

BAB 3. PEMROGRAMAN PYTHON TINGKAT LANJUT

Daftar Isi

- 3.1. PEMROGRAMAN DATABASE
- 3.2. WEB PROGRAMMING DENGAN FLASK
- 3.3. NETWORK PROGRAMMING
- 3.4. WEB SCRAPPING DENGAN PYTHON
- 3.5. PEMROGRAMAN GUI PADA PYTHON
- 3.6. MULTITHREADING DAN MULTIPROCESSING PADA PYTHON
- 3.7. NUMPY
- 3.8. PANDAS
- 3.9. SPEECH RECOGNITION DAN TEXT TO SPEECH DENGAN PYTHON

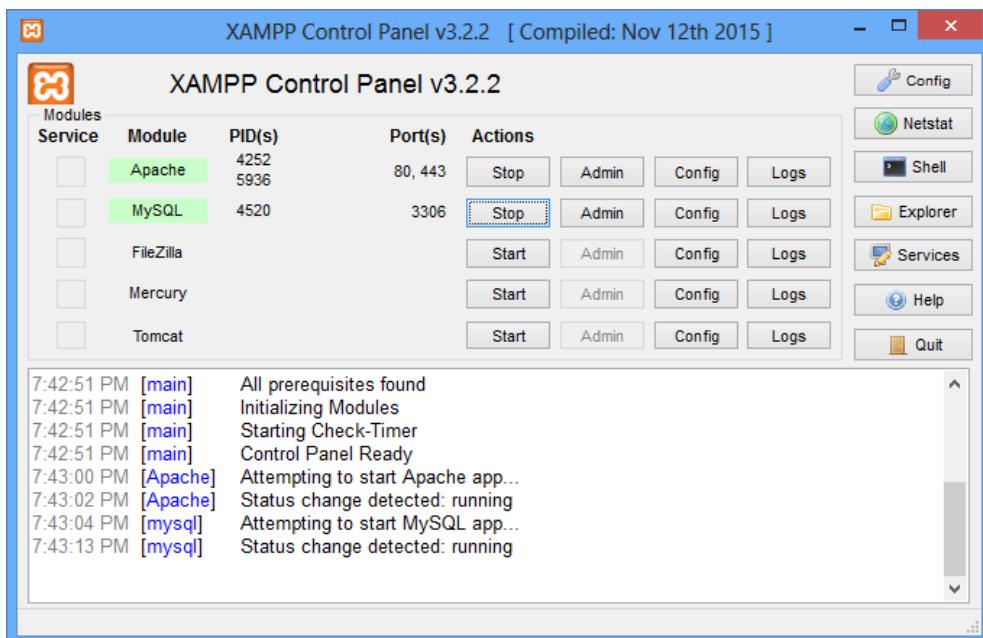
© Copyright by Antonius – www.jasaplus.com All Rights Reserved

Barangsiapa didapati memperjual belikan materi training ini tanpa seizin dari pencipta maka pencipta berhak menuntut ganti rugi yang jumlahnya ditentukan oleh pencipta materi ini.

BAB 3. PEMROGRAMAN PYTHON TINGKAT LANJUT

3.1. PEMROGRAMAN DATABASE

Sebelum lanjut pada materi kali ini, kita perlu menjalankan apache dan mysql server, pada windows bisa dijalankan dengan menjalankan xampp controller.



Jika menggunakan linux, jalankan dulu xampp dari terminal

```
cd /opt/lampp;sudo ./xampp start
```

Python mendukung berbagai macam pemrograman untuk bermacam database, pada kesempatan kali ini kita akan menggunakan python untuk pemrograman mysql. Library yang akan digunakan adalah pymysql.

Untuk menginstall library pymysql ketikkan ini :

```
pip3 install PyMySQL
```

Sebelum memulai, kita akan mempersiapkan database sederhana, buat database mysql baru melalui phpmyadmin dengan nama database pymysql1.

Selanjutnya klik sql lalu jalankan sql ini untuk membuat struktur table dan isi pada database pymysql1 :

```
CREATE TABLE `books` (
```

```
`id` int(11) NOT NULL,  
 `title` tinytext NOT NULL,  
 `author` tinytext NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
INSERT INTO `books` (`id`, `title`, `author`) VALUES  
(1, 'Good to Great', 'Jim Collins'),  
(2, 'Juniper Lemon\'s Happiness Index', 'Julie Israel'),  
(3, 'The Summer Of Impossible Things', 'By Rowan Coleman');
```

```
CREATE TABLE `movies` (  
 `id` int(11) NOT NULL,  
 `title` tinytext NOT NULL,  
 `category` tinytext NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
INSERT INTO `movies` (`id`, `title`, `category`) VALUES  
(1, 'Captain Marvel', 'action'),  
(2, 'X-men', 'action');
```

```
ALTER TABLE `books`  
 ADD PRIMARY KEY (`id`);
```

```
ALTER TABLE `movies`  
 ADD PRIMARY KEY (`id`);
```

```
ALTER TABLE `books`  
 MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=4;
```

```
ALTER TABLE `movies`  
 MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=3;COMMIT;
```

3.1.1. Menghubungan ke Database Mysql

Untuk terhubung ke database mysql,kita menggunakan perintah connect. Berikut ini contoh untuk koneksi ke database mysql , buat file python baru dengan nama koneks.py :

```
#!/usr/bin/python3  
"  
koneks.py  
Sample mysql database connection for python course material on www.jasaplus.com  
"  
import pymysql  
  
def _koneksi():  
    try:  
        con = pymysql.connect('localhost', 'root', "", 'pymysql1')  
        return con  
    except:  
        raise  
  
if __name__ == "__main__":  
    con = _koneksi()  
    print("connected to database")
```

Penjelasan

Pada source code di atas kita membuat fungsi _koneksi() yang bertujuan untuk membuat koneksi ke database mysql.

Baris ini :

```
con = pymysql.connect('localhost', 'root', '', 'pymysql1')
```

connect() merupakan fungsi di dalam class Connection, class ini merupakan salah satu class di dalam modul pymysql. Karena fungsi ini berada di dalam class Connection, maka fungsi ini disebut sebagai method dari class Connection. Method connect() digunakan untuk terhubung ke server mysql di localhost dengan username root dan password adalah kosong, di mana database yang dipilih untuk digunakan adalah pymysql1

3.1.2. Menjalankan Kueri Sql

Untuk menjalankan kueri mysql, kita bisa menggunakan fungsi execute.

Berikut ini contoh menjalankan kueri sql :

Buat file python baru dengan nama kueri.py

```
#!/usr/bin/python3
"""
kueri.py
Sample mysql database connection and query for python course material on www.jasaplus.com
"""

import pymysql

def _koneksi():
    try:
        con = pymysql.connect('localhost', 'root', '', 'pymysql1')
        return con
    except:
        raise

def _kueri(con,sql):
    try:
        with con:
            cur = con.cursor()
            cur.execute(sql)
            return cur
    except:
        raise

if __name__ == "__main__":
    con = _koneksi()
    print("connected to database")
    sql = "show tables"
    cur = _kueri(con, sql)
    rows = cur.fetchone()
    print(rows)
```

Penjelasan

Pada contoh di atas, pertama tama script akan mencoba untuk terhubung ke server database mysql :

```
con = _koneksi()
```

Selanjutnya aplikasi di atas akan mengeksekusi perintah sql : show tables.

```
sql = "show tables"  
cur = _kueri(con, sql)
```

Di mana fungsi `_kueri` adalah fungsi yang kita buat sendiri untuk wrapper fungsi di `pymysql`. Pada fungsi `_kueri` :

```
cur = con.cursor()
```

kita akan mencoba mengambil objek berupa `Cursor`. Di mana objek ini merupakan objek dari `pymysql` yang berguna untuk berinteraksi dengan database `mysql`. Objek ini memiliki fungsi `execute` yang berguna untuk mengeksekusi kueri sql.

Pada baris selanjutnya kita mengeksekusi perintah sql dengan method `execute()` :

```
cur.execute(sql)
```

3.1.3. Menampilkan Data

Untuk menampilkan data, kita akan melakukan select data, kemudian kita melakukan fetch (pengambilan data). Untuk mengambil hanya 1 baris data kita bisa menggunakan fungsi fetchone() , sedangkan untuk mengambil seluruh data menggunakan objek Cursor, kita bisa menggunakan fungsi fetchall().

Berikut ini adalah contoh aplikasi untuk menampilkan seluruh data di tabel books. Buat file baru dengan nama view.py

```
#!/usr/bin/python3
"""
view.py
Sample mysql database app for python course material on www.jasaplus.com
"""

import pymysql

def _koneksi():
    try:
        con = pymysql.connect('localhost', 'root', "", 'pymysql1')
        return con
    except:
        raise

def _kueri(con,sql):
    try:
        with con:
            cur = con.cursor()
            cur.execute(sql)
            return cur
    except:
        raise

con = _koneksi()
print("connected to database")
sql = "select * from `books`"
cur = _kueri(con, sql)
rows = cur.fetchall()
for row in rows:
    print(row)
```

Ketika aplikasi ini dijalankan maka akan menampilkan data dari tiap baris di dalam tabel books dalam bentuk tuple :

```
python3 view.py

connected to database
(1, 'Good to Great', 'Jim Collins')
(2, "Juniper Lemon's Happiness Index", 'Julie Israel')
(3, 'The Summer Of Impossible Things', 'By Rowan Coleman')
```

Pada contoh di atas, kita menampilkan data keseluruhan dalam bentuk tuple. Misal kita ingin menampilkan hanya data pada kolom title di tabel books. Berikut ini adalah contoh kedua :

```
#!/usr/bin/python3
"""
```

```

view2.py
Sample mysql database connection and query for python course material on www.jasaplus.com
"""

import pymysql

def _koneksi():
    try:
        con = pymysql.connect('localhost', 'root', '', 'pymysql1')
        return con
    except:
        raise

def _kueri(con,sql):
    try:
        with con:
            cur = con.cursor()
            cur.execute(sql)
            return cur
    except:
        raise

if __name__ == "__main__":
    con = _koneksi()
    print("connected to database")
    sql = "select * from `books`"
    cur = _kueri(con, sql)
    rows = cur.fetchall()
    print(type(rows))
    for row in rows:
        print(row[1])
        print(type(row))

```

Penjelasan

Pada contoh di atas, pertama tama kita terhubung ke server mysql dan melakukan select database pymysql1 dengan fungsi _koneksi(), di mana di dalam fungsi _koneksi() terdapat rutin untuk terhubung ke mysql server :

```
con = pymysql.connect('localhost', 'root', '', 'pymysql1')
```

Selanjutnya kita mengeksekusi mysql melalui fungsi _kueri() :

```
cur = con.cursor()
cur.execute(sql)
```

Fungsi _kueri akan menghasilkan return value berupa objek Cursor pymysql. Untuk mengambil seluruh data dengan menggunakan objek Cursor, kita akan menggunakan method fetchall() dari objek Cursor.

```
rows = cur.fetchall()
```

Selanjutnya hasil dari fetchall akan tersimpan pada tuple rows kita tinggal melakukan looping untuk menampilkan data dari tuple rows :

```
for row in rows:
    print(row[1])
```

Pada contoh kedua kita menampilkan data dari tuple dengan index ke 1. Perhatikan, berikut ini adalah struktur tabel books:

		+ Options	← →	▼	id	title	author
<input type="checkbox"/>	 Edit	 Copy	 Delete	1	Good to Great		Jim Collins
<input type="checkbox"/>	 Edit	 Copy	 Delete	2	Juniper Lemon's Happiness Index		Julie Israel
<input type="checkbox"/>	 Edit	 Copy	 Delete	3	The Summer Of Impossible Things	By Rowan Coleman	

Pada tabel books, id akan disimpan pada indeks tuple ke 0, title akan disimpan pada indeks tuple ke 1 dan author akan disimpan pada indeks tuple ke 2.

3.1.4. Menghapus Data

Untuk menghapus data, caranya adalah dengan mengeksekusi perintah sql delete. Berikut ini contoh aplikasi yang digunakan untuk menghapus data. Buat file baru dengan nama delete.py

```
#!/usr/bin/python3
"""
delete.py
Sample mysql database connection and query for python course material on www.jasaplus.com
"""

import pymysql

def _koneksi():
    try:
        con = pymysql.connect('localhost', 'root', "", 'pymysql1')
        return con
    except:
        raise

def _kueri(con,sql):
    try:
        with con:
            cur = con.cursor()
            cur.execute(sql)
            return cur
    except:
        raise

con = _koneksi()
print("connected to database")
sql = "delete from `books` where `id` like '4'"
cur = _kueri(con, sql)
```

Penjelasan

Pada contoh di atas, untuk menghapus data kita menggunakan statement sql : delete, lengkapnya :

```
delete from `books` where `id` like '4'
```

Pada contoh di atas, kita menghapus data di table books yang memiliki id 4.

Untuk mengeksekusi statement sql tersebut, kita gunakan fungsi wrapper _kueri dengan 2 parameter berupa objek koneksi mysql dan string sql.

```
cur = _kueri(con, sql)
```

Fungsi _kueri adalah fungsi wrapper untuk method execute :

```
with con:  
    cur = con.cursor()  
    cur.execute(sql)
```

Pada contoh di atas, pertama tama kita ambil objek class pymysql.cursors.Cursor di class tersebut terdapat method execute. Method execute digunakan untuk mengeksekusi statement sql delete yang telah disiapkan pada string variable sql.

3.1.5. Mengupdate Data

Untuk mengupdate data, caranya adalah dengan mengeksekusi perintah sql update.

Berikut ini contoh aplikasi yang digunakan untuk mengupdate data. Buat file baru dengan nama update.py

```
#!/usr/bin/python3  
""  
update.py  
Sample mysql database connection and query for python course material on www.jasaplus.com  
""  
import pymysql  
  
def _koneksi():  
    try:  
        con = pymysql.connect('localhost', 'root', "", 'pymysql1')  
        return con  
    except:  
        raise  
  
def _kueri(con,sql):  
    try:  
        with con:  
            cur = con.cursor()  
            cur.execute(sql)  
            return cur  
    except:  
        raise  
  
if __name__ == "__main__":  
    con = _koneksi()  
    print("connected to database")  
    sql = "update `books` set `title` = 'New Title' where `id` like '4'"  
    cur = _kueri(con, sql)
```

Penjelasan

Pada contoh di atas, pertama tama kita terhubung ke server mysql dan melakukan select database pymysql1 dengan fungsi _konek(), di mana di dalam fungsi _konek() terdapat rutin untuk terhubung ke mysql server :

```
con = pymysql.connect('localhost', 'root', '', 'pymysql1')
```

Fungsi _kueri() merupakan fungsi wrapper untuk mengeksekusi method execute yang akan digunakan untuk mengeksekusi sql untuk mengupdate data :

```
def _kueri(con,sql):
    try:
        with con:
            cur = con.cursor()
            cur.execute(sql)
            return cur
    except:
        raise
```

Pada thread utama, kita jalankan fungsi fungsi yang telah kita siapkan sebelumnya, di mana pertama tama kita melakukan koneksi ke database mysql dengan fungsi _konek() , selanjutnya kita mempersiapkan variabel string berisi statement untuk mengeksekusi perintah sql.

```
if __name__ == "__main__":
    con = _konek()
    print("connected to database")
    sql = "update `books` set `title` = 'New Title' where `id` like '4'"
    cur = _kueri(con, sql)
```

Pada baris ini :

```
cur = _kueri(con, sql)
```

kita menjalankan fungsi _kueri yang berguna menjalankan statement sql yang telah kita siapkan di atasnya yang berguna untuk mengupdate data dengan perintah sql update.

3.1.6. Menutup Koneksi Database

Untuk menutup koneksi ke server mysql, caranya adalah dengan memanggil fungsi close pada objek koneksi pymysql.

Berikut ini contoh aplikasi yang membuka dan kemudian menutup koneksi , buat file baru dengan nama tutup.py .

```
#!/usr/bin/python3
"""
tutup.py
Sample mysql database connection for python course material on www.jasaplus.com
"""

import pymysql

def _koneksi():
    try:
        con = pymysql.connect('localhost', 'root', '', 'pymysql1')
        return con
    except:
        raise
if __name__ == "__main__":
    con = _koneksi()
    print("connected to database")
    print("closing connection ... ")
    con.close()
```

Penjelasan

```
con = _koneksi()
```

_koneksi() merupakan fungsi wrapper untuk method connect untuk membuka koneksi dengan database. Untuk menutup koneksi dengan database kita gunakan method close() :

```
con.close()
```

3.2. WEB PROGRAMMING DENGAN FLASK

Flask adalah kerangka kerja aplikasi web mikro yang ditulis dalam bahasa pemrograman Python. Flask banyak digunakan untuk membuat aplikasi aplikasi web professional.

Untuk menginstall flask, gunakan pip :

```
pip install flask
```

3.2.1. Contoh Flask Hello World

Pada contoh kali ini, kita akan membuat aplikasi flask sederhana untuk menampilkan hello world di browser.

Buat file baru dengan nama hello_world.py :

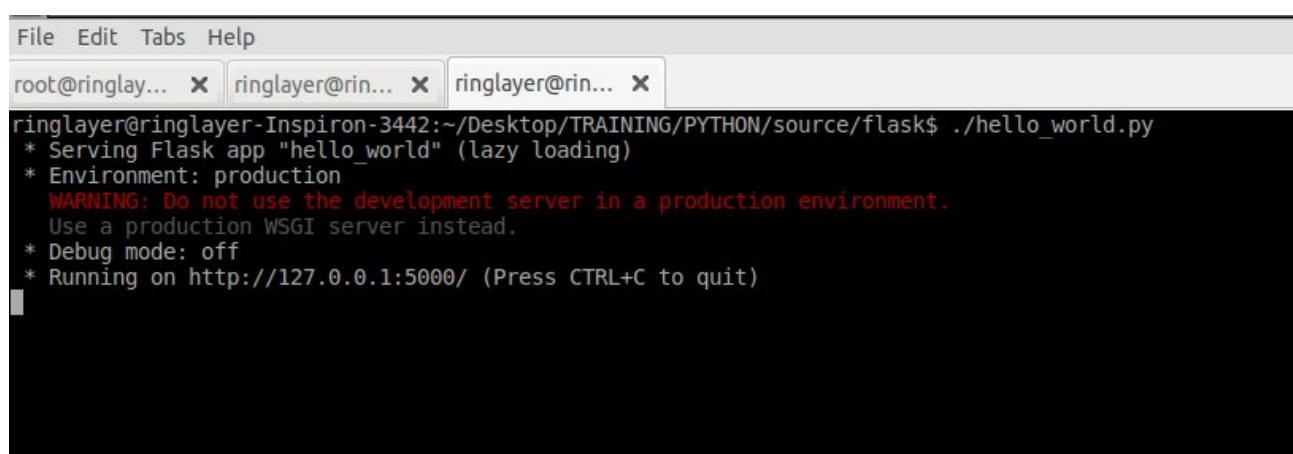
```
#!/usr/bin/env python3

from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello World!"

if __name__ == "__main__":
    app.run()
```

Jalankan script di atas :



The screenshot shows a terminal window with three tabs at the top: 'File', 'Edit', 'Tabs', and 'Help'. The active tab is 'Tabs'. Below the tabs, there are three terminal sessions:

- Session 1: 'root@ringlayer... X' - This session is inactive.
- Session 2: 'ringlayer@ringlayer... X' - This session is active.
- Session 3: 'ringlayer@ringlayer... X' - This session is inactive.

The active session (Session 2) displays the command and its output:

```
ringlayer@ringlayer-Inspiron-3442:~/Desktop/TRAINING/PYTHON/source/flask$ ./hello_world.py
 * Serving Flask app "hello_world" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Penjelasan

```
from flask import Flask
```

Pada baris ini kita mengimport class Flask dari modul flask

```
app = Flask(__name__)
```

Pada baris ini kita membuat instance objek dari class Flask dengan nama app

```
@app.route("/")
```

Merupakan suatu method yang berfungsi mendaftarkan suatu fungsi untuk decorator pada saat suatu path url diakses. Pada contoh di atas, kita memberikan route untuk path “/”

```
def hello():
    return "Hello World!"
```

Fungsi hello akan dipanggil saat http request pada path /, fungsi ini diregister oleh route(“/”)

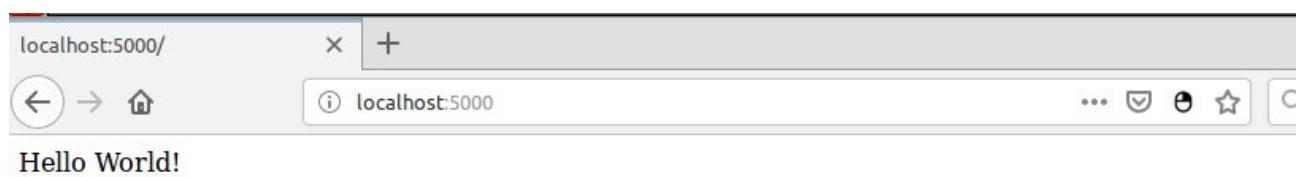
```
if __name__ == "__main__":
    app.run()
```

run() merupakan method yang digunakan menjalankan servis aplikasi flask, pada contoh di atas, kita tidak memberikan parameter nomor port yang digunakan sehingga flask akan melisten pada port default dengan nomor port 5000.

Untuk menguji jalanya aplikasi tadi, buka browser dan ketikkan :

<http://localhost:5000>

Tampilanya :



3.2.2. Menambahkan Routing untuk Request URL

Pada contoh selanjutnya kita akan menambahkan beberapa routing untuk url yang diakses dan penggunaan custom port.

Buat script baru dengan nama flask2.py :

```
#!/usr/bin/env python3
"""
flask2.py

sample flask for python3 training
www.jasaplus.com
www.ringlayer.com
"""

from flask import Flask
app = Flask(__name__)

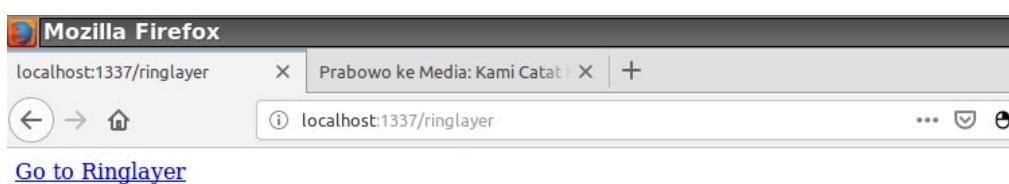
@register a decorator for uri : /
@app.route("/")
def hello():
    return "Main Page"

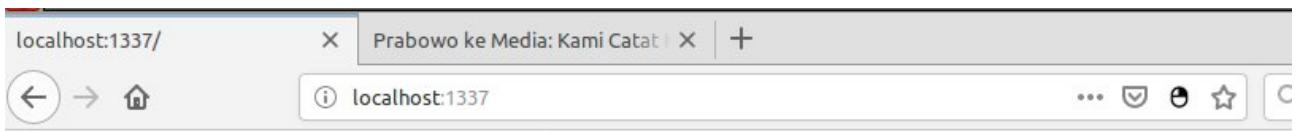
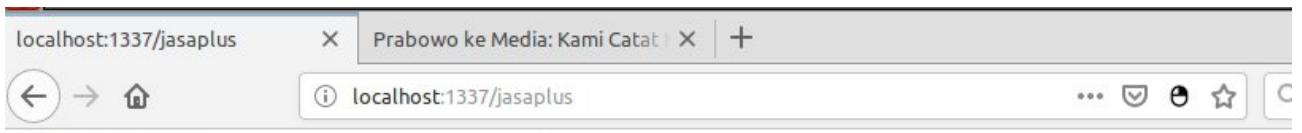
@register a decorator for uri : /jasaplus
@app.route("/jasaplus")
def ProcJasaplus():
    return "<a href='https://www.jasaplus.com'>Go to Jasaplus</a>"

@register a decorator for uri : /ringlayer
@app.route("/ringlayer")
def ProcRing():
    return "<a href='https://www.ringlayer.com'>Go to Ringlayer</a>"

if __name__ == "__main__":
    app.run(port=1337)
```

Jalankan script di atas, berikut ini contoh contoh akses dengan browser :





3.2.3. Contoh Flask Halaman Web Sederhana

Pada contoh selanjutnya kita akan menampilkan halaman web html sederhana dengan flask. Untuk merender suatu template web, kita gunakan method render_template()

Pada contoh kali ini, kita akan membuat struktur direktori sederhana untuk proyek flask kali ini

```
web1
  flaskweb.py
  static/
    css/
    js/
    images/
  templates/
    home.html
    about.html
    contact.html
```

Pada direktori css digunakan untuk menyimpan file file css. Direktori js digunakan untuk menyimpan file file java script, direktori images digunakan menyimpan images.

Direktori templates digunakan menyimpan file file html. Pada direktori utama terdapat file script python dengan nama flaskweb.py

Source code lengkap untuk aplikasi ini bisa didownload di github repository untuk training ini (alamat ada pada BAB 1)

Berikut ini source code lengkap flaskweb.py :

```
#!/usr/bin/env python3
"""
flaskweb.py

sample flask for python3 training
www.jasaplus.com
www.ringlayer.com
"""

from flask import Flask, render_template
app = Flask(__name__)

@register a decorator for uri : /
@app.route("/")
def Home():
    return render_template("home.html")

@register a decorator for uri : /about
@app.route("/about")
def About():
    return render_template("about.html")

@register a decorator for uri : /contact
@app.route("/contact")
```

```
def Contact():
    return render_template("contact.html")

if __name__ == "__main__":
    app.run(port=8888)
```

Penjelasan

```
from flask import Flask, render_template
```

Pada baris ini kita mengimport class flask.app.Flask dari modul flask dan mengimport fungsi render_template dari modul flask

```
app = Flask(__name__)
```

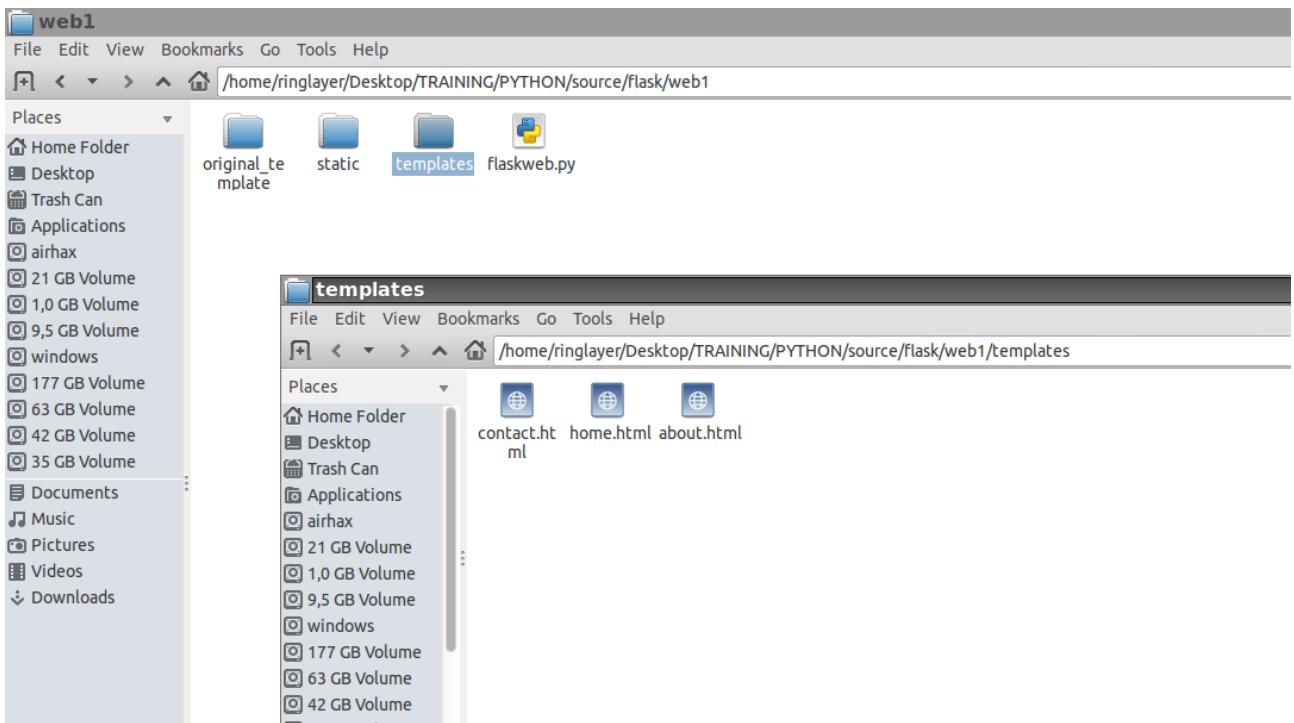
Pada baris ini kita membuat instance objek dari class Flask dengan nama app

```
#register a decorator for uri : /
@app.route("/")
def Home():
    return render_template("home.html")

#register a decorator for uri : /about
@app.route("/about")
def About():
    return render_template("about.html")

#register a decorator for uri : /contact
@app.route("/contact")
def Contact():
    return render_template("contact.html")
```

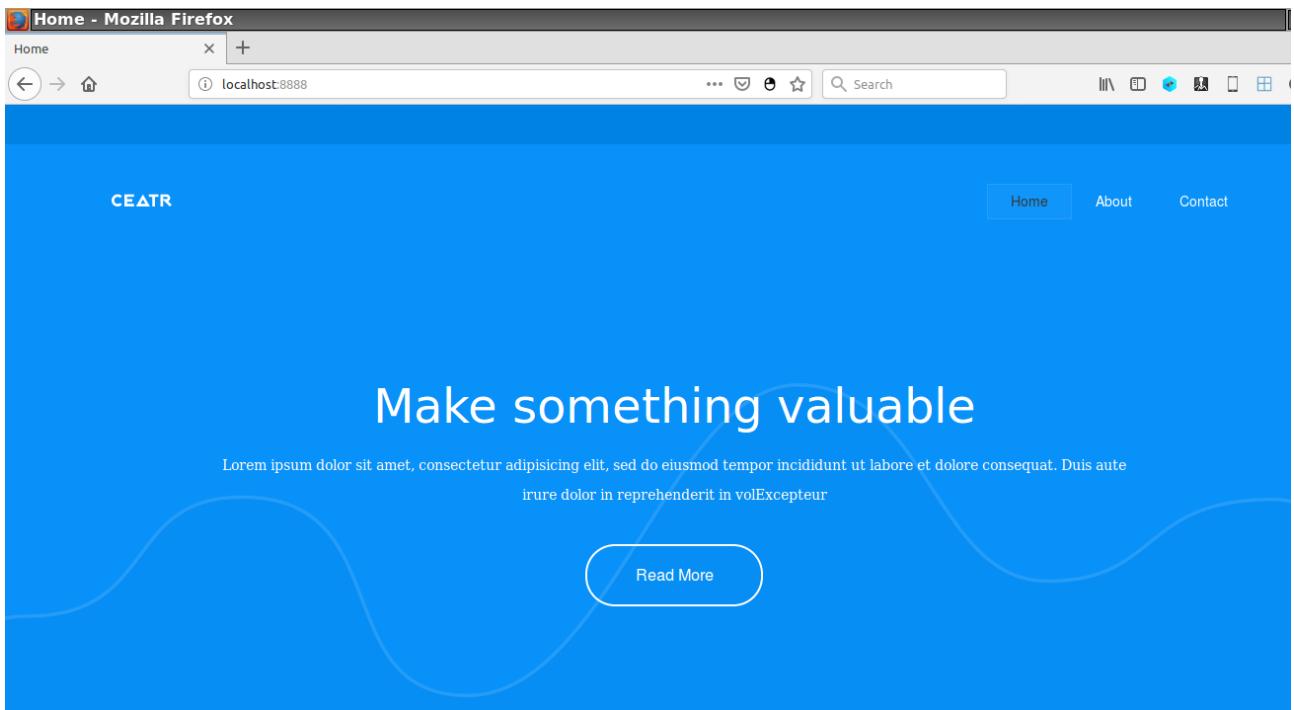
Pada baris baris di atas, kita melakukan register fungsi fungsi decorator untuk route pada beberapa url yaitu : / , /about , /contact . Di mana masing masing decorator berfungsi untuk melakukan render template yang telah disiapkan pada direktori templates.



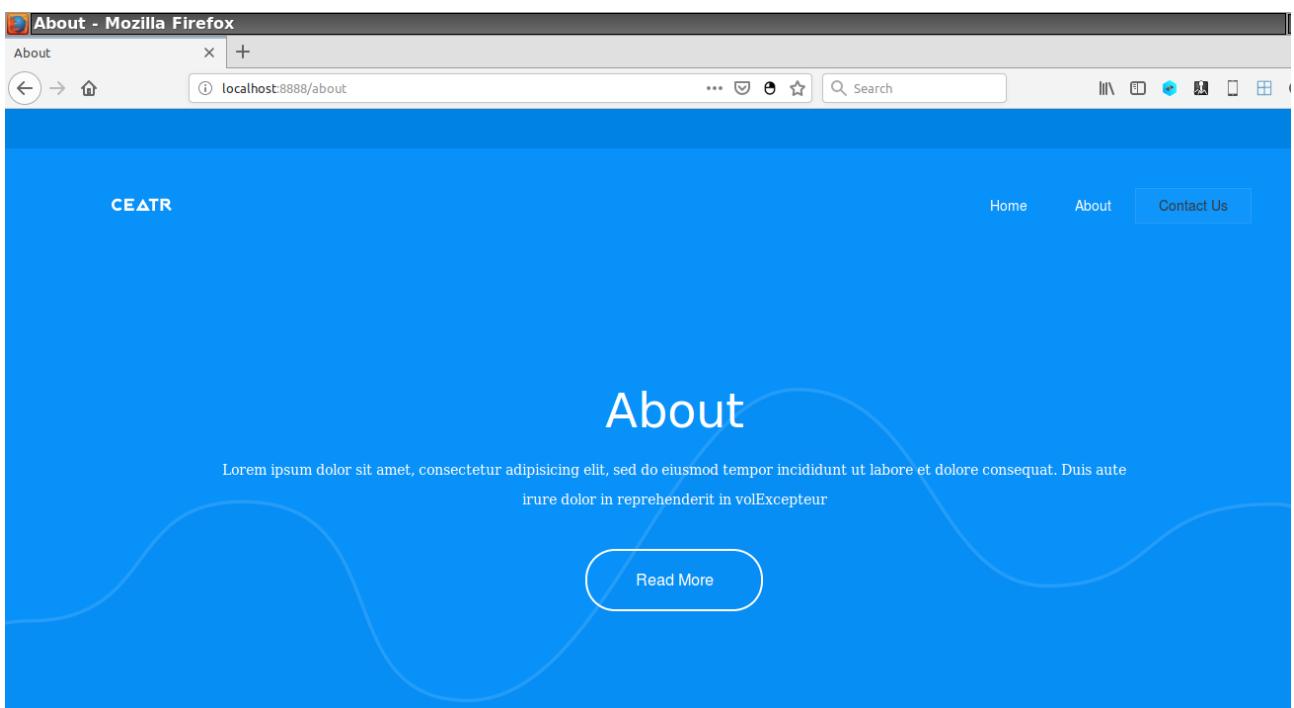
Untuk menguji script di atas, jalankan dan akses lewat browser di port 8888

```
ringlayer@ringlayer-Inspiron-3442:~/Desktop/TRAINING/PYTHON/source/flask/web1$ ls
flaskweb.py original_template static templates
ringlayer@ringlayer-Inspiron-3442:~/Desktop/TRAINING/PYTHON/source/flask/web1$ ./flaskweb.py
 * Serving Flask app "flaskweb" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:8888/ (Press CTRL+C to quit)
```

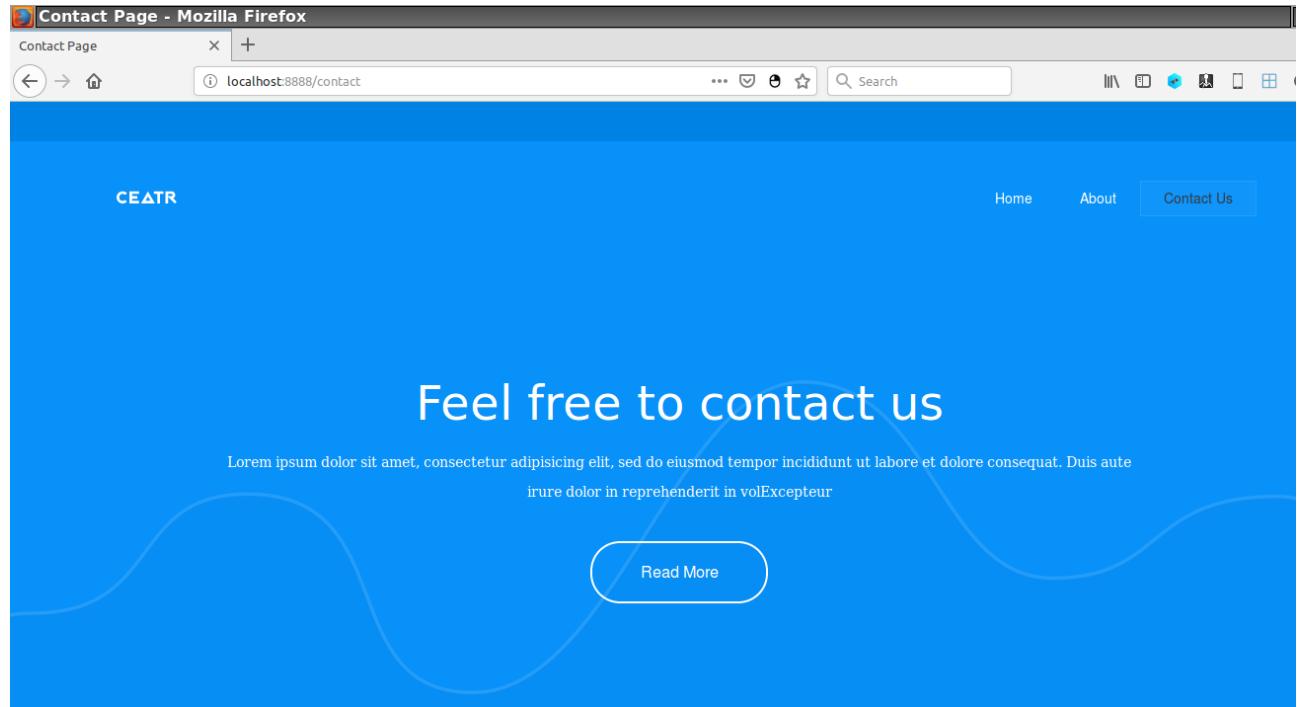
Hasilnya jika diakses pada browser, maka template yang telah disiapkan akan dirender :



Uji coba routing untuk url : about



Uji coba routing untuk url / contact



terlihat script python berhasil merender template yang telah kita siapkan sebelumnya

3.2.4. Contoh Aplikasi Flask dengan MySQL

Pada contoh kali ini, kita akan membuat aplikasi web dengan database mysql. Sebelum memulai, buat database baru melalui phpmyadmin dengan nama database : flask

Import sql flask.sql berikut ini ke database flask melalui phpmyadmin (bagian eksekusi sql)

```
CREATE TABLE `page` (
  `id` int(11) NOT NULL,
  `header` text NOT NULL,
  `container` text NOT NULL,
  `mode` tinytext NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
INSERT INTO `page` (`id`, `header`, `container`, `mode`) VALUES
(1, 'About Ringlayer', 'I\'m a roboticist based in Indonesia.\r\nmy expertises are : robotic, programming, digital electronic, computer vision and artificial intelligence (machine learning for robotic).\r\n  
\r\nThis site was developed using python3 and flask, source code can be cloned from github.\r\n  
\r\nhttps://github.com/ringlayerIf you wish to become one of my friend, you can follow me on twitter : <a href=https://www.twitter.com/ringlayer>@ringlayer</a>.\r\n  
\r\nAll right ! Hopefully you didn\'t think this site as a ludicrous site..Have a nice day buddy !\r\n', 'about'),
(2, 'Contact Me', 'Contact Me on Github : @ringlayer\r\n  
\r\nThank you \r\n  
\r\nMr Ringlayer\r\n', 'kontak'),
(3, 'Portfolio', '<b>Portfolio page coming soon !!!</b>', 'portfolios');
```

```
ALTER TABLE `page`
  ADD PRIMARY KEY (`id`);
ALTER TABLE `page`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=4;COMMIT;
```

The screenshot shows the phpMyAdmin interface for the 'flask' database. The left sidebar lists databases like 'flask', 'geopos', 'hax', 'herb', 'hova', 'hovacake_2014', 'indispanel', 'indotalent', 'inem', 'information_schema', 'input', 'inten2', and 'ires'. The main area shows the 'Structure' tab for the 'page' table. The table has 3 rows and the following schema:

Table	Action	Rows	Type	Collation	Size	Overhead
page	Browse Structure Search Insert Empty Drop	3	InnoDB	latin1_swedish_ci	16 KiB	-
Sum		3	InnoDB	latin1_swedish_ci	16 KiB	0 B

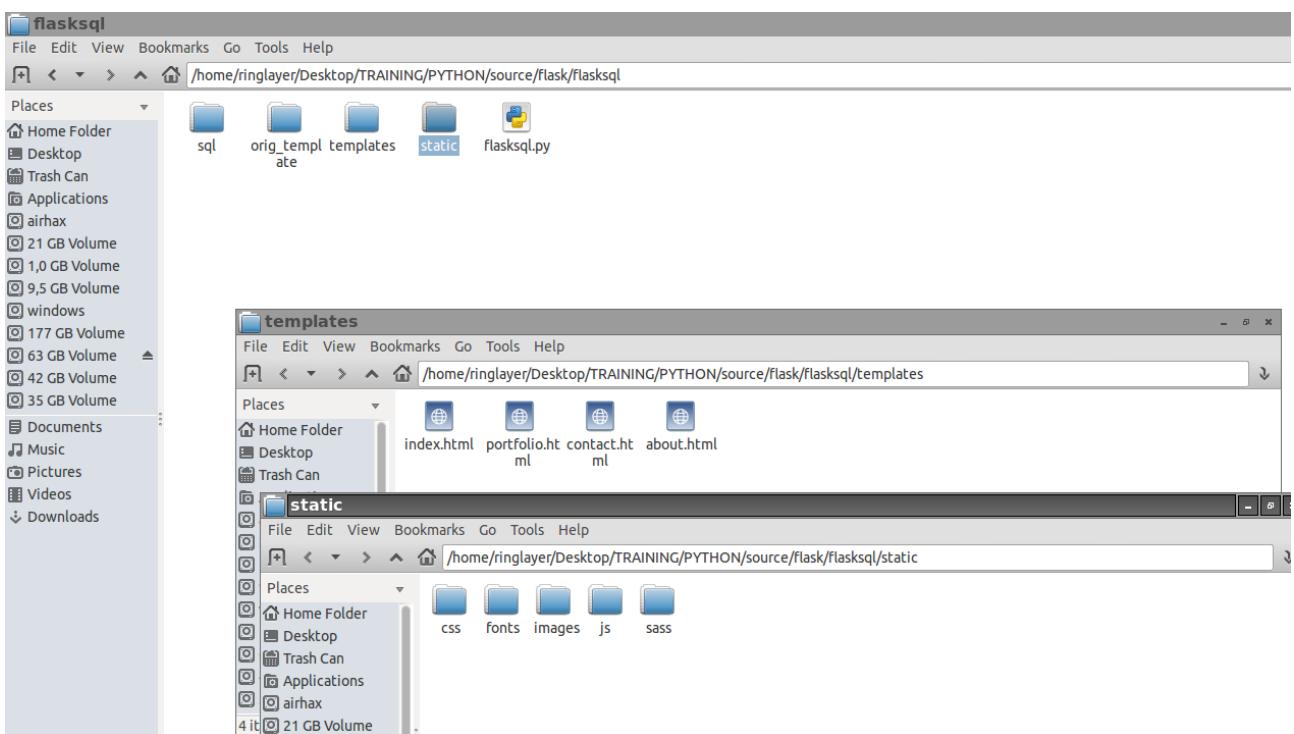
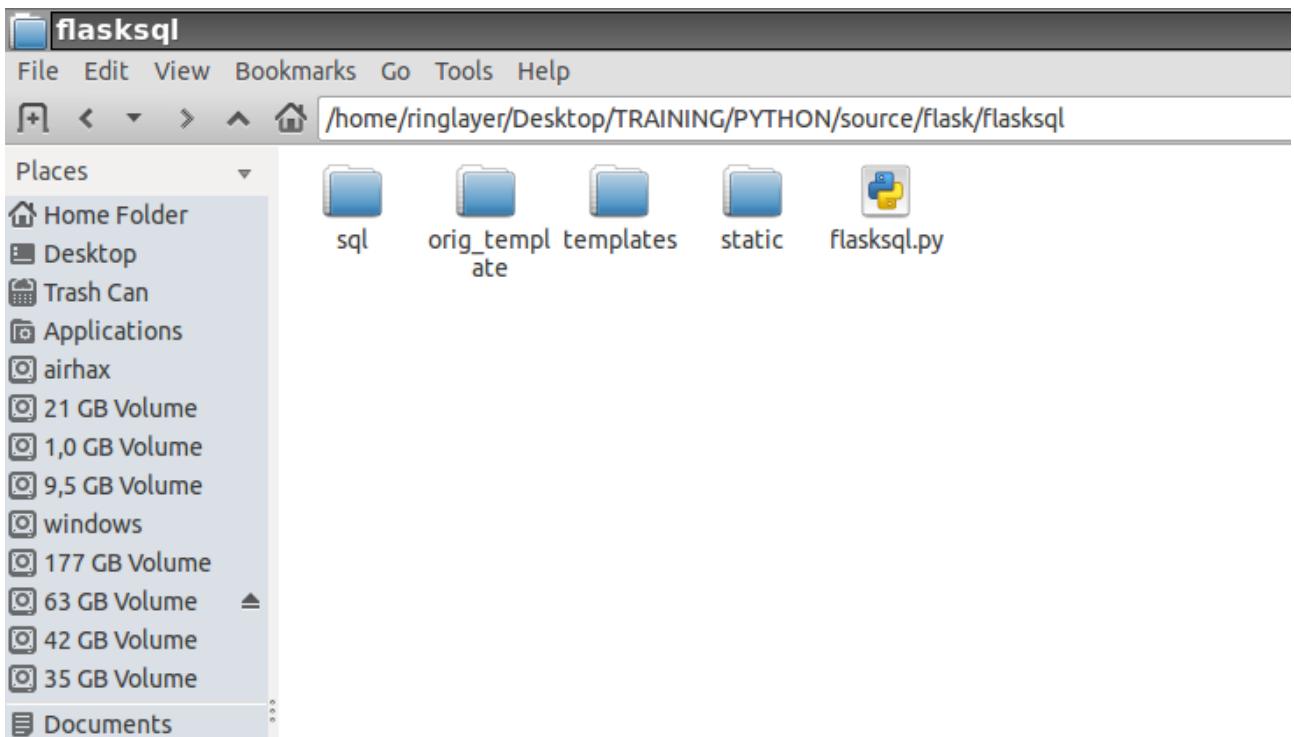
The screenshot shows the phpMyAdmin interface for the 'flask' database. The left sidebar lists databases like 'flask', 'geopos', 'hax', 'herb', 'hova', 'hovacake_2014', 'indispelan', 'indotent', 'inem', 'information_schema', 'input', 'inten2', and 'ires'. The 'flask' database is selected. Inside 'flask', the 'page' table is selected. The table has four columns: 'id', 'header', 'container', and 'mode'. There are three rows:

	id	header	container	mode
<input type="checkbox"/>	1	About Ringlayer	I'm a robotist based in Indonesia. my expertise...	about
<input type="checkbox"/>	2	Contact Me	<ul class="list-unstyled"> <a href...	kontak
<input type="checkbox"/>	3		<div class="container"> <div class="sec...">	portfolios

Pada database flask di atas, kita memiliki table page dengan 4 kolom : id, header, container dan mode.

Untuk proyek, kali ini akan memiliki struktur yang tidak jauh berbeda dari proyek flask sebelumnya

```
flasksql
    flasksql.py
    static/
        css/
        js/
        images/
    templates/
        home.html
        about.html
        contact.html
```



Untuk alasan keamanan, jika script di atas ingin diinstall dan dijalankan di server, direktori sql harus dihapus.

Source code lengkap aplikasi ini bisa didownload di github repo untuk training ini (alamat github ada pada BAB 1)

Berikut ini source script utama flasksql.py :

```
#!/usr/bin/env python3
"""
flaskweb.py

sample flask for python3 training
www.jasaplus.com
www.ringlayer.com
"""

import pymysql
from flask import Flask, render_template
app = Flask(__name__)

#start global default vars
HeaderAbout = "about"
ContainerAbout = "welcome to about"

HeaderKontak = "contact"
ContainerKontak= "welcome to contact"

HeaderPortfolio = "portfolio"
ContainerPortfolio = "welcome to portfolio"
#eof global default vars

def _koneksi():
    try:
        con = pymysql.connect('localhost', 'root', '', 'flask')
        return con
    except:
        raise

def _kueri(con,sql):
    try:
        with con:
            cur = con.cursor()
            cur.execute(sql)
            return cur
    except:
        raise

def _get_all_page():
    try:
        global HeaderAbout,ContainerAbout, HeaderKontak, ContainerKontak, HeaderPortfolio, ContainerPortfolio
        con = _koneksi()
        sql = "select header, container, mode from `page`"
        cur = _kueri(con, sql)
        rows = cur.fetchall()
        for row in rows:
            cur_header = row[0]
            cur_container = row[1]
            cur_mode = row[2]
            if cur_mode == "about":
                HeaderAbout = cur_header
                ContainerAbout = cur_container
            elif cur_mode == "kontak":
                HeaderKontak = cur_header
                ContainerKontak = cur_container
            elif cur_mode == "portfolios":
```

```

HeaderPortfolio = cur_header
ContainerPortfolio = cur_container
except:
    raise

#register a decorator for uri : /
@app.route("/")
def Home():
    header = "Home"
    container = "Welcome to my Home"
    return render_template("index.html", header_home=header, container_home=container)

#register a decorator for uri : /about
@app.route("/about")
def About():
    return render_template("about.html", header_about = HeaderAbout , container_about = ContainerAbout)

#register a decorator for uri : /contact
@app.route("/contact")
def Contact():
    return render_template("contact.html", header_contact = HeaderKontak, container_contact = ContainerKontak)

#register a decorator for uri : /portfolio
@app.route("/portfolio")
def Portfolio():
    return render_template("portfolio.html", header_portfolio = HeaderPortfolio, container_portfolio =
ContainerPortfolio)

if __name__ == "__main__":
    _get_all_page()
    app.run(port=8888)

```

Penjelasan

Bagian penting pada template :

Misal pada template contact.html, terdapat sisipan tag :

```
<h1 class="no-margin">{{ header_contact }}</h1>
<p>{{ container_contact|safe }}</p>
```

di mana {{ header_contact }} dan {{ container_contact|safe }} merupakan tag yang kita sisipkan agar saat template dirender, kita bisa menyisipkan nilai ke variabel header_contact dan container_contact . Khusus untuk container contact, kita tambahkan !safe agar auto escape terhadap tag html dari database tidak terjadi saat render (karena isi kolom container di table page pada database merupakan html).

Cara kerja script di atas, pada dasarnya pada saat pertama kali aplikasi dijalankan, sebelum melakukan listen akan melakukan pembacaan seluruh entry yang dibutuhkan untuk ditampilkan dari database untuk disimpan ke variabel variabel global yang telah disiapkan sebelumnya :

```
def _get_all_page():
    try:
```

```

global HeaderAbout, ContainerAbout, HeaderKontak, ContainerKontak, HeaderPortfolio, ContainerPortfolio
con = _koneksi()
sql = "select header, container, mode from `page`"
cur = _kueri(con, sql)
rows = cur.fetchall()
for row in rows:
    cur_header = row[0]
    cur_container = row[1]
    cur_mode = row[2]
    if cur_mode == "about":
        HeaderAbout = cur_header
        ContainerAbout = cur_container
    elif cur_mode == "kontak":
        HeaderKontak = cur_header
        ContainerKontak = cur_container
    elif cur_mode == "portfolios":
        HeaderPortfolio = cur_header
        ContainerPortfolio = cur_container
except:
    raise

```

Sebagai contoh pada salah satu decorator, saat fungsi render_template, di sini ditambahkan parameter berupa variabel yang akan ditampilkan di template.

Variabel header_contact dan container_contact terdapat pada html di template, di sini kita isikan nilainya dengan string variabel global yang telah kita inisialisasi sebelumnya (data diambil dari database mysql)

```

@register a decorator for uri : /contact
@app.route("/contact")
def Contact():
    return render_template("contact.html", header_contact = HeaderKontak, container_contact = ContainerKontak)

```

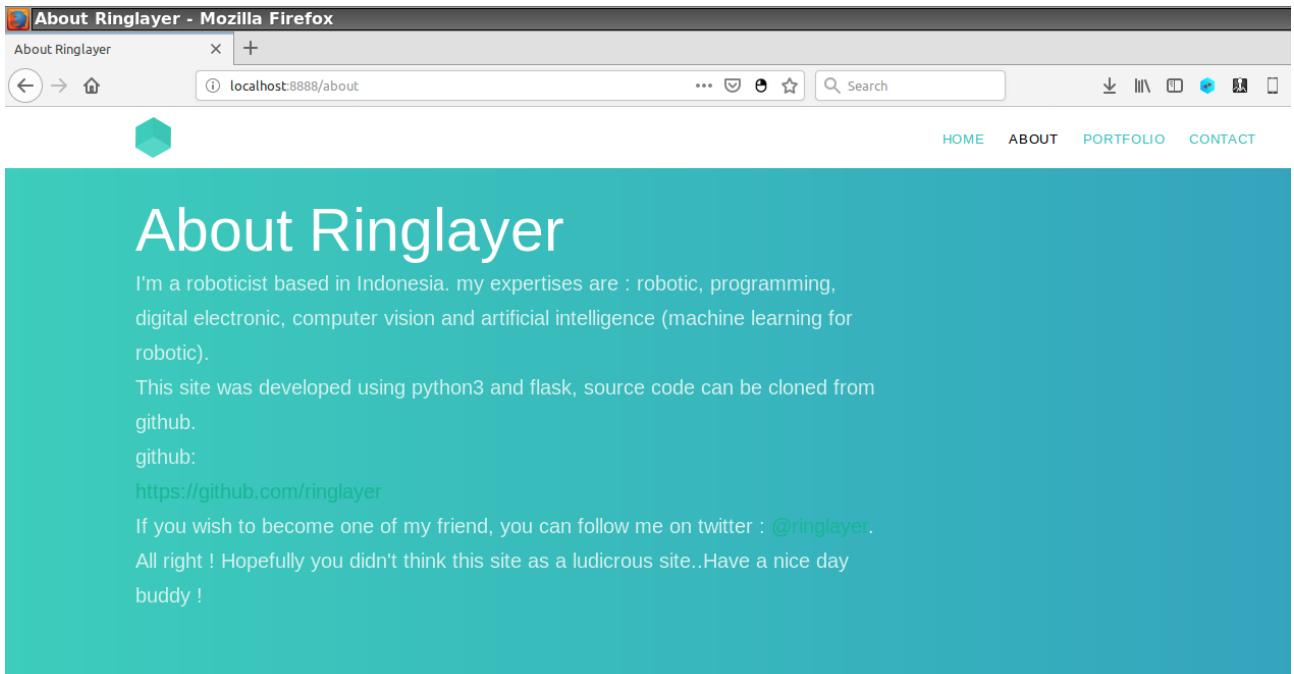
Jalankan script di atas

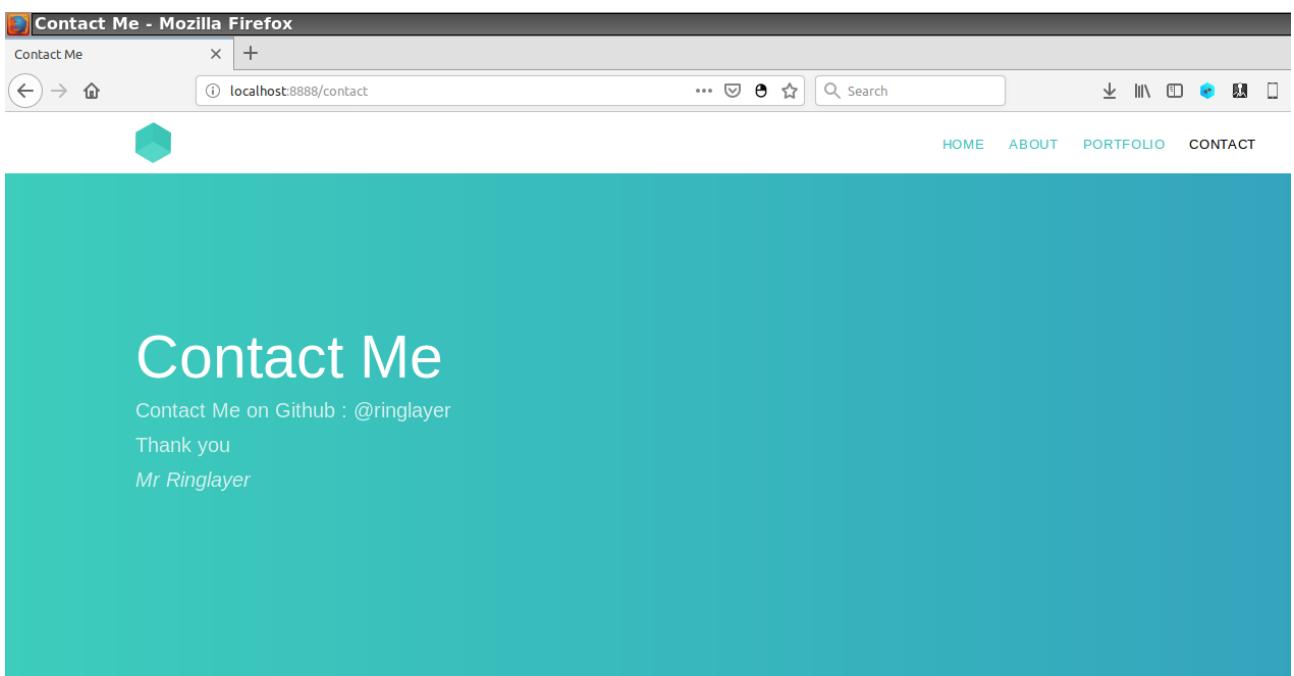
```

ringlayer@ringlayer-Inspiron-3442:~/Desktop/TRAINING/PYTHON/source/flask/flasksql$ ./flasksql.py
Serving Flask app "flasksql" (lazy loading)
Environment: production
WARNING: Do not use the development server in a production environment.
Use a production WSGI server instead.
Debug mode: off
Running on http://127.0.0.1:8888/ (Press CTRL+C to quit)
7.0.0.1 - - [03/May/2019 20:49:08] "GET /about HTTP/1.1" 200 -
7.0.0.1 - - [03/May/2019 20:49:08] "GET /static/css/animate.css HTTP/1.1" 304 -
7.0.0.1 - - [03/May/2019 20:49:08] "GET /static/css/icomoon.css HTTP/1.1" 304 -
7.0.0.1 - - [03/May/2019 20:49:08] "GET /static/css/themify-icons.css HTTP/1.1" 304 -
7.0.0.1 - - [03/May/2019 20:49:08] "GET /static/css/bootstrap.css HTTP/1.1" 304 -
7.0.0.1 - - [03/May/2019 20:49:08] "GET /static/css/owl.carousel.min.css HTTP/1.1" 304 -
7.0.0.1 - - [03/May/2019 20:49:08] "GET /static/css/magnific-popup.css HTTP/1.1" 304 -
7.0.0.1 - - [03/May/2019 20:49:08] "GET /static/css/owl.theme.default.min.css HTTP/1.1" 304 -
7.0.0.1 - - [03/May/2019 20:49:08] "GET /static/css/style.css HTTP/1.1" 304 -
7.0.0.1 - - [03/May/2019 20:49:08] "GET /static/js/modernizr-2.6.2.min.js HTTP/1.1" 304 -

```

Berikut ini beberapa contoh hasil tampilan saat routing :

A screenshot of a Mozilla Firefox browser window. The title bar says "About Ringlayer - Mozilla Firefox". The address bar shows "localhost:8888/about". The main content area has a teal background and features a large white hexagonal logo at the top. Below it, the text "About Ringlayer" is displayed in a large, bold, white sans-serif font. Underneath the heading, there is a paragraph of white text about the author's expertise in robotics, programming, digital electronics, computer vision, and artificial intelligence. It also mentions that the site was developed using Python3 and Flask, and provides a GitHub link for cloning the source code. The text ends with a friendly message encouraging users to follow the author on Twitter and GitHub.

A screenshot of a Mozilla Firefox browser window. The title bar says "Contact Me - Mozilla Firefox". The address bar shows "localhost:8888/contact". The main content area has a teal background and features a large white hexagonal logo at the top. Below it, the text "Contact Me" is displayed in a large, bold, white sans-serif font. Underneath the heading, there is a paragraph of white text providing contact information: "Contact Me on Github : @ringlayer", "Thank you", and "Mr Ringlayer".

3.3. NETWORK PROGRAMMING

Pemrograman jaringan adalah pembuatan aplikasi yang digunakan untuk proses komunikasi dengan komputer atau perangkat lain pada jaringan.

Pada training kali ini, kita akan menggunakan library socket pada python.

3.3.1. TCP Klien dan Server

Pada contoh pertama kali ini, kita akan membuat aplikasi tcp klien dan server dengan python.

TCP Server

Pada dasarnya untuk membuat suatu tcp server kita akan melalui serangkaian rutin di bawah ini :

1. Membuat socket

Langkah pertama adalah membuat socket descriptor. Kita akan membuat instance object dari class socket pada modul socket :

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

2. Bind

Bind digunakan untuk menghubungkan file descriptor yang sudah disiapkan dengan alamat ip lokal di server tersebut.

```
s.bind((HOST, PORT))
```

3. Listen

Listen digunakan untuk menandai socket yang telah kita siapkan sebelumnya telah siap menerima koneksi.

```
s.listen()
```

4. Accept

Accept digunakan untuk menerima koneksi dari klien ke server.

```
conn, addr = s.accept()
```

5. Melakukan handle koneksi

Untuk melakukan handle koneksi, terdapat beberapa fungsi dasar, yang sering dipakai adalah fungsi recv dan send.

Berikut ini adalah contoh tcp server dengan python3, nama file server.py

```

#!/usr/bin/python3
"""

server.py
Sample tcp server for python course material on www.jasaplus.com
"""

import socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
host = "127.0.0.1"
port = 1337
s.bind((host, port))
print("listening on ", port)
s.listen()
conn,addr = s.accept()
print("Got connection from" + str(addr))
while True:
    msg = 'Server Message' + "\n"
    conn.send(msg.encode('ascii'))
    request = conn.recv(4096)
    if request:
        print("[+] Message from client : ", repr(request.decode('utf-8')))
    conn.close()

```

Penjelasan

```

import socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

```

Di sini kita import modul socket dengan import socket, lalu menjalankan method socket untuk membuat socket file descriptor dengan parameter AF_INET (address family) dan SOCK_STREAM untuk tcp (untuk udp menggunakan SOCK_DGRAM),

```
s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
```

Method ini digunakan untuk memberikan opsi tambahan pada socket file descriptor, di sini kita set parameter untuk set socket options dengan SOL_SOCKET untuk memanipulasi opsi socket pada level API socket, parameter kedua menggunakan SO_REUSEADDR yang merupakan konstan untuk mengizinkan server melakukan bind ulang pada alamat ip lokal dan port yang sama jika server direstart.

```

s.bind((host, port))
s.listen()
conn,addr = s.accept()

```

Pada baris ini, kita lakukan bind, kemudian listen dan accept. Di sini server sudah siap menerima koneksi tcp.

```

while True:
    msg = 'Server Message' + "\n"
    conn.send(msg.encode('ascii'))
    request = conn.recv(4096)
    if request:
        print("[+] Message from client : ", repr(request.decode('utf-8')))

```

Pada rutin rutin ini diadakan pengulangan untuk menangani koneksi dari klien : mengirim dan menangkap pesan dari klien. Pada bagian con.send pesan dikirim setelah diencode ke bentuk ASCII.

Pada bagian menerima pesan klien, pesan ditampilkan setelah didecode ke dalam bentuk string utf8.

Untuk informasi tentang ascii dan utf-8 bisa dibaca di wifi :

<https://en.wikipedia.org/wiki/ASCII>

<https://en.wikipedia.org/wiki/UTF-8>

TCP Client

Tcp client merupakan aplikasi python pada node klien yang berguna untuk menghubungkan / berkomunikasi dengan aplikasi tcp server yang dijalankan pada server.

Berikut ini adalah beberapa method yang akan digunakan oleh tcp client pada contoh di bawah:

1. Membuat socket

Langkah pertama adalah membuat socket descriptor. Kita akan membuat instance object dari class socket :

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

2. Menghubungkan dengan Server

Untuk terhubung ke server kita gunakan method connect.

Contoh :

```
sock.connect((host, port))
```

3. Menerima Data dari Server

Untuk menerima paket data dari server, kita gunakan method recv.

Contoh :

```
response = sock.recv(4096)
```

Pada contoh di atas, kita menerima data sebanyak 4096 bytes.

4. Mengirim Data ke Server

Untuk mengirim data ke server pada contoh di bawah ini, kita gunakan method sendall.

Contoh :

```
sock.sendall(data.encode('utf-8'))
```

5. Menutup Socket File Descriptor

Untuk menutup socket file descriptor (mengakhiri koneksi), kita gunakan method close(). Contoh dari object instance sock (hanya contoh nama identifier) :

```
sock.close()
```

Untuk contoh aplikasi lengkap, buat file baru dengan nama client.py :

```
#!/usr/bin/python3
"""
client.py
Sample tcp client for python course material on www.jasaplus.com
"""

import socket, time
host = "127.0.0.1"
port = 1337
data = "client message"
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.connect((host, port))

while True:
    response = sock.recv(4096)
    if response:
        print("[+] From Server : ", repr(response.decode('utf-8')))
        sock.sendall(data.encode('utf-8'))
        time.sleep(2)
    sock.close()
```

Penjelasan

```
import socket, time
host = "127.0.0.1"
port = 1337
data = "client message"
```

Pada rutin di atas, kita impor terlebih dahulu modul modul yang dibutuhkan, selanjutnya kita setting alamat ip dan port server yang akan dikontak dan pengaturan pesan yang akan kita kirimkan.

```
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.connect((host, port))
```

Pada rutin di atas, koneksi ke server diinisialisasi.

```
while True:
    response = sock.recv(4096)
    if response:
        print("[+] From Server : ", repr(response.decode('utf-8')))
        sock.sendall(data.encode('utf-8'))
        time.sleep(2)
```

Pada rutin di atas, dilakukan looping untuk menerima data dan mengirim data ke server.

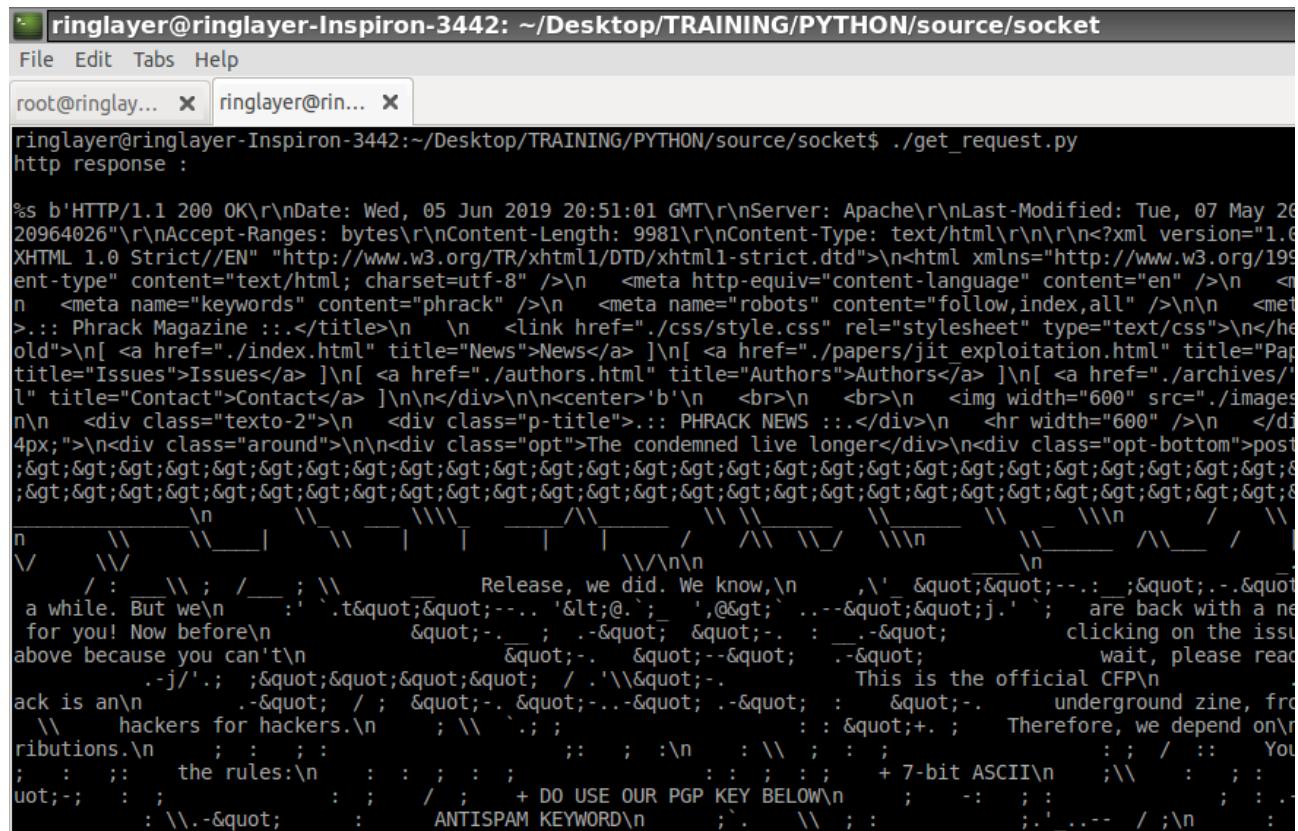
3.3.2. Contoh Aplikasi Klien dengan Socket

Pada contoh kali ini kita akan membuat contoh aplikasi klien untuk get http request ke suatu alamat web. Buat aplikasi baru dengan nama get_request.py :

```
#!/usr/bin/env python3
"""
sample http get request client for python training course
at www.jasaplus.com
"""

import socket
target_host = "www.phrack.org"
target_port = 80
sockfd = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sockfd.connect((target_host,target_port))
request = "GET /index.html HTTP/1.1\r\nHost: www.phrack.org\r\n\r\n"
sockfd.sendall(request.encode())
full_source = ""
response = sockfd.recv(4096)
while True:
    full_source += repr(response)
    response = sockfd.recv(4096)
    if not response:
        break
print("http response : \n\n%s", full_source)
```

Hasil dari menjalankan aplikasi di atas :



```
ringlayer@ringlayer-Inspiron-3442: ~/Desktop/TRAINING/PYTHON/source/socket
File Edit Tabs Help
root@ringlay... x ringlayer@rin... x
ringlayer@ringlayer-Inspiron-3442:~/Desktop/TRAINING/PYTHON/source/socket$ ./get_request.py
http response :

%> b'HTTP/1.1 200 OK\r\nDate: Wed, 05 Jun 2019 20:51:01 GMT\r\nServer: Apache\r\nLast-Modified: Tue, 07 May 2019 04:26:40 GMT\r\nAccept-Ranges: bytes\r\nContent-Length: 9981\r\nContent-Type: text/html\r\n<?xml version="1.0 encoding="UTF-8"?><!DOCTYPE html PUBLIC "-//IETF//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd"><html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en"><head><meta name="keywords" content="phrack" /><meta name="robots" content="follow,index,all" /><title>Phrack Magazine</title><link href=".css/style.css" rel="stylesheet" type="text/css"/><head><div class="text-2"><div class="p-title">PHRACK NEWS</div><hr width="600" /><div class="around"><div class="opt">The condemned live longer</div><div class="opt-bottom">post issues</div></div></div><br><br><p>Release, we did. We know, we are back with a new issue! Now before clicking on the links above because you can't wait, please read the rules: the rules: + DO USE OUR PGP KEY BELOW + 7-bit ASCII + ANTI-SPAM KEYWORD</p>
```

Terlihat hasil dari request di atas akan menampilkan source code lengkap dari web beralamat di phrack.org.

Penjelasan

```
import socket
```

pada baris ini kita mengimport modul socket

```
target_host = "www.phrack.org"  
target_port = 80
```

pada kedua baris ini kita menyiapkan variabel target_host dan target_port untuk koneksi dengan socket file descriptor.

```
sockfd = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  
sockfd.connect((target_host,target_port))
```

pada kedua baris di atas, kita membuat socket file descriptor kemudian kita lakukan connect

```
request = "GET /index.html HTTP/1.1\r\nHost: www.phrack.org\r\n\r\n"
```

pada kedua baris di atas, kita mengirimkan http get request ke server.

```
full_source = ""  
response = sockfd.recv(4096)  
while True:  
    full_source += repr(response)  
    response = sockfd.recv(4096)  
    if not response:  
        break  
print("http response : \n\n%s", full_source)
```

baris baris di atas berguna untuk menangkap hasil response server dan menampilkannya.

Aplikasi di atas biasanya berguna untuk mengambil data json dari suatu alamat url di mana data json itu bisa diproses lebih lanjut dengan python.

3.3.3. Contoh Aplikasi Klien dengan Request

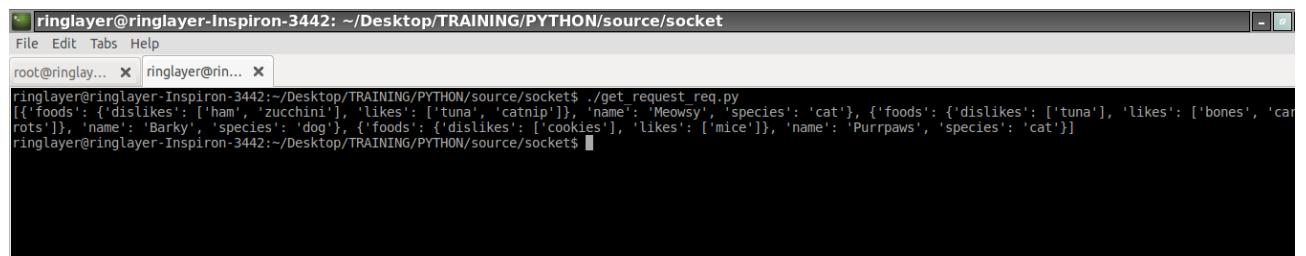
Penggunaan socket di atas pada dasarnya adalah penggunaan low level operation untuk membuat http request ke server web. Untuk memudahkan pembuatan http request client, kita bisa menggunakan modul python requests, bisa diinstall dengan pip :

```
pip3 install requests
```

Berikut ini contoh aplikasi get http request dengan menggunakan modul requests :

```
#!/usr/bin/env python3
import requests
r = requests.get('https://raw.githubusercontent.com/LearnWebCode/json-example/master/animals-1.json')
print(r.json())
```

Simpan file di atas dengan nama get_request_req.py, Berikut ini hasil dari menjalankan aplikasi di atas :



```
ringlayer@ringlayer-Inspiron-3442: ~/Desktop/TRAINING/PYTHON/source/socket
File Edit Tabs Help
root@ringlay... x ringlayer@rin... x
ringlayer@ringlayer-Inspiron-3442:~/Desktop/TRAINING/PYTHON/source/socket$ ./get_request_req.py
[{"foods": {"dislikes": ["ham", "zucchini"], "likes": ["tuna", "catnip"]}, "name": "Meowsy", "species": "cat"}, {"foods": {"dislikes": ["tuna"], "likes": ["bones", "carrots"]}, "name": "Barky", "species": "dog"}, {"foods": {"dislikes": ["cookies"], "likes": ["mice"]}, "name": "Purrraws", "species": "cat"}]
ringlayer@ringlayer-Inspiron-3442:~/Desktop/TRAINING/PYTHON/source/socket$
```

terlihat kita berhasil mendapatkan data json dari url.

3.3.4. Konkurensi pada Klien

Konkurensi artinya adalah secara bersama sama. Konkurensi pada suatu klien misal suatu klien yang melakukan get request pada beberapa alamat url web bisa dilakukan dengan modul python seperti octopus-http.

Kita bisa menginstall modul ini dengan pip :

```
pip3 install octopus-http
```

Berikut ini contoh penggunaan octopus-http untuk pengambilan data json dari 2 alamat url berbeda secara bersamaan :

```
#!/usr/bin/env python3
"""
octo.py
sample octopus http for python3 training
at www.jasaplus.com
"""
from octopus import Octopus
```

```

otto = Octopus(
    concurrency=2, auto_start=True
)

def handle_url_response(url, response):
    print(response.text)

otto.enqueue('https://raw.githubusercontent.com/brandiqa/json-examples/master/src/google_markers.json',
handle_url_response)
otto.enqueue('https://raw.githubusercontent.com/brandiqa/json-examples/master/src/products.json',
handle_url_response)
otto.wait()

```

Simpan dengan nama octo.py, berikut ini hasil script di atas :

```

ringlayer@ringlayer-Inspiron-3442:~/Desktop/TRAINING/PYTHON/source/socket$ ./octo.py
{
  "markers": [
    {
      "name": "Rixos The Palm Dubai",
      "position": [25.1212, 55.1535],
    },
    {
      "name": "Shangri-La Hotel",
      "location": [25.2084, 55.2719]
    },
    {
      "name": "Grand Hyatt",
      "location": [25.2285, 55.3273]
    }
  ]
}

[{
  "_id": {
    "$oid": "5968dd23fc13ae04d9000001"
  },
  "product_name": "sildenafil citrate",
  "supplier": "Wisozk Inc",
  "quantity": 261,
  "unit_cost": "$10.47"
}, {
  "_id": {
    "$oid": "5968dd23fc13ae04d9000002"
  },
  "product_name": "Mountain Juniperus ashei",
  "supplier": "Keebler-Hilpert",
  "quantity": 292
}]

```

Penjelasan

```

from octopus import Octopus

otto = Octopus(
    concurrency=2, auto_start=True
)

```

Baris di atas berguna untuk mengimport modul lalu membuat instance object dari class Octopus

```
def handle_url_response(url, response):
    print(response.text)
```

Fungsi di atas berfungsi sebagai caller untuk memproses hasil dari http response dari octopus.

```
otto.enqueue('https://raw.githubusercontent.com/brandiqa/json-examples/master/src/google_markers.json',
            handle_url_response)
otto.enqueue('https://raw.githubusercontent.com/brandiqa/json-examples/master/src/products.json',
            handle_url_response)
otto.wait()
```

Baris baris di atas digunakan melakukan http request secara konkuren pada 2 alamat url json.

3.4. WEB SCRAPPING DENGAN PYTHON

Pada python, web scrapping adalah teknik yang digunakan untuk mengambil data dari halaman website. Library yang akan kita gunakan kali ini adalah selenium.

Ada beberapa driver yang bisa digunakan untuk web scrapping dengan selenium. Pada contoh kali ini, kita akan menggunakan driver chromedriver.

3.4.1. Persiapan

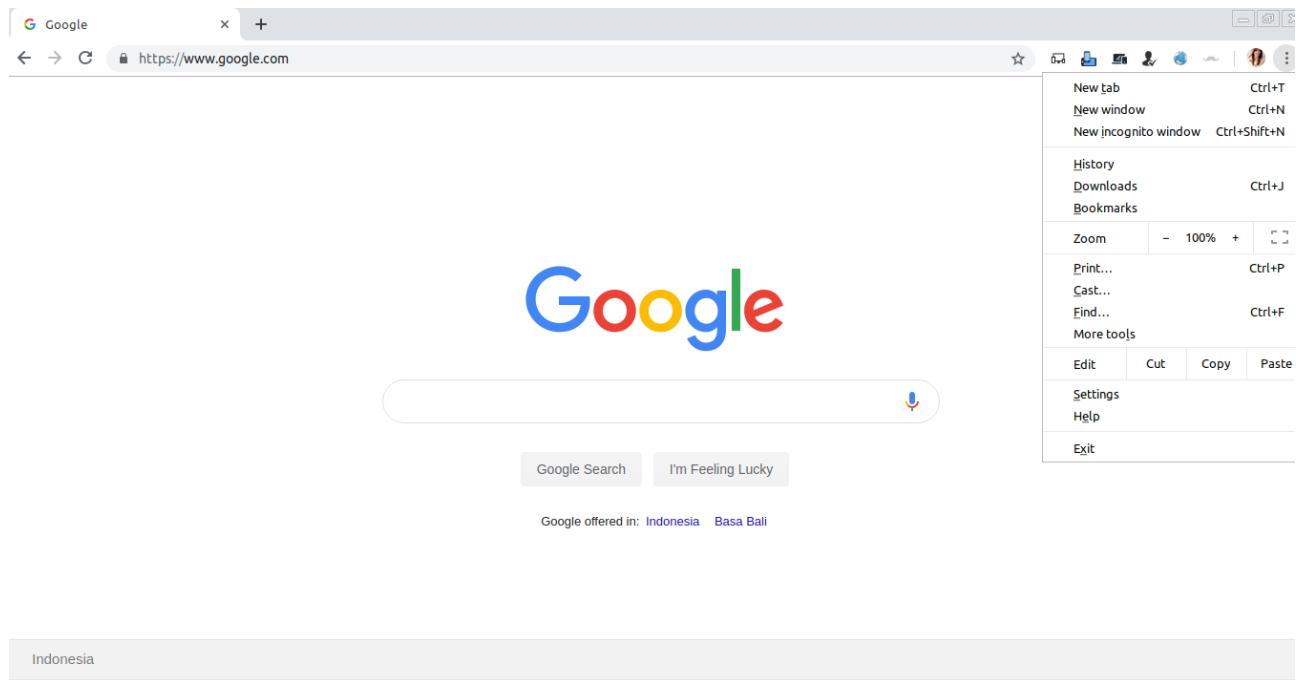
Sebelum menggunakan chromedriver, kita perlu menginstall terlebih dahulu google chrome jika belum ada.

Pastikan Anda menyesuaikan versi chromedriver yang didownload dengan versi google chrome yang terinstall dan jenis sistem operasi yang digunakan.

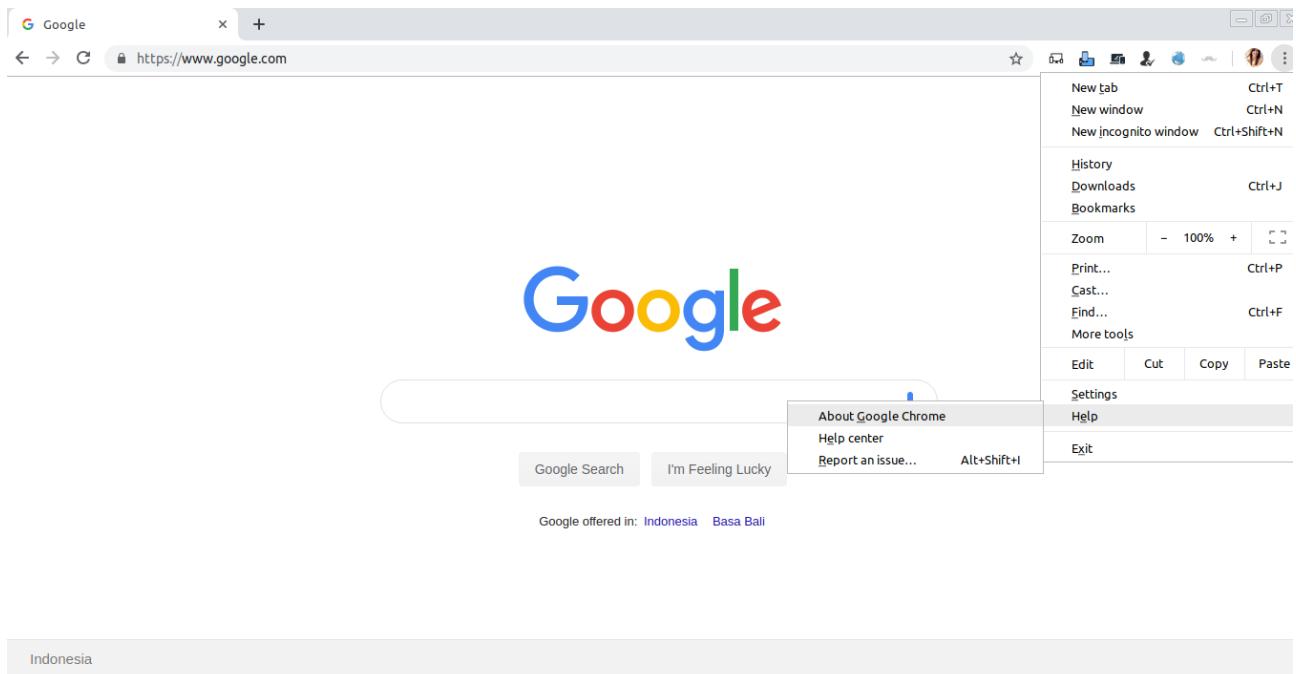
Trik lengkap untuk mendownload versi chromedriver yang cocok untuk google chrome anda bisa dibaca di <http://chromedriver.chromium.org/downloads/version-selection>

Berikut ini adalah trik untuk mendownload versi chromedriver yang tepat untuk google chrome di sistem Anda.

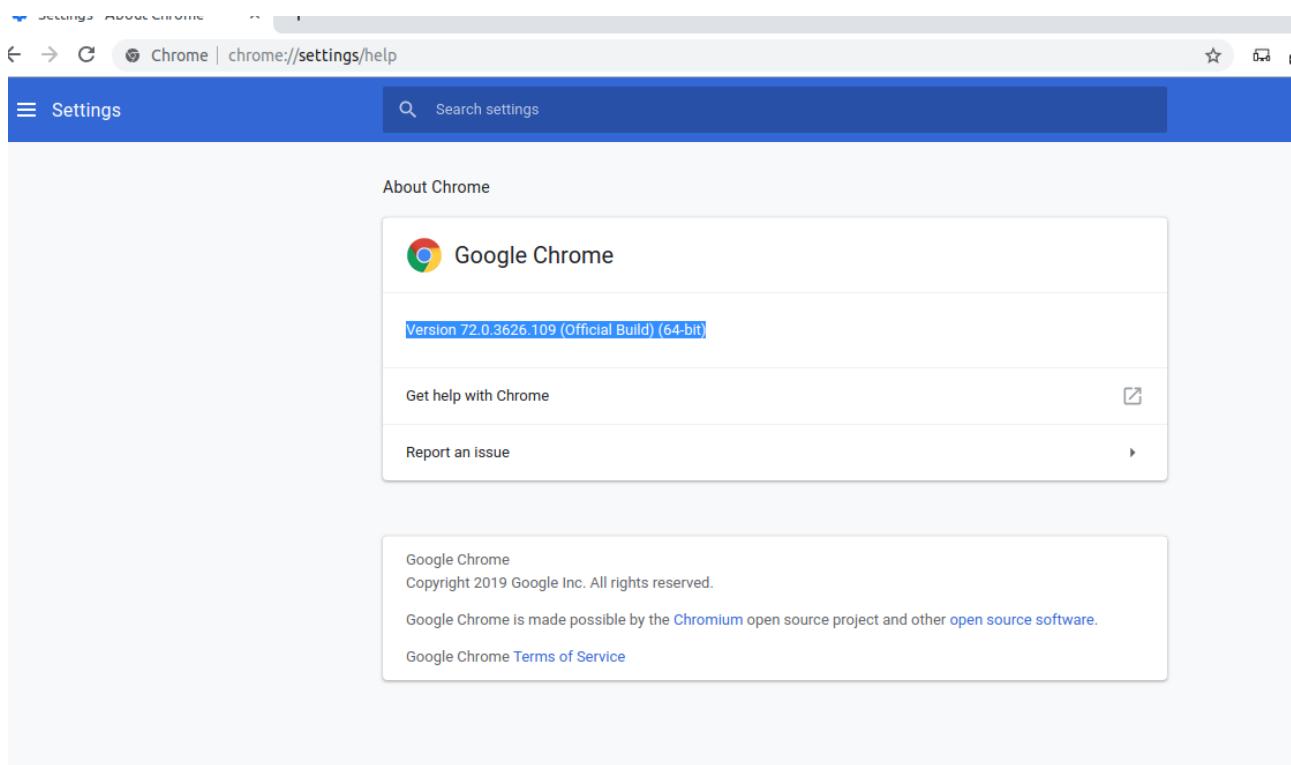
Pertama tama buka google chrome, lalu tekan alt+f



Selanjutnya, pilih menu “Help”



Selanjutnya, pilih “About Google Chrome”

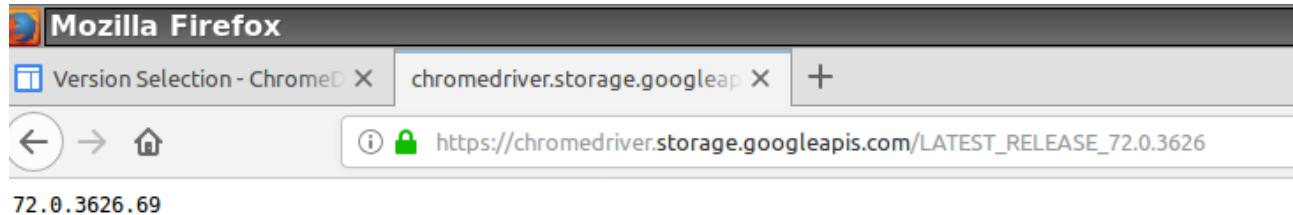


Pada contoh kali ini, saya menggunakan google chrome versi 72.0.3626.109 (Official Build) (64-bit) pada sistem operasi linux ubuntu.

Pada string versi : 72.0.3626.109 di sini kita cukup mengambil string 72.0.3626 yang akan kita tambahkan pada url untuk mengecek versi chromedriver yang cocok pada https://chromedriver.storage.googleapis.com/LATEST_RELEASE

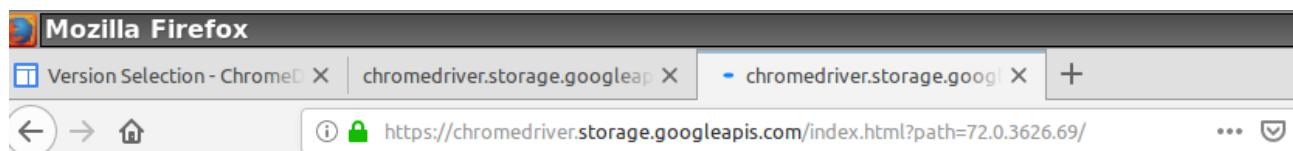
Buka browser dan ketikkan :

https://chromedriver.storage.googleapis.com/LATEST_RELEASE_72.0.3626



Pada contoh di atas terlihat versi chromedriver yang cocok yaitu : 72.0.3626.69, selanjutnya string ini tinggal kita tambahkan pada url : <https://chromedriver.storage.googleapis.com/index.html?path=72.0.3626.69/>

Buka browser pada alamat <https://chromedriver.storage.googleapis.com/index.html?path=72.0.3626.69/>



Index of /72.0.3626.69/

Name	Last modified	Size	ETag
Parent Directory		-	-
chromedriver_linux64.zip	2019-01-22 07:21:41	4.81MB	902a5fae03b63cc6deeb750d63211f67
chromedriver_mac64.zip	2019-01-22 07:21:42	6.59MB	8ce656dcb0a145ffb887eb08a1806723
chromedriver_win32.zip	2019-01-22 07:21:44	4.38MB	09af2515e314569d11636efc72a7597b

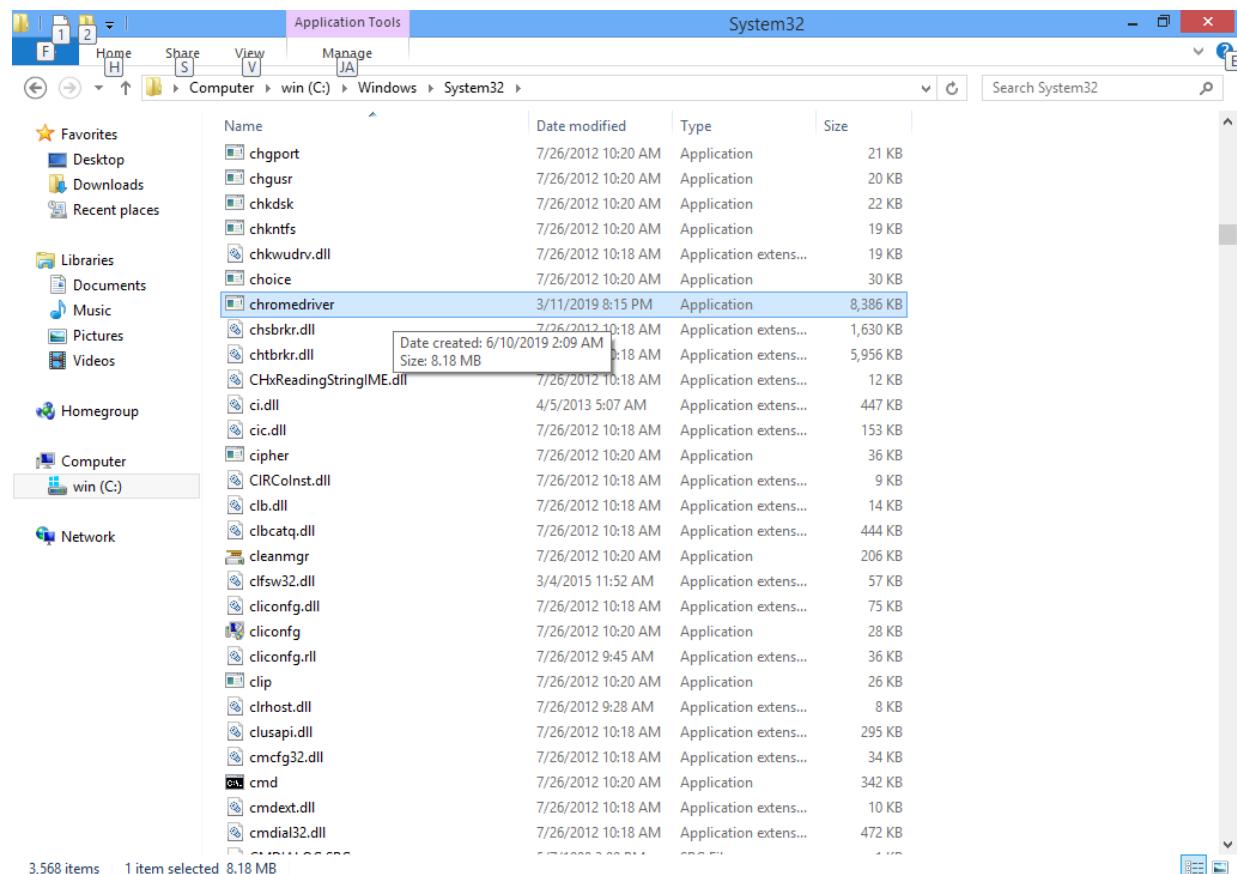
terlihat ada beberapa jenis chromedriver untuk linux, windows dan mac. Karena saya menggunakan sistem operasi linux 64 bit maka saya akan download [chromedriver_linux64.zip](https://chromedriver.storage.googleapis.com/72.0.3626.69/chromedriver_linux64.zip)
Jika menggunakan windows maka yang didownload adalah chromedriver_win32.zip

```
ringlayer@ringlayer-Inspiron-3442: ~/Downloads
File Edit Tabs Help
root@ringlay... x root@ringlay... x ringlayer@rin... x
ringlayer@ringlayer-Inspiron-3442:~/Downloads$ wget -c "https://chromedriver.storage.googleapis.com/72.0.3626.69/chromedriver_linux64.zip"
--2019-05-09 21:17:13-- https://chromedriver.storage.googleapis.com/72.0.3626.69/chromedriver_linux64.zip
Resolving chromedriver.storage.googleapis.com (chromedriver.storage.googleapis.com)... 74.125.24.128, 2404:6800:4003:c04::80
Connecting to chromedriver.storage.googleapis.com (chromedriver.storage.googleapis.com)|74.125.24.128|:443... connected.
HTTP request sent, awaiting response... 206 Partial Content
Length: 5040584 (4,8M), 4999796 (4,8M) remaining [application/zip]
Saving to: 'chromedriver_linux64.zip'

chromedriver_linux64.zip          6%[=====] 318,1
```

Setelah selenium dan chromedriver didownload, ekstrak lalu kopikan binary chromedriver ke path binary jika menggunakan windows, kopikan chromedriver.exe ke

c:\windows\system32



atau bisa juga nanti binary chromedriver.exe ditaruh di folder tempat script python akan dijalankan jika menggunakan linux, pada contoh ini chromedriver dikopi ke

/usr/local/bin

```
root@ringlayer-Inspiron-3442:/home/ringlayer/Downloads/chromedriver# ls
chromedriver  chromedriver_linux64.zip
root@ringlayer-Inspiron-3442:/home/ringlayer/Downloads/chromedriver# mv  chromedriver /usr/local/bin/chromedriver
root@ringlayer-Inspiron-3442:/home/ringlayer/Downloads/chromedriver#
```

Selanjutnya kita perlu menginstall modul python selenium, ketikkan :

pip install selenium.

Atau

pip3 install selenium

Setelah semua terinstall dengan benar, kita sudah bisa mulai membuat script untuk web scrapping dengan python selenium.

3.4.2. Mulai Menggunakan Selenium

Untuk melakukan web scrapping dengan selenium, pertama tama kita perlu mengimport sub modul webdriver di dalam modul selenium :

```
from selenium import webdriver
```

Selanjutnya, kita perlu melakukan inisialisasi browser terlebih dahulu. Untuk menjalankan browser chrome, kita akan menggunakan chromiumdriver :

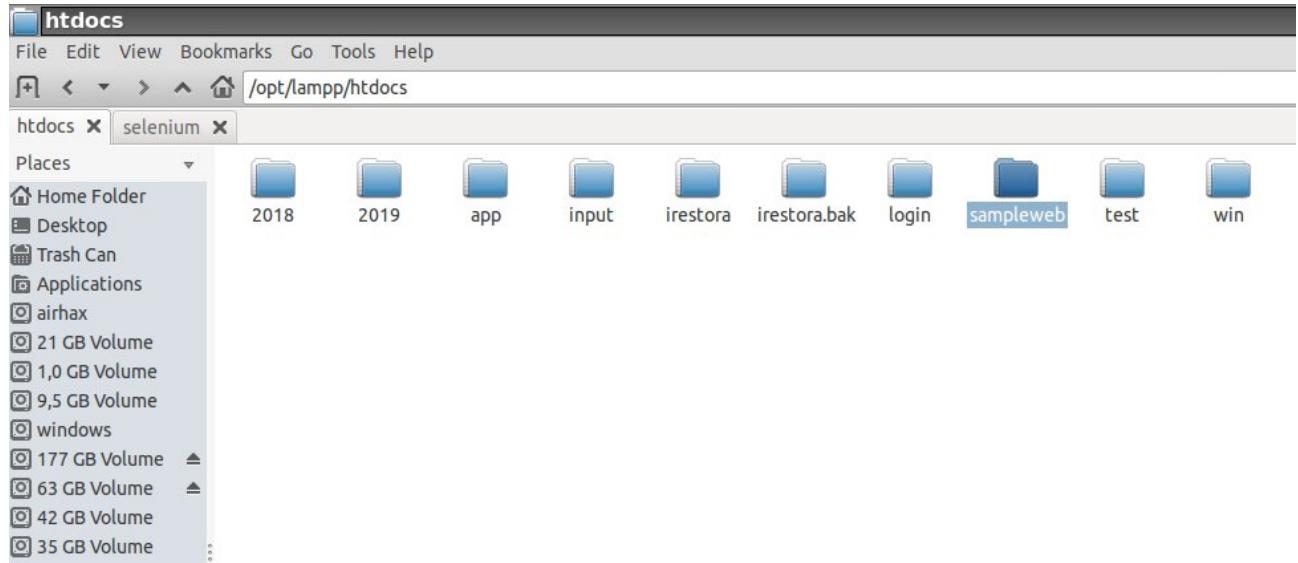
```
your_chromium_path = "/usr/local/bin/chromedriver"
browser = webdriver.Chrome(your_chromium_path)
```

Pada contoh di atas, path chromedriver ada pada /usr/local/bin/ (sesuaikan dengan letak chromedriver pada sistem Anda).

Jika menggunakan windows,

Sebelum memulai penggunaan basic selenium, pastikan xampp terinstall dan berjalan di komputer lokal anda karena kita akan mencoba mengakses web yang kita siapkan di localhost.

Pada contoh di komputer ini, karena menggunakan xampp for linux maka direktori root web ada di /opt/lampp/htdocs , direktori sampleweb berada pada direktori ini.



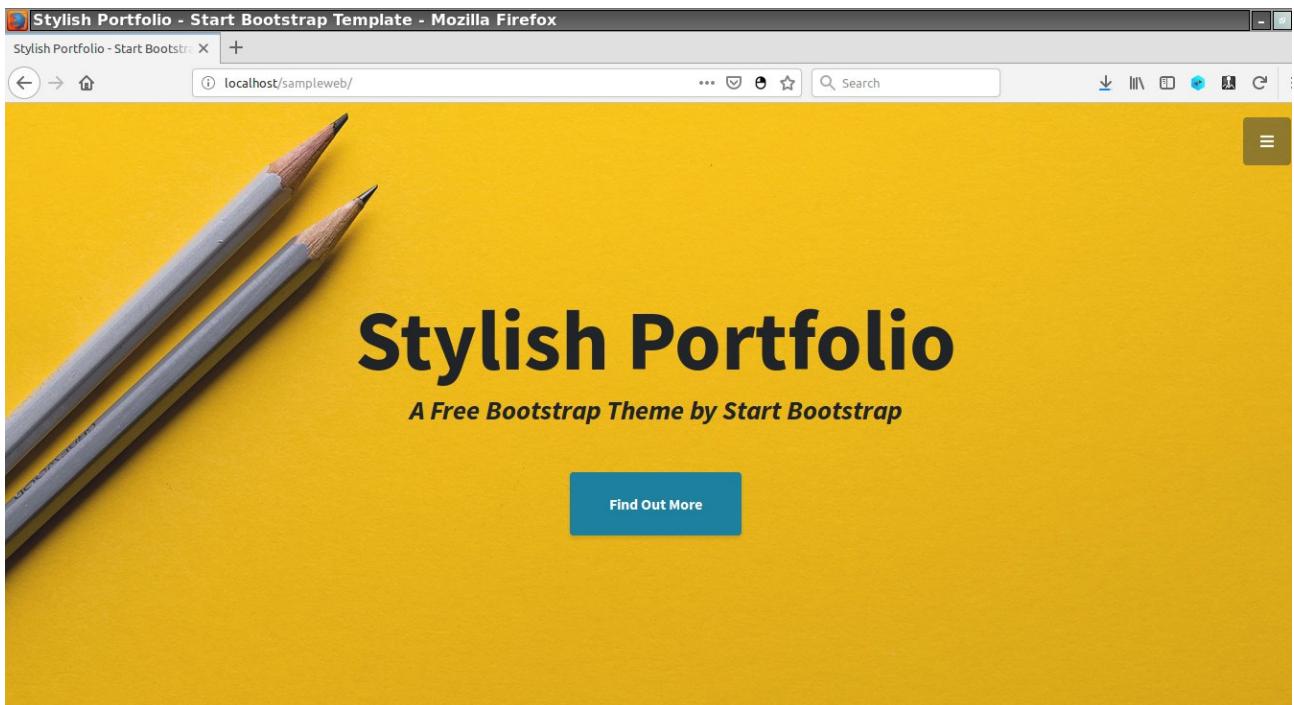
Jika menggunakan windows dan xampp di mana xampp diinstall secara default pada drive c, maka path root web ada di c:\xampp\htdocs, simpan folder sampleweb pada c:\xampp\htdocs

Source code web sample : “sampleweb” bisa didownload di github repo untuk training ini (alamat ada di BAB 1)

Simpan direktori sampleweb pada htdocs agar bisa diakses pada alamat <http://localhost/sampleweb>.

Jalankan xampp dan buka alamat <http://localhost/sampleweb>

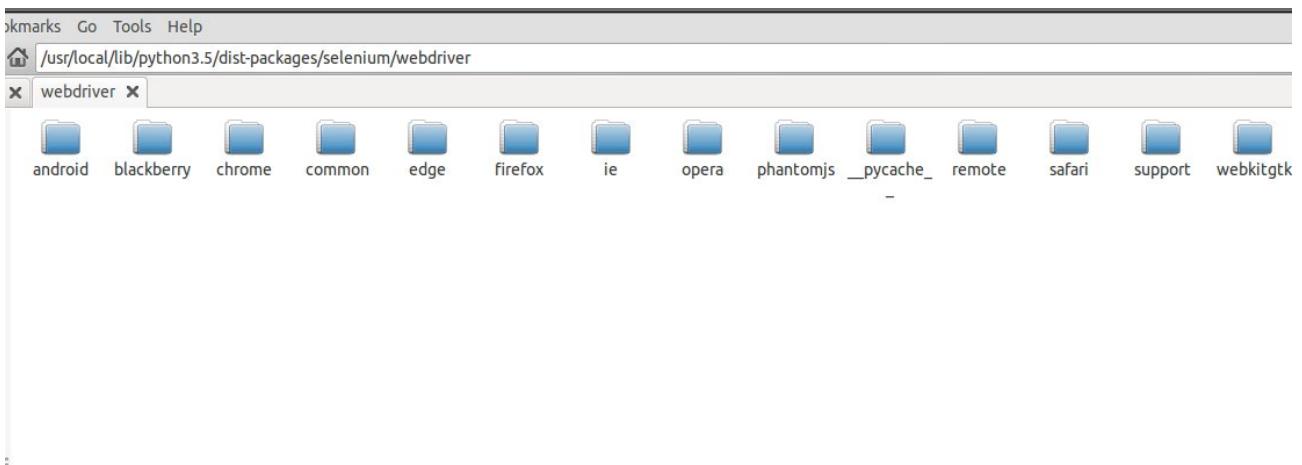
Jika penempatan benar berikut ini tampilan yang muncul



Untuk memulai web scrapping, pertama tama kita perlu mengimport modul selenium :

```
from selenium import webdriver
```

Selanjutnya kita buat object instance dari class Webdriver, class Webdriver merupakan kelas di dalam salah satu submodul selenium.webdriver.chrome.



Pada modul selenium terdapat sub modul webdriver yang didalamnya terdapat sub sub modul yang dipisahkan dengan folder folder tersendiri, untuk masing masing modulnya terdapat implementasi kelas Webdriver tersendiri.

Untuk membuat object instance dari class Webdriver, kita perlu memberikan parameter berupa letak binary chromedriver di sistem.

Contoh pada linux :

```
your_chromium_path = "/usr/local/bin/chromedriver  
browser = webdriver.Chrome(your_chromium_path)
```

Jika elf binary chromedriver terdapat pada folder yang sama dengan script python yang dijalankan maka your_chromium_path = “chromedriver”

Contoh pada windows :

Pada windows untuk memudahkan bisa juga binary chromedriver.exe disimpan di c:\windows\system32.

Jika binary chromedriver.exe ditaruh di folder yang sama, maka :

```
webdriver.Chrome(executable_path=r'chromedriver.exe')
```

Contoh lain jika binary chromedriver.exe terdapat pada <c:\Users>

```
webdriver.Chrome(executable_path=r'c:\Users\chromedriver.exe')
```

Untuk melihat bantuan pada objek Webdriver, bisa dilakukan pada shell python. Jalankan shell python :

```
>>> from selenium import webdriver  
>>> browser = webdriver.Chrome()  
>>> help(browser)
```

akan muncul method method pada class Webdriver milik chrome :

```
Help on class WebDriver in module selenium.webdriver.chrome.webdriver:

class WebDriver(selenium.webdriver.remote.webdriver.WebDriver)
    Controls the ChromeDriver and allows you to drive the browser.

    You will need to download the ChromeDriver executable from
    http://chromedriver.storage.googleapis.com/index.html

    Method resolution order:
        WebDriver
        selenium.webdriver.remote.webdriver.WebDriver
        builtins.object

    Methods defined here:

        __init__(self, executable_path='chromedriver', port=0, options=None, service_args=None, desired_capabilities=None)
            Creates a new instance of the chrome driver.

            Starts the service and then creates new instance of chrome driver.

            :Args:
                - executable_path - path to the executable. If the default is used it assumes the executable is in the
                - port - port you would like the service to run, if left as 0, a free port will be found.
                - desired_capabilities: Dictionary object with non-browser specific
                    capabilities only, such as "proxy" or "loggingPref".
                - options: this takes an instance of ChromeOptions

        create_options(self)

        get_network_conditions(self)
            Gets Chrome network emulation settings.

            :Returns:
                A dict. For example:

                {'latency': 4, 'download_throughput': 2, 'upload_throughput': 2,
                 'offline': False}

        launch_app(self, id)
            Launches Chrome app specified by id.

    :
```

```
Help on class WebDriver in module selenium.webdriver.chrome.webdriver:

class WebDriver(selenium.webdriver.remote.webdriver.WebDriver)
    Controls the ChromeDriver and allows you to drive the browser.

    You will need to download the ChromeDriver executable from
    http://chromedriver.storage.googleapis.com/index.html

    Method resolution order:
        WebDriver
        selenium.webdriver.remote.webdriver.WebDriver
        builtins.object

    Methods defined here:

        __init__(self, executable_path='chromedriver', port=0, options=None, service_args=None, desired_capabilities=None)
            Creates a new instance of the chrome driver.

            Starts the service and then creates new instance of chrome driver.

            :Args:
                - executable_path - path to the executable. If the default is used it assumes the executable is in the
                - port - port you would like the service to run, if left as 0, a free port will be found.
                - desired_capabilities: Dictionary object with non-browser specific
                    capabilities only, such as "proxy" or "loggingPref".
                - options: this takes an instance of ChromeOptions

        create_options(self)

        get_network_conditions(self)
            Gets Chrome network emulation settings.

            :Returns:
                A dict. For example:

                {'latency': 4, 'download_throughput': 2, 'upload_throughput': 2,
                 'offline': False}

        launch_app(self, id)
            Launches Chrome app specified by id.

    :
```

Selanjutnya, buat aplikasi baru dengan nama selenium_basic.py :

```
#!/usr/bin/env python3

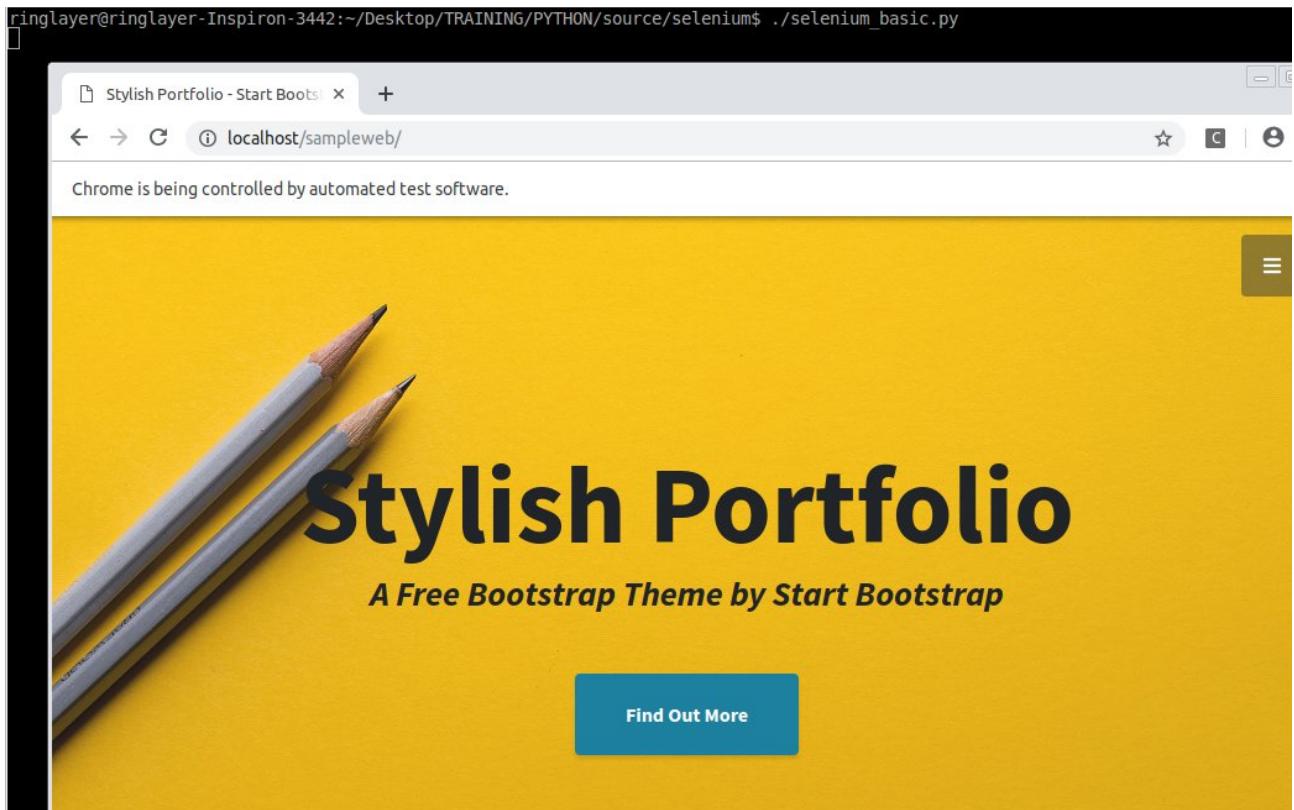
"""
selenium_basic.py

Basic Selenium Example
for python training at www.jasaplus.com
"""

from selenium import webdriver
import time
your_chromium_path = "/usr/local/bin/chromedriver"
browser = webdriver.Chrome(your_chromium_path)
browser.get("http://localhost/sampleweb")
time.sleep(8)
html_source = browser.page_source
print("html_source : ")
print(html_source)
```

versi di atas adalah untuk linux, untuk windows perlu sedikit penyesuaian di bagian path chromedriver.

Jalankan script di atas, jika benar maka akan menampilkan web sampleweb :



Penjelasan

```
browser = webdriver.Chrome(your_chromium_path)
browser.get("http://localhost/sampleweb")
```

Di sini kita buat instance objek baru dari class WebDriver kemudian kita gunakan method get() milik class ini untuk mengakses halaman web dengan alamat http://localhost/sampleweb

```
html_source = browser.page_source
print("html_source : ")
print(html_source)
```

Rutin di atas berfungsi untuk menampilkan source code html dari halaman web pada layar.

Tips

Pada saat pengembangan aplikasi web scrapper akan lebih cepat dan mudah dilakukan saat uji coba secara langsung dari console python, misalnya kita bisa menambah opsi browser, menguji eksekusi script secara langsung pada objek Webdriver yang sedang aktif tanpa perlu menjalankan ulang script dari awal.

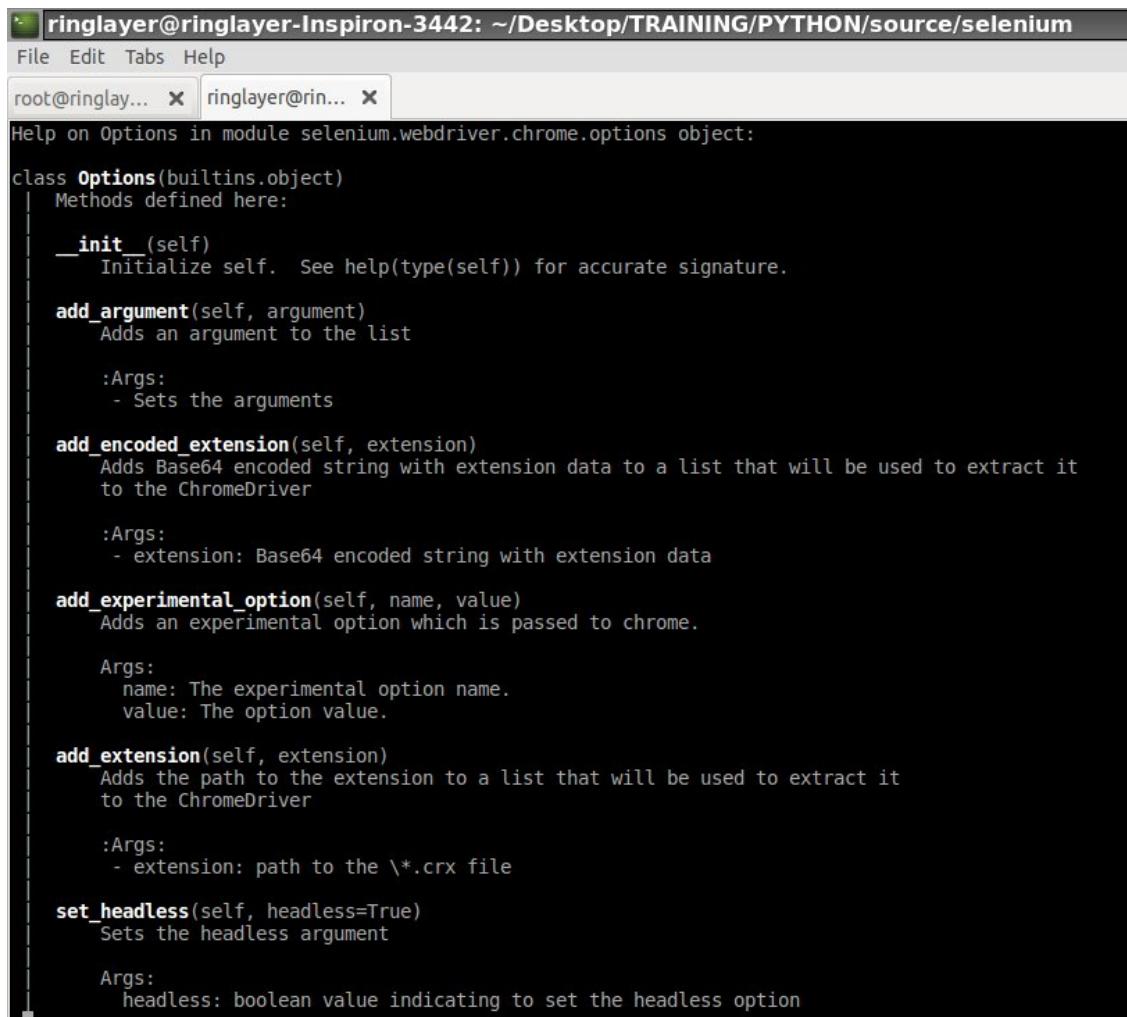
3.4.3. Operasi Dasar Selenium

1. Opsi pada Selenium Webdriver

Sebelum melakukan inisialisasi browser pada selenium, kita bisa memberikan beberapa pengaturan pada opsi webdriver.

Untuk membuat pengaturan ini kita memerlukan objek instance dari class Options. Untuk melihat daftar method pada class Options ini, bisa dilakukan dari python console :

```
$ python3
Python 3.5.2 (default, Nov 23 2017, 16:37:01)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from selenium import webdriver
>>> chrome_options = webdriver.ChromeOptions()
>>> help(chrome_options)
```



The screenshot shows a terminal window with the following details:

- Title bar: ringlayer@ringlayer-Inspiron-3442: ~/Desktop/TRAINING/PYTHON/source/selenium
- Menu bar: File Edit Tabs Help
- Current directory: root@ringlay... x ringlayer@rin... x
- Output content:

```
Help on Options in module selenium.webdriver.chrome.options object:

class Options(builtins.object)
    Methods defined here:

        __init__(self)
            Initialize self. See help(type(self)) for accurate signature.

        add_argument(self, argument)
            Adds an argument to the list

            :Args:
                - Sets the arguments

        add_encoded_extension(self, extension)
            Adds Base64 encoded string with extension data to a list that will be used to extract it
            to the ChromeDriver

            :Args:
                - extension: Base64 encoded string with extension data

        add_experimental_option(self, name, value)
            Adds an experimental option which is passed to chrome.

            Args:
                name: The experimental option name.
                value: The option value.

        add_extension(self, extension)
            Adds the path to the extension to a list that will be used to extract it
            to the ChromeDriver

            :Args:
                - extension: path to the \*.crx file

        set_headless(self, headless=True)
            Sets the headless argument

            Args:
                headless: boolean value indicating to set the headless option
```

Berikut ini penjelasan method method pada class Options :

add_argument(self, argument)

Method ini digunakan untuk menambahkan argumen berupa list pada Chrome.

Beberapa contoh argument yang sering ditambahkan dan penggunaanya (chrome_options merupakan nama objek instance dari webdriver.ChromeOptions() yang telah kita siapkan sebelumnya, namanya bebas sesuai keinginan Anda, pada contoh ini kita menggunakan nama objek chrome_options) :

- chrome_options.add_argument("--start-maximized")
digunakan agar browser start dengan jendela maksimal
- chrome_options.add_argument('--ignore-certificate-errors')
digunakan untuk mengacuhkan error sertifikat ssl
- chrome_options.add_argument('--headless')
digunakan untuk memulai chrome tanpa gui
- chrome_options.add_argument('--disable-gpu')
digunakan untuk deaktivasi fitur yang membutuhkan gpu, biasa digunakan bersamaan dengan --headless
- chrome_options.add_argument("--disable-extensions")
digunakan untuk menonaktifkan ekstensi google chrome
- chrome_options.add_argument("user-agent=Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/50.0.2661.86 Safari/537.36")
Contoh di atas digunakan untuk mengganti user agent menjadi "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/50.0.2661.86 Safari/537.36"

add_encoded_extension(self, extension)

Method ini digunakan untuk menambahkan string yang dienkrip dengan base64 yang berisi data ekstensi ke list argumen.

add_experimental_option(self, name, value)

Method ini digunakan untuk menambahkan opsi eksperimental

add_extension(self, extension)

Method ini digunakan untuk menambahkan ekstensi ke chrome

set_headless(self, headless=True)

Method ini digunakan untuk menjalankan chrome tanpa tampilan GUI

2. Pencarian Elemen dan Klik Pada Elemen

Untuk melakukan klik pada elemen, kita bisa melakukan pencarian elemen terlebih dahulu berdasarkan :

- Pencarian elemen berdasarkan nama elemen

Untuk melakukan pencarian elemen berdasarkan nama elemen, kita menggunakan method `find_element_by_name()` atau `find_elements_by_name()`. Perbedaan antara keduanya adalah yang pertama akan menghasilkan objek dari elemen yang dicari sedangkan yang kedua akan menghasilkan list berupa objek dari elemen yang ditemukan.

Sebagai contoh, kita akan mencoba melakukan scrapping pada <http://localhost/sampleweb> yang sebelumnya kita telah persiapkan, pada source code index.html terdapat beberapa element dengan nama element, terlihat pada bagian source code di bawah ini :

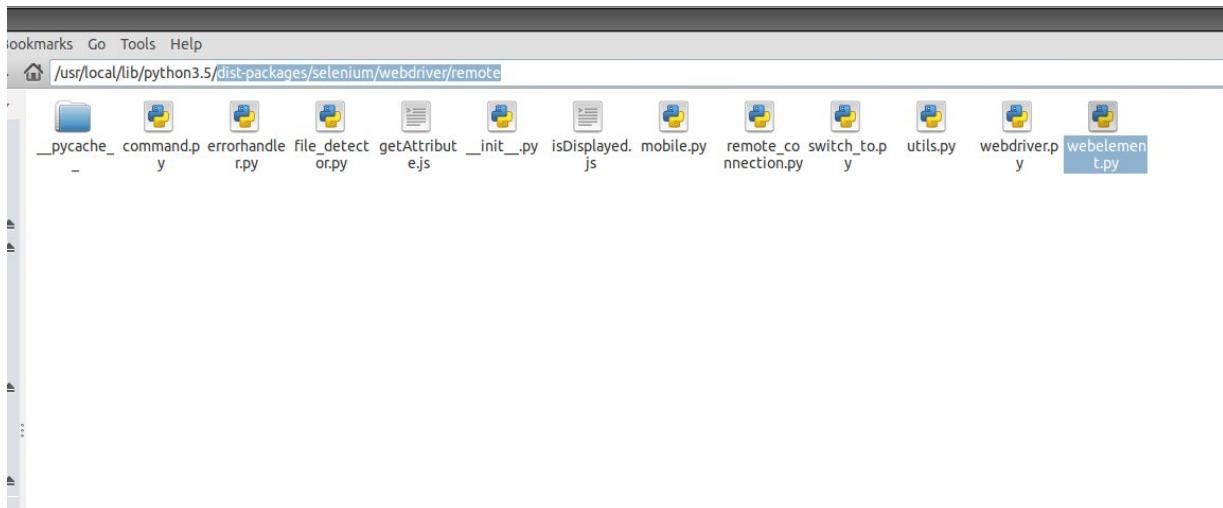
```
<!-- Header -->
<header class="masthead d-flex">
  <div class="container text-center my-auto">
    <h1 class="mb-1" name="data">Stylish Portfolio</h1>
    <h3 class="mb-5">
      <em name="data">A Free Bootstrap Theme by Start Bootstrap</em>
    </h3>
    <a name="linkz" class="btn btn-primary btn-xl js-scroll-trigger" href="#about">Find Out More</a>
  </div>
  <div class="overlay"></div>
</header>
```

Perhatikan contoh pada console python di bawah ini untuk mendapatkan objek dari elemen dengan nama “data” dan elemen dengan nama “linkz” :

```
>>> from selenium import webdriver
>>> import time
>>> your_chromium_path = "/usr/local/bin/chromedriver"
>>> browser = webdriver.Chrome(your_chromium_path)
>>> browser.get("http://localhost/sampleweb")
>>> time.sleep(3)
>>> datas = browser.find_elements_by_name("data")
>>> print(type(datas))
<class 'list'>
>>> linkz = browser.find_element_by_name("linkz")
>>> print(type(linkz))
<class 'selenium.webdriver.remote.webelement.WebElement'>
>>>
>>> for l in datas:
...     print(type(l))
...
<class 'selenium.webdriver.remote.webelement.WebElement'>
<class 'selenium.webdriver.remote.webelement.WebElement'>
>>>
```

Terlihat method `find_elements_by_name()` menghasilkan list berupa instance object `selenium.webdriver.remote.webelement.WebElement` sedangkan `find_element_by_name()` menghasilkan objek instance tunggal `selenium.webdriver.remote.webelement.WebElement`,

WebElement ini adalah class yang terdapat pada sub direktori dist-packages/selenium/webdriver/remote



on3.5/dist-packages/selenium/webdriver/remote — Atom

```
  selenium_basic.py      FindElement.py      webelement.py      index.html
41 # not relying on __package__ here as it can be None in some situations (see #4558)
42 _pkg = '..'.join(__name__.split('.')[1:-1])
43getAttribute_js = pkgutil.get_data(_pkg, 'getAttribute.js').decode('utf8')
44isDisplayed_js = pkgutil.get_data(_pkg, 'isDisplayed.js').decode('utf8')

45
46
47 class WebElement(object):
48     """Represents a DOM element.
49
50     Generally, all interesting operations that interact with a document will be
51     performed through this interface.
52
53     All method calls will do a freshness check to ensure that the element
54     reference is still valid. This essentially determines whether or not the
55     element is still attached to the DOM. If this test fails, then an
56     ``StaleElementReferenceException`` is thrown, and all future calls to this
57     instance will fail."""
58
59     def __init__(self, parent, id_, w3c=False):
60         self._parent = parent
61         self._id = id_
62         self._w3c = w3c
63
64     def __repr__(self):
65         return '<{0.__module__}.{0.__name__} (session="{1}", element="{2}")>'.format(
66             type(self), self._parent.session_id, self._id)
67
68     @property
69     def tag_name(self):
70         """This element's ``tagName`` property."""
71         return self._execute(Command.GET_ELEMENT_TAG_NAME)['value']
72
73     @property
```

Untuk mendapatkan isi teks atau html dari objek instance WebElement, perhatikan contoh di bawah ini di konsol python :

```
>>>
>>> from selenium import webdriver
>>> import time
>>> your_chromium_path = "/usr/local/bin/chromedriver"
```

```

>>> browser = webdriver.Chrome(your_chromium_path)
>>> browser.get("http://localhost/sampleweb")
>>> time.sleep(3)
>>> linkz = browser.find_element_by_name("linkz")
>>> print(linkz.text)
Find Out More
>>> print(linkz.get_attribute('innerHTML'))
Find Out More
>>>

```

- Pencarian elemen berdasarkan id

Untuk melakukan pencarian elemen berdasarkan id, kita menggunakan method `find_element_by_id()` atau `find_elements_by_id()`. Perbedaanya sama dengan yang di atas, yang pertama menghasilkan objek tunggal sedangkan yang kedua menghasilkan list berupa objek.

Berikut ini adalah contoh penggunaan `find_element_by_id()` pada konsol python :

```

>>> from selenium import webdriver
>>> import time
>>> your_chromium_path = "/usr/local/bin/chromedriver"
>>> browser = webdriver.Chrome(your_chromium_path)
>>> browser.get("http://localhost/sampleweb")
>>> time.sleep(3)
>>> start = browser.find_element_by_id("mulai")
>>> start.click()
>>> time.sleep(1)
>>> about = browser.find_element_by_id("menu_about")
>>> if about != None:
...     about.click()
...
>>>

```

Contoh di diterapkan pada sampleweb yang telah kita siapkan pada contoh sebelumnya, di mana pada bagian menu terdapat beberapa id :

```

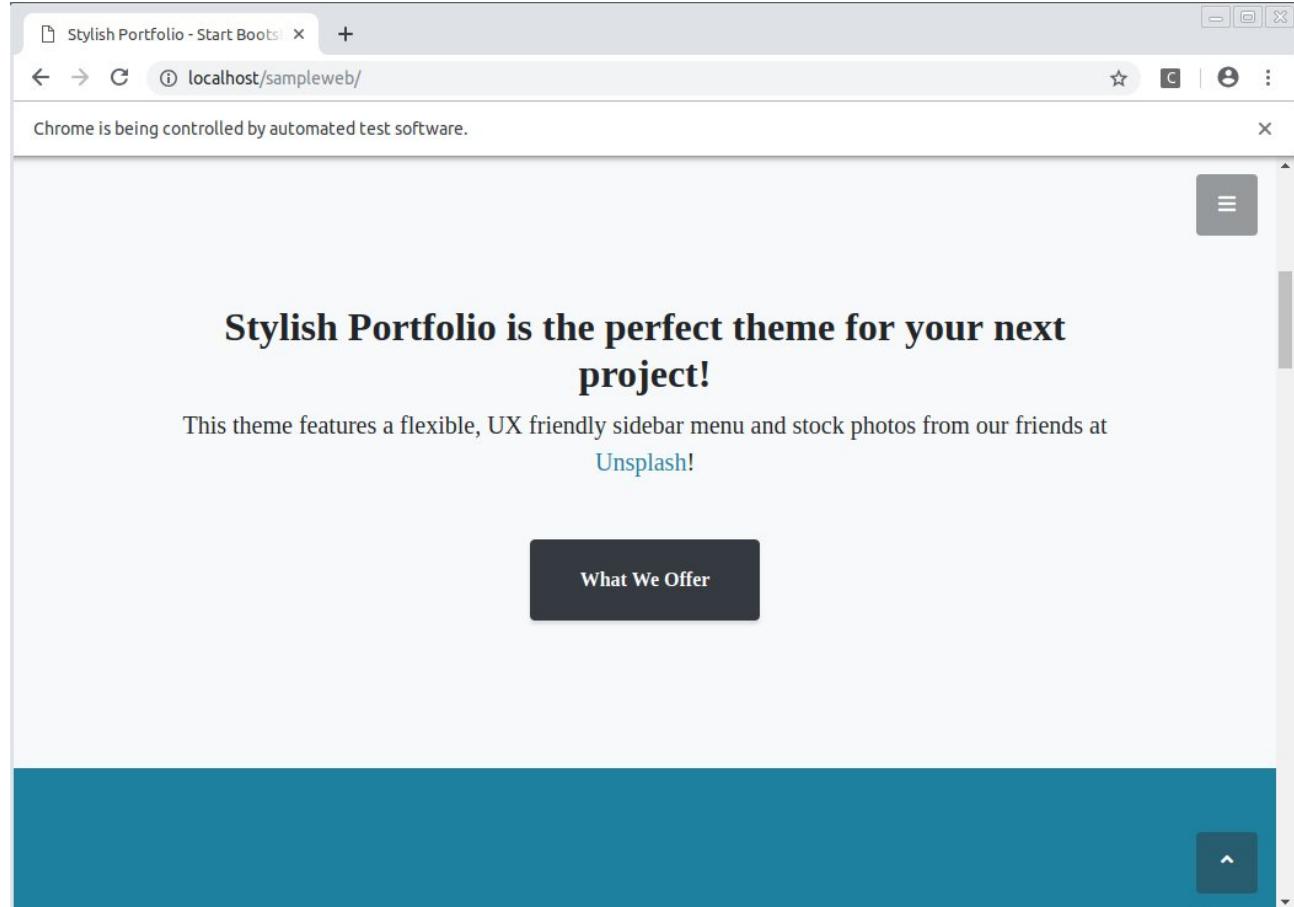
<a class="menu-toggle rounded" href="#mulai" id=mulai>
    <i class="fas fa-bars"></i>
</a>
<nav id="sidebar-wrapper">
    <ul class="sidebar-nav">
        <li class="sidebar-brand">
            <a class="js-scroll-trigger start" href="#page-top" id="menu_start">Start Bootstrap</a>
        </li>
        <li class="sidebar-nav-item">
            <a class="js-scroll-trigger home" href="#page-top" id="menu_home">Home</a>
        </li>
        <li class="sidebar-nav-item">
            <a class="js-scroll-trigger about" href="#about" id=menu_about">About</a>
        </li>
        <li class="sidebar-nav-item">
            <a class="js-scroll-trigger services" href="#services" id="menu_services">Services</a>
        </li>
        <li class="sidebar-nav-item">
            <a class="js-scroll-trigger portfolio" href="#portfolio" id="menu_portfolio">Portfolio</a>
        </li>

```

```
<li class="sidebar-nav-item">
  <a class="js-scroll-trigger contact" href="#contact" id="menu_contact">Contact</a>
</li>
</ul>
</nav>
```

Pada contoh di atas, kita melakukan klik pada objek dom dengan id “mulai” dan id “menu_about”

Hasil eksekusi dari script akan menyebabkan link navigasi About diklik dan page menjadi scroll ke bagian about :



- Pencarian elemen berdasarkan teks link

Untuk melakukan pencarian elemen berdasarkan teks link, kita menggunakan method `find_element_by_link_text()` dan `find_elements_by_link_text()`

Pada source code sampleweb, kita bisa melihat beberapa link dengan teks pada bagian navigasi :

```
<nav id="sidebar-wrapper">
  <ul class="sidebar-nav">
    <li class="sidebar-brand">
      <a class="js-scroll-trigger start" href="#page-top" id="menu_start">Start Bootstrap</a>
    </li>
    <li class="sidebar-nav-item">
      <a class="js-scroll-trigger home" href="#page-top" id="menu_home">Home</a>
    </li>
    <li class="sidebar-nav-item">
```

```

<a class="js-scroll-trigger about" href="#about" id="menu_about">About</a>
</li>
<li class="sidebar-nav-item">
  <a class="js-scroll-trigger services" href="#services" id="menu_services">Services</a>
</li>
<li class="sidebar-nav-item">
  <a class="js-scroll-trigger portfolio" href="#portfolio" id="menu_portfolio">Portfolio</a>
</li>
<li class="sidebar-nav-item">
  <a class="js-scroll-trigger contact" href="#contact" id="menu_contact">Contact</a>
</li>
</ul>
</nav>
```

Kita akan membuat aplikasi web scrapper mengklik link dengan teks “Services” di atas.

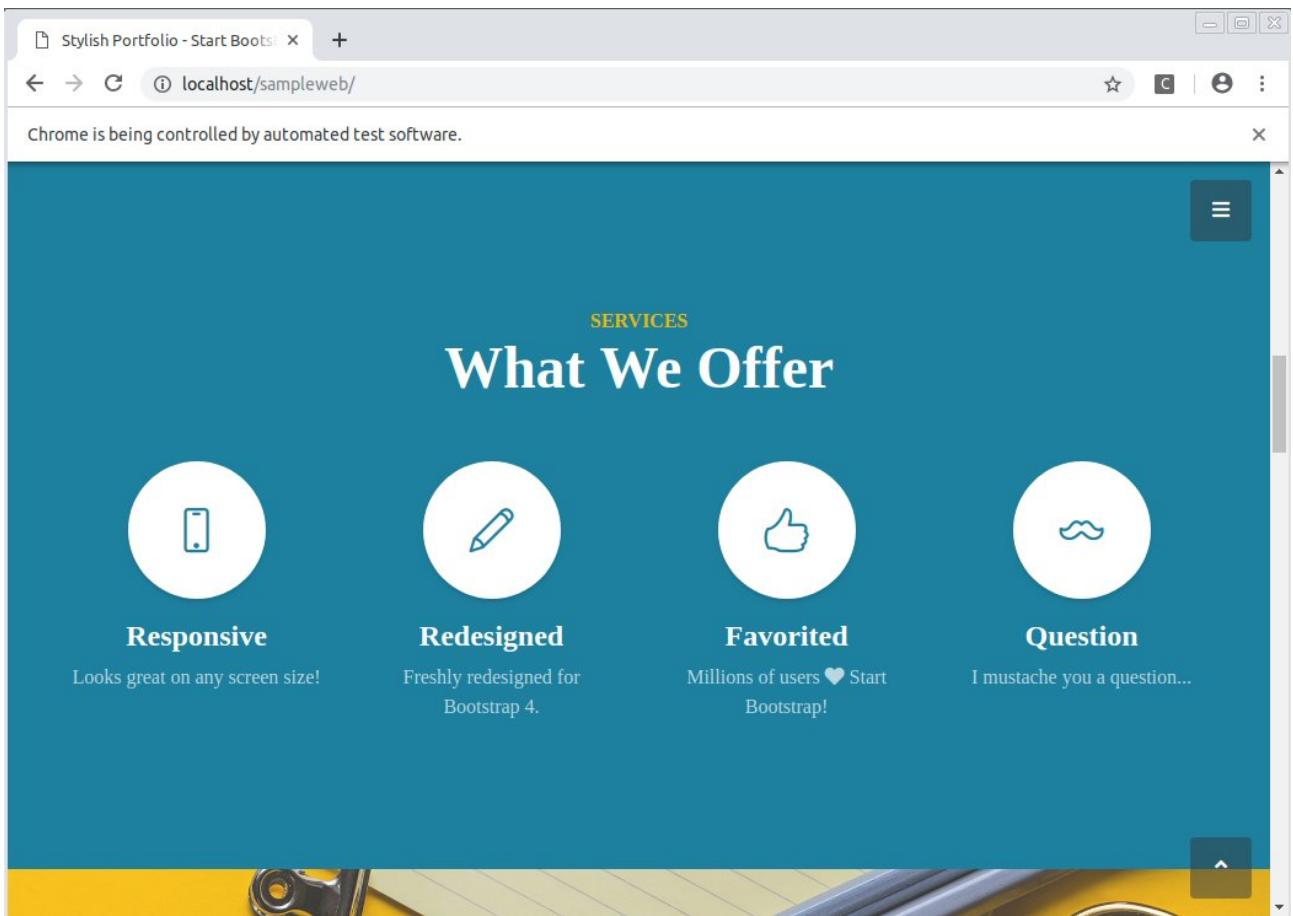
Berikut ini adalah contoh penggunaan `find_element_by_link_text()` pada python console :

```

ringlayer@ringlayer-Inspiron-3442:~/Desktop/TRAINING/PYTHON/source/selenium$ python3
Python 3.5.2 (default, Nov 23 2017, 16:37:01)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from selenium import webdriver
>>> import time
>>> your_chromium_path = "/usr/local/bin/chromedriver"
>>> browser = webdriver.Chrome(your_chromium_path)

>>> browser.get("http://localhost/sampleweb")
>>> time.sleep(1)
>>>
>>>
>>> start = browser.find_element_by_id("mulai")
>>> start.click()
>>> time.sleep(1)
>>> el1 = browser.find_element_by_link_text('Services')
>>> if el1 != None:
...     el1.click()
...
>>>
```

Pada script di atas kita melakukan locate element link dengan isi teks “Services”
Hasil eksekusi script di atas menyebabkan halaman web scroll ke bagian Services :



- Pencarian elemen berdasarkan sebagian teks link

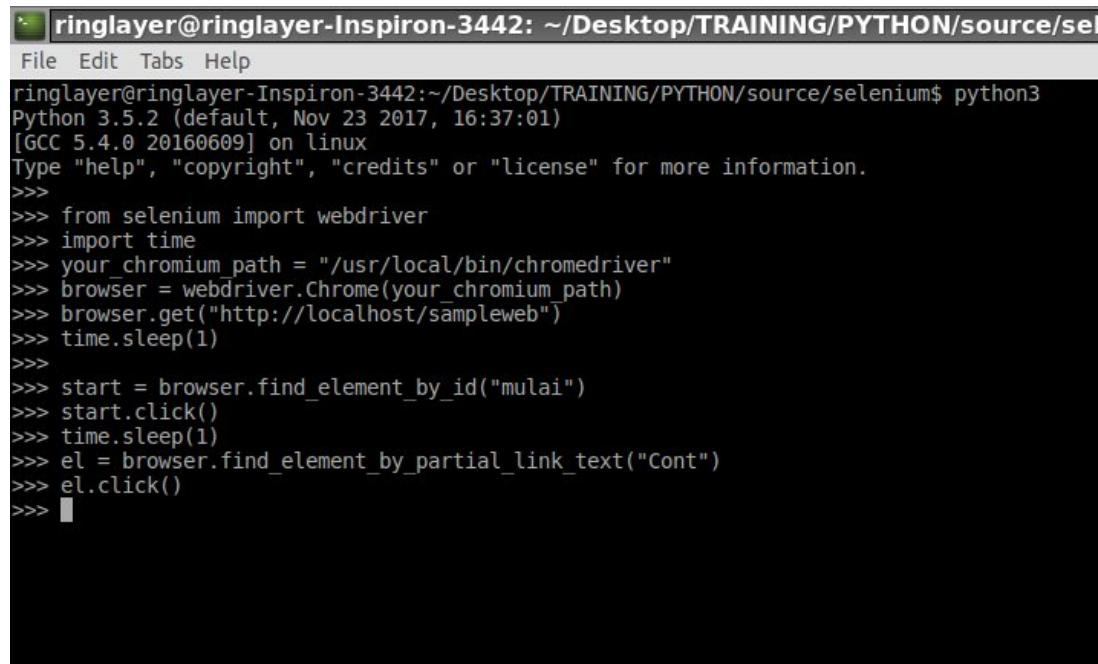
Untuk melakukan pencarian elemen berdasarkan sebagian teks link, kita menggunakan method `find_element_by_partial_link_text()` dan `find_elements_by_partial_link_text()`

Prinsipnya pada dasarnya serupa dengan `find_element_by_link_text()`, yang membedakan adalah yang dicari adalah link yang mengandung sebagian teks. Contoh pada navigasi di halaman sampleweb yang telah kita siapkan tadi, terdapat navigasi :

```
<nav id="sidebar-wrapper">
  <ul class="sidebar-nav">
    <li class="sidebar-brand">
      <a class="js-scroll-trigger start" href="#page-top" id="menu_start">Start Bootstrap</a>
    </li>
    <li class="sidebar-nav-item">
      <a class="js-scroll-trigger home" href="#page-top" id="menu_home">Home</a>
    </li>
    <li class="sidebar-nav-item">
      <a class="js-scroll-trigger about" href="#about" id="menu_about">About</a>
    </li>
    <li class="sidebar-nav-item">
      <a class="js-scroll-trigger services" href="#services" id="menu_services">Services</a>
    </li>
    <li class="sidebar-nav-item">
      <a class="js-scroll-trigger portfolio" href="#portfolio" id="menu_portfolio">Portfolio</a>
    </li>
    <li class="sidebar-nav-item">
      <a class="js-scroll-trigger contact" href="#contact" id="menu_contact">Contact</a>
    </li>
  </ul>
```

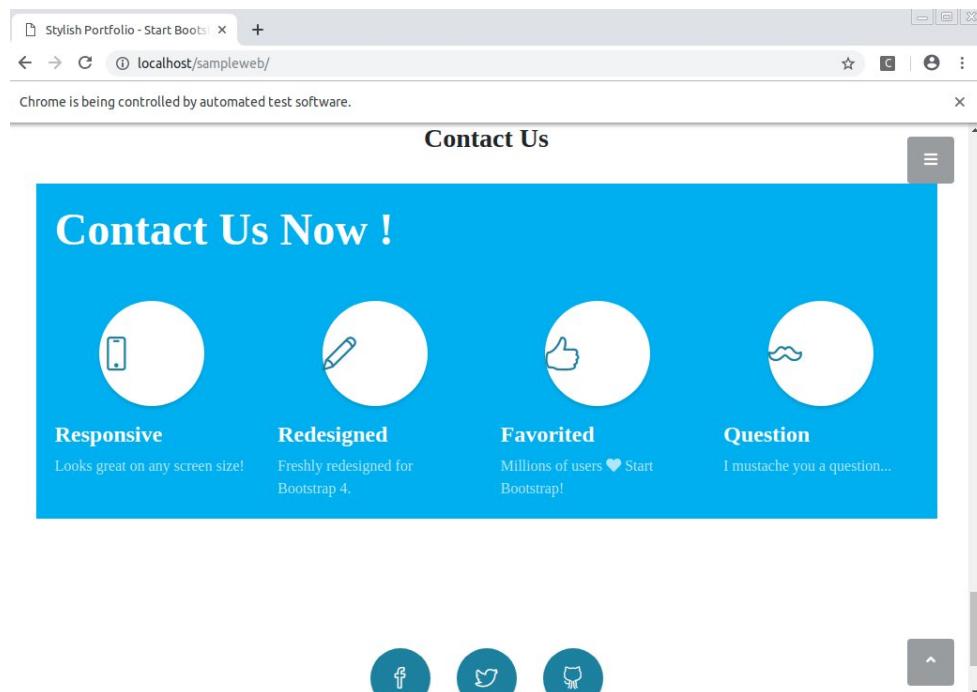
```
</nav>
```

Pada contoh di atas, kita bisa mendapatkan objek untuk link Contact dengan sebagian teks : “Cont”. Berikut ini contoh operasi pada python console :



```
ringlayer@ringlayer-Inspiron-3442: ~/Desktop/TRAINING/PYTHON/source/sele
File Edit Tabs Help
ringlayer@ringlayer-Inspiron-3442:~/Desktop/TRAINING/PYTHON/source/selenium$ python3
Python 3.5.2 (default, Nov 23 2017, 16:37:01)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>> from selenium import webdriver
>>> import time
>>> your_chromium_path = "/usr/local/bin/chromedriver"
>>> browser = webdriver.Chrome(your_chromium_path)
>>> browser.get("http://localhost/sampleweb")
>>> time.sleep(1)
>>>
>>> start = browser.find_element_by_id("mulai")
>>> start.click()
>>> time.sleep(1)
>>> el = browser.find_element_by_partial_link_text("Cont")
>>> el.click()
>>> |
```

Hasil eksekusi script di atas akan menyebabkan link dengan sebagian teks “Cont” terklik, pada sampleweb, link dengan teks lengkap Contact memenuhi kriteria sebagian teks : “Cont”, hasil eksekusi :



- Pencarian elemen berdasarkan xpath

Untuk melakukan pencarian elemen berdasarkan xpath, kita menggunakan method `find_element_by_xpath()` dan `find_elements_by_xpath()`. Perbedaan antara keduanya adalah yang pertama akan menghasilkan objek dari elemen yang dicari sedangkan yang kedua akan menghasilkan list berupa objek dari elemen yang ditemukan.

- Pencarian elemen berdasarkan css selector

Untuk melakukan pencarian elemen berdasarkan css selector, kita menggunakan method `find_element_by_css_selector()` dan `find_elements_by_css_selector()`. Perbedaan antara keduanya adalah yang pertama akan menghasilkan objek dari elemen yang dicari sedangkan yang kedua akan menghasilkan list berupa objek dari elemen yang ditemukan.

- Pencarian elemen berdasarkan nama tag html

Untuk melakukan pencarian elemen berdasarkan nama tag html, kita menggunakan method `find_element_by_tag_name()` dan `find_elements_by_tag_name()`. Perbedaan antara keduanya adalah yang pertama akan menghasilkan objek dari elemen yang dicari sedangkan yang kedua akan menghasilkan list berupa objek dari elemen yang ditemukan.

Berikut ini contoh penggunaan `find_elements_by_tag_name()` dari konsol python :

```
ringlayer@ringlayer-Inspiron-3442:~/Desktop/TRAINING/PYTHON/source/selenium$ python3
Python 3.5.2 (default, Nov 23 2017, 16:37:01)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>> from selenium import webdriver
>>> import time
>>>
>>> your_chromium_path = "/usr/local/bin/chromedriver"
>>> browser = webdriver.Chrome(your_chromium_path)
>>> browser.get("http://localhost/sampleweb")
>>> time.sleep(1)
>>> start = browser.find_element_by_id("mulai")
>>> start.click()
>>> time.sleep(1)
>>> links = browser.find_elements_by_tag_name('a')
>>> for link in links:
...     if link.text == "About":
...         link.click()
...         break
...
>>>
```

Penjelasan

Pada source code di atas, kita melakukan pencarian semua link pada kode ini :

```
links = browser.find_elements_by_tag_name('a')
```

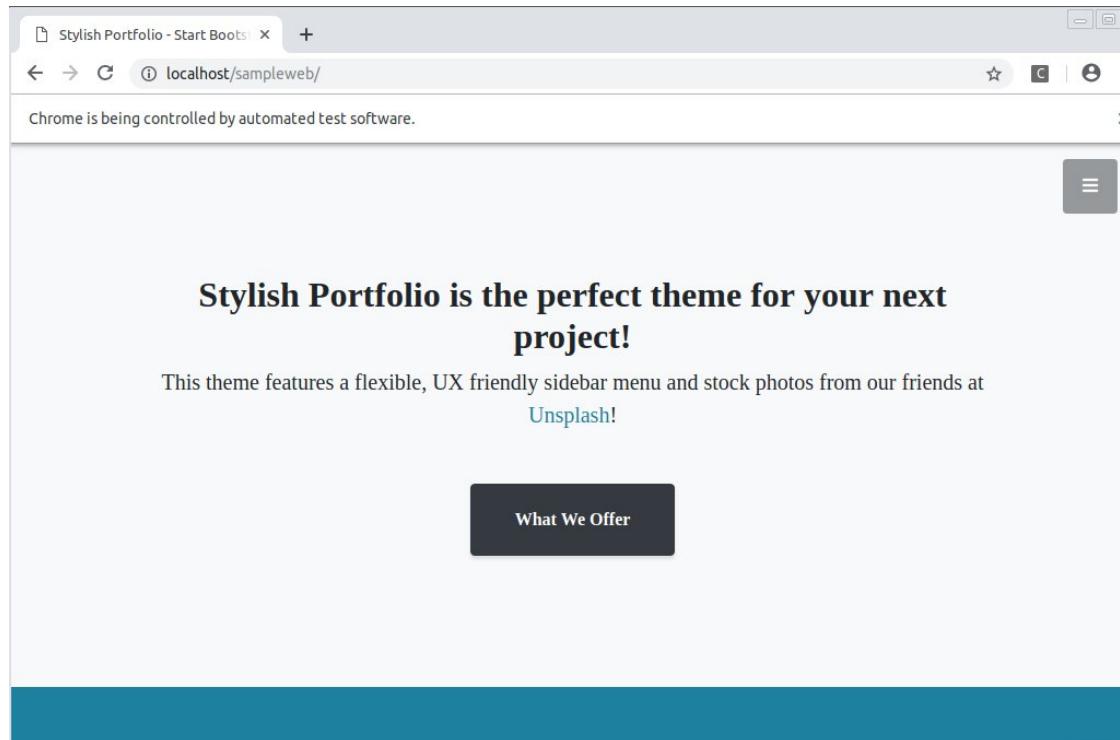
Kode di atas menyimpan list berupa objek untuk semua link di halaman pada `links`. Selanjutnya dilakukan looping untuk memeriksa masing masing teks link :

for link in links:

```
if link.text == "About":  
    link.click()  
    break
```

Jika link teks adalah “About” maka link akan diklik dan looping akan dibreak

Hasil eksekusi dari script di atas :



Link dengan teks “About” diklik dan halaman scroll ke bagian “About”.

- Pencarian elemen berdasarkan nama class

Untuk melakukan pencarian elemen berdasarkan nama class, kita menggunakan method `find_element_by_class_name()` dan `find_elements_by_class_name()`. Perbedaan antara keduanya adalah yang pertama akan menghasilkan objek dari elemen yang dicari sedangkan yang kedua akan menghasilkan list berupa objek dari elemen yang ditemukan.

Berikut ini contoh pada python console :

```
>>> from selenium import webdriver  
>>> import time  
>>>  
>>> your_chromium_path = "/usr/local/bin/chromedriver"  
>>> browser = webdriver.Chrome(your_chromium_path)  
>>> browser.get("http://localhost/sampleweb")  
>>> time.sleep(1)  
>>> start = browser.find_element_by_id("mulai")
```

```

>>> start.click()
>>> time.sleep(1)
>>> portfolios = browser.find_elements_by_class_name("js-scroll-trigger")
>>> for port in portfolios:
...     if port.text == "Portfolio":
...         port.click()
...
>>>

```

Penjelasan

```

portfolios = browser.find_elements_by_class_name("js-scroll-trigger")
for port in portfolios:
    if port.text == "Portfolio":
        port.click()

```

Rutin di atas berguna untuk mencari semua elemen dengan nama class : “js-scroll-triger”. Hasil dari `find_elements_by_class_name()` akan disimpan di list bernama “portfolios”.

Setelah itu dilakukan looping untuk mencari elemen dengan teks : “Portfolio” untuk kemudian diklik.

3. Eksekusi Java Script

Selenium memungkinkan eksekusi java script pada halaman web. Method yang digunakan untuk eksekusi java script : `execute_script()`

Untuk contoh buat script baru dengan nama `selenium_execute_script_sample.py` :

```

#!/usr/bin/env python3

"""
selenium_execute_script_sample.py
Example
for python training at www.jasaplus.com
"""

from selenium import webdriver
import time
your_chromium_path = "/usr/local/bin/chromedriver"
browser = webdriver.Chrome(your_chromium_path)
browser.get("http://localhost/sampleweb")

while True:
    script1 = "setTimeout(function(){ $('#mulai').click();$('#menu_about').click();}, 2000);"
    browser.execute_script(script1)

    script2 = "setTimeout(function(){ $('#mulai').click();$('#menu_services').click();}, 4000);"
    browser.execute_script(script2)

    script3 = "setTimeout(function(){ $('#mulai').click();$('#menu_contact').click();}, 6000);"
    browser.execute_script(script3)

    time.sleep(7)

```

Contoh di atas akan menyebabkan web melakukan self navigate secara berulang. Bagian eksekusi java script menggunakan rutin dari jquery karena web sampleweb menggunakan library jquery sehingga memungkinkan kita memanggil rutin rutin jquery.

4. Modifikasi / Pengaturan Input

Pada selenium, kita bisa melakukan modifikasi beberapa komponen inputan pengguna seperti input, textarea, select, checkbox.

Berikut ini contoh modifikasi input, contoh kali ini langsung kita ketikkan pada python console :

```
ringlayer@ringlayer-Inspiron-3442:~/Desktop/TRAINING/PYTHON/source/selenium$ python3
Python 3.5.2 (default, Nov 23 2017, 16:37:01)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>> from selenium import webdriver
>>> from selenium.webdriver.common.keys import Keys
>>> import time
>>> your_chromium_path = "/usr/local/bin/chromedriver"
>>> browser = webdriver.Chrome(your_chromium_path)
>>> browser.get("https://www.google.com")
>>> time.sleep(1)
>>> search = browser.find_element_by_name("q")
>>> search.clear()
>>> search.send_keys("hacker")
>>> search.send_keys(Keys.RETURN)
```

Berikut ini hasil eksekusi script di atas :

The screenshot displays a terminal session on the left and a web browser window on the right. The terminal window shows the execution of a Python script using the Selenium library to perform a Google search for the term 'hacker'. The browser window shows the search results for 'hacker' on Google, with the first result being a link to 'What The Hack 2019' on the kalibr.id website. Below the search results, there are three news cards related to WhatsApp security:

- WhatsApp Versi Baru Sudah Kebal Terhadap Hacker. Sudah Update?**
- Peringatan! Tanpa Disadari, WhatsApp Rawan Di-hack untuk**
- Waspadai WhatsApp Rentan Dibobol Hacker**

Penjelasan

```
>>> from selenium.webdriver.common.keys import Keys
```

Baris ini digunakan untuk mengimpor class Keys

```
>>> browser.get("https://www.google.com")
```

Pada baris ini objek browser melakukan get request ke google

```
>>> search = browser.find_element_by_name("q")
```

Baris ini digunakan untuk mencari elemen dengan nama q yang merupakan input untuk pencarian google

```
>>> search.clear()
```

Baris ini digunakan untuk melakukan clear input

```
>>> search.send_keys("hacker")
```

Baris ini digunakan untuk mengirimkan teks : “hacker” ke input pencarian

```
>>> search.send_keys(Keys.RETURN)
```

Baris ini akan mengirimkan konstan pada class Key yaitu : “RETURN” atau enter.

3.5. PEMROGRAMAN GUI PADA PYTHON

Untuk pemrograman GUI (Graphical User Interface) kali ini kita akan menggunakan tkinter. Tkinter bisa langsung diimport tanpa perlu menginstall apapun. Untuk mengimport tkinter :

```
import tkinter
```

atau bisa juga dengan mengimport komponen di dalam modul yang dibutuhkan, misal :

```
from tkinter import Tk
```

Perhatian !

Untuk efisiensi penggunaan memory sebaiknya jangan import * karena import * akan menyebabkan seluruh fungsi di dalam modul diimport sehingga memerlukan alokasi memori lebih.

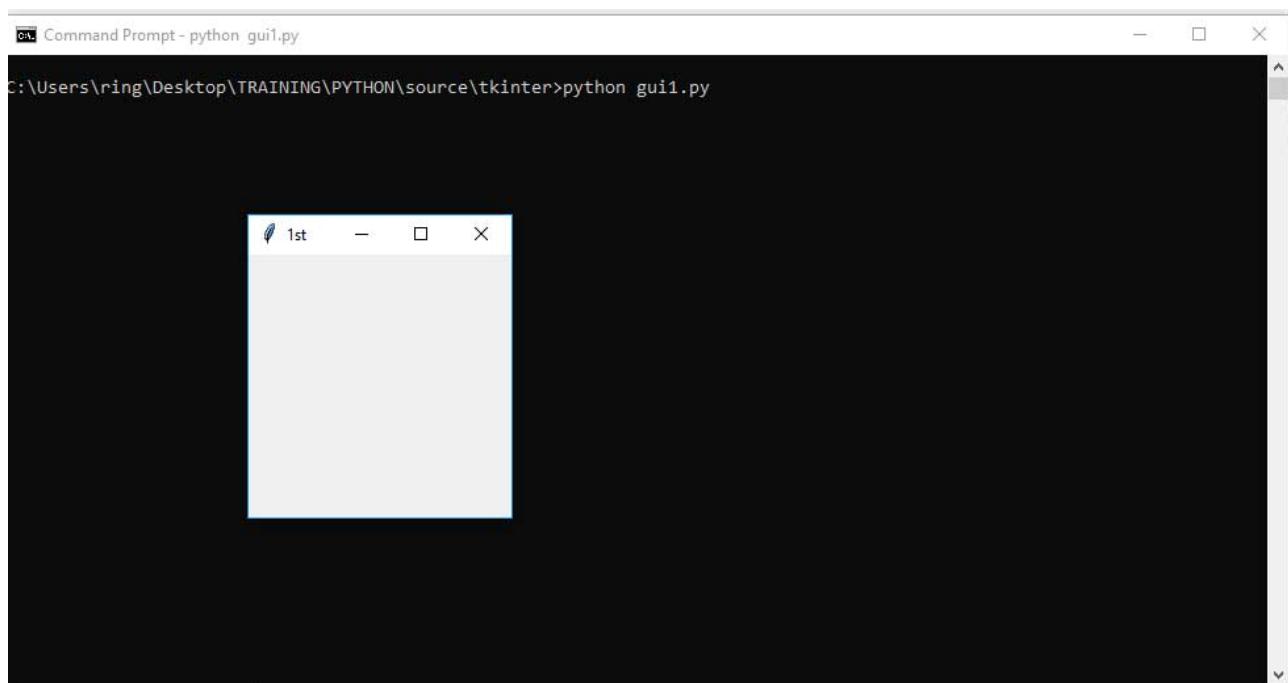
Berikut ini contoh aplikasi gui pertama kita dengan tkinter , buat script baru dengan nama gui1.py :

```
#!/usr/bin/env python3
"""
gui1.py
tkinter sample
for python3 training at jasaplus.com
"""

from tkinter import Tk

window = Tk()
window.title("1st")
window.mainloop()
```

Simpan dan jalankan file gui1.py :



Terlihat muncul aplikasi GUI dengan title yang telah kita tentukan

Penjelasan

```
from tkinter import Tk
```

Digunakan untuk mengimport class Tk dari modul tkinter.

```
window = Tk()
```

Pada baris di atas, kita membuat instance baru dari kelas Tk dengan nama instance window. Instance ini akan memiliki semua method dan atribut dari class Tk.

```
window.title("1st")
```

Selanjutnya kita memanggil method title di dalam class Tk, method ini digunakan untuk mengeset title dari aplikasi GUI.

```
window.mainloop()
```

mainloop() merupakan method yang terdapat pada class Tk agar window gui bisa ditampilkan. Kode ini bekerja seperti looping dan akan menyebabkan blocking terhadap potongan kode di bawahnya.

3.5.1. Pengaturan Window

Pada contoh kali ini kita akan mencoba beberapa pengaturan untuk mengatur tampilan window gui.

Berikut ini contoh untuk mengatur ukuran window , mengatur posisi di layar dan memberikan background color . Buat aplikasi baru dengan nama gui2.py :

```
#!/usr/bin/env python3
"""
gui2.py
tkinter sample
for python3 training at jasaplus.com
"""
from tkinter import Tk

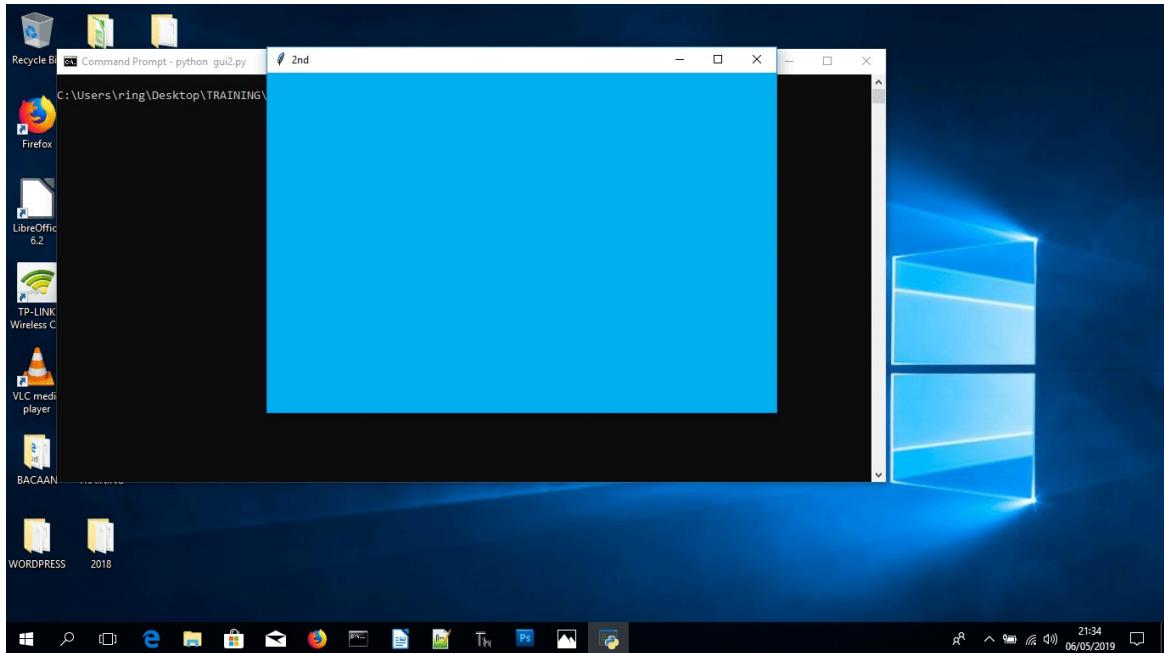
window = Tk()
window.title("2nd")

window.geometry("600x400+300+50")

window.configure(bg="#00aff0")

window.mainloop()
```

Jalankan aplikasi di atas. Berikut ini tampilan aplikasi di atas saat dijalankan :



Penjelasan

```
from tkinter import Tk
```

Pada baris ini kita import class Tk

```
window = Tk()
```

Pada baris ini, kita buat instance baru dari class Tk dengan nama window

```
window.title("2nd")
```

Di sini kita set title untuk window dengan teks 2nd

```
window.geometry("600x400+300+50")
```

geometry() merupakan method di dalam class Tk yang fungsinya untuk pengaturan posisi dan ukuran window. Di mana parameter berupa string :

width x height + position x + position y

jadi ukuran width adalah 600px, height adalah 400px, position x = 300px dari pinggir kiri layar dan position y = 50px dari atas layar.

```
window.configure(bg="#00aff0")
```

Pada baris ini, kita melakukan configure atribut warna background menjadi warna #00aff0
Selanjutnya buat script baru dengan nama gui3.py :

```
#!/usr/bin/env python3
""
```

```
gui3.py
tkinter sample
```

for python3 training at jasaplus.com

'''

```
from tkinter import Tk,Canvas, PhotoImage, Label
```

```
window = Tk()
```

```
window.title("3rd")
```

```
window.geometry("600x400+300+10")
```

```
C = Canvas(window, bg="white", height=600, width=400)
```

```
filename = PhotoImage(file = "images/bg.gif")
```

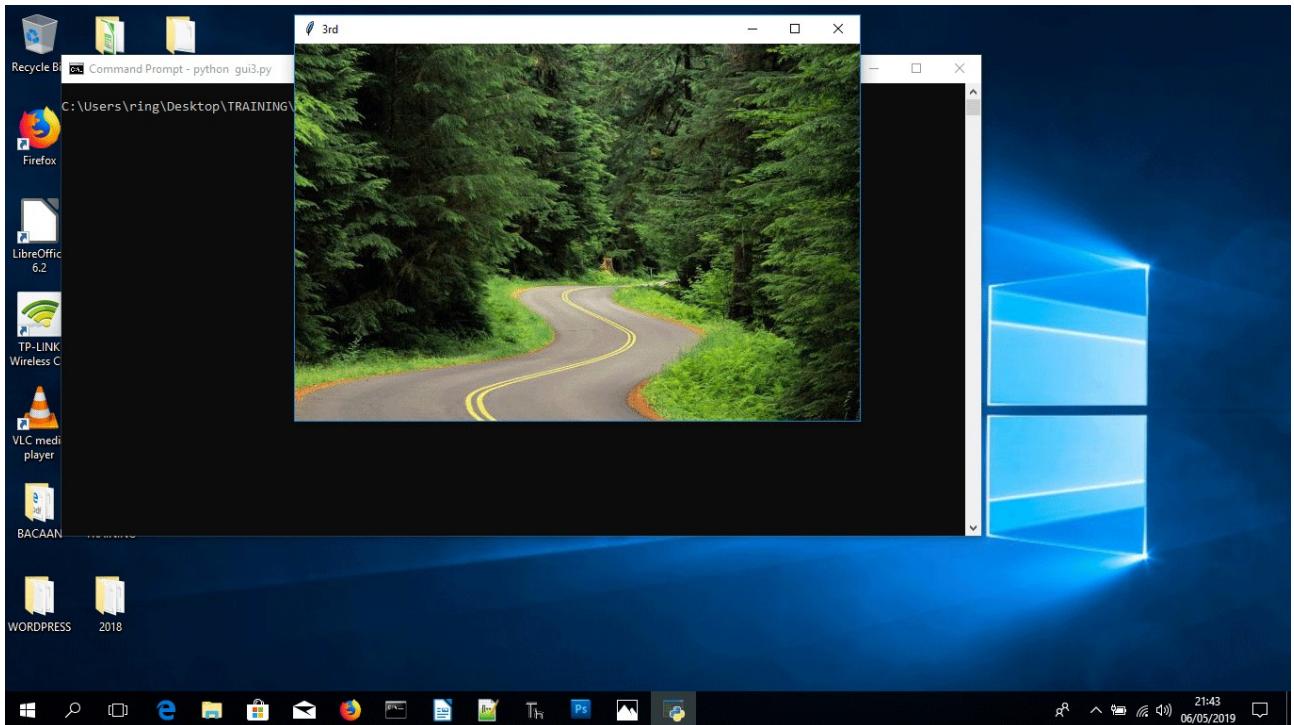
```
background_label = Label(window, image=filename)
```

```
background_label.place(x=0, y=0, relwidth=1, relheight=1)
```

```
C.pack()
```

```
window.mainloop()
```

Jalankan script di atas :



Penjelasan

```
C = Canvas(window, bg="white", height=600, width=400)
```

```
filename = PhotoImage(file = "images/bg.gif")
```

```
background_label = Label(window, image=filename)
```

```
background_label.place(x=0, y=0, relwidth=1, relheight=1)
```

```
C.pack()
```

Di sini kita membuat objek Canvas yang berguna membuat layer baru pada window, di mana layer tersebut memiliki background image. Agar background image bisa dirender dengan PhotoImage harus menggunakan image gif.

3.5.2. Layout Management

Berikut ini metode yang digunakan pada tkinter untuk pengaturan manajemen layout

1. pack manager

Pack manager merupakan pengaturan penempatan komponen pada layout di mana widget diposisikan berdasarkan blok-blok vertical dan horizontal. Layout diatur dengan opsi fill, expand, side, padx, pady, dan lain sebagainya.

Untuk melihat semua opsi pada penggunaan pack, kita bisa melihat dengan cara melihat fungsi pack pada salah satu objek, misal pada Button. Contoh dengan python shell :

```
>>> from tkinter import Button  
>>> help(Button.pack)
```

Berikut ini hasil eksekusi perintah di atas :

Help on function pack_configure in module tkinter:

```
pack_configure(self, cnf={}, **kw)  
    Pack a widget in the parent widget. Use as options:  
    after=widget - pack it after you have packed widget  
    anchor=NSEW (or subset) - position widget according to  
        given direction  
    before=widget - pack it before you will pack widget  
    expand=bool - expand widget if parent size grows  
    fill=NONE or X or Y or BOTH - fill widget if widget grows  
    in=master - use master to contain this widget  
    in_=master - see 'in' option description  
    ipadx=amount - add internal padding in x direction  
    ipady=amount - add internal padding in y direction  
    padx=amount - add padding in x direction  
    pady=amount - add padding in y direction  
    side=TOP or BOTTOM or LEFT or RIGHT - where to add this widget.  
(END)
```

Contoh penggunaan pack manager, buat file baru dengan nama pack_manager.py :

```
#!/usr/bin/env python3  
"  
pack_manager.py  
tkinter pack manager sample  
for python3 training at www.jasaplus.com  
"  
  
from tkinter import Tk, Button, Frame, LEFT, RIGHT, BOTTOM, TOP, RAISED, X, Y, BOTH  
  
master = Tk()  
master.title("Pack Manager")  
master.geometry("600x400+300+10")  
  
frame1 = Frame(master, relief=RAISED, borderwidth=1, bg="#3b6a12", height='200')  
frame1.pack(fill=X)  
  
#side option  
btn1 = Button(frame1, text="Button 1")  
btn1.pack(side=LEFT)
```

```

btn2 = Button(frame1, text="Button 2")
btn2.pack(side=RIGHT)

btn3 = Button(frame1, text="Button 3")
btn3.pack(side=TOP)

btn4 = Button(frame1, text="Button 4")
btn4.pack(side=BOTTOM)

frame2 = Frame(master, relief=RAISED, borderwidth=1, bg='#8bde41',height='200')
frame2.pack(fill=X)

#fill option
btn1x = Button(frame2, text="Button 1")
btn1x.pack(fill=X)

btn1x = Button(frame2, text="Button 1")
btn1x.pack(fill=None)

master.mainloop()

```

Penjelasan

Pada contoh aplikasi ini kita memiliki 2 buah frame yaitu :

```

frame1 = Frame(master, relief=RAISED, borderwidth=1, bg='#3b6a12',height='200')
frame1.pack(fill=X)

frame2 = Frame(master, relief=RAISED, borderwidth=1, bg='#8bde41',height='200')
frame2.pack(fill=X)

```

Frame 1 berada di atas frame 2 karena objek dibuat sebelum frame 2. Objek ditempatkan dengan pack manager dengan opsi fill = X , opsi ini akan menyebabkan frame mengisi sepanjang axis x pada parentnya, parentnya merupakan instance objek Tk dengan identifier “master”.

Frame 1 memiliki tinggi 200 pixel, frame 2 memiliki tinggi 200.

Pada Frame 1 terdapat objek objek child di dalamnya yaitu :

```

#side option
btn1 = Button(frame1, text="Button 1")
btn1.pack(side=LEFT)

```

Objek btn1 ditempatkan di layout dengan pack, opsi side=LEFT akan menyebabkan widget ditaruh di kiri

```

btn2 = Button(frame1, text="Button 2")
btn2.pack(side=RIGHT)

```

Objek btn2 ditempatkan di layout dengan pack, opsi side=RIGHT akan menyebabkan widget ditaruh di kanan

```

btn3 = Button(frame1, text="Button 3")
btn3.pack(side=TOP)

```

Objek btn3 ditempatkan di layout dengan pack, opsi side=TOP akan menyebabkan widget ditaruh di atas

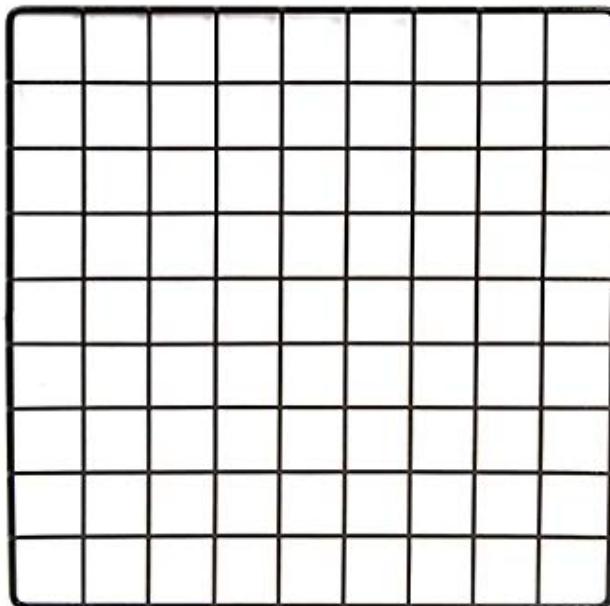
```
btn4 = Button(frame1, text="Button 4")
btn4.pack(side=BOTTOM)
```

Objek btn4 ditempatkan di layout dengan pack, opsi side=BOTTOM akan menyebabkan widget ditaruh di bagian bawah

2. grid manager

Grid manager merupakan pengaturan penempatan komponen pada layout di mana widget diletakkan berdasarkan grid 2 dimensi mirip seperti kalkulator / table dengan kolom dan baris.

Pada grid manager objek diatur posisi baris dan kolomnya untuk penempatanya.



Contoh penggunaan grid manager , buat file baru dengan nama grid_manager.py :

```
#!/usr/bin/env python3
"""
grid_manager.py
tkinter grid manager sample
for python3 training at www.jasaplus.com
"""

from tkinter import Tk,Button

master = Tk()
master.title("Grid Manager")
master.geometry("600x400+300+10")

btn1 = Button(master, text="Button 1")
btn1.grid(row=1, column=1)
```

```
btn2 = Button(master, text="Button 2")
btn2.grid(row=1, column=2)
```

```
btn3 = Button(master, text="Button 3")
btn3.grid(row=1, column=3)
```

```
btn4 = Button(master, text="Button 4")
btn4.grid(row=2, column=1)
```

```
btn5 = Button(master, text="Button 5")
btn5.grid(row=2, column=2)
```

```
btn6 = Button(master, text="Button 6")
btn6.grid(row=2, column=3)
```

```
master.mainloop()
```

Penjelasan

```
btn1 = Button(master, text="Button 1")
btn1.grid(row=1, column=1)
```

Posisi Button ditempatkan pada baris 1 kolom ke 1

```
btn2 = Button(master, text="Button 2")
btn2.grid(row=1, column=2)
```

Posisi Button ditempatkan pada baris 1 kolom ke 2

```
btn3 = Button(master, text="Button 3")
btn3.grid(row=1, column=3)
```

Posisi Button ditempatkan pada baris 1 kolom ke 3

```
btn4 = Button(master, text="Button 4")
btn4.grid(row=2, column=1)
```

Posisi Button ditempatkan pada baris 1 kolom ke 1

```
btn5 = Button(master, text="Button 5")
btn5.grid(row=2, column=2)
```

Posisi Button ditempatkan pada baris 1 kolom ke 2

```
btn6 = Button(master, text="Button 6")
btn6.grid(row=2, column=3)
```

Posisi Button ditempatkan pada baris 1 kolom ke 3

3. place manager

Place manager merupakan pengaturan penempatan widget di dalam layout secara absolut mirip seperti pada css dengan posisi absolute. Komponen disusun berdasarkan sumbu axis x dan y.

Contoh penggunaan place manager :

```
#!/usr/bin/env python3
"""
place_manager.py
tkinter place manager sample
for python3 training at www.jasaplus.com
"""

from tkinter import Tk,Button

master = Tk()
master.title("Place Manager")
master.geometry("600x400+300+10")

btn1 = Button(master, text="Button 1")
btn1.place(x=1, y=200)

btn2 = Button(master, text="Button 2")
btn2.place(x=100, y=200)

btn3 = Button(master, text="Button 3")
btn3.place(x=200, y=200)

master.mainloop()
```

Penjelasan

```
btn1 = Button(master, text="Button 1")
btn1.place(x=1, y=200)
```

Objek btn1 ditempatkan pada posisi x = 1 dan y = 200

```
btn2 = Button(master, text="Button 2")
btn2.place(x=100, y=200)
```

Objek btn2 ditempatkan pada posisi x = 100 dan y = 200

```
btn3 = Button(master, text="Button 3")
btn3.place(x=200, y=200)
```

Objek btn3 ditempatkan pada posisi x = 200 dan y = 200

Keterangan : semakin besar nilai y maka semakin ke bawah suatu objek, semakin besar nilai x maka semakin ke kanan posisi dari objek (penempatan dengan place() bersifat absolute positioning).

3.5.3. Membuat dan Mengatur Widget

Pada contoh kali ini, kita akan mencoba membuat dan mengatur widget pada window.

Berberapa widget yang akan kita gunakan antara lain :

label , entry, button, checkbutton, listbox, menubutton, text dan tkmessagebox

1. Membuat Label

Digunakan untuk membuat label teks pada tampilan GUI aplikasi.

Pola :

LabelInstance = tkinterObject.Label(TkInstance, string)

Untuk contoh, buat aplikasi baru dengan nama label_sample.py :

```
#!/usr/bin/env python
import tkinter as tk
root = tk.Tk()
root.title("label sample")
root.geometry("600x400+300+10")
w = tk.Label(root, text="Label Sample")
w.pack()
root.mainloop()
```

Penjelasan

```
import tkinter as tk
```

Pertama tama kita mengimport objek tkinter dengan alias tk.

```
root = tk.Tk()
```

Selanjutnya kita buat object instance dari tk

```
root.title("label sample")
root.geometry("600x400+300+10")
```

Rutin di atas adalah penggunaan method dari class tkinter yaitu title untuk mengatur title aplikasi gui, dan method geometry yang berguna untuk mengatur posisi jendela aplikasi gui.

```
w = tk.Label(root, text="Label Sample")
w.pack()
```

Rutin di atas digunakan untuk membuat label, pack() digunakan untuk menggunakan tata letak berdasarkan penataan kotak kotak horizontal dan vertical

2. Membuat Entry

Digunakan untuk membuat input box mirip Text (contoh Text ada di bawah), untuk jenis input yang lebih memudahkan pengaturan propertinya bisa menggunakan widget Text sebagai alternatif widget ini.

Pola :

EntryInstance = tkinterObject.Entry(TkInstance)

Untuk contoh entry, buat aplikasi baru dengan nama entry_sample.py :

```
#!/usr/bin/env python3
from tkinter import Label, Entry, Tk, mainloop
master = Tk()
master.title("entry sample")
master.geometry("600x400+300+10")
e1 = Entry(master)
e2 = Entry(master)
e1.place(x = 200, y = 50)
e2.place(x = 200, y = 100)
e1.insert(0,"Username")
e2.insert(0,"Password")
mainloop()
```

Penjelasan

e1 = Entry(master)

Rutin ini berguna untuk membuat instance object baru dari class Entry

e1.place(x = 200, y = 50)

Selanjutnya kita menggunakan penataan layout secara absolute (seperti pada css html), di mana posisi pojok paling kiri dari objek e1 dimulai pada x = 200 pixel dan posisi pojok paling atas dari objek e1 terdapat pada y = 500 pixel.

Keterangan : semakin besar nilai y maka semakin ke bawah suatu objek, semakin besar nilai x maka semakin ke kanan posisi dari objek (penempatan dengan place() bersifat absolute positioning).

e1.insert(0,"Username")

Rutin di atas berguna untuk menyisipkan string placeholder pada entry

3. Membuat Button

Berfungsi menampilkan widget berupa tombol pada aplikasi gui dengan tkinter.

Pola :

ButtonInstance = tkinterObject.Button(TkInstance, string, command)

Untuk contoh, buat aplikasi baru dengan nama button_sample.py :

```
#!/usr/bin/env python3
"""
button_sample.py
for course material at www.jasaplus.com
"""
from tkinter import Tk, Button, messagebox

def OnClick(num):
    messagebox.showinfo("Information", "I got clicked at " + str(num))

window = Tk()
window.title("button sample")
window.geometry("600x400+300+10")
btn = Button(window, text="Click Button 1!", command=lambda:OnClick(1))
btn.place(x = 200, y = 50)
btn2 = Button(window, text="Click Button 2 !", command=lambda:OnClick(2))
btn2.place(x = 200, y = 100)
window.mainloop()
```

Penjelasan

```
btn = Button(window, text="Click Button 1!", command=lambda:OnClick(1))
btn.place(x = 200, y = 50)
```

Rutin di atas berfungsi untuk membuat Button dengan penempatan secara absolute pada posisi x = 200 dan y = 50

4. Membuat Checkbutton

Checkbutton merupakan widget yang berfungsi untuk menampilkan checkbox seperti pada form html.

Pola :

CheckButtonInstance = tkinterObject.CheckButton(TkInstance, string, variable)

Untuk lebih jelasnya, buat aplikasi baru dengan nama checkbutton_sample.py :

```
#!/usr/bin/env python3
"""
checkbutton_sample.py
for course material at www.jasaplus.com
"""
from tkinter import Tk, Checkbutton, IntVar, BooleanVar
window = Tk()
window.title("Checkbutton sample")
window.geometry("600x400+300+10")
var = IntVar()
c1 = Checkbutton(window, text=" Check 1", variable=var)
c1.place(x = 200, y = 100)

c2Val = BooleanVar()
c2Val.set(True)
c2 = Checkbutton(window, text=' Check 2', var=c2Val)
c2.pack()
```

```
window.mainloop()
```

Penjelasan

```
c1 = Checkbutton(window, text=" Check 1", variable=var)
c1.place(x = 200, y = 100)
```

Rutin di atas berguna untuk membuat widget Checkbutton dengan teks “ Check 1” di mana objek ini ditempatkan secara absolut pada x = 200 dan y = 100

```
c2Val = BooleanVar()
c2Val.set(True)
c2 = Checkbutton(window, text=' Check 2', var=c2Val)
c2.pack()
```

Rutin di atas berguna untuk membuat Checkbutton yang dicentang secara default, penempatan komponen ini dilakukan dengan pack manager di mana tata letak berdasarkan kotak kotak horizontal dan vertical.

5. Membuat Listbox

Listbox merupakan widget yang berfungsi untuk menampilkan pilihan berupa list.

Pola :

```
ListBoxInstance = tkinterObject.ListBox(TkInstance)
```

Buat aplikasi baru dengan nama listbox_sample.py

```
#!/usr/bin/env python3
"""
_listbox.py
for course material at www.jasaplus.com
"""

from tkinter import Tk, Listbox, END

window = Tk()
window.title("button sample")
window.geometry("600x400+300+10")

listbox = Listbox(window)
listbox.place(x = 200, y = 50)
listbox.insert(END, "select menu")

for item in ["cheese", "burger", "ice cream"]:
    listbox.insert(END, item)

window.mainloop()
```

Penjelasan

```
listbox = Listbox(window)
listbox.place(x = 200, y = 50)
listbox.insert(END, "select menu")
```

Digunakan untuk menampilkan Listbox, selanjutnya Listbox ditempatkan secara absolut pada x = 200 dan y=500. Pada listbox ditambahkan pilihan teks : “select menu”

```
for item in ["cheese", "burger", "ice cream"]:  
    listbox.insert(END, item)
```

Penambahan komponen dari Listbox dari masing masing komponen pada list

6. Membuat Menubutton

Menubutton merupakan widget yang berfungsi untuk menampilkan tombol dengan pilihan submenu.

Pola :

```
ListBoxInstance = tkinterObject.MenuButton(TkInstance, string)
```

Buat aplikasi baru dengan nama menubutton_sample.py

```
#!/usr/bin/env python3  
"  
menubutton_sample.py  
for course material at www.jasaplus.com  
"  
from tkinter import Tk, Menubutton, Menu, RAISED  
  
def page(container):  
    print("Page selected: " + container)  
  
window = Tk()  
window.title("Menubutton sample")  
window.geometry("600x400+300+10")  
window.configure(bg="#00aff0")  
mbutton = Menubutton(window, text="Menu")  
picks = Menu(mbutton)  
mbutton.config(menu=picks)  
picks.add_command(label='Home', command=lambda:page("home"))  
picks.add_command(label='About', command=lambda:page("about"))  
picks.add_command(label='Contact', command=lambda:page("contact"))  
mbutton.place(x = 10, y = 20)  
mbutton.config(bg='#ffffff', bd=1, relief=RAISED)  
window.mainloop()
```

Penjelasan

```
mbutton = Menubutton(window, text="Menu")
```

mbutton merupakan object instance dari Menubutton, di sini kita siapkan untuk menampilkan widget Menubutton nanti (objek widget ini akan muncul setelah dilakukan penempatan oleh pack manager atau place manager atau grid manager).

```
picks = Menu(mbutton)
```

Selanjutnya kita buat objek instance dari Menu, di mana untuk membuat objek Menu ini kita berikan parameter berupa objek instance dari kelas Menubutton, objek instance dari Menu ini nanti akan berguna untuk menampilkan pilihan menu saat terjadi event berupa klik mouse pada objek Menubutton.

```
mbutton.config(menu=picks)
```

Method config merupakan method class Menubutton yang dapat menerima parameter berupa object instance menu, dengan pemberian parameter ini berguna untuk menambahkan tampilan menu saat objek Menubutton diklik

```
picks.add_command(label='Home', command=lambda:page("home"))
picks.add_command(label='About', command=lambda:page("about"))
picks.add_command(label='Contact', command=lambda:page("contact"))
```

add_command() adalah method dari class Menu di mana di sini ditambahkan label string, parameter command digunakan untuk memberikan perintah callback yang akan dipanggil saat Menu diklik, pada contoh ini kita gunakan lambda function karena fungsi yang akan dipanggil memerlukan parameter

```
mbutton.place(x = 10, y = 20)
mbutton.config(bg='#ffffff', bd=1, relief=RAISED)
```

Rutin di atas berguna untuk pengaturan posisi dan tampilan mbutton

7. Membuat Text

Untuk lebih jelasnya, berikut ini contoh aplikasi dengan widget Text untuk input user.

Pola :

```
TextInstance = tkinterObject.Text(TkInstance, height, width)
```

Buat aplikasi baru dengan nama text_sample.py :

```
#!/usr/bin/env python3
"""
text_sample.py
for course material at www.jasaplus.com
"""

from tkinter import Tk, Text, END

window = Tk()
window.title("text sample")
window.geometry("600x400+300+10")

Teks = Text(window, height=2, width=30)
Teks.pack()
Teks.insert(END, "first line\n\ttwo line\n\t\tthree line")

Teks2 = Text(window, height=5, width=40)
Teks2.insert(END, "first line\n\ttwo line\n\t\tthree line")
Teks2.place(x = 200, y = 100)
window.mainloop()
```

Penjelasan

```
Teks = Text(window, height=2, width=30)
Teks.pack()
Teks.insert(END, "first line\n\ttwo line\n")
```

Rutin di atas berguna untuk membuat widget berupa inputan teks di mana penempatan menggunakan pack manager (berdasarkan kotak kotak horizontal dan vertical)

8. Membuat TkMessageBox

TkMessageBox digunakan untuk membuat popup pada aplikasi gui dengan tkinter.

Pola :

tkinterObject.messagebox.messageboxFunction(string)

Untuk mengimport sub modul messagebox dari tkinter :

```
from tkinter import Tk, Button, messagebox
```

Beberapa fungsi yang bisa digunakan dari modul messagebox :

showinfo()

Digunakan menampilkan pop up berupa info

showwarning()

Digunakan menampilkan pop up berupa warning

showerror ()

Digunakan menampilkan pop up berupa error

askquestion()

Digunakan menampilkan pop up berupa pertanyaan

askokcancel()

Digunakan menampilkan pop up berupa ok atau cancel

askyesno ()

Digunakan menampilkan pop up berupa yes atau no

askretrycancel ()

Digunakan menampilkan pop up berupa retry atau cancel

Untuk lebih jelasnya, buat contoh aplikasi dengan nama popup_sample.py

```
#!/usr/bin/env python3
"""
popup_sample.py
```

```
tkinter sample
for python3 training at jasaplus.com
"""
from tkinter import Tk, Button, messagebox

def OnClick():
    messagebox.showinfo("Information", "Info !")
    messagebox.showerror("Error", "Error !")
    messagebox.askokcancel("OK CANCEL", "Ok or Cancel")

window = Tk()
window.title("Pop up Sample")
window.geometry("600x400+300+50")
window.configure(bg="#00aff0")

btn = Button(window, text="Click Me !", command=OnClick)
btn.place(x = 200, y = 50)

window.mainloop()
```

Penjelasan

```
btn = Button(window, text="Click Me !", command=OnClick)
btn.place(x = 200, y = 50)
```

Rutin di atas berguna untuk membuat objek Button di mana saat diklik maka callback function “OnClick” akan dipanggil.

```
def OnClick():
    messagebox.showinfo("Information", "Info !")
    messagebox.showerror("Error", "Error !")
    messagebox.askokcancel("OK CANCEL", "Ok or Cancel")
```

Callback function OnClick akan menampilkan 3 messagebox (pesan popup) dengan 3 fungsi berbeda

3.6. MULTITHREADING DAN MULTIPROCESSING PADA PYTHON

3.6.1. Multi Threading

Multithreading pada aplikasi python adalah kemampuan suatu aplikasi yang dibuat dengan python untuk menjalankan lebih dari satu pekerjaan dengan menggunakan thread yang berbeda.

Untuk multithreading di python3 kali ini kita akan menggunakan modul threading.

1. Mempersiapkan Thread Baru

Untuk mempersiapkan thread baru, kita membuat instance object baru dari class Thread() , dengan parameter berupa worker. Sebelumnya pastikan sudah mengimport modul threading : import threading

Contoh :

```
t = threading.Thread(target=f)
```

Pada saat pembuatan instance object dari class Thread(), parameter lengkap yang dibutuhkan bisa terlihat dari initializer untuk class Thread :

```
__init__(self, group=None, target=None, name=None, args=(), kwargs=None, *, daemon=None)
```

Yang biasa dibutuhkan adalah parameter :

- target : berupa objek yang bisa dipanggil (callable)
- name : merupakan nama thread (bersifat opsional)
- args : merupakan argumen, misal callable berupa fungsi maka yang diberikan adalah argumen / parameter fungsi tersebut

2. Mulai Menjalankan Thread

Untuk menjalankan thread yang sudah dipersiapkan melalui instance object thread, kita tinggal memanggil method start()

Contoh :

```
t.start()
```

3. Menunggu Hingga Semua Thread Selesai

Untuk menunggu hingga semua proses thread selesai dieksekusi, kita tinggal memanggil method join() . Dengan thread join maka thread __main__ pada suatu aplikasi , atau jalanya program utama (untuk lebih simple dijelaskannya) akan terhenti (blocking) hingga seluruh proses dari thread yang dijalankan sebelumnya selesai dijalankan.

Contoh :

misal kita memiliki 3 thread yang tersimpan dalam list dengan identifier nama : "threadlists"

```
for TheThread in threadlists:  
    TheThread.join()
```

Untuk contoh, buat file baru dengan nama threadsample.py

```
#!/usr/bin/python3  
""  
threadsaple.py  
Sample multithreading application for python course material on www.jasaplus.com  
""  
import threading, socket, time  
  
def task(host, port):  
    try:  
        data = "GET / HTTP/1.1\r\nHost: " + host + ":80\r\nUser-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:65.0)  
Gecko/20100101 Firefox/65.0\r\nConnection: keep-alive\r\n\r\n"  
        print("[+] connect to " + host + " on port", port)  
        sock = socket.socket(socket.AF_INET,socket.SOCK_STREAM)  
        sock.connect((host, port))  
        sock.sendall(data.encode('utf-8'))  
        resp = sock.recv(4096)  
        if resp:  
            print("[+] From Server : ", repr(resp.decode('utf-8')))  
    except:  
        raise  
  
if __name__ == "__main__":  
    print("[+] python3 multi threading sample")  
    threadlists = []  
    hosts = ["ringlayer.com", "jasaplus.com"]  
    t1 = threading.Thread(target=task, args=(hosts[0], 80))  
    threadlists.append(t1)  
    t2 = threading.Thread(target=task, args=(hosts[1], 80))  
    threadlists.append(t2)  
    for TheThread in threadlists:  
        TheThread.start()  
  
    for TheThread in threadlists:  
        TheThread.join()
```

Penjelasan

```
threadlists = []
```

Pada contoh di atas, sebelum mulai multi threading, kita persiapkan list untuk menyimpan objek instance dari Thread() (masing masing objek untuk 1 thread)

```
t1 = threading.Thread(target=task, args=(hosts[0], 80))
threadlists.append(t1)
t2 = threading.Thread(target=task, args=(hosts[1], 80))
threadlists.append(t2)
```

Selanjutnya kita buat 2 object instance dari class Thread masing masing bernama t1 dan t2 yang kemudian disimpan di list “threadlists”

```
for TheThread in threadlists:
    TheThread.start()
```

Langkah selanjutnya adalah tinggal menjalankan masing masing object tadi yang tersimpan di list “threadlists” dengan method start()

3.6.2. Multi Processing

Multiprocessing pada python pada dasarnya adalah sama dengan multi threading yang membedakan adalah multi threading menggunakan thread sedangkan multi processing menggunakan process.

Multi Processing akan memerlukan process yang berbeda untuk mengeksekusi masing masing tugas, di mana masing proses memiliki memory space yang berbeda beda, sedangkan pada multi threading masing tugas berbeda akan dikerjakan oleh thread tersendiri, di mana masing thread masih menggunakan memory space yang sama.

Untuk multiprocessing di python3, kali ini kita akan menggunakan modul multiprocessing. Pada contoh program dengan multiprocessing kali ini kita akan menggunakan modul multiprocessing.

1. Memulai

Untuk memulai kita perlu mengimport modul multiprocessing :

```
import multiprocessing
```

2. Mempersiapkan objek instance dari class multiprocessing.context.Process

Karena pada dasarnya multiprocessing ini akan melakukan fork / membuat proses baru maka kita perlu membuat proses baru. Untuk masing proses yang akan buat akan direpresentasikan dengan satu objek instance dari class Process. Untuk membuat object instance baru contoh :

```
p = multiprocessing.Process(target=callable_worker, args=(argument1, argument2))
```

Pada contoh di atas, callable_worker adalah suatu objek yang bisa dipanggil, bisa berupa function atau method, argument diberikan dalam bentuk tuple jika ada (bersifat opsional)

3. Mulai menjalankan suatu proses

Untuk mulai menjalankan suatu proses yang telah disiapkan sebelumnya, kita tinggal mengeksekusi method start() .

Contoh :

```
p.start()
```

Di mana "p" merupakan instance object dari class Process di dalam modul multiprocessing. Instance object dari class Process ini mewakili suatu proses yang akan dibuat dan berfungsi sebagai objek yang khusus menangani proses tersebut.

Untuk contoh lengkap, buat file baru dengan nama mpsample.py

```
#!/usr/bin/python3
"""
mpsample.py
Sample multiprocessing application for python course material on www.jasaplus.com
"""

import multiprocessing, psutil, time, os

def PsutilRoutines():
    try:
        while True:
            print("[+] cpu usage percent:", psutil.cpu_percent(interval=1.0))
            time.sleep(1.5)
            process = psutil.Process(os.getpid())
            print("[+] memory usage percent:", process.memory_percent())
            time.sleep(1.5)
            print(psutil.sensors_battery())
            time.sleep(1.5)
    except:
        raise

def PointLessRoutines(param):
    while True:
        print("hi I'm pointless routine, I got param : " + param)
        time.sleep(1)

if __name__ == "__main__":
    print("[+] python3 multi processing sample")
    processes = []
    for start in range(2):
        proc = multiprocessing.Process(target=PointLessRoutines, args=(str(start),))
        processes.append(proc)
        proc.start()
    proc_cpu = multiprocessing.Process(target=PsutilRoutines)
    processes.append(proc_cpu)
    proc_cpu.start()
```

Penjelasan

Pada aplikasi di atas, kita menggunakan tambahan modul psutil yang akan kita gunakan pada operasi operasi di dalam fungsi PsutilRoutines

```
print("[+] cpu usage percent:", psutil.cpu_percent(interval=1.0))
```

Rutin di atas berfungsi menghitung persentasi penggunaan cpu

```
process = psutil.Process(os.getpid())
print("[+] memory usage percent:", process.memory_percent())
```

Rutin di atas berfungsi menghitung persentasi penggunaan memory

```
print(psutil.sensors_battery())
```

Rutin di atas berfungsi mengukur sisa baterai (jika menggunakan laptop)

Selanjutnya pada thread `__main__` , kita mulai operasi utama untuk melakukan multiprocessing :

```
if __name__ == "__main__":
    print("[+] python3 multi processing sample")
    processes = []
    for start in range(2):
        proc = multiprocessing.Process(target=PointLessRoutines, args=(str(start),))
        processes.append(proc)
        proc.start()
    proc_cpu = multiprocessing.Process(target=PsutilRoutines)
    processes.append(proc_cpu)
    proc_cpu.start()
```

3.7. NUMPY

Numpy adalah library python yang banyak digunakan untuk matematika dan data science. Pada dasarnya numpy adalah library matematika di python.

Numpy menyediakan beberapa fasilitas / kemampuan :

1. Objek berupa array (n-dimensional array)
2. Perhitungan cepat matematika dengan operasi array
3. Aljabar linear, Transformasi Fourier, Pembuatan angka acak.

Untuk menginstall numpy dengan pip ketikkan :

```
pip install numpy
```

Untuk mulai menggunakan numpy pada script python kita tinggal melakukan import numpy,

Contoh:

```
import numpy as np
```

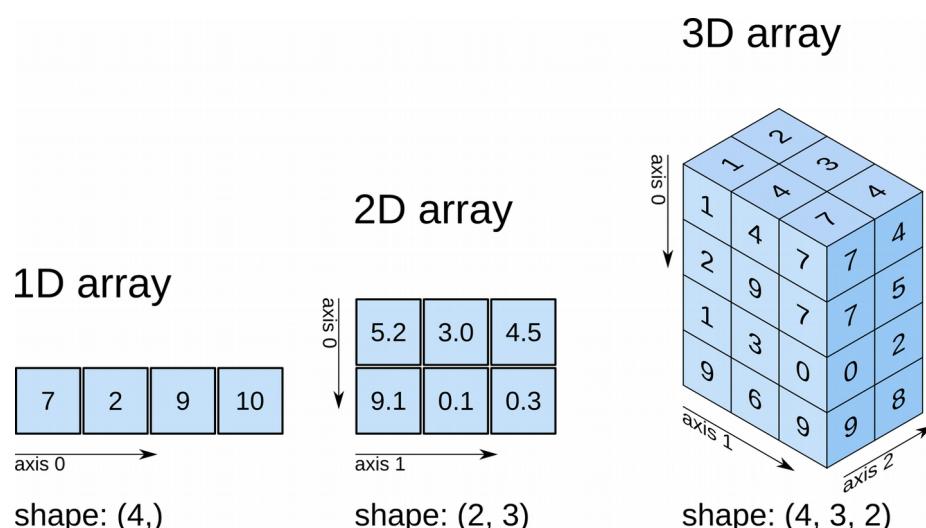
3.7.1. Dasar Numpy

Numpy Ndarray

Array n-dimensional merupakan jantung operasi dari modul numpy. Numpy Ndarray merupakan suatu objek dari modul numpy untuk merepresentasikan objek array. Objek ini merupakan array multi dimensi (n-dimensional array).

Untuk membuat suatu array multi dimensi (n-dimensional array), kita bisa menggunakan fungsi built-in dari modul numpy yaitu : array()

Untuk gambaran dimensi pada numpy array ini perhatikan image ini :



Berikut ini adalah beberapa contoh penggunaan

1. Membuat array dari list

Contoh 1

```
import numpy as np  
a = np.array([1,2,3])
```

Hasil :

```
[1 2 3]
```

Contoh 2

```
import numpy as np  
lis = [1,2,3]  
a = np.array(lis)
```

2. Membuat array dari tuple

Contoh 1

```
import numpy as np  
a = np.array((2, 4, 5))
```

Hasil :

```
[2 4 5]
```

Contoh 2

```
import numpy as np  
tup = (3, 6, 9)  
np.array(tup)
```

3. Membuat array 1d

Contoh

```
import numpy as np  
a = np.array([1,2,3])
```

4. Membuat array 2d

Contoh

```
import numpy as np  
a = np.array( [[0,1,2], [3,4,5], [6,7,8]])
```

Hasil :

```
[[0 1 2]  
[3 4 5]  
[6 7 8]]
```

Contoh 2

```
import numpy as np  
l = [[0,1,2], [3,4,5], [6,7,8]]  
a = np.array(l)
```

Berikut ini adalah beberapa fungsi built-in dari modul numpy yang sering digunakan :

1. zeros()

fungsi ini merupakan fungsi built-in dari modul numpy, digunakan untuk membuat array / matriks dengan bentuk dan tipe yang telah dispesifikasi dengan parameter fungsi ini, di mana masing masing elemen dari array / matriks yang akan dihasilkan adalah nol.

Sintaks

```
np.zeros(shape, dtype=float, order='C')
```

Keterangan

- *shape* : *integer atau sequence integer*

- *dtype* : *tipe data output yang dihasilkan (opsional)*

- *order* : {'C', 'F'} (optional), merupakan pengaturan penyimpanan data di memori, apakah akan menggunakan style c atau style fortran

Untuk melihat manual dari built-in function numpy zeros, bisa dilakukan dengan cara berikut ini pada shell python :

```
>>> import numpy as np  
>>> help(np.zeros)
```

Contoh

```
#!/usr/bin/env python  
""  
numpy zeros sample  
for python training at www.jasaplus.com  
"  
import numpy as np
```

```

a1 = np.zeros(1, dtype=int)
print(a1)

print("=" * 50)

a2 = np.zeros(2, dtype=int)
print(a2)

print("=" * 50)

a3 = np.zeros([1, 1], dtype=int)
print(a3)

print("=" * 50)

a4 = np.zeros([1, 2], dtype=int)
print(a4)

print("=" * 50)

a5 = np.zeros([2, 1], dtype=int)
print(a5)

print("=" * 50)

a5 = np.zeros([2, 2], dtype=int)
print(a5)

```

Jalankan script di atas, berikut ini hasil yang ditampilkan :

```

ringlayer@ringlayer-Inspiron-3442:~/Desktop/TRAINING/PYTHON/source/numpy$ ./numpy_zeros.py
[0]
=====
[[0]]
=====
[[[0]]]
=====
[[[[0]]]]
=====
[[[[0, 0]]]]
=====
```

Keterangan :

```
a1 = np.zeros(1, dtype=int)
print(a1)
```

Hasil yang ditampilkan :

[0]

pada contoh di atas, kita hanya menginput

```
a2 = np.zeros(2, dtype=int)
print(a2)
```

Hasil yang ditampilkan :

```
[0 0]
```

```
a3 = np.zeros([1, 1], dtype=int)
print(a3)
```

Hasil yang ditampilkan :

```
[[0]]
```

```
a4 = np.zeros([1, 2], dtype=int)
print(a4)
```

Hasil yang ditampilkan :

```
[[0 0]]
```

```
a5 = np.zeros([2, 1], dtype=int)
print(a5)
```

Hasil yang ditampilkan :

```
[[0]
 [0]]
```

```
a5 = np.zeros([2, 2], dtype=int)
print(a5)
```

Hasil yang ditampilkan :

```
[[0 0]
 [0 0]]
```

2. ones()

fungsi ini merupakan fungsi built-in dari modul numpy, digunakan untuk membuat array / matriks dengan bentuk dan tipe yang telah dispesifikasi dengan parameter fungsi ini, di mana masing masing elemen dari array / matriks yang dihasilkan adalah 1 (satu).

Sintaks

```
np.ones(shape, dtype=None, order='C')
```

Keterangan

- *shape* : *integer atau sequence integer*

- *dtype* : *tipe data output yang dihasilkan (opsional)*

- *order : {'C', 'F'}* (optional), merupakan pengaturan penyimpanan data di memori, apakah akan menggunakan style c atau style fortran

Contoh

Untuk lebih jelasnya tentang operasi dari numpy ones, berikut ini contoh script, buat file dengan nama numpy_ones.py

```
#!/usr/bin/env python
"""
numpy ones sample
for python training at www.jasaplus.com
"""

import numpy as np

a1 = np.ones(1, dtype=int)
print(a1)

print("=" * 50)

a2 = np.ones(2, dtype=int)
print(a2)

print("=" * 50)

a3 = np.ones([1, 1], dtype=int)
print(a3)

print("=" * 50)

a4 = np.ones([1, 2], dtype=int)
print(a4)

print("=" * 50)

a5 = np.ones([2, 1], dtype=int)
print(a5)

print("=" * 50)

a5 = np.ones([2, 2], dtype=int)
print(a5)
```

Jalankan script di atas, berikut ini tampilan yang di print :

```
ringlayer@ringlayer-Inspiron-3442:~/Desktop/TRAINING/PYTHON/source/numpy$ ./numpy_ones.py
[1 1]
=====
[[1]]
=====
[[1 1]]
=====
[[1]
 [1]]
=====
[[1 1]
 [1 1]]
ringlayer@ringlayer-Inspiron-3442:~/Desktop/TRAINING/PYTHON/source/numpy$
```

3. matrix()

fungsi ini merupakan fungsi built-in dari modul numpy, fungsi ini digunakan untuk membuat matriks.

Sintaks

```
np.matrix(data, dtype=None, copy=True)
```

Keterangan

- *data* : merupakan parameter input berupa array atau string

- *dtype* : tipe data output yang dihasilkan (opsional)

Contoh

```
>>> import numpy as np
>>> x = np.matrix( ((1,2), (3, 4)) )
>>> y = np.matrix( ((5,6), (7, 8)) )
>>> print(x)
[[1 2]
 [3 4]]
>>> print(y)
[[5 6]
 [7 8]]
>>> x * y
matrix([[19, 22],
 [43, 50]])
```

Pada contoh di atas, kita memiliki matrix x :

$$\begin{vmatrix} & 1 & 2 \\ & 3 & 4 \end{vmatrix}$$

Selanjutnya kita membuat matrix y :

$$\begin{vmatrix} & 5 & 6 \\ & 7 & 8 \end{vmatrix}$$

Di mana hasil perkalian dari matrix x dan y adalah :

$$\begin{vmatrix} & 19 & 22 \\ & 43 & 50 \end{vmatrix}$$

3.7.2. Numpy untuk Image Processing dan Manipulation

Pada contoh kali ini, kita akan mencoba penggunaan numpy untuk pemrosesan image dan manipulasi image. Selain itu kita akan menggunakan beberapa modul tambahan yang akan kita coba yaitu : scikit-image, matplotlib, pillow dan scipy

3.7.2.1. Persiapan

Sebelum memulai, kita perlu menginstall beberapa modul tambahan, berikut ini adalah beberapa modul yang akan kita install. Install scikit-image, matplotlib, Pillow dan scipy:

```
pip install scikit-image
pip install matplotlib
pip install Pillow
pip install scipy
```

3.7.2.2. Meload Image ke Dalam Numpy Array

Setelah semua modul di atas terinstall, kita bisa memulai untuk meload image. Untuk contoh image kita akan menggunakan image img1.png :



Image di atas merupakan image jpg berukuran width 200px dan height 168px, agar image bisa diproses dengan numpy kita perlu meload image menjadi bentuk numpy array.

Berikut ini adalah contoh meload image ke dalam numpy array :

```
ringlayer@ringlayer-Inspiron-3442:~/Desktop/TRAINING/PYTHON/source/numpy$ python3
Python 3.5.2 (default, Nov 23 2017, 16:37:01)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
>>> from PIL import Image
>>> img = Image.open('img1.png')
>>> array = np.array(img)
>>> print(type(img))
<class 'PIL.PngImagePlugin.PngImageFile'>
>>> print(array.shape)
(168, 200, 3)
```

```
>>> import numpy as np
>>> from PIL import Image
>>> img = Image.open('img1.png')
>>> array = np.array(img)
>>> print(array.shape)
(168, 200, 3)
```

Penjelasan

```
>>> import numpy as np
>>> from PIL import Image
```

Pertama tama kita import dulu modul numpy dan PIL. Selanjutnya kita menggunakan PIL untuk membuka image :

```
>>> img = Image.open('img1.jpg')
```

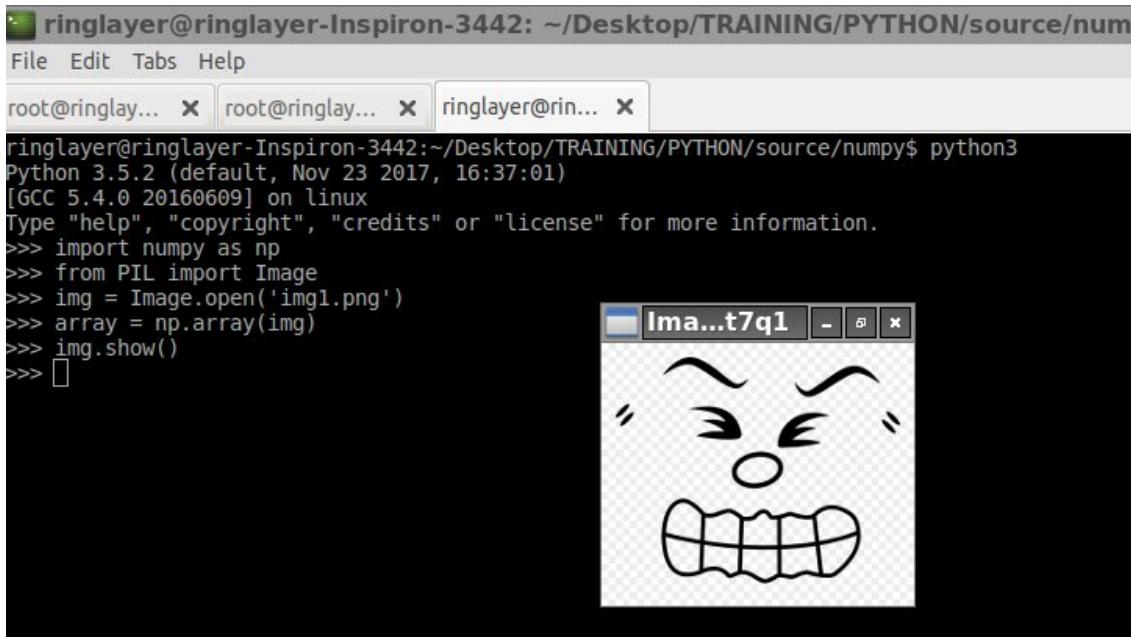
Selanjutnya kita simpan ke dalam bentuk numpy array :

```
>>> array = np.array(img)
```

Hasilnya adalah image tersebut akan disimpan ke dalam bentuk array numpy 3 dimensi :

```
>>> print(array.shape)
(168, 200, 3)
```

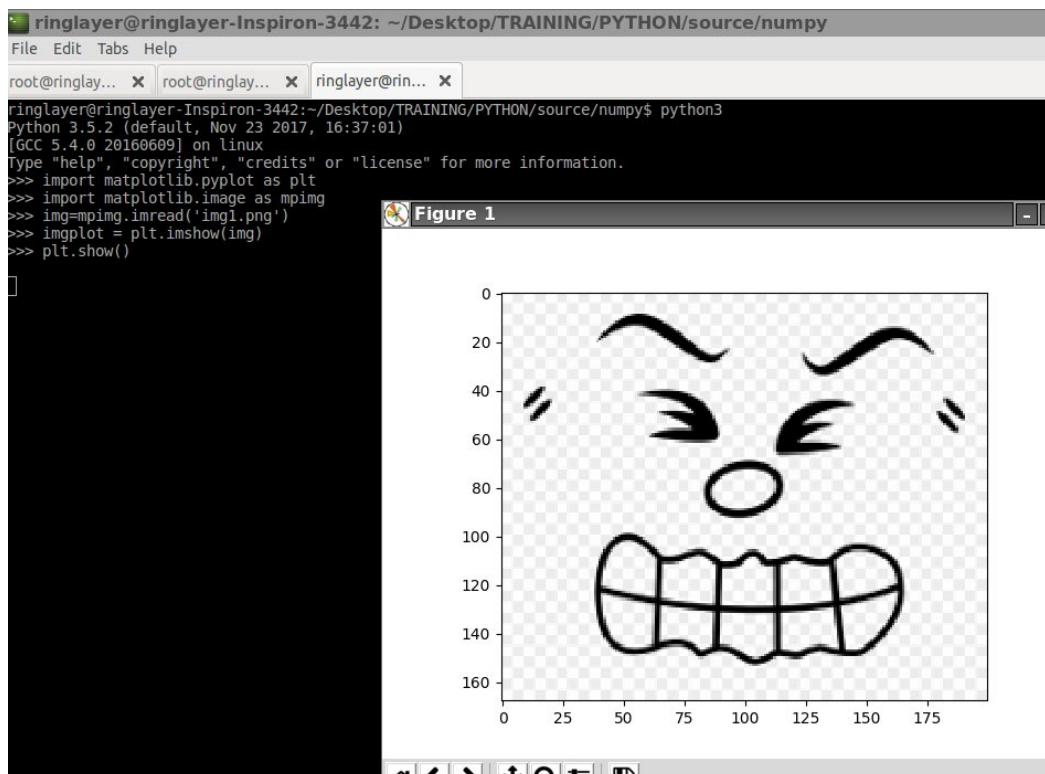
Untuk menampilkan image dengan PIL kita bisa menggunakan method show() :



The screenshot shows a terminal window titled "ringlayer@ringlayer-Inspiron-3442: ~/Desktop/TRAINING/PYTHON/source/numpy". The window has tabs for "File", "Edit", "Tabs", and "Help". The command "python3" is run, followed by importing numpy and PIL, opening "img1.png", creating a numpy array from it, and displaying it with img.show(). A separate window titled "Image" shows a cartoon character's face with a wide-open mouth and visible teeth.

```
ringlayer@ringlayer-Inspiron-3442:~/Desktop/TRAINING/PYTHON/source/numpy$ python3
Python 3.5.2 (default, Nov 23 2017, 16:37:01)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
>>> from PIL import Image
>>> img = Image.open('img1.png')
>>> array = np.array(img)
>>> img.show()
```

Selain menggunakan PIL untuk meload image, kita bisa menggunakan matplotlib. Contoh :



The screenshot shows a terminal window titled "ringlayer@ringlayer-Inspiron-3442: ~/Desktop/TRAINING/PYTHON/source/numpy". The command "python3" is run, followed by importing matplotlib.pyplot and matplotlib.image, reading "img1.png" into img, and displaying it with plt.imshow(). A separate window titled "Figure 1" shows the same cartoon character's face with a wide-open mouth and visible teeth, plotted on a coordinate system with axes ranging from 0 to 160.

```
ringlayer@ringlayer-Inspiron-3442:~/Desktop/TRAINING/PYTHON/source/numpy$ python3
Python 3.5.2 (default, Nov 23 2017, 16:37:01)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import matplotlib.pyplot as plt
>>> import matplotlib.image as mpimg
>>> img=mpimg.imread('img1.png')
>>> imgplot = plt.imshow(img)
>>> plt.show()
```

```
>>> import matplotlib.pyplot as plt
>>> import matplotlib.image as mpimg
>>> img = mpimg.imread('img1.png')
```

```
>>> imgplot = plt.imshow(img)
>>> plt.show()
```

img sudah tersimpan dalam bentuk numpy ndarray seperti bisa kita lihat di bawah ini, akan tetapi nilai elemen warna pada masing masing pixel merupakan float32, agar image mudah diproses kita bisa mengkonversi numpy array image dari matplotlib menjadi numpy array bentuk PIL dengan cara berikut ini :

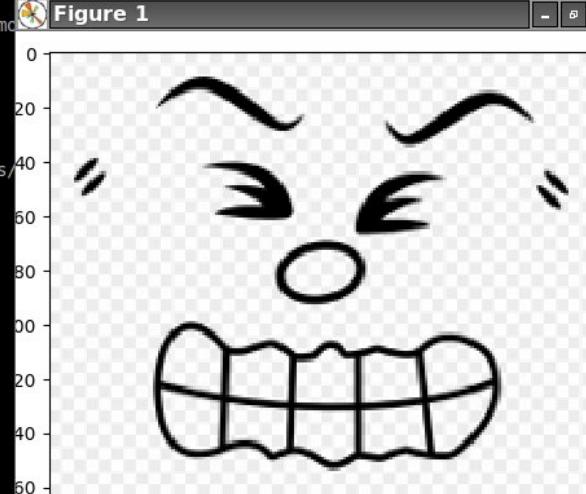
```
formatted = (img * 255 / np.max(img)).astype('uint8')
img2 = Image.fromarray(formatted)
```

```
ringlayer@ringlayer-Inspiron-3442:~/Desktop/TRAINING/PYTHON/source/numpy$ python3
Python 3.5.2 (default, Nov 23 2017, 16:37:01)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>> from PIL import Image
>>> import numpy as np
>>> import matplotlib.pyplot as plt
>>> import matplotlib.image as mpimg
>>> img = mpimg.imread('img1.png')
>>> imgplot = plt.imshow(img)
>>> plt.show()
>>> formatted = (img * 255 / np.max(img)).astype('uint8')
>>> img2 = Image.fromarray(formatted)
>>> img2.show()
>>> 
```



Selain matplotlib, kita bisa juga menggunakan skimage untuk meload image ke dalam bentuk numpy array

```
ringlayer@ringlayer-Inspiron-3442: ~/Desktop/TRAINING/PYTHON/source/numpy
File Edit Tabs Help
root@ringlay... x root@ringlay... x ringlayer@rin... x ringlayer@rin... x
ringlayer@ringlayer-Inspiron-3442:~/Desktop/TRAINING/PYTHON/source/numpy$ python3
Python 3.5.2 (default, Nov 23 2017, 16:37:01)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import skimage
>>> from skimage import io
>>> img = io.imread("img1.png")
>>> print(type(img))
<class 'numpy.ndarray'>
>>> io.imshow(img)
/usr/local/lib/python3.5/dist-packages/matplotlib/axes/
" since 2.2.", cbook.mplDeprecation)
<matplotlib.image.AxesImage object at 0x7f10060c73c8>
>>> io.show()
```



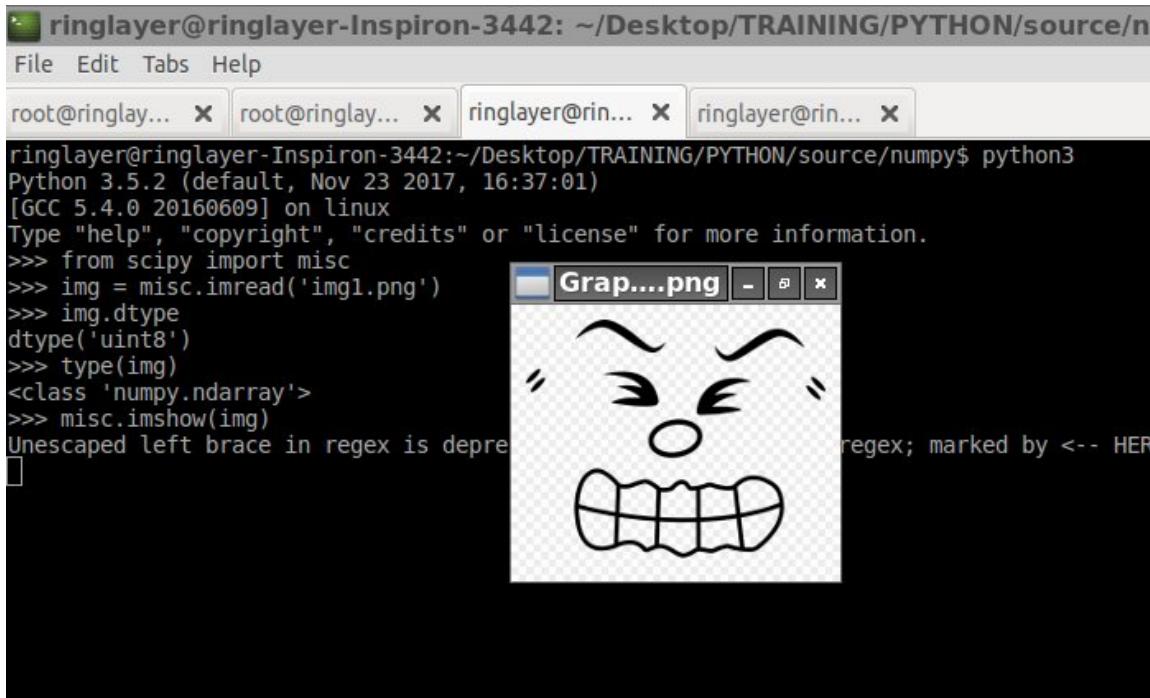
```
>>> import skimage
>>> from skimage import io
>>> img = io.imread("img1.png")
>>> print(type(img))
<class 'numpy.ndarray'>
```

```

>>> io.imshow(img)
/usr/local/lib/python3.5/dist-packages/matplotlib/axes/_base.py:1400: MatplotlibDeprecationWarning: The 'box-forced'
keyword argument is deprecated since 2.2.
" since 2.2.", cbook.mplDeprecation)
<matplotlib.image.AxesImage object at 0x7f10060c73c8>
>>> io.show()

```

Pada dasarnya imshow pada modul skimage akan memanggil imshow pada modul matplotlib. Berikut ini contoh meload image ke dalam bentuk numpy array dan menampilkan image dengan scipy :



The screenshot shows a terminal window titled 'ringlayer@ringlayer-Inspiron-3442: ~/Desktop/TRAINING/PYTHON/source/n'. It contains the following text:

```

File Edit Tabs Help
root@ringlay... x root@ringlay... x ringlayer@rin... x ringlayer@rin... x
ringlayer@ringlayer-Inspiron-3442:~/Desktop/TRAINING/PYTHON/source/numpy$ python3
Python 3.5.2 (default, Nov 23 2017, 16:37:01)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from scipy import misc
>>> img = misc.imread('img1.png')
>>> img.dtype
dtype('uint8')
>>> type(img)
<class 'numpy.ndarray'>
>>> misc.imshow(img)
Unescaped left brace in regex is depre

```

To the right of the terminal window, there is a small image window titled 'Grap....png' showing a cartoon character's face with a wide-open mouth.

```

>>> from scipy import misc
>>> img = misc.imread('img1.png')
>>> img.dtype
dtype('uint8')
>>> type(img)
<class 'numpy.ndarray'>
>>> misc.imshow(img)

```

Semua modul di atas, akan meload image ke dalam numpy array dengan format susunan RGB. Pada image png biasanya terdapat 1 tambahan channel , pada contoh di atas tidak tersimpan data channel transparansi sehingga shape :

(168, 200, 3)

jika menyimpan nilai rgb channel maka array 3 dimensinya adalah

(168, 200, 4)

Pada opencv secara default saat meload image, image akan diload ke dalam numpy array, di mana elemen-elemen di dalam array susunan nilai warnanya adalah BGR

3.7.2.3. Memahami Numpy Image Array

Untuk memahami numpy image array, kita akan menggunakan contoh image berukuran width 10 pixel dan height 10 pixel dengan nama image 10px.png

Berikut ini image jika dizoom dengan image editor :



Image 10px.png ini berukuran sangat kecil sehingga perlu dizoom agar bisa melihat masing masing pixelnya. Untuk image editor yang kita gunakan untuk zooming di sini adalah pinta, yang bisa didownload gratis di <https://pinta-project.com/pintaproject/pinta/releases>

Selanjutnya, Load image 10px.png tadi menjadi numpy array :

```

>>> import numpy as np
>>> from PIL import Image
>>> img = Image.open('10px.png')
>>> arr = np.array(img)
>>> print(arr)
[[[ 0  0  0 255]
 [255 127 127 255]
 [ 0  0  0 255]
 [160 160 160 255]
 [160 160 160 255]
 [ 0  0  0 255]
 [255 127 127 255]
 [ 76 255  0 255]
 [ 76 255  0 255]
 [ 76 255  0 255]]

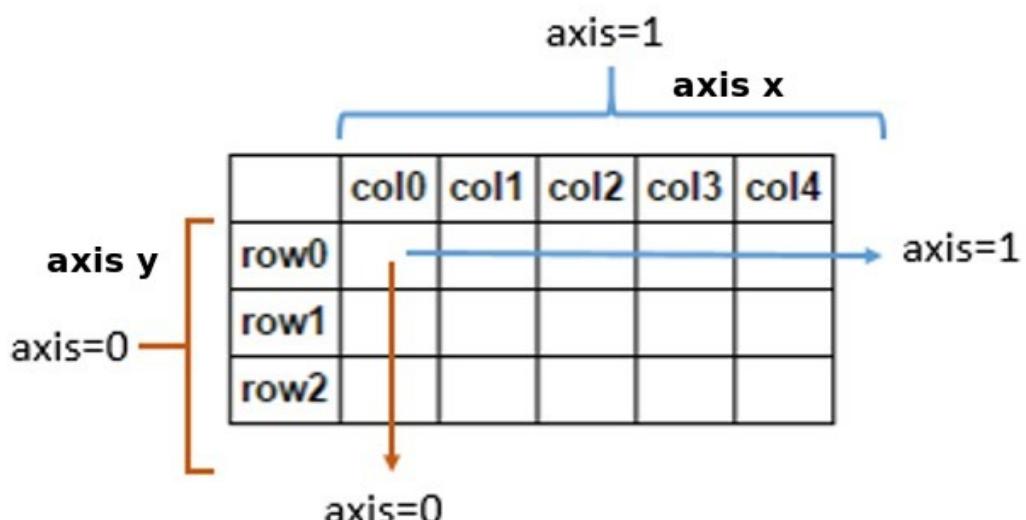
[[255 255 255 255]
 [255 255 255 255]

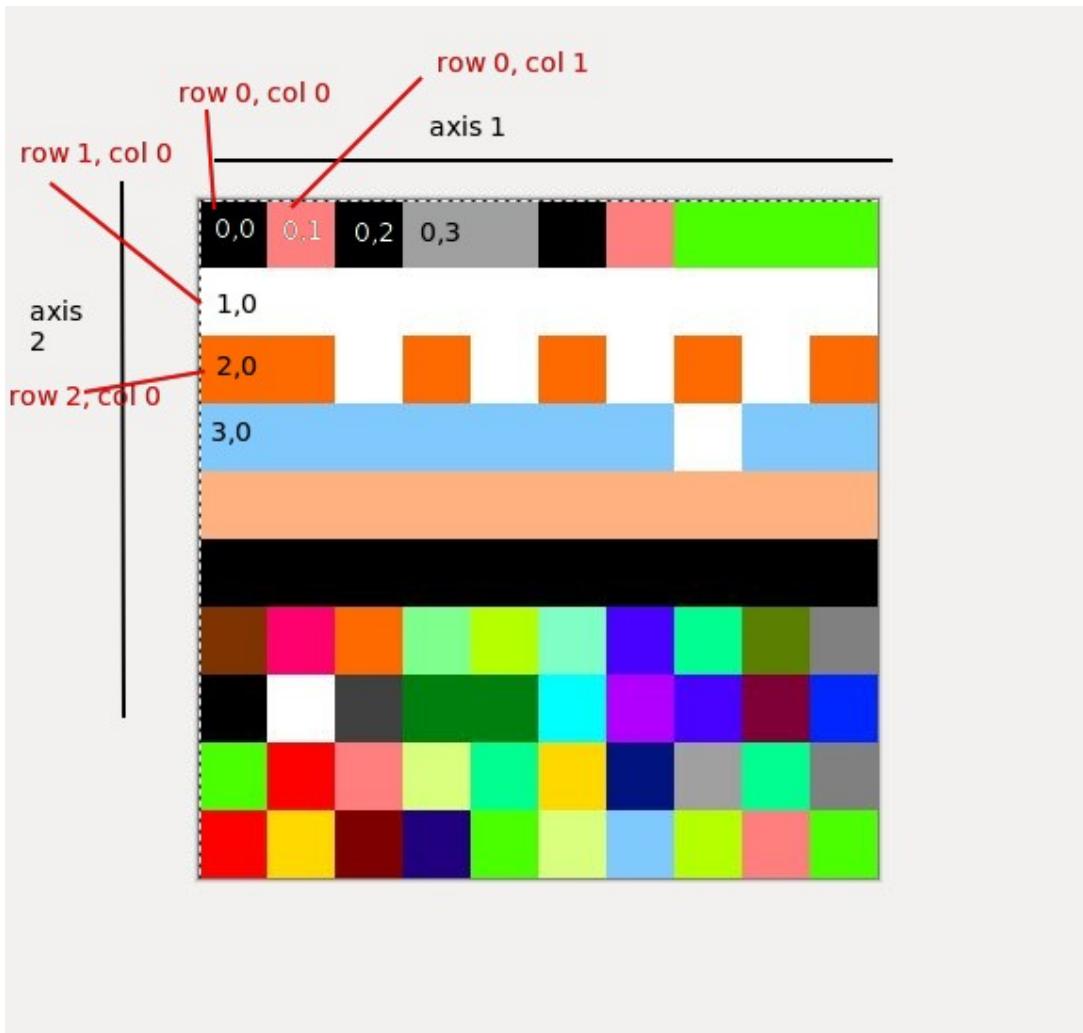
>>> arr.shape
(10, 10, 4)

```

Penjelasan

Untuk lebih jelasnya perhatikan gambar di bawah ini :





Pada gambar di atas, masing masing kotak mewakili 1 pixel. Pixel paling pojok kiri atas merupakan row 0, column 0. Axis 1 adalah untuk column dan axis 2 adalah untuk row. Total row adalah 10 dan total column adalah 10.

Saat image 10px.png di atas diload menjadi numpy array maka akan dijadikan numpy array 3 dimensi, di mana shape (berupa tuple) dari numpy array tersebut adalah (10, 10, 4)

di mana :

```
>>> print(arr[0])
[[ 0  0  0 255]
 [255 127 127 255]
 [ 0  0  0 255]
 [160 160 160 255]
 [160 160 160 255]
 [ 0  0  0 255]
 [255 127 127 255]
 [ 76 255  0 255]
 [ 76 255  0 255]
 [ 76 255  0 255]]
```

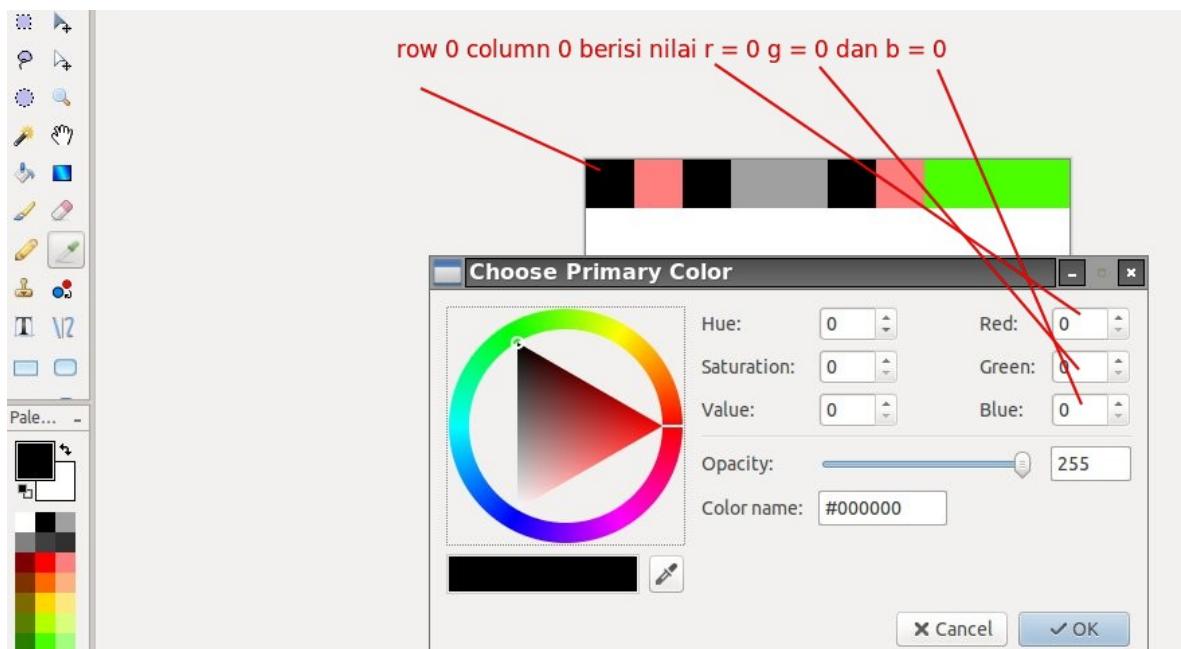
kita bisa melihat, pada numpy array di atas masing masing elemen terdiri dari elemen yang merupakan suatu list.

Di dalam list ini terdapat 10 list yang masing masing mewakili 1 pixel,

```
>>> print(arr[0])
[[ 0  0  0 255]
 [255 127 127 255]
 [ 0  0  0 255]
 [160 160 160 255]
 [160 160 160 255]
 [ 0  0  0 255]
 [255 127 127 255]
 [ 76 255  0 255]
 [ 76 255  0 255]
 [ 76 255  0 255]]
```

Pada list ke 0 tersimpan data untuk 10 pixel (10 kolom) pada baris ke 0. di mana

[0 0 0 255] = row ke 0, column ke 0



[255 127 127 255] = row ke 0, column ke 1

[0 0 0 255] = row ke 0, column ke 2

[160 160 160 255] = row ke 0, column ke 3

[160 160 160 255] = row ke 0, column ke 4

Untuk mengakses 1 pixel kita menggunakan prinsip berdasarkan gambar di atas, misal :

```
>>> print(arr[0][0])
[ 0  0  0 255]
```

terlihat pada row 0 dan baris 0 terdapat list dengan 4 item. Ketiga item pertama merupakan nilai RGB dan nilai keempat yaitu 255 merupakan data channel transparansi (khusus pada image png biasanya terdapat 1 tambahan data ini).

Pada numpy kita bisa mengakses langsung elemen ini dengan pemberian parameter langsung pada numpy image array dengan pola :

```
numpy_image_array[row_number, column_number]
```

Contoh:

```
>>> print(arr[0,0])  
[ 0  0  0 255]
```

list di atas itemnya merupakan susunan nilai rgb (rgb adalah channel warna red, green dan blue) pada pixel dengan row 0 dan column 0 :

r = 0
g = 0
b = 0
channel = 255

```
>>> print(arr[0,1])  
[255 127 127 255]
```

adalah sama dengan

```
>>> print(arr[0][1])  
[255 127 127 255]
```

angka di atas adalah nilai rgb pada pixel dengan row 0 dan column 1 :

r = 255
g = 127
b = 127
channel = 255

Jika ingin menghilangkan data channel untuk transparansi pada image png di atas, kita menggunakan trik ini :

```
img_no_alpha = arr[:, :, :3]
```

Dengan slice di atas, maka akan membuang semua item dengan nomor indeks ke 3 pada dimensi ketiga. Berikut ini contoh di python shell :

```

>>> import numpy as np
>>> from PIL import Image
>>> img = Image.open('10px.png')
>>> arr = np.array(img)
>>> img_no_alpha = arr[:, :, :3]
>>> print(img_no_alpha)
[[[ 0  0  0]
 [255 127 127]
 [ 0  0  0]
 [160 160 160]
 [160 160 160]
 [ 0  0  0]
 [255 127 127]
 [ 76 255  0]
 [ 76 255  0]
 [ 76 255  0]]

[[255 255 255]
 [255 255 255]]

```

Dengan trik di atas, maka numpy array tidak akan menyimpan nilai channel transparansi dari image png. Untuk menyimpan numpy array menjadi image, kita bisa menggunakan method fromarray dari PIL:

```
new_im = Image.fromarray(img_no_alpha)
```

contoh pada python shell:

```

>>> import numpy as np
>>> from PIL import Image
>>> img = Image.open('10px.png')
>>> arr = np.array(img)
>>> img_no_alpha = arr[:, :, :3]
>>> new_im = Image.fromarray(img_no_alpha)
>>> new_im.save("png_no_alpha_channel.png")

```

Dengan trik di atas, kita menyimpan 1 image sebagai png tanpa data channel untuk transparansi.

3.7.2.4. Crop Image

Pada contoh selanjutnya kita akan mencoba manipulasi image dengan numpy. Contoh pertama kita akan melakukan cropping image. Pola untuk cropping image numpy array dengan teknik slicing :

cropped_img_numpy_array = img[y1:y2, x1:x2] , di mana x1 dan y1 adalah top left corner. x2 dan y2 adalah bottom right. Misal kita melakukan crop image di atas dengan :

```
cropimg = img_no_alpha[1:8, 1:8]
```

maka kita akan mengambil, mulai dari baris ke 1 hingga sebelum baris ke 8, dan mulai kolom ke 1 hingga sebelum kolom ke 8, sisa pixel lainnya akan dibuang.

Berikut ini contoh di python shell :

```

>>> import numpy as np
>>> from PIL import Image
>>> img = Image.open('10px.png')

```

```

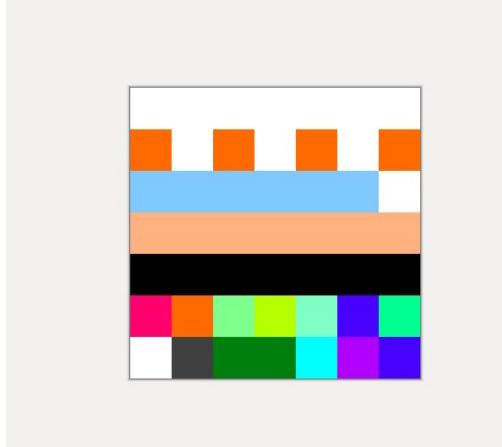
>>> arr = np.array(img)
>>> img_no_alpha = arr[:, :, :3]
>>> cropimg = img_no_alpha[1:8, 1:8]
>>> cr = Image.fromarray(cropimg)
>>> cr.save("cropped.png")

```

Berikut ini operasi crop yang akan dilakukan dari image asli :



Berikut ini hasil crop :



```
cropimg = img_no_alpha[1:8, 1:8]
```

akan menyebabkan `cropimg` menyimpan mulai dari row ke 1 hingga row ke 7 (slice), dan menyimpan mulai dari kolom 1 hingga kolom 7

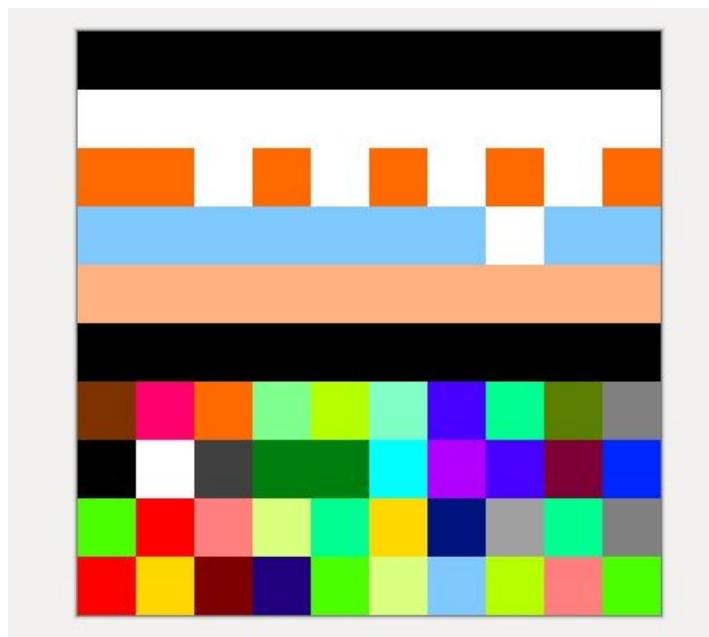
3.7.2.5. Manipulasi Pixel

Kita bisa melakukan manipulasi image dengan numpy array. Misal pada image asli `10px.png`, kita akan manipulasi pada seluruh pixel di baris pertama akan kita jadikan hitam dan hasil numpy arraynya kita simpan ke file baru.

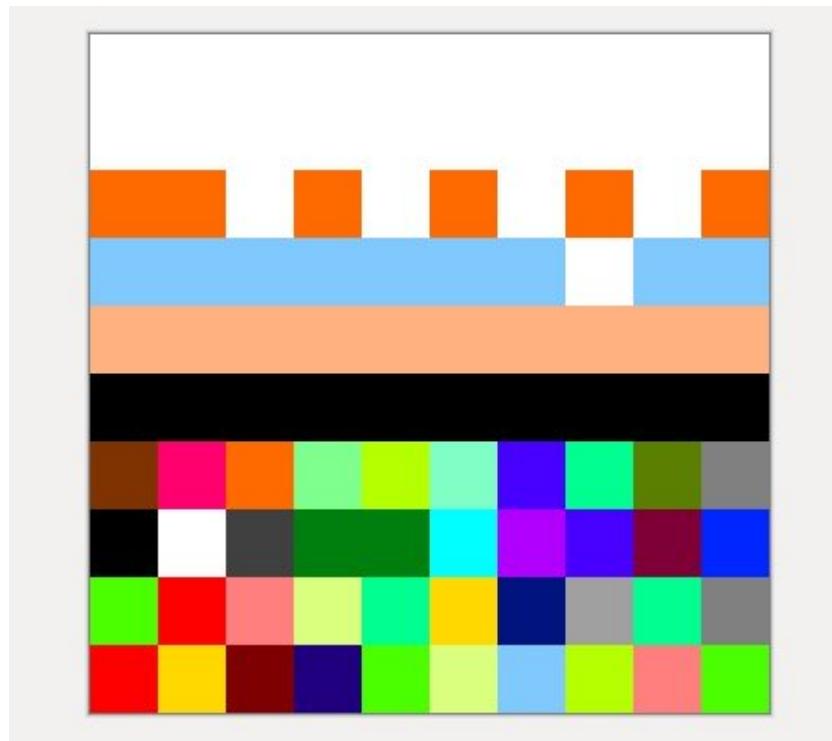
Contoh berikut ini akan mengganti pixel pada baris pertama (seluruh kolom) menjadi hitam (`rgb = 0 0 0`) kemudian kita simpan sebagai image dengan `modif.png`, yang pertama kita lakukan dengan looping. Kemudian kita mengubah seluruh pixel pada baris pertama menjadi putih (`rgb = 255 255 255`) dengan teknik slicing.

```
>>> import numpy as np
>>> from PIL import Image
>>> img = Image.open('10px.png')
>>> arr = np.array(img)
>>> img_no_alpha = arr[:, :, :3]
>>> for i in range(0, 10):
...     img_no_alpha[0, i] = [0, 0, 0]
...
>>>
>>> modif = Image.fromarray(img_no_alpha)
>>> modif.save("modif.png")
>>> img_no_alpha[0, 0:10] = [255, 255, 255]
>>> modif2 = Image.fromarray(img_no_alpha)
>>> modif2.save("modif2.png")
```

Berikut ini file modif.png :



Berikut ini file modif2.png



3.7.2.6. Mengganti seluruh pixel pada image

Berikut ini contoh menghitamkan seluruh pixel pada image :

```
>>> import numpy as np
>>> from PIL import Image
>>>
>>>
>>> img = Image.open('10px.png')
>>> arr = np.array(img)
>>> img_no_alpha = arr[:, :, :3]
>>>
>>> img_no_alpha[:, :] = [0, 0, 0]
>>> black = Image.fromarray(img_no_alpha)
>>> black.save("black.png")
```

Rutin ini `img_no_alpha[:, :] = [0, 0, 0]` , digunakan untuk mengganti seluruh nilai pada pixel menjadi rgb 0,0,0 (hitam)

3.7.2.7. Looping untuk Mengakses Seluruh Pixel Image

Untuk mengakses satu persatu pixel pada seluruh image, kita bisa menggunakan looping atau bisa juga dengan slice. Buat aplikasi baru dengan nama img7.py :

```
#!/usr/bin/env python3
"""
img7.py
demo for python course
at www.jasaplus.com
"""

import numpy as np
from PIL import Image

def _loop_through_all_pixels(img, width, height):
    x = 0
    while x < height:
        y = 0
        while y < width:
            print(img[x, y])
            y += 1
            if y == (width):
                print(" " * 20)
        x += 1

img = Image.open('10px.png')
arr = np.array(img)
img_no_alpha = arr[:, :, 3]
_loop_through_all_pixels(img_no_alpha, 10, 10)
```

Hasil dari looping akan menampilkan nilai rgb pada masing masing pixel :

```
ringlayer@ringlayer-Inspiron-3442:~/Desktop/TRAINING/PYTHON/source/numpy$ ./img7.py
[[0 0 0]
 [255 127 127]
 [0 0 0]
 [160 160 160]
 [160 160 160]
 [0 0 0]
 [255 127 127]
 [76 255 0]
 [76 255 0]
 [76 255 0]

 [255 255 255]
 [255 255 255]
 [255 255 255]
 [255 255 255]
 [255 255 255]
 [255 255 255]
 [255 255 255]
 [255 255 255]
 [255 255 255]
 [255 255 255]

 [255 106 0]
 [255 106 0]
 [255 255 255]
 [255 106 0]
 [255 255 255]
 [255 106 0]
 [255 255 255]
 [255 106 0]
 [255 255 255]
 [255 106 0]

 [127 201 255]
```

3.7.2.8. Konversi Image ke Grayscale

Pada contoh selanjutnya kita akan mengubah image menjadi grayscale dengan numpy. Image yang akan diproses adalah cat.jpg :



Buat aplikasi baru dengan nama img8.py:

```
#!/usr/bin/env python3
"""
img8.py
demo for python course
at www.jasaplus.com
"""

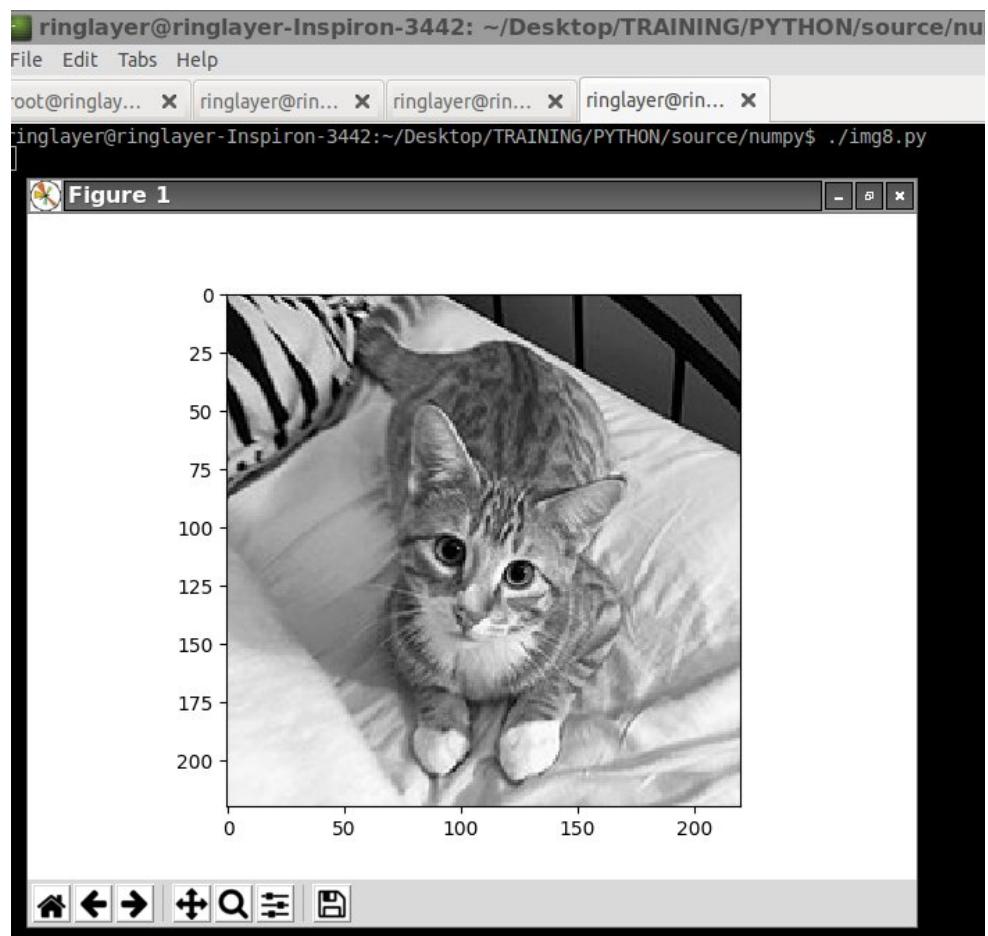
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

def rgb2gray(rgb):
    return np.dot(rgb[...,:3], [0.299, 0.587, 0.144])

img = mpimg.imread('cat.jpg')
gray = rgb2gray(img)
plt.imshow(gray, cmap = plt.get_cmap('gray'))
plt.show()

formatted = (gray * 255 / np.max(gray)).astype('uint8')
cr = Image.fromarray(formatted)
cr.save("gray.jpg")
```

Hasil dari menjalankan script di atas :



aplikasi di atas akan menghasilkan file gray.jpg :



Pada script di atas :

```
def rgb2gray(rgb):
    return np.dot(rgb[...,:3], [0.299, 0.587, 0.144])
```

kita menggunakan np.dot untuk konversi ke grayscale dengan rumus nilai rgb dikalikan konstan :

```
val1 = r * 0.299
val2 = r * 0.587
val3 = r * 0.144
```

Untuk lebih jelasnya bisa dibaca di wikipedia untuk konversi 3 channel warna rgb menjadi 1 channel warna grayscale :

https://en.wikipedia.org/wiki/Grayscale#Converting_color_to_grayscale

Untuk materi lanjutan dengan numpy akan dibahas dalam kursus selanjutnya “Python for Machine Learning”

3.7.2.9. Rotate Image 180 Derajat

Untuk contoh rotate image 180 derajat siapkan script python baru dengan nama img9.py :

```
#!/usr/bin/env python3
"""
img9.py
demo for python course
at www.jasaplus.com
"""

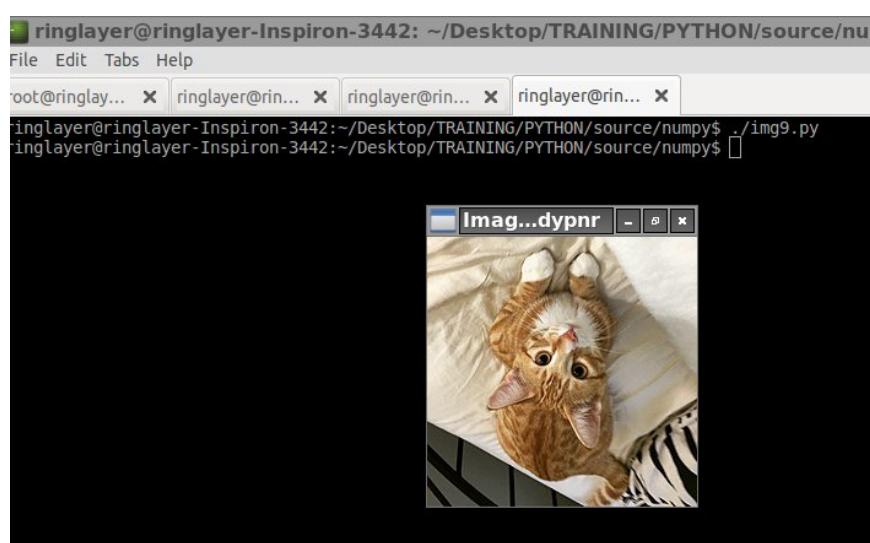
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

img = mpimg.imread('cat.jpg')
img2 = np.rot90(img)
img_final = Image.fromarray(img2)

img3 = np.rot90(img_final)
img_final = Image.fromarray(img3)
img_final.show()
```

Script di atas akan melakukan 2 kali rotasi sebanyak 90 derajat, sehingga total rotasi adalah 180 derajat.

Berikut ini hasil script di atas :



3.8. PANDAS

Pandas adalah library python yang digunakan untuk analisis data. Library ini cukup terkenal penggunaanya pada bidang data science.

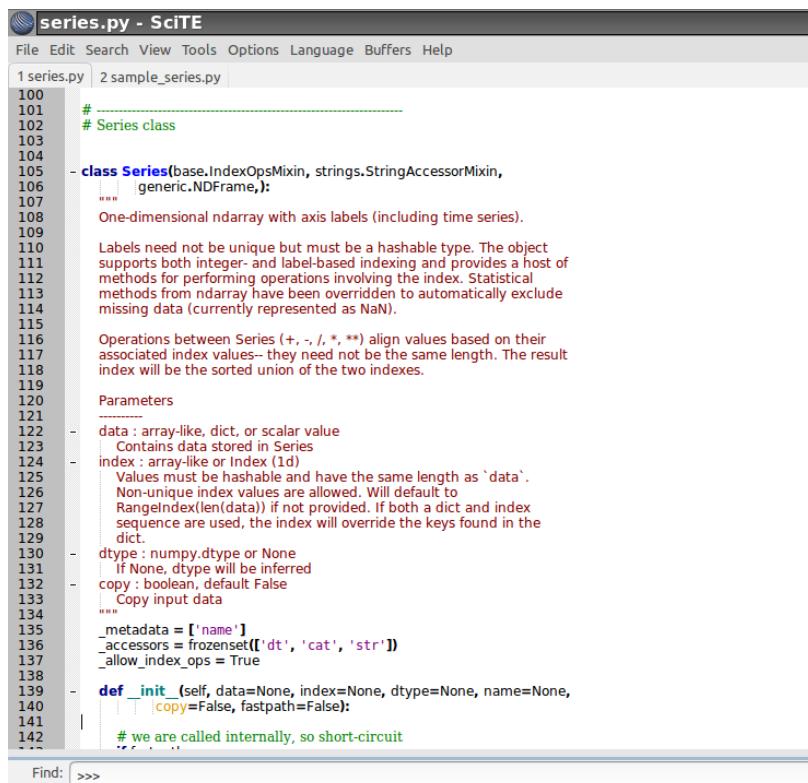
3.8.1. Series

Series adalah array 1 dimensi yang bisa digunakan untuk menyimpan data dengan jenis apapun. Untuk membuat series, kita bisa menggunakan class Series pada modul panda, di mana kita membuat objek instance dari class ini.

```
>>> import pandas as pd
>>> import numpy as np
>>> data = np.array([1,2,3])
>>> ser = pd.Series(data)
>>> print(type(ser))
<class 'pandas.core.series.Series'>
>>> print(isinstance(ser, pd.Series))
True
>>> 
```

Class Series ini bisa dilihat pada core/Series.py pada modul pandas :

```
ringlayer@ringlayer-Inspiron-3442:/usr/local/lib/python3.5/dist-packages/pandas$ grep -R "class Series(" * core/series.py: class Series(base.IndexOpsMixin, strings.StringAccessorMixin,
ringlayer@ringlayer-Inspiron-3442:/usr/local/lib/python3.5/dist-packages/pandas$ 
```



The screenshot shows the SciTE code editor with the file 'series.py' open. The code defines the Series class, which inherits from base.IndexOpsMixin and strings.StringAccessorMixin. The class is described as a one-dimensional ndarray with axis labels (including time series). It supports both integer- and label-based indexing and provides a host of methods for performing operations involving the index. Statistical methods from ndarray have been overridden to automatically exclude missing data (currently represented as NaN). Operations between Series (+, -, /, *, **) align values based on their associated index values— they need not be the same length. The result index will be the sorted union of the two indexes. The class has parameters: data (array-like, dict, or scalar value), index (array-like or Index 1d), dtype (numpy.dtype or None), and copy (boolean, default False). The __init__ method initializes the object with metadata, accessors, and allow_index_ops. A note at the bottom indicates that the class is called internally, so short-circuiting is used.

```
series.py - SciTE
File Edit Search View Tools Options Language Buffers Help
1 series.py | 2 sample_series.py |
100
101 # -----
102 # Series class
103
104
105 - class Series(base.IndexOpsMixin, strings.StringAccessorMixin,
106     generic.NDFrame,):
107     """
108     One-dimensional ndarray with axis labels (including time series).
109
110     Labels need not be unique but must be a hashable type. The object
111     supports both integer- and label-based indexing and provides a host of
112     methods for performing operations involving the index. Statistical
113     methods from ndarray have been overridden to automatically exclude
114     missing data (currently represented as NaN).
115
116     Operations between Series (+, -, /, *, **) align values based on their
117     associated index values-- they need not be the same length. The result
118     index will be the sorted union of the two indexes.
119
120     Parameters
121     -----
122     - data : array-like, dict, or scalar value
123         Contains data stored in Series
124     - index : array-like or Index (1d)
125         Values must be hashable and have the same length as `data`.
126         Non-unique index values are allowed. Will default to
127         RangeIndex(len(data)) if not provided. If both a dict and index
128         sequence are used, the index will override the keys found in the
129         dict.
130     - dtype : numpy.dtype or None
131         If None, dtype will be inferred
132     - copy : boolean, default False
133         Copy input data
134         """
135     _metadata = ['name']
136     _accessors = frozenset(['dt', 'cat', 'str'])
137     _allow_index_ops = True
138
139 -     def __init__(self, data=None, index=None, dtype=None, name=None,
140                 | copy=False, fastpath=False):
141
142         # we are called internally, so short-circuit
```

Berikut ini contoh pembuatan Series pada python shell :

```
>>> import numpy as np
>>> import pandas as pd
>>> data = np.array([1,2,3])
>>> seri1 = pd.Series(data)
>>> print(seri1)
0    1
1    2
2    3
dtype: int64
>>> data2 = [1,2,3]
>>> seri2 = pd.Series(data2)
>>> print(seri2)
0    1
1    2
2    3
```

Penjelasan

```
>>> data = np.array([1,2,3])
>>> seri1 = pd.Series(data)
>>> print(seri1)
```

Pada contoh di atas, kita membuat Series dari numpy array.

```
>>> data2 = [1,2,3]
>>> seri2 = pd.Series(data2)
>>> print(seri2)
```

Pada contoh di atas kita membuat Series dari list

3.8.2. Data Frame

Data frame adalah struktur data tabular yang berisi array 2 dimensi, yang ukuranya bisa berubah ubah, dengan 2 label berupa baris dan kolom. Data frame bisa kita anggap sebagai tabel berisi data.

Untuk membuat data frame pada panda, kita bisa memberikan inputan berupa numpy array, dictionary, series atau data frame lain.

Untuk membuat data frame, kita membuat instance dari class DataFrame. Berikut ini contoh pembuatan Data Frame. Buat script baru dengan nama sample_dataframe.py :

```
#!/usr/bin/env python3
"""
sample_dataframe.py
panda dataframe sample
for training material at www.jasaplus.com
"""

import numpy as np
import pandas as pd

"""
input type : dictionary
"""
print("tdictionary")
```

```

dict1 = {
    'id' : ['0', '1', '2'],
    'product' : ["Servo Motor", "DC Motor", "Muscle Wire"],
    'stock' : [50, 200, 20]
}
df2 = pd.DataFrame(dict1)
print(df2)

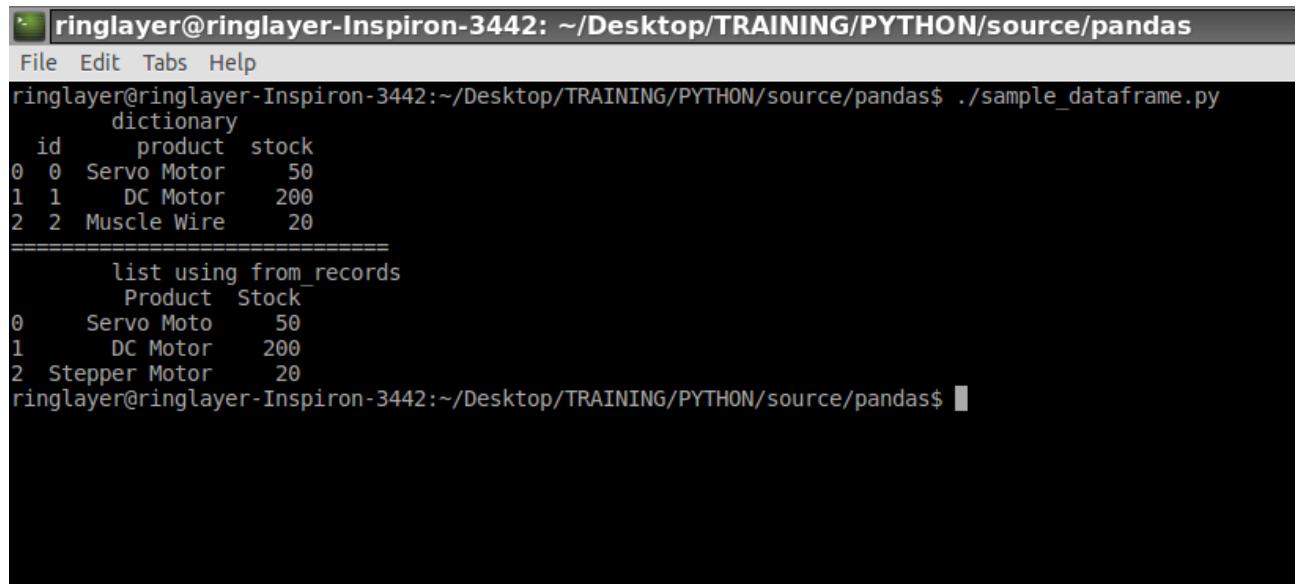
print("==" * 30)

"""
input type : list
"""
print("tlist using from_records")

prods = [('Servo Moto', 50),
          ('DC Motor', 200),
          ('Stepper Motor', 20)]
labels = ['Product', 'Stock']
df = pd.DataFrame.from_records(prods, columns=labels)
print(df)

```

Jalankan script di atas, berikut ini hasil yang ditampilkan :



```

ringlayer@ringlayer-Inspiron-3442: ~/Desktop/TRAINING/PYTHON/source/pandas
File Edit Tabs Help
ringlayer@ringlayer-Inspiron-3442:~/Desktop/TRAINING/PYTHON/source/pandas$ ./sample_dataframe.py
      dictionary
      id      product  stock
0  0  Servo Motor     50
1  1      DC Motor    200
2  2  Muscle Wire     20
=====
      list using from_records
      Product  Stock
0  Servo Moto     50
1      DC Motor    200
2  Stepper Motor     20
ringlayer@ringlayer-Inspiron-3442:~/Desktop/TRAINING/PYTHON/source/pandas$ 

```

Penjelasan

```

dict1 = {
    'id' : ['0', '1', '2'],
    'product' : ["Servo Motor", "DC Motor", "Muscle Wire"],
    'stock' : [50, 200, 20]
}
df2 = pd.DataFrame(dict1)
print(df2)

print("==" * 30)

"""
input type : list
"""
print("tlist using from_records")

prods = [('Servo Moto', 50),
          ('DC Motor', 200),
          ('Stepper Motor', 20)]
labels = ['Product', 'Stock']
df = pd.DataFrame.from_records(prods, columns=labels)
print(df)

```

Pada potongan kode di atas, kita menyiapkan dictionary untuk membuat DataFrame dari dictionary.

```

prods = [('Servo Moto', 50),
          ('DC Motor', 200),

```

```
('Stepper Motor', 20)]  
labels = ['Product', 'Stock']  
df = pd.DataFrame.from_records(prods, columns=labels)
```

Pada contoh di atas, kita membuat DataFrame dari list di mana masing masing elemen di dalam list merupakan tuple.

from_records merupakan method di dalam class DataFrame yang digunakan untuk membuat DataFrame dari record berupa ndarray, list dari tuple atau dictionary.

3.8.3. Membaca CSV dan Meload CSV ke Data Frame

Untuk membaca dan meload file csv ke DataFrame kita menggunakan method read_csv.
Untuk lebih jelasnya berikut ini kita memiliki sample file csv dengan nama sample.csv :

```
name,phone,status  
anton,+6281210249020,good  
adam,+6281210249121,bad  
linda,+628122222222,good
```

Pastikan csv memiliki label pada baris pertama agar bisa diparsing dengan benar.

Berikut ini contoh meload data csv di atas dengan read_csv dari konsol python :

```
>>> import pandas as pd  
>>> df = pd.read_csv("sample.csv")  
>>> print(df)  
      name    phone  status  
0  anton  6281210249020   good  
1   adam  6281210249121    bad  
2  linda  628122222222   good  
>>>
```

```
ringlayer@ringlayer-Inspiron-3442:~/Downloads$ python3  
Python 3.5.2 (default, Nov 23 2017, 16:37:01)  
[GCC 5.4.0 20160609] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> import pandas as pd  
>>> df = pd.read_csv("sample.csv")  
>>> print(df)  
      name    phone  status  
0  anton  6281210249020   good  
1   adam  6281210249121    bad  
2  linda  628122222222   good  
>>> █
```

method read_csv juga bisa digunakan untuk membaca csv dari url.

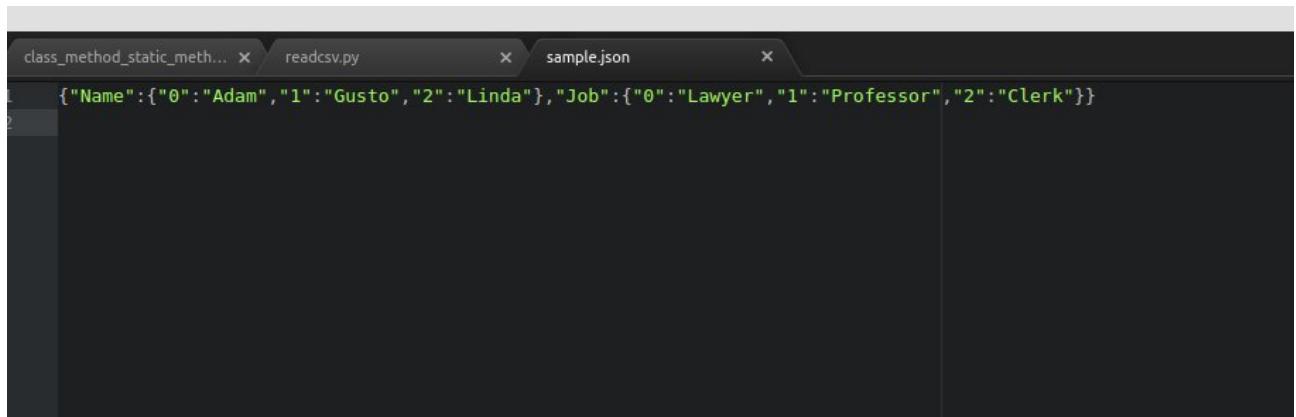
3.8.4. Membaca Data dari JSON

Untuk membaca data dari json, kita bisa menggunakan method read_json.

Berikut ini contoh pembacaan json dan konversi dataframe ke json.

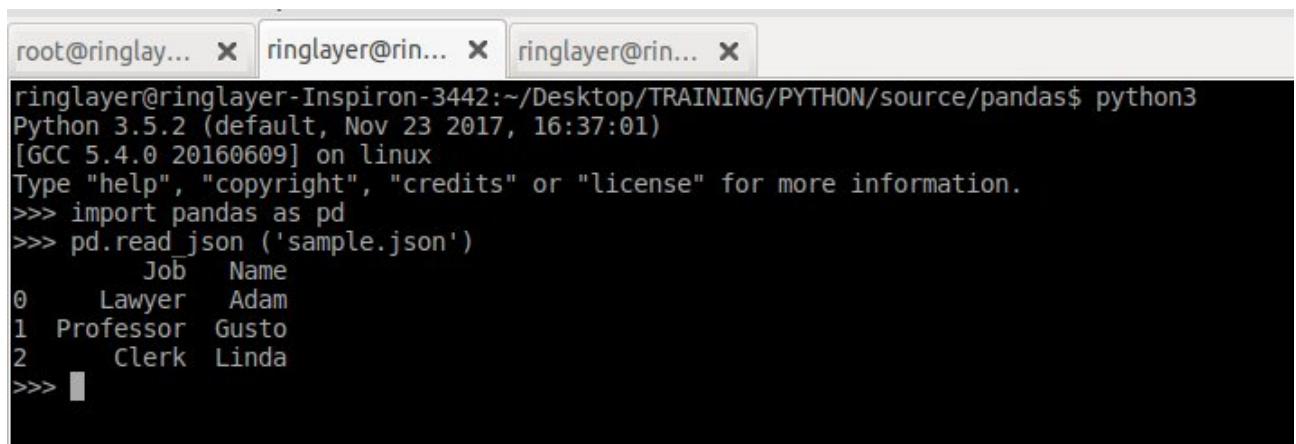
Pertama tama siapkan file json dengan nama : sample.json

```
{"Name":  
  {"0":"Adam","1":"Gusto","2":"Linda"}, "Job": {"0":"Lawyer", "1":"Professor", "2":"Clerk"}}
```



Contoh pembacaan json ke DataFrame dari python console :

```
>>> import pandas as pd  
>>> pd.read_json ('sample.json')  
      Job   Name  
0    Lawyer Adam  
1  Professor Gusto  
2     Clerk Linda
```



```
root@ringlay... x ringlayer@rin... x ringlayer@rin... x  
ringlayer@ringlayer-Inspiron-3442:~/Desktop/TRAINING/PYTHON/source/pandas$ python3  
Python 3.5.2 (default, Nov 23 2017, 16:37:01)  
[GCC 5.4.0 20160609] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> import pandas as pd  
>>> pd.read_json ('sample.json')  
      Job   Name  
0    Lawyer Adam  
1  Professor Gusto  
2     Clerk Linda  
>>> █
```

Selain dari file, bisa juga data json dari url.

3.8.5. Seleksi Data pada Data Frame

Untuk seleksi data baris dan kolom pada panda, kita bisa menggunakan loc, iloc & ix.

3.8.5.1. loc

loc (locate) merupakan object yang digunakan untuk seleksi data pada DataFrame, loc ini digunakan untuk seleksi data berdasarkan label.

```
>>> print(type(df.loc))
<class 'pandas.core.indexing._LocIndexer'>
```

Loc merupakan alias dari class _LocIndexer.

Berikut ini format dari loc :

loc(parameter_nomor_baris, parameter_berupa_label)

Parameter pertama pada loc merupakan nomor baris, parameter kedua pada loc merupakan label pada kolom . Pemberian tanda : “:” artinya melakukan seleksi pada seluruh data bisa pada baris atau pada kolom.

Berikut ini contoh penggunaan loc , kembali lagi ke contoh read_csv tadi, berikut ini contoh loc pada python shell :

```
>>> import pandas as pd
>>> df = pd.read_csv("sample.csv")
>>> loc1 = (df.loc[:, 'phone'])
>>> print(loc1)
0    6281210249020
1    6281210249121
2    628122222222
Name: phone, dtype: int64
>>> print(df.loc[:, 'phone'])
0    6281210249020
1    6281210249121
2    628122222222
Name: phone, dtype: int64
>>> print(df.loc[:, 'name'])
0    anton
1    adam
2    linda
Name: name, dtype: object
>>>
```

```
ringlayer@ringlayer-Inspiron-3442:~/Desktop/TRAINING/PYTHON/source/pandas$ python3
Python 3.5.2 (default, Nov 23 2017, 16:37:01)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import pandas as pd
>>> df = pd.read_csv("sample.csv")
>>> loc1 = (df.loc[:, 'phone'])
>>> print(loc1)
0    6281210249020
1    6281210249121
2    6281222222222
Name: phone, dtype: int64
>>> print(df.loc[:, 'phone'])
0    6281210249020
1    6281210249121
2    6281222222222
Name: phone, dtype: int64
>>> print(df.loc[:, 'name'])
0    anton
1    adam
2    linda
Name: name, dtype: object
>>> █
```

Penjelasan

```
>>> loc1 = (df.loc[:, 'phone'])
```

Parameter pertama pada loc di atas adalah seleksi pada seluruh baris, parameter kedua merupakan label phone. Sehingga artinya loc akan menyeleksi seluruh baris pada kolom “phone”

```
>>> print(df.loc[:, 'name'])
```

Parameter pertama pada loc di atas adalah seleksi pada seluruh baris, parameter kedua merupakan label name. Sehingga artinya loc akan menyeleksi seluruh baris pada kolom “name”

Contoh selanjutnya :

```
>>> print(df.loc[:])
   name    phone status
0  anton  6281210249020  good
1  adam   6281210249121  bad
2  linda  6281222222222  good
>>> print(df.loc[:, :])
   name    phone status
0  anton  6281210249020  good
1  adam   6281210249121  bad
2  linda  6281222222222  good
```

```
>>> print(df.loc[:])
      name      phone  status
0  anton  6281210249020   good
1  adam   6281210249121    bad
2 linda  6281222222222   good
>>> print(df.loc[:,:])
      name      phone  status
0  anton  6281210249020   good
1  adam   6281210249121    bad
2 linda  6281222222222   good
>>> █
```

```
>>> print(df.loc[:])
```

Pada loc di atas, kita hanya memberikan parameter pertama yang merupakan parameter baris, parameter kedua tidak diberikan. Parameter kedua secara default jika tidak diberikan akan melakukan seleksi seluruh kolom.

```
>>> print(df.loc[:,:])
```

loc di atas akan menampilkan seluruh data pada DataFrame. Parameter pertama merupakan seluruh baris, parameter kedua merupakan operator untuk seluruh label.

```
>>> print(df.loc[0])
name      anton
phone   6281210249020
status     good
Name: 0, dtype: object
```

```
>>> print(df.loc[0])
name      anton
phone   6281210249020
status     good
Name: 0, dtype: object
>>> █
```

Pada loc di atas, parameter yang diberikan adalah baris ke 0 sehingga data pada baris pertama yang ditampilkan

Contoh selanjutnya :

```
>>> print(df.loc[:, 'name'])
0  anton
1  adam
2  linda
Name: name, dtype: object
```

Pada contoh di atas seluruh baris akan diseleksi dan kolom nama akan diseleksi.

Kesimpulan

Pada penggunaan loc, parameter pertama merupakan nomor baris yang ingin diseleksi dan parameter kedua merupakan label (kolom). Jika parameter pertama diberikan tanda ":" maka akan menyeleksi seluruh baris. Jika parameter kedua diberikan tanda ":" maka akan menyeleksi seluruh kolom.

3.8.5.2. iloc

iloc merupakan object yang digunakan untuk seleksi pada DataFrame berdasarkan integer. Iloc merupakan alias untuk class “_iLocIndexer”.

Pada dasarnya penggunaan iloc adalah serupa dengan loc, di mana parameter pertama merupakan baris dan parameter kedua merupakan kolom.

Berikut ini format dari iloc :

loc(parameter_nomor_baris, parameter_berupa_nomor_kolom)

Berikut ini contoh contoh penggunaan iloc dari python shell :

```
>>> import pandas as pd
>>> df = pd.read_csv("sample.csv")
>>>
>>> #rows
... print(df.iloc[0]) # first row of data frame
name      anton
phone   6281210249020
status     good
Name: 0, dtype: object
>>> print(df.iloc[1]) # second row of data frame
name      adam
phone   6281210249121
status     bad
Name: 1, dtype: object
>>> print(df.iloc[-1]) # last row of data frame
name      linda
phone   628122222222
status     good
Name: 2, dtype: object
>>> #columns
... print(df.iloc[:,0]) # first column of data frame
0    anton
1    adam
2    linda
Name: name, dtype: object
>>> print(df.iloc[:,1]) # second column of data frame
0    6281210249020
1    6281210249121
2    628122222222
Name: phone, dtype: int64
>>> print(df.iloc[:, -1]) # last column of data frame
0    good
1    bad
2    good
Name: status, dtype: object
```

```

>>>
    type: http , copyright , credits or license for more information.
>>> import pandas as pd

>>> df = pd.read_csv("sample.csv")
>>>
>>> #rows
... print(df.iloc[0]) # first row of data frame
name      anton
phone    6281210249020
status     good
Name: 0, dtype: object
>>> print(df.iloc[1]) # second row of data frame
name      adam
phone    6281210249121
status     bad
Name: 1, dtype: object
>>> print(df.iloc[-1]) # last row of data frame
name      linda
phone    6281222222222
status     good
Name: 2, dtype: object
>>> #columns
... print(df.iloc[:,0]) # first column of data frame
0    anton
1    adam
2    linda
Name: name, dtype: object
>>> print(df.iloc[:,1]) # second column of data frame
0    6281210249020
1    6281210249121
2    6281222222222
Name: phone, dtype: int64
>>> print(df.iloc[:, -1]) # last column of data frame
0    good
1    bad
2    good
Name: status, dtype: object
>>> █

```

Penjelasan

#rows
print(df.iloc[0]) # first row of data frame

iloc di atas digunakan untuk seleksi pada baris pertama

print(df.iloc[1]) # second row of data frame

iloc di atas digunakan untuk seleksi pada baris kedua

print(df.iloc[-1]) # last row of data frame

iloc di atas digunakan untuk seleksi pada baris terakhir

#columns
print(df.iloc[:,0]) # first column of data frame

iloc di atas digunakan untuk seleksi pada seluruh baris pada kolom pertama

```
print(df.iloc[:,1]) # second column of data frame
```

iloc di atas digunakan untuk seleksi pada seluruh baris pada kolom kedua

```
print(df.iloc[:, -1]) # last column of data frame
```

iloc di atas digunakan untuk seleksi pada seluruh baris pada kolom terakhir

3.8.5.3. ix

ix digunakan untuk seleksi data berdasarkan integer dan label. Ix ini merupakan hybrid dari loc dan iloc.

Berikut ini format dari ix :

loc(parameter_nomor_baris, parameter_berupa_nomor_kolom_atau_nomor_kolom)

Jika parameter pertama diberikan tanda “:” maka akan menyeleksi seluruh baris. Jika parameter kedua diberikan tanda “:” maka akan menyeleksi seluruh kolom.

Berikut ini contoh penggunaan ix :

```
>>> print(df.ix[0])
name      anton
phone    6281210249020
status     good
Name: 0, dtype: object
>>> print(df.ix[0, 'name'])
anton
>>> print(df.ix[0, 0])
anton
```

```
>>> print(df.ix[0])
name      anton
phone    6281210249020
status     good
Name: 0, dtype: object
>>> print(df.ix[0, 'name'])
anton
>>> print(df.ix[0, 0])
anton
>>> █
```

Penjelasan

```
>>> print(df.ix[0])
```

Contoh di atas digunakan untuk menampilkan data pada baris pertama

```
>>> print(df.ix[0, 'name'])
```

Contoh di atas digunakan untuk menampilkan data pada baris pertama kolom nama

```
>>> print(df.ix[0, 0])
```

Contoh di atas digunakan untuk menampilkan data pada baris pertama, kolom pertama

3.8.6. Slice Data Frame

Kita bisa melakukan slicing pada dataframe. Berikut ini contoh slicing dataframe :

```
>>> print(df[0:3])
   name      phone status
0 anton 6281210249020  good
1 adam 6281210249121  bad
2 linda 6281222222222  good
>>> print(df[:2])
   name      phone status
0 anton 6281210249020  good
1 adam 6281210249121  bad
>>> print(df[-1:])
   name      phone status
2 linda 6281222222222  good
```

```
>>> import pandas as pd
>>> df = pd.read_csv("sample.csv")
>>> print(df[0:3])
   name      phone status
0 anton 6281210249020  good
1 adam 6281210249121  bad
2 linda 6281222222222  good
>>> print(df[:2])
   name      phone status
0 anton 6281210249020  good
1 adam 6281210249121  bad
>>> print(df[-1:])
   name      phone status
2 linda 6281222222222  good
>>> █
```

Penjelasan

```
>>> print(df[0:3])
```

Digunakan untuk menampilkan dari row 0 hingga row 2

```
>>> print(df[:2])
```

Karena parameter slice pertama tidak diberikan maka dianggap 0, parameter kedua adalah 2, sehingga data akan ditampilkan mulai row 0 hingga row 1

```
>>> print(df[-1:])
```

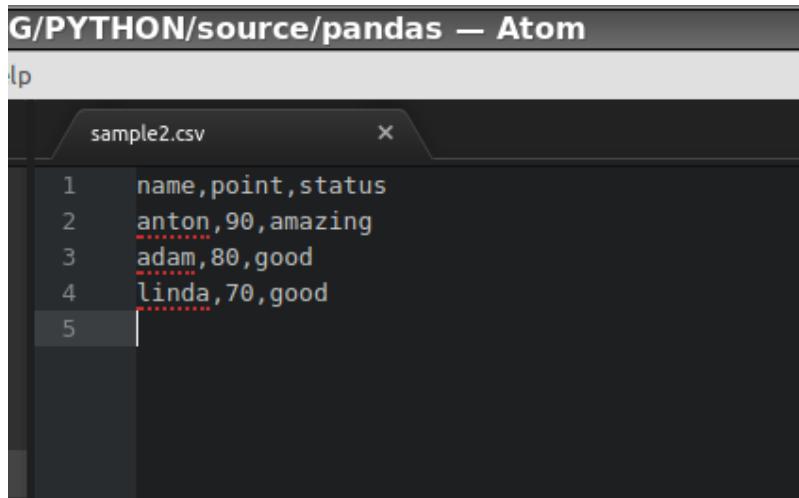
Digunakan untuk menampilkan baris terakhir

3.8.7. Pencarian Data pada Series dan DataFrame

Untuk mencari data di DataFrame kita bisa menggunakan method where().

Untuk contoh pencarian pada dataframe, kita akan menyiapkan file csv baru yaitu sample2.csv :

```
name,point,status
anton,90,amazing
adam,80,good
linda,70,good
```



The screenshot shows a dark-themed code editor window titled 'G/PYTHON/source/pandas — Atom'. A tab labeled 'sample2.csv' is open. The file contains the following CSV data:

```
1 name,point,status
2 anton,90,amazing
3 adam,80,good
4 linda,70,good
5
```

Misal kita ingin melakukan seleksi pada baris dengan data point di atas 85, berikut ini contoh pada python shell :

```
>>> data = df['name'].where(df['point'] > 85)
>>> print(data)
0    anton
1    NaN
2    NaN
Name: name, dtype: object
```

Pada contoh di atas kita berhasil menampilkan data nama dengan point di atas 85 akan tetapi terdapat banyak NaN pada tampilan data. Untuk membersihkannya, kita bisa menggunakan method dropna()

Berikut ini contoh dengan dropna() :

```
>>> print(data.dropna())
0    anton
Name: name, dtype: object
>>> print(data.dropna().values)
['anton']
```

Untuk materi lanjutan dengan pandas akan dibahas dalam kursus selanjutnya “Python for Machine Learning”

3.9. SPEECH RECOGNITION DAN TEXT TO SPEECH DENGAN PYTHON

Speech recognition merupakan teknik translasi dari suara manusia menjadi teks.

Text to Speech merupakan teknik translasi dari teks menjadi suara manusia.

3.9.1. Persiapan

Sebelum memulai, kita perlu menginstall beberapa modul python dan beberapa modul tambahan. Selain itu diperlukan persiapan hardware yang mendukung speech recognition dan text to speech.

3.9.1.1. Persiapan di Hardware

Untuk memulai speech recognition, kita memerlukan beberapa perangkat berikut ini :

USB Sound Card Adapter

Perangkat ini merupakan adapter untuk usb ke input (microphone) dan output sound (speaker). Berikut ini beberapa model yang tersedia di pasaran :



atau bisa juga yang memiliki beberapa channel input dan output. Contoh:



Selanjutnya kita memerlukan speaker dan microphone. Bisa menggunakan 1 paket headset berikut ini yang terdapat microphone dan speaker sekaligus :



atau bisa juga terpisah berupa speaker tersendiri dan microphone tersendiri. Contoh speaker yang beredar di pasaran :



Contoh microphone yang tersedia di pasaran :

Mic For Laptop



SOONHUA



3.9.1.2. Persiapan di Linux

“Sebelum memulai pastikan usb soundcard, speaker dan microphone sudah terpasang di komputer”

Pada contoh ini, kita akan menggunakan pulseaudio untuk pengaturan input dan output suara di linux. Secara default, pada ubuntu sudah menggunakan pulseaudio sehingga tidak perlu dikutak katik lagi.

Modul utama yang perlu diinstall adalah SpeechRecognition. Dari terminal linux, ketikkan :

```
sudo apt-get install alsa-utils  
git clone http://people.csail.mit.edu/hubert/git/pyaudio.git  
cd pyaudio  
sudo python setup.py install  
sudo apt-get install libportaudio-dev  
sudo apt-get install python-dev  
sudo apt-get install libportaudio0 libportaudio2 libportaudiocpp0 portaudio19-dev  
sudo pip install SpeechRecognition
```

Pada linux, kita bisa menggunakan SoX untuk merekam suara, SoX digunakan untuk melakukan recording suara dari mic komputer ke dalam format file wav. Untuk menginstall SoX dari root shell :

```
apt install sox
```

atau

```
sudo apt install sox
```

Pastikan sox yang terinstall adalah versi SoX v14.4.1:

```
$ sox --version  
sox:    SoX v14.4.1
```

Jika bukan lakukan :

```
sudo apt-get purge sox
```

Download sox versi 14.4.1 dari : <https://pkgs.org/download/sox>

Sesuaikan dengan distribusi linux yang digunakan. Setelah didownload, instalasi paket. Misal kita mendownload untuk ubuntu 16.04 64 bit :

https://ubuntu.pkgs.org/16.04/ubuntu-universe-amd64/sox_14.4.1-5_amd64.deb.html

Setelah didownload, install dengan :

```
sudo dpkg -i sox_14.4.1-5_amd64.deb
```

Selanjutnya kita perlu menginstall vlc :

```
sudo add-apt-repository ppa:videolan/master-daily  
sudo apt-get update  
sudo apt-get install vlc mpg321
```

Pada kesempatan kali ini kita juga akan menggunakan text to speech offline, kita akan menggunakan espeak untuk text to speech. Install espeak :

```
apt install espeak
```

atau

```
sudo apt install espeak
```

espeak ini memiliki kelemahan menghasilkan suara yang kurang natural, untuk menghasilkan kualitas suara yang lebih baik, kita bisa memanfaatkan modul python tambahan yaitu gTTS.

Sebelum menginstall gTTS, upgrade terlebih dahulu pip dan setuptools.

```
sudo pip3 install --upgrade pip
```

```
sudo pip3 install --upgrade setuptools
```

Setelah itu download gTTS dari <https://pypi.org/project/gTTS/>

Silahkan download source code gTTS versi terbaru, pada saat penulisan materi ini, gTTS terbaru adalah versi 2.0.3.

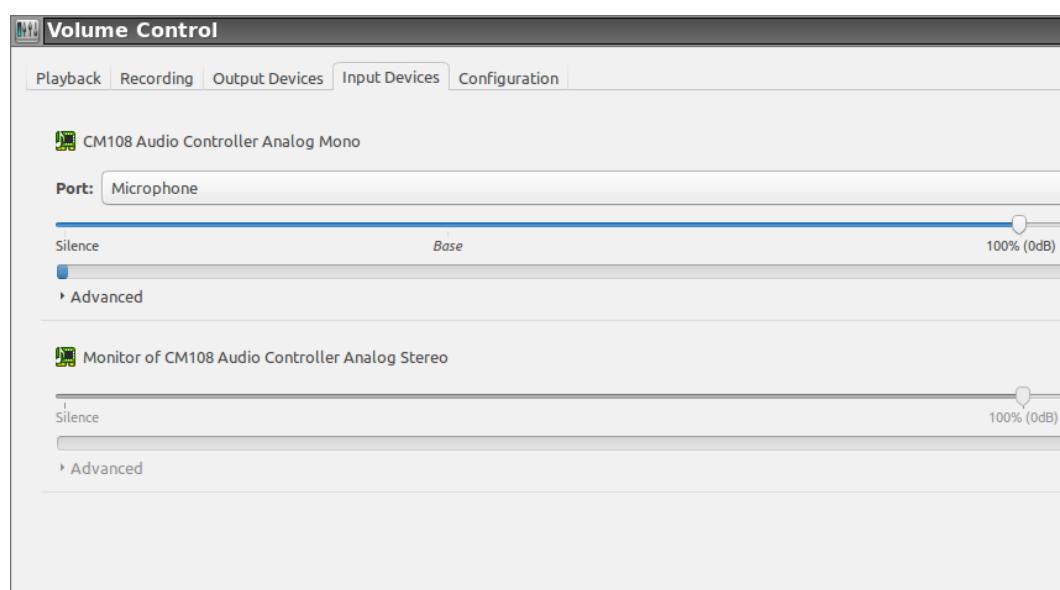
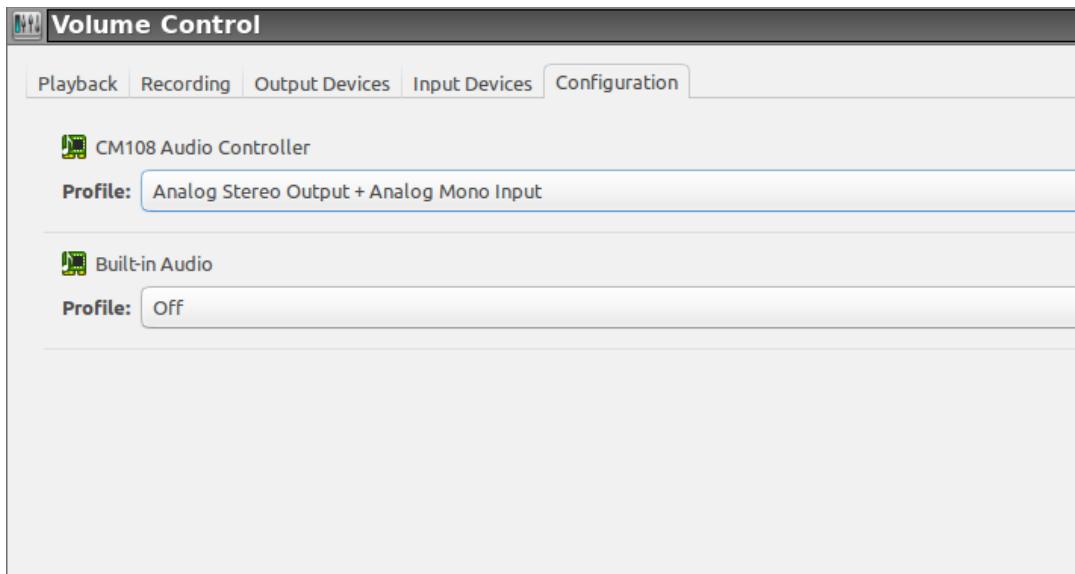
Setelah didownload ekstrak lalu lakukan install secara manual dari direktori ekstrakan :

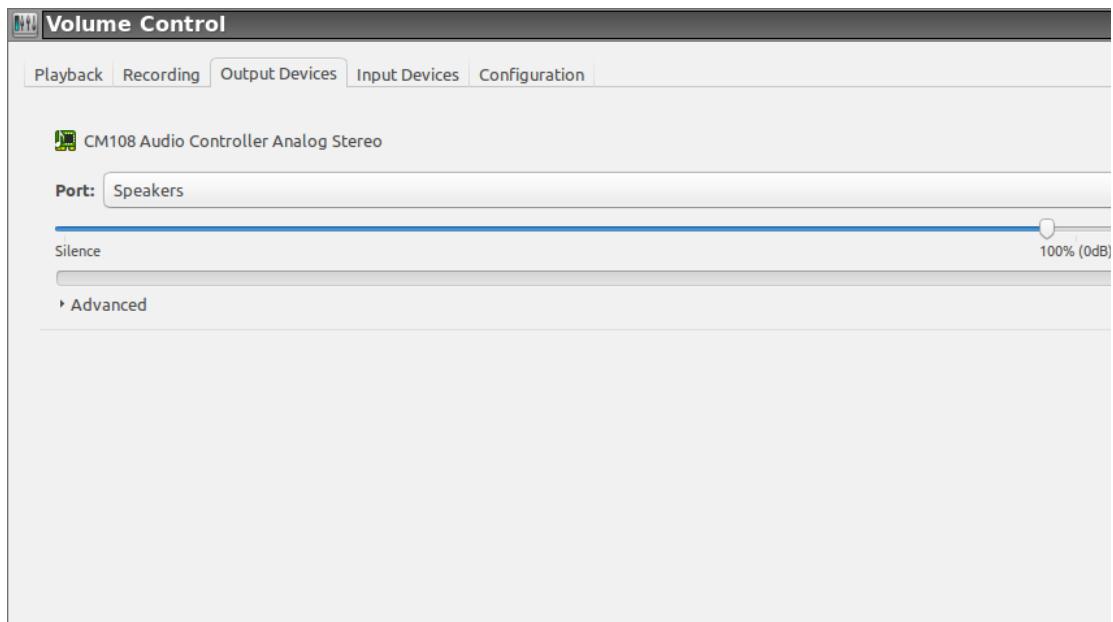
```
sudo python3 setup.py install
```

atau

```
python3 setup.py install → jika menggunakan windows
```

Selanjutnya kita perlu mengatur masukan dan keluaran channel suara dengan pulseaudio. Pada ubuntu, buka menu “Sound & Video”, selanjutnya pilih “PulseAudio Volume Control”. Berikut ini adalah settingan settingan pada volume control pulseaudio :



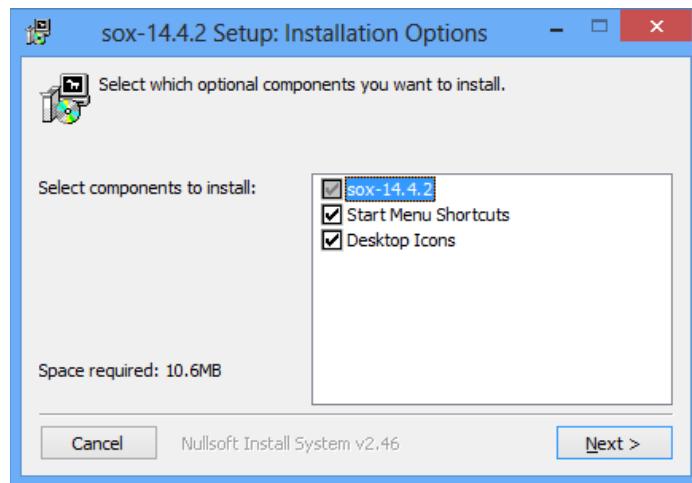


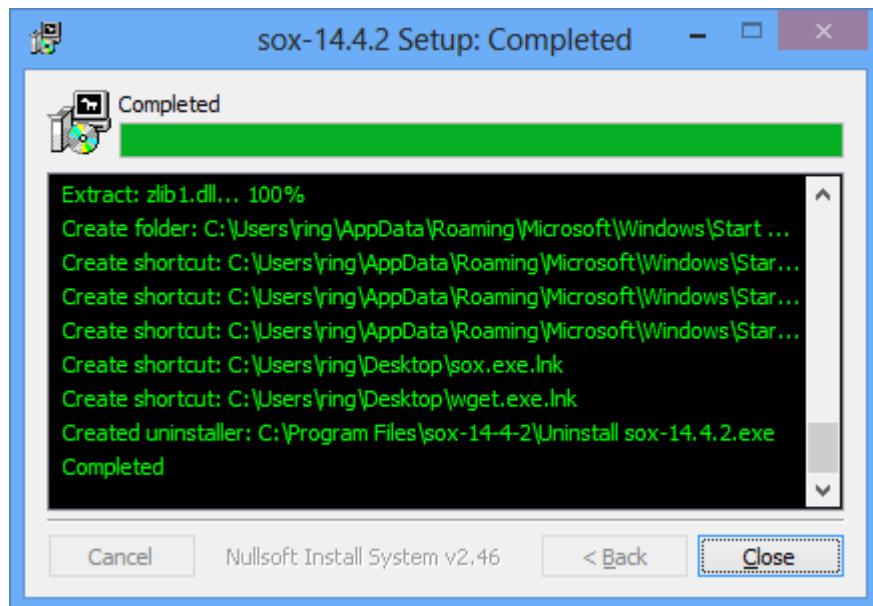
3.9.1.3. Persiapan di Windows

Pada windows, untuk merekam suara kita bisa menggunakan fmedia atau SoX yang bisa kita jalankan melalui ms dos prompt.

SoX untuk windows bisa didownload di <https://sourceforge.net/projects/sox/files/sox/14.4.2/sox-14.4.2-win32.exe/download>

Selanjutnya install sox

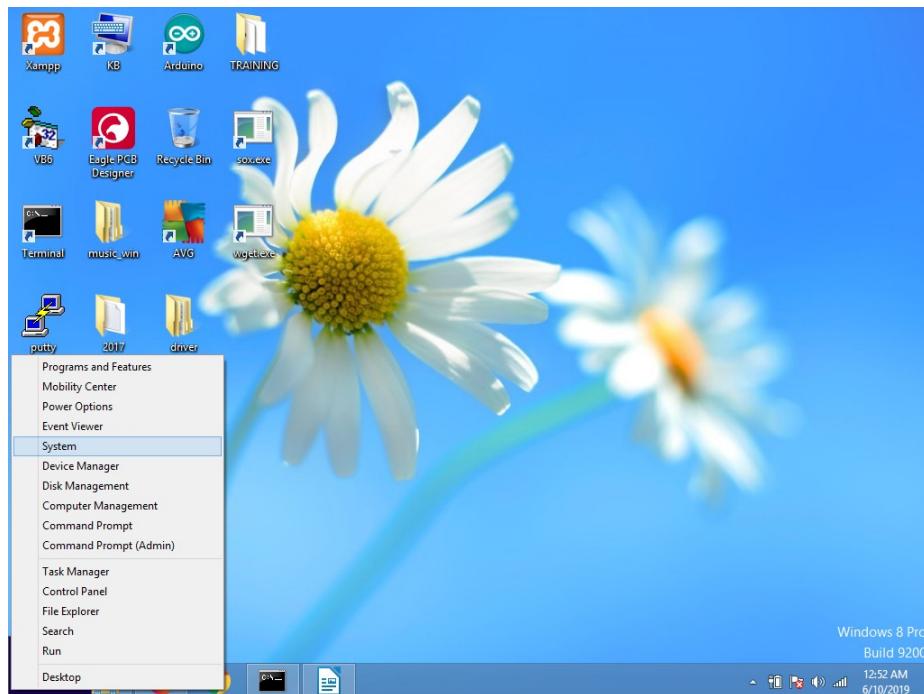


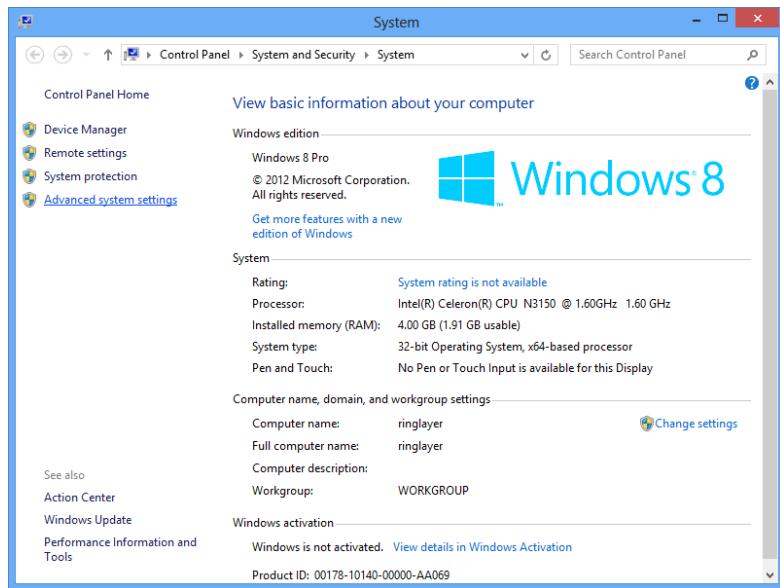


Selanjutnya kita perlu melakukan setting environment path agar SoX bisa dijalankan secara global melalui ms dos.

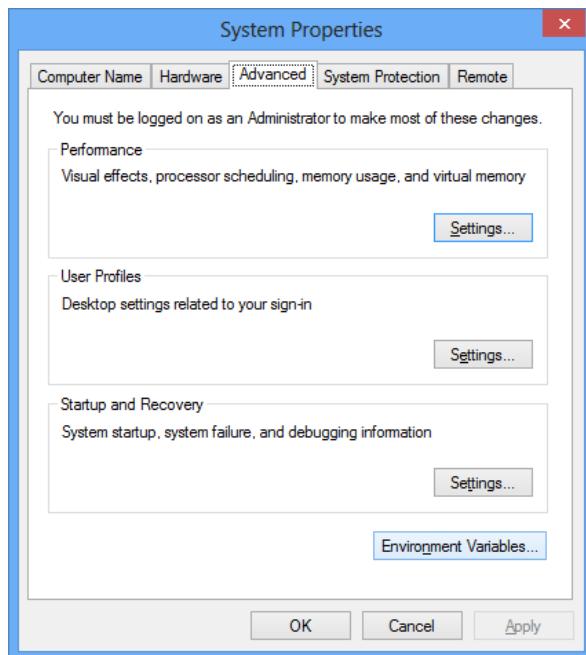
Cara untuk setting environment path untuk windows bisa dicek di
<https://www.computerhope.com/issues/ch000549.htm>

Pada contoh ini menggunakan windows 8, arahkan mouse ke pojok paling kiri layar bagian bawah lalu klik kanan dan pilih "System"

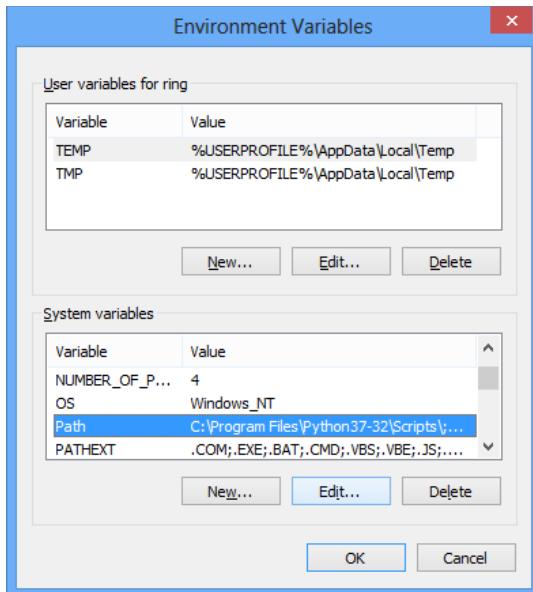




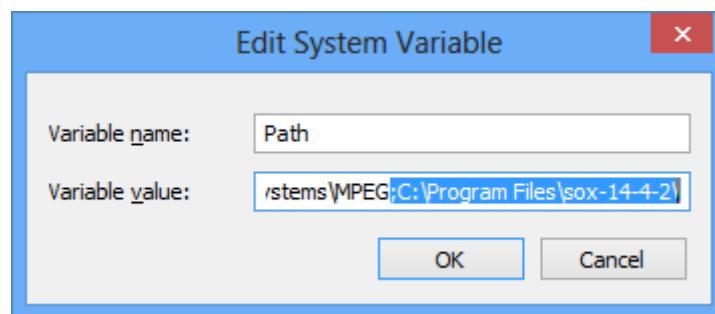
selanjutnya pilih tab advance lalu pilih environment variables



Edit pada bagian path



Tambahkan :



;C:\Program Files\sox-14-4-2\

Selanjutnya klik ok. Jika berhasil maka sox bisa dijalankan langsung dari path manapun dari ms dos

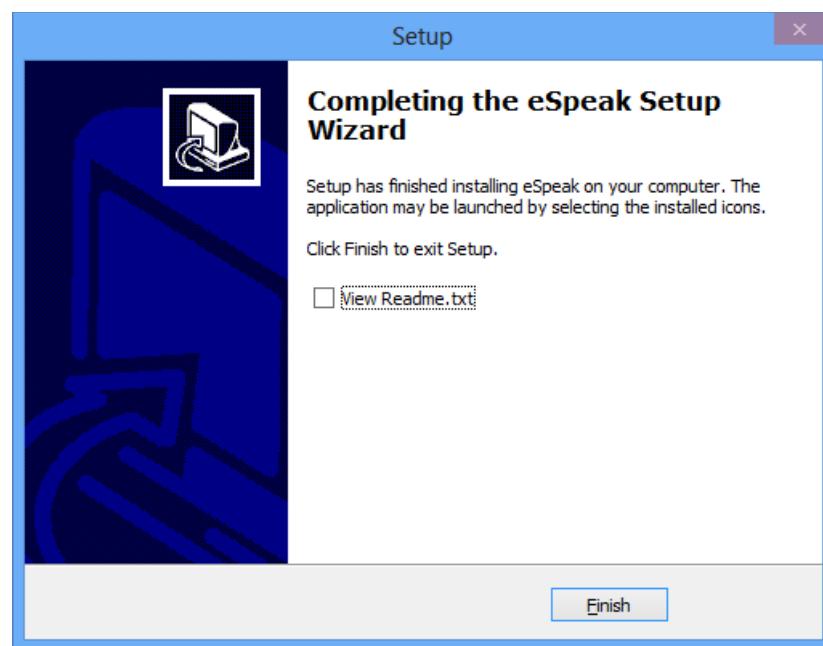
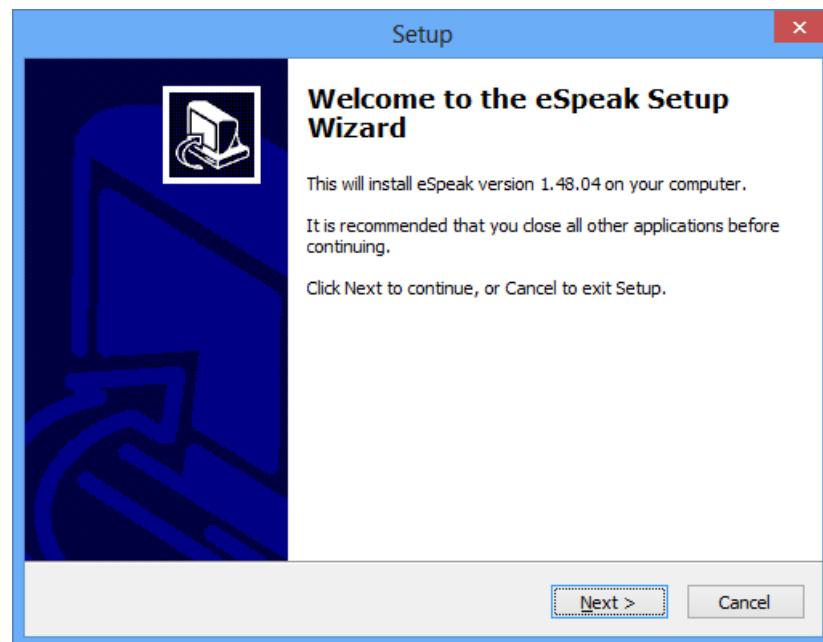
```
Microsoft Windows [Version 6.2.9200]
(c) 2012 Microsoft Corporation. All rights reserved.

C:\windows\system32>cd \users
C:\Users>sox
sox: SoX v14.4.2
sox FAIL sox: Not enough input filenames specified
Usage summary: [gopts] [[fopts] infile]... [fopts] outfile [effect [effopts]]...
SPECIAL FILENAMES <infile, outfile>:
- Pipe/redirect input/output <stdin/stdout>; may need -t
-d, --default-device Use the default audio device (where available)
-n, --null Use the 'null' file handler; e.g. with synth effect
-p, --sox-pipe Alias for '-t sox -'
SPECIAL FILENAMES <infile only>:
"!program [options] ..." Pipe input from external program (where supported)
http://server/file Use the given URL as input file (where supported)

GLOBAL OPTIONS <gopts> (can be specified at any point before the first effect):
--buffer BYTES Set the size of all processing buffers (default 8192)
--clobber Don't prompt to overwrite output file (default)
```

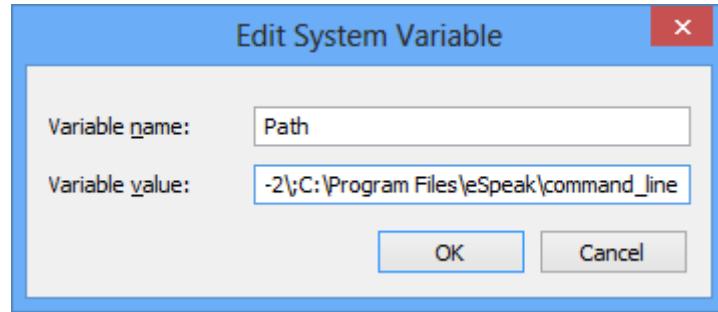
Selanjutnya anda perlu menginstall espeak , bisa didownload dari
http://sourceforge.net/projects/espeak/files/espeak/espeak-1.48/setup_espeak-1.48.04.exe

Install espeak :

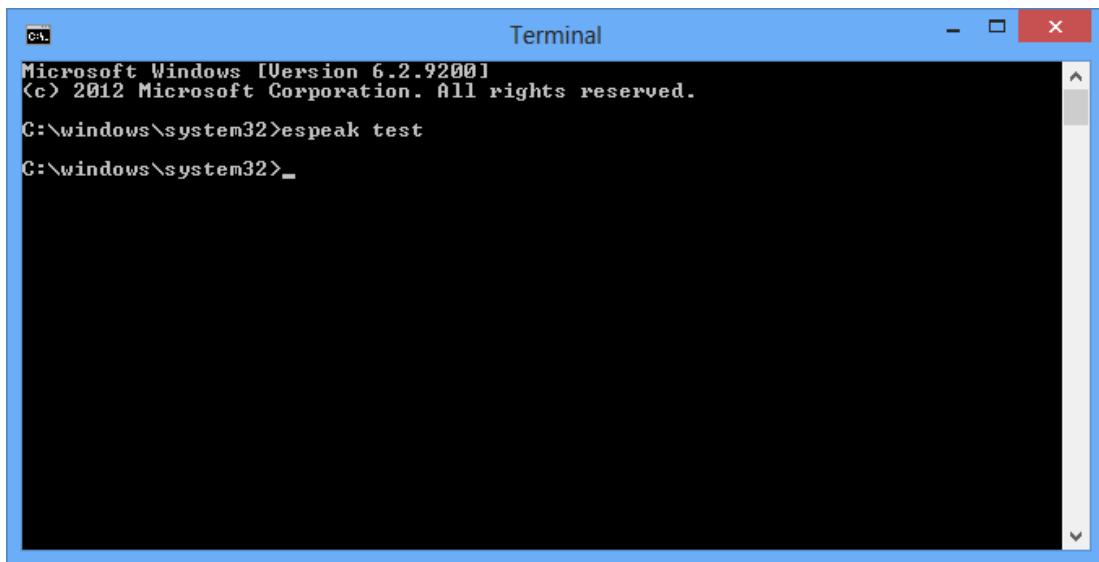


Sama seperti tadi, kita perlu meregister espeak di path environment, Pada path tambahkan :

;C:\Program Files\eSpeak\command_line



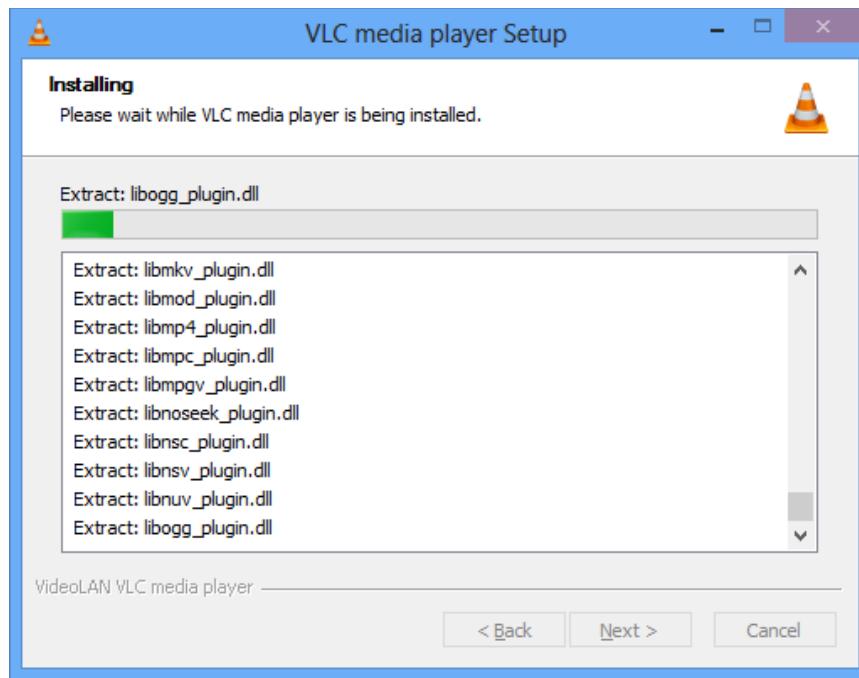
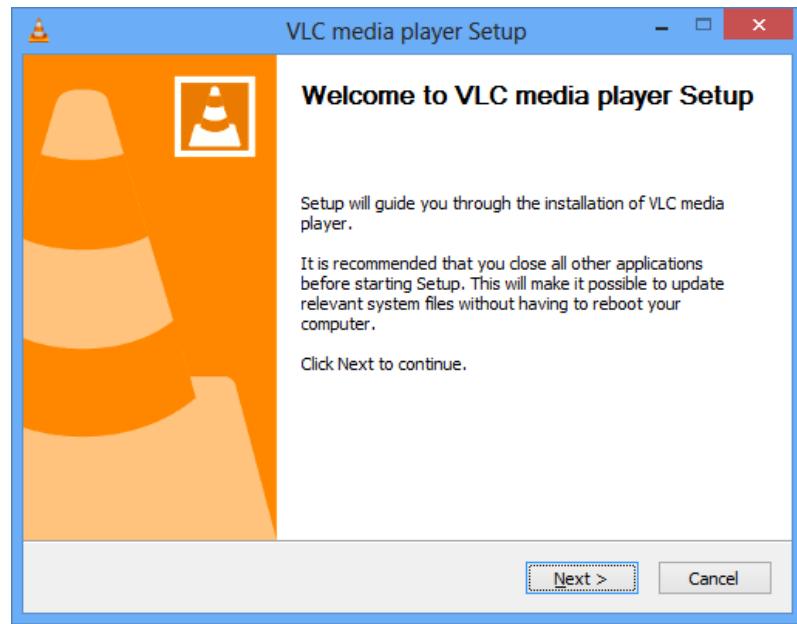
Jika sudah espeak bisa dijalankan dari path manapun dari command prompt



Selanjutnya kita memerlukan vlc yang bisa didownload di <https://www.videolan.org/vlc/download-windows.id.html>

<https://mirror.downloadvn.com/videolan/vlc/3.0.7/win32/vlc-3.0.7-win32.exe>

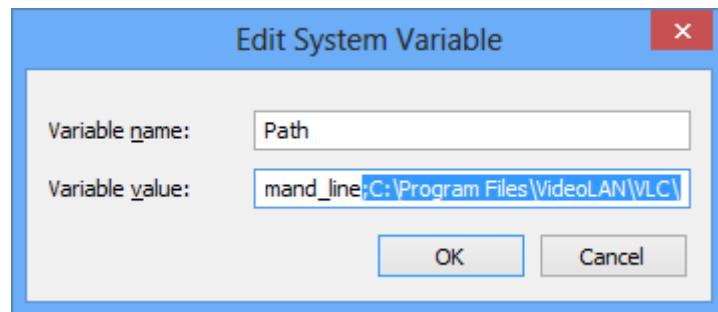
Download dan install vlc di komputer anda



Setelah vlc terinstall, sama seperti sebelumnya kita perlu menambahkan path environment untuk vlc

Tambahkan line ini :

;C:\Program Files\VideoLAN\VLC\



Selanjutnya kita perlu mendownload dan menginstall fmedia. Fmedia merupakan aplikasi untuk merekam suara dari microphone di windows.

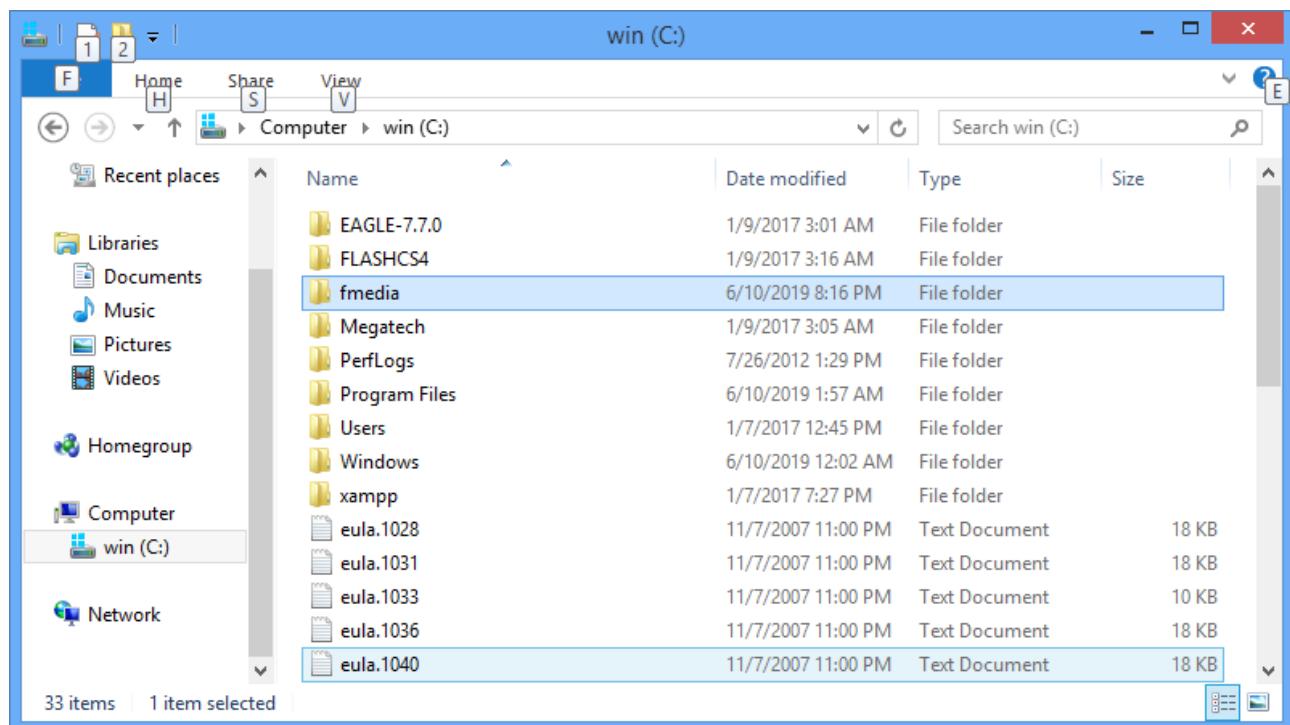
Untuk windows 32 bit :

<http://fmedia.firmdev.com/fmedia-1.6-win-x86.zip>

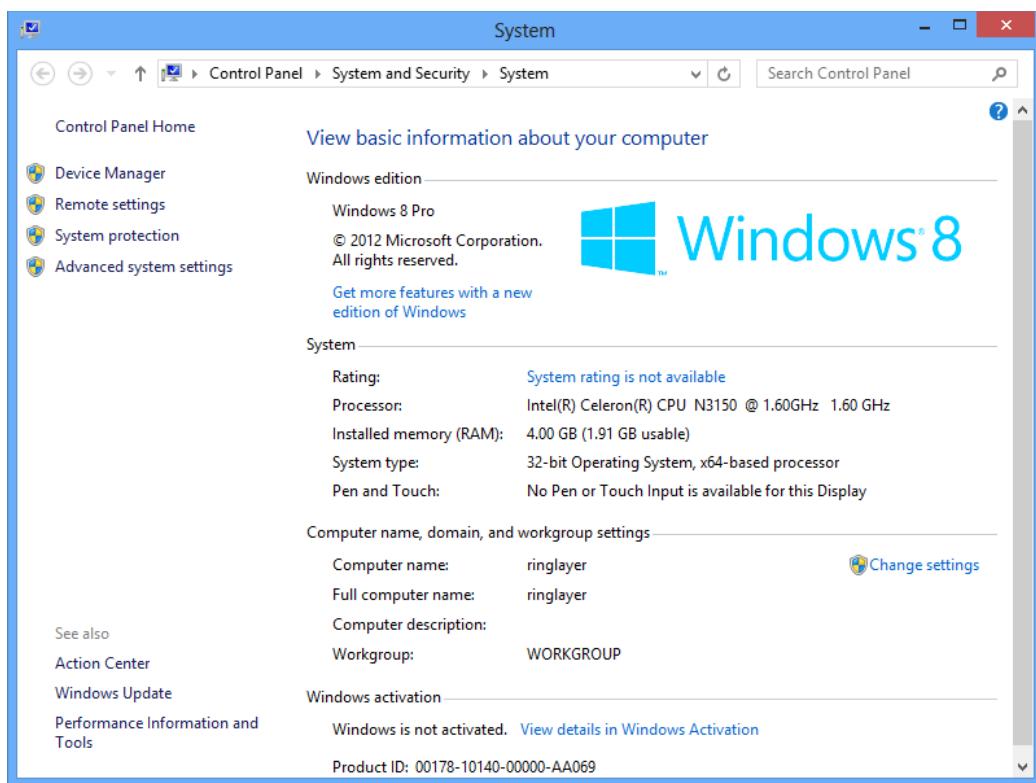
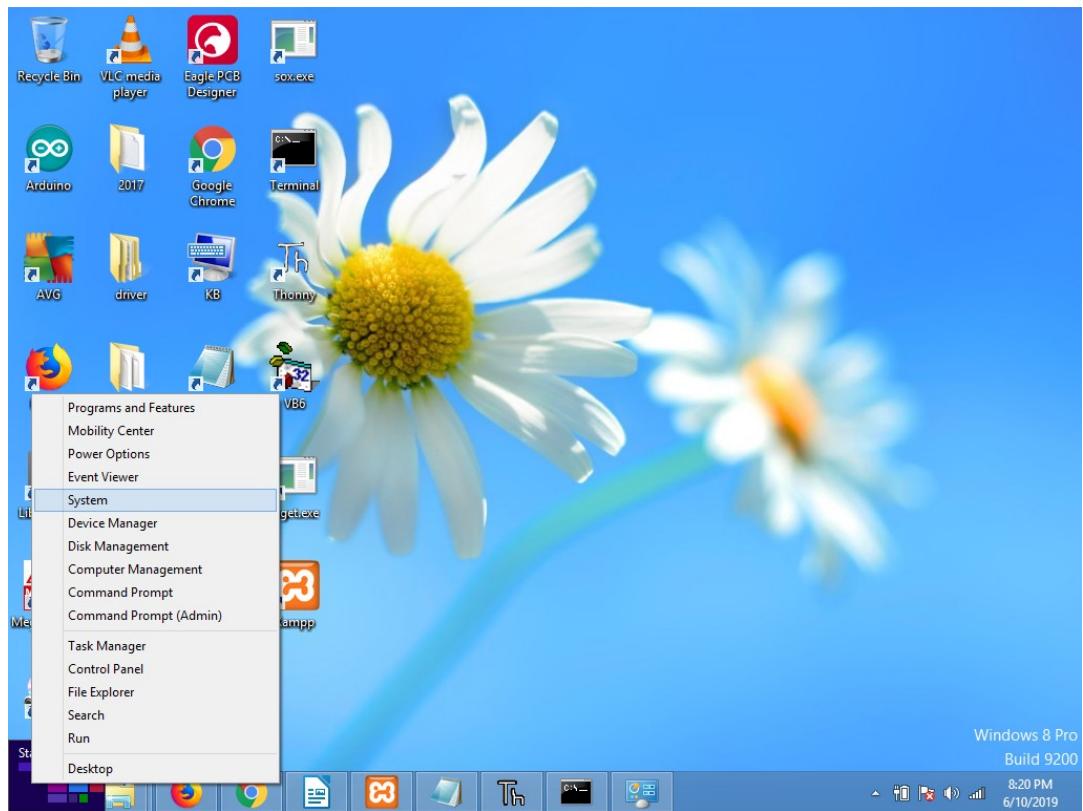
Untuk windows 64 bit :

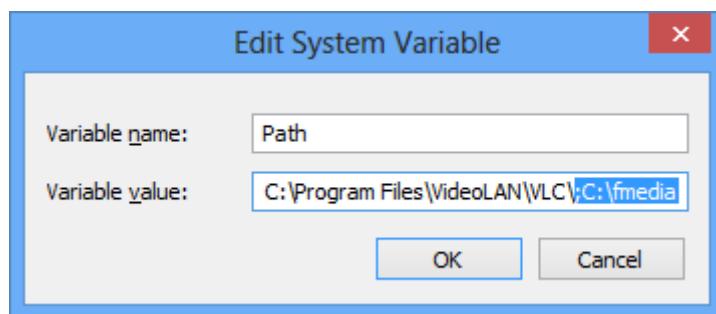
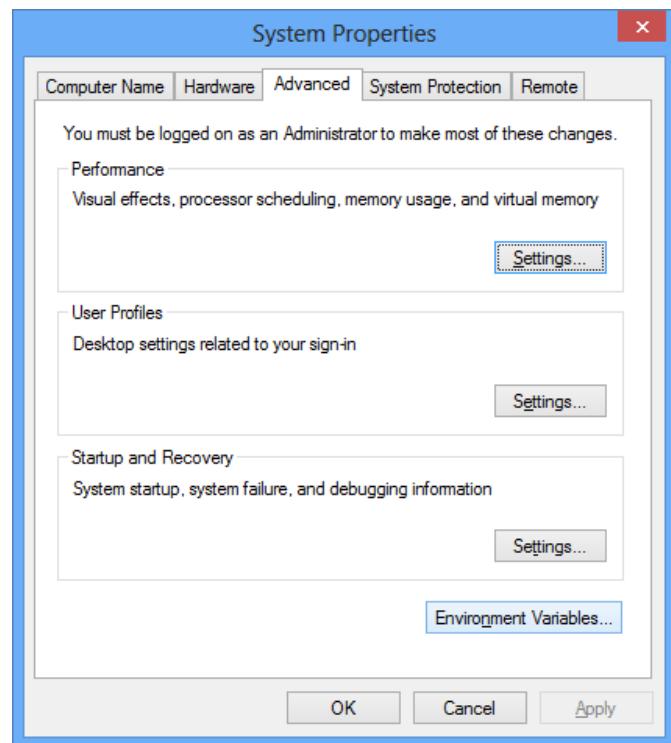
<http://fmedia.firmdev.com/fmedia-1.9-win-x64.zip>

Setelah didownload, ekstrak dan kopikan folder fmedia ke <c:\>



Sama seperti langkah langkah sebelumnya, kita perlu menambahkan path exe fmedia ke environment path windows.

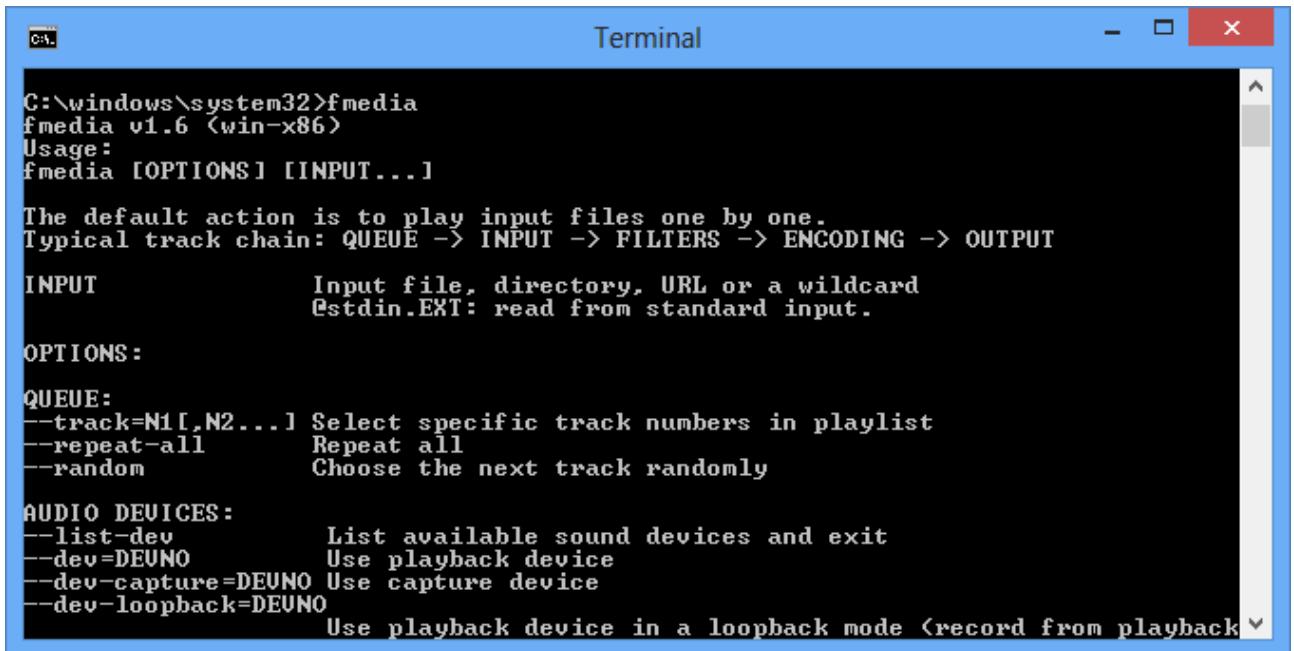




Tambahkan pada path :

`;C:\fmedia`

Untuk menguji apakah sudah berhasil, buka ms dos prompt lalu ketik : fmedia



```
C:\>fmedia
fmedia v1.6 <win-x86>
Usage:
fmedia [OPTIONS] [INPUT...]

The default action is to play input files one by one.
Typical track chain: QUEUE -> INPUT -> FILTERS -> ENCODING -> OUTPUT

INPUT           Input file, directory, URL or a wildcard
@stdin.EXT: read from standard input.

OPTIONS:

QUEUE:
--track=N1[,N2...] Select specific track numbers in playlist
--repeat-all      Repeat all
--random          Choose the next track randomly

AUDIO DEVICES:
--list-dev        List available sound devices and exit
--dev=DEUNO       Use playback device
--dev-capture=DEUNO Use capture device
--dev-loopback=DEUNO Use playback device in a loopback mode <record from playback>
```

Setelah semua terinstall, kita install modul-modul python dari ms dos command prompt (jalankan command prompt sebagai administrator agar bisa menginstall) :

```
pip install --upgrade pip
pip install --upgrade setuptools
pip install gtts
pip install SpeechRecognition
```

3.9.2. Speech Recognition dengan Google

Pada contoh kali ini kita akan membuat aplikasi speech recognition dengan google api, cara kerja aplikasi ini adalah aplikasi akan melakukan looping terus menerus di mana di dalam looping, aplikasi akan merekam suara selama 5 detik ke dalam format file flac, kemudian file flac tersebut akan dikirim ke server google api untuk diproses server api google kemudian aplikasi akan membaca response hasil recognize dari google, pada langkah terakhir aplikasi akan menjalankan text to speech espeak untuk menyebutkan hasil recognize dari server google. Selain menggunakan google, ada beberapa alternatif untuk speech recognition, yang offline misalnya bisa menggunakan deepspeech, akan dibahas pada training python khusus untuk machine learning.

3.9.2.1. Merekam File Suara untuk Speech Recognition di Linux

Sebelum memulai pastikan usb soundcard, speaker dan microphone sudah terpasang di komputer. Pertama tama pada linux kita perlu mengidentifikasi nomor perangkat untuk microphone yang terpasang , langkah ini diperlukan karena sox memerlukan nomor device dan nomor card. Pola perekaman suara yang kita lakukan dengan sox :

```
AUDIODEV=plughw:1,0 rec --channels=1 --bits=24 --rate=44100 tmp.flac trim 0 5
```

Keterangan :

plughw:1,0 → Kita menggunakan usb pnp (plughw) dengan nomor device 1 dan nomor card 0.

--channels=1 → yang kita butuhkan adalah mono (kita hanya perlu 1 channel suara saja)

--bits=24 → atau bisa juga *--bits=16*

--rate=44100 → file flac memerlukan minimal rate 44100

“trim 0 5” → berfungsi untuk menghentikan rekaman setelah 5 detik

Untuk deteksi perangkat suara, dari terminal ketikkan :

```
cat /proc/asound/cards
```

Pada contoh ini :

```
$ cat /proc/asound/cards
0 [Device      ]: USB-Audio - USB PnP Sound Device
                  USB PnP Sound Device at usb-0000:00:14.0-1.1, full speed
1 [PCH        ]: HDA-Intel - HDA Intel PCH
                  HDA Intel PCH at 0xc4520000 irq 139
```

Di sini kita menggunakan device dengan nomor **1**.

Selanjutnya ketikkan : *aplay -L | grep hw*

Berikut ini contoh tampilan :

```
$ aplay -L | grep hw
```

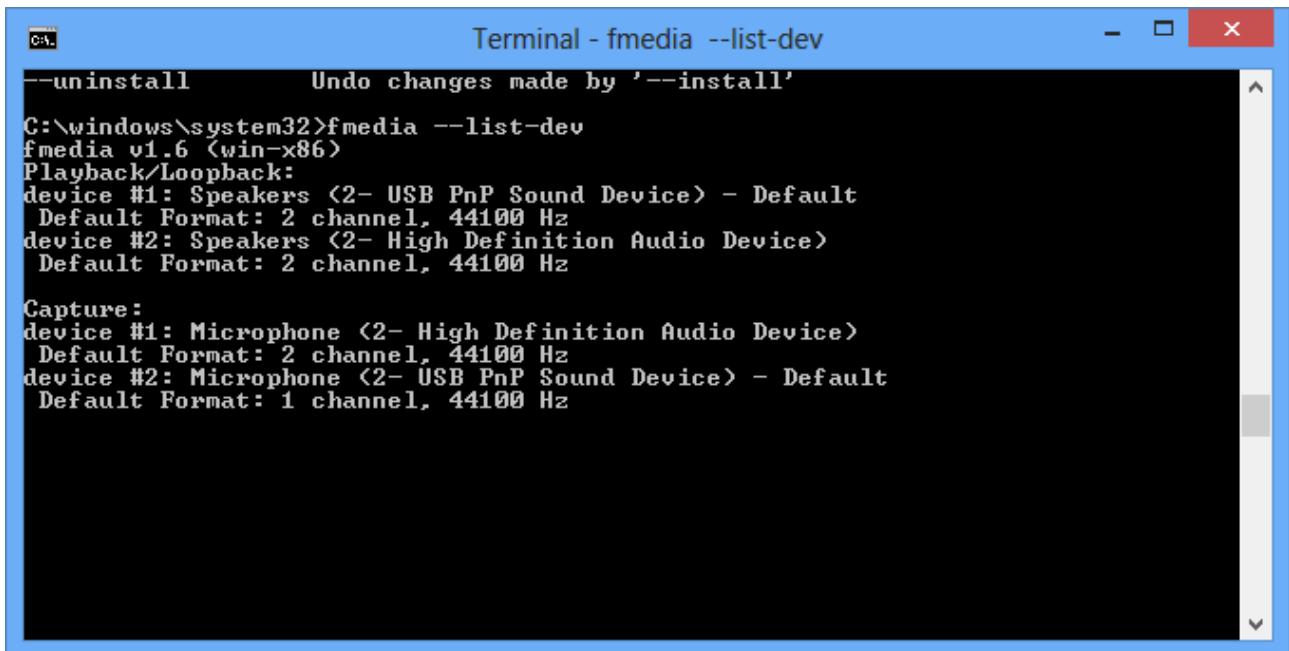
```
hw:CARD=Device,DEV=0
plughw:CARD=Device,DEV=0
hw:CARD=PCH,DEV=0
hw:CARD=PCH,DEV=3
hw:CARD=PCH,DEV=7
hw:CARD=PCH,DEV=8
plughw:CARD=PCH,DEV=0
plughw:CARD=PCH,DEV=3
plughw:CARD=PCH,DEV=7
```

Pada contoh di atas kita akan menggunakan plughw dengan nomor device **0**. Sehingga format dari perintah sox yang kita lakukan adalah :

```
AUDIODEV=plughw:1,0 rec --channels=1 --bits=24 --rate=44100 tmp.flac trim 0 5
```

3.9.2.2. Merekam File Suara untuk Speech Recognition di Windows

Sebelum memulai pastikan usb soundcard, speaker dan microphone sudah terpasang di komputer. Untuk merekam suara di windows kita akan menggunakan fmedia. Sebelumnya cek dulu nomor device dengan mengetik fmedia –list-dev



```
--uninstall      Undo changes made by '--install'
C:\windows\system32>fmedia --list-dev
fmedia v1.6 <win-x86>
Playback/Loopback:
device #1: Speakers <2- USB PnP Sound Device> - Default
  Default Format: 2 channel, 44100 Hz
device #2: Speakers <2- High Definition Audio Device>
  Default Format: 2 channel, 44100 Hz

Capture:
device #1: Microphone <2- High Definition Audio Device>
  Default Format: 2 channel, 44100 Hz
device #2: Microphone <2- USB PnP Sound Device> - Default
  Default Format: 1 channel, 44100 Hz
```

Kita akan menggunakan usb pnp, sehingga kita akan menggunakan nomor device 2

Berikut ini untuk merekam suara dengan fmedia :

```
fmedia --dev-capture=2 --volume=98 --channels=mono --rate=16000 --record -o mono.flac --until=5
```

3.9.2.3. Text to Speech dengan Espeak

Untuk text to speech kita akan menggunakan espeak, berikut ini pola perintah yang akan kita gunakan :

```
espeak -ven-us+f5 -s 100
```

-ven-us+f5 → di sini kita gunakan suara female berbahasa inggris

-s 100 → di sini kita gunakan speed 100

Selain menggunakan espeak, kita bisa juga menggunakan modul python gtts di mana modul bisa juga digunakan untuk pengucapan kata berbahasa indonesia.

3.9.2.4. Melakukan Translasi Suara ke Text

Untuk melakukan translasi suara ke text kita memerlukan modul SpeechRecognition.

Pertama tama import terlebih dahulu modul ini :

```
import speech_recognition as sr
```

Selanjutnya, buat objek recognizer baru :

```
r = sr.Recognizer()
```

selanjutnya :

```
AUDIO_FILE = "tmp.flac"
with sr.AudioFile(AUDIO_FILE) as source:
    r.adjust_for_ambient_noise(source)
    audio = r.record(source)
    res = r.recognize_google(audio)
    print("\n" + res + "\n")
```

Pada contoh di atas kita akan merecognize dari file flac tmp.flac (selain file flac, bisa juga dari file wav), `adjust_for_ambient_noise()` berfungsi untuk membersihkan noise, selanjutnya dilakukan proses recognize dengan google api:

```
audio = r.record(source)
res = r.recognize_google(audio)
```

Berikut ini contoh source speech recognition untuk linux

```
#!/usr/bin/env python3
"""
speechrecog_google.py
```

```

for python training at jasaplus.com
"""

import speech_recognition as sr
import os
AUDIO_FILE = "tmp.flac"
r = sr.Recognizer()
while True:
    try:
        os.system("AUDIODEV=plughw:1,0 rec --channels=1 --bits=24 --rate=44100 tmp.flac trim 0 5")
        with sr.AudioFile(AUDIO_FILE) as source:
            r.adjust_for_ambient_noise(source)
            audio = r.record(source)
            res = r.recognize_google(audio)
            print("\n" + res + "\n")
            os.system("espeak -ven-us+f5 -s 100 '"+res+"'")
    except Exception as e:
        print(e)
    pass

```

Untuk windows memerlukan sedikit penyesuaian saat perekaman suara, di mana kita tidak menggunakan sox, tetapi fmedia

```

#!/usr/bin/env python3
"""

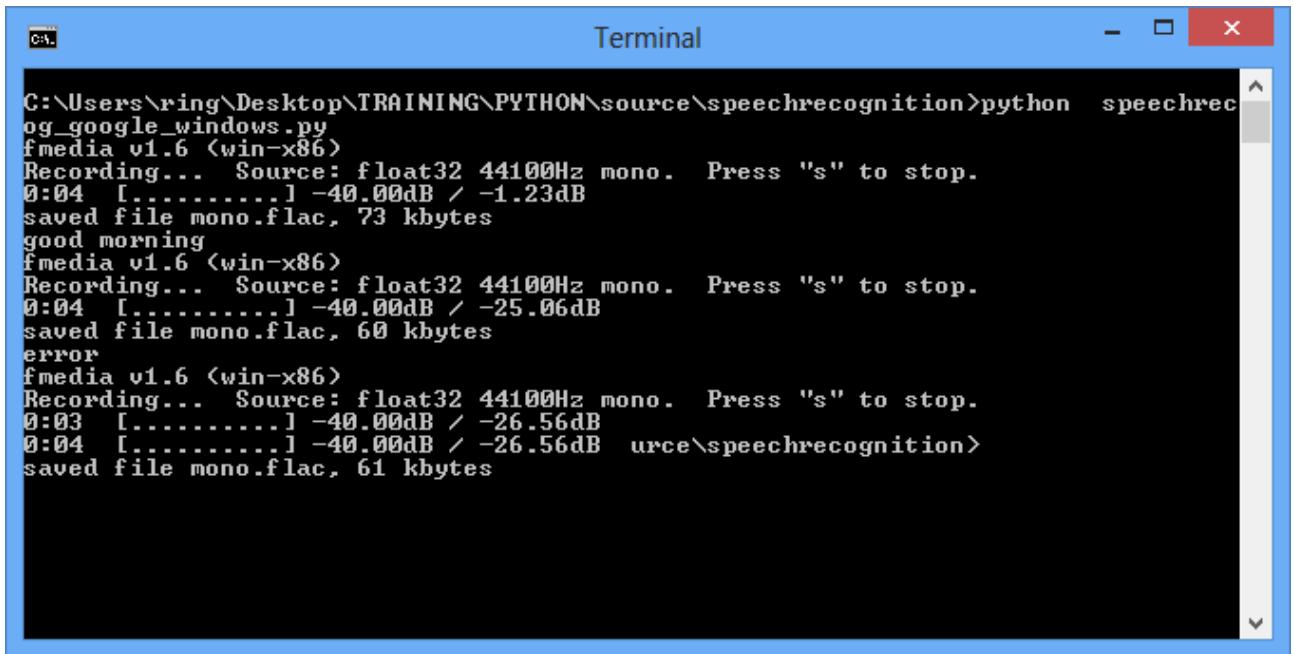
speechrecog_google_windows.py
sample for python training
at jasaplus.com
"""

import speech_recognition as sr
import time, os
AUDIO_FILE = "mono.flac"

while True:
    try:
        os.system("del mono.flac")
        os.system("fmedia --dev-capture=2 --volume=98 --channels=mono --rate=16000 --record -o mono.flac --until=5")
        r = sr.Recognizer()
        with sr.AudioFile(AUDIO_FILE) as source:
            r.adjust_for_ambient_noise(source)
            audio = r.record(source)
            res = r.recognize_google(audio)
            print(res)
            #os.system("espeak -ven-us+f5 -s 100 '"+res+"' --stdout | aplay -D 'default'")
            os.system('espeak -ven-us+f5 -s 100 "'+res+'"')
            time.sleep(1)
    except:
        print("error")
    pass

```

Jalankan aplikasi di atas dan ucapkan kata berbahasa inggris saat perekaman



A screenshot of a Windows Terminal window titled "Terminal". The window displays command-line output from a Python script named "speechrecognition.py". The script uses the "fmedia v1.6 <win-x86>" library to record audio at 44100Hz mono. It shows three recordings: one for "good morning" (73 kbytes), one for "error" (60 kbytes), and one for "source\speechrecognition>" (61 kbytes). Each recording includes a waveform visualization and a dB range from -40.00dB to -26.56dB.

```
C:\Users\ring\Desktop\TRAINING\PYTHON\source\speechrecognition>python speechrecognition_google_windows.py
fmedia v1.6 <win-x86>
Recording... Source: float32 44100Hz mono. Press "s" to stop.
0:04 [.....] -40.00dB / -1.23dB
saved file mono.flac, 73 kbytes
good morning
fmedia v1.6 <win-x86>
Recording... Source: float32 44100Hz mono. Press "s" to stop.
0:04 [.....] -40.00dB / -25.06dB
saved file mono.flac, 60 kbytes
error
fmedia v1.6 <win-x86>
Recording... Source: float32 44100Hz mono. Press "s" to stop.
0:03 [.....] -40.00dB / -26.56dB
0:04 [.....] -40.00dB / -26.56dB
source\speechrecognition>
saved file mono.flac, 61 kbytes
```

aplikasi akan mengikuti ucapan Anda dalam bahasa inggris.

3.9.2.5. Text to Speech dengan gTTS

Untuk menguji coba text to speech dengan gtts ini siapkan file baru dengan nama gtts_sample.py :

```
#!/usr/bin/env python3
from gtts import gTTS
"""
gtts_sample.py
"""

import os
words = 'Saya lapar sekali'
tts = gTTS(text= words, lang='id', slow=True).save("tmp.mp3")
cmd = "vlc --intf dummy tmp.mp3 vlc://quit"
pipe = os.popen(cmd).read()
print (pipe)
```

Jalankan aplikasi di atas, jika berhasil maka akan terdengar suara “Saya lapar sekali”.

3.9.2.6. Speech Recognition Bahasa Indonesia

Untuk contoh lain, kita akan mencoba membuat speech recognition untuk bahasa indonesia dan text to speech dengan google api.

Buat aplikasi baru dengan nama speech_indo.py (pada linux)

```
#!/usr/bin/env python3
"""
speech_indo.py
for python training at jasaplus.com
"""

import speech_recognition as sr
import os
from gtts import gTTS
AUDIO_FILE = "tmp.flac"
r = sr.Recognizer()
while True:
    try:
        os.system("AUDIODEV=plughw:1,0 rec --channels=1 --bits=24 --rate=44100 tmp.flac trim 0 5")
        with sr.AudioFile(AUDIO_FILE) as source:
            r.adjust_for_ambient_noise(source)
            audio = r.record(source)
            res = r.recognize_google(audio, language='ID')
            print("\n" + res + "\n")
            tts = gTTS(text= res, lang='id', slow=True).save("tmp.mp3")
            cmd = "mpg123 tmp.mp3"
            pipe = os.popen(cmd).read()
            print (pipe)
    except Exception as e:
        print(e)
        pass
```

Untuk windows, aplikasi di atas memerlukan sedikit penyesuaian, buat aplikasi baru dengan nama speech_indo_windows.py

```
#!/usr/bin/env python3
"""
speech_indo_windows.py
for python training at jasaplus.com
"""

import speech_recognition as sr
import os
from gtts import gTTS
AUDIO_FILE = "mono.flac"
r = sr.Recognizer()
while True:
    try:
        os.system("del mono.flac")
        os.system("fmedia --dev-capture=2 --volume=98 --channels=mono --rate=16000 --record -o mono.flac --until=5")
        with sr.AudioFile(AUDIO_FILE) as source:
            r.adjust_for_ambient_noise(source)
            audio = r.record(source)
            res = r.recognize_google(audio, language='ID')
            print("\n" + res + "\n")
            tts = gTTS(text= res, lang='id', slow=True).save("tmp.mp3")
            cmd = "vlc --intf dummy tmp.mp3 vlc://quit"
            pipe = os.popen(cmd).read()
            print (pipe)
    except:
```

```
except Exception as e:  
    print(e)  
    pass
```

Jalankan aplikasi di atas, saat perekaman, coba berbicara bahasa indonesia, setelah proses recognize, komputer akan mengikuti suara Anda.

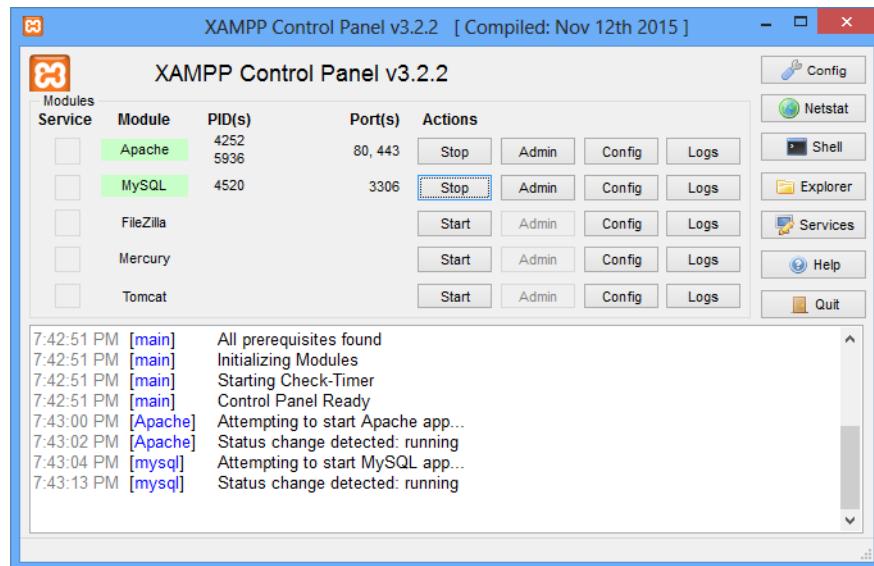
3.9.2.7. Speech Recognition dengan Python dan Java Script

Pada contoh ini, kita akan mencoba membuat aplikasi speech recognition dengan menggunakan selenium dan java script. Sebelum memulai pastikan xampp (apache dan mysql) berjalan dan pastikan google chrome sudah terinstall.

Pada terminal linux, start xampp:

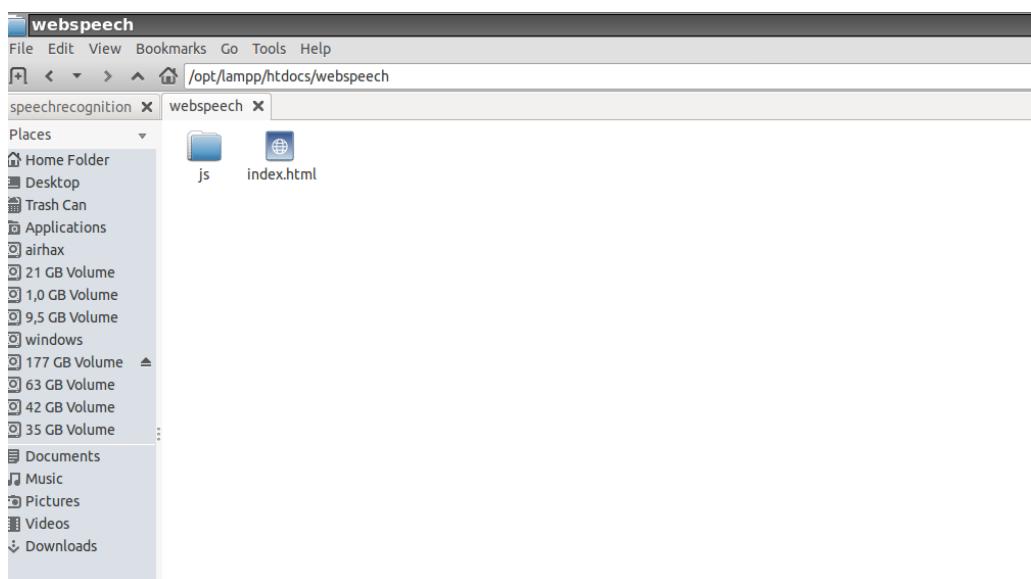
```
sudo /opt/lampp/.lampp start
```

Pada windows, jalankan dengan cara menjalankan xampp controller lalu start pada apache dan mysql

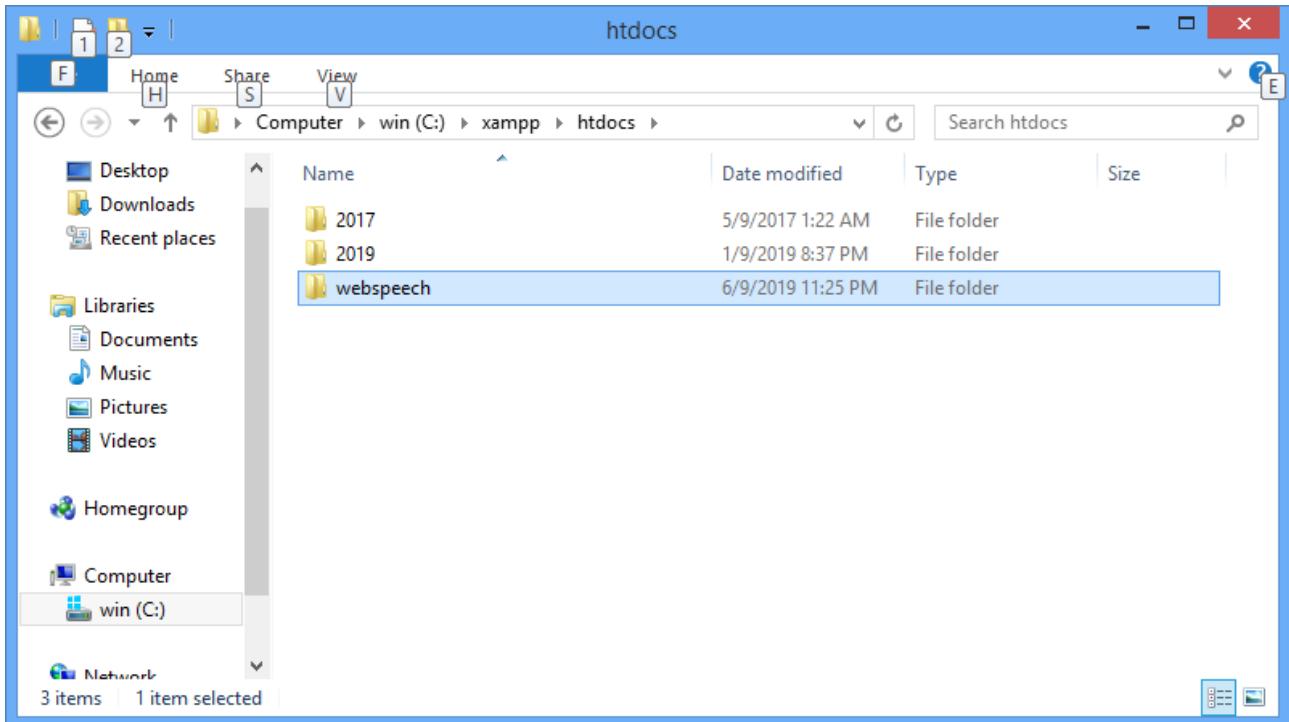


Source code webspeech bisa didownload di github repo untuk training ini (alamat github ada pada BAB 1).

Simpan webspeech pada direktori htdocs , jika menggunakan linux, simpan di /opt/lampp/htdocs



Pada windows, simpan webspeech di c:\xampp\htdocs



Berikut ini source code webspeech

index.html :

```
<!-- webspeech -->
<script src="js/jquery.min.js"></script>
<script src="js/script.js"></script>
<textarea id="data" cols=40 rows="6"></textarea>
<button id=startme OnClick="recognition.start();"></button>
```

script.js :

```
/*
webspeech
*/
try {
  var SpeechRecognition = window.SpeechRecognition || window.webkitSpeechRecognition;
  var recognition = new SpeechRecognition();
}
catch(e) {
  console.error(e);
  alert("browser not supported or no mic found !");
}
recognition.continuous = true;
recognition.onresult = function(event) {
  var data = $('#data');
  var current = event.resultIndex;
  var transcript = event.results[current][0].transcript;
  console.log("transcript : " + transcript);
```

```

    data.val(transcript);
};

recognition.onerror = function(event) {
  if(event.error == 'no-speech') {
    console.log("error");
  };
}

recognition.start();

```

karena training ini bukan mengenai java script maka silahkan Anda baca baca sendiri di internet tentang webkitSpeechRecognition.

Script ini memerlukan jquery.min.js :

<https://code.jquery.com/jquery-3.4.1.min.js> simpan dengan nama jquery.min.js

Selanjutnya siapkan script python dengan selenium.

Jika menggunakan linux, siapkan script dengan nama : speech_selenium.py

```

#!/usr/bin/env python3

"""
speech_selenium.py
Example
for python training at www.jasaplus.com
"""

from selenium import webdriver
from gtts import gTTS
import time, os

data = ""
data_prev = ""

def speech(data):
    try:
        tts = gTTS(text=data, lang='en', slow=True).save("tmp.mp3")
        cmd = "mpg321 tmp.mp3"
        pipe = os.popen(cmd)
    except:
        raise

def setopt():
    opt = webdriver.ChromeOptions()
    opt.add_experimental_option("prefs", { \
        "profile.default_content_setting_values.media_stream_mic": 1,
        "profile.default_content_setting_values.notifications": 1
    })
    return opt

opti = setopt()
browser = webdriver.Chrome(chrome_options = opti)
browser.get("http://localhost/webspeech/")
browser.set_window_position(800, 370)
browser.set_window_size(520, 350)
count = 0

```

```

while True:
    data = browser.find_element_by_id("data").get_attribute("value");
    if data != "None":
        data = data.strip()
        if (data != data_prev) and (len(data) > 0):
            data_prev = data
            print(data)
            speech(data)
            count = 0
    time.sleep(1)
    count+=1
    if (count > 15):
        startme = browser.find_element_by_id("startme")
        if startme != None:
            startme.click()

```

Jika menggunakan windows, siapkan script dengan nama : speech_selenium_windows.py

```

#!/usr/bin/env python3

"""
speech_selenium_windows.py
Example
for python training at www.jasaplus.com
"""

from selenium import webdriver
from gtts import gTTS
import time, os

data = ""
data_prev = ""

def speech(data):
    try:
        tts = gTTS(text=data, lang='en', slow=True).save("tmp.mp3")
        cmd = "vlc --intf dummy tmp.mp3 vlc://quit"
        pipe = os.popen(cmd)
    except:
        raise

def setopt():
    opt = webdriver.ChromeOptions()
    opt.add_experimental_option("prefs", { \
        "profile.default_content_setting_values.media_stream_mic": 1,
        "profile.default_content_setting_values.notifications": 1
    })
    return opt

opti = setopt()
browser = webdriver.Chrome(chrome_options = opti)
browser.get("http://localhost/webspeech/")
browser.set_window_position(800, 370)
browser.set_window_size(520, 350)
count = 0
while True:
    data = browser.find_element_by_id("data").get_attribute("value");

```

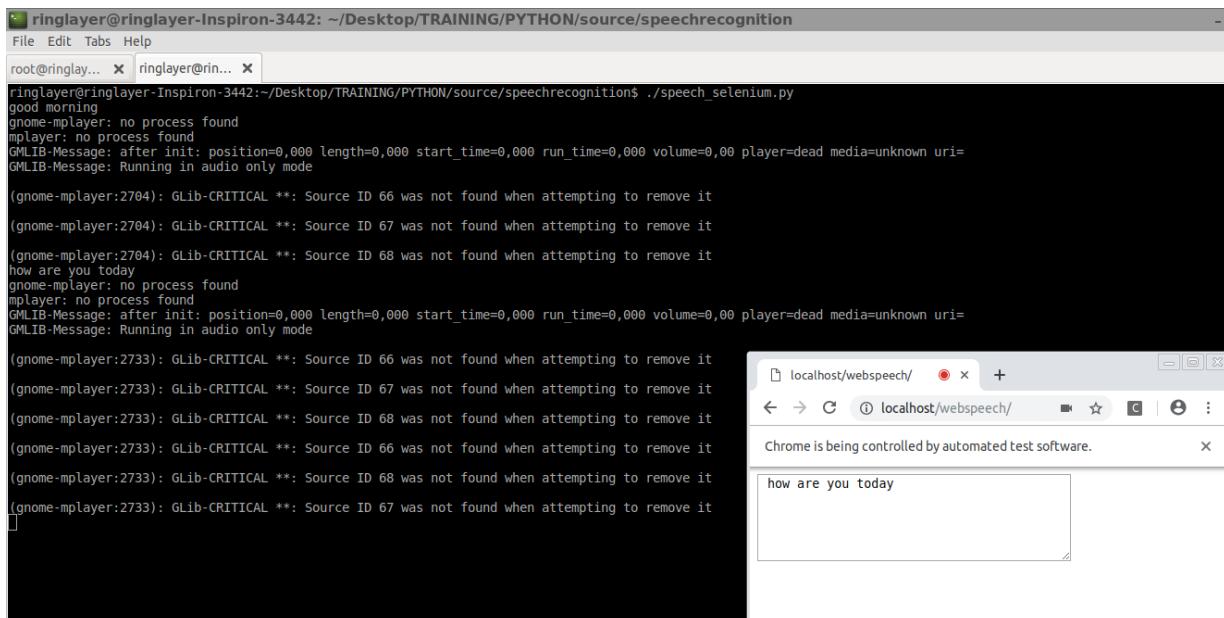
```

if data != "None":
    data = data.strip()
    if (data != data_prev) and (len(data) > 0):
        data_prev = data
        print(data)
        speech(data)
        count = 0
    time.sleep(1)
    count+=1
    if (count > 15):
        startme = browser.find_element_by_id("startme")
        if startme != None:
            startme.click()

```

Aplikasi di windows ini akan berjalan lebih lambat dibandingkan di linux, untuk hasil optimal bisa menggunakan linux ubuntu 16.04 atau lubuntu 16.04.

Jalankan script di atas maka saat google chrome berjalan, speech recognition sudah dimulai, silahkan diuji dengan menyebutkan kalimat berbahasa inggris dengan benar. Berikut ini adalah saat speech recognition berjalan



Penjelasan

```

from selenium import webdriver
from gtts import gTTS
import time, os

```

Pertama tama kita import dulu modul modul yang dibutuhkan

```

def speech(data):
    try:
        tts = gTTS(text=data, lang='en', slow=True).save("tmp.mp3")
        cmd = "mpg321 tmp.mp3"
        pipe = os.popen(cmd).read()

    except:

```

```
raise
```

Fungsi speech merupakan fungsi yang kita buat untuk menjalankan gTTS text to speech.

```
def setopt():
    opt = webdriver.ChromeOptions()
    opt.add_experimental_option("prefs", { \
        "profile.default_content_setting_values.media_stream_mic": 1,
        "profile.default_content_setting_values.notifications": 1
    })
    return opt
```

Kita set beberapa opsi untuk chrome agar microphone bisa diallow secara otomatis

```
opti = setopt()
browser = webdriver.Chrome(chrome_options = opti)
browser.get("http://localhost/webspeech/")
browser.set_window_position(800, 370)
browser.set_window_size(520, 350)
```

Rutin di atas akan menginisialisasi google chrome dan menjalankan webspeech yang tadi kita siapkan di localhost

```
count = 0
while True:
    data = browser.find_element_by_id("data").get_attribute("value")
    if data != "None":
        data = data.strip()
        if (data != data_prev) and (len(data) > 0):
            data_prev = data
            print(data)
            speech(data)
            count = 0
    time.sleep(1)
    count+=1
    if (count > 15):
        startme = browser.find_element_by_id("startme")
        if startme != None:
            startme.click()
```

Pada dasarnya script java script untuk speech recognition “webspeech” tadi akan berhenti otomatis jika kita idle selama 15 detik. Pada rutin di atas kita akali setelah idle selama 15 detik untuk menjalankan ulang speech recognition.

Untuk materi lanjutan speech recognition akan dibahas dalam kursus selanjutnya “Python for Machine Learning”