|  |  |
| --- | --- |
| A picture containing text, clipart  Description automatically generated | **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: Metode String

1. Metode upper() dan lower()

Metode ini digunakan untuk mengubah string menjadi huruf kapital atau huruf kecil.

string = "Hello, World!"

kapital = string.upper()

kecil = string.lower()

print(kapital) # "HELLO, WORLD!"

print(kecil) # "hello, world!"

1. Metode strip()

Metode ini menghapus karakter spasi (atau karakter lain yang disebut) dari awal dan akhir string.

string = " Hello, World! "

bersih = string.strip()

print(bersih) # "Hello, World!"

1. Metode replace()

Metode ini menggantikan setiap kemunculan string dengan string lain.

string = "Hello, World!"

ganti = string.replace("Hello", "Hi")

print(ganti) # "Hi, World!"

## MATERI 4: Format String

1. Format String dengan “%”

Digunakan untuk menyematkan nilai ke dalam string dengan format tertentu.

nama = "John"

umur = 25

teks = "Nama: %s, Umur: %d" % (nama, umur)

print(teks) # "Nama: John, Umur: 25"

1. Format String dengan format()

Metode yang lebih modern untuk menyisipkan nilai ke dalam string.

nama = "John"

umur = 25

teks = "Nama: {}, Umur: {}".format(nama, umur)

print(teks) # "Nama: John, Umur: 25"

# 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.