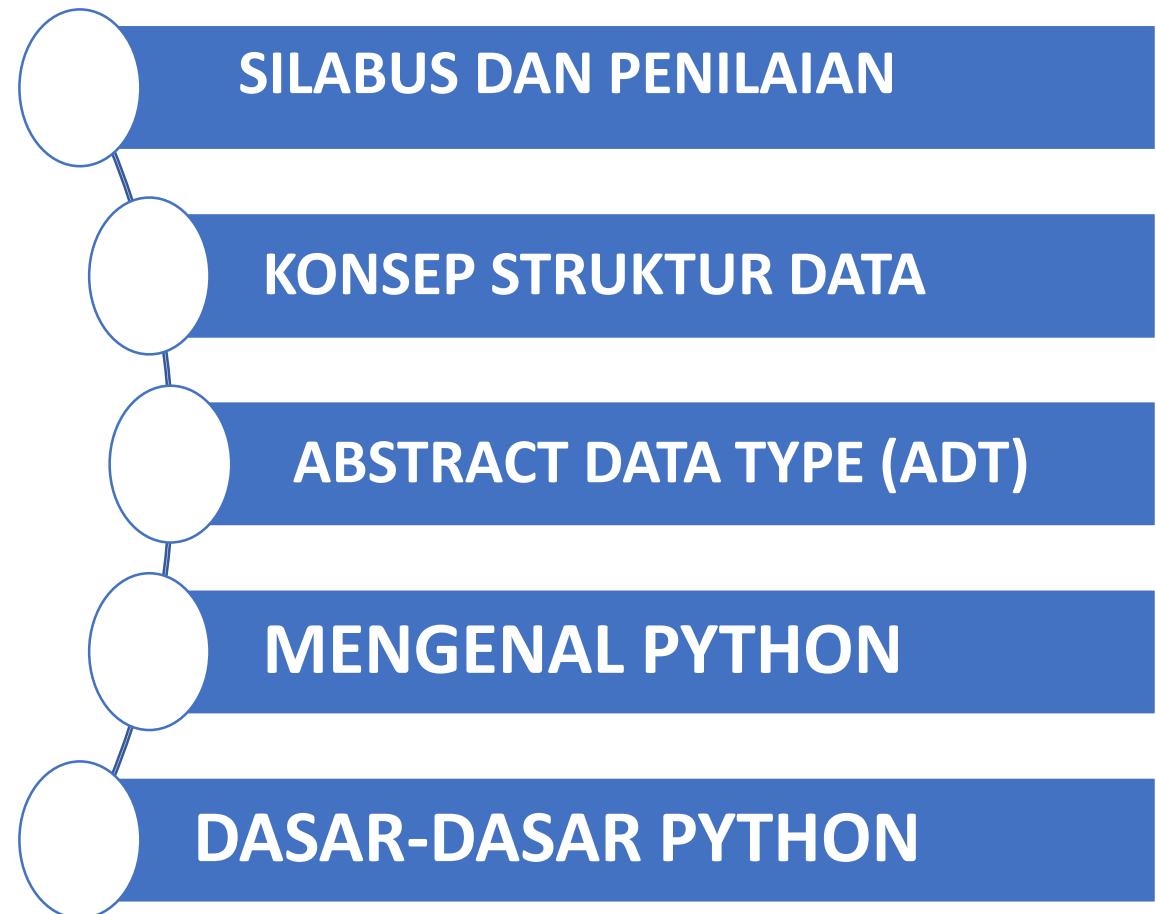




STRUKTUR DATA (PYTHON) “PENDAHULUAN”

[@SUARGA | [Pertemuan 01]

OutLine



SILABUS



- 1. Pendahuluan & Mengenal Python**
- 2. Elemen Dasar Bahasa Python**
- 3. Instruksi Utama Bahasa Python**
- 4. Struktur Larik (Array)**
- 5. Struktur data internal Python**
- 6. Struktur Tumpukan (Stack)**
- 7. Struktur Antrian (Queue)**
- 8. UTS**
- 9. Struktur Untaian Tunggal (Singly Linked List)**
- 10. Struktur Untaian Melingkar (Circular Linked List)**
- 11. Struktur Untaian Ganda (Doubly Linked List)**
- 12. Struktur Antrian Prioritas**
- 13. Struktur Pohon Biner (Binary Tree)**
- 14. Struktur Pohon Seimbang (Balanced Tree)**
- 15. Struktur Tabel Hash (Hashed Table)**
- 16. UAS**

Penilaian

1. **Kehadiran:** harus hadir minimal 12 kali ==> 10%
2. Memasukkan **Tugas-tugas** : 20%
3. Ikut **MID TEST (UTS)**: Maksimum 35%
4. Ikut **FINAL TEST (UAS)** : Maksimum 35%

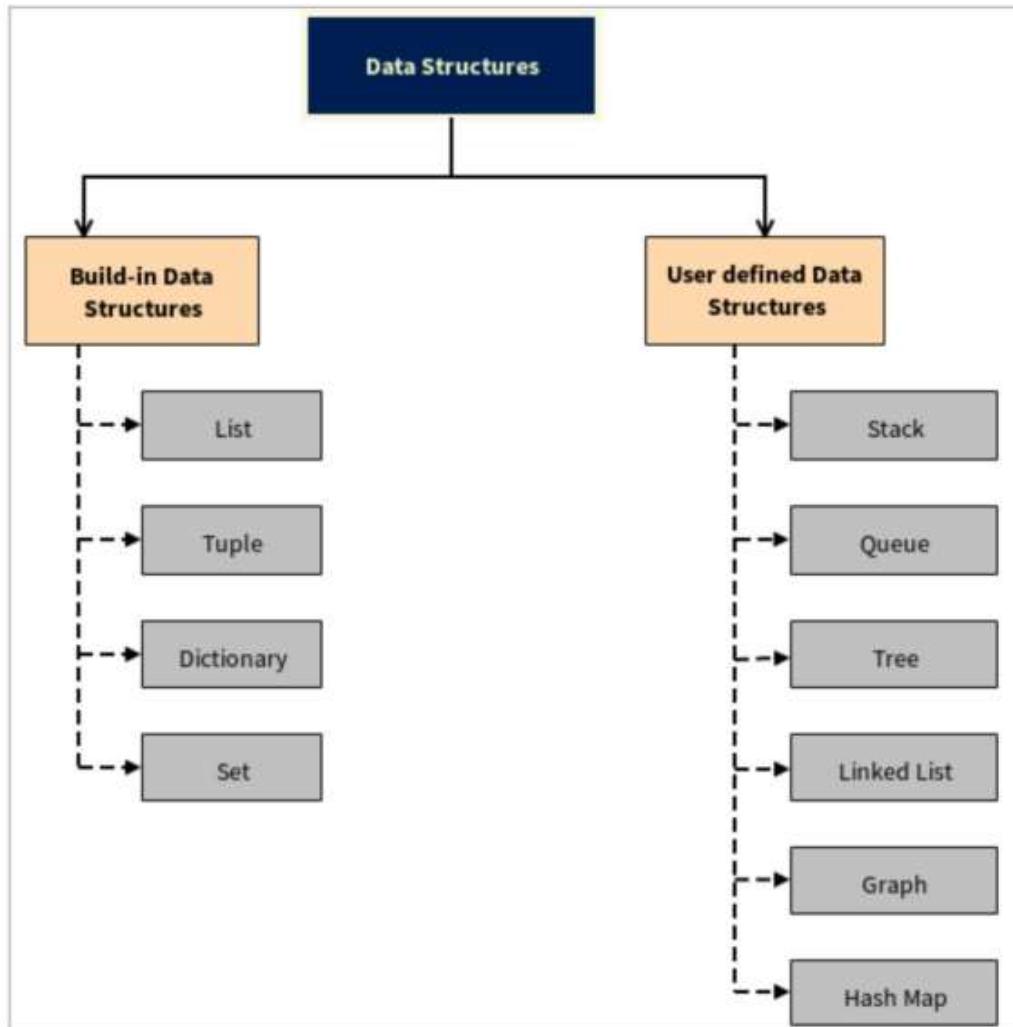
NILAI AKHIR

A	:	86	-	100
A-	:	81	-	85
B+	:	76	-	80
B	:	71	-	75
B-	:	66	-	70
C+	:	61	-	65
C	:	51	-	60
D	:	41	-	50
E	:	0	-	40

Konsep Struktur Data

- Struktur data dapat didefinisikan sebagai kelas/objek, atau abstraksi tipe data (ADT) yang memiliki **tiga unsur dasar**, yaitu:
- **Seperangkat fungsi/method**: dimana setiap fungsi melakukan suatu jenis operasi terhadap nilai yang tersimpan didalam struktur data.
- **Struktur Penyimpanan (Storage)** : representasi suatu struktur abstrak memory yang dipakai menyimpan data / nilai yang dimasukkan, sehingga membentuk suatu organisasi data.
- **Algoritma** : dalam setiap fungsi terdapat algoritma yang menjelaskan langkah yang harus dilakukan terhadap nilai data dalam struktur penyimpanan-nya sehingga diperoleh hasil sesuai dengan tujuan dari fungsi tersebut.

Jenis Struktur Data



Built-in Python Data Structure:

- **List**
- **Tuple**
- **Dictionary**
- **Set**

User-defined Data Structures

- Larik (Array)
- Tumpukan (Stack)
- Antrian (Queue)
- Untaian (Linked List)
- Pohon (Tree)
- Tabel Hash
- Heap
- Graph

Strong & Weakness

Struktur Data	Kekuatan	Kelemahan
Larik / Array	Mudah menyisip data, akses data cepat bila indeks diketahui. Menyimpan data sesuai indeks	Bila indeks tidak diketahui maka mencari, menghapus data menjadi lambat. Ukuran (size) tetap
Stack (Tumpukan)	Akses LIFO, cepat bila data berada diposisi top	Akses LIFO cepat, yang berada diposisi top

Antrian (Queue)	Akses FIFO cepat, yang berada di posisi depan	Akses item yang lain lambat
Untaian (Linked List)	Insert cepat, delete cepat, di posisi head atau tail	Lambat dalam pencarian item yang berada di posisi lain
Pohon Biner (Binary Tree)	Mencari data cepat, insert cepat, delete cepat	Algoritma delete rumit, bergantung pada posisi item
Pohon Seimbang (AVL Tree)	Search, Insert, Delete cepat	Delete item rumit, struktur AVL harus dipertahankan.
Hash Table	Sangat cepat akses bila kunci diketahui, insert cepat	Lambat dalam delete, akses lambat bila kunci tidak diketahui, pemakaian memory kurang efisien
Heap	Insert, delete bisa cepat	Akses lambat pada item lain
Graph	Model dari situasi nyata	Algoritma bisa lambat dan rumit

Jenis Algoritma

- Struktur data pada umumnya memerlukan algoritma/program sebagai implementasi dari fungsi struktur data, antara lain:
 1. menciptakan (**init**) struktur data
 2. menyisipkan (**insert/add/append**) data baru
 3. mencari (**search**) data yang ada
 4. menghitung elemen-nya (**len atau size**)
 5. menghapus (**delete/remove**) data
 6. mengurutkan (**sort**) data, atau order data

ADT (Abstract Data Type)

- Pada ilmu struktur data juga telah ditemukan cara untuk memecahkan hal struktur data yang sulit, yaitu dengan abstraksi, sehingga dikenal istilah *Abstract Data Type* (ADT), dimana dilakukan pemisahan antara properti struktur data dan implementasi struktur data.
- Pengguna (users) struktur data hanya memerlukan properti data tanpa harus mengetahui bagaimana di-implementasi-kan.
- Programmer yang menyusun ADT ini yang akan pusing memikirkan bagaimana implementasi-nya.

User programs interact with ADTs through their interface or set of operations.

User Program



The implementation details are hidden as if inside a black box.

ADT memisahkan sifat objek dengan implementasi-nya

ADT

- Kategori dari sifat/properti atau fungsi dalam ADT dapat di-kategorikan menjadi empat bagian/macam yaitu:
- **constructor** : menciptakan dan inisialisasi objek baru
- **accessor** : melihat isi atau akses data/objek tanpa mengubahnya
- **mutator** : mengubah isi atau nilai data field dari objek ADT
- **iterator** : mengolah elemen data secara ber-urutan secara individu

Contoh ADT

- **Date** adalah **kelas/objek** yang mewakili suatu hari dalam kalender Gregorian, dimana hari pertama dianggap terjadi pada 24-Nopember-4713 SM (sebelum masehi). Properti/fungsi dari Date (ADT) adalah sebagai berikut:
- **Date (month, day, year)** : menciptakan satu objek date yang sah menurut penanggalan Gregorian, dimana tanggal sebelum masehi dinyatakan dengan memiliki komponen year negatif
- **day()** : fungsi untuk menyatakan angka Gregorian untuk hari tersebut
- **month()** : fungsi untuk menyatakan angka Gregorian untuk bulan

- **year()** : fungsi untuk menyatakan angka Gregorian untuk tahun
- **monthName()** : menyatakan nama dari bulan kalender Gregorian
- **dayOfWeek()** : menyatakan kode nama hari, 0 = senin – 6 = minggu
- **numDays(otherDate)** : menyatakan jumlah hari antara hari ini dengan tanggal otherDate
- **isLeafYear()** : memeriksa apakah tahun ini kabisat atau bukan, True bila kabisat, False bila bukan kabisat.
- **advanceBy(days)** : memajukan tanggal sejumlah hari days yang diberikan
- **comparable(otherDate)** : membandingkan hari ini dengan otherDate
- **toString()** : mengubah tanggal menjadi format string bulan/hari/tahun

- Ketika user/pengguna ADT mau menggunakan method atau fungsi, maka yang pertama adalah melakukan inisialisasi (constructor), misalnya:
- **tanggal = Date(1, 31, 2014)**
- kemudian fungsi/method dari ADT (accessor) dapat digunakan sebagai berikut:
 - `tanggal.monthName()` # memberikan nama bulan, misal Januari
 - `tanggal.dayOfWeek()` # memberikan kode hari, misal Jum'at
 - `tanggal.year()` # memberikan tahun, misal 2014
- dan sebagai-nya. Perhatikan bagimana fungsi dari suatu objek digunakan, yaitu fungsi dinyatakan setelah objek *tanggal* yang di-ikuti titik.

Mengenal Python

- Python adalah bahasa program yang di-klaim oleh penciptanya sebagai bahasa yang “simple” dan “powerful”, mudah dipelajari, memiliki struktur data yang efisien, serta berorientasi objek.
- Pencipta Python adalah *Guido van Rossum* yang menamakan bahasa ini berdasarkan nama suatu film serial TV berjudul “Monty Python’s Flying Circus”,
- jadi nama ini bukan dari “Ular Python”, bahkan Guido kabarnya tidak menyukai ular python. Namun simbol yang digunakan oleh para pengguna python adalah simbol “ular”.

Guido van Rossum (born 31 January 1956) is a [Dutch computer programmer](#) who is best known as the author of the [Python programming language](#). In the Python community, Van Rossum is known as a "[Benevolent Dictator For Life](#)" (BDFL), meaning that he continues to oversee the Python development process, making decisions where necessary. He was employed by [Google](#) from 2005 until December 7th 2012, where he spent half his time developing the Python language. In January 2013, Van Rossum started working for [Dropbox](#). [\[Wikipedia\]](#)



Simbol Python (www.python.org)

Menurut Guido, program yang ditulis dalam Python lebih singkat dari program C/C++ maupun Java antara lain karena:

- Python memiliki tipe data tingkat tinggi sehingga suatu operasi yang rumit dapat dilaksanakan dalam satu statement saja;
- Tanda indicator group seperti begin/end pada Pascal, atau { } pada C++ dan Java, tidak perlu pada Python, digantikan hanya oleh “indentation”;
- Variable atau argument tidak perlu di-definisikan terlebih dahulu sebelum digunakan dengan kata lain tidak perlu ada deklarasi variabel;

- Secara singkat Python memiliki kelebihan berikut ini:
 - Simple, tidak rumit, cukup sederhana
 - Easy to learn, mudah dipelajari
 - Free and Open Source, gratis bersifat open source
 - High-level language, bahasa tingkat tinggi
 - Portable / multi-platform, mudah di-install di berbagai platform
 - Interpreted, dapat di-kompilasi dan juga di-interpret
 - Object-oriented, berorientasi objek
 - Extensible (bisa memanggil prosedur yang ditulis dalam C/C++)
 - Embeddable, dapat di-tambahkan ke perangkat lain
 - Extensive Libraries, pustaka fungsi-nya sangat besar

Instalasi Python

- Instalasi Python dapat di-download dari situs: <http://www.python.org>, yang penting diperhatikan adalah sistem operasi dari komputer dimana Python akan di-install. Karena file installer berbeda untuk sistem operasi Windows, OS X, atau versi Linux yang digunakan.
- Hal selanjutnya adalah versi dari Python, karena versi 2 tidak sesuai dengan versi 3 maka anda harus memilih dengan tepat versi yang akan digunakan, tentu saja versi terbaru biasanya lebih lengkap dari versi lama. File installer untuk komputer dengan sistem operasi Windows, yang digunakan dalam kuliah ini, nama file-nya adalah: **python-3.10.6.msi**.

A screenshot of a web browser window. The address bar shows 'python.org/downloads/'. The page content displays a table of Python versions with their maintenance status, first release date, end of support date, and PEP number.

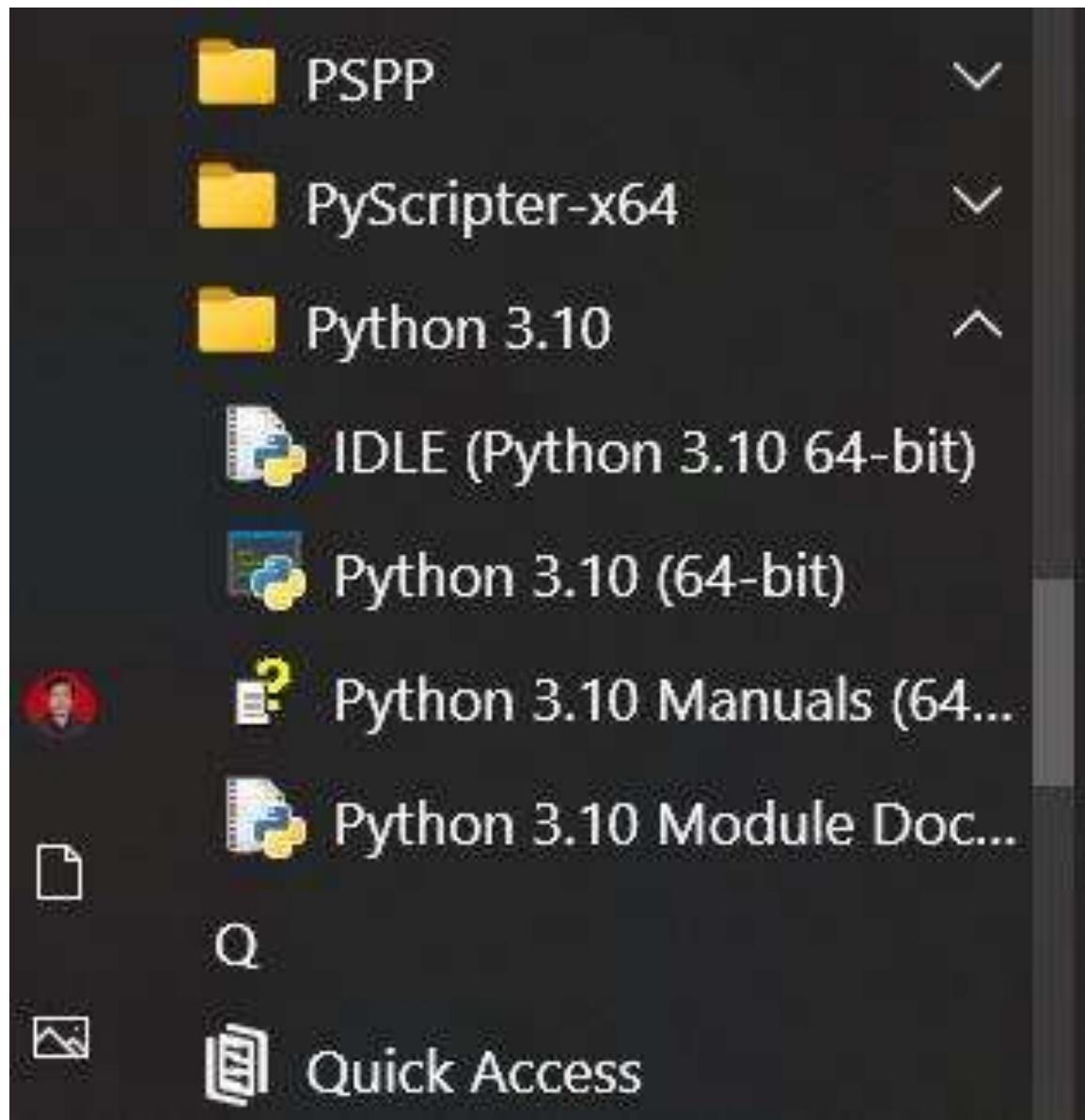
Python version	Maintenance status	First released	End of support	schedule
3.10	bugfix	2021-10-04	2026-10	PEP 619
3.9	security	2020-10-05	2025-10	PEP 596
3.8	security	2019-10-14	2024-10	PEP 569
3.7	security	2018-06-27	2023-06-27	PEP 537
2.7	end-of-life	2010-07-03	2020-01-01	PEP 373

Looking for a specific release?

Python releases by version number:

Release version	Release date	Click for more	
Python 3.10.6	Aug. 2, 2022	Download	Release Notes
Python 3.10.5	June 6, 2022	Download	Release Notes
Python 3.9.13	May 17, 2022	Download	Release Notes
Python 3.10.4	March 24, 2022	Download	Release Notes

- Setelah proses download dari file installer selesai, maka file tersebut bisa langsung di-click, anda hanya perlu meng-klik tombol [Next], hingga muncul-nya tombol [Finish], ketika tombol Finish di-click berarti instalasi Python telah selesai. Setelah selesai maka sistem Python bisa dicoba digunakan.
- Pada OS Windows, klik tombol Start, kemudian klik “All Programs” dan cari folder “Python3.10”, ketika diklik, muncul pilihan, pilihlah IDLE(Python GUI), klik, dan muncul window yang disebut “Python Shell”, tempat dimana instruksi bisa dimasukkan. Python Shell adalah “interpreter” dari Python, instruksi akan langsung di-laksanakan ketika tombol [Enter] ditekan setelah instruksi.



```
Python 3.10.5 (tags/v3.10.5:f377153, Jun  6 2022, 16:14:13) [MSC v.1929 64 bit  
AMD64]) on win32
```

```
Type "help", "copyright", "credits" or "license()" for more information.
```

```
>>>
```

Python Shell, ketik instruksi setelah >>> ...

```
Python 3.10.5 (tags/v3.10.5:f377153, Jun  6 2022, 16:14:13) [MSC v.1929 64 bit  
AMD64] on win32
```

```
Type "help", "copyright", "credits" or "license()" for more information.
```

>>>



The screenshot shows the Python IDLE interface. At the top, there's a menu bar with File, Edit, Shell, Debug, Options, Window, and Help. Below the menu bar, a large text area displays Python version information and help instructions. In the bottom-left corner of this area, the text '>>>' is visible. To the right of this text area is a separate window titled '*untitled*' with its own menu bar (File, Edit, Format, Run, Options, Window, Help). Inside this window, the text '#EDITOR PYTHON ...' is displayed in red, indicating it's a comment or a placeholder. The overall layout is typical of a Python development environment.

editor program python, didepan shell

IDLE Shell 3.10.5

File Edit Shell Debug Options Window Help

Python 3.10.5 (tags/v3.10.5:f377153, Jun 6 2022, 16
AMD64)] on win32

Type "help", "copyright", "credits" or "license()" fo

>>>

D:\contohWhile.py - D:\USER\Python\contohWhile.py (3.10.5)

File Edit Format Run Options Window Help

```
#contohWhile
#menampilkan "Selamat : instruksi while" n kali
print("Masukkan satu angka positif:")
n = int(input())
while (n > 0):
    print("Selamat : instruksi while")
    n = n-1
```

Editor berisi program “contohWhile.py”

```
Python 3.10.5 (tags/v3.10.5:f377153, Jun  6 2022, 16:  
14:13) [MSC v.1929 64 bit (AMD64)] on win32  
Type "help", "copyright", "credits" or "license()" fo  
r more information.
```

```
>>>  
===== RESTART: D:\USER\Python\contohWh  
ile.py =====  
Masukkan satu angka positif:  
Masukkan satu angka positif:5  
Traceback (most recent call last):  
  File "D:\USER\Python\contohWhile.py", line 4, in <m  
odule>  
    n = int(input())  
ValueError: invalid literal for int() with base 10: '  
Masukkan satu angka positif:5'
```

RUN ada Error

```
>>>  
===== RESTART: D:\USER\Python\contohWh  
ile.py =====  
Masukkan satu angka positif:  
5  
Selamat : instruksi while  
Selamat : instruksi while  
Selamat : instruksi while  
Selamat : instruksi while  
Selamat : instruksi while
```

RUN program
berhasil

Beberapa IDE/Editor Python

- Selain Python IDLE, tersedia beberapa editor program Python yang lain, misalnya:
 - a.Thonny
 - b.PyScripter
 - c.Sublime Text 3
 - d.Atom
 - e.Pycharm
 - f.visual Studio Code
 - g.Spyder
 - h.Online Editor: Programiz, onlinegdb

Thonny

The screenshot shows the Thonny Python IDE interface. The top bar displays the title "Thonny - D:\USER\Documents\myLecture\ProgramLanguage\Bahasa_Python_Program\algorith" and a menu bar with File, Edit, View, Run, Tools, Help. Below the menu is a toolbar with icons for file operations and execution. The main area contains a code editor with the file "Depth_First.py" open. The code implements a Depth-First Search (DFS) algorithm. The editor has line numbers on the left and syntax highlighting for Python keywords and comments. Below the code editor is a shell window titled "Shell *". It shows the Python version "Python 3.7.9 (bundled)" and a prompt ">>> |".

```
1 # DFS algorithm in Python
2
3 # DFS algorithm
4 def dfs(graph, start, visited=None):
5     if visited is None:
6         visited = set()
7     visited.add(start)
8
9     print(start)
10
11    for next in graph[start] - visited:
12        dfs(graph, next, visited)
13    return visited
14
15
16 graph = {'0': set(['1', '2']),
17           '1': set(['0', '3', '4']),
18           '2': set(['0']),
19           '3': set(['1']),
20           '4': set(['2', '3'])}
```

Shell *

```
Python 3.7.9 (bundled)
>>> |
```

PyScripter

The screenshot shows the PyScripter IDE interface. The title bar reads "PyScripter - D:\USER\Documents\myLecture\ProgramLanguage\Bahasa_Python_Program\SOAL2_SP.PY". The menu bar includes File, Edit, Search, View, Project, Run, Tools, and Help. The toolbar contains various icons for file operations like Open, Save, Print, and Run. The left sidebar is the "File Explorer" showing the file structure: This PC, 3D Objects, Desktop, Documents, Downloads, Music, Pictures, Videos, Acer (C:), Data (D:), and Data II (E:). The main code editor window displays the following Python script:

```
x = float(X.get())
Y.set(x*x*x + 10*x*x + 5*x + 10)
except ValueError:
    pass

root = Tk()
root.title("MENGHITUNG NILAI Y=F(X)")
mainframe = ttk.Frame(root, padding="3 3 12 12")
mainframe.grid(column=0, row=0, sticky=(N, W, E, S))
mainframe.columnconfigure(0, weight=1)
mainframe.rowconfigure(0, weight=1)

#variable data
X = StringVar()
Y = StringVar()

#komponen widget
x_entry = ttk.Entry(mainframe, width=7, textvariable=X)
x_entry.grid(column=1, row=2, sticky=(W,E))
y_entry = ttk.Entry(mainframe, width=7, textvariable=Y)
y_entry.grid(column=1, row=3, sticky=(W,E))
ttk.Button(mainframe, text="HITUNG", command=HITUNG).grid(c
```

Sublime Text 3

The screenshot shows the Sublime Text 3 interface with the following details:

- Title Bar:** C:\Users\ASUS\Desktop\scraper\main.py (scraper) - Sublime Text (UNREGISTERED)
- Menu Bar:** File Edit Selection Find View Goto Tools Project Preferences Help
- Folders List:** Shows the project structure with folders "scraper", "bs_object.py", "main.py", "test.py", and "urls.py". "main.py" is currently selected.
- Code Editor:** The main window displays Python code for a web scraper. The code includes functions for concatenating article content, creating file structures, and writing articles to files. A red box highlights the word "folders" in the line "folders = relative_url_list[:-1]".
- Status Bar:** Line 47, Column 13, Spaces: 4, Python

```
article = ""
for child in main_content.children:
    article = article + str(child)
return article

def create_file_structure(self, relative_url):
    if relative_url.startswith('/'):
        relative_url = relative_url[1:]

    relative_url_list = relative_url.split("/")
    file_name = relative_url_list[-1] + ".html"
    folders = relative_url_list[:-1]

    file_structure = {'folders': folders, 'file_name': file_name}
    return file_structure

def article_to_file(self, relative_url):
    article = self.get_article()
    file_structure = self.create_file_structure(relative_url)

    initial_path = 'C:/Users/ASUS/Desktop/scrape'

    for folder in file_structure['folders']:
        initial_path = initial_path + '/' + folder
        if not os.path.exists(initial_path):
            os.makedirs(initial_path)

        os.chdir(initial_path)
```

Atom

The screenshot shows the Atom code editor interface. The title bar reads "main.py — C:\Users\ASUS\Desktop\scraper — Atom". The menu bar includes File, Edit, View, Selection, Find, Packages, Help, and PlatformIO. On the left is a sidebar with icons for Home, Project, Checkmark, Right Arrow, Cloud, Trash, Find, and Search. The "Project" section shows a folder "scraper" containing files: bs_object.py, bs_object.pyc, main.py (selected), test.py, and urls.py. The main editor area displays the following Python code:

```
main.py
1 def make_multiplier_of(n):
2     def multiplier(x):
3         return x * n
4     return multiplier
5
6 # Multiplier of 3
7 times3 = make_multiplier_of(3)
8
9 # Multiplier of 5
10 times5 = make_multiplier_of(5)
11
12 # Output: 27
13 print(times3(9))
14
15 # Output: 15
16
```

Below the code, the terminal window shows the execution of the script:

```
Copyright (C) Microsoft Corporation. All rights reserved.
PS C:\Users\ASUS\Desktop\scraper> python main.py
27
15
30
PS C:\Users\ASUS\Desktop\scraper>
```

The status bar at the bottom shows "+ main.py ⚡ 0 ▲ 0 ⌂ 0 19:25", "LF", "UTF-8", "Python", "GitHub", and "Git (0)".

PyCharm

The screenshot shows the PyCharm IDE interface with the following details:

- Title Bar:** scrape [C:\Users\ASUS\PycharmProjects\scrape] - main.py [scrape] - PyCharm
- Menu Bar:** File Edit View Navigate Code Refactor Run Tools VCS Window Help
- Toolbar:** Includes icons for test, run, and search.
- Project Explorer:** Shows the project structure under the 'scrape' folder:
 - scrape (C:\Users\ASUS\PycharmProjects\scrape)
 - venv
 - library.zip
 - bs_object.py
 - main.py
 - test.py
 - urls.py
- Code Editor:** Displays Python code for the 'main.py' file. The code defines two methods: 'create_file_structure' and 'article_to_file'. The 'article_to_file' method calls 'create_file_structure' and then iterates over the resulting file structure to create folders in a specified initial path.
- Status Bar:** Content: article_to_file, Event Log, 54:20 CRLF: UTF-8: 4 spaces.

Visual Studio Code

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer (Left):** Shows extensions installed for Python, including "Python 2019.1.0" by Microsoft, "AREPL for python 1.0.8" by Almenon, "Python Extension 1.4.0" by Popular Visual Studio Code, "Python Preview 0.0.4" by dongli, and "Python for VSCode 0.2.3" by Thomas Haakon.
- Code Editor (Center):** Displays a Python file named "decorator.py" with the following code:

```
1 def star(func):
2     def inner(*args, **kwargs):
3         print("*" * 30)
4         func(*args, **kwargs)
5         print("*" * 30)
6     return inner
7
8 def percent(func):
9     def inner(*args, **kwargs):
10        print("%" * 30)
11        func(*args, **kwargs)
12        print("%" * 30)
13    return inner
14
15 @star
16 @percent
```
- Terminal (Bottom):** Shows the output of running the code in a cmd terminal on Microsoft Windows.

```
Microsoft Windows [Version 10.0.17134.523]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\ASUS\Desktop\scraper>
```
- Status Bar (Bottom):** Shows the Python version (Python 3.7.2 32-bit), status bar indicators (x 0, 0, INSERT), and file statistics (Ln 12, Col 24, Spaces: 4, UTF-8, CRLF, Python).

Spyder

The screenshot shows the Spyder IDE interface with a Python script named `balik.py` open in the editor. The code implements a stack to reverse a sentence. A tooltip for the `push` method is displayed, explaining its usage. The IPython console shows the execution of the script and its output.

```
#balik.py => membalikkan kalimat
from ArrayStack import * #menggali implementasi stack
S=ArrayStack() #menciptakan tumpukan S
kalimat = input('Masukkan satu kalimat : ')
panjang=len(kalimat)
for c in kalimat:
    S.push(c)
x=''
for i in range(panjang):
    x+=S.pop()
print('Baca terbalik : ')
print(x)
```

Usage

Here you can get help of any object by pressing **Ctrl+H** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in Preferences > Help.

New to Spyder? Read our [tutorial](#).

```
In [14]: runfile('D:/USER/Python/balik.py', wdir='D:/USER/Python')
Masukkan satu kalimat : selamat pagi
Baca terbalik :
igap tameles

In [15]:
```

Windows Taskbar:

- Type here to search
- File Explorer
- File History
- File Ready
- LSP Python ready
- Console (Python 3.8.8) Line 20 Col 1 ASCII
- 11:47
- 30PC Selogian cerah
- 24/08/2022
- 9%

Online Editor : Programiz.com

The screenshot shows the Programiz Python Online Compiler interface. At the top, there's a browser header with the URL https://www.programiz.com/python-programming. Below it, the Programiz logo and "Python Online Compiler" text are visible, along with a "Interactive Python Course" button.

The main area has tabs for "main.py", "Run", "Shell", and "Clear". On the far left, there are icons for Python, C, C++, Java, JavaScript, and SQL. The "Run" tab is active, showing the following Python code:

```
1 # Online Python compiler
2 print("This is Programiz Python
       editor ")
3 print("Below are Hello World to
       you !")
4 for i in range(5):
5     print("Hello World")
6
7
```

The output window shows the results of the code execution:

```
This is Programiz Python editor
Below are Hello World to you !
Hello World
Hello World
Hello World
Hello World
Hello World
> |
```

A small pop-up box in the bottom right corner says "certified." and contains a "Claim Discount" button.

Online Editor: onlinetgdb.com

The screenshot shows the OnlineGDB beta IDE interface. On the left, there's a sidebar with navigation links like 'code. compile. run.', 'debug. share.', 'IDE', 'My Projects', 'Classroom', 'Learn Programming', 'Programmino', 'About • FAQ • Blog •', 'Terms of Use •', 'Contact Us • GDB', 'Tutorial • Credits •', 'Privacy', '© 2016 - 2022', and 'GDB Online'. The main area has tabs for 'main.py' and 'Run'. The code editor contains the following Python script:

```
1 print("Hello Every Body")
2 print("I will print 5 hello World")
3 for i in range(5):
4     print("Hello World my Dear")
5 
```

The output window shows the program's execution:

```
Hello Every Body
I will print 5 hello World
Hello World my Dear
...Program finished with exit code 0
Press ENTER to exit console.
```

Struktur Umum Program Python

```
#komentar tentang program  
import pustaka fungsi
```

```
#pemberian nilai awal
```

```
... ...
```

```
# definisi fungsi  
def fungsi-1:  
    ... instruksi ...
```

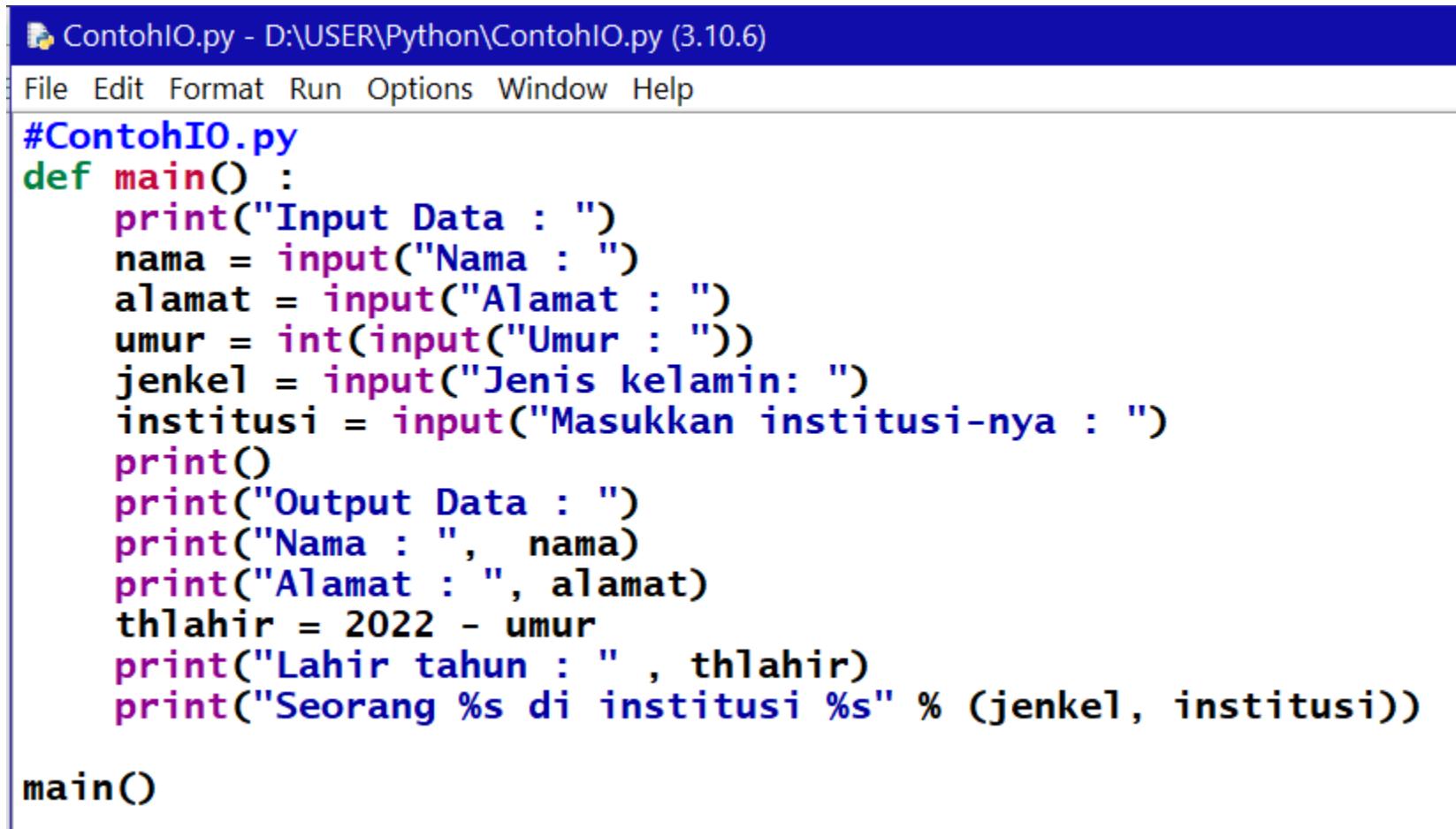
```
def fungsi-2:  
    ... instruksi ...
```

```
...
```

```
#program utama  
def main()  
    ... instruksi utama ...
```

```
#menjalankan program  
main()
```

Contoh program python



```
ContohIO.py - D:\USER\Python\ContohIO.py (3.10.6)
File Edit Format Run Options Window Help
#ContohIO.py
def main():
    print("Input Data : ")
    nama = input("Nama : ")
    alamat = input("Alamat : ")
    umur = int(input("Umur : "))
    jenkel = input("Jenis kelamin: ")
    institusi = input("Masukkan institusi-nya : ")
    print()
    print("Output Data : ")
    print("Nama : ", nama)
    print("Alamat : ", alamat)
    thlahir = 2022 - umur
    print("Lahir tahun : " , thlahir)
    print("Seorang %s di institusi %s" % (jenkel, institusi))

main()
```

Beberapa Pustaka Python

string pustaka untuk proses text
textwrap format text paragraph
re regular expression
collections container data types
array pustaka untuk larik
heapq heap sort
struct binary data structures
time jam, waktu
datetime penanggalan
decimal fixed and floating point
fractions rational numbers
random bilangan acak
math fungsi matematika

os sistem operasi / path
glob filename pattern matching
shutil high level file operation
mmap memory map files
codecs string encoding and decoding
StringIO text buffers with a file API
filecmp compare files
pickle object serialization
shelve persistent storage of objects
anydbm database
sqlite3 relational database
csv comma separated value files
gzip GNU zip files
tarfile Tar archive files
zipfile ZIP archive access
hashlib kriptografi

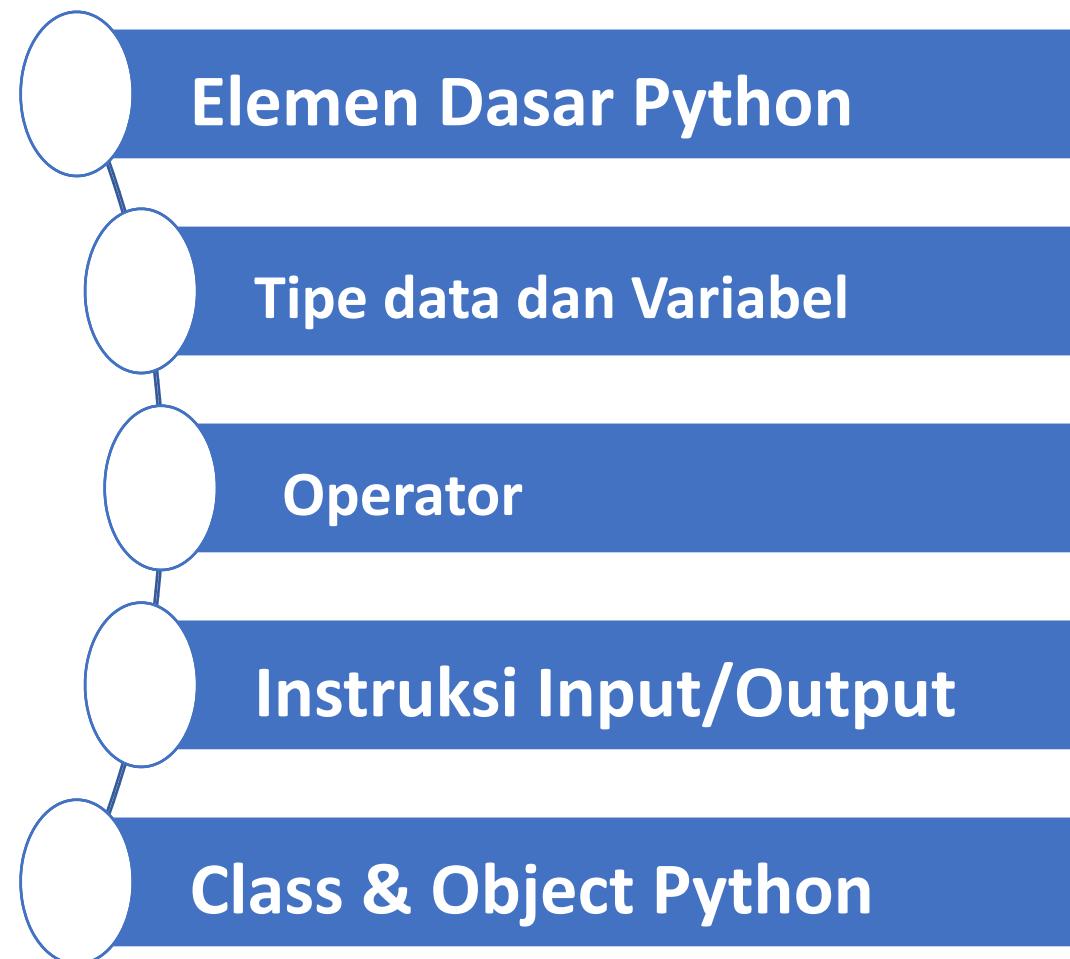


STRUKTUR DATA (PYTHON)

“Struktur Bahasa Python”

[@SUARGA | [Pertemuan 02]

OutLine





Elemen Dasar Bahasa Python

- 1. CharacterSet** #modul-2
- 2. Keywords**
- 3. Tipe data dan Variabel**
- 4. Operator**
- 5. Instruksi Input-Output**
- 6. Instruksi Seleksi** #modul-3
- 7. Instruksi Perulangan**
- 8. Pembuatan Fungsi**
- 9. Akses File**



Character Set

Character set dari Python adalah standard ASCII yaitu dari kode 0 (NUL) hingga kode 127 (DEL) , namun Python bisa mengerti set karakter lain apabila dinyatakan terlebih dahulu misalnya kode utf-8 atau iso-8859-1. Cara menyatakan-nya sebagai berikut:

```
# -*- coding : utf-8 -*-
```

diawali coding program.

	0	1	2	3	4	5	6	7	8	9
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT
1	LF	VT	FF	CR	SO	SI	DLE	DCI	DC2	DC3
2	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS
3	RS	US	SP	!	“	#	\$	%	&	`
4	()	*	+	,	-	.	/	0	1
5	2	3	4	5	6	7	8	9	:	;
6	<	=	>	?	@	A	B	C	D	E
7	F	G	H	I	J	K	L	M	N	O
8	P	Q	R	S	T	U	V	W	X	Y
9	Z	[\]	^	_	‘	a	b	c
10	d	e	f	g	h	i	j	k	l	m
11	n	o	p	q	r	s	t	u	v	w
12	x	y	z	{		}	~	DEL		

ASCII standard character (dalam desimal)

Keyword

- **Keyword** adalah perbendaharaan kata dari Python, dimana setiap kata merupakan suatu instruksi tertentu, oleh sebab itu pengguna hanya boleh memakainya sebagai instruksi, tidak untuk keperluan lain.
- **Keyword** pada dasarnya diperlukan pertama untuk mengetahui sintaks dari command, kedua untuk menjadi acuan ketika membuat variable agar tidak sama dengan keyword.

- **Keyword** dalam Python terdiri atas 35 kata, yaitu:

```
and      def      finally      in      or      try
as       del      for         is       pass     while
assert   elif     from        lambda   print   with
break    else     global      None    raise   yield
class    except   if          nonlocal return  exec
continue False    import     not     True
```

Keyword tdk boleh digunakan sebagai variabel

Variabel

- **Variable** adalah nama atau simbol yang mewakili suatu data/nilai. **Variabel** dalam Python harus diberi nama yang dimulai huruf, bukan angka. Sebagai contoh Nama dan Umur adalah variabel yang melekat pada seseorang, nilai-nya misalnya “Abdus Salam” dan “25 tahun”.
- Programmer Python biasanya memakai huruf kecil diawal nama variable atau huruf kecil pada kata pertama, dan dimulai huruf besar pada kata yang kedua.
- Perlu di-ingat bahwa Python membedakan nama dengan huruf kecil dan nama dengan huruf besar.
- Variabel juga tidak boleh menggunakan spasi.

- **Literal** adalah tetapan angka maupun string, misalnya: 10, 1.23, 9.2e-3, dan “ini adalah string”. Nilai literal adalah nilai yang diberikan kepada variabel.
- **Angka** dalam python bisa dibedakan empat macam, yaitu: **integer**, **long integer**, **floating point**, dan **complex number**, misalnya:
- Integer : 2, 4, 45, 127
- Long integer : 123456789654321L
#memakai huruf l atau L
- Floating point : 3.23, 50.78, 6.34E-4
- Complex number : (-5 + 4j) , (2.3 – 4.7j) #memakai hurf j atau J

- **Komentar** : baris-baris komentar harus diberi tanda # didepan tiap baris
- contoh: #program ini ditulis oleh Suarga
- **String** adalah untaian karakter. String dapat dibentuk menggunakan single-qoute seperti 'Quote me on this', atau memakai double-quote seperti "What's your name?". Disamping itu string yang terdiri atas beberapa baris dapat dibentuk memakai triple-quote '''' atau """", contohnya sebagai berikut:

""" This is a multi-line string, and this is the first line

This is the second line

"What's your name?," I asked.

He said "Bond, James Bond."

"""

- Triple-quote juga dipakai untuk menyatakan komentar yang terdiri atas beberapa baris, dan komentar tersebut harus ditutup dengan triple-quote lagi.

“““ Berikut ini disajikan listing program sebagai dokumentasi dari program yang di tulis oleh Suarga.

“““

- Selanjutnya suatu string yang ingin ditampilkan dalam beberapa baris harus diberi tanda escape \n untuk memisahkan setiap baris, contoh:

```
>>> A = 'Tanda \n pada string membuat string  
dipisahkan \n beberapa baris'
```

```
>>> print(A)
```

Tanda

pada string membuat string dipisahkan
beberapa baris

```
>>>
```

Objek Python

Jenis Objek	Contoh literal
Angka (Numbers)	34215, 3.1415, 1.34E+02, 3+4j, 0b111, Decimal(), Fraction()
Strings	'Spam', "Bob's", b'a\x01c, u'sp\xc4m'
List	[1, [2, 'three'], 4, 5], list(range(10))
Dictionary	{'food' : 'spam', 'taste' : 'yum'}, dict(hours=10)
Tuple	(1, 'spam', 4, 'U'), tuple('spam')
File	Open('myfile.txt', 'a+'), open(r'C:\ham.bin','wb')
Set	Set('abc'), {'a', 'b', 'c'}
Tipe lainnya:	Boolean, types, None
Unit program	Function, Modules, Class

string ADT

- **str()** : mengubah integer ke string
- **int()** : mengubah string ke integer, int(s,B) dapat mengubah string s dalam bentuk sistem bilangan B menjadi integer, misalnya int("10101",2) memberikan nilai 21
- **chr()** : mengubah integer ke karakter, chr(97) adalah 'a'
- **ord()** : mengubah karakter ke nilai integer, ord('A') adalah 65
- **center()** : membuat string berada ditengah
- **ljust()** : membuat string rapat ke kiri
- **rjust()** : membuat string rapat ke kanan

- **strip()** : membuang spasi dari dalam string
- **lstrip()** : membuang spasi di-depan string
- **rstrip()** : membuang spasi di-belakang string
- **count()** : menghitung frekuensi huruf dalam string
- **find()** : mencari suatu substring/karakter dalam string
- **startswith()** : memeriksa apakah huruf tertentu mengawali string
- **endswith()** : memeriksa apakah huruf tertentu mengakhiri string
- **lower()** : mengubah string menjadi huruf kecil
- **upper()** : mengubah string menjadi huruf besar
- **isalpha()** : memeriksa string apakah abjad
- **isupper()** : memeriksa string apakah semuanya huruf besar
- **islower()** : memeriksa string apakah semuanya huruf kecil
- **isdigit()** : memeriksa string apakah semuanya angka

- Cara menggunakan atribut fungsi string ini adalah dengan menulis nama fungsi dibelakang objeknya, misalnya sebagai berikut:

```
>>> Nama = 'abdul rahman'    # definisi objek string
>>> Nama.isalpha()          # spasi bukan alpha?
False
>>> Nama.isupper()          # apa nama huruf besar?
False
>>> Nama.islower()          # apa nama huruf kecil?
True
>>> Nama.isdigit()          # apa nama adalah digit?
False
>>> Nama.startswith('a')     # apa nama diawali huruf 'a'?
True
>>> Nama.endswith('a')       # apa nama diakhiri huruf 'a'?
False
```

Deklarasi Variabel

- Berbeda dengan bahasa program pada umumnya dimana tipe dari variable harus dinyatakan, **Python tidak memerlukan deklarasi tipe dari variable**, karena tipe variable akan disesuaikan secara otomatis dengan nilai data yang diberikan, misalnya:

```
n = 17 # n diberikan nilai integer 17
pi = 3.14159265 # pi diberikan nilai riel atau
                  # floating point
message = 'ini adalah string' #message diberi
                             # nilai string
```

Operator

- Operator ***aritmetika*** dasar terdiri atas:
 - + (**tambah**), - (**kurang**),
 - * (**kali**), / (**bagi**) dan
 - ** (**pangkat**).

Contoh:

```
>>> a=5  
>>> b=8  
>>> c=2  
>>> d=4  
>>> e = a + b*c - d**c + a*b/c  
>>> e  
25.0  
>>>
```

- Pada string operator + berarti (concatenation) atau menyambung string, dan operator * berarti (repeat) atau penggandaan, misalnya:

first = 'Selamat '

second = 'pagi'

maka:

first + second menghasilkan Selamat pagi.

third = first * 3, maka third akan sama dengan "SelamatSelamatSelamat".

- Python menggunakan tanda # untuk memberi komentar pada program, misalnya:

V = 5 # V diberikan nilai 5

v = 10 # laju partikel adalah 10

Python juga memanfaatkan escape letter seperti pada C dan Java, antara lain yang penting adalah:

\n - baris baru

\t - tabulasi

\' - single-quote dalam string, seperti ‘What’s your Name’

\\" - double quote dalam string

Operator Aritmetik

+	plus	Menjumlah 2 objek	$3 + 5 \rightarrow 8$ $'a' + 'b' \rightarrow 'ab'$
-	minus	Mengubah ke negative atau mengurangkan dua objek	-5 ubah ke negative $50 - 24$ mengurangkan
*	multiply	Mengalikan dua objek	$5 * 8 \rightarrow 40$ $'la' * 3 \rightarrow 'lalala'$
**	power	Memangkatkan bilangan	$3 ** 4 \rightarrow 81$
/	divide	Membagi dua bilangan	$4/3 \rightarrow 1$ $4.0/3 \rightarrow 1.33333$

Operator Aritmetik

//	Floor division	Pembulatan ke bawah hasil bagi	$4/3.0 \rightarrow 1.0$
%	modulo	Sisa pembagian bulat	$8\%3 \rightarrow 2$ $25.5 \% 2.25 \rightarrow 1.5$

Operator bit

<code><<</code>	Left shift	Geser bit ke kiri	$2 << 2 \rightarrow 8$, karena 10 digeser 2 kali ke kiri menjadi 1000
<code>>></code>	Right shift	Geser bit ke kanan	$11 >> 1 \rightarrow 5$, karena 1011 geser kanan 1 bit menjadi 0101
<code>&</code>	Bitwise AND	Operasi bit dari AND	$5 \& 3 \rightarrow 1$
<code> </code>	Bitwise OR	Operasi bit dari OR	$5 3 \rightarrow 7$
<code>^</code>	Bitwise XOR	Operasi bit dari XOR	$5^3 \rightarrow 6$
<code>~</code>	Bitwise invert	$\sim x$ adalah $-(x+1)$	$\sim 5 \rightarrow -6$

Operator Relasi

<	Less than	Memeriksa apakah relasi lebih kecil terpenuhi	$5 < 3 \rightarrow 0$ (false) $3 < 5 \rightarrow 1$ (true)
>	Greater than	Memeriksa apakah relasi lebih besar terpenuhi	$5 > 3 \rightarrow 1$ (true) $3 > 5 \rightarrow 0$ (false)
<=	Less than or equal	Relasi lebih kecil atau sama	$3 <= 6 \rightarrow 1$ (true)
>=	Greater than or equal	Relasi lebih besar atau sama	$4 >= 3 \rightarrow 1$ (true) $3 >= 4 \rightarrow 0$ (false)
==	Equal to	Memeriksa kesamaan	$x=2; y=2, (x == y) \rightarrow 1$ 'str' == 'stR' $\rightarrow 0$
!=	Not equal to	Memeriksa ketidaksamaan	$2 != 3 \rightarrow 1$ 'str' != 'str' $\rightarrow 0$
Not	Boolean not	Membalikkan logik	
And	Boolean and	Relasi "DAN"	
Or	Boolean or	Relasi "ATAU"	

Indentation

- Seperti disinggung di awal tulisan bahwa **Python meringkas kelompok instruksi dengan memakai indentation**, sehingga sangat perlu suatu konsistensi dalam membentuk indentation ini. Indentation boleh dipilih salah satu dari: *tab*, *dua-spasi kekanan*, atau *empat-spasi kekanan*.
- Hanya perlu konsistensi. Perhatikan bahwa terkadang suatu error muncul hanya karena salah indentation. Apabila memakai “tab” maka seterusnya pakai tab, jangan dicampur dengan pemakaian spasi, demikian pula sebaliknya.

```
#contoh pemakaian indentation salah
```

```
a=5
```

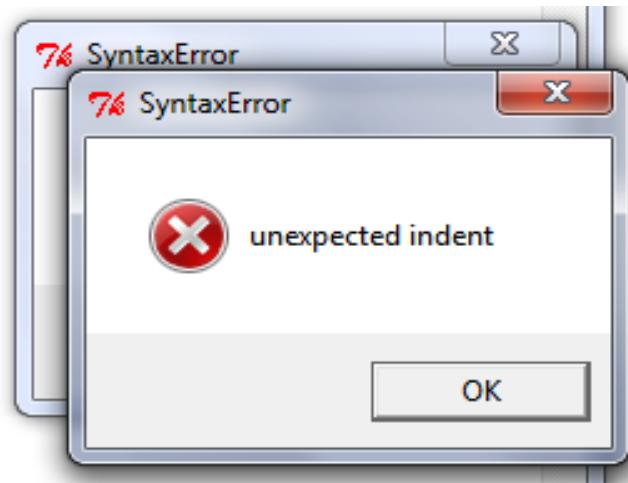
```
b=2
```

```
c=4
```

```
d=8
```

```
e = a + b*c - d
```

- Perhatikan bahwa baris-baris instruksi ini menggunakan indent yang tidak konsisten, maka ketika di-run akan memunculkan error,



Struktur Program Python

```
#komentar tentang program      #program utama main()
import pustaka

#pemberian nilai awal

...
# definisi fungsi
def fungsinya:
    ... instruksi ...
    ...

def fungsinya2:
    ... instruksi ...
    ...

#menjalankan fungsi main()
main()
```

Pustaka fungsi

- **string** pustaka untuk proses text string
- **textwrap** format text paragraph
- **re** regular expression
- **collections** container data types
- **array** pustaka untuk larik
- **heapq** heap sort
- **struct** binary data structures
- **time** jam, waktu
- **datetime** penanggalan
- **decimal** fixed and floating point
- **fractions** rational numbers
- **random** bilangan acak
- **math** fungsi matematika

Pustaka Fungsi

- **os** sistem operasi / path
- **glob** filename pattern matching
- **shutil** high level file operation
- **mmap** memory map files
- **codecs** string encoding and decoding
- **StringIO** text buffers with a file API
- **filecmp** compare files
- **pickle** object serialization
- **shelve** persistent storage of objects
- **anydbm** database

Lakukan “import” untuk menggunakan fungsi pustaka, misalnya:

```
import os      #untuk menggunakan fungsi OS
```

Instruksi Input

- Instruksi untuk menerima data bertipe character dari keyboard adalah :

```
var = input("keterangan : ");
```

contoh:

```
Nama = input('Masukkan nama anda : ')  
umur = int(input('Masukkan umur-nya : '))
```

- bila tipe variabel input bukan char maka berikan tipe data-nya misalnya int.

Instruksi Output

- Instruksi untuk menampilkan sesuatu di monitor adalah:

print('keterangan', variabel);

print("nama : %s, umur:%d" % (nama,umur))

contoh:

x = int(input('Masukkan satu angka : ')

print('Angka yang dimasukkan = ', x)

Program Test operator

```
# testOperator.py
x=6
y=4
a=12.5
b=7

print('x = ', x, '; y = ', y)
print('x + y = ', x+y)
print('x - y = ', x-y)
print('x * y = ', x*y)
print('x / y = ', x/y)
print('x ** y = ', x**y)
print('x // y = ', x//y)
print('x % y = ', (x%y))
print()
print('a = ', a, '; b = ', b)
print('a > b is ', (a>b))
print('a < b is ', (a<b))
```

Hasil test operator

x = 6 ; y = 4

x + y = 10

x - y = 2

x * y = 24

x / y = 1.5

x ** y = 1296

x // y = 1

x % y = 2

a = 12.5 ; b = 7

a > b is True

a < b is False

Membuat program Python

1. Pikirkan logik/algoritma dari masalah yang akan di program
2. Aktifkan editor Python, kemudian edit program yang merupakan translasi algoritma ke Python
3. Rekam program, beri nama dgn ekstensi .py
4. Run program, periksa hasilnya

Contoh

- Buat program Python untuk konversi suhu Celcius ke suhu Fahrenheit
- Logika solusi:
 - masukkan suhu Celcius (C)
 - konversi dgn rumus: $F = 9/5 * C + 32$
 - Tampilkan nilai F

C_to_F.py - D:/USER/Documents/myLecture/Algoritma/BukuStrukturData/ModulStrukturData

File Edit Format Run Options Window Help

#C_to_F.py konversi Celcius ke Fahrenheit

```
def main():
    C = float(input("Masukkan suhu Celcius: "))
    F = 9.0/5.0 * C + 32.0
    print("Untuk Suhu Celcius = ", C)
    print("Fahrenheit = ", F)

main()
```

```
= RESTART: D:/USER/Documents/myLecture/ALGORITMA/STRUKTURDATA/C_to_F.py
Masukkan suhu Celcius: 25
Untuk Suhu Celcius =  25.0
Fahrenheit =  77.0
```

Beberapa Contoh Program

- [`investasi.py`](#) : data input: modal, bunga, jangka waktu (tahun), output: daftar investasi dari tahun awal ke tahun akhir
- [`pisahdigit.py`](#): data input, sebuah angka bulat, output: setiap digit-nya dipisah
- [`ContohIO.py`](#): data input: nama, alamat, umur, jenis kelamin, dan institusi, output: tampilan kembali dari data input

investasi.py

```
# investasi.py - menghitung bunga uang simpanan
modal = input('Masukkan simpanan awal : ')
modal = float(modal)
rate = 0.07
bunga = modal * rate
investasi = modal + bunga
print('Modal awal = %10.2f' % modal)
print('Bunga = %10.2f' % bunga)
print('Nilai investasi setelah 1 tahun = %10.2f' % investasi)
```

pisahdigit.py

```
#pisahdigit.py - memisah digit angka bulat
angka = int(input('Masukkan angka 4 digit : '))
d1 = angka//1000
sisa = angka % 1000
d2 = sisa//100
sisa = sisa % 100
d3 = sisa //10
sisa = sisa % 10
d4 = sisa
print('Digit pertama = ', d1)
print('Digit kedua    = ', d2)
print('Digit ketiga   = ', d3)
print('Digit keempat  = ', d4)
```

ContohIO.py

```
#ContohIO.py
def main() :
    print("Input Data : ")
    nama = input("Nama : ")
    alamat = input("Alamat : ")
    umur = int(input("Umur : "))
    jenkel = input("Jenis kelamin: ")
    institusi = input("Masukkan institusi-nya : ")
    print()
    print("Output Data : ")
    print("Nama : ", nama)
    print("Alamat : ", alamat)
    thlahir = 2022 - umur
    print("Lahir tahun : " , thlahir)
    print("Seorang %s di institusi %s" % (jenkel, institusi))

main()
```

Instruksi Control

- Instruksi control terdiri atas:
- instruksi seleksi / pemilihan : **if/else**
- instruksi perulangan: **for dan while**
- Akan dibahas dalam modul-3

Preview Tipe data lanjut

- Python memiliki **struktur data internal** yang bersifat objek seperti array, string, list, tuple, dictionary, dan set.
- Secara rinci tipe data objek ini akan diuraikan pada Modul-5 nanti
- Pada modul ini hanya akan diperkenalkan saja sebagai tipe data lanjut.

list

- Tipe data “list” dimanfaatkan untuk menyimpan data seperti sebuah daftar.
- Contoh: daftar = [2, 3, 4, 5]
- tanda kurung [] menandakan objek list, sehingga daftar akan bertipe list dengan isi 2, 3, 4, dan 5.
- Karena bersifat objek maka list menyediakan member function dalam ADT-nya antara lain:

- **append(x)** : menambahkan satu item di ujung kanan dari list
- **clear()** : mengosongkan isi list
- **count(x)** : menghitung frekuensi x dalam list
- **copy()** : menyalin isi list
- **extend(L)** : meng-eksetensi suatu list dengan menambahkan L
- **insert(i,x)** : menyisipkan item x pada posisi i
- **remove(x)** : membuang item pertama bernilai x dari list
- **pop(i)** : membuang item yang berada pada posisi i,
- **pop()** : membuang item terakhir
- **index(x)** : menemukan index dari item x
- **sort()** : mengurutkan item dalam list
- **reverse()** : membalikkan urutan item

```
>>> daftar=[2, 3, 4, 5]      [2, 3, 4, 5, 8, 6]
>>> daftar.append(6)        >>> daftar.sort()
>>> daftar.append(7)        >>> daftar
>>> daftar                  [2, 3, 4, 5, 6, 8]
[2, 3, 4, 5, 6, 7]          >>> daftar.reverse()
>>> daftar.insert(4,8)       >>> daftar
>>> daftar                  [8, 6, 5, 4, 3, 2]
[2, 3, 4, 5, 8, 6, 7]        >>> daftar.remove(4)
>>> daftar.pop()            >>> daftar
7                            [8, 6, 5, 3, 2]
>>> daftar                  >>>
```

tuple

- hampir sama dengan list, tetapi tuple tidak memerlukan tanda kurung atau memakai tanda kurung biasa ()

```
>>> t = 12345, 54321, 'hello!'
```

```
>>> t[0]
```

```
12345
```

```
>>> t
```

```
(12345, 54321, 'hello!')
```

```
>>> u = t,(1, 2, 3, 4, 5)
```

```
>>> u
```

```
((12345, 54321, 'hello!'), (1, 2, 3, 4, 5))
```

```
>>>
```

dictionary (dict)

- dictionary merupakan daftar dimana setiap item memerlukan dua elemen, yaitu: key dan data
- dictionary memerlukan tanda kurung { } untuk menciptakan-nya.

```
>>> phone = {'Amir':3456, 'Bandu':7349, 'Chomas':5436}
>>> phone['Amir']
3456
>>> phone['Daud']=2645
>>> phone
{'Bandu': 7349, 'Chomas': 5436, 'Daud': 2645, 'Amir':
3456}
>>> phone.keys()
dict_keys(['Bandu', 'Chomas', 'Daud', 'Amir'])
>>> del phone['Daud']
>>> phone
{'Bandu': 7349, 'Chomas': 5436, 'Amir': 3456}
>>>
```

Class / Object Python

- **Sifat-sifat Class:**
 - **attribut**
 - **method**
 - **constructor**
 - **encapsulation**
 - **inheritance**
 - **polymorphism**

- Suatu class digunakan untuk mendefinisikan suatu model object
- Object diciptakan melalui constructor class
- sifat-object dinyatakan sebagai attribut didalam class
- kelakuan/fungsi object dinyatakan sebagai method di dalam class
- atribut dan method suatu class terlindung dari class lain (encapsulation)
- atribut dan method suatu class dapat diwariskan (inheritance) ke class lainnya
- beberapa method boleh memakai nama yang sama (polymorphism) asalkan attribut-nya berbeda

membuat Class

Bentuk umum class:

```
class Nama-class:  
    def __init__(self, data, ...):  
        self.atribut = data  
        ...  
    def method(self, parameter):  
        ...
```

contoh: class Circle

```
#kelas Circle, Circle.py
from math import pi, hypot
class myCircle:
    #constructor
    def __init__(self, x=0, y=0, radius=1):
        self.r = radius
        self.x = x
        self.y = y

    def keliling(self):
        return 2.0*pi*(self.r)
```

```
def luas(self):
    return pi*(self.r ** 2)

def pusat(self):
    print("Lingkaran pusat: (%.2f, %.2f)" % (self.x, self.y))
    print("Jari-jari : %.2f" % self.r)

def distFromOrigin(self):
    return hypot(self.x, self.y)

def main():
    C = myCircle(37.0, 43.0, 2.50)
    C.pusat()
    area = C.luas()
    kel = C.keliling()
    jarak = C.distFromOrigin()
    print("Luasnya %f" % area)
    print("Keliling %f" % kel)
    print("Jarak lingkaran dari titik pusatnya %f" % jarak)

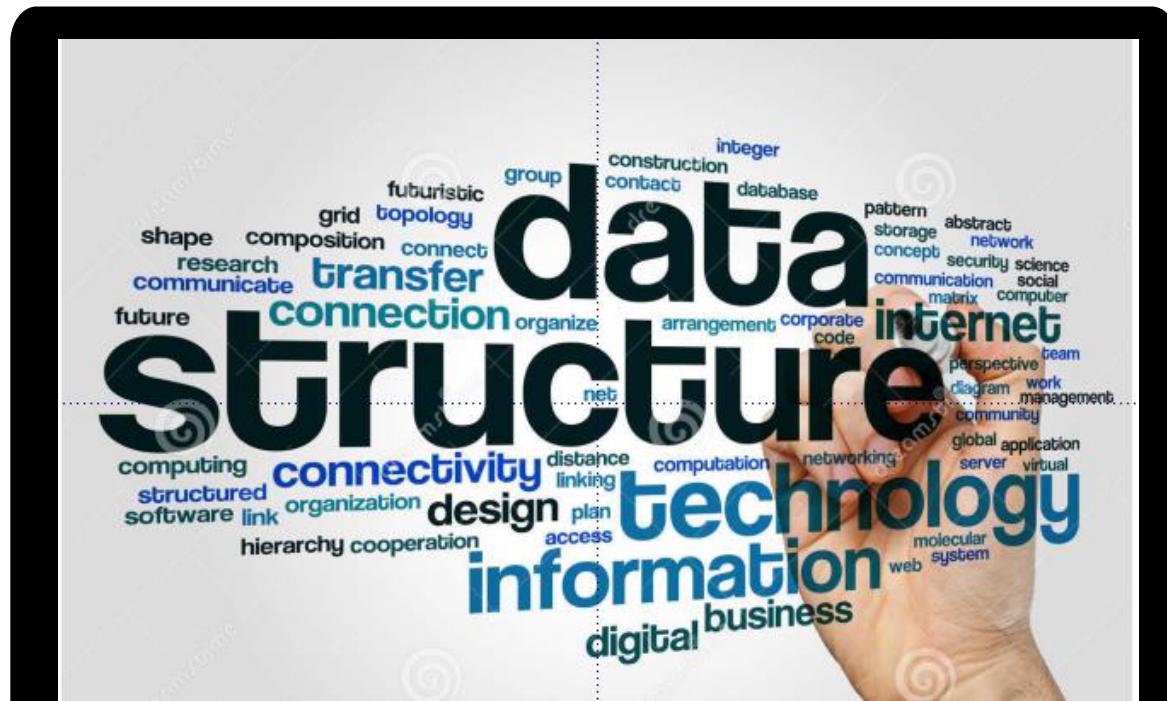
main()
```

hasil RUN

```
>>> %Run Circle.py  
Lingkaran pusat: (37.00, 43.00)  
Jari-jari : 2.50  
Luasnya 19.634954  
Keliling 15.707963  
Jarak lingkaran dari titik pusatnya 56.727418
```

Kuiz

1. Bahasa Python termasuk kategori Compiler atau Interpreter?
2. Sebutkan objek dasar dari Pyhton beserta contoh-nya
3. Kalau mau meng-install Python kemana anda mencari installer-nya dan apa nama file installer-nya?
4. Berapa nilai dari ekspressi Pyhton berikut ini:
 - a. $2 * (3 + 4) ** 2$
 - b. $2 * 3 + 4 ** 2$
 - c. $2 + 3 * 4 ** 2$
5. Berapa nilai ekspressi Python berikut ini:
 - a. $5 * 4 ** 2 / 2 - 4 * 5 ** 2$
 - b. $(5 * 4) ** 2 / (2 - 4 * 5) ** 2$
 - c. $(5 * 4 ** 2 / 2) - 4 * 5 ** 2$



STRUKTUR DATA (PYTHON)

“Instruksi Utama Python”

[@SUARGA | Pertemuan 03+04]

OutLine





Instruksi Utama Python

- 1. Instruksi Seleksi:** digunakan untuk memilih instruksi yang akan di-eksekusi selanjutnya berdasarkan pada suatu syarat / kondisi. Bilamana syarat dipenuhi maka instruksi di-laksanakan, bila tidak maka hal lain dilakukan
- 2. Instruksi Perulangan:** digunakan untuk mengulang serangkaian instruksi pada perulangan yang terbatas atau tidak terbatas, namun dapat dihentikan berdasarkan suatu kondisi.
- 3. Pembuatan Fungsi :** Pembuatan Fungsi merupakan bagian penting dari pemrograman Python

Instruksi Seleksi if/else

- Instruksi **if** digunakan untuk memilih instruksi yang akan dilaksanakan berdasarkan suatu syarat atau relasi, bentuknya sebagai berikut:

if (relasi) :

true statements #kerjakan bila benar

else:

false statements #kerjakan bila salah

- Contoh apabila anda ingin membuat pilihan antara menampilkan nilai $(x+10)$ bila x bernilai kurang dari 5, dengan nilai $(x*10)$ bila x bernilai lebih dari 5, maka secara algoritmik anda menuliskan:

Algoritma:

Bila $x < 5$:

maka tampilkan $(x + 10)$

selain itu tampilkan $(x * 10)$.

Python:

```
if ( x < 5 ) :
```

```
    print( x + 10)
```

```
else:
```

```
    print( x * 10)
```

pemilihan multi-syarat

- Instruksi seleksi dapat dibuat multi syarat/relasi, dengan memakai *elif* diantaranya, bentuknya sebagai berikut.

```
if (relasi-1):
    statement-1
elif (relasi-2):
    statement-2
elif (relasi-3):
    statement-3
else:
    statement-4
```

Contoh

```
#pilih_Kode.py
#contoh pemakaian if/elif/else
kode = int(input("Masukkan kode agama (1-5) : "))
if (kode==1):
    agama = "Islam"
elif (kode==2):
    agama = "Kristen"
elif (kode==3):
    agama = "Katholik"
elif (kode==4):
    agama = "Hindu"
elif (kode==5):
    agama = "Budha"
else:
    agama = "????? "
    print("Salah kode : harus antara 1 sd 5")

print("Anda memasukkan kode %d untuk agama %s" % (kode,agama))
```

Instruksi match/case (v3.10)

```
kode = int(input("Masukkan angka antara 1 dan 5 : "))
match kode:
    case 1:
        agama = "Islam"
    case 2:
        agama = "Kristen"
    case 3:
        agama = "Katolik"
    case 4:
        agama = "Hindu"
    case 5:
        agama = "Budha"
    case other:
        print("Salah Kode : harus antara 1 sd 5")
        agama='?????'
print("Anda memasukkan kode %d untuk agama %s" % (kode,agama))
```

Contoh Persamaan Kuadrat

```
#akar.py - mencari akar pers ax**2 + bx + c
import math
a = float(input('Masukkan koefisien a : '))
b = float(input('Masukkan koefisien b : '))
c = float(input('Masukkan koefisien c : '))

D = b**2 - 4*a*c
if (D > 0):          #akar riell
    D = math.sqrt(D)
    x1 = (-b + D)/(2*a)
    x2 = (-b - D)/(2*a)
elif (D < 0):        #akar complex
    D = math.sqrt(-D)
    x1 = (-b/2*a) + (D/2*a)*1j
    x2 = (-b/2*a) - (D/2*a)*1j
else:                 #akar kembar
    x1 = -b/(2*a)
    x2 = -b/(2*a)

print('Akar persamaan kuadrat:')
print('x1 = ', x1)
print('x2 = ', x2)
```

Instruksi Perulangan

- instruksi **for** memakai daftar (list)
- instruksi for memakai batas perulangan (range)
- instruksi **while** dengan syarat/kondisi

Contoh for

- Program:

```
# bentuk pertama  
a =['pintu', 'jendela',  
'kusen', 'teralis']  
for x in a:  
    print (x, len(x))
```

```
# bentuk kedua  
for i in range(10):  
    print(i, i*5)
```

- Hasil :

pintu 5
jendela 7
kusen 5
teralis 7

0 0
1 5
2 10
3 15
4 20
5 25
6 30
7 35
8 40
9 45

Contoh for bersusun

```
for n in range(2,21):      # Loop 1 : n bernilai 2 s/d 9
    for x in range(2,n):    # Loop 2 : x bernilai 2 s/d n
        if (n % x) == 0:    # bila n habis dibagi oleh x
            print (n, ' = ', x, '*', n/x)
            break             # Loop 2 x berakhir
    else:                  # keluar loop x, else dari for
        print (n, ' adalah bilangan prima')
```

```
= RESTART: D:\USER\Documents: 11 adalah bilangan prima
2 adalah bilangan prima 12 = 2 * 6.0
3 adalah bilangan prima 13 adalah bilangan prima
4 = 2 * 2.0 14 = 2 * 7.0
5 adalah bilangan prima 15 = 3 * 5.0
6 = 2 * 3.0 16 = 2 * 8.0
7 adalah bilangan prima 17 adalah bilangan prima
8 = 2 * 4.0 18 = 2 * 9.0
9 = 3 * 3.0 19 adalah bilangan prima
10 = 2 * 5.0 20 = 2 * 10.0
```

Perulangan while

- Selain dengan instruksi for, serangkaian instruksi dapat diulang memakai instruksi while, bentuknya:

While (syarat) :

do-something

Instruksi tersebut sepadan dengan kalimat berikut:

**Selama (syarat) ini masih berlaku
maka lakukan: do-something
atau: lakukan do-something hingga syarat tidak
berlaku**

Contoh Program

Disain program untuk menebak angka dengan interface dialog sebagai berikut:

Halo kawan, siapa nama anda?

Eko

Ok, Eko saya memikirkan satu angka antara 1 s/d 20

Coba anda tebak,

10

Angka tersebut lebih kecil dari 10

Tebak lagi,

2

Angka tersebut lebih besar dari 2

Tebak lagi,

4

Bagus, anda telah menebaknya dalam 3 langkah.

```
# program tebakan angka
#tebakan.py
import random          # import pustaka fungsi acak
maxi = 6                # jumlah tebakan maksimum

jumTebakan = 0
print('Hallo kawan, siapa nama anda?')
nama = input()
angka = random.randint(1, 20) # menciptakan angka acak antara 1 s/d 20

print('Ok ' + nama + ' saya memikirkan satu angka antara 1 s/d 20')
while (jumTebakan < maxi):
    print('Coba anda tebak, ')
    tebak = input()
    tebak = int(tebak)
    jumTebakan = jumTebakan + 1
    if (tebak < angka) :
        print('Angka tersebut lebih besar dari ' + str(tebak))
    if (tebak > angka):
        print('Angka tersebut lebih kecil dari ' + str(tebak))
    if (tebak == angka):
        break

if (tebak == angka):
    print ('Bagus, anda telah menebaknya dalam ' + str(jumTebakan) \
          + ' langkah')
if (tebak != angka):
    print ('Maaf, anda tidak berhasil menebak angka ' + str(angka))
```

Contoh Run :

>>>

Hallo kawan, siapa nama anda?

EKO

Ok EKO saya memikirkan satu angka antara 1 s/d 20

Coba anda tebak,

4

Angka tersebut lebih besar dari 4

Coba anda tebak,

20

Angka tersebut lebih kecil dari 20
Coba anda tebak,
12

Angka tersebut lebih kecil dari 12
Coba anda tebak,
8

Angka tersebut lebih kecil dari 8
Coba anda tebak,

6
Angka tersebut lebih kecil dari 6
Coba anda tebak,

5
Bagus, anda telah menebaknya dalam 6 langkah

Definisi Fungsi

- Suatu fungsi dalam python didefinisikan melalui keyword “**def**”, diikuti oleh nama fungsi dan argumen-nya. Sebagai contoh akan dibuat fungsi untuk menghitung:
- $y = x^{**3} + 10*x^{**2}-15*x + 20$
- dimana nilai x dimasukkan lewat keyboard

```
#fungsi_x.py
#definisi fungsi y(x)
def y(x):
    y = x**3 + 10*x**2-15*x + 20
    return y

print("Program menghitung fungs:")
print("y(x) = x**3 + 10*x**2-15*x + 20")
while(True):
    x = int(input("Masukkan nilai x : "))
    yx = y(x) #memanggil y(x) untuk dihitung nilainya
    print("untuk x = %d, y = %d" % (x,yx))
    print()
    jawab = input("Masih mau tanya lagi? (y/n):")
    if (jawab == 'n'):
        print("See you again !")
        break
```

Contoh RUN

```
= RESTART: D:/USER/Documents/myLec
```

```
Program menghitung fungsi:
```

```
y(x) = x**3 + 10*x**2 - 15*x + 20
```

```
Masukkan nilai x : 2
```

```
untuk x = 2, y = 38
```

```
Masih mau tanya lagi? (y/n):y
```

```
Masukkan nilai x : 5
```

```
untuk x = 5, y = 320
```

```
Masih mau tanya lagi? (y/n):n
```

```
See you again !
```

```
def fibo2(n):
    # menampilkan deret fibonacci sampai
    # nilainya <=n.
    # hasil ditampilkan horisontal
    hasil = []
    a, b = 0, 1
    while (b < n):
        hasil.append(b)
        a, b = b, a+b
    return hasil
```

- Run Agar hasilnya bisa ditampilkan, lakukan sebagai berikut.

```
>>> x=fibo2(2000)
>>> x
[1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597]
```

```
>>>
```

Konsep Larik

- Larik atau array adalah struktur data fundamental dalam berbagai bahasa pemrograman, karena struktur data lainnya dapat di-implementasi menggunakan array sebagai struktur dasar.
- Larik berarti suatu deretan tempat/lokasi/sel yang bisa diisi data, setiap lokasi dapat diisi satu datum. Abstraksi dari lokasi memori komputer juga menggunakan larik. Semua bahasa program tingkat tinggi menyediakan struktur larik/array ini.
- Larik 1D disebut vektor, larik 2D disebut matriks

Deklarasi Larik

- Cara mendefinisikan larik/array sangat bergantung pada bahasa program yang digunakan, misalnya: suatu array dengan 6 elemen berupa karakter akan didefinisikan maka pada bahasa:

```
PASCAL :  x : array [0 .. 5] of character; atau,  
          : type larik : array[0 .. 5] of character;  
          var     A : larik;
```

```
C/C++   : char A[6];
```

```
JAVA    : char A[];  A[] = new char[6];
```

- Python pada hakikatnya tidak memerlukan definisi awal (deklarasi) variabel, tetapi langsung ke pemberian nilai variabel, sehingga pernyataan untuk contoh array x yang berisi ‘SAMPLE’ dapat ditulis sebagai:

PYTHON : `x = array('u', ['S','A','M','P','L','E'])`

dimana ‘u’ menyatakan bahwa karakter memakai unicode.

```
primes = array('i', [2,3,5,7,11,13,17,19])
```

mendefinisikan primes sebagai larik bilangan bulat (integer), yaitu angka prima yang < 20.

Pemakaian array ini memiliki keterbatasan, hanya yang memiliki jenis kode tertentu yang dapat dimasukkan ke array.

Code	C Data Type	Typical Number of Bytes
'b'	<code>signed char</code>	1
'B'	<code>unsigned char</code>	1
'u'	<code>Unicode char</code>	2 or 4
'h'	<code>signed short int</code>	2
'H'	<code>unsigned short int</code>	2
'i'	<code>signed int</code>	2 or 4
'I'	<code>unsigned int</code>	2 or 4
'l'	<code>signed long int</code>	4
'L'	<code>unsigned long int</code>	4
'f'	<code>float</code>	4
'd'	<code>float</code>	8

Tabel code tipedata array

selain code diatas tidak dapat dimasukkan dalam array, misalnya nama tipe string tidak dapat dimasukkan.

Cara memakai modul array

```
# 1. import module array
>>> from array import array
# 2. cara definisi array integer
>>> a=array('i',[1,2,3])
# 3 menampilkan array a
>>> a          #atau print(a)
array('i', [1, 2, 3])
>>> a[0]      #akses a[0], index mulai 0
1
>>> print(a) #print isi a
array('i', [1, 2, 3])
```

ADT Modul Array

The constructor is:

`array(typecode [, initializer]) -- create a new array`

| Methods:

- | `append() -- append a new item to the end of the array`
- | `buffer_info() -- return information giving the current memory info`
- | `byteswap() -- byteswap all the items of the array`
- | `count() -- return number of occurrences of an object`
- | `extend() -- extend array by appending multiple elements from an iterable`
- | `fromfile() -- read items from a file object`
- | `fromlist() -- append items from the list`
- | `frombytes() -- append items from the string`
- | `index() -- return index of first occurrence of an object`

ADT Modul Array

```
| insert() -- insert a new item into the array at a provided  
|           position  
| pop() -- remove and return item (default last)  
| remove() -- remove first occurrence of an object  
| reverse() -- reverse the order of the items in the array  
| tofile() -- write all items to a file object  
| tolist() -- return the array converted to an ordinary list  
| tobytes() -- return the array converted to a string
```

```
| Attributes:
```

```
| typecode -- the typecode character used to create the array  
| itemsize -- the length in bytes of one array item
```

Menciptakan Modul array

- Berikut ini disajikan satu model larik (array) biasa, dibangun sebagai suatu program Python, dan memiliki fungsi ADT sebagai berikut:
- **Array(size)** : inisialisasi array berukuran size, A = Array(7), inisialisasi larik dengan 7 lokasi data
- **length()** : panjang atau banyaknya data dalam array, A.length()
- **getitem(index)** : mengambil isi array pada lokasi index, x=A[5]
- **setitem(index, value)** : mengganti isi pada lokasi index menjadi value, misalnya A[4]=10
- **clear(value)** : membersihkan larik dan menetapkan semua nilai menjadi value, A.clear(3), nilai semuanya 3
- **iterator()** : menciptakan objek iterasi untuk larik
- **fromList(L)** : menyalin isi list L kedalam array
- **fromString(Str)** : menyimpan nilai ord dari setiap huruf dalam string Str

inisialisasi class Array

```
class Array :  
    # menciptakan array dengan 'size' elemen.  
    def __init__( self, size ):  
        assert size > 0, "Array size must be > 0"  
        self._size = size  
        # menciptakan array memakai ctypes module.  
        PyArrayType = ctypes.py_object * size  
        self._elements = PyArrayType()  
        # Initialisasi tiap element.  
        self.clear( None )
```

dengan tipe `ctypes.py_object` maka `Array` dapat menerima angka maupun string

```
# Array.py => Implementasi ADT Array ADT memakai struktur ctypes module.
#copyright @Suarga
import ctypes

class Array :
    # menciptakan array dengan 'size' elemen.
    def __init__( self, size ):
        assert size > 0, "Array size must be > 0"
        self._size = size
        # menciptakan array memakai ctypes module.
        PyArrayType = ctypes.py_object * size
        self._elements = PyArrayType()
        # Initialisasi tiap element.
        self.clear( None )

    # memberikan 'size' dari array.
    def __len__( self ):
        return self._size

    # memberikan elemen sesuai index.
    def __getitem__( self, index ):
        assert index >= 0 and index < len(self), "Array subscript out of range"
        return self._elements[ index ]

    # mengganti elemen pada index dengan nilai value.
    def __setitem__( self, index, value ):
        #print('index = ',index)
        assert index >= 0 and index < len(self), "Array subscript out of range"
        self._elements[ index ] = value
```

```
# mengganti semua elemen array menjadi value.
def clear( self, value ):
    for i in range( len(self) ) :
        self._elements[i] = value

# memberikan iterator dari array.
def __iter__( self ):
    return _ArrayIterator( self._elements )

# menyalin isi list L ke array
def fromList(self,L):
    n = self._size
    m = len(L)
    if (m > n):
        self.__init__(m)
    i=0
    for x in L:
        self._elements[i] = x
        i += 1

# menyalin nilai setiap huruf dari string Str ke array
def fromString(self, Str):
    n = self._size
    m = len(Str)
    if (m > n):
        self.__init__(m)

    for i in range(m):
        self._elements[i]=ord(Str[i])
```

```
# An iterator for the Array ADT.
class _ArrayIterator :
    def __init__( self, theArray ) :
        self._arrayRef = theArray
        self._curNdx = 0

    def __iter__( self ) :
        return self

    def __next__( self ) :
        if self._curNdx < len( self._arrayRef ) :
            entry = self._arrayRef[ self._curNdx ]
            self._curNdx += 1
            return entry
        else :
            raise StopIteration
```

Contoh pemakaian ADT array :

```
>>> from Array import Array
>>> larik = Array(7)
>>> larik.clear(5)
>>> for x in larik: print(x)
5
5
5
5
5
5
5
5
>>> larik.__setitem__(3,7) # bisa ditulis sebagai
larik[3]=7
>>> larik.__setitem__(5,3) # bisa ditulis sebagai
larik[5]=3
>>> for x in larik: print(x)
5
5
5
7
5
3
5
```

```
>>> B=Array(5)
>>> L=[1, 4, 6, 3, 2]
>>> B.fromList(L)
>>> for x in B: print(x)
1
4
6
3
2
Lst=["Ali","Badu","Cephi",""
Daud"]
B = Array(4)
B.fromList(Lst)
for i in B: print(i)
Ali
Badu
Cephi
Daud
```

```
>>> D=Array(5)
>>> x='abcdefghijklk'
>>> D.fromStr(x)
>>> for y in D: print(y)
#nilai desimal dari huruf a sd k
97
98
99
100
101
102
103
104
105
106
107
```

Larik memakai python list

```
# Implementasi ADT Larik memakai python list.
# Larik.py == @Suarga
class Larik :
    # menciptakan array dengan 'size' elemen.
    def __init__( self, size ):
        # assert size > 0, "Array size must be > 0"
        self._size = size
        # menciptakan array kosong memakai list
        self._elements = []
        for i in range(size):
            self._elements.append(None)

    # memberikan 'size' dari array.
    def __len__( self ):
        return self._size

    # memberikan elemen sesuai index.
    def __getitem__( self, index ):
        assert index >= 0 and index < len(self), "indeks larik diluar batas"
        return self._elements[ index ]
```

```
# mengganti elemen pada index dengan nilai value.
def __setitem__( self, index, value ):
    #print('index = ',index)
    assert index >= 0 and index < len(self), "indeks tarik diluar batas"
    self._elements[ index ] = value

# mengganti semua elemen array menjadi value.
def clear( self, value ):
    for i in range( len(self) ) :
        self._elements[i] = value

# memberikan iterator dari array.
def __iter__( self ):
    return _ArrayIterator( self._elements )

# menyalin isi list L ke array
def fromList(self,L):
    n = self._size
    m = len(L)
    if (m > n):
        self.__init__(m)
    i=0
    for x in L:
        self._elements[i] = x
        i += 1
```

```
# menyalin nilai setiap huruf dari string Str ke array
def fromString(self, Str):
    n = self._size
    m = len(Str)
    if (m > n):
        self.__init__(m)

    for i in range(m):
        self._elements[i]=ord(Str[i])

# mengisi larik
def isiLarik(self):
    n = self._size
    for i in range(n):
        print('indeks ',i)
        self._elements[i] = int(input(' : '))

# mencari x dalam larik
def cariX(self,x):
    n = self._size
    ketemu = False
    for i in range(n):
        if (x == self._elements[i]):
            print('Ketemu pd index ', i)
            ketemu = True
    if (not ketemu):
        print(x, ' tidak ada dalam larik')
```

```
# menghapus x dari larik
def hapusX(self,x):
    n = self._size
    ketemu = False
    for i in range(n):
        if (x == self._elements[i]):
            print('Ketemu pd index ',i)
            ketemu = True
            print('Sudah dihapus ')
            A[i]=''
    if (not ketemu):
        print(x,' tidak ada dalam array ')

#baca isi larik
def bacaLarik(self):
    r = input('0-Semua-data  1-hanya satu data : ')
    r = int(r)
    if r < 1:
        for x in self._elements:
            print(x)
    else:
        i = int(input('index : '))
        print(self._elements[i])
```

```
# An iterator for the Array ADT.
class _ArrayIterator :
    def __init__( self, theArray ):
        self._arrayRef = theArray
        self._curNdx = 0

    def __iter__( self ):
        return self

    def __next__( self ):
        if self._curNdx < len( self._arrayRef ) :
            entry = self._arrayRef[ self._curNdx ]
            self._curNdx += 1
            return entry
        else :
            raise StopIteration
```

ADT dari Larik.py

Beberapa ADT tambahan adalah:

isiLarik : mengisi data ke larik

cariX : mencari data dalam larik

hapusX : menghapus satu data

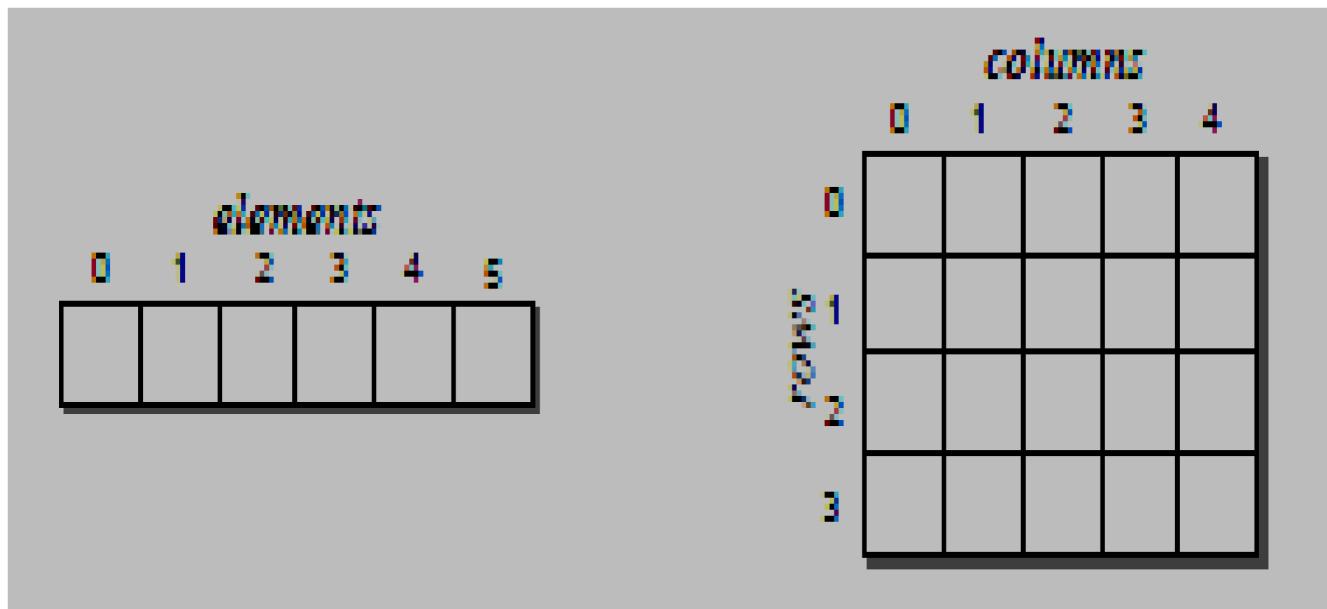
bacaLarik : membaca isi larik

```
>>> A=Larik(5)
>>> A.isiLarik()
indeks 0
: 5
indeks 1
: 3
indeks 2
: 7
indeks 3
: 8
indeks 4
: 2
>>> A.bacaLarik()
0-Semua-data 1-hanya satu data : 0
5
3
7
8
2
```

```
>>> A.bacaLarik()
0-Semua-data 1-hanya satu data : 1
index : 3
8
>>> A.hapusX(1)
1 tidak ada dalam array
>>> A.cariX(4)
4 tidak ada dalam larik
>>> A.cariX(7)
Ketemu pd index 2
>>> A.hapusX(2)
Ketemu pd index 4
Sudah dihapus
>>> A.bacaLarik()
0-Semua-data 1-hanya satu data : 0
5
3
7
8
```

Larik 2D

- Larik 2D adalah larik dengan dua indeks yaitu indeks baris dan indeks kolom.



ADT Larik 2D

- Abstraksi ADT dari larik 2D adalah sebagai berikut.
- **Array2D(nrows, ncols)** : inisialisasi larik 2D dengan nrows baris dan ncols kolom, mula-mula diisi None
- **numRows()** : fungsi untuk memperoleh jumlah baris
- **numCols()** : fungsi untuk memperoleh jumlah kolom
- **clear(value)** : menginisialisasi dengan semua data bernilai value
- **getitem(b, k)** : mengambil data pada baris-b dan kolom-k, boleh ditulis $y = M[b, k]$
- **setitem(b, k, value)** : mengganti nilai data pada baris-b dan kolom-k, boleh ditulis $M[b,k] = value$.
- **dispArr2D()** : menampilkan isi larik 2D

```
>>> B=Larik2D(3,3)          >>> B.dispArr2D()
>>> B.isi2D()
index [0, 0]:
: 1
index [0, 1]:
: 2
index [0, 2]:
: 3
index [1, 0]:
: 4
index [1, 1]:
: 5
index [1, 2]:
: 6
index [2, 0]:
: 7
index [2, 1]:
: 8
index [2, 2]:
: 9
```

```
[1, 2, 3]
[4, 5, 6]
[7, 8, 9]
>>> B.numRows()
3
>>> B.numCols()
3
>>> print(B[1,1])
5
>>> B[1,1]=6
>>> B.dispArr2D()
[1, 2, 3]
[4, 6, 6]
[7, 8, 9]
>>>
```

Inisialisasi Larik2D

```
from Larik import Larik
class Larik2D :
# menciptakan larik 2-D dengan ukuran numRows x numCols.
    def __init__( self, numRows, numCols ):
        # Create a 1-D array to store an array reference
        # for each row.
        self._theRows = Larik( numRows )
        # Create the 1-D arrays for each row of the 2-D
        # array.
        for i in range( numRows ) :
            self._theRows[i] = Larik( numCols )
```

contoh: A = Larik2D(4,5) : 4 baris 5 kolom

Implementasi Larik2D

```
# Implementasi kelas Larik2D, memakai larik dari larik.
# Larik2D.py == @Suarga
from Larik import Larik
class Larik2D :
    # menciptakan larik 2-D dengan ukuran numRows x numCols.
    def __init__( self, numRows, numCols ) :
        # Create a 1-D array to store an array reference for each row.
        self._theRows = Larik( numRows )
        # Create the 1-D arrays for each row of the 2-D array.
        for i in range( numRows ) :
            self._theRows[i] = Larik( numCols )

    # memberikan jumlah baris dari larik 2-D.
    def numRows( self ) :
        return len( self._theRows )

    # memberikan jumlah kolom dari larik 2-D.
    def numCols( self ) :
        return len( self._theRows[0] )
```

```

# memberikan jumlah kolom dari Larik 2-D.
def numCols( self ):
    return len( self._theRows[0] )

# mengisi semua sel Larik 2-D dengan nilai value.
def clear( self, value ):
    for row in range( self.numRows() ):
        self._theRows[row].clear(value)

# menampilkan isi Larik 2-D pada index [i, j]
def __getitem__( self, ndxTuple ):
    assert len(ndxTuple) == 2, "Invalid number of array subscripts."
    row = ndxTuple[0]
    col = ndxTuple[1]
    assert row >= 0 and row < self.numRows() \
        and col >= 0 and col < self.numCols(), \
        "Array subscript out of range."
    the1dArray = self._theRows[row]
    return the1dArray[col]

# mengganti isi sel Larik 2-D pada index [i,j] menjadi value.
def __setitem__( self, ndxTuple, value ):
    assert len(ndxTuple) == 2, "Invalid number of array subscripts."
    row = ndxTuple[0]
    col = ndxTuple[1]
    assert row >= 0 and row < self.numRows() \
        and col >= 0 and col < self.numCols(), \
        "Array subscript out of range."
    the1dArray = self._theRows[row]
    the1dArray[col] = value

```

```
# mengisi Larik 2-D
def isi2D(self):
    for i in range(self.numRows()):
        for j in range(self.numCols()):
            print('index [%d, %d]: ' % (i,j))
            self[i,j] = int(input(' : '))

# menampilkan semua isi Larik 2-D
def dispArr2D( self ):
    for i in range(self.numRows()):
        myRow=[]
        for j in range(self.numCols()):
            myRow.append(self[i,j])
        print(myRow)
```

Matriks

- Matrik adalah larik 2D namun ada beberapa fungsi khusus yang harus ditambahkan khusus untuk ADT dari matrik, yaitu:
- **Matriks(nrows, ncols)** : inisialisasi objek matrik dengan nrows baris dan ncols kolom
- **numRows()** : memberikan jumlah baris dari matrik
- **numCols()** : memberikan jumlah kolom dari matrik
- **getitem(row, col)** : memberikan data pada baris row dan kolom col
- **setitem(row, col)** : mengganti nilai data pada baris row dan kolom col
- **scaleBy(scalar)** : mengalikan setiap data dengan nilai scalar
- **transpose()** : melakukan transpose dari matrik
- **add(rhsMat)** : menjumlahkan matrik ini dengan matrik rhsMat
- **subtract (rhsMat)** : mengurangkan matrik ini dengan matrik rhsMat
- **multiply(rhsMat)**: mengalikan matrik ini dengan matrik rhsMat

fungsi: scaleBy

```
# Scales the matrix by the given scalar.  
def scaleBy( self, scalar ):  
    for r in range( self numRows() ) :  
        for c in range( self numCols() ) :  
            self[ r, c ] *= scalar
```

#mengalikan elemen matrik dengan bil scalar

fungsi: transpose()

```
def transpose( self ):  
    bar = self numRows()  
    kol = self numCols()  
    temp = Matrix(kol, bar)  
    for i in range(bar):  
        for j in range(kol):  
            temp[j,i] = self[i,j]  
    return temp
```

#melakukan transpose baris <==> kolom

fungsi : add()

```
def __add__( self, rhsMatrix ):  
    assert rhsMatrix numRows() == self numRows() and \  
           rhsMatrix numCols() == self numCols(), \  
           "Matrix sizes not compatible for the add \  
           operation."  
    # Create the new matrix.  
    newMatrix = Matrix( self numRows(), self numCols() )  
    # Add the corresponding elements in the two \  
    # matrices.  
    for r in range( self numRows() ) :  
        for c in range( self numCols() ) :  
            newMatrix[ r, c ] = self[ r, c ] + \  
                               rhsMatrix[ r, c ]  
    return newMatrix
```

fungsi subtract: __sub__()

```
def __sub__( self, rhsMatrix ):  
    assert rhsMatrix numRows() == self numRows() and \  
           rhsMatrix numCols() == self numCols(), \  
           "Matrix sizes not compatible for the add \  
           operation."  
    # Create the new matrix.  
    newMatrix = Matrix( self numRows(), self numCols() )  
    # Add the corresponding elements in the two  
    # matrices.  
    for r in range( self numRows() ) :  
        for c in range( self numCols() ) :  
            newMatrix[ r, c ] = self[ r, c ] - \  
                               rhsMatrix[ r, c ]  
    return newMatrix
```

fungsi multiply: __mul__()

```
def __mul__( self, rhsMatrix ):
    assert rhsMatrix numRows() == self numCols(), \
        "Matrix sizes not compatible for the \
        multiplication."
    # create the new matrix
    newMatrix = Matrix( self numRows(), \
                        rhsMatrix numCols() )
    # multiply the matrices
    for r in range( self numRows() ):
        for c in range( rhsMatrix numCols() ):
            newMatrix[r, c] = 0
            for k in range( rhsMatrix numRows() ):
                newMatrix[r, c] += self[r,k] * \
                    rhsMatrix[k, c]
    return newMatrix
```

Implementasi Matriks

```
# Implementasi kelas Matriks memakai Larik2D.
# Matriks.py == @Suarga
from Larik2D import Larik2D

class Matriks :
    # Creates a matrix of size numRows x numCols initialized to 0.
    def __init__( self, numRows, numCols ):
        self._theGrid = Larik2D( numRows, numCols )
        self._theGrid.clear( 0 )

    # Returns the number of rows in the matrix.
    def numRows( self ):
        return self._theGrid.numRows()

    # Returns the number of columns in the matrix.
    def numCols( self ):
        return self._theGrid.numCols()

    # Returns the value of element (i, j): x[i,j]
    def __getitem__( self, ndxTuple ):
        return self._theGrid[ ndxTuple[0], ndxTuple[1] ]
```

```
# Sets the value of element (i,j) to the value s: x[i,j] = s
def __setitem__( self, ndxTuple, scalar ):
    self._theGrid[ ndxTuple[0], ndxTuple[1] ] = scalar

# Scales the matrix by the given scalar.
def scaleBy( self, scalar ):
    for r in range( self.numRows() ) :
        for c in range( self.numCols() ) :
            self[ r, c ] *= scalar

# Creates and returns a new matrix that is the transpose of this matrix.
# added by Suarga
def tranpose( self ):
    bar = self.numRows()
    kol = self.numCols()
    temp = Matriks(kol, bar)
    for i in range(bar):
        for j in range(kol):
            temp[j,i] = self[i,j]
    return temp
```

```
# Creates and returns a new matrix that results from matrix addition.
def __add__( self, rhsMatrix ):
    assert rhsMatrix numRows() == self numRows() and \
        rhsMatrix numCols() == self numCols(), \
        "Matrix sizes not compatible for the add operation."
    # Create the new matrix.
    newMatriks = Matriks( self numRows(), self numCols() )
    # Add the corresponding elements in the two matrices.
    for r in range( self numRows() ) :
        for c in range( self numCols() ) :
            newMatriks[ r, c ] = self[ r, c ] + rhsMatrix[ r, c ]
    return newMatriks

# Creates and returns a new matrix that results from matrix subtraction.
# completed by Suarga
def __sub__( self, rhsMatrix ):
    assert rhsMatrix numRows() == self numRows() and \
        rhsMatrix numCols() == self numCols(), \
        "Matrix sizes not compatible for the add operation."
    # Create the new matrix.
    newMatrix = Matriks( self numRows(), self numCols() )
    # Add the corresponding elements in the two matrices.
    for r in range( self numRows() ) :
        for c in range( self numCols() ) :
            newMatrix[ r, c ] = self[ r, c ] - rhsMatrix[ r, c ]
    return newMatrix
```

```

# Creates and returns a new matrix resulting from matrix multiplication.
# added by Suarga
def __mul__( self, rhsMatrix ):
    assert rhsMatrix.numRows() == self.numCols(), \
        "Matrix sizes not compatible for the multiplication."
    # create the new matrix
    newMatrix = Matriks( self.numRows(), rhsMatrix.numCols() )
    # multiply the matrices
    for r in range( self.numRows() ):
        for c in range( rhsMatrix.numCols() ):
            newMatrix[r, c] = 0
            for k in range( rhsMatrix.numRows() ):
                newMatrix[r, c] += self[r,k] * rhsMatrix[k, c]
    return newMatrix

# display matrix
def dispMatriks( self ):
    for i in range(self.numRows()):
        myRow=[]
        for j in range(self.numCols()):
            myRow.append(self[i,j])
        print(myRow)

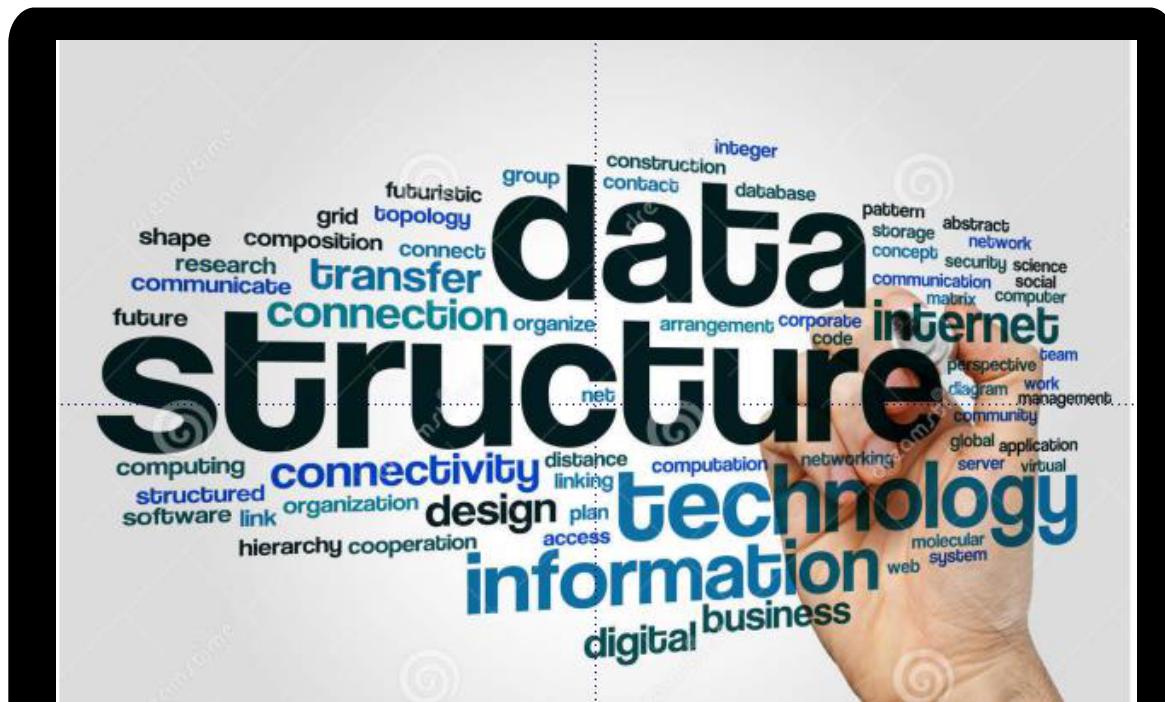
# isi matriks
def isiMatriks( self ):
    for i in range(self.numRows()):
        for j in range(self.numCols()):
            print('index [%d, %d]: ' % (i,j))
            self[i,j] = int(input(' : '))

```

```
>>> A=Matriks(3,3)          >>> B[0,0]=9  
>>> A[0,0]=1                >>> B[0,1]=8  
>>> A[0,1]=2                >>> B[0,2]=7  
>>> A[0,2]=3                >>> B[1,0]=6  
>>> A[1,0]=4                >>> B[1,1]=5  
>>> A[1,1]=5                >>> B[1,2]=4  
>>> A[1,2]=6                >>> B[2,0]=3  
>>> A[2,0]=7                >>> B[2,1]=2  
>>> A[2,1]=8                >>> B[2,2]=1  
>>> A[2,2]=9                >>> C=A+B  
>>> A.dispMatriks()        >>> C.dispMatriks()  
[1, 2, 3]                  [10, 10, 10]  
[4, 5, 6]                  [10, 10, 10]  
[7, 8, 9]                  [10, 10, 10]  
>>> B=Matriks(3,3)
```

```
>>> D=B-A  
>>> D.dispMatriks()  
[8, 6, 4]  
[2, 0, -2]  
[-4, -6, -8]  
>>> X=A.transpose()  
>>> X.dispMatriks()  
[1, 4, 7]  
[2, 5, 8]  
[3, 6, 9]  
>>> E=A*B  
>>> E.dispMatriks()  
[30, 24, 18]  
[84, 69, 54]  
[138, 114, 90]  
>>>
```

SELESAI



STRUKTUR DATA (PYTHON)

“Instruksi Utama Python”

[@SUARGA] | [Pertemuan 03]

OutLine





Instruksi Utama Python

- 1. Instruksi Seleksi:** digunakan untuk memilih instruksi yang akan di-eksekusi selanjutnya berdasarkan pada suatu syarat / kondisi. Bilamana syarat dipenuhi maka instruksi di-laksanakan, bila tidak maka hal lain dilakukan
- 2. Instruksi Perulangan:** digunakan untuk mengulang serangkaian instruksi pada perulangan yang terbatas atau tidak terbatas, namun dapat dihentikan berdasarkan suatu kondisi.
- 3. Pemakaian File :** instruksi untuk membaca file dan merekam ke file
- 4. Pembuatan Fungsi :** Pembuatan Fungsi merupakan bagian penting dari pemrograman Python

Instruksi Seleksi if/else

- Instruksi **if** digunakan untuk memilih instruksi yang akan dilaksanakan berdasarkan suatu syarat atau relasi, bentuknya sebagai berikut:

if (relasi) :

true statements #kerjakan bila benar

else:

false statements #kerjakan bila salah

- Contoh apabila anda ingin membuat pilihan antara menampilkan nilai $(x+10)$ bila x bernilai kurang dari 5, dengan nilai $(x*10)$ bila x bernilai lebih dari 5, maka secara algoritmik anda menuliskan:

Algoritma:

Bila $x < 5$:

maka tampilkan $(x + 10)$

selain itu tampilkan $(x * 10)$.

Python:

```
if ( x < 5 ) :
```

```
    print( x + 10)
```

```
else:
```

```
    print( x * 10)
```

pemilihan multi-syarat

- Instruksi seleksi dapat dibuat multi syarat/relasi, dengan memakai *elif* diantaranya, bentuknya sebagai berikut.

```
if (relasi-1):
    statement-1
elif (relasi-2):
    statement-2
elif (relasi-3):
    statement-3
else:
    statement-4
```

contoh

```
x = int(input('Masukkan satu angka : '))
if x < 0 :
    x = 0
    print('Angka negatif dijadikan nol!!!')
elif x == 0:
    print('Anda memasukkan angka nol!')
else :
    print('Angka positif ',x)
```

Contoh eksekusi :

```
>>>
Masukkan satu angka : 10
Angka positif 10
>>>
Masukkan satu angka : 0
Anda memasukkan angka nol!
>>>
```

Contoh

- Buat sebuah program yang menerima masukkan berupa koefisien dari persamaan kuadrat $ax^2 + bx + c = 0$, kemudian menampilkan dua akar dari persamaan tersebut.
- Algoritma:

Masukkan koefisien a, b, dan c

Hitung diskriminan $D = b^2 - 4ac$

Bila $D > 0$ maka: # akar riil

 Hitung akar dari D, $D = \sqrt{D}$

$x_1 = (-b + D)/(2a)$

$x_2 = (-b - D)/(2a)$

Bila $D < 0$ maka: # akar kompleks

 Hitung akar dari $-D$, $D = \sqrt{-D}$

$x_1 = -b/2a + D/2a j$

$x_2 = -b/2a - D/2a j$

Bila $D=0$ maka: # akar kembar

$x_1 = -b/2a$

$x_2 = -b/2a$

Tampilkan x_1, x_2

```
#akar.py - mencari akar pers ax**2 + bx + c
import math
a = float(input('Masukkan koefisien a : '))
b = float(input('Masukkan koefisien b : '))
c = float(input('Masukkan koefisien c : '))

D = b**2 - 4*a*c
if (D > 0):          #akar riel
    D = math.sqrt(D)
    x1 = (-b + D)/(2*a)
    x2 = (-b - D)/(2*a)
elif (D < 0):         #akar complex
    D = math.sqrt(-D)
    x1 = (-b/2*a) + (D/2*a)*1j
    x2 = (-b/2*a) - (D/2*a)*1j
else:                  #akar kembar
    x1 = -b/(2*a)
    x2 = -b/(2*a)

print('Akar persamaan kuadrat:')
print('x1 = ', x1)
print('x2 = ', x2)
```

>>>

Masukkan koefisien a : 1

Masukkan koefisien b : 10

Masukkan koefisien c : 30

Akar persamaan kuadrat:

$x_1 = (-5+2.23606797749979j)$

$x_2 = (-5-2.23606797749979j)$

>>>

Masukkan koefisien a : 2

Masukkan koefisien b : 10

Masukkan koefisien c : 12

Akar persamaan kuadrat:

$x_1 = -2.0$

$x_2 = -3.0$

>>>

Perulangan for

- Python memiliki bentuk for yang unik, dibanding dengan for pada bahasa C/C++ atau Java. For pada python berbasis pada struktur data “list”, bentuk pertama adalah:

```
for x in listvar:
    do-something
```

dimana listvar adalah variabel bertipe list (list akan dibicarakan pada bagian lain). Instruksi ini sepadan dengan kalimat algoritma berikut ini:

Untuk setiap nilai x dalam listvar, kerjakan: do-something

- Bentuk kedua:

```
for i in range(intvar):
```

```
    do-something
```

dimana intvar adalah variabel bertipe integer. Instruksi ini sepadan dengan kalimat algoritma berikut ini:

Untuk setiap nilai i mulai dari 0 hingga (intvar-1),
kerjakan do-something

Berarti *do-something* akan dikerjakan berulang sebanyak (invar) kali, misalnya bila intvar = 10 maka akan diulangi 10 kali dengan nilai i = 0, 1, 2, ..., 9.

contoh

```
# bentuk pertama
```

```
a =['pintu', 'jendela', 'kusen',  
'teralis']
```

```
for x in a:
```

```
    print (x, len(x))
```

- pintu 5
- jendela 7
- kusen 5
- teralis 7
-
- i 2i i^2
- 0 0 0
- 1 2 1
- 2 4 4
- 3 6 9
- 4 8 16
- 5 10 25
- 6 12 36

```
print()
```

```
print('i \t 2i \t  $i^2$ ')
```

```
# bentuk kedua
```

```
for i in range(len(x)):
```

```
    print(i,' \t', 2*i, ' \t', i**2)
```

Hasil-nya:

>>>

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

hasilnya adalah : 0 1 2 3 4 5 6 7 8 9

```
b = ['guru', 'mengajar', 'pelajaran', 'budi', 'pekerja']
for k in range(len(b)):
    print (k, b[k])
```

hasilnya adalah:

guru

mengajar

pelajaran

budi

pekerja

- Instruksi for dari python juga memiliki option **else**, yang bisa melaksanakan instruksi ketika perulangan for berakhir, atau diakhiri dengan **break**.

```
for n in range(2,10):      # n bernilai 2 s/d 9
    for x in range(2,n):    # x bernilai 2 s/d n
        if ( n % x) == 0:  # bila n habis dibagi oleh x
            print (n, ' = ', x, '*', n/x)
            break    # loop x berakhir
        else:          # keluar loop x, else dari for
            print (n, ' adalah bilangan prima')
```

Hasilnya adalah:

>>>

2 adalah bilangan prima

3 adalah bilangan prima

4 = 2 * 2.0

5 adalah bilangan prima

6 = 2 * 3.0

7 adalah bilangan prima

8 = 2 * 4.0

9 = 3 * 3.0

>>>

Perulangan while

- Selain dengan instruksi for, serangkaian instruksi dapat diulang memakai instruksi while, bentuknya:

While (relasi) :
do-something

Instruksi tersebut sepadan dengan kalimat berikut:

Selama (relasi) ini masih benar maka lakukan: do-something

Contoh Program

Disain program untuk menebak angka dengan interface dialog sebagai berikut:

Halo kawan, siapa nama anda?

Eko

Ok, Eko saya memikirkan satu angka antara 1 s/d 20
Coba anda tebak,

10

Angka tersebut lebih kecil dari 10

Tebak lagi,

2

Angka tersebut lebih besar dari 2

Tebak lagi,

4

Bagus, anda telah menebaknya dalam 3 langkah.

```
# program tebakan angka
#tebakan.py
import random          # import pustaka fungsi acak
maxi = 6                # jumlah tebakan maksimum

jumTebakan = 0
print('Hallo kawan, siapa nama anda?')
nama = input()
angka = random.randint(1, 20) # menciptakan angka acak antara 1 s/d 20

print('Ok ' + nama + ' saya memikirkan satu angka antara 1 s/d 20')
while (jumTebakan < maxi):
    print('Coba anda tebak, ')
    tebak = input()
    tebak = int(tebak)
    jumTebakan = jumTebakan + 1
    if (tebak < angka) :
        print('Angka tersebut lebih besar dari ' + str(tebak))
    if (tebak > angka):
        print('Angka tersebut lebih kecil dari ' + str(tebak))
    if (tebak == angka):
        break

if (tebak == angka):
    print ('Bagus, anda telah menebaknya dalam ' + str(jumTebakan) \
          + ' langkah')
if (tebak != angka):
    print ('Maaf, anda tidak berhasil menebak angka ' + str(angka))
```

Contoh Run :

>>>

Hallo kawan, siapa nama anda?

EKO

Ok EKO saya memikirkan satu angka antara 1 s/d 20

Coba anda tebak,

4

Angka tersebut lebih besar dari 4

Coba anda tebak,

20

Angka tersebut lebih kecil dari 20
Coba anda tebak,
12

Angka tersebut lebih kecil dari 12
Coba anda tebak,
8

Angka tersebut lebih kecil dari 8
Coba anda tebak,

6
Angka tersebut lebih kecil dari 6
Coba anda tebak,

5
Bagus, anda telah menebaknya dalam 6 langkah

Menangani Error

- Program sering kali menimbulkan error baik karena adanya kesalahan logik program, maupun akibat yang lain. Error yang diperkirakan dapat terjadi dalam program dapat ditangani dengan perintah **raise** suatu kesalahan, misalnya:

```
def sqrt(x):  
    if not isinstance(x, (int, float)):  
        raise TypeError( 'x harus numerik' )  
    elif x < 0:  
        raise ValueError( 'x harus positif' )  
    else:  
        return(math.sqrt(x))
```

- Pada contoh fungsi `sqrt(x)`, yaitu mencari akar kuadrat dari bilangan x , maka ada dua kemungkinan error, pertama x harus numerik (`float`) dan kedua x harus positif. Oleh sebab itu, pada fungsi ini diperkirakan kemungkinan dua macam error yaitu: kesalahan tipe-data (`type error`) dan kesalahan nilai data (`value error`).
- Perlu diperhatikan bahwa `TypeError` dan `ValueError` merupakan error standard yang disediakan oleh Python, jenis error yang lain disajikan dalam tabel berikut ini.

Class	Description
Exception	A base class for most error types
AttributeError	Raised by syntax <code>obj.foo</code> , if <code>obj</code> has no member named <code>foo</code>
EOFError	Raised if “end of file” reached for console or file input
IOError	Raised upon failure of I/O operation (e.g., opening file)
IndexError	Raised if index to sequence is out of bounds
KeyError	Raised if nonexistent key requested for set or dictionary
KeyboardInterrupt	Raised if user types ctrl-C while program is executing
NameError	Raised if nonexistent identifier used
StopIteration	Raised by <code>next(iterator)</code> if no element; see Section 1.8
TypeError	Raised when wrong type of parameter is sent to a function
ValueError	Raised when parameter has invalid value (e.g., <code>sqrt(-5)</code>)
ZeroDivisionError	Raised when any division operator used with 0 as divisor

Jenis-jenis error Python

try ... except

- Cara yang lain adalah dengan memakai pasangan instruksi **try ... except** misalnya pada pembacaan file berikut ini:

```
try :  
    fp = open( 'sample.txt' )  
except IOError as e :  
    print( 'tidak dapat membuka file: ', str(e) )
```

- Berarti file ‘sample.txt’ akan coba dibuka, bila tidak bisa, akan tampil error (‘tidak dapat membuka file’).

Kuiz

- Buat sebuah program yang meminta user memasukkan angka bulat X, kemudian menghitung S1 sebagai jumlah semua bilangan ganjil antara 1 dan X, dan S2 sebagai jumlah semua bilangan genap antara 1 dan X.

Definisi Fungsi

- Suatu fungsi dalam python didefinisikan melalui keyword “**def**”, diikuti oleh nama fungsi dan argumen-nya. Sebagai contoh akan dibuat fungsi yang menghitung nilai suku ke-n bilangan Fibonacci, dari suatu deret Fibonacci. Deret Fibonacci hingga suku ke-10 adalah sebagai berikut:

0 1 1 2 3 5 8 13 21 35

- Terlihat bahwa dua nilai awal adalah 0 dan 1, kemudian nilai berikutnya adalah $1 = 0 + 1$, lalu $2 = 1+1$, kemudian $3 = 1+2$, dan seterusnya. Andaikan nilai awal adalah $a=0$, dan $b=1$, kemudian nilai berikutnya adalah $a + b$, kemudian nilai a , b digeser ke depan, yaitu b digeser ke a , dan $(a+b)$ digeser ke b , sehingga dapat dibuat suatu fungsi fibo(n) sebagai berikut:

```
def fibo(n):
    # menampilkan deret fibonacci sampai nilainya <= n
    a, b = 0, 1
    while (b < n):
        print (b),
        a, b = b, a+b
```

setelah di-save dengan nama **fibo.py**, maka dapat dicoba dengan memanggil namanya disertai parameter, misalnya untuk menampilkan bilangan Fibonacci hingga maksimum bernilai 2000,

```
>>> fibo(2000)
```

```
1  
1  
2  
3  
5  
8  
13  
21  
34  
55  
89  
144  
233  
377  
610  
987  
1597
```

27 Agustus 2022
>>>

```
def fibo2(n):
    # menampilkan deret fibonacci sampai
    # nilainya <=n.
    # hasil ditampilkan horisontal
    hasil = []
    a, b = 0, 1
    while (b < n):
        hasil.append(b)
        a, b = b, a+b
    return hasil
```

- Agar hasilnya bisa ditampilkan lakukan sebagai berikut.

```
>>> x=fibo2(2000)
>>> x
[1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597]
```

```
>>>
```

Akses File Teks

- Membaca File Teks:

Membaca suatu file dapat dilakukan dengan urutan instruksi sebagai berikut:

1. Menyatakan variable nama file : `infile = "nama-file"`
2. Membuka file untuk dibaca : `infh = open(infile, mode)`
- 3 . Membaca baris-demi-baris : `line = infh.readline()`
 `while line:`
 `# proses baris data`
 `line = infh.readline()`
- 4 . Menutup file : `infh.close()`

dimana mode-file untuk membaca adalah:

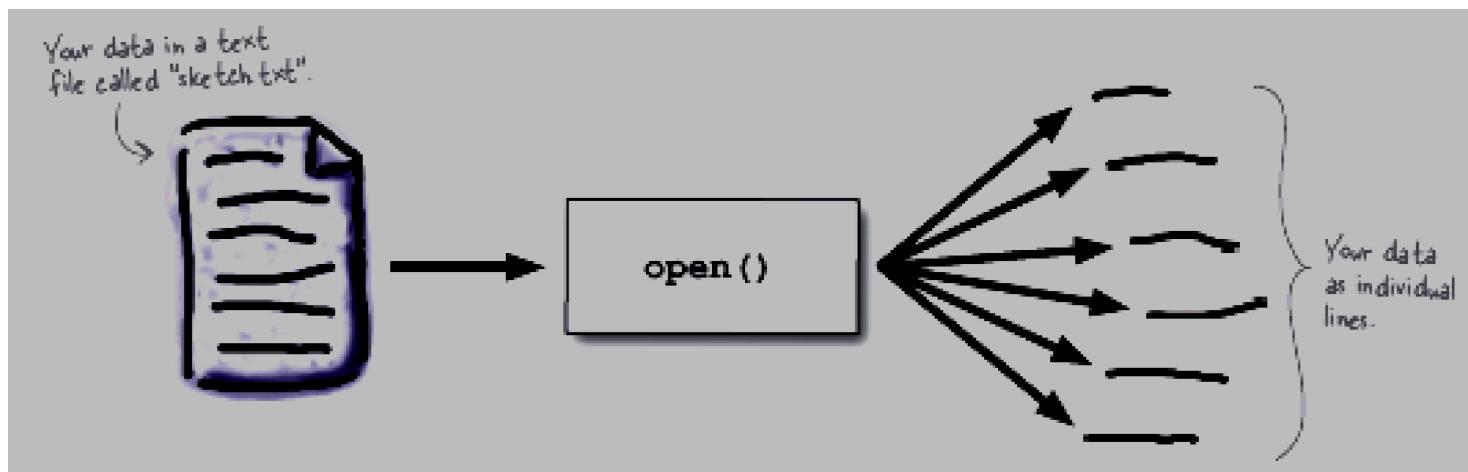
“r” – membaca file text

“r+” – membaca sekaligus bisa merekam

“rb” – membaca file binary

Contoh baca file

```
path="D:\\USER\\Python\\sample.txt"
myfile = open(path, 'r')
for line in (myfile):
    print(line, end='')
myfile.close()
```



Merekam Teks ke File

- Prosedur-nya

1. Menyatakan variable nama file : outfile = “nama-file”
2. Membuka file untuk diisi : outfh = open(outfile, mode)
3. Rekam data baris-demi-baris : outfh.write('...data...')
4. Menutup file : outfh.close()

Modus perekam ke file adalah:

- ‘w’ : merekam ke file, selalu mulai dari awal
- ‘w+’ : merekam ke file, mulai dari awal, dan juga membaca
- ‘a’ : merekam ke file, mulai dari akhir file
- ‘a+’ : menambah isi file, dan dapat dibaca

Contoh merekam ke File

```
path="D:\\USER\\Python\\sample.txt"
myfile = open(path, 'a')
print('Ketik kalimat yang akan direkam')
lagi=True
while lagi:
    print('Ketik stop bila selesai')
    baris = input()
    if baris == 'stop':
        lagi = False
    else:
        baris = baris + '\n'
        myfile.write(baris)

myfile.close()
```

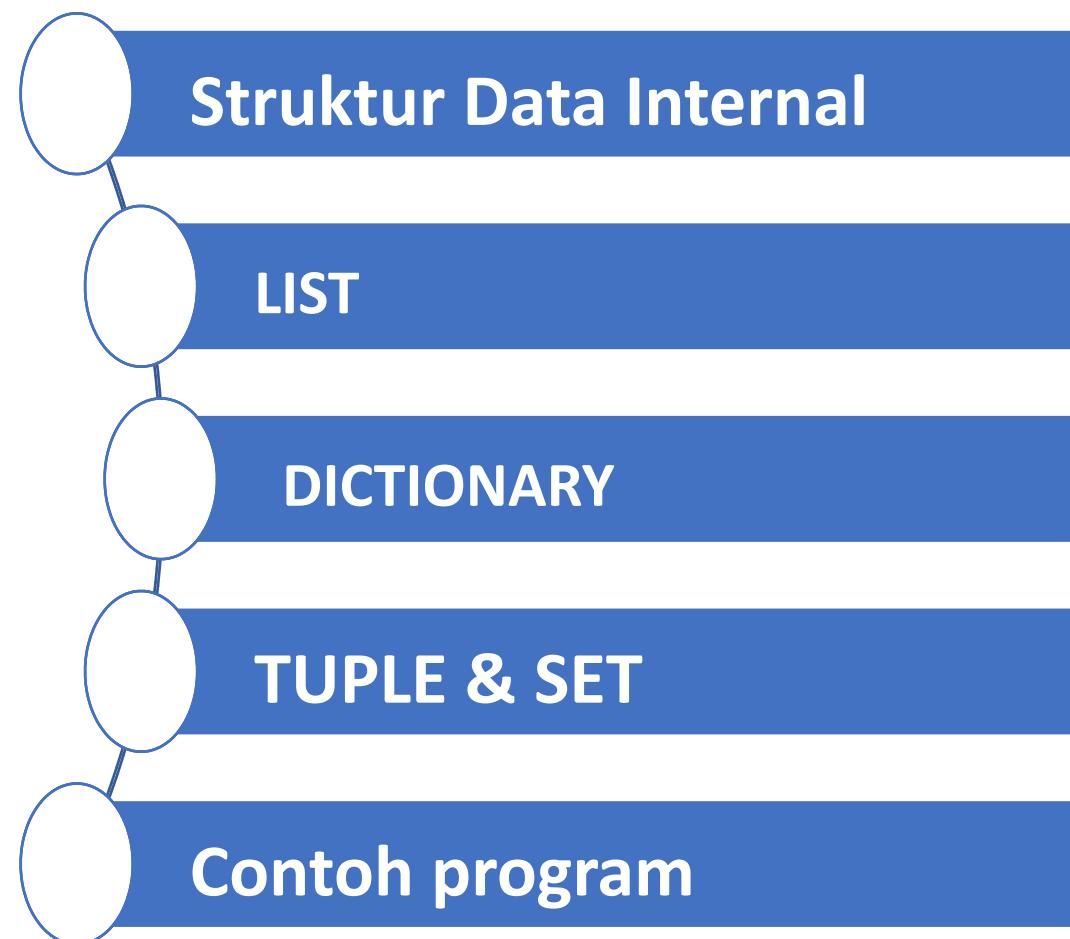


STRUKTUR DATA (PYTHON)

“Struktur Data Internal Python”

[@SUARGA | [Pertemuan 05]

OutLine



Struktur Internal Data Python

- Python memiliki empat macam struktur data yang disajikan dalam format objek, yaitu: **List**, **Tuple**, **Dictionary** dan **Set**
- List, Tuple dan Dictionary mirip dengan larik
- List dapat dibuat langsung dengan menggunakan tanda kurung kotak []
- Tuple dapat dibuat langsung dengan menggunakan tanda kurung bulat ()
- Dictionary dapat dibuat langsung dengan menggunakan tanda kurung keriting { }

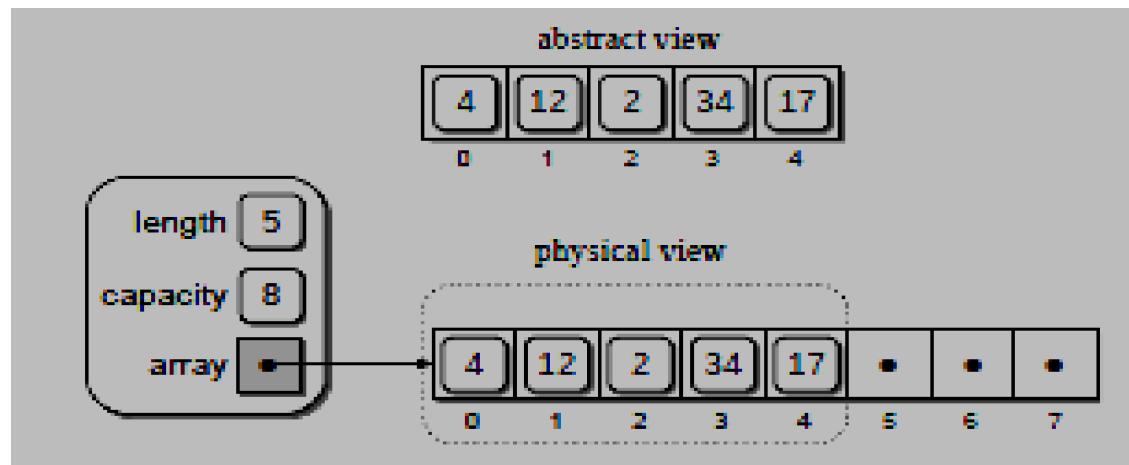
List

- Bahasa Python memiliki satu struktur data yang diberi nama “list” (sepintas telah dibicarakan pada modul-2), yang pada hakekatnya sama dengan struktur larik, namun secara internal struktur list merupakan “dynamic array”, larik yang size-nya bisa berkembang sesuai kebutuhan. Sebagai contoh pemakaian instruksi:
- `a = [0, 0, 0, 0, 0]`
- menciptakan sebuah list bernama **a** dengan lima elemen semuanya bernilai nol.

- Elemen-elemen pada python list bisa diakses dengan suatu indeks, indeks dimulai dari 0 (nol), sehingga kelima elemen list a diatas masing-masing adalah: $a[0]$, $a[1]$, ..., $a[4]$.
- python juga membolehkan indeks negatif (tidak diperkenankan oleh bahasa program lain seperti C dan Java), namun indeks -1 harus diberikan kepada elemen yang paling kanan, dengan demikian list a diatas bisa diakses melalui indeks: $a[-5]$, $a[-4]$, ..., $a[-1]$.

Abstraksi Memory LIST

- definisi struktur list seperti:
 $\text{pyList} = [4, 12, 2, 34, 17]$
- pada hakikatnya memiliki abstraksi memory sebagai berikut.



operator dan list

Operator + menyambung dua list:

```
>>> a = [1, 2, 3]
>>> b = [4, 5, 6]
>>> c = a + b
>>> print(c)
[1, 2, 3, 4, 5, 6]
```

Operator * akan mengulang isi list

```
>>> [0] * 4
[0, 0, 0, 0]
>>> [1, 2, 3] * 3
[1, 2, 3, 1, 2, 3, 1, 2, 3]
```

- Operator slice (yaitu a:b) bekerja pada list

```
>>> t = ['a', 'b', 'c', 'd', 'e', 'f']
```

```
>>> t[1:3] #index 1 hingga sebelum index 3  
['b', 'c']
```

```
>>> t[:4] #index 0 hingga sebelum index 4  
['a', 'b', 'c', 'd']
```

```
>>> t[2:] #index 2 hingga akhir  
['c', 'd', 'e', 'f']
```

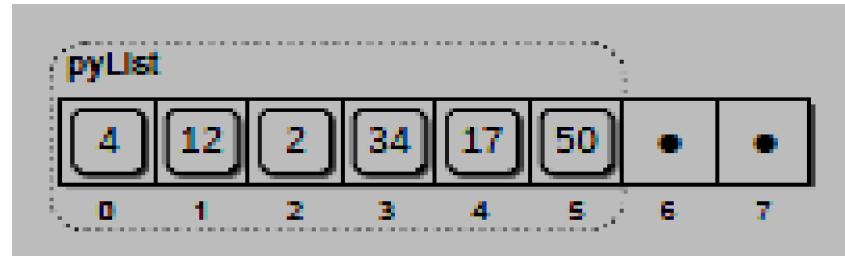
```
>>>
```

list ADT

- **append(x)**: menambahkan x diakhir list
- **extend(L)**: melakukan ekstensi list dengan menggandeng isi list L
- **insert(i,x)** : menyisipkan x dalam list pada index i
- **remove(x)** : menghapus elemen pertama dari list bernilai x, error bila x tidak ada
- **pop([i])** : menghapus elemen list pada posisi i
- **index(x)** : mencari index elemen bernilai x
- **count(x)** : mencacah elemen bernilai x
- **sort()** : men-sort elemen dalam list, ascending
- **reverse()** : mengurutkan terbalik elemen dalam list

- Contoh pemakain ADT List:

```
>>> pyList.append(50)
```



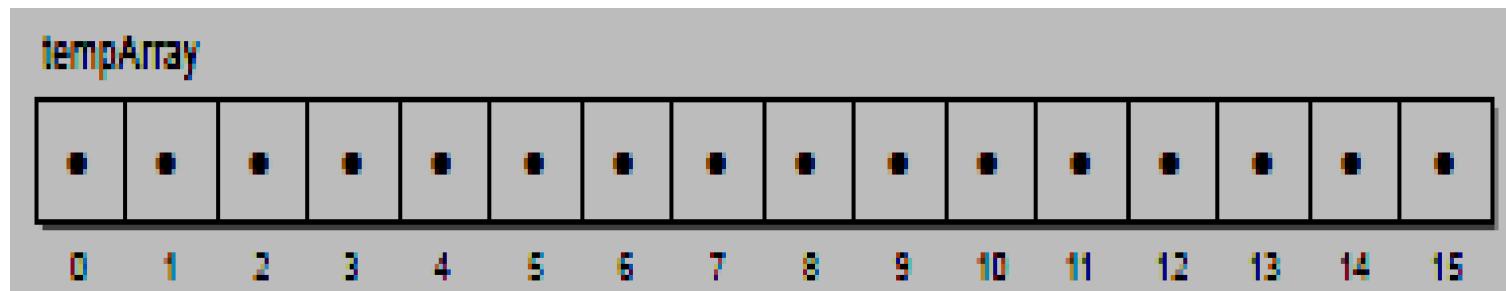
```
>>> pyList.append(18)
```

```
>>> pyList.append(64)
```

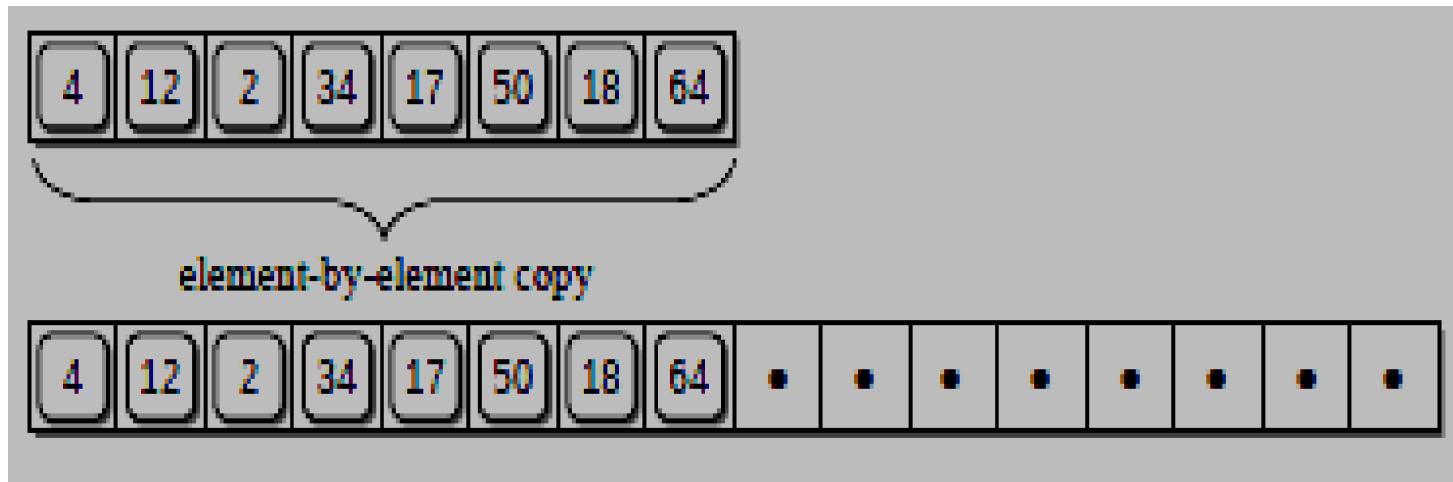
```
>>> pyList.append(6)
```



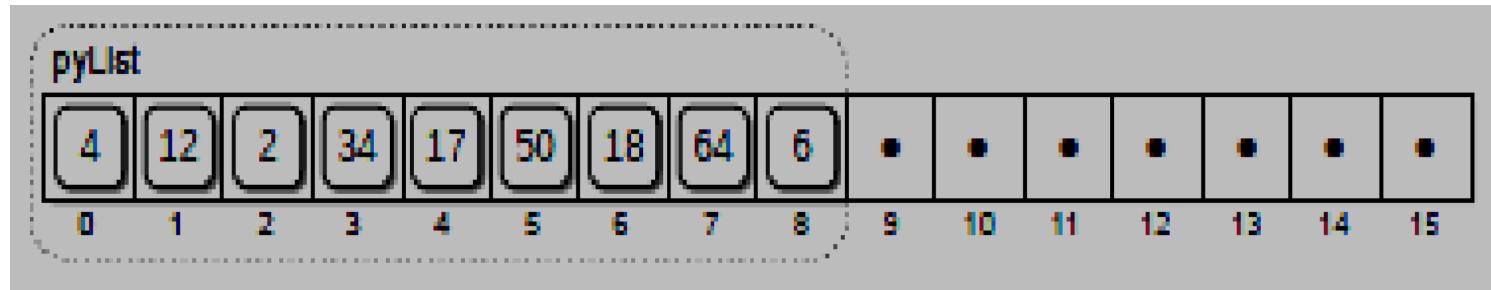
- Sebenarnya array pyList hanya tersisa dua lokasi, ketika 18 dan 64 dimasukkan maka array telah penuh.
 - Namun karena Python list adalah *array dinamis* maka secara otomatis ruang untuk data pada pyList akan diperbesar dua kali lipat oleh sistem python, ketika append(6) berlangsung, proses-nya berjalan sebagai berikut.
- (a) Sebuah array baru dengan kapasitas 2 kali dibentuk.



(b) Isi array lama di-salin ke array baru.



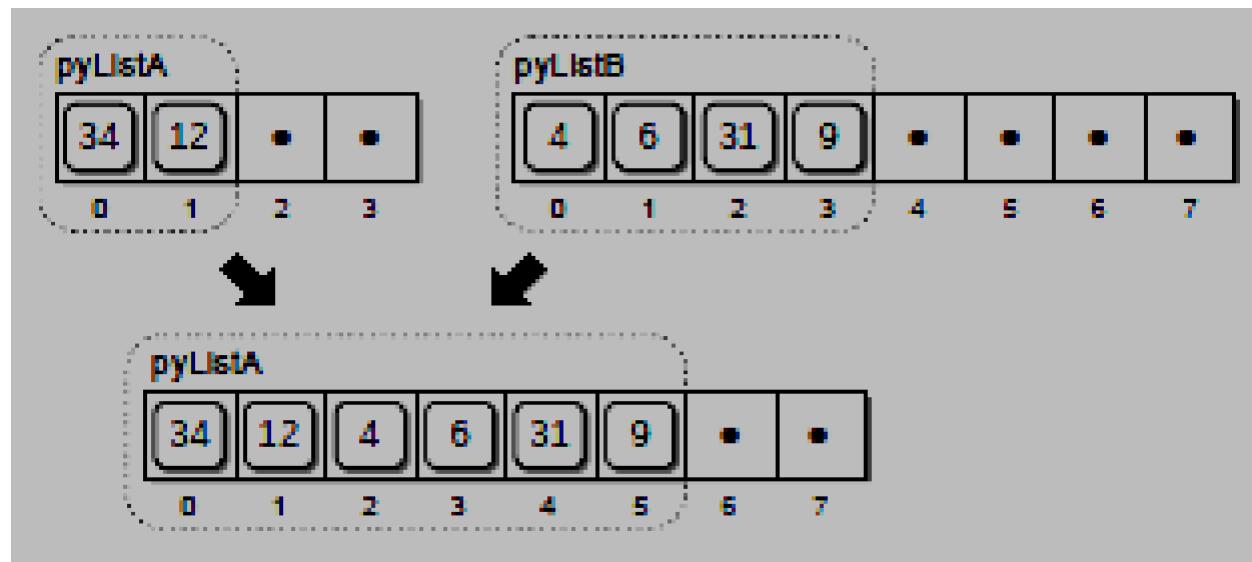
(c) nilai 6 dimasukkan ke posisi 8



```
>>> pyListA = [34, 12]  
>>> pyListB = [4, 6, 31, 9]  
>>> pyListA.extend(pyListB)  
>>> pyListA
```

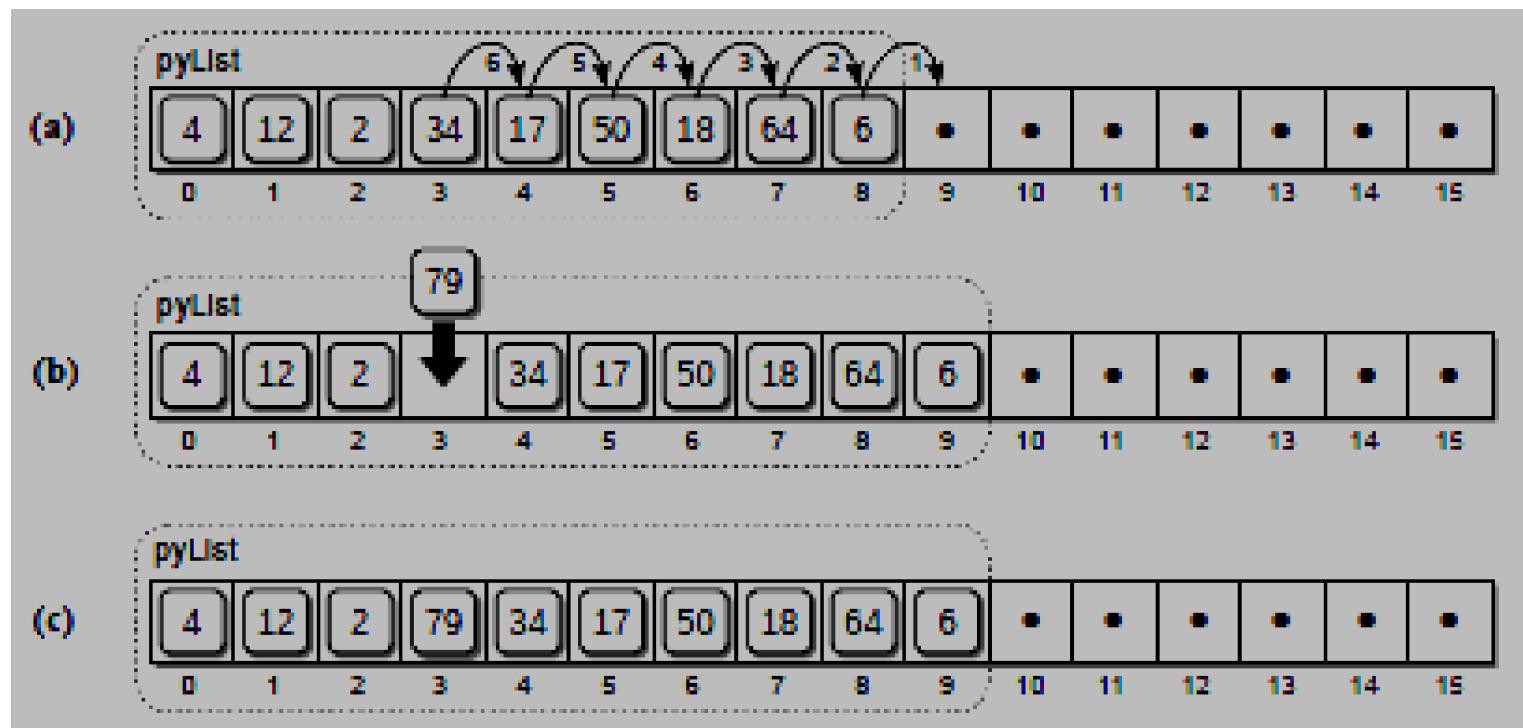
```
[34, 12, 4, 6, 31, 9]
```

```
>>>
```



```
>>> pyList.insert(3, 79) # menyisipkan 79 pada index 3
```

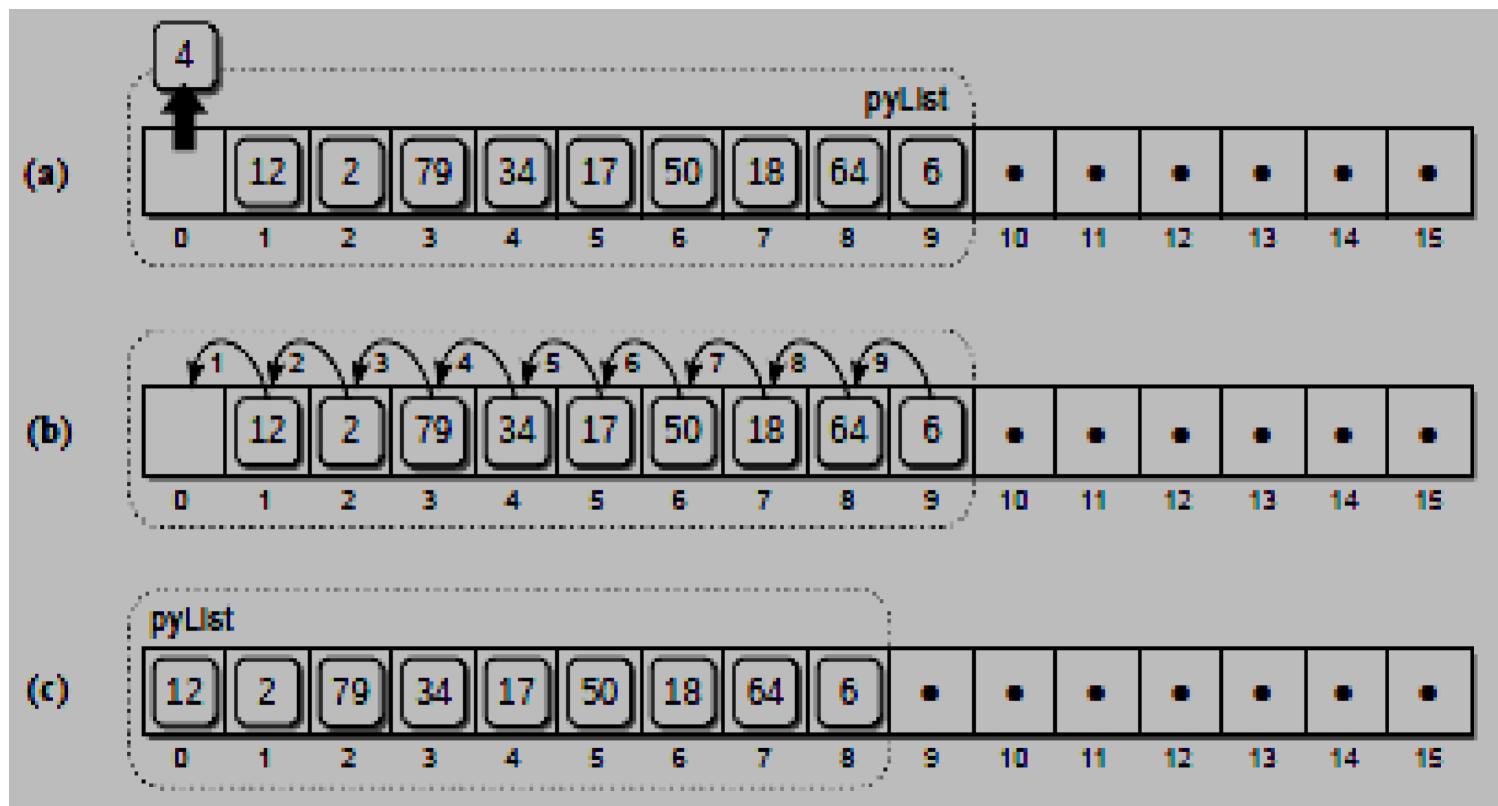
Penyisipan akan mengakibatkan penggeseran data kekanan agar data baru bisa disisipkan.



```
>>> pyList.pop(0) # ambil elemen pertama
```

4

Setelah elemen pertama di keluarkan maka elemen lain digeser ke kiri secara otomatis.



```
>>> pyList.pop()    # ambil elemen terakhir  
6  
>>> a=[66.6, 333, 1, 1234.5, 333, 66.6]  
>>> a.count(333), a.count(66.6), a.count('x')  
(2, 2, 0)  
>>> a.remove(333)  
>>> a  
[66.6, 1, 1234.5, 333, 66.6]  
>>> a.sort()  
>>> a  
[1, 66.6, 66.6, 333, 1234.5]  
>>> a.reverse()  
>>> a  
[1234.5, 333, 66.6, 66.6, 1]  
>>>
```

```
>>> t = ['a', 'b', 'c']
>>> del t[1]
>>> print t
['a', 'c']
```

```
>>> t = ['a', 'b', 'c', 'd', 'e', 'f']
>>> del t[1:5]
>>> print t
['a', 'f']
```

```
>>> s = 'spam'
>>> t = list(s)
>>> print t
['s', 'p', 'a', 'm']
```

```
>>> s = 'pining for the fjords'  
>>> t = s.split()  
>>> print t  
['pining', 'for', 'the', 'fjords']
```

```
>>> t = ['pining', 'for', 'the', 'fjords']  
>>> delimiter = ' '  
>>> delimiter.join(t)  
'pining for the fjords'
```

```
>>> s = 'spam-spam-spam'  
>>> delimiter = '-'  
>>> s.split(delimiter)  
['spam', 'spam', 'spam']
```

Dictionary

- Struktur lain dalam Python yang mirip dengan list adalah “dictionary”. Pada struktur dictionary, ada dua jenis elemen yaitu: key dan value. Key merupakan index untuk membaca value, susunan elemennya sebagai berikut,
- `dict = {kunci : value, kunci : value, }`

misalnya:

- `eng2sp = {'one' : 'uno', 'two' : 'dos', 'three' : 'tres'}`

- Dictionary kosong dapat diciptakan dengan keyword dict().
- Perhatikan contoh prosedur/fungsi berikut, dimana akan dibentuk suatu dictionary d yang berisi frekuensi huruf dalam kalimat s,

```
def histogram(s):  
    d = dict()  
    for c in s:  
        if c not in d:  
            d[c]=1  
        else:  
            d[c] += 1  
    return d
```

```
h=histogram('dinosaurus')
print(h)
```

```
{'a': 1, 'd': 1, 'i': 1, 'o': 1, 'n': 1, 's': 2,
'r': 1, 'u': 2}
```

Fungsi histogram diatas membentuk suatu dictionary dimana key adalah huruf-huruf dari “dinosaurus” dan value-nya adalah frekuensi huruf-nya, misalnya

```
>>> print(h['a'])
```

```
1
```

```
>>> print(h['s'])
```

```
2
```

- Kapan struktur dictionary perlu digunakan?
 - ketika diperlukan assosiasi antara kunci (key) dan nilai (value)
 - ketika suatu nilai perlu diakses cepat berdasarkan kunci
 - ketika data sering berubah (mutable)

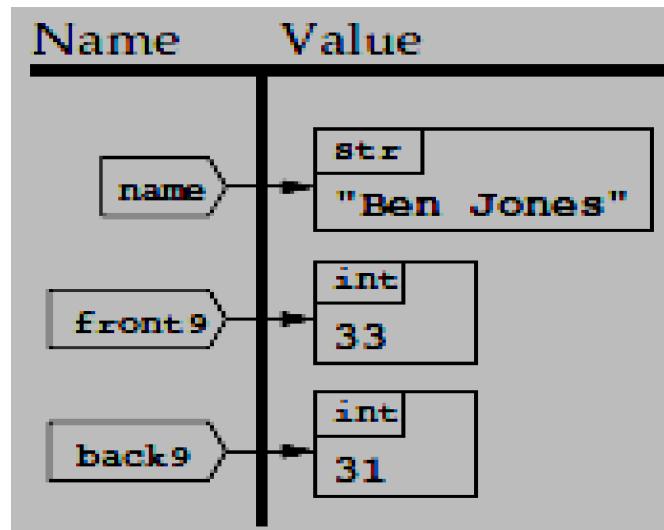
dictionary ADT

- **Beberapa method/fungsi untuk dictionary adalah:**
- **has_key(k)** : memeriksa apakah kunci k ada dalam dictionary
- **fromkeys(S,v)** : bentuk dict baru mulai dari kunci S bernilai v
- **clear()** : menghapus semua isi dictionary
- **keys()** : menampilkan semua kunci dalam dictionary
- **values()** : menampilkan semua nilai-nya
- **items()** : menampilkan semua pasangan key dan nilai
- **get(k)** : memperoleh nilai suatu kunci k, None bila tidak ada
- **pop(k)** : mengambil item dgn kunci k, dan memberikan nilainya
- **popitem()** : mengambil pasangan kunci dan nilai dari dict
- **update(d2)** : menyambung dictionary dengan dictionary d2

Abstraksi Memory dictionary

- Gambaran struktur data dari dictionary adalah sebagai berikut.

`d = { 'name' : 'Ben Jones', 'front9' : 33, 'back9' : 31 }`



Tuple

- Satu struktur dasar yang memiliki basis array yang lain adalah “tuple”, mirip list tetapi isi-nya tidak bisa di-ubah (immutable) sehingga cocok digunakan sebagai daftar key dalam dictionary.
- Membuat tuple menggunakan kurung biasa (), misalnya sebagai berikut:
- `t = ('a', 'b', 'c', 'd', 'e')`
- elemen-nya bisa diakses memakai indeks, misalnya:

```
print(t[0])
```

```
'a'
```

Karena bersifat immutable maka elemen-nya tidak bisa diubah, misalnya:

```
t[0] = 'A'
```

```
TypeError : object doesn't support item assignment
```

tuple ADT

- ADT dari tuple antara lain memiliki fungsi:
- **count()** : menghitung banyaknya data dalam tuple
- **index(v)** : mencari index dari data bernilai v, error bila v tidak ada
- **len()** : panjang dari tuple
- **iter(x)** : menciptakan objek yang bisa di-iterasi dari tuple
- **in()** : memeriksa apakah suatu nilai ada dalam tuple
- Selain itu objek tuple bisa dijumlahkan (+), dikalikan dengan skalar (*), dan membandingkan dua tuple dengan operasi logik ($>$, $<$, $==$, $>=$, $<=$, $!=$).

Struktur Set

- Set atau himpunan adalah wadah untuk koleksi data yang tidak boleh berulang, atau tidak boleh ada duplikasi nilai, semua elemen dalam set harus unik. Data dalam set berada diantara tanda kurung { .. }

Berikut ini adalah contoh menciptakan set:

```
>>> a=[1, 2, 3]      #buat sebuah list a
>>> s=set(a)        #kemudian dijadikan set s
>>> a
[1, 2, 3]
>>> s
{1, 2, 3}
```

ADT dari Set

Fungsi-fungsi atau operasi pada ADT set adalah:

- **union** : operasi union (gabung) dua himpunan (|)
- **intersection** : operasi intersection (nilai sama) pada dua himpunan (&)
- **difference** : mencari elemen beda dari dua set (-)
- **symmetric difference** : elemen berada di set 1 atau di set 2 tapi tidak kedua-dua-nya (^)

Contoh operasi Set

```
>>> set1={1,2,3}
>>> set2={3,4,5}
>>> a = set1 | set2    # union
>>> a
{1, 2, 3, 4, 5}
>>> b = set1 & set2    # intersection
>>> b
{3}
>>> c = set1 - set2    # difference
>>> c
{1, 2}
>>> d = set1 ^ set2    # symmetric difference
>>> d
{1, 2, 4, 5}
>>>
```

Beberapa Aplikasi

- **listArray.py** : membuat Array memakai List
- **MadLib.py** : aplikasi permainan kata memakai List
- **histoKata.py** : membuat histogram kata, memakai dictionary

listArray.py

```
#listArray.py => membuat array berbasis python List
#1.menciptakan array berisi 0
def ciptaArray(A,n):
    #n = jumlah elemen
    for i in range(n):
        A.append(0)

#2.mengisi array
def isiArray(A,n):
    for i in range(n):
        A[i] = int(input('isi: '))

#3.baca isi array
def bacaArray(A):
    r = input('0-Semua 1-satu data : ')
    r = int(r)
    if r < 1:
        print(A)
    else:
        x = int(input('posisi : '))
        print(A[x-1])
```

```
#4.cari data
def cariArray(x,A):
    n = len(A)
    ketemu = False
    for i in range(n):
        if (x == A[i]):
            print('Ketemu di posisi ',i+1,'\n')
            ketemu = True
    if (not ketemu):
        print(x,' tidak ada dalam array \n')

#5.hapus elemen
def hapus(x,A):
    n = len(A)
    ketemu = False
    for i in range(n):
        if (x == A[i]):
            print('Ketemu di posisi ',i+1,'\n')
            ketemu = True
            A[i] = ''
    if (not ketemu):
        print(x,' tidak ada dalam array \n')
```

list_Array_test.py

```
#list_Array_test.py
from listArray import *

def main():
    A = []
    ciptaArray(A,7)
    isiArray(A,7)
    print("Isi Larik : ")
    bacaArray(A)
    print("Mencari angka dalam larik")
    z = int(input("Data akan dicari: "))
    cariArray(z,A)
    z = int(input('Data yang akan dihapus: '))
    print("Menghapus elemen %d" % z)
    hapus(z,A)
    print("Larik setelah hapus data:")
    bacaArray(A)

main()
```

Hasil test:

```
*** Remote Interpreter Reinitialized ***
isi: 4
isi: 2
isi: 5
isi: 6
isi: 9
isi: 8
isi: 7
Isi Larik :
0-Semua 1-satu data : 0
[4, 2, 5, 6, 9, 8, 7]
Mencari angka dalam larik
Data akan dicari: 7
Ketemu di posisi 7

Data yang akan dihapus:5
Menghapus elemen 5
Ketemu di posisi 3

Larik setelah hapus data:
0-Semua 1-satu data : 0
[4, 2, '', 6, 9, 8, 7]
>>>
```

```
# MadLib.py
import random
namesList = [ 'Weird Al Yankovic', 'The Teenage Mutant Ninja Turtles', \
              'Supergirl', 'The Stay Puft Marshmallow Man', 'Shrek', \
              'Sherlock Holmes', 'The Beatles', 'Powerpuff Girl', \
              'The Pillsbury Doughboy' ]

while True:
    # Choose a random index into the namesList
    nameIndex = random.randrange(0, 9)
    name = namesList[nameIndex] # Use the index to choose a random name
    verb = input('Enter a verb: ')
    adjective = input('Enter an adjective: ')
    noun = input('Enter a noun: ')
    sentence = name + ' ' + verb + \
               ' through the forest, hoping to escape the ' + \
               adjective + ' ' + noun + '.'
    print()
    print(sentence)
    print()
    # See if the user wants to quit or continue
    answer = input('Type "q" to quit, or Enter to continue: ')
    if answer == 'q':
        break

print('Bye')
```

Hasil test

```
Enter a verb: running
```

```
Enter an adjective: slowly
```

```
Enter a noun: lion
```

```
The Beatles running through the forest, hoping to escape  
the slowly lion.
```

```
Type "q" to quit, or Enter to continue:
```

```
Enter a verb: jumping
```

```
Enter an adjective: faster
```

```
Enter a noun: monkey
```

```
Shrek jumping through the forest, hoping to escape the  
faster monkey.
```

```
Type "q" to quit, or Enter to continue: q
```

```
Bye
```

```
>>>
```

histoKata.py

```
#histoKata.py - histogram kata dari file teks
import string
def proses_file(filename):
    # hist adalah dictionary yang menampung kata
    hist = {}
    try:
        fp = open(filename)

        for line in fp:
            proses_line(line, hist)
    return hist
except IOError as ioerr:
    print('File tidak bisa dibuka ...' + str(ioerr))
return (None)
```

```
def proses_line(line, hist):
    # memisah setiap baris menjadi beberapa kata
    # memasukkan setiap kata ke dalam hist{}
    # Apabila ada hyphens maka ganti dengan spasi
    line = line.replace('-', ' ')
    for word in line.split():
        # mengabaikan semua tanda baca (punctuation)
        # dan mengubah ke huruf kecil (lowercase)
        word = word.strip(string.punctuation + \
                           string.whitespace)
        word = word.lower()
        # memasukkan kata ke histogram
        hist[word] = hist.get(word, 0) + 1

def daftar_kata(hist):
    # membuat daftar kata dan frekuensinya
    # ber-urut mulai dari yang tertinggi (descending)
    t = []
    for key, value in hist.items():
        t.append((value, key))

    t.sort()
    t.reverse()
    return t
```

```
def cetak_daftar_kata(hist, num=10):
    t = daftar_kata(hist)
    print ('Kata yang paling sering muncul adalah:')
    for freq, word in t[:num]:
        print (word, '\t\t', freq)

def total_kata(hist):
    # memberikan jumlah kata yang ada dalam file
    return sum(hist.values())

def different_words(hist):
    # memberikan banyaknya kata yang berbeda."""
    return len(hist)

# program utama
def main():
    path = "D:\\USER\\Python\\jaringan.txt"
    hist = proses_file(path)
    print ('Jumlah kata dalam file: : ', total_kata(hist))
    print ('Banyaknya kata berbeda: ', different_words(hist))

    t = daftar_kata(hist)
    print ('20 Kata dengan frekuensi tertinggi:')
    for freq, word in t[0:20]:
        print ('%12s %4d' % (word, freq))

main()
```

Hasil test

```
===== RESTART: D:\USER\Python\histoKata.py ===
Jumlah kata dalam file: : 555
Banyaknya kata berbeda: 255
20 Kata dengan frekuensi tertinggi:
jaringan      20
komputer      15
    dan        14
    yang        13
    data        13
    untuk       10
dengan         10
    dapat       10
    satu        9
    pada        9
    dari        9
    dalam       9
informasi      8
    atau        8
    lain        7
secara         6
    maka        6
    ada         6
    ú           5
suatu          5
```

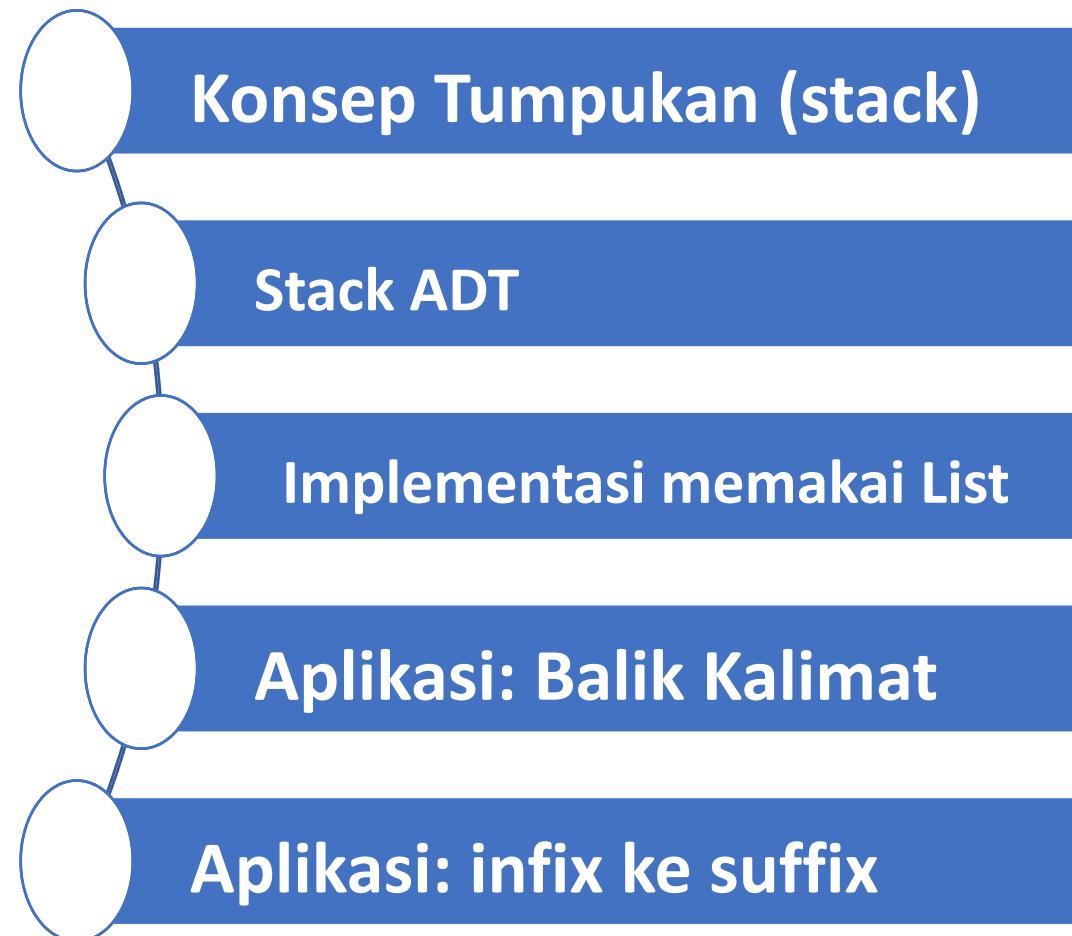


STRUKTUR DATA (PYTHON)

“Struktur Data Tumpukan (Stack)”

[@SUARGA] | [Pertemuan 06]

OutLine



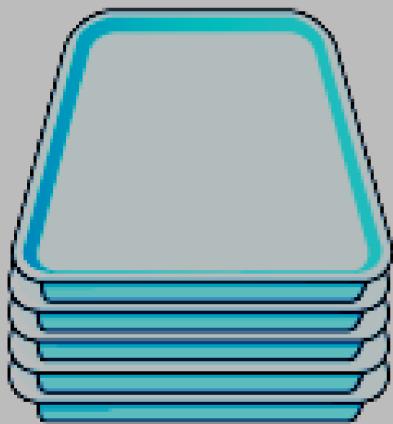


Konsep Tumpukan (STACK)

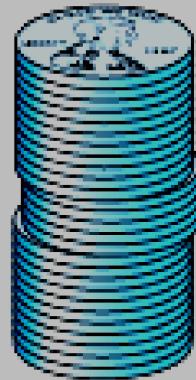
- Struktur Tumpukan (Stack) adalah struktur data yang meniru bagaimana proses menyimpan dan mengambil satu benda pada suatu tumpukan benda yang ada dilantai.
- Apabila diperhatikan dengan saksama maka proses menyimpan buku (disebut *push*) dan proses mengambil buku (disebut *pop*) dari suatu tumpukan selalu dilakukan pada bagian atas tumpukan (*top of the stack*).
- Sehingga terjadi urutan yang disebut LIFO (Last In First Out), artinya benda yang terakhir disimpan pada tumpukan adalah benda yang pertama yang bisa diambil karena benda inilah yang berada pada urutan teratas dari tumpukan.

Contoh Tumpukan

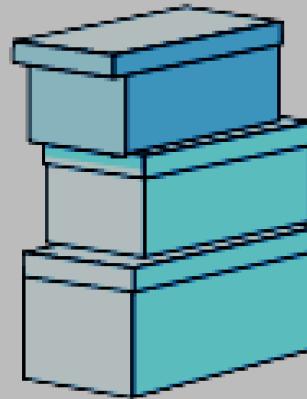
A stack of
cafeteria trays



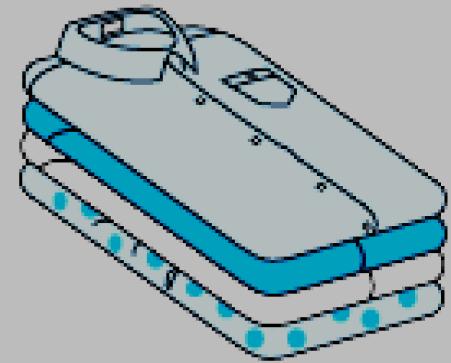
A stack
of pennies



A stack of
shoe boxes



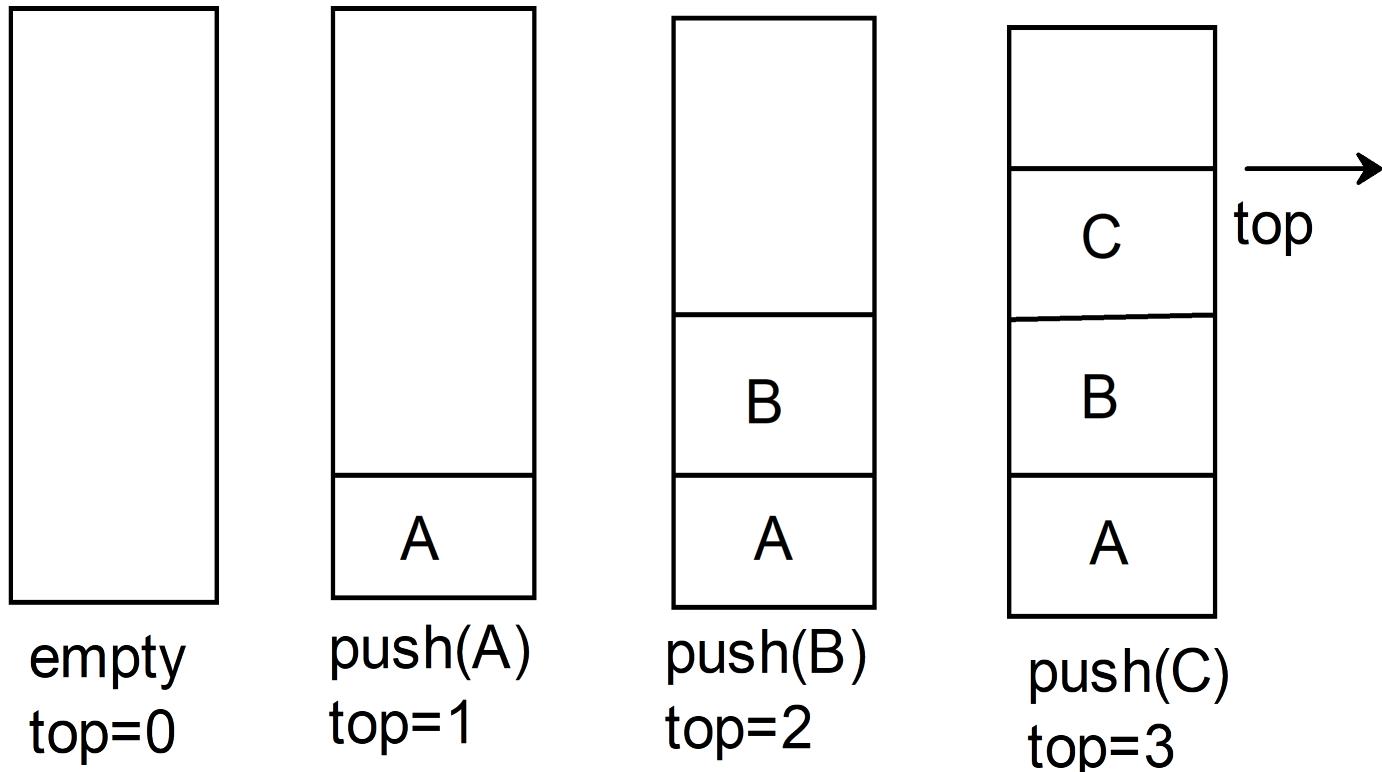
A stack of
neatly folded shirts



Stack ADT

- Tumpukan (Stack) digunakan apabila data akan di-akses dengan urutan “Last In First Out” (LIFO), data yang pertama bisa diakses adalah data yang terakhir dimasukkan. Beberapa fungsi pada ADT tumpukan yang perlu dibentuk untuk keperluan akses data adalah sebagai berikut:
- **push(x)** : menambahkan elemen x ke bagian top
- **pop()** : mengambil data dari posisi top
- **peek()** : melihat isi yang teratas/pertama
- **top()** : elemen teratas, sama dengan peek()
- **isEmpty()** : memeriksa apakah tumpukan kosong
- **len()** : menghitung jumlah elemen dalam tumpukan

Contoh : push(x)



Algoritma Push(x)

```
Prosedur push (input x : item; in-out S : Stack)
{ prosedur menempatkan item x pada posisi top dari stack }
```

Definisi Variabel

```
int atas;
```

Rincian Langkah

```
If S.top = maks
  then write ("sudah penuh");
else
  S.top ← S.top + 1;
  atas ← S.top;
  S.isi[atas] ← x;
endif.
```

Python: Fungsi push(x)

```
def push(self, x) :  
    if self.top == maks:  
        print ('Sudah penuh')  
    else:  
        self.top += 1  
        atas = self.top  
        self.isi[atas] = x
```

- menggunakan python list sebagai struktur data dasar akan memudah implementasi ADT dari stack, perhatikan fungsi push berikut:

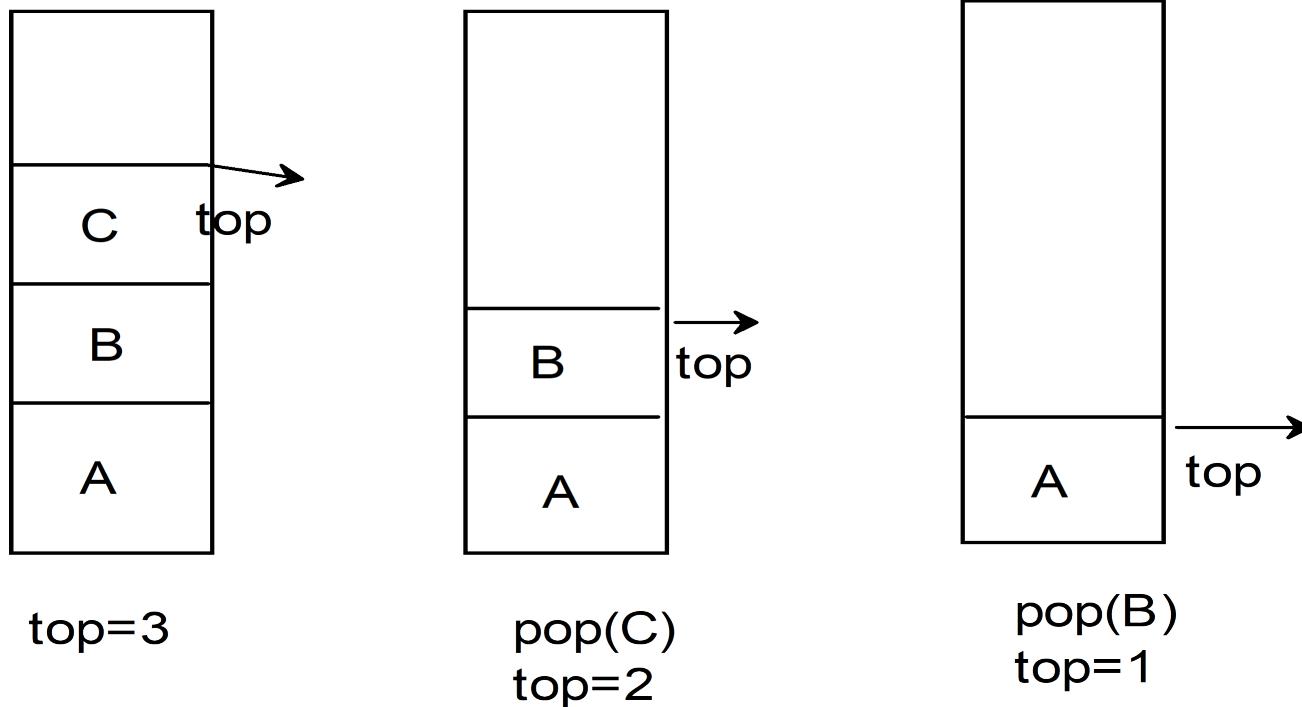
```
def push(self, e) :  
    # memasukkan elemen e di posisi top  
    self._data.append(e)
```

- fungsi append() pada list otomatis menempatkan elemen di posisi atas
- memakai list maka stack tidak akan penuh (list adalah dynamic array)

Class ArrayStack

```
class ArrayStack:  
    #LIFO Stack implementation using a Python list  
    #as underlying storage.  
  
    def __init__(self):  
        #Create an empty stack.  
        self._data = [] # nonpublic list instance  
  
    def __len__(self):  
        #Return the number of elements in the stack.  
        return len(self._data)
```

Contoh fungsi pop()



Algoritma fungsi Pop()

Prosedur pop (output x : item; in-out S:Stack)
{ prosedur mengambil data dari tumpukan pada posisi top
dari stack }

Definisi Variabel

int atas;

Rincian Langkah:

```
If S.top = 0
  then write ("tumpukan kosong");
else
  atas ← S.top;
  x ← S.isi[atas];
  S.top ← S.top - 1;
endif
```

Python: fungsi pop()

- berbasis operasi python list, prosedur pop()

```
def pop(self):  
    # mengambil elemen di posisi top  
    #Raise Empty exception if the stack is empty.  
  
    if self.is_empty():  
        raise Empty('Stack is empty')  
    return self._data.pop()
```

Python: fungsi top()

- ke pointer ter-atas

```
def top(self):  
    #bila stack kosong error  
    if self.is_empty():  
        raise Empty('stack is empty')  
    #bila stack ber-isi, ambil yang ter-atas  
    return self._data[-1]
```

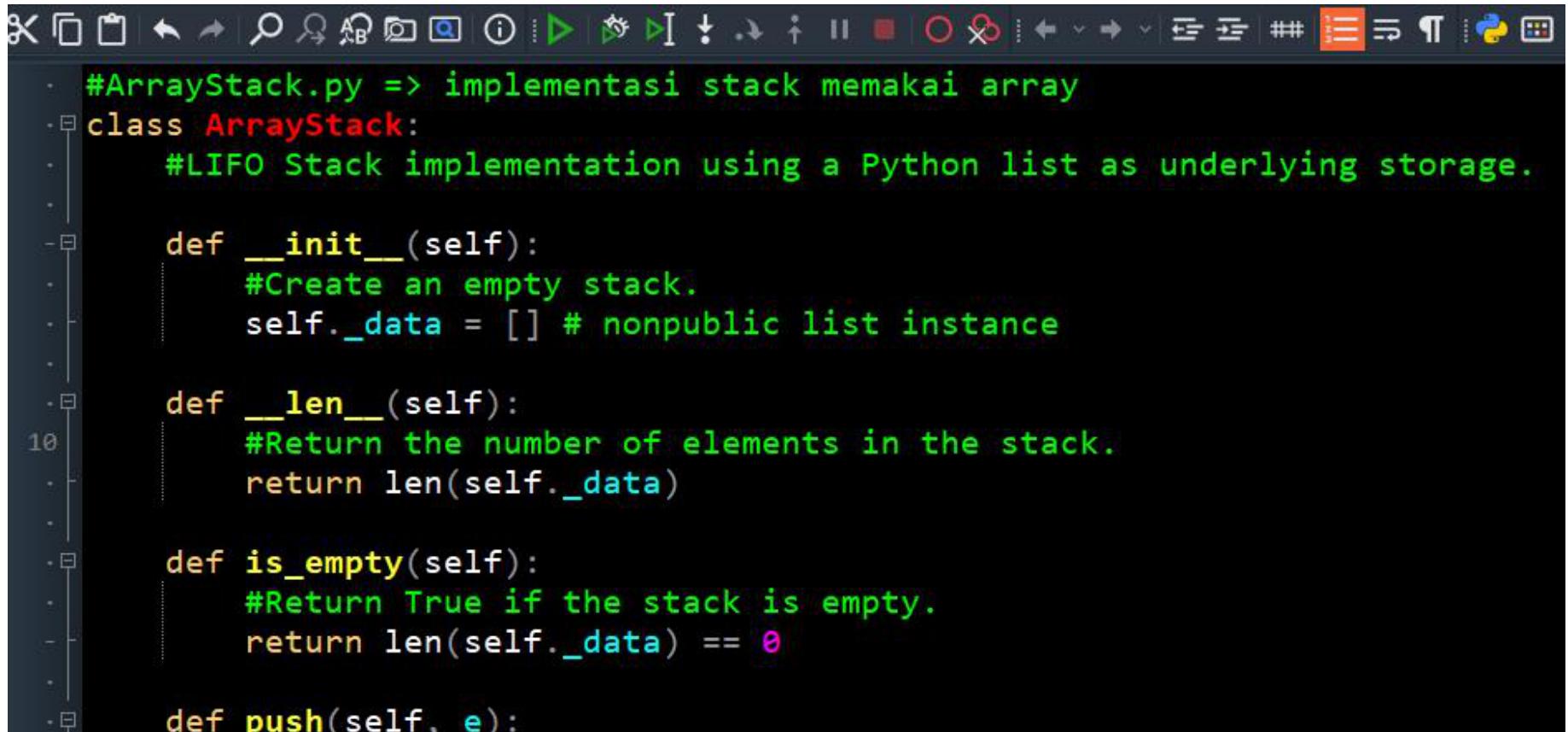
Fungsi peek() : melihat data ter-atas, sama dengan fungsi top()

Fungsi: lihat_isi()

- untuk melihat susunan data dalam stack
- dari yang ter-atas hingga ter-bawah

```
def lihat_isi(self):  
    n = len(self._data)  
    for i in range(n):  
        print(self._data[-i-1])
```

Implementasi Stack



```
#ArrayStack.py => implementasi stack memakai array
class ArrayStack:
    #LIFO Stack implementation using a Python list as underlying storage.

    def __init__(self):
        #Create an empty stack.
        self._data = [] # nonpublic list instance

    def __len__(self):
        #Return the number of elements in the stack.
        return len(self._data)

    def is_empty(self):
        #Return True if the stack is empty.
        return len(self._data) == 0

    def push(self, e):
```

```
def push(self, e):
    #Add element e to the top of the stack.
    self._data.append(e) # new item stored at end of list
20
def top(self):
    #Return (but do not remove) the element at the top
    #of the stack.
    #Raise Empty exception if the stack is empty.

    if self.is_empty():
        raise Empty('Stack is empty')
    return self._data[-1] # the last item in the list

30
def pop(self):
```

```
30 def pop(self):
    """
    Remove and return the element from the top of
    the stack (i.e., LIFO).
    Raise Empty exception if the stack is empty.

35     if self.is_empty():
        raise Empty('Stack is empty')
    return self._data.pop()

40 def peek(self):
    """
    peek the top element
    if self.is_empty():
        raise Empty('Stack is empty')
    return self._data[-1] #peek() sama dengan top()
```

Contoh pemakaian:

```
>>> S=ArrayStack()          #menciptakan stack S
>>> S.push(5)              #memasukkan data 5,4,3,2
>>> S.push(4)
>>> S.push(3)
>>> S.push(2)
>>> print(len(S))         #tinggitumpukan
4
>>> print(S.pop())         #ambil elemen teratas => 2
2
>>> print(S.top())          #elemen teratas berikutnya
3
>>> print(S.peek())         #melihat elemen ter-atas
3
>>> print(S.is_empty())     #apakah stack s kosong?
False
>>>
```

Aplikasi: Balik Kalimat

- Salah satu contoh aplikasi dari struktur data tumpukan ini adalah membalik suatu kalimat yang dimasukkan lewat keyboard. Misalkan input-nya adalah “program tumpukan” maka output-nya adalah “nakupmut margorp”.
- Prosesnya dapat dijelaskan secara sederhana, setiap huruf, mulai dari kiri, di masukkan (push) ke dalam tumpukan (stack) S, hingga seluruhnya ada di dalam S. Kemudian satu persatu huruf ini diambil (pop) dari S, dengan demikian kalimat akan terbaca terbalik.

```
#balik.py => membalikkan kalimat
from ArrayStack import *      #memanggil implementasi Stack
S=ArrayStack()                  #menciptakan tumpukan S

kalimat = input('Masukkan satu kalimat : ')
panjang=len(kalimat)

#masukkan tiap huruf ke stack
for c in kalimat:
    S.push(c)

#baca isi stack
x=''
for i in range(panjang):
    x=x+S.pop()

print('Baca terbalik : ')
print(x)
```

Python Interpreter

```
*** Remote Interpreter Reinitialized ***
Masukkan satu kalimat : Selamat Pagi
Baca terbalik :
igaP tamaleS
>>>
*** Remote Interpreter Reinitialized ***
Masukkan satu kalimat : Ini program tumpukan
Baca terbalik :
nakupmut margorp inI
>>> |
```

Aplikasi membalik isi file

```
#reverse_file.py => membalikkan isi file teks
from ArrayStack import *

def reverse_file(filename):
    S=ArrayStack()
    original = open(filename)
    print("Ini isi file aslinya:")
    for line in original:
        print(line.rstrip('\n'))
        S.push(line.rstrip('\n'))
    original.close()

    outfile = "D:\\USER\\Python\\reverse.txt"
    output = open(outfile, 'w')
    while not S.is_empty():
        output.write(S.pop() + '\n')

    output.close()
```

```
def main():
    filename = input("Masukkan nama file teks: ")
    reverse_file(filename) # contoh:'D:\\USER\\Python\\emma2.txt')
    print("\n\nIni hasil proses reverse:")
    outfile="D:\\USER\\Python\\reverse.txt"
    reverse = open(outfile)
    for line in reverse:
        print(line.rstrip('\\n'))

main()
```

Hasil proses

The image shows two windows of the Windows Notepad application. The top window, titled '*emma2.txt - Notepad', contains the first half of a text from Jane Austen's 'Emma'. The bottom window, titled 'reverse.txt - Notepad', contains the second half of the same text, with the lines reversed. A status bar at the bottom of the bottom window indicates 'Ln 1, Col 1' and '100% Windows (CRL) UTF-8'.

*emma2.txt - Notepad

File Edit Format View Help

Emma Woodhouse, handsome, clever, and rich, with a comfortable home and happy disposition, seemed to unite some of the best blessings of existence; and had lived nearly twenty-one years in the world with very little to distress or vex her.

She was the youngest of the two daughters of a most affectionate, indulgent father; and had, in consequence of her sister's marriage, been mistress of his house from a very early period. Her mother had died too long ago for her to have more than an indistinct remembrance of her caresses; and her place had been supplied by an excellent woman as governess, who had fallen little short of a mother in affection.]

reverse.txt - Notepad

File Edit Format View Help

Ln 12, Col 26

bf a mother in affection.
by an excellent woman as governess, who had fallen little short
remembrance of her caresses; and her place had been supplied
had died too long ago for her to have more than an indistinct
been mistress of his house from a very early period. Her mother
indulgent father; and had, in consequence of her sister's marriage,
She was the youngest of the two daughters of a most affectionate,

with very little to distress or vex her.
of existence; and had lived nearly twenty-one years in the world
and happy disposition, seemed to unite some of the best blessings
Emma Woodhouse, handsome, clever, and rich, with a comfortable home

Ln 1, Col 1 | 100% Windows (CRL) UTF-8

Aplikasi: infix ke suffix

- Aplikasi yang lain dari tumpukan (stack) adalah penulisan notasi “polish” dari ekspresi matematis. Ketika compiler akan melaksanakan ekspresi matematis maka notasi yang dikenal (notasi infix) diubah ke notasi polish (atau notasi suffix), misalnya sebagai berikut:

infix : $a - b * c$

suffix (polish): $a b c * -$

$(a + b) * (c + d)$

$a b + c d + *$

$a - (b + c) / d + e$

$a b c + d / - e +$

Agar proses penterjemahan notasi infix ke notasi polish dapat dilakukan maka diperlukan tabel atau vektor dari urutan (precedence) simbol operator serta rank-nya, seperti pada tabel berikut ini.

Simbol	Precedence (f)	Rank (r)
+	1	-1
*	2	-1
a, b, c, ...	3	1
#	0	

- Input (masukan) algoritma adalah string infix yang diakhiri oleh tanda '#', output (keluaran) algoritma adalah notasi polish. Secara umum algoritma-nya sebagai berikut:

1. Inisialisasi isi tumpukan S dengan simbol #
2. Baca notasi infix, kemudian ambil satu simbol mulai dari sisi paling kiri
3. Lakukan perulangan hingga langkah ke-6 selama simbol input bukan #
4. Ambil dan keluarkan (pop) simbol pada tumpukan apabila nilai precedence-nya \geq nilai precedence dari input saat ini.
5. Masukkan (push) simbol input ke dalam tumpukan
6. Baca kembali simbol berikutnya dari string infix.

Listing Program: `polish.py`

The screenshot shows a Java IDE interface with a dark theme. The title bar reads "D:\USER\Python\polish.py". The menu bar includes "File", "View", "Project", "Run", "Tools", and "Help". Below the menu is a toolbar with various icons. The main area displays Python code for converting infix expressions to postfix. The code defines a function `f` that maps operators to precedence levels (1 for +/-, 2 for */, and 3 for #). The code is as follows:

```
#polish.py => ubah infix menjadi postfix
from ArrayStack import *

#memeriksa precedence
def f(c):
    if c in ('+', '-'):
        return 1
    else:
        if c in ('*', '/'):
            return 2
        else:
            if c == '#':
                return 0
            else:
                return 3
```

```
·
·     #memeriksa ranking
·     def r(c):
·         if c in ('+', '-'):
·             return -1
·         else:
·             if c in ('*', '/'):
·                 return -1
·             else:
·                 if c == '#':
·                     return 0
·                 else:
·                     return 1
·
·
30     def main():
```

```
30 def main():
    S=ArrayStack()
    S.push('#')
    infix = input('Masukkan notasi infix diakhir # : ')
    current = infix[0]
    polish=''
    idx=0
    rank=0
    while (current != '#'):
        if S.is_empty():
            print('Invalid, stack kosong')
            break

        while (f(current) <= f(S.top())):
            temp = S.pop()
            polish = polish + temp
            rank = rank + r(temp)
            if (rank < 1):
                print('invalid infix !')
                break

        S.push(current)
        idx=idx+1
        current=infix[idx]

    while (S.top() != '#'):
```

```
54 |
|     while (S.top() != '#'):
|         temp = S.pop()
|         polish = polish + temp
|         rank = rank + r(temp)
|         if rank < 1:
|             print('Invalid !!!')
|             break
|
|         if (rank==1):
|             print('Valid-polish : ')
|             print(polish)
|         else:
|             print('Invalid result')
|
|
70 main()
```

Contoh RUN

```
|           while (s.top() != '#'):  
Python Interpreter  
*** Remote Interpreter Reinitialized ***  
Masukkan notasi infix diakhir # : a*b-c/d/e+f#  
Valid-polish :  
ab*c d/e/-f+  
>>>  
*** Remote Interpreter Reinitialized ***  
Masukkan notasi infix diakhir # : 3*x+5*x/y-2*y/x#  
Valid-polish :  
3x*5x*y/+2y*x/-  
>>>
```

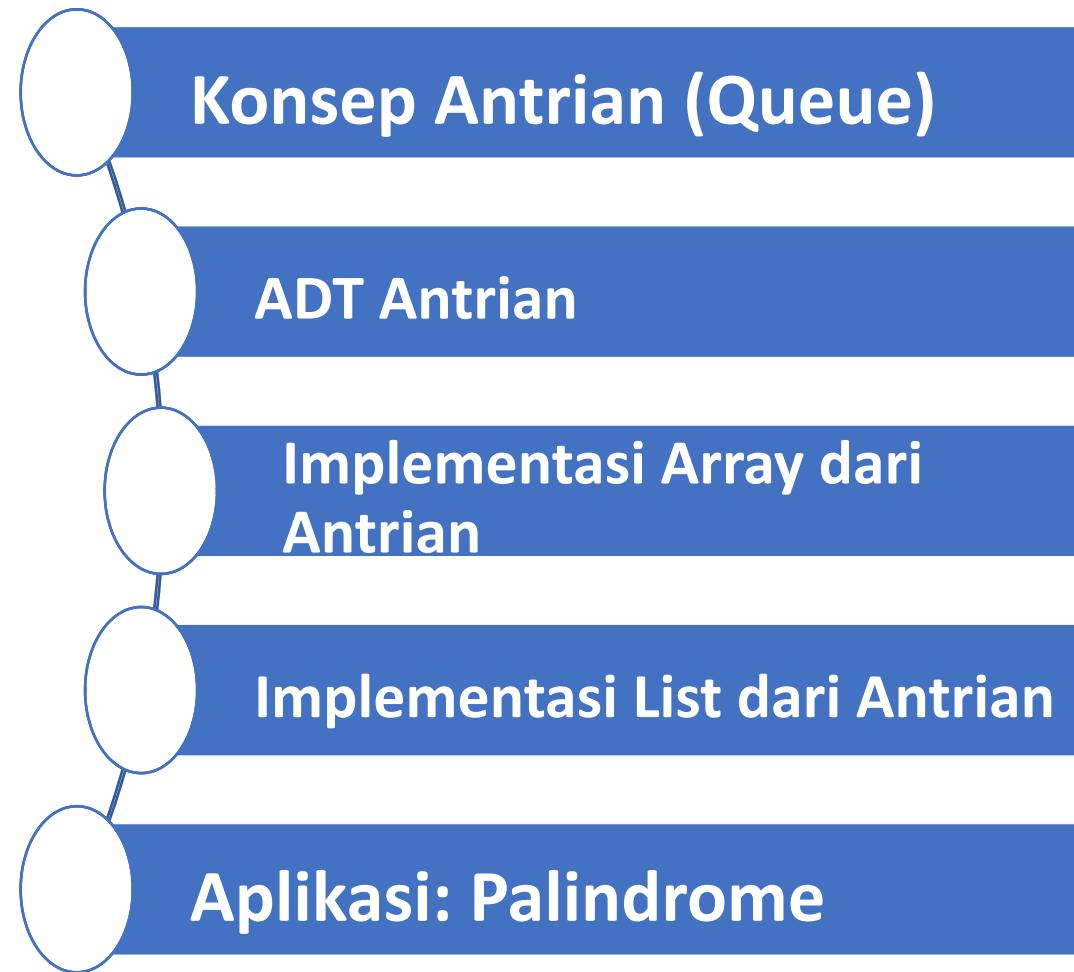


STRUKTUR DATA (PYTHON)

“Struktur Antrian (Queue)”

[@SUARGA | [Pertemuan 07]

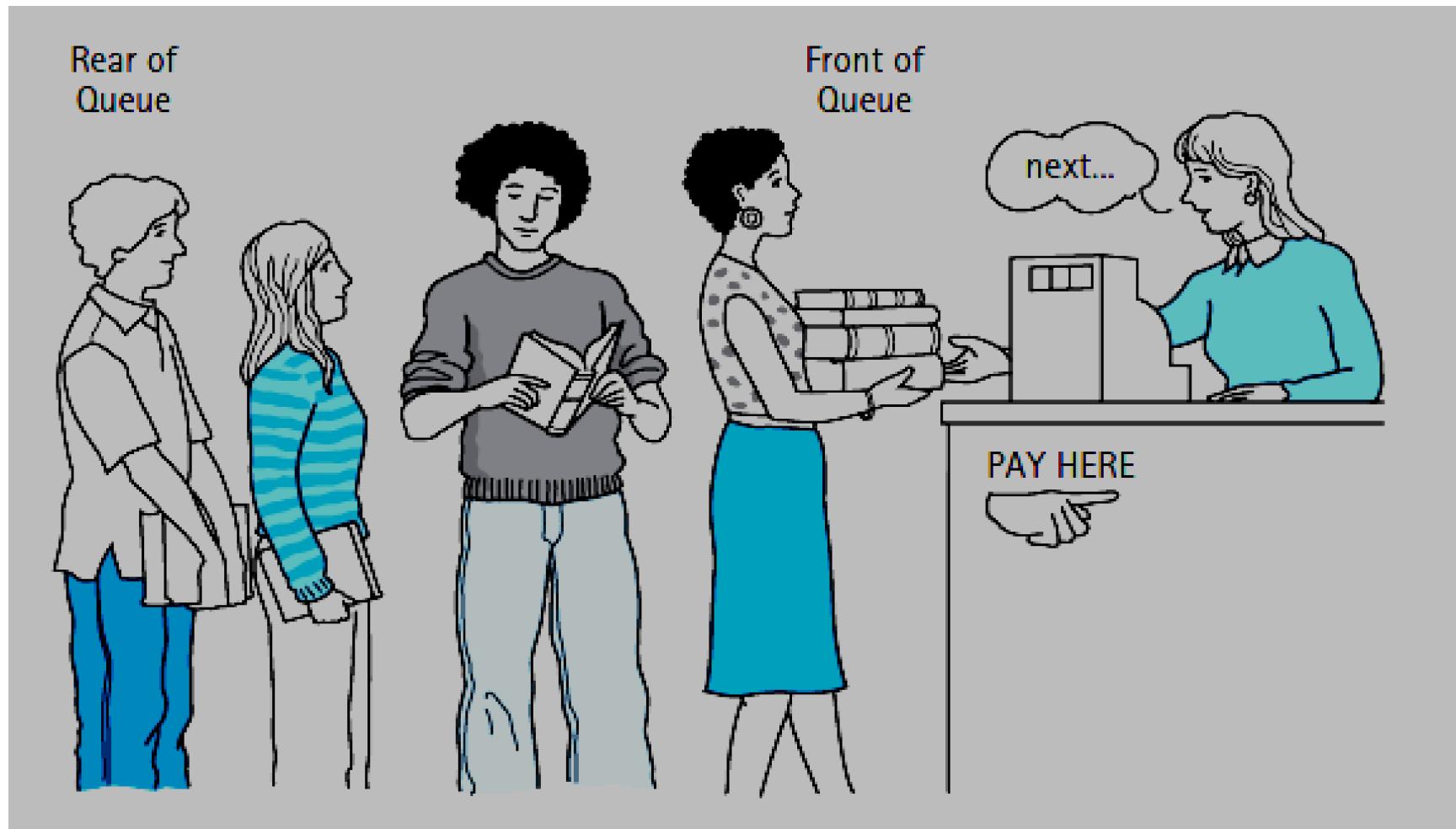
OutLine



Konsep Antrian (Queue)



- Struktur Antrian adalah struktur data yang meniru antrian orang yang sedang menunggu pelayanan misalnya didepan seorang teller bank, atau antrian orang yang sedang beli karcis pertunjukkan.
- Apabila diperhatikan dengan saksama maka penambahan orang pada suatu antrian selalu dilakukan pada urutan paling belakang (rear of queue), dan pelayanan selalu dilakukan pada urutan depan (front of queue), sehingga urutan proses antrian sering disebut sebagai FIFO (First In First Out), yang pertama masuk antrian itulah yang pertama dilayani.



Abstraksi Antrian

- Implementasi Antrian dapat dilakukan dengan membuat tipe data buatan bernama Queue, dengan dua variabel pointer, misalnya sebagai berikut:

Type Queue : record

<

```
    size : integer  
    front, rear : pointer;  
    isi : array[1..maks] of item;
```

>

- Dimana **size** adalah variabel untuk mencacah jumlah elemen dalam antrian, **front** adalah variabel yang menunjuk pada awal (bagian depan) antrian, **rear** adalah variabel yang menunjuk pada bagian akhir antrian, dan **isi** adalah array yang menyimpan isi antrian.



Antrian dengan sisi depan (**front**) dan
sisi belakang (**back** atau **rear**)

ADT antrian

- Fungsi ADT yang diperlukan dalam struktur data antrian antara lain adalah:
- **Fungsi untuk memulai antrian (Queue):** menciptakan array untuk antrian, kemudian me-mulai semua indeks, size=0, rear=0, front=1.
- **Fungsi untuk menambahkan elemen (enQueue):** elemen ditambahkan dari belakang, sehingga size bertambah 1, juga rear bertambah 1.
- **Fungsi untuk mengambil elemen (deQueue):** elemen diambil dari antrian selalu dari depan, sehingga size berkurang 1, dan front bertambah 1
- **Fungsi untuk menghitung jumlah elemen (len):** berapa nilai count
- **Fungsi untuk melihat elemen pertama (first)**
- **Fungsi untuk mengubah kapasitas (resize)**
- **Fungsi untuk memeriksa apakah antrian kosong (is_Empty)**

`x = Q.dequeue()`



`Q.enqueue(21)`



`Q.enqueue(74)`



dequeue() dan enqueue() pada antrian

Algoritma Implementasi Antrian

- Ketika mulai diciptakan (Queue), semua pointer harus di-beri nilai awal, prosedur-nya:

Prosedur Queue(in-out Q:Queue)
{ memulai suatu antrian }

Definisi variabel:

{ tidak ada }

Rincian Langkah:

```
Q.size <- 0;  
Q.front <- 0;  
Q.rear <- 1;  
return.
```

Prosedur Enqueue(x)

- Memasukkan objek x ke dalam antrian (enqueue) pada posisi paling belakang (rear) sebagai berikut:

Prosedur enqueue(**input** x:item; **in-out** Q:Queue)

{ menambah elemen kedalam antrian }

Definisi variabel:

{ tidak ada }

Rincian Langkah:

```
if (Q.size = maks) // atau i(Q.isFull())
then write ("tak ada tempat kosong");
else
    Q.size <- Q.size + 1;
    Q.rear <-(Q.rear % maks) + 1;
    Q.isi[Q.rear] <- x;
endif
return.
```

Prosedur deQueue()

- Proses melayani isi antrian, atau mengambil elemen dari antrian, yaitu pada posisi terdepan (front)

Prosedur deQueue(output x:item; in-out Q:Queue)

{ mengambil satu elemen dari antrian }

Definisi variabel:

{ tidak ada }

Rincian Langkah:

```
if (Q.size = 0) // atau: if (Q.isEmpty())
then write (“antrian kosong ”);
else
    Q.size <- Q.size - 1;
    x <- Q.isi[front];
    front <- (front % maks) + 1;
endif
return;
```

Fungsi len()

- Menghitung jumlah objek / elemen dalam antrian melalui fungsi len()

Fungsi len(in-out Q:Queue) à integer
{ menghitung jumlah elemen dalam antrian }

Definisi variabel:

```
int n;
```

Rincian Langkah:

```
n <- Q.size;  
return n;
```

Implementasi Python Queue, memakai Array

```
#ArrayQueue.py == @Suarga
#=> implementasi Antrian memakai Array
class ArrayQueue:
    #FIFO queue implementation using an array as underlying storage.
    DEFAULT_CAPACITY = 10 # moderate capacity for all new queues

    def __init__(self):
        #Create an empty queue.""""
        self._data = [None]*ArrayQueue.DEFAULT_CAPACITY
        self._size = 0
        self._front = 0
        self._rear = 1

    def __len__(self):
        #Return the number of elements in the queue.""""
        return self._size

    def is_empty(self):
        #Return True if the queue is empty.""""
        return self._size == 0
```

```
def first(self):
    #Return (but do not remove) the element at the front of the queue.
    #Raise Empty exception if the queue is empty.
    if self.is_empty():
        print("Antrian MASIH KOSONG")
        raise Exception('Queue is empty')

    return self._data[self._front]

def dequeue(self):
    #Remove and return the first element of the queue (i.e., FIFO).
    #Raise Empty exception if the queue is empty.
    if self.is_empty():
        print("AWAS dequeue tidak boleh dilakukan!")
        print("Error Antrian sudah KOSONG")
        #raise Exception('Queue is empty')

    x = self._data[self._front]
    self._data[self._front] = None # help garbage collection
    self._front = (self._front + 1) % len(self._data)
    self._size -= 1
    return x
```

```

def enqueue(self, x):
    #Add an element to the back of queue.""""
    if self._size == len(self._data):
        self._resize(2*len(self._data)) # double the array size
    self._rear = (self._front + self._size) % len(self._data)
    self._data[self._rear] = x
    self._size += 1

def _resize(self, cap): # we assume cap >= len(self)
    #Resize to a new list of capacity >= len(self).""""
    old = self._data # keep track of existing list
    self._data = [None]*cap # allocate list with new capacity
    walk = self._front
    for k in range(self._size): # only consider existing elements
        self._data[k] = old[walk] # intentionally shift indices
        walk = (1 + walk) % len(old) # use old size as modulus
    self._front = 0

```

```
#testArrayQueue.py
from ArrayQueue import *
def main():
    Q = ArrayQueue()
    Q.enqueue(5)
    Q.enqueue(3)
    Q.enqueue(4)
    Q.enqueue(2)

    n = len(Q)
    print("Ada : ", n , " data dalam antrian")

    print ("Data pertama : ", Q.first())
    print("melakukan pelayanan:")
    for i in range(n):
        x = Q.dequeue()
        print("Mengambil data : ", x)

    #memeriksa antrian
    if (Q.is_empty()):
        print("Antrian sudah kosong: ")
    else:
        print("Masih ada data dalam antrian")

    print("mencoba mengambil data!")
    Q.dequeue()

main()
```

Hasil RUN

```
===== RESTART: D:/USER/Python/testArrayQueue.py ====
Ada : 4 data dalam antrian
Data pertama : 5
melakukan pelayanan:
Mengambil data : 5
Mengambil data : 3
Mengambil data : 4
Mengambil data : 2
Antrian sudah kosong:
mencoba mengambil data!
AWAS dequeue tidak boleh dilakukan!
Error Antrian sudah KOSONG
```

Implementasi Queue memakai List

```
1 #antrian.py => antrian memakai Python List @SUARGA
2 class antrian:
3     #implementasi antrian memakai Python list.
4
5     def __init__(self):
6         #inisialisasi antrian.
7         self._data = [] # memulai dengan list kosong
8
9     def __len__():
10        #memberikan jumlah elemen dalam antrian.
11        return len(self._data)
12
13    def size():
14        return len(self._data)
15
16    def is_empty():
17        #memeriksa apakah antrian kosong.
18        return len(self._data) == 0
19
20    def enqueue(x):
```

```
20     def enqueue(self, x):
21         #memasukkan x ke dalam antrian.
22         self._data.append(x)
23
24     def dequeue(self):
25         #mengambil elemen terdepan dari tumpukan
26         #Raise error Exception bila kosong.
27
28     if self.is_empty():
29         print("Tumpukan KOSONG !")
30         print("dequeue TIDAK BOLEH !")
31         print("ERROR !!")
32         raise Exception('Tumpukan kosong')
33     else:
34         x = self._data[0]
35         self._data.pop(0)
36     return x
37
38     def front(self):
```

```
38     def front(self):
39         #melihat elemen terdepan
40         if self.is_empty():
41             print("ERROR Tumpukan MASIH KOSONG !")
42             print("Belum ada elemen !!!")
43         else:
44             return self._data[0]
45
46     def rear(self):
47         #melihat elemen paling belakang
48         if self.is_empty():
49             print("ERROR Tumpukan MASIH KOSONG !")
50             print("Belum ada elemen !!!")
51             #raise Exception('Tumpukan kosong')
52         else:
53             return self._data[-1]
54
55     def lihat_isi(self):
56         n = self.size()
57         for i in range(n):
58             print(self._data[i])
59
```

```
#Contoh Pemakaian: antrian_test.py
from antrian import *
def main():
    A = antrian()      #membuat antrian
    print("memasukkan 4 elemen:4  6  8  4")
    A.enqueue(4)
    A.enqueue(6)
    A.enqueue(8)
    A.enqueue(4)
    print("Ada ", len(A), " data dalam antrian")
    print("Melihat isi Antrian:")
    A.lihat_isi()
    print("mengambil elemen terdepan")
    x = A.dequeue()
    print(x)
    print("ambil satu lagi ..")
    print(A.dequeue())
    print("periksa elemen terdepan")
    print(A.front())
    print("periksa elemen belakang")
    print(A.rear())
    print("Apakah A sudah kosong?")
    print(A.is_empty())
main()
```

Hasil RUN

```
===== RESTART: D:\USER\Python\antrian_test.py ===
memasukkan 4 elemen:4 6 8 4
Ada 4 data dalam antrian
Melihat isi Antrian:
4
6
8
4
mengambil elemen terdepan
4
ambil satu lagi ..
6
periksa elemen terdepan
8
periksa elemen belakang
4
Apakah A sudah kosong?
False
```

Aplikasi Palindrome

- Kalimat palindrome adalah kalimat yang apabila dibaca dari kiri ke kanan akan sama apabila dibaca terbalik dari kanan ke kiri. Contoh dari kalimat kalimat palindrome adalah sebagai berikut:
- Kalimat yang diucapkan Napoleon Bonaparte ketika dibuang ke pulau Elba,
“Able was I ere, I saw Elba.”
- Pujian ke Teddy Roosevelt untuk pembangunan terusan Panama,
“A man, a plan, a canal—Panama!”
- Kalimat yang mungkin terdengar di sebuah restoran Cina,
“Won ton, not now”
- Salah satu nomer plat mobil di Sulawesi Selatan, **“DD 151 DD”**

- Salah satu teknik untuk menguji apakah suatu kalimat, dalam hal ini adalah suatu string, merupakan palindrome atau bukan adalah dengan mengambil setiap karakter dari string ini mulai dari sisi kiri dan memasukkannya ke dalam Stack dan juga ke dalam Queue.
- Kemudian satu persatu diambil dari Stack dan juga dari Queue, lalu dibandingkan, apabila ada karakter yang tidak sama dari pengambilan ini maka kalimat ini bukan palindrome, apabila semua-nya sama maka kalimat ini palindrome.
- Satu hal yang perlu diperhatikan adalah spasi dan tanda baca diabaikan, huruf besar dan huruf kecil dianggap sama. Sebagai contoh perhatikan isi Stack dan isi Queue, ketika diisi dengan kalimat ketiga “Won ton, not now”.

Stack :

w	o	n	t	o	n	n	o	t	n	o	W
---	---	---	---	---	---	---	---	---	---	---	---

top

Queue :

w	o	n	t	o	n	n	o	t	n	o	W
---	---	---	---	---	---	---	---	---	---	---	---

rear

front

Ketika dilakukan perbandingan maka lakukan pop() terhadap Stack dan deQueue pada Queue, sehingga satu persatu huruf dibandingkan, hasilnya sama, maka kalimat ini Palindrome.

algoritma palindrome

1. Baca kalimat yang akan diperiksa, `readln(kalimat);`
2. Hitung panjangnya: `m = length(kalimat)`
3. Ulangi untuk `idx = 1 s/d m`
 - 3.1 `x = lower(subs(kalimat, idx, 1));`
 - 3.2 bila `x <> spasi & x <> tanda-baca`
maka: `push(x, S);`
`addQueue(x, Q);`
4. `palindrome = true;`
5. Ulangi untuk `idx = 1 s/d m`
 - 5.1 `pop(x1, S);`
 - 5.2 `deleteQueue(x2, Q);`
 - 5.3 bila `(x1 <> x2)`
maka: `palindrome = false;`
6. Bila (`palindrome`)
maka Cetak(`kalimat, " adalah kalimat palindrome"`);
bilatidak maka Cetak(`kalimat, " bukan kalimat
palindrome"`);

coding: palin.py

```
#palin.py => aplikasi palindrome
from ArrayQueue import ArrayQueue
from ArrayStack import ArrayStack
from Empty import Empty
import string

S=ArrayStack()          #membuat stack
Q=ArrayQueue()          #membuat queue

10 kalimat = input('Masukkan sebuah kalimat : ')
kalimat=kalimat.lower() #jadikan kalimat ini huruf kecil semua
m = len(kalimat)

for c in kalimat:        #ambil setiap huruf dari kalimat
    if (c.isalpha() or c.isdigit()):
        S.push(c)           #masukkan ke stack
        Q.enqueue(c)         #masukkan ke queue

palin = True             #anggaplah palindrome
```

```
palin = True                                #anggaplah palindrome
for i in range(m):
    if (not S.is_empty()):
        x1 = S.pop()                         # x1 ambil dari stack
        x2 = Q.dequeue()                     # x2 ambil dari queue
        if (x1 != x2):
            palin = False                   # bila x1 tidak = x2
            break                          # kalimat bukan palindrome

if (palin):
    print(kalimat, ': palindrome')
else:
    print(kalimat, ': bukan palindrome')

}
```

Contoh RUN

```
Python Interpreter
*** Remote Interpreter Reinitialized ***
Masukkan sebuah kalimat : palindrome opo ora
palindrome opo ora : bukan palindrome
>>>
*** Remote Interpreter Reinitialized ***
Masukkan sebuah kalimat : Able was I ere, I saw Elba
able was i ere, i saw elba : palindrome
>>>
*** Remote Interpreter Reinitialized ***
Masukkan sebuah kalimat : Won ton, not now
won ton, not now : palindrome
>>> |
```

Aplikasi: Hot Potato Game

- Sebuah permainan yang sering dimainkan apabila beberapa teman berkumpul, game ini disebut “Hot Potato”.
- Semua orang kumpul membentuk sebuah lingkaran, dan sebuah tongkat atau bendera dipegang oleh orang pertama, kemudian sebuah lagu dimainkan, tongkat atau bendera dioper ke orang yang ada disebelahnya, demikian seterusnya. Kemudian lagu tersebut tiba-tiba di-stop, siapa yang memegang tongkat pada saat itu, akan dikeluarkan dari lingkaran permainan (dialah hot potato).
- Kemudian lagu diputar lagi, dan permainan dilanjutkan. Demikian game ini berlangsung sampai sisa satu orang dalam permainan itu.
- Permainan ini dapat di simulasi menggunakan antrian, implementasinya sbb:

```
11 #HotPotato.py
from antrian import *

def hot_potato(name_list, num):
    sim_queue = antrian()
    for nama in name_list:
        print("%8s masuk lingkaran" % nama)
        sim_queue.enqueue(nama)

    print("\nPermainan Hot Potato di mulai")

    while sim_queue.size() > 1:
        out = sim_queue.dequeue()
        print("%8s keluar dari lingkaran" % out)

    print("Pemain terakhir adalah : %5s " % sim_queue.dequeue())

30 def main():
    print(hot_potato(["Bill","David","Susan","Jane","Kent", "Brad", "Edo"], 7))

if __name__ == '__main__':
    main()
```

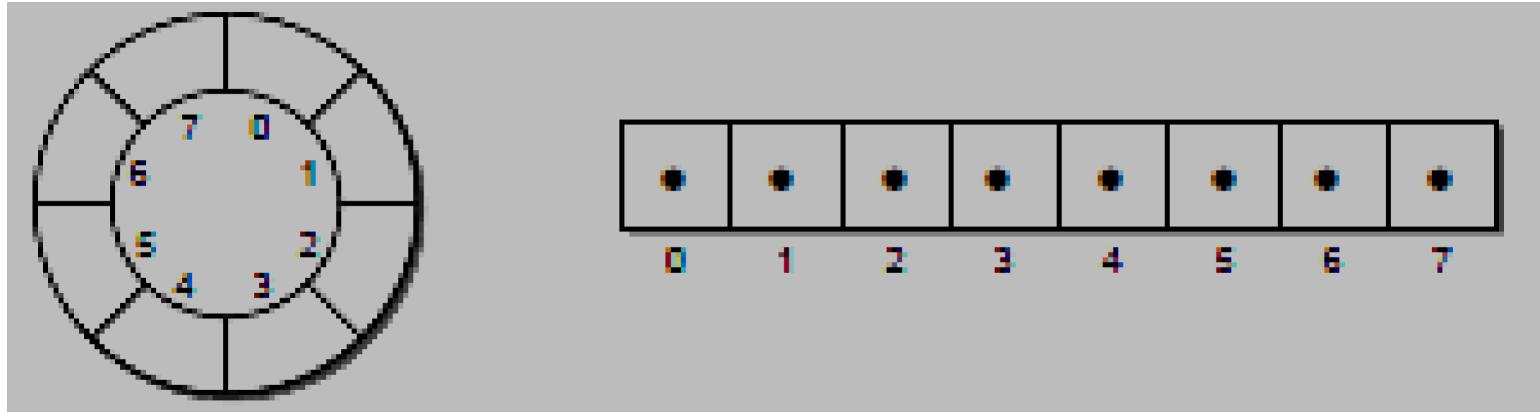
Hasil RUN

```
>>>
*** Remote Interpreter Reinitialized ***
Bill masuk lingkaran
David masuk lingkaran
Susan masuk lingkaran
Jane masuk lingkaran
Kent masuk lingkaran
Brad masuk lingkaran
Edo masuk lingkaran

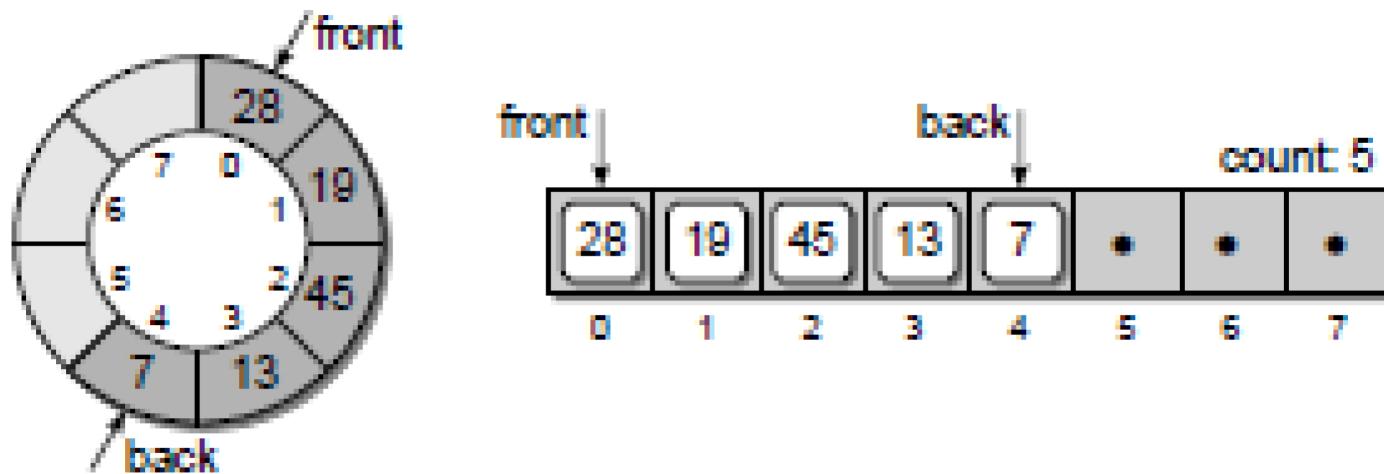
Permainan Hot Potato di mulai
Bill keluar dari lingkaran
David keluar dari lingkaran
Susan keluar dari lingkaran
Jane keluar dari lingkaran
Kent keluar dari lingkaran
Brad keluar dari lingkaran
Pemain terakhir adalah : Edo
None
>>> |
```

Circular Array untuk Antrian

- Salah satu implementasi yang unik dari antrian adalah implementasi yang memanfaatkan larik lingkaran (circular array).
- Pada implementasi ini suatu larik $Q[0:n-1]$, diberi indeks 0 hingga $(n-1)$, atau terdiri atas n buah lokasi. Ketika pointer *rear* adalah $(n-1)$ maka berarti lokasi berikutnya untuk diisi adalah $Q[0]$, bila lokasi ini kosong.

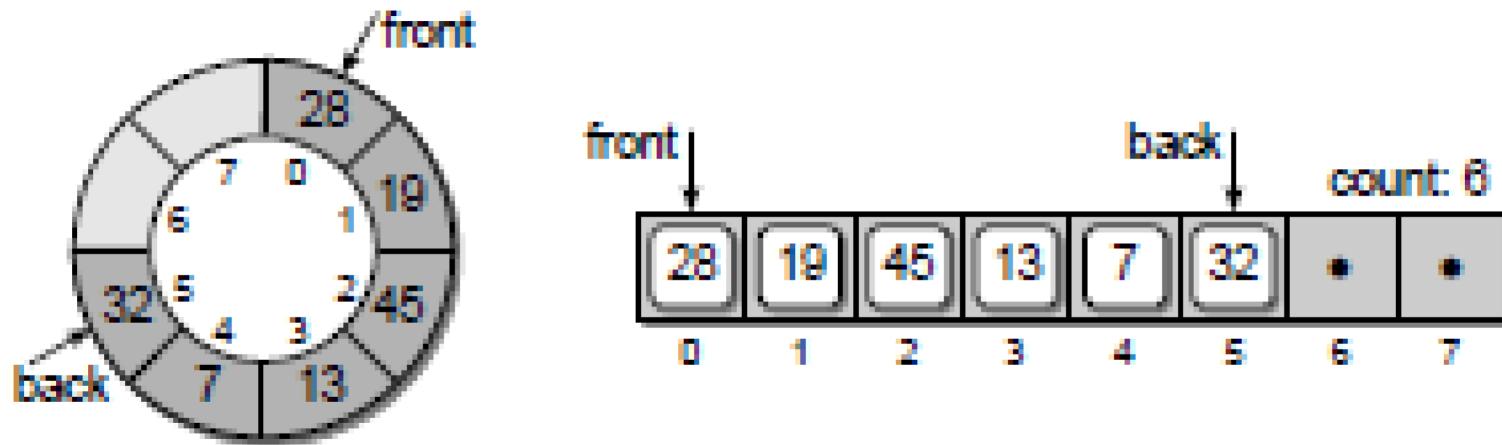


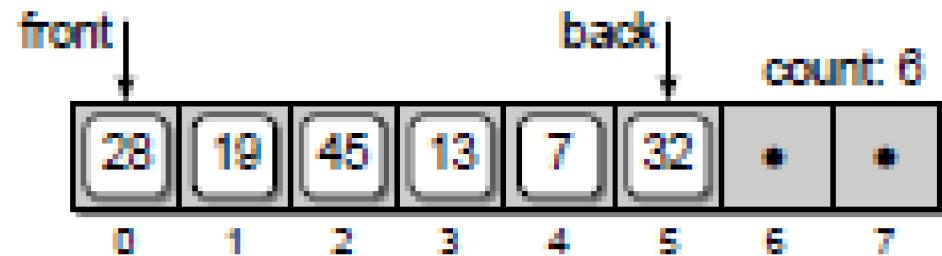
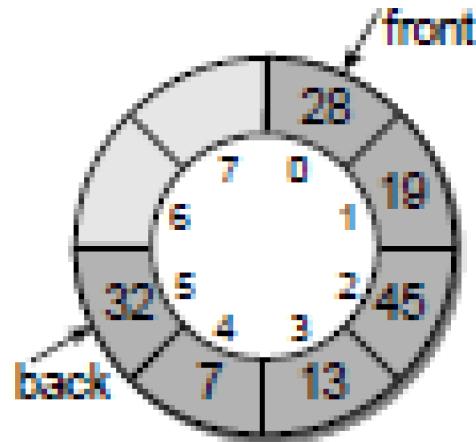
Model representasi Circular Array untuk Queue



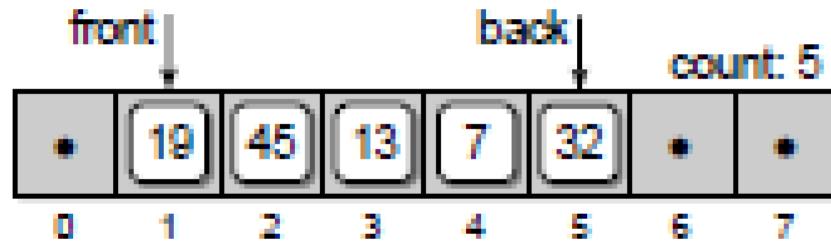
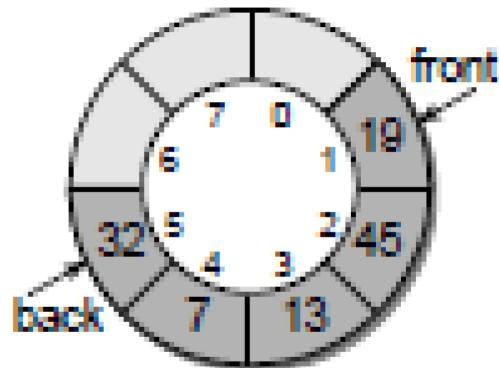
Contoh Circular Queue, n=5

Apabila kedalam circular queue ini dimasukkan nilai 32, (addCQ) maka nilai ini ditempatkan dibelakang memakai pointer back (rear). Pertama pointer back menunjuk lokasi no 5, lokasi dibelakang nilai 7, kemudian nilai 32 dimasukkan.





Nilai 28 diambil (dequeue/deleteCQ) dari antrian



Prosedur addCQ()

Prosedur addCQ(**input** x : item, **in-out** Q:queue)
{ prosedur untuk menambahkan satu elemen ke dalam
antrian melingkar }
Deklarasi
{ tidak ada }

Deskripsi

```
    Q.rear <- (Q.rear + 1) mod n;  
    if (front = rear)  
    then writeln("Antrian penuh !");  
        Exit;  
    endif;  
    Q(Q.rear) <- x;
```

Procedur deleteCQ()

Prosedur deleteCQ(out x:item, in-out Q:queue)
{ prosedur untuk mengambil satu elemen dari antrian }

Deklarasi

{ tidak ada }

Deskripsi

```
if (front = rear)
then writeln("Antrian kosong !");
      Exit;
endif;
Q.front ß (Q.front + 1) mod n;
x ßQ(Q.front);
```

Implementasi Python: CQueue.py

```
#CQueue.py => circular queue @SUARGA
from Array import Array
class CQueue :
    # Creates an empty queue.
    def __init__( self, maxSize ) :
        self._count = 0
        self._front = 0
        self._rear = maxSize - 1
        self._qArray = Array( maxSize )
    # Returns True if the queue is empty.
    def isEmpty( self ) :
        return self._count == 0
    # Returns True if the queue is full.
    def isFull( self ) :
        return self._count == len(self._qArray)
    # Returns the number of items in the queue.
    def __len__( self ) :
        return self._count
    # Adds the given item to the queue.
```

```
# Adds the given item to the queue.
def addCQ( self, item ):
    assert not self.isFull(), "Cannot enqueue to a full queue."
    maxSize = len(self._qArray)
    self._rear = (self._rear + 1) % maxSize
    self._qArray[self._rear] = item
    self._count += 1

# Removes and returns the first item in the queue.
def deleteCQ( self ):
    assert not self.isEmpty(),"Cannot dequeue from an empty queue."
    item = self._qArray[ self._front ]
    maxSize = len(self._qArray)
    self._front = (self._front + 1) % maxSize
    self._count -= 1
    return item

#mencoba cqueue
```

```
#mencoba cqueue
q=CQueue(6)
print("Memasukkan 6 elemen:")
q.addCQ(1)
q.addCQ(2)
q.addCQ(3)
q.addCQ(4)
q.addCQ(5)
q.addCQ(6)
if q.isFull():
    print('Sudah penuh!')
else:
    print('Masih ada tempat')
print("Menghapus 3 elemen")
print(q.deleteCQ())
print(q.deleteCQ())
print(q.deleteCQ())
print("Apakah CQ kosong?")
if q.isEmpty():
    print('Sudah Kosong ')
else:
    print('Masih ada isi-nya')

print("Masih ada : ")
print(q.__len__())
```

Hasil RUN

```
Python Interpreter
4
>>>
*** Remote Interpreter Reinitialized ***
Masukkan 6 elemen:
Sudah penuh!
Menghapus 3 elemen
1
2
3
Apakah CQ kosong?
Masih ada isi-nya
Masih ada :
3
>>>
```