

# Struktur Data Dengan Python

Aji Prasetya Wibawa • Felix Andika Dwiyanto  
Triyanna Widiyaningtyas • Wayan Firdaus Mahmudy



TM

# python



# **STRUKTUR DATA DENGAN PYTHON**

**AJI PRASETYA WIBAWA  
FELIX ANDIKA DWIYANTO  
TRIYANNA WIDIYANINGTYAS  
WAYAN FIRDAUS MAHMUDY**



Penerbit:

**AHLIMEDIA PRESS**

# STRUKTUR DATA DENGAN PYTHON

**Penulis:**

Aji Prasetya Wibawa  
Felix Andika Dwiyanto  
Triyanna Widiyaningtyas  
Wayan Firdaus Mahmudy

**Editor:**

Yayuk Umay

**Penyunting:**

Masyrifatul Khairiyyah

**Desain Cover:**

Aditya Rendy T.

**Penerbit:**

Ahlimedia Press (Anggota IKAPI: 264/JTI/2020)  
Jl. Ki Ageng Gribig, Gang Kaserin MU No. 36  
Kota Malang 65138  
Telp: +6285232777747  
Telp Penulis: -  
[www.ahlimediapress.com](http://www.ahlimediapress.com)

**ISBN: 978-623-6351-35-2**

Cetakan Pertama, Juli 2021

Hak cipta oleh Penulis dan Dilindungi Undang-Undang Republik Indonesia Nomor 19 Tahun 2002 Tentang Hak Cipta, Pasal 72.

Dilarang keras menerjemahkan, memfotokopi, atau memperbanyak sebagian atau seluruh isi buku ini tanpa izin tertulis dari Penerbit.

## KATA PENGANTAR

Puji syukur dihaturkan kepada Tuhan Yang Maha Esa atas anugerah berupa ilmu, kesehatan, dan rahmat yang diberikan sehingga buku berjudul *Struktur Data dengan Python* bisa diselesaikan.

Pada pembelajaran struktur data, mahasiswa sering mengalami kesulitan dalam memahami konsep abstrak dari struktur data. Hal tersebut terjadi karena mahasiswa dibingungkan dengan kode program yang rumit dan susah dipahami. Padahal, belajar kode atau bahasa program berbeda dengan belajar logika program.

*Python* merupakan bahasa pemrograman tingkat tinggi yang memiliki berbagai keunggulan. Salah satu keunggulan *python* sebagai bahasa pemrograman adalah *syntax* program yang sederhana dan lebih mudah dimengerti. Selain itu, struktur pemrograman juga tidak jauh berbeda dengan bahasa yang lebih dahulu digunakan, yaitu C++. Harapannya, tidak perlu waktu yang lama untuk bisa menyesuaikan dengan menggunakan bahasa *python* dalam menguasai konsep struktur data.

Buku ini menerapkan konsep CDIO (*Conceive, Design, Implementation, Operation*), sebuah metode pembelajaran yang dikembangkan oleh MIT (Massachusetts Institute of Technology) dan dikhususkan untuk pembelajaran di bidang teknik. Tujuan dari metode ini adalah pembelajaran yang menekankan pada dasar pengetahuan dan kemampuan teknis yang sesuai dengan standar kompetensi. Buku ini dilengkapi dengan pemecahan masalah yang akan memicu kemampuan berpikir lebih kritis, kreatif, dan menumbuhkan sikap kerja sama tim yang diperlukan dalam pemecahan masalah di kehidupan nyata.

Penyusunan buku ini masih jauh dari kesempurnaan. Kritik membangun sangat kami harapkan demi peningkatan kualitas struktur maupun isi buku ini di masa mendatang.

Malang, 18 Desember 2020

Penulis

# DAFTAR ISI

|  |            |
|--|------------|
| <b>KATA PENGANTAR .....</b>                          | <b>iii</b> |
| <b>DAFTAR ISI .....</b>                              | <b>iv</b>  |
| <br>   |            |
| <b>PENDAHULUAN.....</b>                              | <b>1</b>   |
| A. Deskripsi .....                                   | 1          |
| B. Prasyarat .....                                   | 1          |
| C. Standar Kompetensi.....                           | 2          |
| D. Materi yang Dibahas.....                          | 2          |
| E. Deskripsi Kompetensi .....                        | 3          |
| F. Keunggulan Modul .....                            | 4          |
| G. Petunjuk Penggunaan Modul .....                   | 4          |
| H. Langkah-langkah dalam Menyelesaikan Latihan ..... | 6          |
| I. Deskripsi Materi.....                             | 11         |
| <br>   |            |
| <b>BAB 1. PENGANTAR PYTHON.....</b>                  | <b>12</b>  |
| 1.1 Tujuan Pembelajaran .....                        | 12         |
| 1.2 Dasar Teori .....                                | 12         |
| 1.3 Struktur Pemrograman Python.....                 | 13         |
| 1.4 Pengenalan Google Colab.....                     | 19         |
| 1.5 Python Dasar dengan Google Colab.....            | 23         |
| 1.6 Latihan .....                                    | 27         |
| <br>   |            |
| <b>BAB 2. LISTS &amp; TUPLE .....</b>                | <b>29</b>  |
| 2.1 Tujuan Pembelajaran .....                        | 29         |
| 2.2 Lists .....                                      | 29         |
| 2.3 Tuple.....                                       | 34         |

|  |           |
|--|-----------|
| 2.4 Latihan.....   | 37        |
| 2.5 Studi Kasus .....  | 39        |
| <b>BAB 3. DICTIONARY &amp; SET .....</b>                         | <b>40</b> |
| 3.1 Tujuan Pembelajaran.....                                     | 40        |
| 3.2 Dictionary .....   | 40        |
| 3.3 Set.....   | 43        |
| 3.4 Latihan.....   | 46        |
| 3.5 Studi Kasus .....  | 46        |
| <b>BAB 4. STACK &amp; QUEUE.....</b>                             | <b>49</b> |
| 4.1 Tujuan Pembelajaran.....                                     | 49        |
| 4.2 Stack.....   | 49        |
| 4.3 Implementasi Stack pada Python.....                          | 51        |
| 4.4 Queue.....   | 52        |
| 4.5 Implementasi Queue pada Python .....                         | 54        |
| 4.6 Latihan.....   | 57        |
| 4.7 Studi Kasus .....  | 57        |
| <b>BAB 5. LINKED LIST .....</b>                                  | <b>58</b> |
| 5.1 Tujuan Pembelajaran.....                                     | 58        |
| 5.2 Single Linked List .....                                     | 58        |
| 5.3 Struktur Pemrograman Single Linked List pada Python.....     | 60        |
| 5.4 Implementasi Program Single Linked List pada Python (1)..... | 61        |
| 5.5 Implementasi Program Single Linked List pada Python (2)..... | 63        |
| 5.6 Latihan.....   | 66        |
| 5.7 Studi Kasus .....  | 67        |

|   |               |
|---|---------------|
| <b>BAB 6. TREE .....</b>                        | <b>69</b>     |
| 6.1 Tujuan Pembelajaran .....                   | 69            |
| 6.2 Tree .....                                  | 69            |
| 6.3 Istilah dalam Tree .....                    | 69            |
| 6.4 Struktur Pemrograman Tree pada Python ..... | 71            |
| 6.5 Binary Search Tree (BST).....               | 73            |
| 6.6 Tree Traversal .....                        | 74            |
| 6.7 Implementasi Program Tree pada Python ..... | 77            |
| 6.8 Latihan .....                               | 79            |
| 6.9 Studi Kasus .....                           | 80            |
| <br><b>DAFTAR PUSTAKA .....</b>                 | <br><b>81</b> |
| <b>LAMPIRAN.....</b>                            | <b>82</b>     |

# PENDAHULUAN

## A. Deskripsi

Buku ini digunakan sebagai media belajar matakuliah struktur data. Standar kompetensi dari mata kuliah struktur data adalah memberikan pengetahuan tentang konsep struktur data dan algoritma dalam pemrograman serta menerapkan konsep struktur data dan algoritma untuk menyelesaikan masalah-masalah pemrograman. Kompetensi yang diharapkan setelah mempelajari buku ini adalah:

1. Memahami konsep struktur data.
2. Menganalisis permasalahan yang berkaitan dengan konsep struktur data.
3. Menentukan solusi dari permasalahan yang berkaitan dengan struktur data.
4. Membangun program yang berkaitan dengan struktur data.

Dalam kegiatan pembelajaran, modul ini menggunakan metode pembelajaran CDIO (Conceive, Design, Implementation, Operation) dan setiap langkah metode akan dijelaskan pada petunjuk umum.

## B. Prasyarat

Modul ini merupakan modul lanjutan yang memerlukan prasyarat bagi mahasiswa. Adapun prasyarat yang harus dilalui oleh siswa adalah telah melaksanakan dan lulus mata kuliah algoritma pemrograman atau pemrograman dasar.

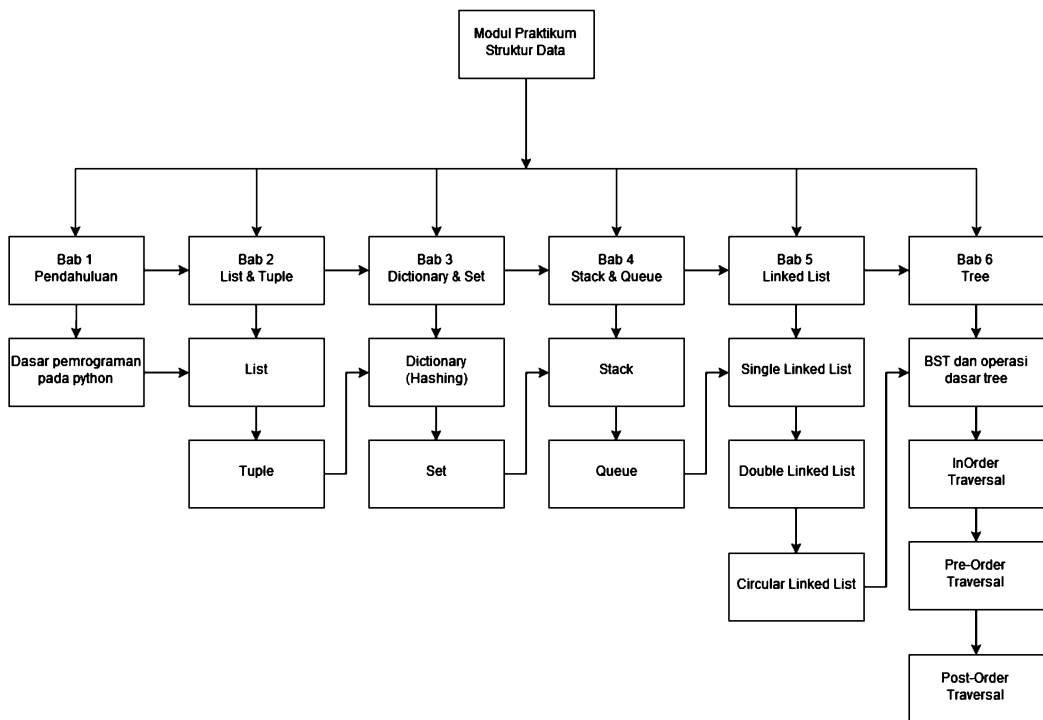


### C. Standar Kompetensi

Memberikan pengetahuan tentang konsep struktur data dan algoritma dalam pemrograman serta menerapkan konsep struktur data dan algoritma untuk menyelesaikan masalah-masalah pemrograman.

### D. Materi yang Dibahas

Materi-materi yang akan dibahas terbagi atas enam bab. Materi dan sub materi yang akan disajikan dijabarkan pada gambar 1 berikut ini. Mahasiswa dapat menggunakannya secara berurutan ataupun memilih bab tertentu sesuai kebutuhan.



Gambar 1. Peta kedudukan modul

## E. Deskripsi Kompetensi

1. Menelaah abstraksi data.
2. Menelaah konsep list, tuple, dictionary, dan set
3. Menganalisis Stack: spesifikasi, representasi, dan aplikasi stack.
4. Menganalisis Queue: spesifikasi queue, representasi queue, dan aplikasi queue.
5. Menganalisis Linked-list: deklarasi, operasi dasar (penciptaan dan penghancuran simpul, inisialisasi linked-list, penyisipan simpul, penghapusan simpul, penelusuran simpul, dan pencarian simpul), dan operasi terhadap linked-list (penghapusan, inversi, penyambungan, serta panjang linked-list).
6. Menganalisis varian singly linked-list: singly linked-list dengan last dan circular linked-list (deklarasi, operasi, implementasi, dan pencarian simpul).
7. Menganalisis Double linked-list: deklarasi dan operasi pada double linked-list.
8. Menganalisis Tree: binary tree, traversing binary tree, algoritma traversal, threads, binary search tree, searching dan inserting dalam binary tree.
9. Membangun program array.
10. Membangun program linked-list.
11. Membangun program Stack & Queue
12. Membangun program tree.

## F. Keunggulan Modul

1. Pada modul ini, kajian teori hanya dijabarkan secukupnya. Modul hanya berisi tentang konsep dasar setiap pokok bahasan, sedangkan teori lebih lanjut disediakan dalam bentuk link, perintah eksplorasi, perintah diskusi, dsb.
2. Soal latihan dan studi kasus yang beragam dan terdiri dari beberapa macam soal.
3. Dari kedua poin di atas, tujuan modul ini mendorong mahasiswa untuk aktif dalam memahami suatu teori atau konsep struktur data dengan cara mencari dan mengeksplorasi konsep atau teori dari berbagai macam sumber dan media. Selain itu, soal latihan yang beragam akan mendorong mahasiswa untuk berfikir beda dan dapat menyelesaikan masalah struktur data dengan cara yang beragam.

## G. Petunjuk Penggunaan Modul

### 1. *Petunjuk Bagi Mahasiswa*

Untuk memperoleh hasil belajar secara maksimal, dalam menggunakan modul ini maka langkah-langkah yang perlu dilaksanakan antara lain:

- a. Pada awal pembelajaran bahan-bahan yang perlu disiapkan adalah memasang python 2 sebagai bahasa pemrograman yang digunakan dan pycharm IDE sebagai perangkat lunak yang digunakan dalam membuat program python.
- b. Pada setiap latihan maupun studi kasus yang dikerjakan selalu gunakan langkah-langkah metode CDIO sesuai petunjuk setiap

jenis latihan atau studi kasus. Adapun langkah penyelesaian CDIO secara utuh adalah sebagai berikut:

| Komponen CDIO      | Langkah-langkah penyelesaian   |
|--------------------|--|
| Concieve           | 1. Membaca dan memahami permasalahan<br>2. Identifikasi pra dan post kondisi dari masalah<br>3. Menentukan beberapa solusi yang mungkin pada masalah |
| Design             | 4. Analisa detail permasalahan<br>5. Tentukan solusi terbaik pada permasalahan   |
| Implementatio<br>n | 6. Buat kode program<br>7. Uji coba program  |
| Operation          | 8. Menampilkan dan Uji coba dengan instruktur disertai perubahan pada program untuk mengetes program sudah berjalan efektif atau belum.              |

Tabel 1. Langkah-langkah penyelesaian soal secara CDIO

- c. Tanyakan hal-hal yang belum dipahami tentang pembelajaran struktur data pada instruktur atau dosen yang bersangkutan.

## 2. *Petunjuk Bagi Instruktur atau Dosen*

- a. Mendampingi mahasiswa dalam kegiatan pembelajaran struktur data.

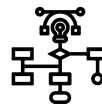
- b. Menjelaskan dan membantu mahasiswa memahami konsep struktur data.
- c. Mengarahkan mahasiswa dalam setiap latihan maupun studi kasus agar sesuai dengan langkah-langkah metode CDIO.
- d. Mengkoordinir mahasiswa dalam tahap akhir uji coba program
- e. Menguji apakah program sudah berjalan dengan efektif atau belum.

## H. Langkah–langkah dalam Menyelesaikan Latihan

Pada modul ini terdapat latihan dan studi kasus pada setiap bab nya. Pada setiap latihan maupun studi kasus terdapat symbol yang merepresentasikan jenis permasalahan dan cara penyelesaian setiap permasalahan tersebut. Untuk menyelesaikan latihan dan studi kasus pada modul ini, kerjakan sesuai dengan langkah-langkah CDIO yang diwakilkan dengan simbol-simbol di bawah ini:



*CONCEIVE*



*DESIGN*

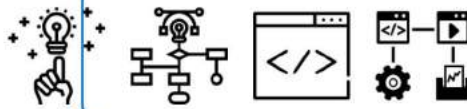


*IMPLEMENTATION*



*OPERATION*

- Contoh Studi Kasus Sebagai Berikut

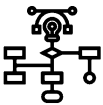


1. Pak Agus adalah seorang petani melon, lahan melon pak Agus memiliki panjang 20 meter dan lebar 10 meter, jika setiap m<sup>2</sup> lahan dapat ditanami 2 melon berapakah jumlah melon pak Agus jika akan dipanen semua?

*Symbol pada bagian atas studi kasus ini terdiri dari symbol Conceive, Design, Implementation, Operation. Hal ini menunjukkan bahwa studi kasus yang dikerjakan harus menggunakan langkah-langkah CDIO secara utuh. berikut ini merupakan contoh dalam menyelesaikan permasalahan sesuai dengan langkah CDIO:*

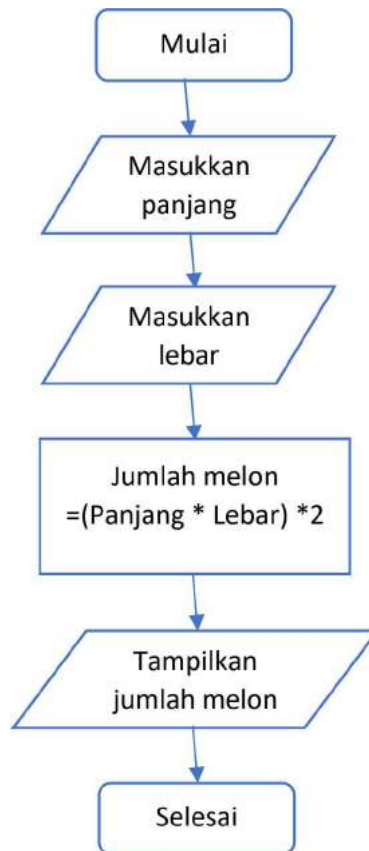
**Langkah Conceive:**

1. Baca dan pahami studi kasus yang diberikan
2. Pre-kondisi: nilai panjang dan lebar lahan pak Agus  
 Post-kondisi: jumlah seluruh melon yang akan dipanen pak Agus
3. Solusi yang mungkin:
  - $(\text{Panjang} * \text{Lebar}) * 2$
  - $(\text{Panjang} * 2) * \text{Lebar}$
  - $(\text{Lebar} * 2) * \text{Panjang}$

**Langkah Design:**

4. Setelah menganalisa detail permasalahan,
5. Menentukan solusi terbaik, yaitu “ $(\text{Panjang} * \text{Lebar}) * 2$ ” dengan alasan penulisan kode program akan lebih rapid an mudah dibaca jika variabel panjang dan lebar dihitung terlebih dahulu kemudian baru dikali dengan 2.
6. Buat algoritma program (flowchart, deskripsi terstruktur, atau pseudocode):

*(Flowchart)*



*(Deskripsi terstruktur)*

1. Masukkan panjang
2. Masukkan lebar
3. Jumlah melon adalah panjang dikali lebar dikali dua
4. Tampilkan jumlah melon pak Agus

*(Pseudocode)*

1. Begin
2. Set variable P as Length



3. Set variable L as width
4. Set variable J as total melon
5. Set the calculate result (value of P \* value of L) \* 2 to J
6. Display J
7. End

### Langkah Implementation:



7. Buat kode program :

```
P = float(input("Masukkan Panjang : "))
L = float(input("Masukkan Lebar : "))
J = (P * L) * 2
print ("Jumlah melon Pak Agus ada", J)
```

8. Uji coba program dengan kelompok kalian

### Langkah Operation:



9. Tampilkan dan uji coba program kalian ke depan kelas. Mintalah pendapat, saran, atau masukkan pada dosen, instruktur, maupun teman kalian yang lain agar program dapat berjalan efektif dan tidak ada kesalahan (bug).

## I. Deskripsi Materi

Materi-materi yang disajikan dalam modul ini dibagi dalam 6 bab, di antaranya:

1. **Bab 1. Pengantar python:** pada bab ini dijelaskan tentang dasar dan pemrograman dasar python.
2. **Bab 2. List & Tuple:** pada bab ini dijelaskan tentang konsep struktur data list dan tuple pada python dan memecahkan masalah menggunakan list dan tuple.
3. **Bab 3. Dictionary & Set:** pada bab ini dijelaskan tentang konsep struktur data dictionary dan set pada python dan memecahkan masalah menggunakan dictionary dan set.
4. **Bab 4. Stack & Queue:** pada bab ini dijelaskan tentang konsep struktur data stack dan queue. Menggunakan operasi dasar dan implementasi stack dan queue dalam pemecahan masalah.
5. **Bab 5. Linked List:** pada bab ini dijelaskan tentang konsep struktur data linked list. Menggunakan operasi dasar dan implementasi linked list dalam pemecahan masalah.
6. **Bab 6. Tree:** pada bab ini dijelaskan tentang konsep struktur data tree. Menggunakan operasi dasar dan implementasi tree dalam pemecahan masalah.

# BAB 1. PENGANTAR PYTHON

## 1.1 Tujuan Pembelajaran

- 1) Memahami struktur bahasa pemrograman Python
- 2) Memahami dasar pemrograman menggunakan Python
- 3) Membuat program sederhana menggunakan Python

## 1.2 Dasar Teori

Struktur data merupakan salah satu elemen penting dalam pemrograman dasar selain algoritma. Pada struktur data, kalian akan belajar bagaimana cara mengatur dan menyimpan data yang akan kalian olah secara efektif dan efisien tergantung permasalahan yang akan kalian pecahkan. Secara umum struktur data dan algoritma dapat diajarkan menggunakan bahasa pemrograman yang berbeda-beda, sebut saja *Delphi*, *C/C++*, *Java*, *Python* dll. Namun dalam beberapa tahun terakhir, banyak perguruan tinggi mulai menggunakan bahasa *Python* pada mahasiswa untuk pemrograman dasar dan pemecahan masalah. Maka dari itu bahan ajar struktur data ini menggunakan bahasa *python* dalam latihan dan pemecahan masalah yang berorientasi pada permasalahan dalam kehidupan sehari-hari.

### Mengapa Menggunakan *Python*?

Python merupakan bahasa pemrograman yang efisien, sintaks program lebih sedikit jika dibandingkan dengan bahasa pemrograman

yang lain, sintaks *python* mirip dengan algoritma pseudocode yang umum, sehingga mahasiswa relatif lebih cepat mempelajari bahasa dan sintaks pemrograman dan lebih banyak waktu yang digunakan untuk memecahkan masalah. Selain itu *python* juga multi platform dan mendukung berbagai pengembangan aplikasi dan perangkat lunak. Dan yang paling penting bahasa ini mudah dipelajari oleh programmer pemula maupun programmer ahli yang ingin bertransisi dari bahasa pemrograman selain *python*.

## 1.3 Struktur Pemrograman Python

### 1.3.1 Aturan Penulisan

Program-program yang ditulis dalam Python secara khas jauh lebih ringkas dibandingkan dengan program-program C atau C++, karena beberapa pertimbangan:

- 1) tipe data tingkat tinggi digunakan untuk menyatakan operasi kompleks dalam suatu statemen tunggal
- 2) pengelompokan statemen telah selesai dengan indentasi sebagai pengganti dari pengurungan mulai dan akhiran
- 3) tidak ada deklarasi-deklarasi argumentasi atau variabel yang diperlukan.

### 1.3.2 Indentasi

Bahasa pemrograman Python adalah bahasa pemrograman yang mudah dibaca dan terstruktur, hal ini karena digunakannya sistem indentasi. Yaitu memisahkan blok-blok program dengan susunan indentasi. Jadi untuk memasukan sub-sub program dalam suatu

blok, sub-sub program tersebut diletakkan satu atau lebih spasi dari kolom suatu blok program. Python memiliki sedikit perbedaan pada cara penulisan program dengan bahasa pemrograman yang lain seperti C/Java. Kalau pada C/Java menggunakan tanda kurung sebagai pemisah blok program, di Python kita hanya menggunakan spasi sebagai pemisah blok program yang biasa disebut sebagai Indentasi. Karena Python menjalankan perintah secara berurutan, maka kita harus pintar menyusun perintah agar mendapatkan hasil seperti yang diinginkan.

```
if a = b :  
    |print a, 'sama dengan', b  
else :  
    |print a, 'tidak sama dengan', b
```

Pada contoh di atas kita dapat melihat jika suatu kondisi  $a = b$  dipenuhi maka program akan menjalankan baris perintah yang ada di dalam suatu blok kondisi tersebut, yang ditandai dengan penggunaan satu spasi atau lebih dari blok kondisi sebelumnya, dalam contoh di atas perintah yang akan dilaksanakan jika suatu kondisi di atasnya terpenuhi menggunakan dua (2) spasi, sedangkan pada pernyataan else, menggunakan satu spasi. Perbedaan penggunaan spasi ini tidak dianjurkan meskipun dalam program Python dibenarkan, karena struktur program akan lebih sulit dibaca. Seharusnya blok-blok program di atas adalah sebagai berikut:

---

```

if a = b :
    |print a, 'sama dengan', b #Menggunakan 3
    spasi
else :
    |print a, 'tidak sama dengan', b
    #Menggunakan 3 spasi

```

### 1.3.3 Baris Perintah

Dalam Python, program tersebut dapat langsung dijalankan sebagai berikut:

```

print "Hello Python!" #Hasilnya akan menampilkan string
"Hello Python!"

```

Bandingkan dengan Kedua bahasa pemograman di bawah ini:

#### **Java :**

```

class hello Python {
public static void main(String argsp[]) {
System.out.println("Hello Python"); }

```

---

#### **C/C++ :**

```

#C
#include <stdio.h>
    int main() {
        printf("Hello Python!"); return 0;
    }

#C++
#include <iostream.h>
    int main() {
        cout << "Hello Python!";
    }

```

Tanda kutip dua ("), yang berarti tempat nilai string diletakkan pada program ini tidak akan ditampilkan pada layar. Sebuah kalimat perintah adalah sebuah instruksi yang dapat dieksekusi oleh interpreter Python. Kita telah melihat dua jenis kalimat perintah, yakni print dan pendeklarasian nilai. Pada saat kalian mengetikkan kalimat perintah pada prompt perintah, maka python mengeksekusinya dan langsung menampilkan hasilnya. Jika ada, hasil dari perintah print adalah sebuah nilai. Pendeklarasian nilai tidak menampilkan hasil. Pada sebuah script biasanya berisikan beberapa kalimat perintah. Jika lebih dari satu kalimat perintah, hasilnya akan tampil sesuai dengan kalimat perintah yang dieksekusi.

Contohnya:

```
>>> print 1
>>> x = 2
>>> print x
```

Menghasilkan hasil;

```
1
2
```

#### 1.3.4 Keterangan Program

Dalam proses debugging, suatu keterangan / komentar instruksi program sangat berguna sekali dalam pembacaan suatu kode. Pada umumnya komentar berisi keterangan tentang kegunaan suatu fungsi itu. Sintaksnya adalah tanda pagar "#". Setelah meletakkan tanda tersebut, kita dapat mengetikkan kalimat apa saja yang berhubungan dengan suatu instruksi perintah, sebab apapun kalimat tersebut tidak akan mempengaruhi jalannya program dan tidak akan di proses

oleh interpreter. Jika interpreter menemukan tanda ini maka mulai tanda ini sampai akhir baris akan dianggap sebagai keterangan.

### 1.3.5 Variabel

Sebuah variabel adalah sebuah nama yang mempunyai sebuah nilai. Pendeklarasian kalimat membuat sebuah variabel - variabel baru dan memberinya nilai.

```
>>> a = "belajar Python"
>>> b = 5
>>> phi = 3.14
```

Pada contoh di atas, pendeklarasian tersebut menciptakan 3 variabel baru. Pendeklarasian pertama, menunjukkan string "belajar Python" ke sebuah variabel yang bernama a. Kedua, variabel b diberi nilai 5 sebagai integer. Dan yang terakhir variabel phi diberi nilai 3.14 sebagai nilai pecahan. Cara yang umum untuk pemberian nama variabel adalah dengan tanda panah menunjuk ke nilai variabel tersebut. Jenis ini dinamai dengan state diagram karena menunjukkan nilai-nilai yang merupakan nilai dari variabel - variabel tersebut, contohnya :

```
a => "belajar Python"
b => 5
phi => 3.14
```

Perintah print juga berlaku untuk kalimat di atas.

```
>>> print a
Belajar Python
```



```
>>> print b
5
>>> print phi
3.14
```

### 1.3.6 Nama Variabel

Pada umumnya, programmer memakai nama variabel sesuai dengan keterangan isi dari variabel tersebut dan variabel juga merupakan simbol yang mewakili nilai tertentu. Pembuatan variabel dalam python sangat sederhana. Berikut adalah ketentuan mengenai variabel dalam python,

- 1) Variabel yang dideklarasikan tidak harus mempunyai tipe data tertentu.
- 2) Tipe data dalam variabel dapat berubah-ubah.
- 3) Penulisan variabel harus diawali dengan huruf, dan untuk karakter selanjutnya bisa berupa huruf atau angka.
- 4) Dapat berupa huruf Kapital, tetapi bersifat case-sensitive, nama Kapital dengan capital adalah variabel yang berlainan.
- 5) Penulisan variabel tidak boleh dipisah oleh <spasi>.
- 6) Untuk variabel yang terdiri dari 2 suku kata, dapat dipisah dengan simbol underscore ( \_ ) seperti nama\_saya, nama\_variabel\_nama.
- 7) Statemen yang tidak boleh dijadikan nama variabel adalah keywords pada Python.

Contoh:

```
>>> 123satu = "angka"
```

variabel 123satu adalah penamaan variabel tidak benar karena diawali dengan sebuah angka.

```
>>> lebih$ = 50000
```

lebih\$ juga tidak benar karena terdapat karakter yang tidak semestinya ada dalam penamaan variabel.

### 1.3.7 Keyword / Kata Kunci

Kata kunci mendefinisikan aturan - aturan dan struktur bahasa, dan mereka tidak dapat digunakan sebagai nama variabel. Python mempunyai 28 kata kunci:

|               |                 |                |               |               |            |              |
|---------------|-----------------|----------------|---------------|---------------|------------|--------------|
| <b>And</b>    | <b>Continue</b> | <b>else</b>    | <b>for</b>    | <b>Import</b> | <b>not</b> | <b>Raise</b> |
| <b>assert</b> | <b>return</b>   | <b>except</b>  | <b>from</b>   | <b>In</b>     | <b>or</b>  | <b>def</b>   |
| <b>break</b>  | <b>del</b>      | <b>exec</b>    | <b>global</b> | <b>pass</b>   | <b>is</b>  | <b>Try</b>   |
| <b>class</b>  | <b>elif</b>     | <b>finally</b> | <b>print</b>  | <b>Lambda</b> | <b>if</b>  | <b>while</b> |

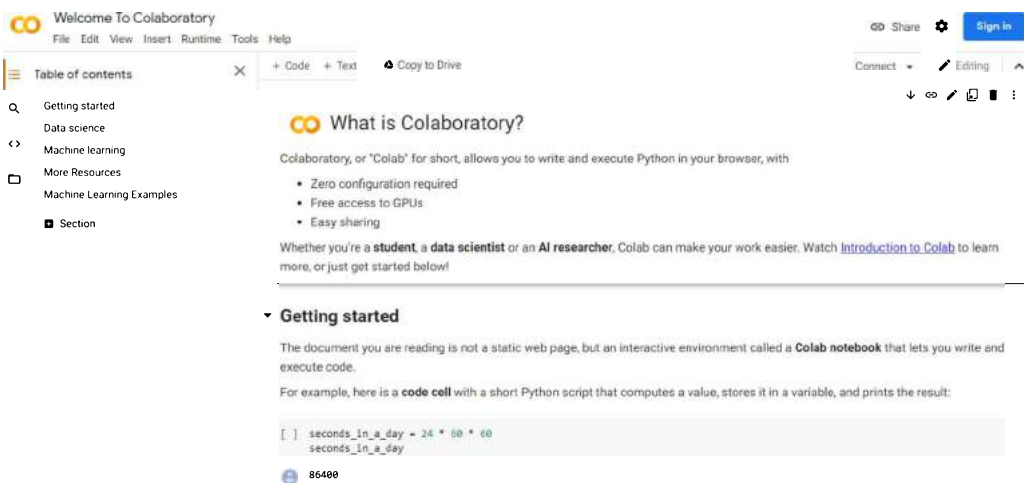
Kata kunci di atas merupakan daftar kata yang tidak bisa kalian gunakan sebagai nama variabel, pada saat interpreter mengeluarkan kesalahan sintaks dari salah satu nama variable, mungkin kalian menggunakan salah satu nama variabel dari daftar di atas.

## 1.4 Pengenalan *Google Colab*

*Google Colaboratory* (Colab) merupakan produk buatan dari *Google Research*. Colab sendiri adalah *executable document*, yang dapat digunakan untuk menulis, menyimpan, menjalankan dan

membagikan program melalui *Google Drive*. Google juga menyediakan layanan GPU gratis untuk pengguna Google Colab sehingga pengguna dapat melakukan pengolahan data tanpa memerlukan hardware yang mumpuni.

- 1) Untuk mulai menggunakan Google Colab, pastikan anda memiliki koneksi internet yang stabil dan buka halaman awal Google Colab di <https://colab.research.google.com/>
- 2) Ketika pertama kali membuka halaman Google Colab, kita akan diarahkan ke *notebook* berjudul “*Welcome To Colaboratory*”.





- 3) Kemudian pada bagian *Getting Started* kita dapat melihat contoh program dalam bahasa python.

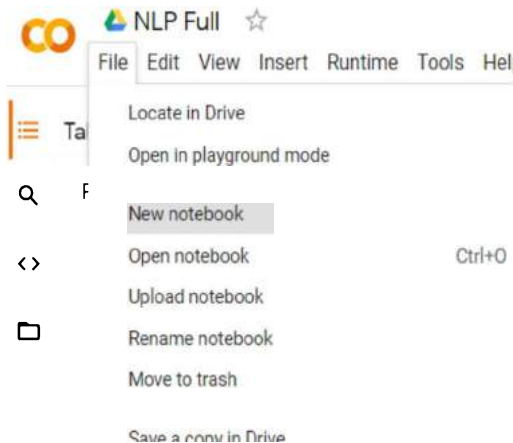
#### ▼ Getting started

The document you are reading is not a static web page, but an interactive environment called a **Colab notebook** that lets you write and execute code.

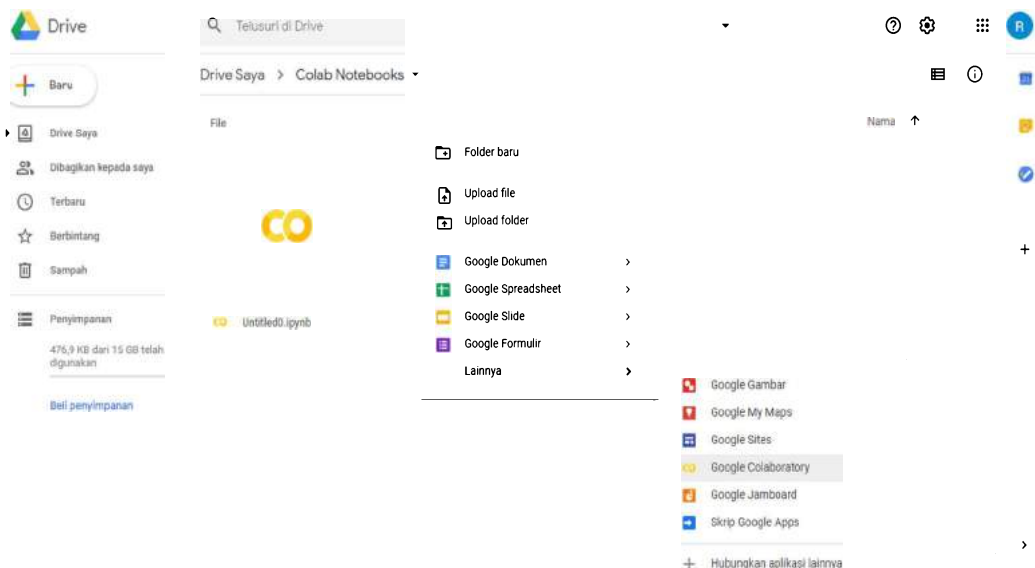
For example, here is a **code cell** with a short Python script that computes a value, stores it in a variable, and prints the result:



- 4) Tekan tombol  (*run*) untuk menjalankan program. Pastikan sudah melakukan login dengan menggunakan akun gmail.
- 5) Untuk membuat *notebook* baru klik *file*  *new notebook*



- 6) Membuat *notebook* baru dapat juga dilakukan melalui Google Drive.



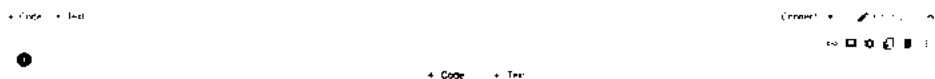
7) Tampilan halaman *notebook* baru:



8) Ubah nama *notebook* menjadi “Pengenalan Colab.ipynb”



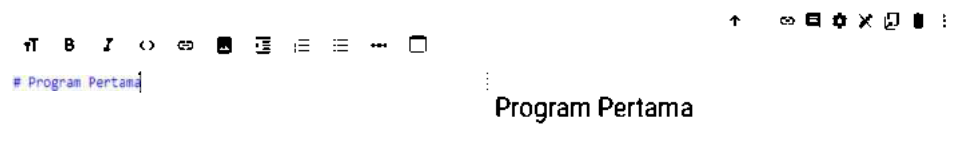
9) Pada Google Colab kita dapat menambahkan teks ataupun syntax dengan menekan tombol +text atau +code.



10) Tambahkan *text* baru dan isi “Program Pertama”. Lalu pindahkan ke bagian paling atas *notebook* dengan menekan tombol ↑.



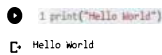
11) Kita dapat merubah *text* menjadi Bab, subab dan sub-subab dengan tombol ¶.



12) Isi dengan syntax sederhana:

```
print("Hello World")
```

13) Lalu jalankan. Setelah selesai dijalankan, hasil program akan muncul di bawahnya.



```
1 print("Hello World")
```

Hello World

14) Pada Google Colab, hasil code yg telah dijalankan akan disimpan. Untuk memahami, buat *code* baru dan isi dengan:

```
n = 23
m = 27
```

lalu jalankan!

15) Buat *code* baru berisi perintah `print(n+m)` dan jalankan!



```
1 n = 23
2 m = 27
```

```
1 print(n+m)
```

50

## 1.5. Python Dasar dengan Google Colab

### 1) Syntax

Python tidak perlu menggunakan delimiter titik koma “;” dan kurung kurawal “{ }”. Sebagai gantinya, pada python indentasi sangat diperhatikan. Indentasi atau *white-space* pada awal baris program harus sama. Perhatikan perbandingan antara c++ dan python berikut:

C++:

```
#include <iostream>
using namespace std;

int main()
{
    int angka1 = 10;
    int angka2 = 20;

    if(angka1 > angka2)
    {
        cout << "angka 1 lebih besar";
    }
}
```

```

    }
    else
    {
        cout << "angka 2 lebih besar";
    }
}

```

### Python:

```

angka1 = 10
angka2 = 20
if(angka1 > angka2):
    print("Angka 1 lebih besar")
else:
    print("Angka 2 lebih besar")

```

### 2) Variable

Terdapat beberapa tipe data seperti pada bahasa pemrograman lainnya seperti *boolean*, *string*, *integer*, *float*, *list*, *tuple*, dan *dictionary*. Pembuatan variable tidak memerlukan deklarasi tipe data. Sebagai contoh:

```

a = 'ini merupakan contoh deklarasi kalimat pada python'
b = 1000
B = "variabel pada python case-sensitive"
c = True
c = False
d = a+B

```

### 3) Fungsi Print

Untuk menampilkan isi variabel yang sudah dibuat, gunakan fungsi `print()`. Contoh penggunaan fungsi `print`:

```

print(a)
print("Variabel b =", b)
print(B)
print(c)
print(d)

```

### 4) Komentar

Komentar pada Python dibuat dengan tanda pagar `"#"`. Komentar berguna untuk menuliskan catatan agar program lebih mudah dipahami. Contoh penggunaan komen:

```

angka1 = 100
# ini merupakan komentar yang tidak akan dibaca oleh program

angka2 = 200 # komentar dapat juga diletakkan diakhir baris

angka3 = angka1 + angka2
"""
Untuk membuat komentar lebih dari satu baris
Dapat menggunakan tanda petik
"""
print(angka3)

```

### 5) Tuple, List, dan Dictionary

Python memiliki tiga tipe data yang mirip dengan array yaitu tuple, list dan dictionary. Ketiganya memiliki perbedaan dan kegunaan sendiri. Tuple didefinisikan menggunakan tanda kurung “( )” dan isinya **tidak dapat** diubah. List didefinisikan menggunakan kurung siku “[ ]”. Sedangkan dictionary didefinisikan menggunakan kurung kurawal “{ }”. Baik list maupun dictionary isinya dapat diubah. Jalankan program di bawah dan perhatikan hasilnya!

```

tup1 = (1, "a", 3, "5", "6")
list1 = ["a", "b", "c", "d", "e", "f"]
dict1 = {"tahun":2021, "bulan":"Februari", "susah":False}
dict2 = {"ganjil":[1,3,5,7,9], "genap":[2,4,6,8,10]}

print(tup1)
print(tup1[2])
print(tup1[2:4])
print()

print(list1)
print(list1[5])
print(list1[1:6])
print()

print(dict1)
print(dict1["tahun"]) # print values dari keys tahun pada dict1
print(dict2["genap"][2]) # print nilai 6 dari values genap

```



## 6) Input

Kita dapat mengisi nilai variable saat menjalankan program dengan fungsi `input()`, seperti:

```
a = input() #string
b = int(input()) #int
c = float(input()) #float
d = input("Masukkan nama anda:")
```

## 7) If-else

Ketika menggunakan if-else, perhatikan indentasi!

```
a = int(input("a = "))
b = int(input("b = "))

if a == b:
    print("a sama dengan b")
elif a > b:
    print("a lebih besar dari b")
else:
    print("a lebih kecil dari b")
```

Contoh lain, if dapat digunakan untuk mengetahui apakah list memiliki data tertentu:

```
list1 = [1,2,3,4,5,6,7,8,9,10]

if 1 in list1:
    print("ada")

if 22 not in list1:
    print("tidak ada")
```

## 8) For

Sedikit berbeda dengan bahasa pemrograman lainnya, pada python perulangan for digunakan seperti contoh syntax berikut:

```
list1 = [1,2,3,4,5,6,7,8,9,10]
for angka in list1:
    print(angka)
```

Jika pada bahasa lain biasanya penggunaan for seperti `for(int i=0; i<10; i++)`. For pada python menggunakan fungsi `range()`, contoh:

```
for i in range(1,10):
    print(i)
```

#### 9) Fungsi/def

Fungsi pada python didefinisikan dengan `def`, `def` merupakan subbagian dari sebuah class. Kita dapat membuat output berupa fungsi `print` untuk langsung menampilkan output seperti pada contoh berikut:

```
def luaslingkaran(diameter):
    print(22/7*diameter/2*diameter/2)
luaslingkaran(14)
```

Untuk menggunakan output dari fungsi sebagai nilai variabel, ganti fungsi `print` menjadi fungsi `return`. Perhatikan contoh berikut:

```
def luaslingkaran(diameter):
    return(22/7*diameter/2*diameter/2)
def volKerucut(diameter, tinggi):
    luasAlas = luaslingkaran(diameter)
    return(luasAlas*tinggi/3)

print(volKerucut(14, 15))
```

## 1.6 Latihan



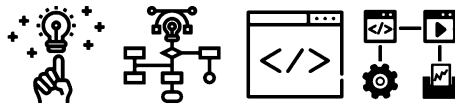
Buka link berikut:

<https://docs.python.org/2/tutorial/controlflow.html>

Amati poin 4.1 – 4.4

- 1) Baca dan pahami setiap poin bahasan
- 2) Deskripsikan setiap poin program dan jelaskan pre-cond, process, dan post-cond pada setiap program.
- 3) Ketik ulang setiap contoh dan jalankan.

- 4) Apakah program berjalan sesuai dengan definisi kalian (pre-condition, post-condition, process)?



- 5) Dari beberapa program di atas, lakukan modifikasi program secara bebas sesuai dengan pemahaman kalian.

## BAB 2. LISTS & TUPLE

### 2.1 Tujuan Pembelajaran

- 1) Mahami konsep dasar *Lists & Tuple*
- 2) Melakukan analisis permasalahan menggunakan konsep *Lists & Tuple*
- 3) Membuat program penyelesaian masalah menggunakan konsep *Lists & Tuple*

### 2.2 Lists

Lists merupakan kumpulan data pada urutan tertentu, kalian dapat membuat lists yang terdiri dari huruf, angka, kata, ataupun kombinasi di antara ketiganya. Kalian bisa merubah, menambah, maupun mengurangi isi dari data yang kalian buat. Dalam python isi dari lists berada dalam tanda kurung siku ( `[]` ) dan dipisahkan oleh koma pada setiap elemennya. Berikut adalah contoh sederhana menggunakan lists pada python:

```
prima = [1,3,5,7]
buah = ['mangga', 'apel', 'jeruk']
campur = ['mangga', 'apel', 'jeruk', 1, 3, 5, 7]
```

Untuk mengakses elemen dari sebuah list, yang perlu diperhatikan adalah setiap elemen terdapat pada indeks tertentu dan indeks pada list berawal dari 0. Kalian bisa mengakses setiap elemen didalam lists sesuai dengan posisi/indeks elemen tersebut. Sebagai

contoh pada list buah[ ] di atas ketika anda ingin mengakses apel, maka sintaks:

```
print(buah[1])
>> apel
```

Pada output akan menampilkan string apel karena sintaks mengakses indeks ke-1 dari list buah. Selain itu anda juga bisa menggabungkan elemen dari list pada suatu variable, sebagai contoh :

```
pesan = "Saya mau membeli" + buah[1] "sebanyak
1Kg."
print(pesan)
>> Saya mau membeli apel sebanyak 1Kg.
```

Pada contoh program di atas terdapat variabel “pesan” berisi string yang ditambahkan elemen list didalamnya, maka pada output program akan ditampilkan isi string dari variabel “pesan” yang ditambahkan elemen list didalamnya.

### 2.2.1 Mengolah data pada lists

Karena sifat list yang dinamis, kita bisa mengolah data dalam list yang kita buat seperti menambah, mengurangi, dan marubah item pada list. Sebagai contoh, ketika kalian ingin membuat program data mahasiswa maka setiap mahasiswa baru data akan ditambahkan pada list, sebaliknya ketika ada mahasiswa yang lulus maka data akan dihapus dalam list. Sintaks untuk mengolah list hampir sama dengan sintaks untuk mengakses elemen list, kalian hanya perlu mengganti nama fungsi sesuai dengan kebutuhan.

Beberapa fungsi pada python untuk mengolah list (sebagai contoh menggunakan list buah sebelumnya) adalah sebagai berikut:

### 1) Append ( `list.append (elemen)` )

Append berfungsi untuk menambah elemen pada list. Dengan menggunakan fungsi ini, maka elemen baru yang ditambahkan akan berada pada indeks terakhir dalam list.

```
buah.append('anggur')
print(buah)
>> ['mangga', 'apel', 'jeruk', 'anggur']
```

Pada sintaks di atas menambahkan elemen 'anggur' dalam list buah, maka elemen pada list buah sekarang adalah manga, apel, jeruk, anggur.

### 2) Extend ( `list.extend(list_ext)` )

Extend berfungsi menggabungkan dua buah list. Dengan fungsi ini kita bisa menggabungkan isi dari dua list menjadi satu list. Misalkan kita mempunyai list buah2 = [ 'nangka', 'sirsak', 'melon' ]

```
buah.extend(buah2)
print(buah)
>> ['mangga', 'apel', 'jeruk', 'anggur',
     'nangka', 'sirsak', 'melon']
```

### 3) Insert `list.insert(idx, elemen)`

Insert berfungsi memasukkan sebuah item pada posisi indeks tertentu. dengan fungsi ini kita bisa memasukkan sebuah elemen dalam list dimana saja tergantung indeks yang kita inginkan.

```
buah.insert(3, pisang)
print(buah)
>> ['mangga', 'apel', 'jeruk', 'pisang',
      'anggur', 'nangka', 'sirsak', 'melon']
```

Pada sintaks di atas kita memasukkan elemen pisang pada indeks ketiga, oleh karena itu indeks ketiga sebelumnya yang ditempati oleh anggur akan bergeser kekanan satu posisi (posisi indeks +1) begitu pula selanjutnya.

#### 4) Remove ( `list.remove(item)` )

Remove berfungsi menghapus elemen pada list. Dengan fungsi ini kita bisa menghapus elemen list berdasar dari nama elemen itu sendiri.

```
buah.remove('jeruk')
print(buah)
>> ['mangga', 'apel', 'pisang', 'anggur',
      'nangka', 'sirsak', 'melon']
```

Dengan menghapus elemen 'jeruk' otomatis jeruk sudah tidak ada dalam elemen list.

#### 5) Delete ( `del list[idx]` )

Del berfungsi menghapus elemen dari list berdasar indeks elemennya.

```
Del buah[5]
print(buah)
>> ['mangga', 'apel', 'pisang', 'anggur',
      'nangka', 'melon']
```

Elemen yang dihapus adalah [5], yaitu sirsak sudah tidak ada dalam elemen list buah.

### 6) Pop ( list.pop(idx) )

Pop berfungsi mengambil item dalam list pada indeks tertentu dan menampilkan pada output program.

```
print buah.pop('2')
>> pisang
```

Dengan mengambil elemen indeks ke-2 maka pisang akan ditampilkan pada output program.

### 7) Count ( list.count(elemen) )

Count berfungsi menghitung jumlah item pada list

```
print "Jumlah manga :", buah.count(mangga)
print "Jumlah anggur :" buah.count(anggur)
>> Jumlah manga : 1
>> Jumlah anggur : 1
```

Karena jumlah manga dan anggur sama-sama satu maka output program akan menampilkan jumlah elemen manga dan anggur.

### 8) Sort ( list.sort() )

Sort berfungsi mengurutkan item pada list

```
Buah.sort()
print(buah)
>> ['anggur', 'apel', 'mangga', 'melon',
    'nangka', 'pisang']
```



## 9) Reverse ( list.reverse() )

Reverse berfungsi menampilkan elemen list dengan indeks yang terbalik.

```
Buah.sort()
print(buah)
>> ['pisang', 'nangka', 'melon', 'mangga',
      'apel', 'anggur']
```

## 2.3 Tuple

Pada materi sebelumnya kalian telah belajar tentang list, yaitu sebuah konsep struktur data pada python untuk menyimpan data secara dinamis dan dapat dirubah sewaktu-waktu. Namun terkadang kita menemui kondisi yang mengharuskan data yang akan diolah pada program bersifat statis atau tidak bisa dirubah, dalam kasus ini kita bisa menggunakan tuple pada python.

Secara konsep dan cara mengkasesnya, tuple hampir sama dengan list. Selain isi pada tuple tidak bisa dirubah, perbedaan lain list dengan tuple adalah ketika kita mendefinisikan tuple kita menggunakan kurung awal dan kurung tutup ().

Contoh :

```
prima = (1,3,5,7)
buah = ('mangga', 'apel', 'jeruk')
campur = ('mangga', 'apel', 'jeruk', 1, 3, 5,
7)
```

### 2.3.1 Mengakses Nilai dalam Tuple

Untuk mengakses nilai di dalam Tuple, gunakan tanda kurung siku dengan memberikan nilai indeks sesuai dengan elemen yang dipilih.

Contoh:

```
tup1 = ('Python', 'Java', 1997, 2000)
tup2 = (1, 2, 3, 4, 5 )
print "tup1[0]: ", tup1[0]
print "tup2[1:5]: ", tup2[1:5]
```

Output:

```
tup1[0]: Python
tup2[1:5]: (2, 3, 4, 5)
```

### 2.3.2 Mengupdate Tuple

Tuple bersifat immutable (nilai konstan). Anda tidak dapat memperbarui atau mengubah nilai-nilai elemen pada Tuple. Teknik lain adalah dengan mengambil bagian dari Tuple yang ada untuk menciptakan Tuple baru seperti contoh berikut:

Contoh:

```
tup1 = (12, 34.56);
tup2 = ('abc', 'xyz');
# statement ini tidak berlaku untuk Tuple
# tup1[0] = 100;
# membuat Tuple baru sebagai berikut
tup3 = tup1 + tup2;
print tup3
```

Output:

```
(12, 34.56, 'abc', 'xyz')
```

### 2.3.3 Menghapus Tuple

Menghapus elemen-elemen secara individu (satu per satu) dari Tuple itu tidaklah mungkin karena sifat Tuple itu Immutable (Tetap Nilainya). Ada cara lain untuk melakukan hal ini yaitu dengan menggunakan kata kunci `del` di ikuti dengan nama Tuplenya.

Contoh:

```
tup = ('Python', 'Java', 1997, 2000);
print tup;
del tup;
print "Setelah Menghapus tup : "
print tup;
```

Output:

```
Traceback (most recent call last):
File "D:\codes\php\latihan\testing.py", line 9,
in <module>
print tup;
NameError: name 'tup' is not defined
('Python', 'Java', 1997, 2000)
Setelah Menghapus tup :
```

### 2.3.4 Contoh Beberapa Operasi Pada Tuple

| Syntax Python                                 | Hasil                             | Ket                           |
|---|-----------------------------------|-------------------------------|
| <code>len((1, 2, 3))</code>                   | 3                                 | Jumlah elemen dalam tuple     |
| <code>(1, 2, 3) + (4, 5, 6)</code>            | (1, 2, 3, 4, 5, 6)                | Menggabungkan tuple           |
| <code>('Halo') * 4</code>                     | ('Halo', 'Halo', 'Halo', 'Halo,') | Mengulang elemen tuple        |
| <code>3 in (1, 2, 3)</code>                   | True                              | Mengecek elemen pada tuple    |
| <code>for x in (1, 2, 3):<br/>print x,</code> | 1 2 3                             | Mengakses seluruh nilai tuple |

Tabel 2.1 Contoh operasi pada tuple

### 2.3.5 Contoh Program Menggunakan Tuple

---

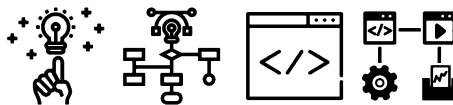
```

kebun_binatang = ('ular python', 'gajah', 'penguin')
print 'jumlah binatang yang ada di kebun binatang
:', len(kebun_binatang)
kebun_binatang_baru = 'monyet', 'unta',
kebun_binatang
print 'jumlah kandang di kebun binatang baru:',
len(kebun_binatang_baru)
print 'binatang yang ada di kebun binatang baru:',
kebun_binatang_baru
print 'binatang dari kebun binatang lama:',
kebun_binatang_baru[2]
print 'binatang terakhir dari kebun binatang lama:',
kebun_binatang_baru[2][2]
jumlah_binatang = len(kebun_binatang_baru) - 1 +
len(kebun_binatang_baru[2])
print 'jumlah binatang yang ada di kebun binatang
baru :', jumlah_binatang

```

---

## 2.4 Latihan



- 1) Setelah belajar tentang list, buatlah satu atau lebih list berisikan tentang apa saja yang kalian inginkan (hewan, buku, negara, hobi, pekerjaan, dll). Kemudian, jika data pada list kalian harus diolah pada suatu program menggunakan bahasa python, kira-kira program seperti apa yang akan kalian buat?
  - a) Diskusikan bersama anggota kelompok tentang apa yang akan kalian buat.
  - b) Tuliskan penjelasan singkat tentang program yang akan kalian buat.

- c) Definisikan pre, post, dan proses dari program yang kalian buat.
- d) Buat program menggunakan bahasa python, dan gunakan fungsi pada list yang telah kalian pelajari minimal 3 fungsi pada program kalian.

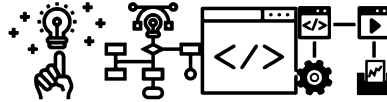


- 2) Bukalah link di bawah ini

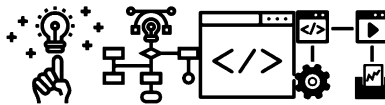
[https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/lec5\\_tuples\\_lists.py](https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/lec5_tuples_lists.py)

- a) Amati setiap contoh program list dan tuple.
- b) Sebelum mencoba menjalankan program tersebut pada IDE kalian, coba tentukan pre dan post setiap contoh program berdasarkan pengamatan kalian.
- c) Setelah menentukan pre dan post setiap program, jalankan pada IDE kalian. Apakah output setelah program dijalankan sesuai dengan prediksi post kalian sebelumnya? Jelaskan sesuai dengan pemahaman kalian.

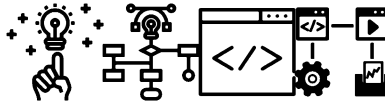
## 2.5 Studi Kasus



- 1) Buatlah sebuah program yang dapat menampilkan 5 elemen pertama dan 5 elemen terakhir dari sebuah list. Dimana elemen-elemen dalam list tersebut didapat dari bilangan pangkat dua bilangan antara 1 dan 20.



- 2) Buatlah sebuah list dengan elemen beberapa bilangan, buatlah program yang dapat menentukan bilangan kedua terkecil dan bilangan kedua terbesar dalam list tersebut.



- 3) Buatlah sebuah program yang menerima inputan sebuah string kata/kalimat , dan menghasilkan output berupa huruf apa saja yang terdapat pada string tersebut beserta jumlahnya.

## BAB 3. DICTIONARY & SET

### 3.1 Tujuan Pembelajaran

- 1) Memahami konsep dasar *Dictionary & Set*.
- 2) Menganalisis permasalahan menggunakan konsep *Dictionary & Set*.
- 3) Membuat program penyelesaian masalah menggunakan konsep *Dictionary & Set*.

### 3.2 Dictionary

Dictionary juga merupakan struktur data pada python seperti list dan tuple. Dictionary bersifat dinamis seperti list yang isinya dapat dirubah-rubah sewaktu-waktu. Dictionary memiliki sepasang key(kunci) dan value(nilai) yang saling berhubungan. Setiap key dan value selalu dipisahkan dengan titik dua (:), key dan value merupakan item, setiap item dipisahkan dengan koma, dan semuanya tertutup dalam kurung kurawal. Untuk Dictionary kosong ditulis hanya dengan dua kurung kurawal, seperti ini: {}.

Ketika memakai dictionary, yang perlu diperhatikan adalah key dari setiap elemen harus unik yang artinya tidak boleh ada key yang sama pada setiap dictionary. Contoh penggunaan dictionary:

```
>>> mobil = {'warna':'hitam', 'tahun':2014}
```

### 3.2.1 Mengolah data pada dictionary

#### 1) Menambah atau merubah elemen dalam dictionary

```
>>> mobil['merk'] = 'Audi'
>>> mobil
{'warna': 'hitam', 'tahun': 2014, 'merk': 'Audi'}
```

```
>>> mobil['merk'] = 'Honda'
>>> mobil
{'warna': 'hitam', 'tahun': 2014, 'merk': 'Honda'}
```

#### 2) Menggabungkan dictionary menggunakan *update()*

```
>>> mobil2 = {'bbm': 'pertamax', 'cc': '1500'}
>>> mobil.update(mobil2)
>>> mobil
{'warna': 'hitam', 'tahun': 2014, 'merk': 'Honda', 'bbm': 'pertamax', 'cc': '1500'}
```

#### 3) Menghapus elemen menggunakan *del*

```
>>> del mobil['cc']
>>> mobil
{'warna': 'hitam', 'tahun': 2014, 'merk': 'Honda', 'bbm': 'pertamax'}
```

#### 4) Menghapus semua elemen menggunakan *clear()*

```
>>> mobil.clear()
>>> mobil
{}
```



### 5) Mengakses elemen dengan [key]

```
>>> mobil['merk']
'Honda'
```

#### 3.2.2 Operasi dasar dictionary

| Metode                     | Fungsi  |
|----------------------------|---|
| len(d)                     | menampilkan nilai kembalian jumlah elemen dalam dictionary menggunakan fungsi len ( )   |
| aDict [key]                | Memasukkan key baru, mengganti nilai, atau mendapatkan nilai dari key yang sudah ada  |
| d.get ( key [ , default] ) | Nilai kembalian dari key yang sudah ada atau default key jika key belum ada (terjadi error jika default key dihilangkan dan key tidak ada)            |
| d.pop( key [ , default ] ) | Menghapus key dan menampilkan nilai jika key sudah ada atau default jika key tidak ada (terjadi error jika default key dihilangkan dan key tidak ada) |
| d.keys ( )                 | Nilai kembalian daftar dari keys  |
| d.values ( )               | Nilai kembalian daftar dari values  |
| d.has_keys ( key )         | Nilai kembalian True jika key ada dan False jika key tidak ada  |
| d.clear ( )                | Menghapus semua keys  |

Tabel 3.1 Operasi dasar pada dictionary

#### 3.2.3 Contoh program menggunakan dictionary

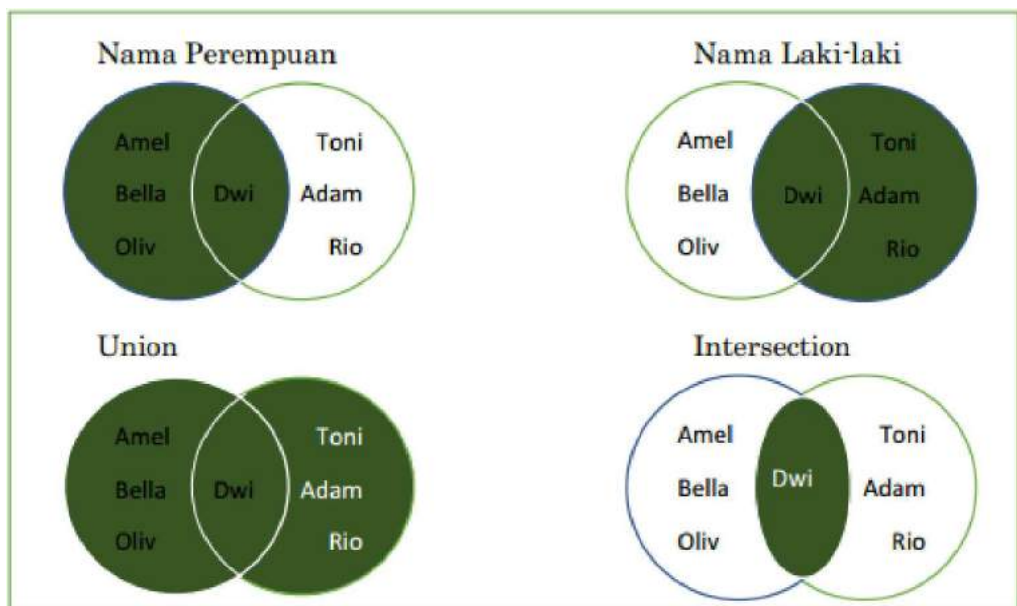
```
ba = {'andi': 'andi@python.org', \
      'bella': 'bella@rocket.org', \
      'doni': 'doni@domain.com'}
print 'alamat email andi:', ba['andi']
# menghapus item
del ba['doni']
print 'ada %s kontak di buku alamat' % len(ba)
for nama, email in ba.items():
```

```
print '%s, email: %s' % (nama, email)
# tambah entri
ba['evan'] = 'evan@evandian.org'
if 'evan' in ba:
    print 'Email evan di', ba['evan']
```

```
alamat email andi: andi@python.org
ada 2 kontak di buku alamat
bella, email: bella@rocket.org
andi, email: andi@python.org
Email evan di evan@evandian.org
```

### 3.3 Set

Set atau sebuah himpunan adalah struktur data pada python yang tidak tergantung dalam urutan tertentu, data pada set bersifat unik yaitu tidak ada data yang sama dalam setiap set.



Gambar 3.1 Contoh Penggunaan Set (Himpunan)

3.3.1 Operasi dasar pada Set

Dalam implementasi set ada beberapa operasi dasar yang digunakan, antara lain:

| Metode             | Fungsi  |
|--------------------|---|
| set()              | membuat set kosong  |
| length()           | menampilkan nilai kembalian jumlah elemen dalam set menggunakan fungsi len()  |
| contains(element): | Mengecek isi dalam set dan menampilkan nilai kembalian (T/F)  |
| add(element)       | Menambahkan elemen dalam set  |
| remove(element)    | Menghapus elemen dalam set  |
| equals(setB)       | Mengecek elemen set dengan set lainnya. Menampilkan nilai kembalian True jika isi set sama, dan false jika sebaliknya   |
| isSubsetOf(setB)   | Membandingkan suatu set (set A) dengan set lainnya (set B). set B menjadi subset A jika semua isi set B juga berada di set A. menampilkan nilai kembalian (T/F) |
| union(setB)        | Membuat set baru yang berisi gabungan set A dan B   |
| intersect(setB)    | Membuat set baru yang berisi nilai yang sama antara set A dan B   |
| difference(setB)   | Membuat set baru yang berisi nilai set A yang yang tidak ada pada set B   |

Tabel 3.2 Operasi dasar pada Set

3.3.2 Contoh penggunaan metode dalam set :

| Set A & B                             | Union              | Intersection | Difference         | Subset |
|---------------------------------------|--------------------|--------------|--------------------|--------|
| A : {12, 5, 17, 6}<br>B : {42, 17, 6} | {12, 5, 42, 17, 6} | {17, 6}      | {12, 5}            | False  |
| A : {21, 76, 10, 3, 9}<br>B : {}      | {21, 76, 10, 3, 9} | {}           | {21, 76, 10, 3, 9} | True   |
| A : {87}<br>B : {22, 87, 23}          | {22, 87, 23}       | {87}         | {}                 | False  |
| A : {22, 87, 23}<br>B : {87}          | {22, 87, 23}       | {87}         | {22, 23}           | True   |

Tabel 3.3 Implementasi Metode pada Set

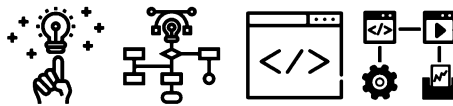
### 3.3.3 Contoh program sederhana menggunakan set:

```
>>> A = set([0, 1, 2])
>>> B = set()
>>> 1 in A
True
>>> A & B
{}
>>> B.add(1)
>>> B.add(1)
>>> B.add(5)
>>> B
{1, 5}
>>> A & B
{1}
>>> A | B
{0, 1, 2, 5}
>>> A - B
{0, 2}
>>> B.remove(5)
>>> B
{1}
>>> B.issubset(A)
True
>>> for item in A:
print(item, end="")
0 1 2
>>>
```

### 3.4 Latihan



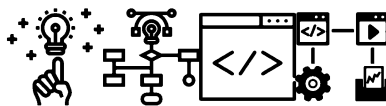
- 1) Bukalah link berikut <https://www.programiz.com/python-programming/set> cobalah program yang berada pada link tersebut, lalu buatlah ringkasan mengenai materi set tersebut.



- 2) Buatlah program kamus sederhana untuk menerjemahkan kata dari bahasa indonesia ke bahasa inggris (minimal 10 kata dalam kamus)

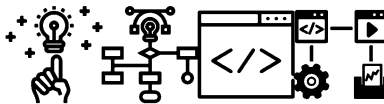
### 3.5 Studi Kasus

| Nama Mhs | Mata Kuliah       |
|----------|-------------------|
| Andi     | BD, PBO, KDJK, PW |
| Budi     | BD, KDJK          |
| Sinta    | SD, TI, KDJK      |



- 1) Pada tabel di atas terdapat data 3 mahasiswa dan matakuliah yang mereka ambil. Buatlah program yang dapat :
  - a) Menampilkan semua matakuliah yang diambil ketiga mahasiswa tersebut
  - b) Menampilkan matakuliah yang sama antara mereka bertiga

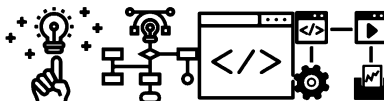
- c) Menampilkan matakuliah yang tidak diambil antara dua mahasiswa (kombinasi mahasiswa bebas)
- d) Menampilkan matakuliah yang sama-sama diambil antara dua mahasiswa (kombinasi mahasiswa bebas)



- 2) Buatlah sebuah program menggunakan konsep dictionary dengan inputan kalimat dari user kemudian output program yang dihasilkan adalah huruf apa saja dalam kalimat tersebut beserta jumlah hurufnya. Contoh output program:

```

Run: asd asd
masukkan kata :teknik informatika
('a', 2)
('e', 1)
('f', 1)
('i', 3)
('k', 3)
('m', 1)
('n', 2)
('o', 1)
('r', 1)
('t', 2)
Process finished with exit code 0
  
```



- 3) Pada kebun binatang x terdapat beberapa jenis hewan yang digolongkan menjadi 4 jenis yaitu hewan karnivora, hewan herbivora, hewan berkaki 4, dan hewan berkaki 2. Buatlah program untuk mendata hewan-hewan tersebut, kemudian output program bisa menentukan:

- a) Hewan apa saja yang berjenis omnivora
- b) Hewan karnivora berkaki 2
- c) Hewan karnivora berkaki 4
- d) Hewan herbivora berkaki 2
- e) Hewan herbivora berkaki 4
- f) Hewan omnivora berkaki 2
- g) Hewan omnivora berkaki 4

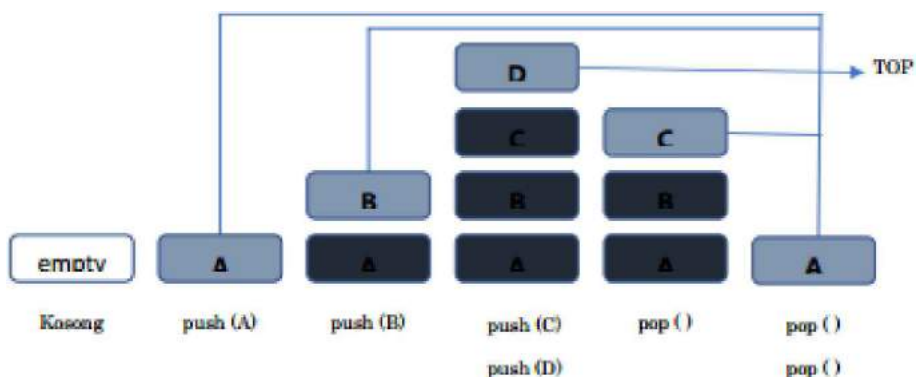
## BAB 4. STACK & QUEUE

### 4.1 Tujuan Pembelajaran

- 1) Memahami konsep dasar *Stack & Queue*
- 2) Menganalisis permasalahan menggunakan konsep *Stack & Queue*
- 3) Membuat program penyelesaian masalah menggunakan konsep *Stack & Queue*

### 4.2 Stack

Stack adalah struktur data linier yang menggunakan konsep LIFO (Last In First Out), dimana data yang dimasukkan akan diletakkan di atas data yang sudah ada. Untuk operasinya, akses data pada stack hanya terdapat pada satu point yang disebut dengan top. Data baru yang dimasukkan akan otomatis menumpuk data yang sudah ada sebelumnya dan posisinya akan menjadi top. Sebaliknya, ketika data pada pop diambil maka posisi top akan digantikan dengan data di bawah top sebelumnya.





Gambar 4.1 Ilustrasi struktur data stack

Pada ilustrasi stack gambar 4.1, operasi stack yang digunakan bisa dijabarkan pada tabel berikut:

| Operasi      | Pre          | Post         | Return Value | Ket                           |
|--------------|--------------|--------------|--------------|-------------------------------|
| Inisialisasi | Stack kosong | Stack kosong |              | Inisialisasi awal stack       |
| s.push (A)   | Stack kosong | A            |              | Top = A                       |
| s.push (B)   | A            | A, B         |              | Top = B                       |
| s.push (C)   | A, B         | A, B, C      |              | Top = C                       |
| s.peek       | A, B, C      | A, B, C      | C            | Menampilkan nilai Top         |
| s.push (D)   | A, B, C      | A, B, C, D   |              | Top = D                       |
| len(s)       | A, B, C, D   | A, B, C, D   | 3            | Menampilkan jumlah data stack |
| s.pop ()     | A, B, C, D   | A, B, C      |              | Top = C                       |
| s.isEmpty () | A, B, C      | A, B, C      | False        | Stack tidak kosong            |
| s.pop ()     | A, B, C      | A, B         |              | Top = B                       |
| s.pop ()     | A, B         | A            |              | Top = A                       |

Tabel 4.1 Proses operasi pada stack

Selain operasi push (menambah) dan pop (mengambil) data yang digunakan pada stack, ada beberapa operasi lainnya pada penggunaan stack, yaitu:

| Operasi      | Fungsi  |
|--------------|---|
| s.push ()    | menambah data pada top stack  |
| s.pop ()     | mengambil data top pada stack   |
| s.peek ()    | menampilkan data top pada stack (stack tidak boleh kosong)                  |
| s.isEmpty () | memeriksa stack dalam keadaan kosong (True), atau (False) jika tidak kosong |
| s.__len__ () | menghitung jumlah data pada stack   |

Tabel 4.2 Operasi lain pada stack

### 4.3 Implementasi Stack Pada Python

```
class Stack:
    def __init__(self):
        self.items = []

    def pop(self):
        if self.isEmpty():
            raise RuntimeError("percobaan pop pada stack
kosong")

        topIdx = len(self.items)-1
        item = self.items[topIdx]
        del self.items[topIdx]
        return item

    def push(self, item):
        self.items.append(item)

    def top(self):
        if self.isEmpty():
            raise RuntimeError("percobaan akses top pada
stack kosong")
        topIdx = len(self.items)-1
        return self.items[topIdx]

    def isEmpty(self):
        return len(self.items) == 0

def main():
    s = Stack()
    lst = list(range(10))
    lst2 = []
    for k in lst:
        s.push(k)
    if s.top() == 9:
        print("Tes 1 berhasil")
    else:
        print("Tes 1 gagal")

    while not s.isEmpty():
        lst2.append(s.pop())
```

```

lst2.reverse()
if lst2 != lst:
    print("Tes 2 gagal")
else:
    print("Tes 2 berhasil")

try:
    s.pop()
    print("Tes 3 gagal")

except RuntimeError:
    print("Tes 3 berhasil")
except:
    print("Tes 3 gagal")

try:
    s.top()
    print("Tes 4 gagal")

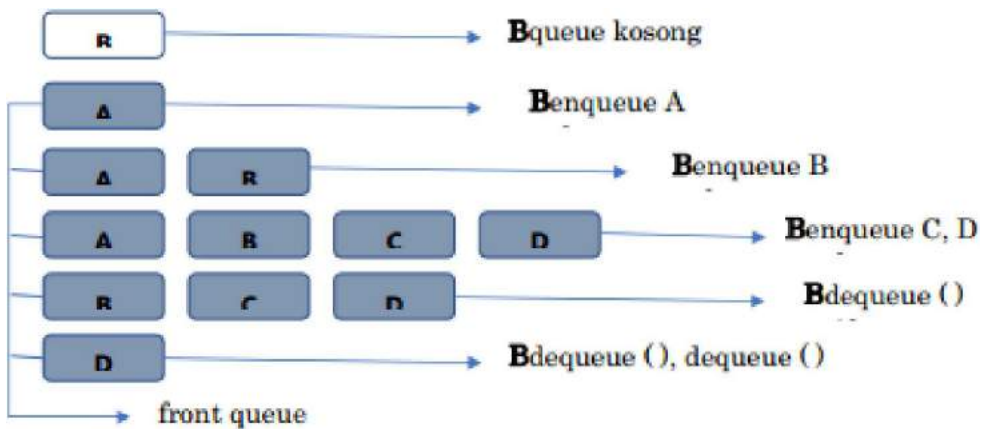
except RuntimeError:
    print("Tes 4 berhasil")
except:
    print("Tes 4 gagal")

if __name__ == "__main__":
    main()

```

#### 4.4 Queue

Hampir sama seperti stack, Queue merupakan struktur data linier yang juga dibatasi pada satu point dalam operasi akses nya. Perbedaannya yaitu queue menggunakan konsep FIFO (First In First Out), yaitu data yang dimasukkan (enqueue) akan diletakkan pada posisi paling belakang (rear), dan data yang diambil (dequeue) adalah data pada posisi (front) atau yang paling awal pada queue.



Gambar 4.2 Ilustrasi struktur data Queue

Pada ilustrasi queue gambar 4.1, operasi queue yang digunakan bisa dijabarkan pada tabel berikut:

| Operasi       | Pre          | Post         | Return Value | Ket                           |
|---------------|--------------|--------------|--------------|-------------------------------|
| Inisialisasi  | Queue kosong | Queue kosong |              | Inisialisasi awal queue       |
| q.enqueue (A) | Queue kosong | A            |              | front = A                     |
| q.enqueue (B) | A            | A, B         |              | front = A, rear = B           |
| q.enqueue (C) | A, B         | A, B, C      |              | front = A, rear = C           |
| q.peek        | A, B, C      | A, B, C      | A            | Menampilkan nilai rear        |
| q.enqueue (D) | A, B, C      | A, B, C, D   |              | front = A, rear = D           |
| len(q)        | A, B, C, D   | A, B, C, D   | 4            | Menampilkan jumlah data queue |
| q.dequeue ()  | A, B, C, D   | B, C, D      |              | front = B, rear = D           |
| q.isEmpty ()  | B, C, D      | B, C, D      | False        | queue tidak kosong            |
| q.dequeue ()  | B, C, D      | C, D         |              | front = C, rear = D           |
| q.dequeue ()  | C, D         | D            |              | front = D                     |

Tabel 4.3 Proses operasi pada queue

Selain operasi enqueue (menambah) dan dequeue (mengambil) data yang digunakan pada queue, ada beberapa operasi lainnya pada penggunaan queue, yaitu:

| Operasi      | Fungsi  |
|--------------|---|
| q.enqueue () | menambah data pada belakang (rear) queue                                    |
| q.dequeue () | mengambil data depan (front) queue dan mengubah [ front+1 ] menjadi front   |
| q.peek ()    | menampilkan data front pada queue (queue tidak boleh kosong)                |
| q.isEmpty () | memeriksa queue dalam keadaan kosong (True), atau (False) jika tidak kosong |
| q.__len__ () | menghitung jumlah data pada queue   |

Tabel 4.4 Proses operasi pada queue

## 4.5 Implementasi Queue Pada Python

```
class Queue:
    def __init__(self):
        self.items = []
        self.frontIdx = 0

    def __compress(self):
        newlst = []
        for i in range(self.frontIdx, len(self.items)):
            newlst.append(self.items[i])

        self.items = newlst
        self.frontIdx = 0

    def dequeue(self):
        if self.isEmpty():
            raise RuntimeError("percobaan dequeue pada queue kosong")

        # meng-kompres queue saat keadaan setengah penuh
```

```

        if self.frontIdx * 2 > len(self.items):
            self.__compress()

        item = self.items[self.frontIdx]
        self.frontIdx += 1
        return item

    def enqueue(self, item):
        self.items.append(item)

    def front(self):
        if self.isEmpty():
            raise RuntimeError("percobaan akses front queue
kosong")

        return self.items[self.frontIdx]

    def isEmpty(self):
        return self.frontIdx == len(self.items)

def main():
    q = Queue()
    lst = list(range(10))
    lst2 = []

    for k in lst:
        q.enqueue(k)

    if q.front() == 0:
        print("Tes 1 berhasil")
    else:
        print("Tes 1 gagal")

    while not q.isEmpty():
        lst2.append(q.dequeue())

    if lst2 != lst:
        print("Tes 2 gagal")
    else:
        print("Tes 2 berhasil")

    for k in lst:

```



```

        q.enqueue(k)

lst2 = []

while not q.isEmpty():
    lst2.append(q.dequeue())

if lst2 != lst:
    print("Tes 3 gagal")
else:
    print("Tes 3 berhasil")

try:
    q.dequeue()
    print("Tes 4 gagal")

except RuntimeError:
    print("Tes 4 berhasil")
except:
    print("Tes 4 gagal")

try:
    q.front()
    print("Tes 5 gagal")

except RuntimeError:
    print("Tes 5 berhasil")
except:
    print("Tes 5 gagal")

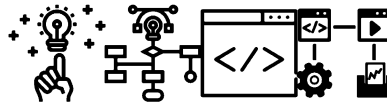
if __name__ == "__main__":
    main()

```

## 4.6 Latihan



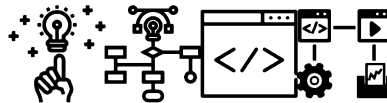
- 1) Pada program implementasi stack dan queue di atas, coba jalankan dan jelaskan program tersebut beserta fungsi-fungsi nya.



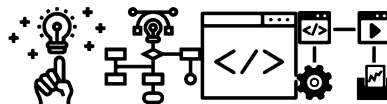
- 2) Buatlah program yang memiliki fungsi untuk membalik kata atau kalimat.

*Input* :informatika  
*Output* :akitamrofni

## 4.7 Studi Kasus



- 1) Buatlah program yang memiliki fungsi mengubah bilangan decimal menjadi bilangan biner.



- 2) Buatlah program penyimpanan data dengan menggunakan konsep stack/queue yang memiliki fungsi paling tidak: menambah, menghapus, dan menampilkan data. tulis juga deskripsi program yang kalian buat.



## BAB 5. LINKED LIST

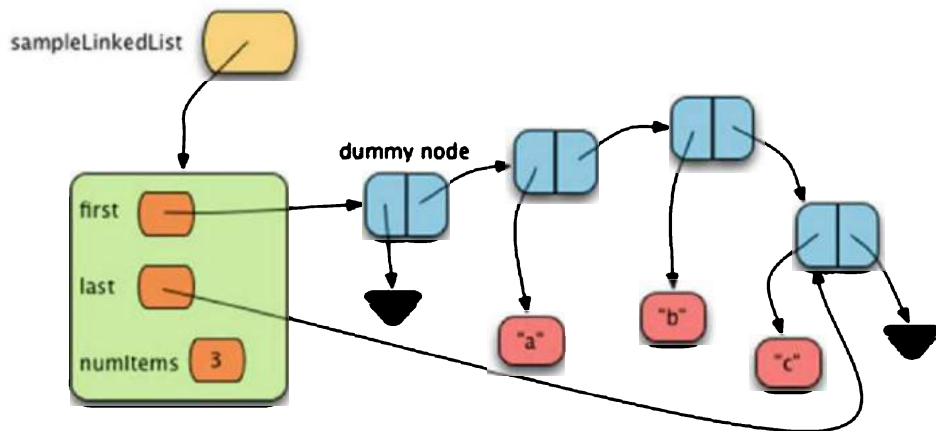
### 5.1 Tujuan Pembelajaran

- 1) Memahami konsep dasar *Linked List*
- 2) Menganalisis permasalahan menggunakan konsep *Linked List*
- 3) Membuat program penyelesaian masalah menggunakan konsep *Linked List*

### 5.2 Single Linked List

Salah satu bentuk struktur data yang berisi kumpulan data yang tersusun secara sekuensial, saling bersambungan, dinamis dan terbatas adalah senarai berkait (linked list). Suatu senarai berkait (linked list) adalah suatu simpul (node) yang dikaitkan dengan simpul yang lain dalam suatu urutan tertentu. Suatu simpul dapat berbentuk suatu struktur atau class.

Secara teori, linked list adalah sejumlah node yang dihubungkan secara linier dengan bantuan pointer. Dikatakan single (singly) linked apabila hanya ada satu pointer yang menghubungkan setiap node. single artinya field pointer-nya hanya satu buah saja dan satu arah.



Gambar 5.1 Ilustrasi Single Linked list  
(Kent D. Lee & Hubbard, 2015)

Dalam implementasi pada Bahasa python, terdapat 4 metode utama dalam membuat class linked list, metode itu di antaranya:

| Operasi   | Fungsi  |
|-----------|---|
| getData() | mendapatkan/mengetahui nilai dalam tiap node      |
| getNext() | mengetahui nilai yang berada pada node setelahnya |
| setData() | mengganti nilai yang terkandung dalam node        |
| setNext() | mendeklarasikan data setelahnya                   |

Tabel 5.1 Operasi dasar Linked list

Selain 4 metode dasar tersebut, terdapat beberapa metode tambahan agar program linked list dapat dijalankan, di antaranya:

| Operasi         | Fungsi                                       |
|-----------------|--|
| unOrderedList() | membuat suatu linked list kosong             |
| add(item)       | menambahkan data pada node suatu linked list |

|                           |   |
|---------------------------|---|
| <code>remove(item)</code> | menghapus data pada node suatu linked list  |
| <code>search(item)</code> | mencari item dalam linked list, metode ini akan menghasilkan nilai kembalian Boolean (T/F)                        |
| <code>isEmpty()</code>    | memeriksa linked list dalam kondisi kosong atau tidak, metode ini akan menghasilkan nilai kembalian Boolean (T/F) |
| <code>size ()</code>      | menampilkan nilai kembalian jumlah data pada linked list  |

Tabel 5.2 Operasi tambahan Linked list

### 5.3 Struktur Pemrograman Single Linked List pada Python

Pada struktur pemrograman menggunakan tree hal yang pertama dilakukan adalah mendeklarasikan *Node* atau data yang akan disimpan pada linked list. *Node* ini akan dideklarasikan pada sebuah class node yang mempunyai dua atribut dasar yaitu `self.data = initdata` yang berfungsi menyimpan data pada variabel global data dan `self.next = None` yang berarti setiap node akan memiliki link dengan tujuan pointer None (Null)

---

```
class Node:
    def __init__(self,initdata):
        self.data = initdata
        self.next = None
```

## 5.4 Implementasi program single linked list pada python (1) :

```

class Node:
    def __init__(self, initdata):
        self.data = initdata
        self.next = None

    def getData(self):
        return self.data

    def getNext(self):
        return self.next

    def setData(self, newdata):
        self.data = newdata

    def setNext(self, newnext):
        self.next = newnext

class UnorderedList:

    def __init__(self):
        self.head = None

    def isEmpty(self):
        return self.head == None

    def add(self, item):
        temp = Node(item)
        temp.setNext(self.head)
        self.head = temp

    def size(self):
        current = self.head
        count = 0
        while current != None:
            count = count + 1
            current = current.getNext()

        return count

```

```

def search(self,item):
    current = self.head
    found = False
    while current != None and not found:
        if current.getData() == item:
            found = True
        else:
            current = current.getNext()

    return found

def remove(self,item):
    current = self.head
    previous = None
    found = False
    while not found:
        if current.getData() == item:
            found = True
        else:
            previous = current
            current = current.getNext()

    if previous == None:
        self.head = current.getNext()
    else:
        previous.setNext(current.getNext())

mylist = UnorderedList()

mylist.add(31)
mylist.add(77)
mylist.add(17)
mylist.add(93)
mylist.add(26)
mylist.add(54)

print(mylist.size())
print(mylist.search(93))
print(mylist.search(100))

mylist.add(100)
print(mylist.search(100))

```

```

print(mylist.size())

mylist.remove(54)
print(mylist.size())
mylist.remove(93)
print(mylist.size())
mylist.remove(31)
print(mylist.size())
print(mylist.search(93))

```

## 5.5 Implementasi program single linked list pada python (2) :

```

import os

class Node(object):

    def __init__(self, data=None, next_node=None):
        self.data = data
        self.next_node = next_node

    def get_data(self):
        return self.data

    def get_next(self):
        return self.next_node

    def set_next(self, new_next):
        self.next_node = new_next

class LinkedList(object):
    def __init__(self, head=None):
        self.head = head

    def insert(self, data):
        new_node = Node(data)
        new_node.set_next(self.head)
        self.head = new_node

    def size(self):
        current = self.head
        count = 0

```

```

    while current:
        count += 1
        current = current.get_next()
    return count

def search(self, data):
    current = self.head
    found = False
    while current and found is False:
        if current.get_data() == data:
            found = True
        else:
            current = current.get_next()
    return found

def delete(self, data):
    current = self.head
    previous = None
    found = False
    while current and found is False:
        if current.get_data() == data:
            found = True
        else:
            previous = current
            current = current.get_next()
    if current is None:
        raise ValueError("Data not in list")
    if previous is None:
        self.head = current.get_next()
    else:
        previous.set_next(current.get_next())

def showData(self):
    os.system('clear')
    print ("Tampilkan list data:")
    print ("Node -> Next Node")
    current_node = self.head
    while current_node is not None:
        print (current_node.data),
        print ("    ->"),
        print (current_node.next_node.data) if
hasattr(current_node.next_node, "data") else None

```



```

        current_node = current_node.next_node

    def mainmenu(self):
        pilih = "y"
        while (pilih == "y"):
            os.system("clear")
            print("=====")
            print("|  Menu aplikasi linked list  |")
            print("=====")
            print("1. Insert data")
            print("2. Delete data")
            print("3. Cari data")
            print("4. Panjang dari linked list")
            print("5. Tampil data")
            print("=====")
            pilihan=str(input(("Silakan masukan pilihan
anda: ")))
            if(pilihan=="1"):
                os.system("clear")
                obj = str(input("Masukan data yang ingin anda
tambahkan: "))
                self.insert(obj)
            elif(pilihan=="2"):
                os.system("clear")
                obj = str(input("Masukan data yang ingin anda
dihapus: "))
                self.delete(obj)
                x = raw_input("")
            elif(pilihan=="3"):
                os.system("clear")
                obj = str(input("Masukan data yang ingin anda
dicari: "))
                status = self.search(obj)
                if status == True:
                    print("Data ditemukan pada list")
                else:
                    print("Data tidak ditemukan")
                x = raw_input("")
            elif(pilihan=="4"):
                os.system("clear")
                print("Panjang dari queue adalah:

```



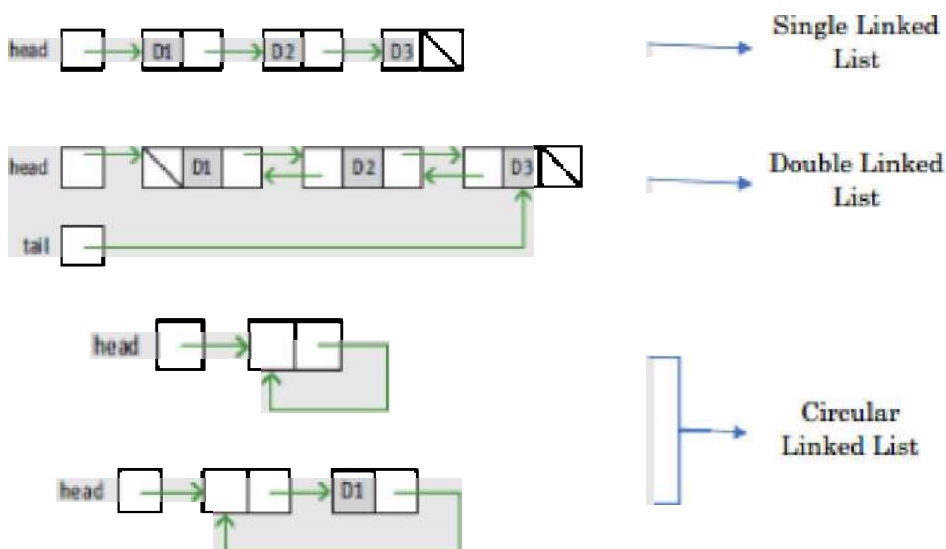
```

"+str(self.size()))
        x = raw_input("")
        elif(pilihan=="5"):
            os.system("clear")
            self.showData()
            x = raw_input("")
        else:
            pilih="n"

if __name__ == "__main__":
    l = LinkedList()
    l.mainmenu()

```

## 5.6 Latihan



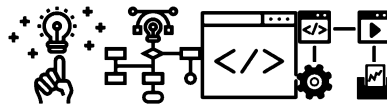
(Lambert, 2010)

- 1) Perhatikan ketiga ilustrasi di atas, merupakan jenis-jenis dari linked list. Menurut pengamatan kalian dari gambar di atas, adakah perbedaan dari ketiga jenis linked tersebut? Jelaskan !

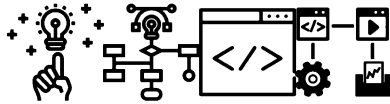


- 2) Cobalah program implementasi single linked list 1 dan 2. Berikan penjelasan pada setiap blok fungsi pada program tersebut.
- 3) Pada bab ini kalian telah mengetahui struktur pemrograman single linked list pada python. Berdasarkan pengamatan kalian pada latihan nomor 1, buatlah deklarasi class node pada double linked list dan circular linked list.

## 5.7 Studi Kasus



- 1) Pada rumah sakit y terdapat sistem antrian untuk pasien yang akan berobat. Setiap pasien baru wajib mendaftarkan diri untuk mendapatkan nomor antrian. Buatlah program untuk mencatat antrian pasien dengan ketentuan:
  - a) Setelah mendaftar pasien dapat mengetahui nomor antrian dan jumlah antrian yang ada
  - b) Dengan asumsi setiap pasien mendapatkan jatah waktu yang sama ketika berobat (misal 10 menit setiap pasien), tampilkan perkiraan waktu tunggu antrian pasien sampai gilirannya



2) Sebuah online shop ingin membuat program untuk memudahkan daftar input data transaksi pembelian untuk diproses. Dimana dalam program ini terdapat 2 macam metode input, di antaranya:

- a) Untuk pelanggan yang memesan barang dan membayar DP (input awal)
- b) Untuk pelanggan yang memesan barang dan sudah melunasi pembayaran (input tertentu)

Kemudian metode proses untuk menghapus data dari daftar yang akan diproses dalam transaksi selanjutnya (tidak ada di program) ada 2 metode proses, di antaranya:

- a) Proses data transaksi yang sudah lunas (proses awal)
- b) Proses data transaksi tertentu (proses tertentu)

## BAB.6 TREE

### 6.1 Tujuan Pembelajaran

- 1) Memahami konsep dasar *Tree*
- 2) Menganalisis permasalahan menggunakan konsep *Tree*
- 3) Membuat program penyelesaian masalah menggunakan konsep *Tree*

### 6.2 Tree

Setelah mempelajari mengenai struktur data linier sebelumnya seperti stack, queue, dan linked list pada chapter ini kita mempelajari tentang struktur data yang memiliki struktur bertingkat (*hierarchical structure*) yaitu tree(pohon). Sesuai namanya, struktur data tree dapat dianalogikan sebagai pohon yang memiliki beberapa bagian seperti akar, cabang, dan daun. Untuk implementasinya, tree dapat merepresentasikan silsilah keluarga, struktur buku, struktur divisi pada perusahaan, dsb.

Hampir sama seperti linked list, sebuah elemen dari tree adalah sebuah node, yang berbeda adalah setiap node dari linked list menunjuk kearah node sebelum/sesudahnya (linier) sedangkan node dari tree memiliki struktur bertingkat seperti sebuah pohon.

### 6.3 Istilah dalam Tree

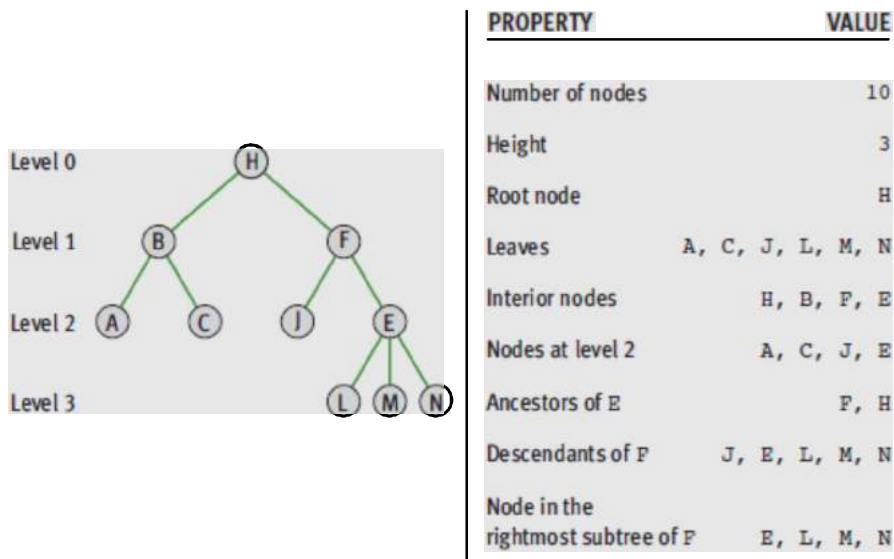
Struktur data tree merupakan struktur data bertingkat (*hierarchical structure*) atau struktur data yang memiliki tingkatan pada setiap elemen atau data yang terdapat didalamnya. Pada

struktur data tree terdapat struktur yang hampir sama pada analogi keluarga pohon. Terdapat beberapa istilah yang diambil dari analogi keluarga pohon ini pada struktur data tree seperti *root*, *child*, *parent*, *leaf*, dsb. Pada tabel 6.1 menunjukkan istilah-istilah pada tree beserta penjelasannya.

| Istilah          | Definisi   |
|------------------|--|
| Node             | Setiap elemen pada tree.   |
| Root             | Node teratas pada tree. Node ini tidak memiliki parent.  |
| Child            | Node yang terhubung langsung di bawah parent.  |
| Parent           | Node di atas node lain secara langsung.  |
| Siblings         | Node lain yang memiliki parent yang sama.  |
| Leaf             | Node yang tidak memiliki children.   |
| Interior Node    | Node yang memiliki setidaknya satu child.  |
| Edge/Branch/Link | Garis yang menghubungkan antara parent ke child.   |
| Descendant       | Node yang berada di bawah node itu sendiri (children, children's children, hingga ke leaf)               |
| Ancestor         | Node yang berada di atas node itu sendiri (parent, parent's parent, hingga ke root)                      |
| Path             | Urutan node yang menghubungkan ke descendant nya   |
| Depth or Level   | Tingkat atau kedalaman dari suatu tree, dimulai dari root(0), children(1), children's children (2), dst. |
| Height           | Panjang jalur terpanjang pada tree   |

Tabel 6.1 istilah pada tree beserta penjelasannya

Gambar 6.1 berikut ini menunjukkan ilustrasi data struktur tree



Gambar 6.1 Ilustrasi sebuah tree dan bagian-bagiannya  
(Lambert, 2010)

## 6.4 Struktur Pemrograman Tree pada Python

### 6.4.1 Deklarasi class Node

Pada struktur pemrograman menggunakan tree hal yang pertama dilakukan adalah mendeklarasikan *Node* atau data yang akan disimpan pada tree. *Node* ini akan dideklarasikan pada sebuah class node yang mempunyai tiga atribut dasar yaitu `self.l = None` dan `self.r = None` yang berarti setiap node akan memiliki dua link dengan tujuan None (Null), serta `self.v = val` yang berfungsi menyimpan nilai data pada variabel val.

---

```
class Node:
    def __init__(self, val):
        self.l = None
        self.r = None
        self.v = val
```

#### 6.4.2 Insert data

Setelah membuat deklarasi node, operasi dasar selanjutnya adalah insert data. Pada operasi ini dibuat dua fungsi utama, yaitu fungsi pertama dilakukan jika belum ada node sama sekali pada tree maka data pertama yang dimasukkan akan menjadi root. Pada fungsi kedua yaitu penambahan data jika sudah ada data pada tree. Pada tahap ini nilai yang akan dimasukkan pada tree akan diperiksa terlebih dahulu dan dibandingkan dengan data yang ada. Jika data lebih besar dari root akan diletakkan ke sebelah kanan root dan sebaliknya. Jika root mempunyai child maka akan setelah dibandingkan dengan root maka akan dibandingkan dengan child pertama, jika lebih besar akan dilanjutkan ke sebelah kanan child dan sebaliknya.

---

```
def add(self, val):
    if(self.root == None):
        self.root = Node(val)
    else:
        self._add(val, self.root)

    def _add(self, val, node):
        if(val < node.v):
            if(node.l != None):
                self._add(val, node.l)
            else:
                node.l = Node(val)
        else:
            if(node.r != None):
```

```

        self._add(val, node.r)
    else:
        node.r = Node(val)

```

## 6.5 Binary Search Tree (BST)

Binary search tree (BST) atau biasa disebut dengan sorted binary tree merupakan struktur data hierarkis yang mempunyai fungsi menyimpan data dalam sebuah tree. Ciri-ciri pada BST adalah nodes pada sebelah kiri dari subtree mempunyai nilai yang lebih kecil dari node di atasnya, dan nodes yang terletak sebelah kanan subtree nilainya lebih besar dari node di atasnya. Dalam implementasi BST, pengaksesan data atau pengoperasian data dapat dilakukan dengan relatif cepat. Hal ini dikarenakan data dilengkapi dengan informasi key atau kunci. BST menempatkan key tersebut secara urut yang menyebabkan pengaksesan data dapat dilakukan dengan cepat. BST mempunyai tiga operasi utama yaitu: insertion (memasukkan data), deletion (menghapus data), dan searching (mencari data).

Sebagai contoh, terdapat list data (5, 1, 3, 7, 4, 8, 9, 6, 2). Data tersebut akan dimasukkan secara urut dalam BST. Langkah-langkah dalam memasukkan data tersebut sebagai berikut:

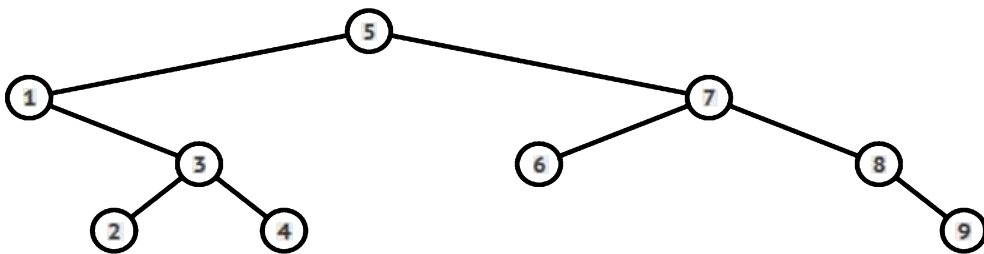
- 1) Data pertama (5) otomatis akan menjadi root.
- 2) Data kedua (1), akan dibandingkan dengan root, karena lebih kecil maka diletakkan pada sebelah kiri root.
- 3) Data ketiga (3), akan dibandingkan dengan root, karena lebih kecil maka diteruskan pada sebelah kiri root, karena sebelah kiri root telah terdapat data, maka akan dibandingkan dengan data tersebut (1). Karena nilainya



lebih besar, maka data dimasukkan kesebelah kanan subtree.

- 4) Data keempat (7), akan dibandingkan dengan root, karena lebih besar maka diletakkan pada sebelah kanan root.
- 5) Begitu seterusnya hingga data terakhir.

Setelah data berhasil dimasukkan dalam BST, semua maka ilustrasi BST akan menjadi seperti gambar 6.2 berikut.



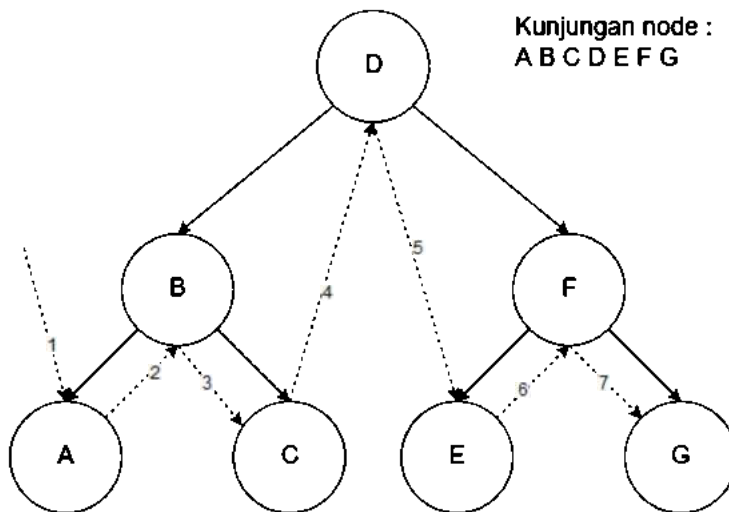
Gambar 6.2 Ilustrasi BST dari insertion data (5, 1, 3, 7, 4, 8, 9, 6, 2)  
(Lambert, 2010)

## 6.6 Tree Traversal

Tree traversal merupakan algoritma pada tree untuk mengakses node yang terdapat pada tree menggunakan path-path tertentu. Terdapat tiga tree traversal yang bisa diimplementasikan yaitu in-order, pre-order, dan post-order.

## 1) Inorder

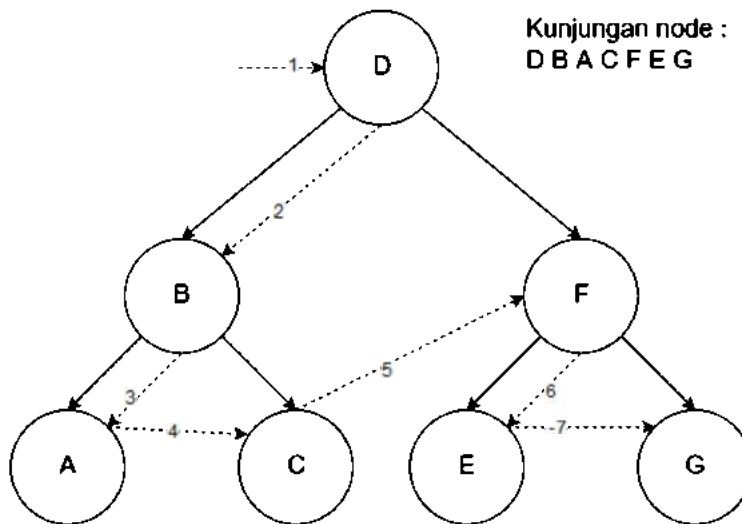
- a) Secara rekursif mencetak seluruh data pada subpohon kiri
- b) Cetak data pada root
- c) Secara rekursif mencetak seluruh data pada subpohon kanan



Gambar 6.3 In-order traversal

## 2) Pre-order

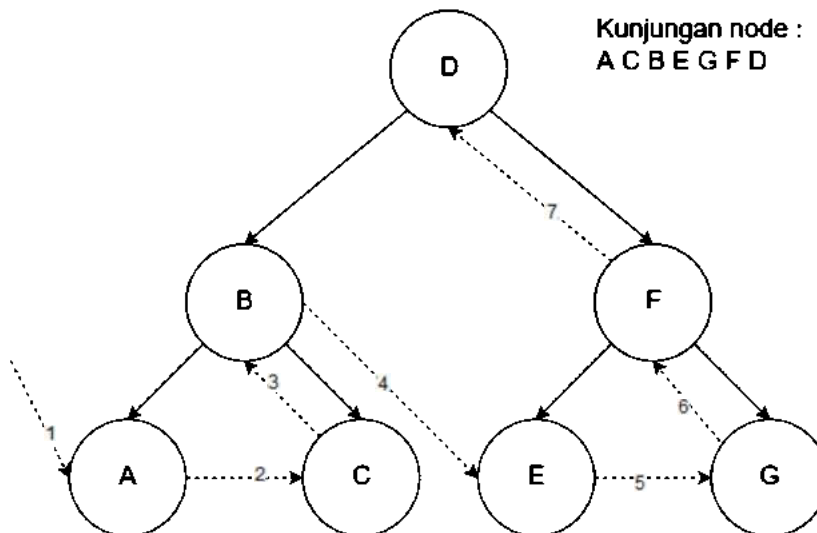
- a) Cetak data pada root
- b) Secara rekursif mencetak seluruh data pada subpohon kiri
- c) Secara rekursif mencetak seluruh data pada subpohon kanan



Gambar 6.4 Pre-order traversal

### 3) Post-order

- Secara rekursif mencetak seluruh data pada subpohon kiri
- Secara rekursif mencetak seluruh data pada subpohon kanan
- Cetak data pada root



Gambar 6.3 In-order traversal

## 6.7 Implementasi program Tree pada python:

```

class Node:
    def __init__(self, val):
        self.l = None
        self.r = None
        self.v = val

class Tree:
    def __init__(self):
        self.root = None

    def getRoot(self):
        return self.root

    def add(self, val):
        if(self.root == None):
            self.root = Node(val)
        else:
            self._add(val, self.root)

    def _add(self, val, node):
        if(val < node.v):
            if(node.l != None):
                self._add(val, node.l)
            else:
                node.l = Node(val)
        else:
            if(node.r != None):
                self._add(val, node.r)
            else:
                node.r = Node(val)

    def find(self, val):
        if(self.root != None):
            return self._find(val, self.root)
        else:
            return None

    def _find(self, val, node):
        if(val == node.v):
            return node
        elif(val < node.v and node.l != None):
            self._find(val, node.l)

```

```

        elif(val > node.v and node.r != None):
            self._find(val, node.r)

    def deleteTree(self):
        # garbage collector will do this for us.
        self.root = None

    def printTree(self):
        if(self.root != None):
            self._printTree(self.root)

    def _printTree(self, node):
        if(node != None):
            self._printTree(node.l)
            print str(node.v) + ' '
            self._printTree(node.r)

#      3
# 0    4
#  2    8
tree = Tree()
tree.add(3)
tree.add(4)
tree.add(0)
tree.add(8)
tree.add(2)
tree.printTree()
print (tree.find(3)).v
print tree.find(10)
tree.deleteTree()
tree.printTree()

```

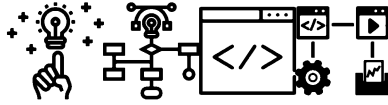
## 6.8 Latihan

- 1) Bukalah link berikut, <https://visualgo.net/en/bst> , pada link tersebut terdapat visualisasi dari BST. Pada sub-menu create, buatlah data kosong, kemudian masukkan sejumlah data dengan menggunakan fungsi insert. Masukkan data sesuai dengan keinginan kalian. Setelah itu lakukan langkah-langkah berikut:

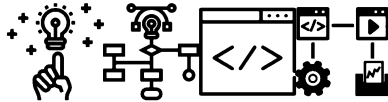


- a) Amati algoritma pada setiap data yang dimasukkan pada bagian kanan-bawah halaman.
- b) Kemudian cobalah menghapus dan mencari data dengan fungsi remove dan search. Amati algoritmanya.
- c) Jelaskan menurut bahasa kalian sendiri tentang proses insert, delete, dan search data berdasarkan algoritma yang terdapat pada visualisasi.

## 6.9 Studi Kasus



- 1) Buat program tree dengan fungsi inputan dari user minimal insert, search dan delete suatu data.



- 2) Dari program nomor 1, tambahkan fungsi yang dapat menampilkan yang dapat menampilkan tree traversal pre-order, inorder, dan post-order.

## DAFTAR PUSTAKA

- Foundation, P. S. (n.d.). Python 2.7.14 documentation. Retrieved from <https://docs.python.org/2/>
- Halim, S. (n.d.). visualising data structures and algorithms through animation. Retrieved from [visualgo.net](http://visualgo.net)
- Kent D. Lee, & Hubbard, S. (2015). *Data Structures and Algorithms with Python*. Springer. <https://doi.org/10.1007/978-3-319-13072-9>
- Lambert, K. A. (2010). *Fundamentals of Python*. Boston, Massachusetts: Course Technology.
- Lubanovic, B. (2015). *Introducing Python*. O'Reilly. <https://doi.org/10.1017/CBO9781107415324.004>
- Matthes, E. (2016). *Python Crash Course*. San Francisco: No Starch Press.
- Necaise, R. D. (2011). *Data Structures and Algorithms Using Python*. Denver: John Wiley & Sons. <https://doi.org/10.1017/CBO9780511547010>
- Programiz. (n.d.). Python Programming Examples. Retrieved from <https://www.programiz.com/python-programming/examples>
- Technology, M. I. of. (n.d.). Introduction to Computer Science and Programming in Python. Retrieved from <https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/>



# LAMPIRAN

## **Kunci Jawaban Bab.1**

### Latihan



1. Menjelaskan pre dan post condition dari contoh program poin 4.1 - 4.4 dari *link* <https://docs.python.org/2/tutorial/controlflow.html> beserta deskripsi penjelasan program.

### Poin 4.1

- Pre:  
User input
- Post:  
Konversi bilangan ke string (*“negative change to zero”, “zero”, “single”, atau “more”*)
- Deskripsi:  
Program if (kondisi) yang berfungsi merubah bilangan inputan menjadi output berupa string sesuai bilangan yang diinputkan. Pada kondisi awal bilangan akan diseleksi, jika bilangan  $< 0$  maka akan dirubah menjadi 0 dan mencetak output string *“negative change to zero”*. Jika kondisi pertama tidak sesuai akan dilanjutkan dengan kondisi kedua, jika bilangan  $= 0$  maka akan mencetak output *“string”*. Jika kondisi kedua tidak sesuai akan dilanjutkan dengan kondisi ketiga, jika bilangan  $= 1$  maka akan mencetak output *“single”*. Jika kondisi ketiga tidak sesuai

akan dilanjutkan dengan kondisi terakhir, jika bilangan  $> 1$  maka akan mencetak output “more”.

#### Poin 4.2

- Pre:  
Kumpulan string dalam list
- Post:  
Cetak string pada list secara berurutan beserta panjang karakter pada tiap string
- Deskripsi:  
Program for (perulangan) yang berfungsi mengakses string dalam list menggunakan fungsi for. String yang diakses kemudian akan dicetak sebagai output secara berurutan dan ditambah fungsi len untuk menghitung berapa banyak karakter pada string tersebut.

#### Poin 4.3

- Pre:  
-
- Post:  
Output hasil dari *range function* (jangkauan)
- Deskripsi :  
*Range function* (jangkauan) merupakan fungsi *built-in* pada python yang berfungsi untuk melakukan perulangan secara sekuensial dengan awal, akhiran, dan batas perulangan yang dapat ditentukan.

#### Poin 4.4

- Pre:
  - a. `n in range 2, 10 (2-9)`
  - b. `num in range 2, 10 (2-9)`
- Post:
  - a. Output `n` = bilangan prima atau faktorial dari 2
- Deskripsi:

*Range function* (jangkauan) merupakan fungsi *built-in* pada python yang berfungsi untuk melakukan perulangan secara sekuensial dengan awal, akhiran, dan batas perulangan yang dapat ditentukan.

## **Kunci Jawaban Bab.2**

### **Latihan**



2. Menjelaskan pre dan post condition beserta deskripsi penjelasan program. dari contoh program dari link:

[https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/lec5\\_tuples\\_lists.py](https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/lec5_tuples_lists.py)

#### **Contoh 1 (*Returning a tuple*)**

- Pre:  
Deklarasi variabel  $(x, y) = (5, 3)$
- Post:  
Output hasil *floor division* dan modulus dari x dan y
- Deskripsi:  
Program mendeklarasikan fungsi “quotient\_and\_remainder” dengan variabel global x dan y. dalam fungsi terdapat proses untuk menghitung *floor division* (berapa kali pembagian yang dapat dilakukan antara dua bilangan) dan modulus (sisir bagi) antar dua bilangan yang disimpan pada variabel global x dan y.

**Contoh 2 (*Iterating over tuples*)**

- Pre:  
Deklarasi dua tuple kosong dengan nama “nums” dan “words”
- Post:  
Output hasil string yang dikombinasikan dengan perulangan data pada tuple.
- Deskripsi:  
Perulangan data pada tuple menggunakan fungsi “for” yang dikombinasikan dengan string dan menghasilkan output yang berkaitan dengan data minimal, maximal, dan jumlah data pada tuple.

**Contoh 3 (*Sum of elements in a list*)**

- Pre:  
Deklarasi dua fungsi list (sum\_elem\_method1 dan sum\_elem\_method2).
- Post:  
Output hasil penjumlahan dari data di dalam list.
- Deskripsi:  
Dua fungsi list memiliki fungsi yang sama yaitu menjumlahkan data didalam list menggunakan perulangan (for) pada setiap elemen didalam list.

**Contoh 4 (*Various list operations*)**

- Pre:  
Deklarasi tiga buah list (L1, L2, dan L3).
- Post:  
Output hasil operasi dasar pada list yaitu; split, join, sort, dan reverse.
- Deskripsi:  
Operasi dasar pada list; split(memisahkan atau membagi data didalam list), join(mennambahkan character paa element list), sort(mengurutkan elemen data dalam list), reverse(mengakses list dengan indeks yang terbalik).

**Contoh 5 (*Aliasing*)**

- Pre:  
Deklarasi variabel a, b, warm, dan hot.
- Post:  
Output hasil penamabahan string pada list.
- Deskripsi:  
Pada python memungkinkan untuk memberikan nilai variabel yang merujuk pada variabel lain. Pada contoh program, syntax “b=a” menunjukkan bahwa nilai b sama dengan nilai a begitu juga “hot = warm” berarti nilai hot sama dengan nilai pada warm. syntax append pada program berfungsi untuk menambahkan data pada list di indeks terakhir.

**Contoh 6 (*Cloning*)**

- Pre:  
Deklarasi “cool” dan “chill”.
- Post:  
Output data elemen pada variabel “cool” dan “chill”.
- Deskripsi:  
Hampir sama pada contoh program sebelumnya python memungkinkan untuk menggandakan (clone) variabel lain dengan memberikan nilai variabel merujuk pada variabel yang lain. Pada program ini variabel “cool” yang telah berisi list di gandakan pada variabel lain bernama “chill”. Kemudian variabel “chill” diberikan fungsi “append” (menambahkan data pada list di akhir indeks).

**Contoh 7 (*Sorting with/without mutation*)**

- Pre:  
Deklarasi list bernama “warm” dan “cool”.
- Post:  
Output data elemen pada variabel “warm” dan “cool” yang telah diurutkan.
- Deskripsi:  
Menampilkan output elemen didalam list yang telah diurutkan dengan menggunakan fungsi “sorted”.

**Contoh 8 (*Lists of lists of lists*)**

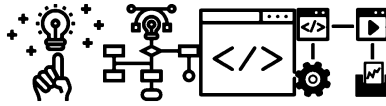
- Pre:  
Deklarasi list bernama “warm”, “hot”, dan “brightcolors”.
- Post:  
Output list sesuai dengan urutan syntax yang ditulis.
- Deskripsi:  
Pada list nilai dalam suatu variabel bersifat dinamis dan memungkinkan untuk berubah nilainya. Pada contoh program variabel “brightcolors” berubah-ubah sesuai dengan fungsi yang ditambahkan pada syntax dan hasil output sesuai dengan kondisi variabel yang terakhir dideklarasikan.

**Contoh 9 (*Mutating a list while iterating over it*)**

- Pre:  
Deklarasi fungsi bernama “remove\_dups” dan “remove\_dups\_new” dengan variabel global “L1” dan “L2”
- Post:  
Output list “L1” dan “L2” setelah dilakukan proses fungsi “remove\_dups” dan “remove\_dups\_new”.
- Deskripsi:  
“remove\_dups” berfungsi untuk menghapus elemen pada “L1” yang sama dengan “L2”. Sedangkan “remove\_dups\_new” berfungsi untuk menghapus elemen pada “L1” (baru) yang sama dengan “L2”.



## Studi Kasus



1. Menampilkan 5 elemen pertama dan terakhir dari list. Dimana elemen dalam list adalah bilangan pangkat dua dari bilangan 1 dan 20.
  - Pre:
 

Deklarasi list kosong.
  - Post:
 

Output 5 elemen pertama dan terakhir dari list
  - Deskripsi:
 

Program menggunakan list yang berisi elemen berupa nilai pangkat dua dari bilangan 1 sampai 20. Output berupa 5 elemen pertama dan terakhir dari list tersebut.
  - Solusi:
 

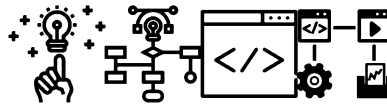
Membuat list dan mengisi elemen list dengan dengan fungsi; (1) range untuk menentukan awal dan akhir bilangan, (2) append untuk menambahkan elemen dalam list, (3) operasi (\*\*) sebagai fungsi kuadrat, (4)syntax “L[:5]” berfungsi untuk mengakses 5 elemen pertama, (5)syntax “L[-5:]” berfungsi untuk mengakses 5 elemen terakhir.
  - Algoritma:
    1. Buat list kosong
    2. Gunakan perulangan, tentukan awal dan akhir bilangan.
    3. Tambahkan elemen menggunakan fungsi kuadrat.

4. Tampilkan 5 elemen awal dan akhir dari list.

- Kode program

```
def printValues():
    l = list()
    for i in range(1,21):
        l.append(i**2)
    print(l[:5])
    print(l[-5:])

printValues()
```



2. Menentukan bilangan kedua terkecil dalam list.

- Pre:

Deklarasi list kosong.

- Post:

Output elemen terkecil kedua.

- Deskripsi:

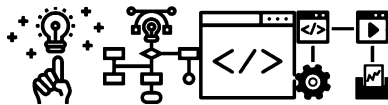
Program menggunakan list yang berisi elemen beberapa bilangan. Dari elemen pada list tersebut output berupa nilai terkecil kedua dalam list.

- Solusi:

Membuat fungsi untuk menentukan nilai terkecil kedua dalam elemen list. Fungsi menggunakan perulangan dan perbandingan kondisi dari elemen pada list tersebut.

- Algoritma:
  1. Deklarasi nama fungsi dengan variabel global (numbers)
  2. Deklarasi variabel a1 dan a2 dengan tipe data float
  3. Gunakan perulangan pada numbers.
  4. Jika  $x \leq a1$ , maka  $a1 = x$  dan  $a2 = a1$
  5. Jika  $x < a2$ , maka  $a2 = x$ .
  6. Nilai kembalian adalah a2 yang merupakan nilai terkecil kedua dalam list.
- Kode program

```
def second_smallest(numbers):
    a1, a2 = float('inf'), float('inf')
    for x in numbers:
        if x <= a1:
            a1, a2 = x, a1
        elif x < a2:
            a2 = x
    return a2
print(second_smallest([1, 2, -8, -2, 0]))
```



3. Membuat program dengan inputan string kata/kalimat, dan menghasilkan output berupa huruf beserta jumlahnya yang terdapat pada string tersebut.
  - Pre:
 

Input string dari user.
  - Post:
 

Output huruf yang ada pada string beserta jumlahnya.

- Deskripsi:

Program menggunakan list yang karakter dari string inputan dari user. Output akan menghasilkan huruf yang ada pada string beserta jumlahnya.

- Solusi:

Membuat list kosong yang akan diisi string dari inputan user. Kemudian membuat list baru yang memiliki 26 (jumlah abjad) elemen dengan value 0. Kemudian dilakukan perulangan untuk menghitung karakter dalam string. Kemudian karakter yang sama pada list akan dihapus dan dijadikan list baru. Terakhir output akan menampilkan huruf pada list dengan jumlahnya.

- Algoritma:

1. Input kalimat dimasukkan ke dalam variable string, variable tersebut akan dijadikan sebagai list (line 2) dengan elemen berupa karakter dari input string.
2. Penghapusan elemen yang berisi spasi (line 3 – line 5)
3. Mengecilkan semua abjad yang ada di dalam string walaupun abjad itu kecil.
4. Elemen dalam list akan di sort (line 8)
5. Membuat list baru yang bernama c1. List tersebut memiliki 26 elemen dan semuanya memiliki value angka 0.
6. Pada line 11, program akan melaksanakan perulangan untuk menghitung jumlah value karakter di dalam string. Caranya yaitu nomor ASCII value index dikurangi dengan

nomor ASCII karakter 'a', hasilnya dijadikan index bagi list c1, lalu elemen list c1 akan ditambah 1. Perulangan akan berhenti sampai jumlah elemen dari string. Selanjutnya akan melaksanakan perulangan untuk menghapus elemen yang value nya adalah angka 0.

7. List string akan diconvert sebagai set string untuk menghilangkan value yang sama dalam list tersebut. Hasilnya memiliki length yang sama dengan length c1.
  8. Setelah dijadikan set, maka program akan menjadikannya sebagai list lagi (line 17-18).
  9. String sama c1 memiliki kaitan karena c1 adalah jumlah value string dan hasil c1 terurut sesuai dengan urutan string sehingga boleh digabungkan.
- Kode program

```
string = raw_input('masukkan kata: ')
string = list(string)
for i in string[:]:
    if i == ' ':
        string.remove(i)
for n in range(len(string)):
    string[n]= string[n].lower()
string.sort()

c1 = [0]*26
for n in range(len(string)):
    pos = ord(string[i]) - ord('a')
```

```
c1[pos] = c1[pos]+1  
for i in c1[:]:  
    if i==0:  
        c1.remove(i)  
set_string = set(string)  
new_string = list(set_string)  
for i in range(len(new_string)):  
    print '(' , new_string[i] , ',' , c1[i] , ')'
```

### **Kunci Jawaban Bab.3**

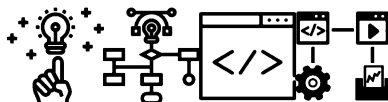
#### **Latihan**



1. Mencoba program dan membuat ringkasan pada link berikut:

<https://www.programiz.com/python-programming/set>

- Set pada python merupakan struktur data yang dapat diartikan himpunan. Dalam set terdapat operasi yang memungkinkan untuk mengakses dua buah kumpulan data atau lebih. Tujuannya adalah mengakses elemen yang berkaitan antar himpunan-himpunan tersebut.
- Elemen pada set dapat berisi gabungan dari beberapa tipe data.
- Elemen pada set tidak boleh berupa elemen yang mutable atau dapat dirubah seperti list.
- Kita dapat merubah list menjadi sebuah set.
- Operasi union berfungsi menggabungkan dua buah set.
- Operasi intersection berfungsi mengakses elemen yang sama pada dua buah set.
- Operasi difference pada setA dan setB berfungsi untuk mengakses elemen pada setA yang tidak ada pada setB
- Operasi symmetric difference berfungsi untuk mengakses elemen yang berbeda pada dua buah set.



## 2. Membuat program kamus sederhana.

- Pre:

Deklarasi dictionary berisi elemen dengan key berupa bahasa Indonesia dan value berupa terjemahan dari kata tersebut.

- Post:

Output hasil terjemahan dari input.

- Deskripsi:

Program menggunakan dictionary yang berisi beberapa elemen dengan key berupa bahasa Indonesia dan value berupa terjemahan dari kata tersebut.

- Solusi:

Membuat dictionary yang berisi beberapa elemen dengan key berupa bahasa Indonesia dan value berupa terjemahan dari kata tersebut. Kemudian inputan string dari user akan dicocokkan dengan key pada dictionary dan akan ditampilkan value dari key tersebut yang merupakan terjemahan kata.

- Algoritma:

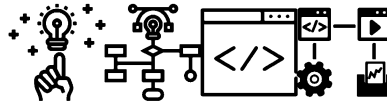
1. Deklarasi dictionary berisi 10 elemen dengan key berupa bahasa Indonesia dan value berupa terjemahan dari kata tersebut.
2. Input string dari user.
3. Menggunakan perulangan, akan dicocokkan inputan dengan key yang ada pada dictionary.
4. Jika ditemukan maka output akan menampilkan value dari key yang dicocokkan.



- Kode program

```
kamus = {'alpukat' : 'avocado', 'pepaya' :  
        'papaya', 'naga' : 'dragon fruit', 'pisang' :  
        'banana', 'apel' : 'apple', 'jeruk' : 'orange',  
        'semangka' : 'watermelon', 'strawberry' :  
        'strawberry', 'nanas' : 'pinapple', 'mangga' :  
        'mango'}  
print '=' * 20  
print 'Menu kata : '  
print '=' * 20  
for i in kamus.keys() :  
    print i  
print '=' * 20  
while 1 :  
    kata = raw_input('Masukkan kata yang sudah  
tersedia untuk ditranslate : ' )  
    if kata in kamus :  
        print kata, ': ', kamus[kata]  
        break  
    else :  
        print 'Inputan tidak sesuai dengan  
menu'
```

## Studi Kasus



1. Program yang dapat menampilkan; (1) semua matakuliah yang diambil ketiga mahasiswa, (2) matakuliah yang sama dari ketiga mahasiswa, (3) matakuliah yang tidak diambil dari dua mahasiswa, (4) matakuliah yang sama-sama diambil oleh dua mahasiswa.
  - Pre:
 

Deklarasi set andi, budi, dan sinta berisi masing-masing matakuliah setiap mahasiswa.
  - Post:
 

Output berupa; (1) semua matakuliah yang diambil ketiga mahasiswa, (2) matakuliah yang sama dari ketiga mahasiswa, (3) matakuliah yang tidak diambil dari dua mahasiswa, (4) matakuliah yang sama-sama diambil oleh dua mahasiswa.
  - Deskripsi :
 

Program menggunakan set dan memanfaatkan operasi dasar dalam set.
  - Solusi:
 

Deklarasi set andi, budi, dan sinta berisi masing-masing matakuliah setiap mahasiswa. Kemudian menggunakan fungsi operasi dalam set seperti union, intersection, dan difference.
  - Algoritma:

1. Deklarasi set andi, budi, dan sinta dengan elemen matakuliah yang mereka ambil.
  2. Menggunakan operasi union dari ketiga mahasiswa.
  3. Menggunakan operasi intersection dari Andi dengan hasil intersection Budi dan Sinta.
  4. Menggunakan operasi difference Budi dengan Sinta.
  5. Menggunakan operasi intersection dari Andi dan Budi.
- Kode program

```

Andi = set(['BD', 'PBO', 'KDJK', 'PW'])
Budi = set(['BD', 'KDJK'])
Sinta = set(['SD', 'TI'])

print 'Menampilkan matakuliah yang diambil oleh
      mahasiswa: '
for i in Andi.union(Budi.union(Sinta)):
    print i, '',
print '\n'
print 'Menampilkan matakuliah yang sama antara
      mereka
      bertiga: '
for i in
(Aнди.intersection(Budi.intersection(Sinta))):
    print i, '',
print '\n'

print 'Menampilkan matakuliah yang tidak diambil
      antara
      dua mahasiswa (kombinasi mahasiswa bebas): '
for i in Budi.difference(Sinta):
    print i, '',
print '\n'

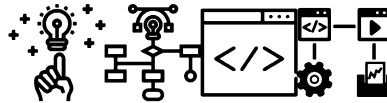
print 'Menampilkan matakuliah yang sama-sama diambil

```

```

    antara dua mahasiswa (kombinasi mahasiswa
    bebas):' for i in Andi.intersection(Budi):
        print i, '', print '\n'

```



2. Program yang dapat menampilkan huruf dan jumlahnya menggunakan dictionary.

- Pre:

Input kata dari user.

- Post:

Output berupa huruf pada kata bersama jumlahnya.

- Deskripsi:

Program berfungsi menyimpan data pada sebuah dictionary. Key merupakan huruf pada kata tersebut dan valu merupakan jumlah pada kata tersebut.

- Solusi:

Deklarasi dictionary kosong yang akan diisi kata dari inputan user. Menggunakan perulangan, setiap karakter akan dihitung dan dimasukkan sebagai value pada dictionary.

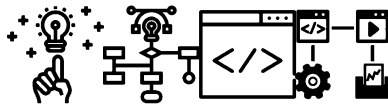
- Algoritma:

1. Deklarasi dictionary kosong.
2. Input kata dari user.
3. Menggunakan perulangan setiap karakter akan dibandingkan dengan kata di dictionary.
4. Jika karakter pada dictionary bernilai false maka value data akan bernilai 1.

5. Jika karakter sudah ada pada dictionary, maka value karakter tersebut akan ditambahkan 1.
6. Tampilkan hasil berupa elemen pada dictionary berupa key dan valuenya.

- Kode program

```
kata = raw_input("Masukan kata :")
data = {}
for x in kata:
    if (x in data) == False:
        data[x] = 1
    else:
        data[x] = data[x]+1
for a in range(len(data)):
    print data.popitem()
```



3. Program yang dapat menampilkan kategori hewan berdasarkan jenis kaki dan makanannya.

- Pre:

Deklarasi set omnivora, herbivora, karnivora, kaki4, dan kaki2.

- Post:

Output berupa hewan berdasarkan golongan jumlah kaki dan jenis makanannya.

- Deskripsi:

Program yang berisi data hewan berdasarkan jumlah.kaki dan jenis makanannya. Output merupakan beberapa kombinasi dari jumlah kaki dan jenis makanan hewan tersebut

- Solusi:

Deklarasi set omnivora, herbivora, karnivora, kaki4, dan kaki2. Elemen pada set tersebut berisi nama hewan yang sesuai dengan golongannya. Output beberapa kombinasi dari jumlah kaki dan jenis makanan hewan didapatkan menggunakan operasi dasar dari set

- Algoritma:

1. Deklarasi set omnivora, herbivora, karnivora, kaki4, dan kaki2 beserta elemennya.
2. Input kata dari user.
3. Tampilkan elemen set omnivora.
4. Tampilkan elemen set karnivora union dengan kaki2.
5. Tampilkan elemen set karnivora union dengan kaki4.
6. Tampilkan elemen set herbivora union dengan kaki2.
7. Tampilkan elemen set herbivora union dengan kaki4.
8. Tampilkan elemen set omnivora union dengan kaki2.
9. Tampilkan elemen set omnivora union dengan kaki4.

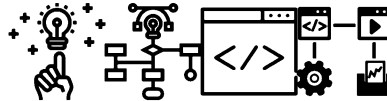
- Kode program

```
omnivora = {'beruang', 'ayam'}
herbivora = {'monyet', 'sapi'}
karnivora = {'singa', 'elang'}
kaki4 = {'beruang', 'sapi', 'singa'}
kaki2 = {'ayam', 'monyet', 'elang'}
print "Hewan omnivora : ",
for a in omnivora :
    print a,
print "\nHewan karnivora berkaki 2 : ",
for b in karnivora & kaki2 :
    print b,
print "\nHewan karnivora berkaki 4 : ",
for c in karnivora & kaki4 :
    print c,
print "\nHewan herbivora berkaki 2 : ",
for d in herbivora & kaki2 :
    print d,
print "\nHewan herbivora berkaki 4 : ",
for e in herbivora & kaki4 :
    print e,
print "\nHewan omnivora berkaki 2 : ",
for f in omnivora & kaki2 :
    print f,
print "\nHewan omnivora berkaki 4 : ",
for g in omnivora & kaki4 :
    print g,
```

## **Kunci Jawaban Bab.4**

### **Latihan**

1. Penjelasan pada modul.



2. Program yang dapat membalik kata.

- Pre:  
Input kata dari user.
- Post:  
Output berupa kebalikan kata dari input user.
- Deskripsi:  
Program yang berfungsi membalik kata dari input user dengan menggunakan konsep stack.
- Solusi:  
Inisialisasi fungsi dasar stack. Kemudian buat fungsi bernama reverse, dengan perulangan pada setiap karakter kata akan di push pada stack. Setelah karakter sudah di push seluruhnya, maka akan dilakukan operasi pop yang akan mengambil dan menampilkan data pada stack dari posisi atas. Hal tersebut menghasilkan output kata yang terbalik.
- Algoritma:
  - a. Inisialisasi fungsi dasar stack.
  - b. Buat fungsi reverse, menggunakan perulangan setiap karakter inputan akan di push pada stack.



- c. Ketika stack tidak kosong, maka data akan di pop dari posisi atas.
- d. Tampilkan output dari kata yang telah dibalik

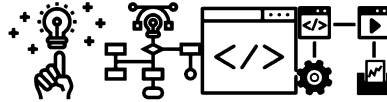
- Kode program

```
class Stack:
    def __init__(self):
        self.items = []
    def push(self, item):
        self.items.append(item)
    def pop(self):
        return self.items.pop()
    def peek(self):
        return self.items[0]
    def isEmpty(self):
        return len(self.items) == 0

def reverse(text):
    s = Stack()
    for c in text:
        s.push(c)
    revText = ""
    while not s.isEmpty():
        revText += s.pop()
    return revText

text = raw_input("masukkan teks : ")
print(reverse(text))
```

## Studi Kasus



1. Program yang dapat mengubah bilangan desimal menjadi biner.
  - Pre:
 

Input angka dari user.
  - Post:
 

Output berupa hasil konversi decimal menjadi biner.
  - Deskripsi:
 

Program yang berfungsi mengkonversi input bilangan decimal menjadi biner dengan menggunakan konsep stack.
  - Solusi:
 

Inisialisasi fungsi dasar stack. Kemudian buat fungsi bernama `divideBy2`, dengan perulangan, jika angka lebih dari 0 maka akan dilakukan operasi modulus 2 pada angka tersebut. Sisa bagi akan disimpan (push) pada stack. Setelah angka sudah tidak bisa dibagi, selanjutnya data sisa bagi pada stack akan diambil (pop) dan ditampilkan menjadi output nilai biner.
  - Algoritma:
    - a. Inisialisasi fungsi dasar stack.
    - b. Buat fungsi `divideBy2`, menggunakan perulangan angka akan dilakukan operasi modulus.
    - c. Hasil dari modulus disimpan (push) pada stack.

- d. Setelah angka sudah tidak bisa dibagi, maka data sisa bagi pada stack diambil (pop)
- e. Tampilkan output nilai biner
- Kode program

```
class Stack:
    def __init__(self):
        self.items = []
    def isEmpty(self):
        return self.items == []
    def push(self, item):
        self.items.append(item)
    def pop(self):
        return self.items.pop()
    def peek(self):
        return self.items[len(self.items)-1]
    def size(self):
        return len(self.items)
def divideBy2(decNumber):
    remstack = Stack()

    while decNumber > 0:
        rem = decNumber % 2
        remstack.push(rem)
        decNumber = decNumber // 2

    binString = ""
    while not remstack.isEmpty():
        binString = binString +
str(remstack.pop())

    return binString

decNumber = input("masukkan angka :")
print(divideBy2(decNumber))
```

## **Kunci Jawaban Bab.5**

### **Latihan**

1. Perbedaan single linked list (SLL), double linked list (DLL), circular linked list (CLL).
  - SLL : mempunyai satu pointer yang mengarah ke “next”.
  - DLL : mempunyai dua pointer yang mengarah ke “previous” dan “next”.
  - CLL : pada node terakhir, pointer akan mengarah ke head node.
2. Penjelasan pada modul
3. Deklarasi class node pada DLL dan CLL

- DLL

```
class Node(object):

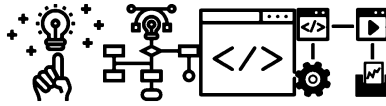
    def __init__(self, data, prev, next):
        self.data = data
        self.prev = prev
        self.next = next
```

- CLL

```
class Link:
    def __init__(self, data, next):
        self.data = data
        self.next = next

class CircularLinkedList:
    def __init__(self):
        self.head = Link(None, None)
        self.head.next = self.head
```

## Studi Kasus



### 1. Program antrian pasien.

- Pre:  
Input user.
- Post:  
Output berupa nomor antrian, jumlah antrian, dan estimasi waktu tunggu.
- Deskripsi:  
Program yang berfungsi mencatat antrian pasien. Dalam program memiliki beberapa fitur di antaranya; (1) mengetahui nomor antrian, (2) mengetahui jumlah antrian yang ada, (3) mengetahui estimasi waktu tunggu.
- Solusi:  
Dengan menggunakan konsep queue dalam membuat program antrian. Pada program terdapat beberapa fitur tambahan yaitu mengetahui jumlah antrian menggunakan fungsi `size`, dan mengetahui estimasi waktu tunggu dengan mengalikan jumlah pasien yang ada dengan 10 menit.
- Algoritma:
  - a. Inisialisasi fungsi dasar queue.
  - b. Buat fungsi utama, dengan tiga menu utama
  - c. Menu daftar antrian, akan menambahkan data (`enqueue`) pada queue dengan tambahan informasi

estimasi waktu yang didapat dari jumlah pasien yang ada dikalikan dengan 10 menit.

- d. Menu panggil antrian, akan menghapus data (dequeue) pada queue.
- e. Menu jumlah antrian, menggunakan fungsi size.

- Kode program

```
import os
class Node(object):
    def __init__(self, item):
        self.data = item
        self.next = None
class Queue(object):
    def __init__(self):
        self.length = 0
        self.head = None
        self.current = None
    def enqueue(self, x):
        newNode = Node(x)

        if self.head == None:
            self.head = newNode
            self.current = newNode
        else:
            self.current.next = newNode
            self.current = newNode
        self.length += 1
    def dequeue(self):
        if self.length == 0:
            return 0
        else:
            item = self.head
            self.head = self.head.next
            self.length -= 1
            return item.data
    def size(self):
```

```

        return self.length
def main():
    os.system('cls')
    Antrian = Queue()
    i = 0
    while True:
        print
        "=====
        "
        print "menu:"
        print "1. Daftar Antrian "
        print "2. Panggil Antrian "
        print "3. Jumlah Antrian"
        print "4. Exit"
        print
        "=====
        "

        p = int(raw_input("Select menu: "))
        os.system('cls')
        if p == 1:
            i +=1
            Antri = 'N' + str(i)
            print "Nomor antrian anda: ",
Antri, " Harap
                menunggu sekitar",
                Antrian.size()*10,
                "menit"
            Antrian.enqueue(Antri)
        elif p == 2:
            panggilan = Antrian.dequeue()
            if panggilan == 0:
                print "Tidak ada yang mengantri
untuk
                berobat"
            else:
                print "Nomor antrian: ",
panggilan,
                "Silahkan menemui dokter
                untuk berobat"

```

```
        elif p == 3:
            print "Jumlah antrian: ",
Antrian.size()
        else:
            s = raw_input("Konfirmasi untuk
exit (Y/N)? ")
            if s.lower() == 'y':
                break
            else:
                continue
main()
```

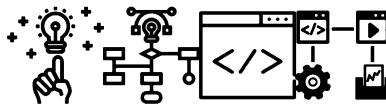


## **Kunci Jawaban Bab.6**

### **Latihan**

1. Penjelasan algoritma insert, remove, dan search pada binary search tree (BST) di visualgo.net.
  - **Insert** : jika nilai kurang dari node pindah kekiri node, jika tidak pindah kekanan. Setelah proses perbandingan, jika menemukan node kosong maka buat node baru
  - **Remove** : cari nilai yang akan dihapus, jika nilai tersebut adalah leaf maka hapus leaf. Jika nilai tersebut memiliki 1 child, maka potong jalur dan child akan menggantikan posisi nilai yang dihapus. Jika nilai tersebut memiliki 2 child, maka gantikan dengan successor (node sebelah kanan dari nilai yang dihapus).
  - **Search** : jika nilai == root, return value. Jika nilai lebih besar dari root, maka kunjungi node kanan. Jika lebih kecil, kunjungi node sebelah kiri. Ulangi fungsi rekursif sampai menemukan nilai yang dicari.

### **Studi Kasus**



1. Program Binary Search Tree (BST).
  - Pre:  
Input user.
  - Post:  
Output berupa elemen pada tree dan traversal tree.

- Deskripsi:

Program yang berfungsi menambah, menghapus, dan mencari elemen pada tree. Selain itu juga menampilkan traversal tree secara inorder, post-order, dan pre-order.

- Algoritma:

- a. Inisialisasi fungsi dasar tree.
- b. Tambahkan fungsi inorder.
- c. Tambahkan fungsi pre-order.
- d. Tambahkan fungsi post-order.

- Kode program

```
class Node:
    def __init__(self, val):
        self.l = None
        self.r = None
        self.v = val

class Tree:
    def __init__(self):
        self.root = None

    def getRoot(self):
        return self.root

    def add(self, val):
        if(self.root == None):
            self.root = Node(val)
        else:
            self._add(val, self.root)

    def _add(self, val, node):
        if(val < node.v):
            if(node.l != None):
                self._add(val, node.l)
            else:
```

```

        node.l = Node(val)
    else:
        if(node.r != None):
            self._add(val, node.r)
        else:
            node.r = Node(val)

    def find(self, val):
        if(self.root != None):
            return self._find(val, self.root)
        else:
            return None

    def _find(self, val, node):
        if(val == node.v):
            return node
        elif(val < node.v and node.l != None):
            self._find(val, node.l)
        elif(val > node.v and node.r != None):
            self._find(val, node.r)

    def deleteTree(self):
        # garbage collector will do this for
us.
        self.root = None

    def printTree(self, c):
        if(self.root != None):
            if c == 2:
                self._printTreeIn(self.root)
            elif c == 3:
                self._printTreePre(self.root)
            elif c == 4:
                self._printTreePost(self.root)

    def _printTreeIn(self, node):
        if(node != None):
            self._printTreeIn(node.l)
            print str(node.v) + ' '
            self._printTreeIn(node.r)
    def _printTreePre(self, node):

```

```

        if (node != None):
            print str(node.v) + ' '
            self._printTreePre(node.l)
            self._printTreePre(node.r)
    def _printTreePost(self, node):
        if (node != None):
            self._printTreePost(node.l)
            self._printTreePost(node.r)
            print str(node.v) + ' '

tree = Tree()
while 1:
    print
    "=====\\n" \
    "1. Add Value\\n" \
    "2. Print Tree in InOrder \\n" \
    "3. Print Tree in Pre-Order \\n" \
    "4. Print Tree in Post-Order \\n" \
    "5. Search Value\\n" \
    "6. Exit \\n" \
    "=====\\n"

    c = int(raw_input('Pilih menu: '))
    if c == 1:
        print "Add value: "
        tree.add(int(raw_input()))
    elif c == 2:
        print "Tree in InOrder:"
        tree._printTreeIn(c),
    elif c == 3:
        print "Tree in Pre-Order:"
        tree._printTreePre(c),
    elif c == 4:
        print "Tree in Post-Order:"
        tree._printTreePost(c),
    elif c == 5:
        print ""
        tree.find(c),
    else:
        break

```





Penerbit:

**Ahlimedia Press (Anggota IKAPI)**

Jl. Ki Ageng Gribig, Gang Kaserin MU No. 36

Kota Malang 65138, Telp: +628523277747

[www.ahlimediapress.com](http://www.ahlimediapress.com)

ISBN 978-623-6351-35-2

