yang memetakan kunci 'satu' ke nilai 'uno'. Dan saat kita mencetak dictionary tersebut, akan terlihat pasangan nilai kunci yang menghubungkan kunci dengan nilainya.

```
print(ini_dictionary_baru)
{'one': 'uno'}
```

Output yang dihasilkan memiliki struktur yang sama dengan format input. Misalnya, jika kita membuat sebuah dictionary baru dengan tiga item, kemudian mencetak hasilnya, maka yang akan ditampilkan adalah seluruh data dari dictionary tersebut, dengan format yang sesuai dengan cara kita membuatnya.

```
ini_dictionary_baru = {'nama': 'Joni', 'umur': 35, 'pekerjaan': 'presiden'}
print(ini_dictionary_baru)
```

```
{'nama': 'Joni', 'umur': 35, 'pekerjaan': 'presiden'}
```

Urutan pasangan kunci-nilai tidak konsisten. Biasanya, urutan item dalam dictionary tidak dapat diprediksi. Ini tidak menjadi masalah utama karena elemen dalam dictionary tidak dapat diakses dengan indeks integer. Sebaliknya, kita menggunakan kunci untuk mengakses nilai yang sesuai (nilai yang sesuai).

```
print(ini_dictionary_baru['nama'])
```

```
Joni
```

Fungsi len pada dictionary digunakan untuk mengembalikan jumlah pasangan kunci-nilai dalam dictionary tersebut.

```
print(len(ini_dictionary_baru))
```

```
3
```

Operator in dalam dictionary mengembalikan nilai True atau False sesuai dengan keberadaan kunci yang dicari dalam dictionary. Jika kunci tersebut ada dalam dictionary, maka hasilnya adalah True, dan jika tidak, hasilnya adalah False.

```
print('nama' in ini_dictionary_baru)  # Output: True
print('alamat' in ini_dictionary_baru)  # Output: False
```

Untuk memperoleh nilai-nilai yang terdapat dalam dictionary, kita dapat menggunakan metode values(). Metode ini menghasilkan nilai-nilai sesuai dengan tipe data aslinya, kemudian dikonversi menjadi list yang dapat digunakan dalam penggunaan operator in.

```
# Membuat dictionary baru
ini_dictionary_baru = {'nama': 'Joni', 'umur': 35, 'pekerjaan':
'presiden'}

# Menggunakan metode values() untuk mendapatkan nilai-nilai dalam
dictionary
nilai_dictionary = ini_dictionary_baru.values()

# Menampilkan nilai-nilai dalam dictionary
print("Nilai-nilai dalam dictionary:", nilai_dictionary)
```

```
# Memeriksa keberadaan nilai tertentu menggunakan operator in
print("Apakah 'Joni' ada dalam nilai-nilai dictionary?", 'Joni' in
nilai_dictionary)  # Output: True
print("Apakah 40 ada dalam nilai-nilai dictionary?", 40 in
nilai_dictionary)  # Output: False
```

Materi 2: Dictionary sebagai set penghitung (counters)

Dictionary memang sering digunakan sebagai set penghitung (counters) dalam Python. Ini terjadi karena kita dapat menggunakan kunci dictionary untuk menyimpan item yang ingin kita hitung dan nilai dictionary untuk menyimpan jumlah kemunculan item tersebut. Berikut adalah contoh penggunaan dictionary sebagai set penghitung:

```
# Mendefinisikan list dengan beberapa angka acak
angka = [1, 2, 3, 4, 1, 2, 3, 1, 2, 1]

# Membuat dictionary kosong sebagai set penghitung
counter = {}

# Menghitung kemunculan setiap angka dalam list
for num in angka:
    if num in counter:
        # Jika angka sudah ada dalam dictionary, tambahkan jumlahnya
        counter[num] += 1
    else:
        # Jika angka belum ada dalam dictionary, inisialisasi dengan
1
        counter[num] = 1

# Menampilkan hasil penghitungan
print("Set penghitung:")
print(counter)
```

Output :

```
Set penghitung:
{1: 4, 2: 3, 3: 2, 4: 1}
```

MATERI 3: Looping dan Dictionary

Looping dan penggunaan dictionary sering kali berjalan bersamaan dalam Python. Kita dapat menggunakan looping untuk mengakses dan memanipulasi data di dalam dictionary. Berikut adalah beberapa contoh penggunaan looping dengan dictionary:

1. Iterasi melalui kunci dan nilai: Kita dapat menggunakan metode items() untuk mendapatkan pasangan kunci-nilai dari dictionary dan kemudian melakukan iterasi melalui pasangan tersebut.

```
# Dictionary

My_dict = {‘nama’: ‘John’, ‘umur’: 30, ‘pekerjaan’: ‘Insinyur’}

# Rumus
for key, value in my_dict.items():
    print(key, "->", value)

# Output :
nama -> John
umur -> 30
pekerjaan -> Insinyur
```

2. Iterasi melalui kunci: Jika kita hanya tertarik dengan kunci-kunci dalam dictionary, kita dapat menggunakan metode keys() untuk mendapatkan daftar kunci-kunci tersebut.

```
for key in my_dict.keys():
    print(key)
```

Output :
nama
umur
pekerjaan

3. Iterasi melalui nilai: Demikian pula, jika kita hanya tertarik dengan nilai-nilai dalam dictionary, kita dapat menggunakan metode values() untuk mendapatkan daftar nilai-nilai tersebut.

```
for value in my_dict.values():
    print(value)
```

Output:
John
30

Insinyur

4. Iterasi menggunakan fungsi zip(): Kita juga dapat menggunakan fungsi zip() untuk menggabungkan kunci dan nilai menjadi pasangan, kemudian melakukan iterasi melalui pasangan tersebut.

```
for key, value in zip(my_dict.keys(), my_dict.values()):
    print(key, "->", value)
```

Output:
nama -> John
umur -> 30
pekerjaan -> Insinyur

MATERI 4: Advanced Text Parsing

Untuk parsing teks ke dalam tipe data dictionary, Anda dapat menggunakan pendekatan yang mirip dengan yang telah kita bahas sebelumnya, tetapi kali ini kita akan menyimpan hasil parsing ke dalam sebuah dictionary.

```
# Teks yang akan di-parse
teks = """
Nama: John Doe
Usia: 30
Pekerjaan: Insinyur
"""

# Membuat dictionary kosong untuk menyimpan hasil parsing
hasil_parsing = {}

# Mem-parsing teks dan menyimpan hasilnya dalam dictionary
lines = teks.strip().split('\n')
for line in lines:
    key, value = line.split(': ', 1)
    hasil_parsing[key] = value

# Menampilkan hasil parsing
print("Hasil parsing teks ke dalam dictionary:")
print(hasil_parsing)
```

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

Buatlah sebuah program untuk mendapatkan nilai key, value, dan item dari sebuah dictionary.

```
Contoh:

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

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

- Source Code dan Output

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

# Cetak header
print("key\tvalue\titem")

# Loop melalui setiap pasangan key-nilai dalam dictionary
for key, value in dictionary.items():
    print(f"{key}\t{value}\t{key}")  # Cetak key, nilai, dan key
lagi sebagai item
```

- Penjelasan

Dalam program ini,

1. Kita mendefinisikan dictionary yang diberikan.
2. Kemudian, kita mencetak header yang menampilkan judul untuk kolom key, value, dan item.
3. Selanjutnya, kita menggunakan loop for untuk mengiterasi melalui setiap pasangan key-nilai dalam dictionary menggunakan metode items().
4. Di dalam loop, kita mencetak nilai key, value, dan item (key lagi) dalam format yang diinginkan.

SOAL 2

Buatlah sebuah program untuk memetakan dua list menjadi satu dictionary.

Contoh:

```
Data List
Lista = ['red', 'green', 'blue']
Listb = ['#FF0000','#008000', '#0000FF']

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

- Source Code dan Output

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

# Membuat bentuk dua list menajdi satu dictionary
result = dict(zip(lista, listb))
print(result)
```

- Penjelasan

  Fungsi hitung_frekuensi_kata(kalimat, kata) yang Anda buat berfungsi untuk menghitung berapa kali sebuah kata tertentu muncul dalam sebuah kalimat dengan cara:

  1. Pertama, kita mendefinisikan dua list: lista yang berisi key (key), dan listb yang berisi nilai (value) yang ingin kita pasangkan dalam dictionary. Misalnya, kita ingin warna-warna ('red', 'green', 'blue') dipetakan ke kode warna HTML ('#FF0000', '#008000', '#0000FF').
  2. Kemudian, kita menggunakan fungsi **zip()** untuk menggabungkan dua list menjadi satu, membentuk pasangan nilai dari setiap elemen list yang sesuai. Kemudian, kita menggunakan konstruktor **dict()** untuk membuat dictionary dari pasangan nilai tersebut. Setiap elemen pada **lista** menjadi key, dan setiap elemen pada **listb** menjadi nilai dalam dictionary.

SOAL 3

Dengan menggunakan file mbox-short.txt, buatlah program yang dapat membaca log email dan sajikan dalam histogram menggunakan dictionary. Kemudian hitung berapa banyak pesan yang masuk dari email dan sajikan dalam bentuk dictionary.

Silakan cek bagian dibawah ini untuk contoh output dari programnya.

```
Masukkan nama file : mbox-short.txt
```

```
{'gopal.ramasammycook@gmail.com': 1, 'louis@media.berkeley.edu': 3,
'cwen@iupui.edu': 5, 'antranig@caret.cam.ac.uk': 1,
'rjlowe@iupui.edu': 2, 'gsilver@umich.edu': 3,
'david.horwitz@uct.ac.za': 4, 'wagnermr@iupui.edu': 1,
'zqian@umich.edu': 4, 'stephen.marquard@uct.ac.za': 2,
'ray@media.berkeley.edu': 1}
```

- Source Code dan Output

```
def count_emails(filename):
    email_counts = {}
    with open(filename, 'r') as file:
        for line in file:
            if line.startswith('From:'):
                email = line.split()[1]
                email_counts[email] = email_counts.get(email, 0) + 1
    return email_counts

filename = input("Masukkan nama file: ")
email_counts = count_emails(filename)
print(email_counts)
```

- Penjelasan

- Fungsi **count_emails(filename)**:
  - Fungsi ini mengambil satu parameter, yaitu **filename**, yang merupakan nama file yang akan dihitung jumlah emailnya.
  - Program membuka file tersebut dan membacanya baris per baris.
  - Setiap baris yang dimulai dengan 'From:' menandakan email pengirim, sehingga program memproses baris tersebut.
  - Dari baris tersebut, program mengambil alamat email pengirim, yang biasanya berada pada indeks kedua setelah 'From:'.
  - Jumlah email dari pengirim tersebut diperbarui dalam dictionary **email_counts**. Jika pengirim sudah ada dalam dictionary, jumlahnya ditambah 1; jika belum, pengirim ditambahkan ke dictionary dengan jumlah 1.
  - Setelah semua baris dalam file dibaca, dictionary **email_counts** yang berisi jumlah email dari setiap pengirim dikembalikan.
- Pengguna diminta untuk memasukkan nama file yang akan dihitung jumlah emailnya menggunakan fungsi **input()**.
- Fungsi **count_emails()** dipanggil dengan argumen nama file yang dimasukkan oleh pengguna.
- Hasilnya berupa dictionary yang berisi jumlah email dari setiap pengirim.
- Dictionary tersebut dicetak untuk ditampilkan kepada pengguna.

SOAL 4

Dengan menggunakan file mbox-short.txt, buat program untuk mencatat data nama domain pengirim pesan. Hitunglah jumlah pesan yang dikirim masing-masing domain. sajikan dalam bentuk dictionary.

Silakan cek bagian dibawah ini untuk contoh output dari programnya.

```
Masukkan nama file: mbox-short.txt
```

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

- Source Code dan Output

```
def count_email_domains(filename):
    email_domains = {}
    with open(filename, 'r') as file:
        for line in file:
            if line.startswith('From '):
                email = line.split()[1]
                domain = email.split('@')[1]
                email_domains[domain] = email_domains.get(domain, 0) + 1
    return email_domains

filename = input("Masukkan nama file: ")
result = count_email_domains(filename)
print(result)
```

- Penjelasan

1. **Fungsi count_emails(filename):**
    - Fungsi ini mengambil satu parameter, yaitu **filename**, yang merupakan nama file yang akan dihitung jumlah emailnya.
    - Pada awalnya, sebuah dictionary **email_counts** dibuat untuk menyimpan jumlah email dari setiap pengirim.
    - File dengan nama yang diberikan dibuka dalam mode baca (**'r'**), dan setiap barisnya dibaca secara berurutan.
    - Jika sebuah baris dimulai dengan string 'From:', artinya itu adalah baris yang berisi alamat email pengirim.
    - Alamat email pengirim diekstrak dari baris tersebut dengan membagi baris menjadi kata-kata dan mengambil kata kedua.

- Jumlah email dari pengirim tersebut diperbarui dalam dictionary **email_counts**. Jika pengirim sudah ada dalam dictionary, jumlahnya ditambah 1; jika belum, pengirim ditambahkan ke dictionary dengan jumlah 1.
- Setelah semua baris dalam file dibaca, dictionary **email_counts** yang berisi jumlah email dari setiap pengirim dikembalikan.

2. **Meminta Input Nama File:**
   - Pengguna diminta untuk memasukkan nama file yang akan dihitung jumlah emailnya menggunakan fungsi **input()**.
3. **Memanggil Fungsi dan Mencetak Hasil:**
   - Fungsi **count_emails()** dipanggil dengan argumen nama file yang dimasukkan oleh pengguna.
   - Hasilnya berupa dictionary yang berisi jumlah email dari setiap pengirim.
   - Dictionary tersebut dicetak untuk ditampilkan kepada pengguna.