Tutorial matplotlib

Kholid Fuadi*

October 5, 2013

Contents

*http://twitter.com/sopier

1 Pendahuluan

1.1 Sekilas Tutorial

Tutorial ini nantinya akan berisi materi tentang modul `matplotlib` dengan sedikit ada penjelasan mengenai kode python disana disini.

Kholid Fuadi
Jogja, 29 September 2013.

1.2 Sekilas matplotlib

`matplotlib` adalah modul python untuk menggambar plot 2D dengan kualitas tinggi. `matplotlib` dapat digunakan dalam *script* python, *interpreter* python dan ipython, server, dan 6 GUI toolkit. `matplotlib` berusaha untuk membuat segalanya jadi mudah, dan yang tadinya seperti tidak menjadi mungkin untuk dilakukan. Dengan `matplotlib`, Anda dapat membuat *plots*, *histograms*, *spectra*, *bar charts*, *errorchards*, *scatterplots*, dan masih banyak lagi.

Pembuat `matplotlib` bernama **John D. Hunter** yang pada 28 Agustus 2012 lalu meninggal dunia setelah bergelut dengan komplikasi kanker yang diidap beliau. Jasa beliau untuk *Python Community* sungguh sangat luar biasa (khususnya python untuk *science*).


Gambar 1: John D. Hunter

Jika Anda merasa mendapatkan manfaat dari modul `matplotlib` yang sudah beliau buat, tidak ada salahnya untuk ikut melakukan kontribusi dengan melakukan donasi ke John Hunter Memorial Fund. Donasi ini nantinya akan diberikan langsung kepada keluarga yang sudah beliau tinggalkan, Miriam (istri), Clara, Ava dan Rahel (anak).

1.3 Pemasangan Modul matplotlib

Jika Anda menggunakan sistem operasi `ubuntu`, Anda dapat memasang modul `matplotlib` dengan:

```
$ sudo apt-get install python-matplotlib
```

Karena tutorial ini nantinya akan banyak bersinggungan dengan modul `numpy` dan `pandas`, maka silakan dipasang juga kedua modul tersebut dengan:

```
$ sudo apt-get install python-numpy
$ sudo apt-get install python-pandas
$ sudo apt-get install python-scipy
```

2 Membuat Plot

Bagian ini nanti akan berisi contoh kode untuk menggambar plot dengan `matplotlib`.

2.1 Dasar-dasar Plot

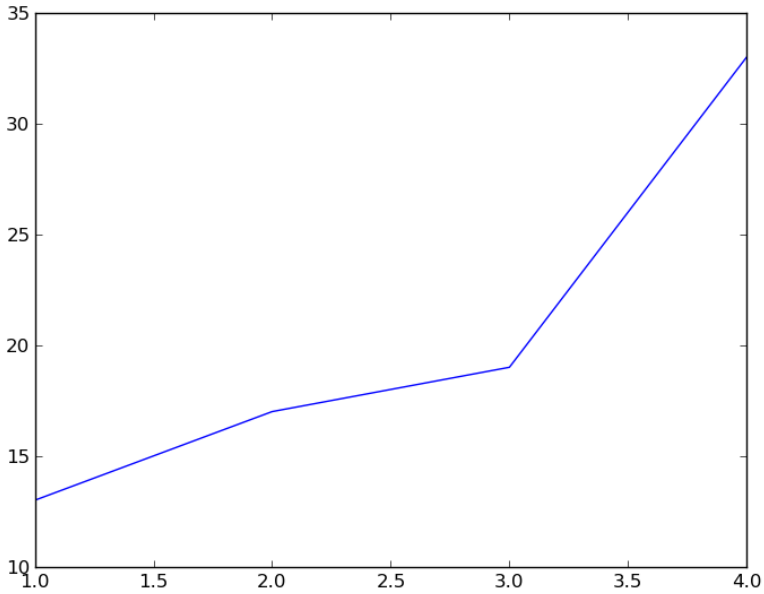
Sesuai dengan namannya, fungsi `plot` berguna untuk menggambar garis atau penanda pada bidang gambar. Mari kita lihat dokumentasi untuk fungsi ini:

```
>>> import matplotlib.pyplot as plt
>>> help(plt.plot)
Help on function plot in module matplotlib.pyplot:

plot(*args, **kwargs)
    Plot lines and/or markers to the
    :class:‘~matplotlib.axes.Axes‘.  *args* is a variable length
    argument, allowing for multiple *x*, *y* pairs with an
    optional format string.  For example, each of the following is
    legal::

        plot(x, y)         # plot x and y using default line style and color
        plot(x, y, 'bo')   # plot x and y using blue circle markers
        plot(y)            # plot y using x as index array 0..N-1
        plot(y, 'r+')      # ditto, but with red plusses

....
```

Contoh menggambar plot:

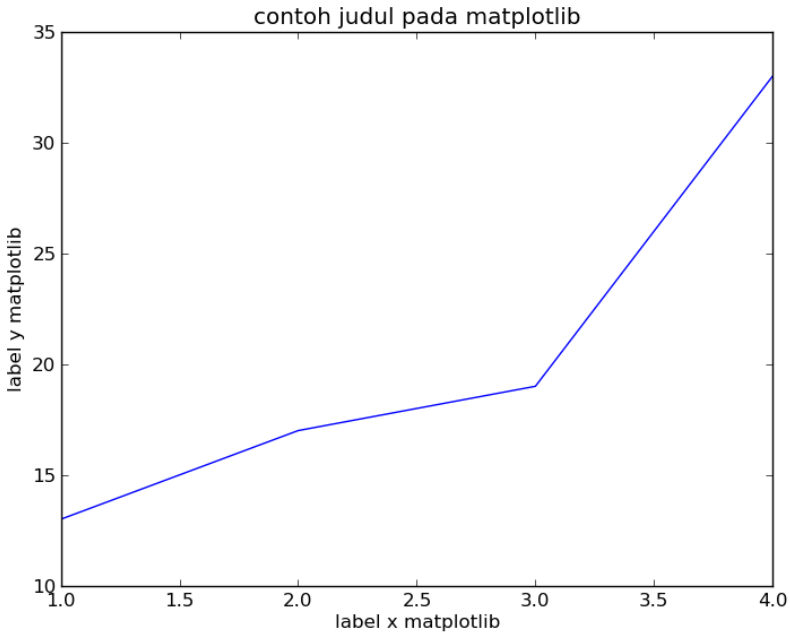
```
>>> import matplotlib.pyplot as plt
>>> x = [1, 2, 3, 4]
>>> y = [13, 17, 19, 33]
>>> plt.plot(x, y)
[<matplotlib.lines.Line2D object at 0x94d1b4c>]
>>> plt.show()
```

Tampilan:



Anda dapat menambahkan judul, label untuk garis horisontal dan vertikal dengan memanggil fungsi `xtitle`, `xlabel`, dan `ylabel` seperti pada contoh berikut:

```
>>> import matplotlib.pyplot as plt
>>> plt.plot([1, 2, 3, 4], [13, 17, 19, 33])
[<matplotlib.lines.Line2D object at 0xa57a46c>]
>>> plt.title('contoh judul pada matplotlib')
<matplotlib.text.Text object at 0xa54fbec>
>>> plt.xlabel('label x matplotlib')
<matplotlib.text.Text object at 0xa54bc4c>
>>> plt.ylabel('label y matplotlib')
<matplotlib.text.Text object at 0xa5475ac>
>>> plt.show()
```



Tabel ringkasan fungsi:

```
|--------+----------------------------------|
| Fungsi | Keterangan                       |
|--------+----------------------------------|
| plot   | melakukan plot                   |
| title  | memberi judul pada gambar plot   |
| xlabel | memberi nama label untuk garis x |
| ylabel | memberi nama label untuk garis y |
| show   | menampilkan gambar plot          |
|--------+----------------------------------|
```

2.2 Membaca Data dari Berkas

Pada contoh di atas, kita menggunakan tipe data `list` yang dideklarasi langsung pada python. Bagaimana jika data kita berada pada sebuah berkas? Tentu data harus kita "baca" terlebih dahulu, sebelum ditampilkan dalam bentuk grafik.

2.2.1 Menggunakan Fungsi Read (Python File I/O)

Sebagai contoh, kita memiliki data berikut pada berkas `data.txt`:

```
1366671909,5
1366671914,6
1366671920,3
1366671937,7
1366671942,1
1366671947,8
```

```
1366671955,4
1366671976,5
1366671981,7
1366671986,3
```

Mari kita "baca" berkas tersebut:

```
>>> open('data.txt').read().strip()
'1366671909,5\n1366671914,6\n1366671920,3\n1366671937,7\n1366671942,1
\n1366671947,8\n1366671955,4\n1366671976,5\n1366671981,7\n1366671986,3'
```

Terlihat bahwa kita menggunakan 3 fungsi dari python untuk membaca berkas tersebut, mari kita baca penggalan dokumentasi dari masing-masing fungsi tersebut:

```
>>> help(open)
Help on built-in function open in module __builtin__:

open(...)
    open(name[, mode[, buffering]]) -> file object

    Open a file using the file() type, returns a file object.  This is the
    preferred way to open a file.  See file.__doc__ for further information.

>>> help(open('data.txt').read)
Help on built-in function read:

read(...)
    read([size]) -> read at most size bytes, returned as a string.

    If the size argument is negative or omitted, read until EOF is reached.
    Notice that when in non-blocking mode, less data than what was requested
    may be returned, even if no size parameter was given.

>>> help(open('data.txt').read().strip)
Help on built-in function strip:

strip(...)
    S.strip([chars]) -> string or unicode

    Return a copy of the string S with leading and trailing
    whitespace removed.
    If chars is given and not None, remove characters in chars instead.
    If chars is unicode, S will be converted to unicode before stripping
```

Dari dokumentasi, kita ketahui bahwa `open` berfungsi untuk membuka berkas dengan nilai kembalian berupa `file object`, `read` berfungsi untuk membaca `file object`, dengan nilai kembalian (*return*) berupa `string`, sedangkan `strip` berguna untuk menghilangkan tanda *whitespace* di depan dan dibelakang `string`.

Fungsi	Keterangan
open	membuka berkas
read	membaca file object
strip	menghilangkan whitespace di depan atau belakang string

Berikut urutan perintah yang kita jalankan untuk membaca dan menampilkan grafik plot dari data tersebut:

```
>>> with open('data.txt') as f:
...     l = f.read().strip().split('\n')
...
>>> l
['1366671909,5', '1366671914,6', '1366671920,3', '1366671937,7',
'1366671942,1', '1366671947,8', '1366671955,4', '1366671976,5',
'1366671981,7', '1366671986,3']
>>> x = []
>>> y = []
>>> for i in l:
...     xny = i.split(',')
...     x.append(int(xny[0]))
...     y.append(int(xny[1]))
...
>>> x
[1366671909, 1366671914, 1366671920, 1366671937, 1366671942,
1366671947, 1366671955, 1366671976, 1366671981, 1366671986]
>>> y
[5, 6, 3, 7, 1, 8, 4, 5, 7, 3]
```

Pada kode di atas, kita menggunakan satu fungsi baru lagi, yakni `split`, mari kita lihat dokumentasi:

```
>>> help('aku, suka, python'.split)
Help on built-in function split:

split(...)
    S.split([sep [,maxsplit]]) -> list of strings

    Return a list of the words in the string S, using sep as the
    delimiter string.  If maxsplit is given, at most maxsplit
    splits are done. If sep is not specified or is None, any
    whitespace string is a separator and empty strings are removed
    from the result.
```

Fungsi `split` berguna untuk memecah `string` menjadi `list` yang berisi `string`.

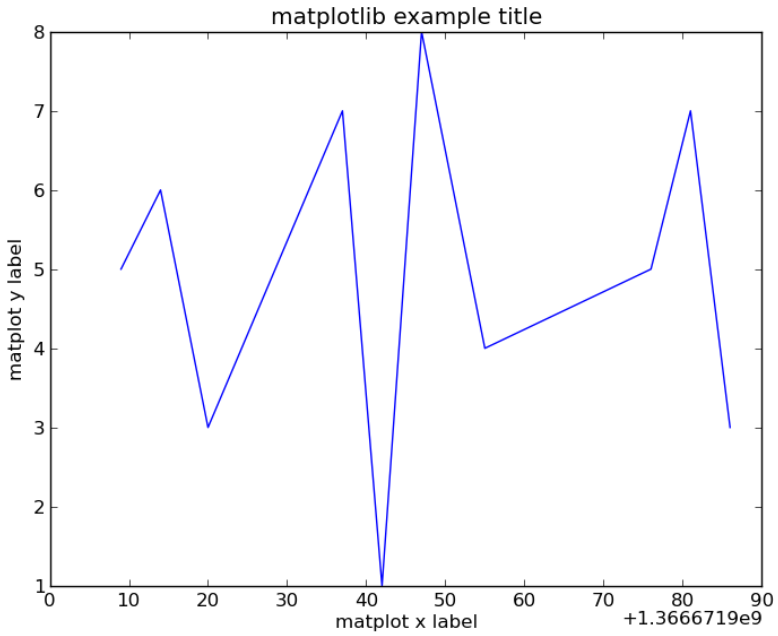
Fungsi	Keterangan
split	memecah string menjadi list of string

Setelah mendapatkan nilai `x` dan `y`, sekarang saatnya menampilkan data tersebut ke dalam grafik plot:

```
>>> plt.plot(x, y)
[<matplotlib.lines.Line2D object at 0xa7d32ec>]
>>> plt.title('matplotlib example title')
<matplotlib.text.Text object at 0xae946cc>
>>> plt.xlabel('matplot x label')
```

```
<matplotlib.text.Text object at 0xa5cf6ec>
>>> plt.ylabel('matplot y label')
<matplotlib.text.Text object at 0xa5e3e0c>
>>> plt.show()
```

Tampilan:



2.2.2 Menggunakan Modul numpy

Selain menggunakan cara di atas, kita juga dapat memanfaatkan salah satu fungsi dari modul `numpy`, yakni `loadtxt` yang sangat *powerful* untuk urusan membaca input dari berkas.

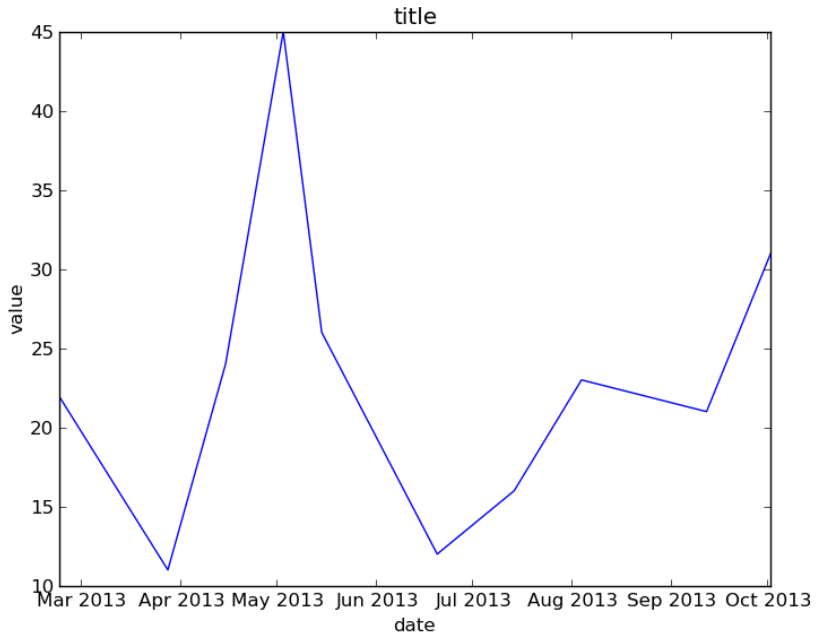
Sebagai contoh kita memiliki berkas `data.csv` dengan isi sebagai berikut:

```
2013-02-22,22
2013-03-28,11
2013-04-15,24
2013-05-03,45
2013-05-15,26
2013-06-20,12
2013-07-14,16
2013-08-04,23
2013-09-12,21
2013-10-02,31
```

Mari kita baca dan impor data tersebut ke dalam python menggunakan fungsi `loadtxt`. Ketik baris kode berikut dan simpan ke dalam berkas `matplot_5.py`

```
#!/usr/bin/python
# matplot_5.py

import numpy as np
import matplotlib.pyplot as plt
import matplotlib.dates as mdates


def graph():
    date, value = np.loadtxt('data.csv', delimiter=',', unpack=True,
                             converters = {0: mdates.strpdate2num('%Y-%m-%d')}
                             )
    fig = plt.figure()
    ax1 = fig.add_subplot(1,1,1, axisbg='white')
    ax1.plot_date(x=date,y=value,fmt='-')
    ax1.set_title('title')
    ax1.set_ylabel('value')
    ax1.set_xlabel('date')
    plt.show()

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

Jalankan dan berikut hasilnya:



Sebenarnya penggunaan fungsi `graph` di atas kurang begitu pas, karena terlalu statis, seharusnya sebuah fungsi harus bersifat dinamis. Apabila Anda ingin lebih dinamis, kita dapat ubah fungsi di atas menjadi:

```
#!/usr/bin/python
```

```
# matplot_5_din.py

import numpy as np
import matplotlib.pyplot as plt
import matplotlib.dates as mdates


def graph(thefile, delimiter, title, ylabel, xlabel):
    date, value = np.loadtxt(thefile, delimiter=delimiter, unpack=True,
                             converters = {0: mdates.strpdate2num('%Y-%m-%d')}
                             )
    fig = plt.figure()
    ax1 = fig.add_subplot(1,1,1, axisbg='white')
    ax1.plot_date(x=date,y=value,fmt='-')
    ax1.set_title(title)
    ax1.set_ylabel(ylabel)
    ax1.set_xlabel(xlabel)
    plt.show()

if __name__ == '__main__':
    graph('data.csv', ',', 'title', 'value', 'date')
```

Terlihat kita sudah berhasil melakukan *abstraksi* terhadap fungsi yang kita buat, sehingga sekarang kita dapat memanggil fungsi tersebut secara berulang-ulang hanya dengan mengganti *parameter* yang tersedia.

Namun perlu diingat, bahwa fungsi `graph()` di atas hanya berlaku untuk data `csv` dengan isi 2 kolom saja, jika Anda ingin lebih fleksibel lagi, Anda dapat melakukan *abstraksi* lagi sesuai keinginan.

Fungsi	Keterangan
plot_date	Membuat plot dari data yang mengandung date (x, atau y, atau keduanya)
set_title	memberi judul figure
set_ylabel	memberi label y
set_xlabel	memberi label x

2.3 Kustomisasi Tampilan Grafik Plot

Sekarang saatnya belajar membuat tampilan *custom* dari grafik kita.

2.3.1 Atribut Background, Foreground dan Garis

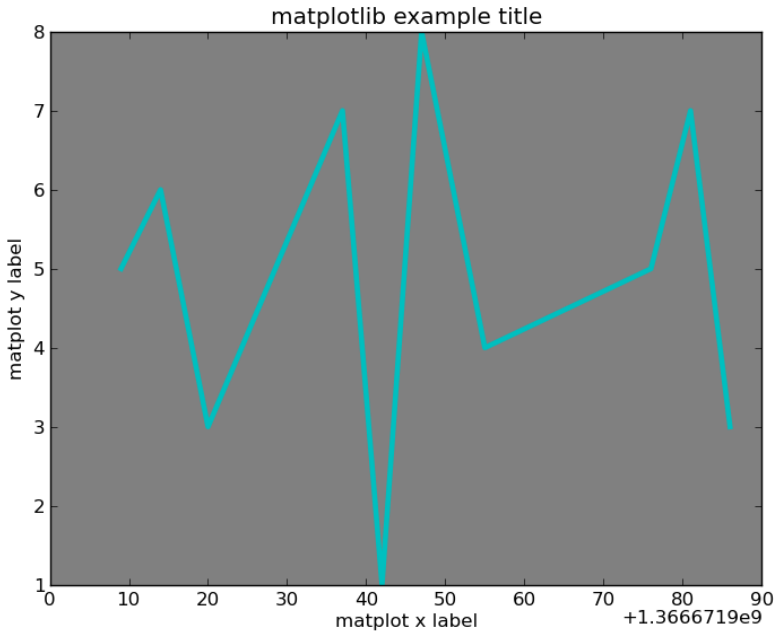
Agar lebih mudah dipahami, mari kita simpan kode-kode di atas ke dalam sebuah berkas dan kemudian menambahkan beberapa fungsi baru untuk mengubah tampilan dari grafik yang kita buat. Simpan kode berikut ke dalam berkas `matplot_1.py`

```
#!/usr/bin/python

import matplotlib.pyplot as plt
```

```
with open('data.txt', 'r') as f:
    l = f.read().strip().split('\n')

x = []
y = []

for i in l:
    xny = i.split(',')
    x.append(int(xny[0]))
    y.append(int(xny[1]))

# sebelum di plot, mari kita ubah tampilan
fig = plt.figure()
rect = fig.patch
rect.set_facecolor('#31312e')

ax1 = fig.add_subplot(1, 1, 1, axisbg='grey')
ax1.plot(x, y, 'c', linewidth=3.3)

# plot dimulai
ax1.set_title('matplotlib example title')
ax1.set_xlabel('matplot x label')
ax1.set_ylabel('matplot y label')
plt.show()
```

Jalankan *script* dengan menulis perintah berikut pada `Terminal`:

```
$ python matplot_1.py
```



Terlihat warna dan ketebalan garis sekarang sudah berubah. Perlu diingat bahwa warna `facecolor` tidak bisa ikut ditampilkan, tapi kalau Anda menjalankan dari `Terminal`, pasti Anda bisa melihat warna di sekitar grafik berubah menjadi abu-abu gelap.

Seperti biasa, jika Anda masih bingung dengan fungsi, baik itu *built-in* maupun *3rd-party*, Anda dapat memanggil dokumentasi menggunakan fungsi `help` seperti pada contoh sebelumnya.

Fungsi	Keterangan
int	mengubah string menjadi integer
figure	mengakses figure
patch	menyunting setting figure
set_facecolor	mengubah warna di sekitar grafik, di luar kotak grafik
add_subplot	mengakses subplot tertentu

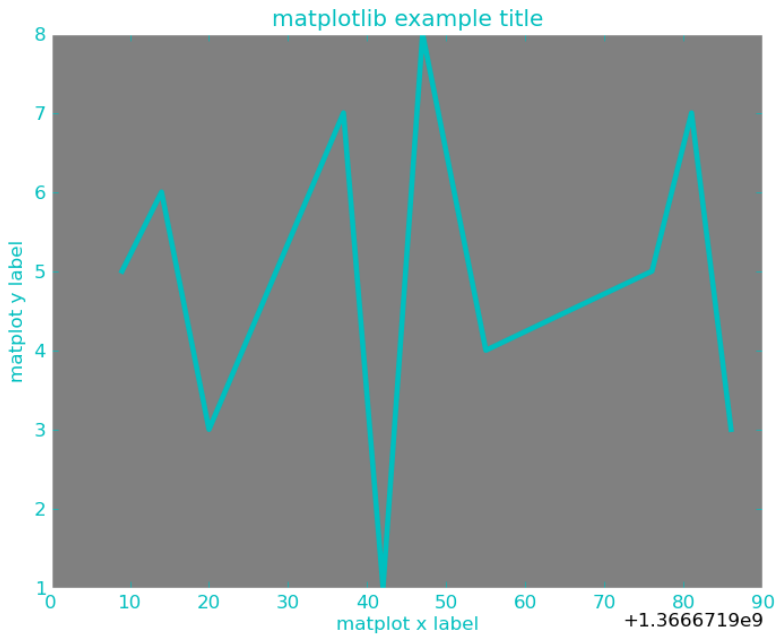
2.3.2 Atribut Warna

Mari kita salin kode pada berkas `matplot_1.py`, dan sunting menjadi seperti berikut:

```
#!/usr/bin/python

import matplotlib.pyplot as plt

with open('data.txt', 'r') as f:
    l = f.read().strip().split('\n')

x = []
y = []

for i in l:
    xny = i.split(',')
    x.append(int(xny[0]))
    y.append(int(xny[1]))

# sebelum di plot, mari kita ubah tampilan
fig = plt.figure()
rect = fig.patch
rect.set_facecolor('#31312e')

ax1 = fig.add_subplot(1, 1, 1, axisbg='grey')
ax1.plot(x, y, 'c', linewidth=3.3)

# customize start
ax1.tick_params(axis='x', color='c')
ax1.tick_params(axis='y', color='c')

ax1.spines['bottom'].set_color('w')
ax1.spines['top'].set_color('w')
ax1.spines['left'].set_color('w')
ax1.spines['right'].set_color('w')
```

```
ax1.yaxis.label.set_color('c')
ax1.xaxis.label.set_color('c')
# customize end

# plot dimulai
ax1.set_title('matplotlib example title')
ax1.set_xlabel('matplot x label')
ax1.set_ylabel('matplot y label')
plt.show()
```

Simpan ke dalam berkas `matplot_2.py` dan jalankan. Berikut hasil tampilan layarnya:



Fungsi	Keterangan
tick_params	mengubah warna huruf tick
spines	mengubah warna border grafik
label.set_color	mengubah warna huruf pada label

2.3.3 Multiple Graph Same Figure

Contoh berikut ini menunjukkan bagaimana kita dapat membuat 2 atau lebih grafik dalam satu *figure*. Sebelumnya, mari kita buat berkas baru, beri nama `data2.txt` dan isikan data berikut:

```
1366671909,2
```

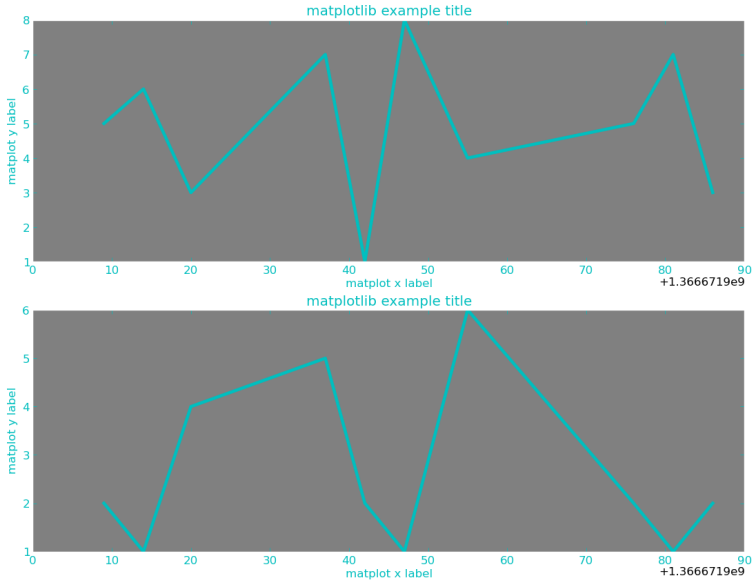
```
1366671914,1
1366671920,4
1366671937,5
1366671942,2
1366671947,1
1366671955,6
1366671976,2
1366671981,1
1366671986,2
```

Kemudian buat berkas python baru, misal `matplot_3.py`, dan isikan baris kode berikut:

```
#!/usr/bin/python
import matplotlib.pyplot as plt

# baca berkas data.txt
with open('data.txt', 'r') as f:
    l = f.read().strip().split('\n')

x = []
y = []

# baca berkas data2.txt
with open('data2.txt', 'r') as f2:
    l2 = f2.read().strip().split('\n')

x2 = []
y2 = []

for i in l:
    xny = i.split(',')
    x.append(int(xny[0]))
    y.append(int(xny[1]))

for i in l2:
    xny2 = i.split(',')
    x2.append(int(xny2[0]))
    y2.append(int(xny2[1]))


# sebelum di plot, mari kita ubah tampilan
fig = plt.figure()
rect = fig.patch
rect.set_facecolor('#31312e')

# set atribut untuk graph1
ax1 = fig.add_subplot(2, 1, 1, axisbg='grey')
ax1.plot(x, y, 'c', linewidth=3.3)

# customize start

# mengubah warna angka pada x dan y
ax1.tick_params(axis='x', colors='c')
ax1.tick_params(axis='y', colors='c')

# mengubah border tepi grafik
```

```
ax1.spines['bottom'].set_color('w')
ax1.spines['top'].set_color('w')
ax1.spines['left'].set_color('w')
ax1.spines['right'].set_color('w')

# mengubah warna label x dan y
ax1.yaxis.label.set_color('c')
ax1.xaxis.label.set_color('c')

ax1.set_title('matplotlib example title', color='c')
ax1.set_xlabel('matplot x label')
ax1.set_ylabel('matplot y label')

# customize end

# set atribut untuk graph2
ax2 = fig.add_subplot(2, 1, 2, axisbg='grey') # height x width and the chart #
ax2.plot(x2, y2, 'c', linewidth=3.3)

# customize start

# mengubah warna angka pada x dan y
ax2.tick_params(axis='x', colors='c')
ax2.tick_params(axis='y', colors='c')

# mengubah border tepi grafik
ax2.spines['bottom'].set_color('w')
ax2.spines['top'].set_color('w')
ax2.spines['left'].set_color('w')
ax2.spines['right'].set_color('w')

# mengubah warna label x dan y
ax2.yaxis.label.set_color('c')
ax2.xaxis.label.set_color('c')
# customize end

# plot dimulai
ax2.set_title('matplotlib example title', color='c')
ax2.set_xlabel('matplot x label')
ax2.set_ylabel('matplot y label')

# show the graph!
plt.show()
```

Dan berikut tampilannya:



Tabel *parameter* dari fungsi `add_subplot`:

Parameter	Keterangan
2, 1, 1	h=2, w=1, chart number 1
2, 1, 2	h=2, w=1, chart #2

Yang agak membingungkan di sini adalah 3 parameter pertama dari fungsi `sub_plot`. Mari kita buat satu lagi grafik agar lebih mudah dalam memahaminya.

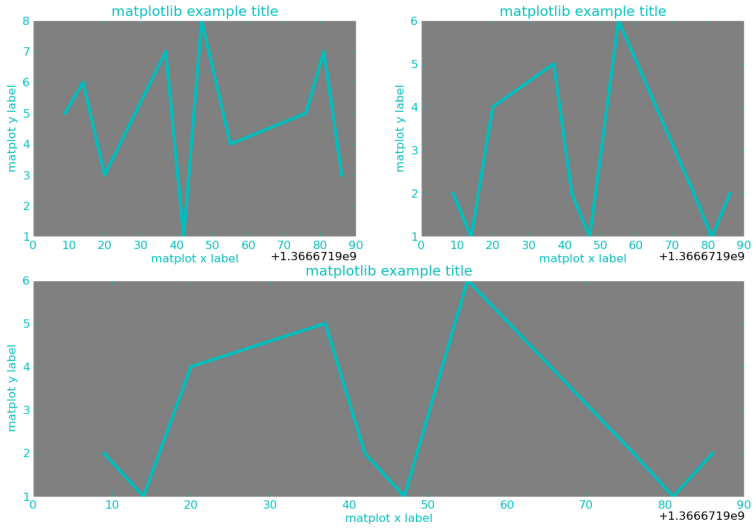
Buat berkas baru, misal `matplot_4.py`, dan isikan baris berikut:

```
#!/usr/bin/python
import matplotlib.pyplot as plt

# baca berkas data.txt
with open('data.txt', 'r') as f:
    l = f.read().strip().split('\n')

x = []
y = []

# baca berkas data2.txt
with open('data2.txt', 'r') as f2:
    l2 = f2.read().strip().split('\n')

x2 = []
y2 = []
```

```
for i in l:
    xny = i.split(',')
    x.append(int(xny[0]))
    y.append(int(xny[1]))

for i in l2:
    xny2 = i.split(',')
    x2.append(int(xny2[0]))
    y2.append(int(xny2[1]))


# sebelum di plot, mari kita ubah tampilan
fig = plt.figure()
rect = fig.patch
rect.set_facecolor('#31312e')

# set atribut untuk graph1
ax1 = fig.add_subplot(2, 2, 1, axisbg='grey')
ax1.plot(x, y, 'c', linewidth=3.3)

# customize start

# mengubah warna angka pada x dan y
ax1.tick_params(axis='x', colors='c')
ax1.tick_params(axis='y', colors='c')

# mengubah border tepi grafik
ax1.spines['bottom'].set_color('w')
ax1.spines['top'].set_color('w')
ax1.spines['left'].set_color('w')
ax1.spines['right'].set_color('w')

# mengubah warna label x dan y
ax1.yaxis.label.set_color('c')
ax1.xaxis.label.set_color('c')

ax1.set_title('matplotlib example title', color='c')
ax1.set_xlabel('matplot x label')
ax1.set_ylabel('matplot y label')

# customize end

# set atribut untuk graph2
ax2 = fig.add_subplot(2, 2, 2, axisbg='grey') # 2x2 grid for the chart number 2
ax2.plot(x2, y2, 'c', linewidth=3.3)

# customize start

# mengubah warna angka pada x dan y
ax2.tick_params(axis='x', colors='c')
ax2.tick_params(axis='y', colors='c')

# mengubah border tepi grafik
ax2.spines['bottom'].set_color('w')
ax2.spines['top'].set_color('w')
ax2.spines['left'].set_color('w')
```

```
ax2.spines['right'].set_color('w')

# mengubah warna label x dan y
ax2.yaxis.label.set_color('c')
ax2.xaxis.label.set_color('c')
# customize end

# plot dimulai
ax2.set_title('matplotlib example title', color='c')
ax2.set_xlabel('matplot x label')
ax2.set_ylabel('matplot y label')

# set atribut untuk graph3
ax2 = fig.add_subplot(2, 1, 2, axisbg='grey') # height x width and the chart #
ax2.plot(x2, y2, 'c', linewidth=3.3)

# customize start

# mengubah warna angka pada x dan y
ax2.tick_params(axis='x', colors='c')
ax2.tick_params(axis='y', colors='c')

# mengubah border tepi grafik
ax2.spines['bottom'].set_color('w')
ax2.spines['top'].set_color('w')
ax2.spines['left'].set_color('w')
ax2.spines['right'].set_color('w')

# mengubah warna label x dan y
ax2.yaxis.label.set_color('c')
ax2.xaxis.label.set_color('c')
# customize end

# plot dimulai
ax2.set_title('matplotlib example title', color='c')
ax2.set_xlabel('matplot x label')
ax2.set_ylabel('matplot y label')

plt.show()
```

Dan berikut tampilannya:

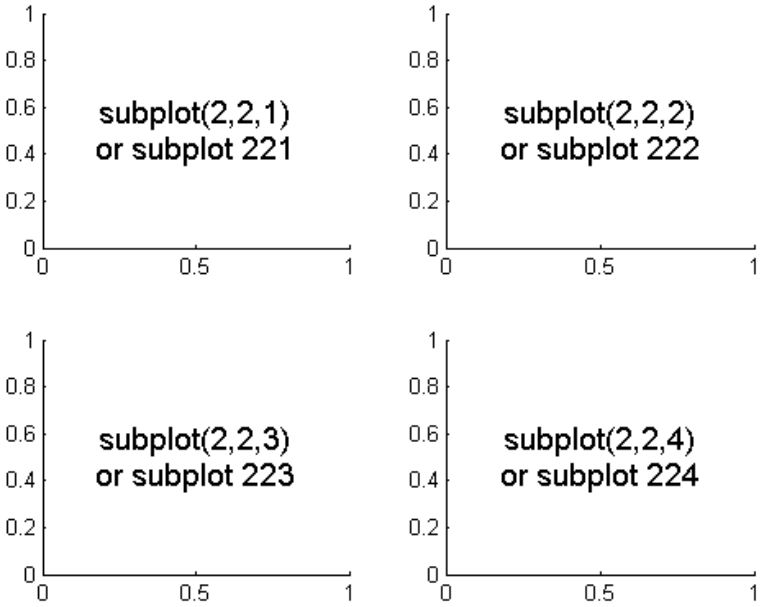


Berikut rangkuman *paremeter* dari fungsi `add_subplot` beserta keterangannya:

Parameter	Keterangan
2, 2, 1	h=2, w=2, chart number 1
2, 2, 2	h=2, w=2, chart #2
2, 1, 2	h=2, w=1, chart #2

Untuk dapat lebih menjelaskan mengenai `add_subplot`, mari kita lihat gambar berikut:[1]

[1] sumber: http://i.stack.imgur.com/AEGXG.png

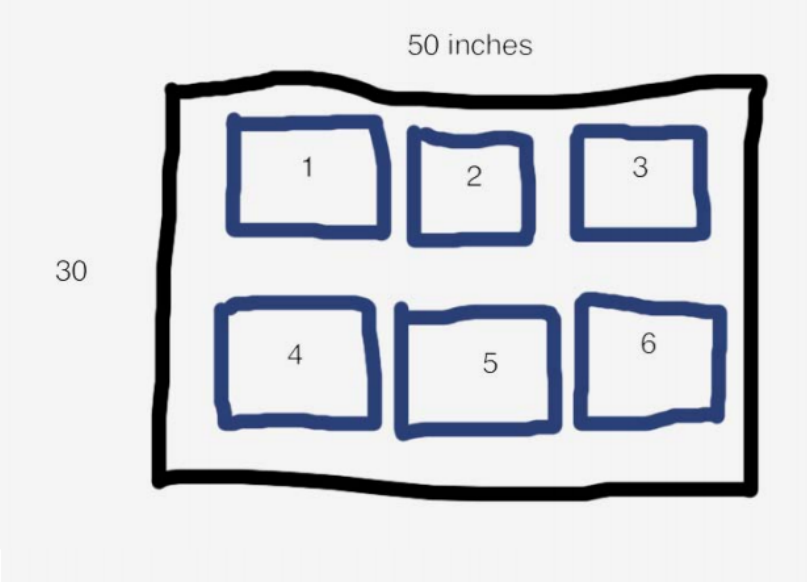


Kode python untuk membuat grafik di atas:

```
fig1.add_subplot(221)   #top left
fig2.add_subplot(222)   #top right
fig3.add_subplot(223)   #bottom left
fig4.add_subplot(224)   #bottom right
```

Dan berikut satu gambar lagi:[2]

[2]sumber: http://bit.ly/15Fcspq



Kode python untuk membuat grafik di atas:

```
fig1.add_subplot(221)   #top left
fig2.add_subplot(222)   #top center
fig3.add_subplot(223)   #top right
fig4.add_subplot(224)   #bottom left
fig5.add_subplot(225)   #bottom center
fig6.add_subplot(226)   #bottom right
```

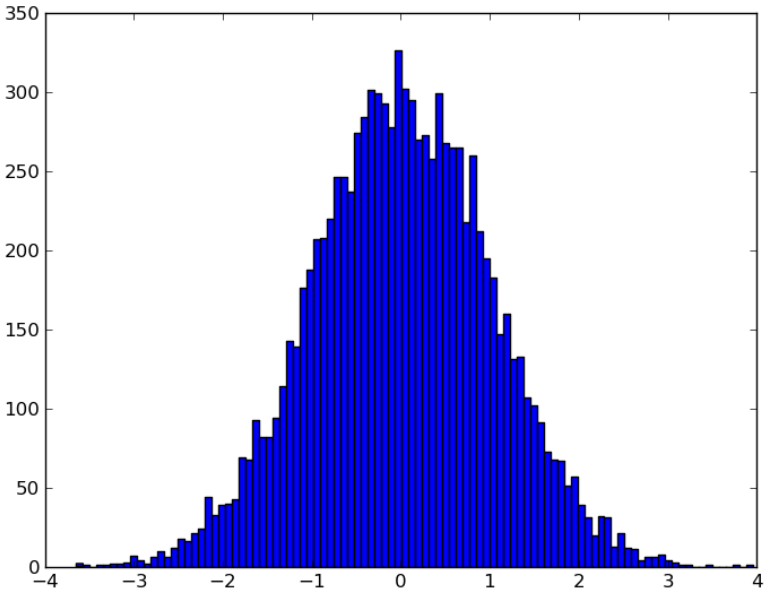
2.4 Histogram

Berikut contoh menggambar *histogram*:

```
>>> import matplotlib.pyplot as plt
>>> import numpy as np
>>> x = np.random.randn(10000)
>>> plt.hist(x, 100)
(array([  2,   1,   0,   1,   1,   2,   2,   3,   7,   4,   2,   6,  10,
         6,  12,  18,  16,  21,  24,  44,  33,  39,  40,  43,  69,  68,
        93,  82,  82,  94, 114, 143, 139, 176, 188, 207, 208, 220, 246,
       246, 237, 274, 284, 301, 299, 293, 278, 326, 302, 295, 270, 273,
       258, 299, 268, 265, 265, 218, 260, 212, 195, 183, 147, 160, 131,
       133, 107, 102,  91,  73,  68,  67,  51,  57,  39,  31,  20,  32,
        31,  13,  21,  12,  11,   4,   6,   6,   8,   4,   3,   1,   1,
         0,   0,   1,   0,   0,   0,   1,   0,   1]), array([
       -3.64645879, -3.57037555, -3.49429231, -3.41820907, -3.34212584,
       -3.2660426 , -3.18995936, -3.11387613, -3.03779289, -2.96170965,
       -2.88562641, -2.80954318, -2.73345994, -2.6573767 , -2.58129347,
       -2.50521023, -2.42912699, -2.35304376, -2.27696052, -2.20087728,
       -2.12479404, -2.04871081, -1.97262757, -1.89654433, -1.8204611 ,
       -1.74437786, -1.66829462, -1.59221138, -1.51612815, -1.44004491,
       -1.36396167, -1.28787844, -1.2117952 , -1.13571196, -1.05962873,
       -0.98354549, -0.90746225, -0.83137901, -0.75529578, -0.67921254,
```

```
       -0.6031293 , -0.52704607, -0.45096283, -0.37487959, -0.29879635,
       -0.22271312, -0.14662988, -0.07054664,  0.00553659,  0.08161983,
        0.15770307,  0.2337863 ,  0.30986954,  0.38595278,  0.46203602,
        0.53811925,  0.61420249,  0.69028573,  0.76636896,  0.8424522 ,
        0.91853544,  0.99461867,  1.07070191,  1.14678515,  1.22286839,
        1.29895162,  1.37503486,  1.4511181 ,  1.52720133,  1.60328457,
        1.67936781,  1.75545105,  1.83153428,  1.90761752,  1.98370076,
        2.05978399,  2.13586723,  2.21195047,  2.2880337 ,  2.36411694,
        2.44020018,  2.51628342,  2.59236665,  2.66844989,  2.74453313,
        2.82061636,  2.8966996 ,  2.97278284,  3.04886608,  3.12494931,
        3.20103255,  3.27711579,  3.35319902,  3.42928226,  3.5053655 ,
        3.58144873,  3.65753197,  3.73361521,  3.80969845,  3.88578168,
        3.96186492]), <a list of 100 Patch objects>)
>>> plt.show()
```

Dan berikut hasilnya:

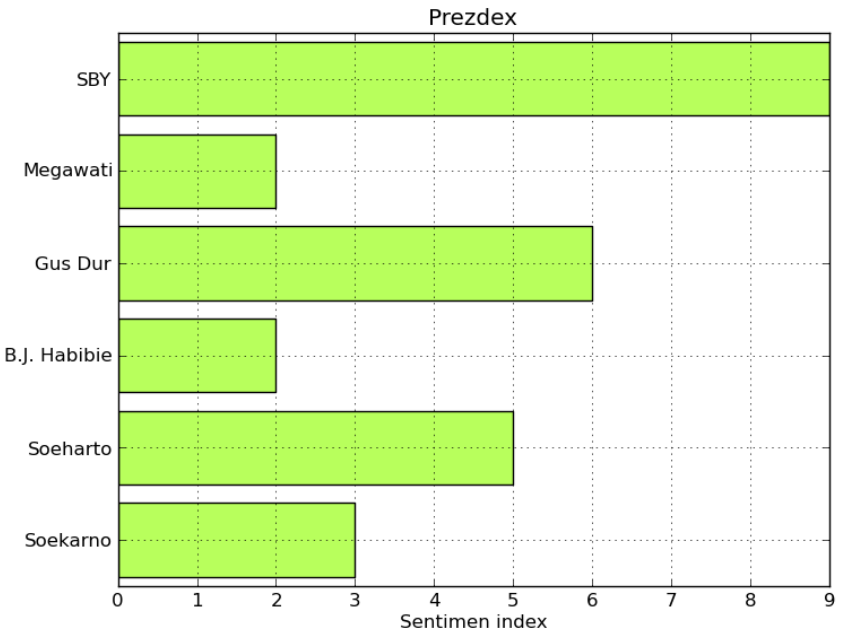


2.5 Bar Charts

Bagian ini akan membahas tentang *bar charts*. Berikut ini contoh sederhananya:

```
>>> from pylab import *
>>> pos = arange(6) + .5
>>> barh(pos, (3, 5, 2, 1, 2, 9), align='center', color='#b8ff5c')
<Container object of 6 artists>
>>> yticks(pos, ('Soekarno', 'Soeharto', 'B.J. Habibie', 'Gus Dur', 'Megawati', 'SBY'))
([<matplotlib.axis.YTick object at 0xac7396c>, .. list of 6 Text yticklabel objects>)
>>> xlabel('Sentimen index')
<matplotlib.text.Text object at 0xac7356c>
```

```
>>> ylabel('candidate')
<matplotlib.text.Text object at 0xac774ec>
>>> title('Prezdex')
<matplotlib.text.Text object at 0xac0f98c>
>>> grid(True)
>>> show()
```



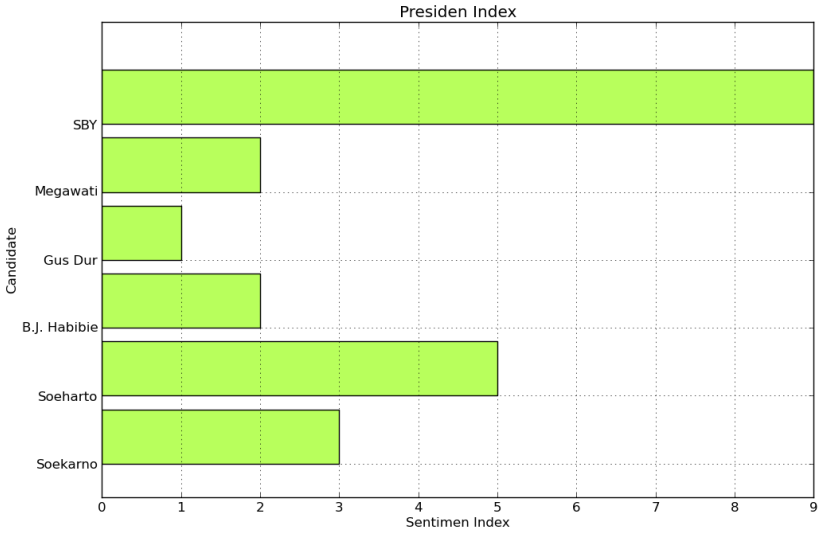
Fungsi	Keterangan
arange	membuat deret angka
barh	membuat bar horisontal
yticks	keterangan setiap titik pada y
grid	memunculkan garis bantu

`barh` sendiri menerima beberapa parameter seperti `height` (dalam contoh di atas `pos`), `width` (lebar bar), `align` (secara *default* nilainya *edge*, namun bisa kita ganti *center* seperti pada contoh di atas, dan `color` (warna dari bar).

Contoh satu lagi menggunakan data di atas, namun kali ini kita tidak akan menggunakan parameter `align` (menggunakan *default*).

```
>>> from pylab import *
>>> pos = arange(6) + .5
>>> barh(pos, (3, 5, 2, 1, 2, 9), color='#b8ff5c')
<Container object of 6 artists>
>>> yticks(pos, ('Soekarno', 'Soeharto', 'B.J. Habibie', 'Gus Dur', 'Megawati', 'SBY'))
([<matplotlib.axis.YTick object at 0xaa2f4cc> .. list of 6 Text yticklabel objects>)
```

```
>>> xlabel('Sentimen Index')
<matplotlib.text.Text object at 0xaa1eecc>
>>> ylabel('Candidate')
<matplotlib.text.Text object at 0xaa2f66c>
>>> title('Presiden Index')
<matplotlib.text.Text object at 0xacbde2c>
>>> grid(True)
>>> show()
```



Terlihat bahwa sekarang posisi bar menempel pada *edge* dari angka `0.5`.

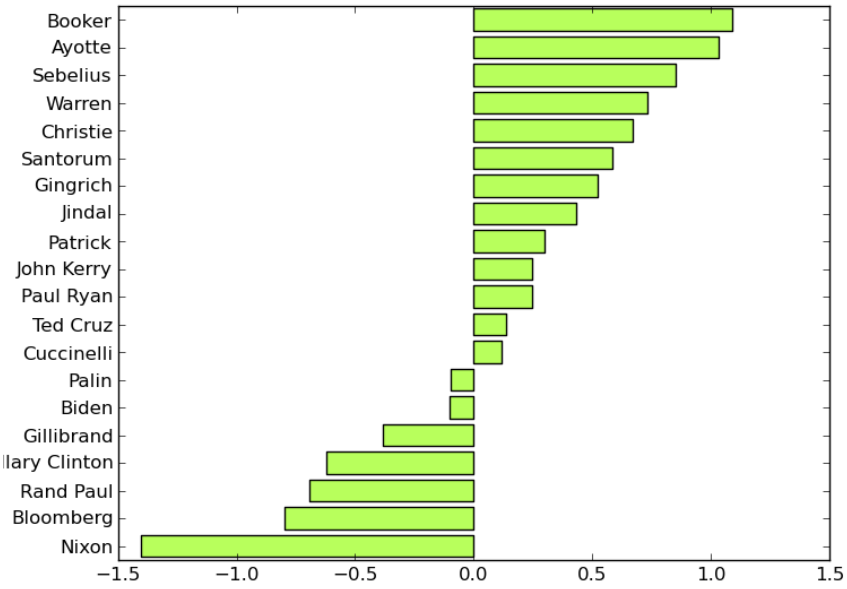
Contoh berikutnya, misalkan kita memiliki data dalam berkas `csv` seperti berikut:

```
Nixon;-1.4013
Bloomberg;-0.7981
Rand Paul;-0.6930
Hillary Clinton;-0.6216
Gillibrand;-0.3818
Biden;-0.0999
Palin;-0.0979
Cuccinelli;0.1184
Ted Cruz;0.1354
Paul Ryan;0.2464
John Kerry;0.2477
Patrick;0.2974
Jindal;0.4317
Gingrich;0.5241
Santorum;0.5854
Christie;0.6704
Warren;0.7332
Sebelius;0.8494
Ayotte;1.0317
Booker;1.0882
```

Mari kita olah dan tampilkan dalam bentuk bar horisontal:

```
>>> from pylab import *
>>> name = []
>>> value = []
>>> with open('presdex.csv') as f:
...     listline = f.read().split('\n')
...
>>> for line in listline:
...     split = line.split(';')
...     name.append(split[0])
...     value.append(float(split[1]))
...
>>> pos = arange(len(name)) + 0.5
>>> barh(pos, value, align='center', color='#b8ff5c')
<Container object of 20 artists>
>>> yticks(pos, name)
([<matplotlib.axis.YTick object at 0xadb112c> .. <a list of 20 Text yticklabel objects>)
>>> show()
```

Dan berikut hasilnya:



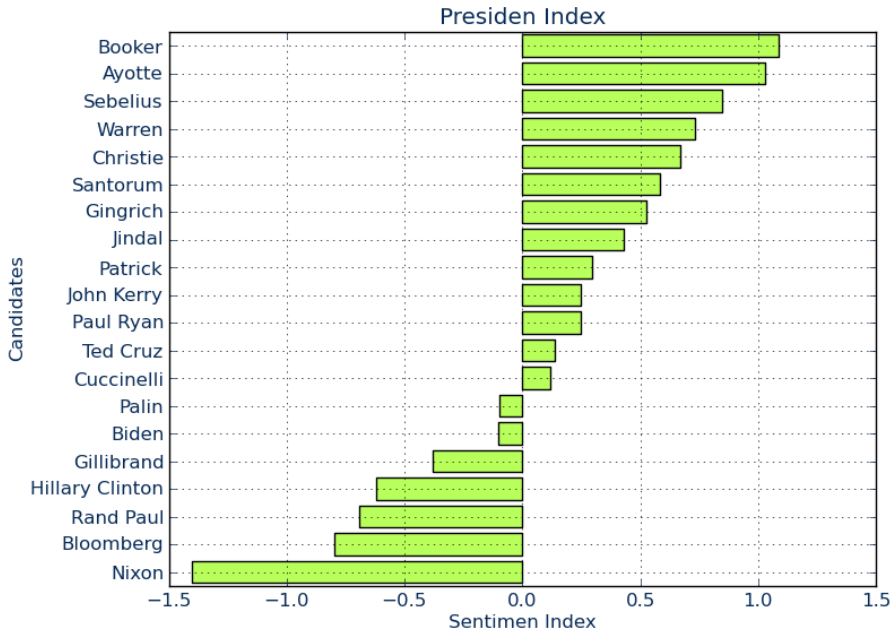
Terlihat dari grafik, ada satu nama yang terpotong yakni calon presiden *Hillary Clinton* karena terlalu panjang, kita dapat menyesuaikan agar nama tersebut masuk ke dalam grafik. Sekalian kita akan lengkapi grafik di atas dengan `title, xlabel, ylabel, grid` dan pernak-pernik lain.

Simpan kode berikut ke dalam berkas python

```
#!/usr/bin/python
```

```
# matplot_6.py

from pylab import *

name = []
value = []

# membaca berkas dan menyimpan tiap baris ke dalam object
with open('presdex.csv') as f:
    listline = f.read().split('\n')

# split tiap baris dengan ;
for line in listline:
    split = line.split(';')
    name.append(split[0])
    value.append(float(split[1]))

# sesuaikan jarak kiri biar semua nama dan atribut tampil
subplots_adjust(left=0.20, right=0.97)
tick_params(axis='x', colors='#072b57')
tick_params(axis='y', colors='#072b57')

# beri label
xlabel('Sentimen Index', color='#072b57')
ylabel('Candidates', color='#072b57')
title('Presiden Index', color='#072b57')

pos = arange(len(name)) + 0.5
barh(pos, value, align='center', color='#b8ff5c')
yticks(pos, name)
grid(True)
show()
```

Jalankan dan berikut hasilnya:



Fungsi	Keterangan
subplots_adjust	menyesuikan jarak tepi grafik, left, right, top, bottom
tick_params	parameter axis x dan y

2.6 Grafik 3D

Sebelumnya, pastikan Anda sudah memasang modul `mpl_toolkits` pada sistem operasi Anda. Jika Anda pengguna Ubuntu, cukup ketikkan:

```
$ sudo apt-get install python-mpltoolkits.basemap
```

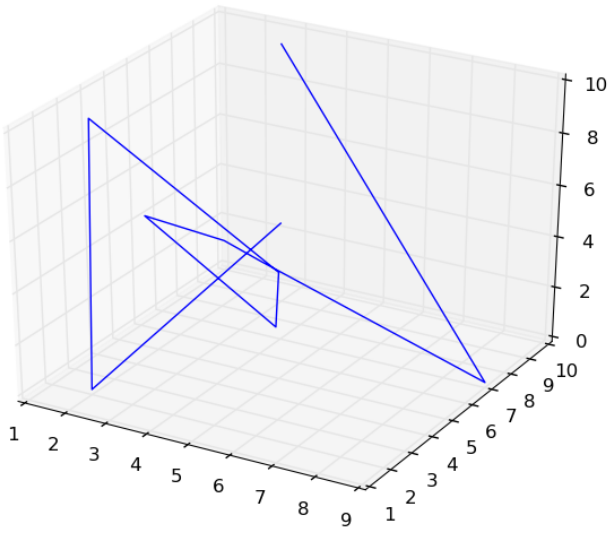
2.6.1 3D Line

Berikut contoh *script* python untuk membuat grafik 3d:

```
>>> from mpl_toolkits.mplot3d import axes3d
>>> import matplotlib.pyplot as plt
>>> from random import randint
>>> X, Y, Z = [randint(0, 10) for i in range(10)],
... [randint(0, 10) for i in range(10)],
... [randint(0, 10) for i in range(10)]
>>> fig = plt.figure()
>>> ax = fig.add_subplot(111, projection='3d')
>>> ax.plot_wireframe(X, Y, Z)
```

```
<mpl_toolkits.mplot3d.art3d.Line3DCollection object at 0x9c738cc>
>>> plt.show()
```

Dan berikut hasilnya:



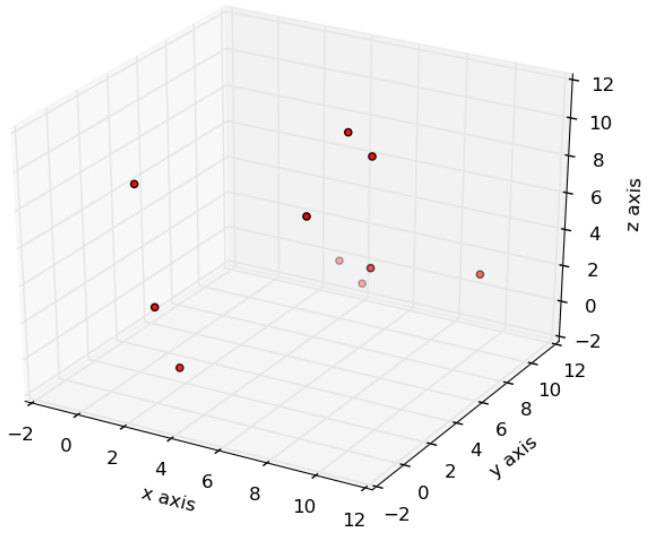
Anda dapat memperbesar grafik di atas dengan menekan klik kanan perangkat *mouse*, kemudian gerakkan maju (*zoom out*) atau mundur (*zoom in*).

2.6.2 3D Scatter Plot

Berikut ini contoh satu lagi untuk membuat grafik 3d scatter plot:

```
>>> from mpl_toolkits.mplot3d import axes3d
>>> import matplotlib.pyplot as plt
>>> from random import randint
>>> fig = plt.figure()
>>> ax = fig.add_subplot(111, projection='3d')
>>> X = [randint(0, 10) for i in range(10)]
>>> Y = [randint(0, 10) for i in range(10)]
>>> Z = [randint(0, 10) for i in range(10)]
>>> ax.scatter(X, Y, Z, c='r', marker='o')
<mpl_toolkits.mplot3d.art3d.Patch3DCollection object at 0x9c9488c>
>>> ax.set_xlabel('x axis')
<matplotlib.text.Text object at 0x9a1206c>
>>> ax.set_ylabel('y axis')
<matplotlib.text.Text object at 0x9b118cc>
>>> ax.set_zlabel('z axis')
<matplotlib.text.Text object at 0x9b1cc8c>
>>> plt.show()
```

Dan berikut hasilnya:



Bila diperhatikan, titik akan berwarna semakin gelap ketika jarak titik dengan pengamat semakin dekat, begitu pun sebaliknya.

```
|---------+---------------------------------|
| Fungsi  | Keterangan }                    |
|---------+---------------------------------|
| scatter | membuat grafik scatter          |
| randint | membuat bilangan random integer |
|---------+---------------------------------|
```

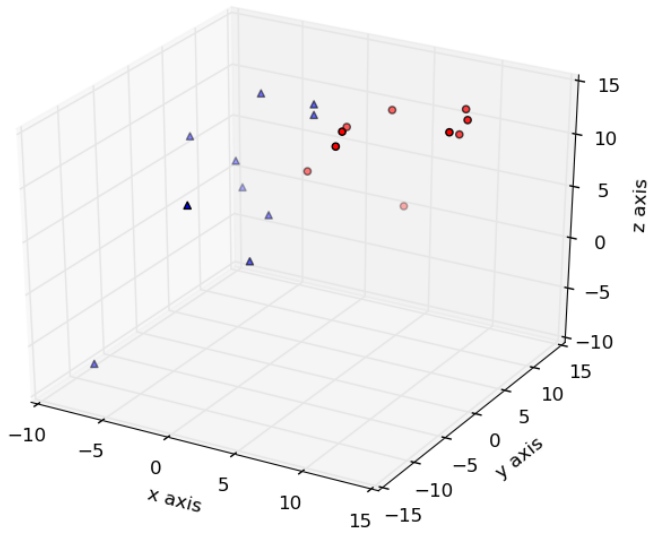
2.6.3 3D Scatter Plot with Multiple Datasets

Pada contoh sebelumnya kita hanya menggunakan satu sumber data, berikut ini contoh penggunaan *scatter plot* menggunakan lebih dari satu sumber data. Ketikkan kode berikut ke dalam python *interpreter* Anda:

```
>>> from mpl_toolkits.mplot3d import axes3d
>>> import matplotlib.pyplot as plt
>>> from random import randint
>>> fig = plt.figure()
>>> ax = fig.add_subplot(111, projection='3d')
>>> X = [randint(0, 10) for i in range(10)]
>>> Y = [randint(1, 13) for i in range(10)]
>>> Z = [randint(2, 12) for i in range(10)]
>>> Xs = [randint(-10, 0) for i in range(10)]
>>> Ys = [randint(-14, 9) for i in range(10)]
>>> Zs = [randint(-8, 12) for i in range(10)]
```

```
>>> ax.scatter(X, Y, Z, c='r', marker='o')
<mpl_toolkits.mplot3d.art3d.Patch3DCollection object at 0x98edf2c>
>>> ax.scatter(Xs, Ys, Zs, c='b', marker='^')
<mpl_toolkits.mplot3d.art3d.Patch3DCollection object at 0x9b8950c>
>>> ax.set_xlabel('x axis')
<matplotlib.text.Text object at 0x98c8a8c>
>>> ax.set_ylabel('y axis')
<matplotlib.text.Text object at 0x98d3bac>
>>> ax.set_zlabel('z axis')
<matplotlib.text.Text object at 0x98dbb2c>
>>> plt.show()
```

Hasil:



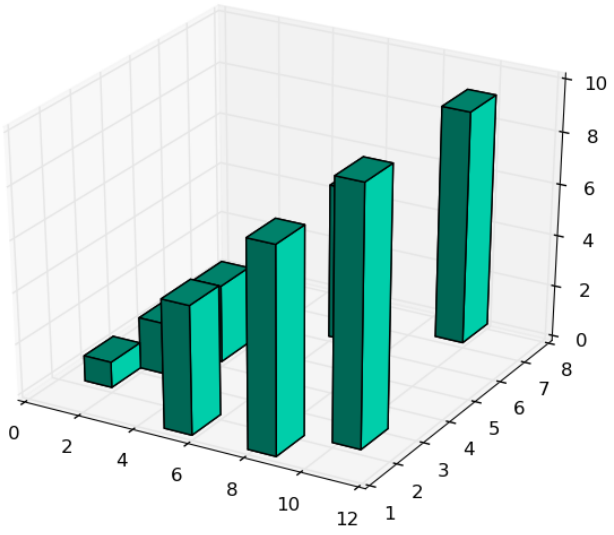
2.6.4 3D Bar Charts

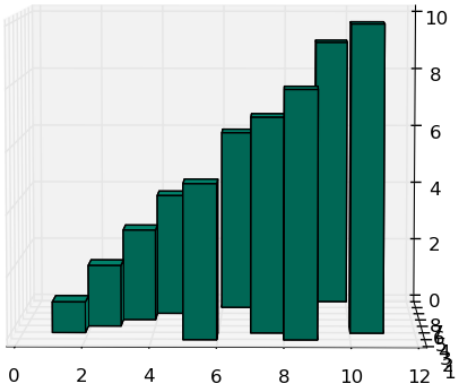
Berikut ini contoh membuat 3d bar charts:

```
>>> from mpl_toolkits.mplot3d import axes3d
>>> import matplotlib.pyplot as plt
>>> import numpy as np
>>> fig = plt.figure()
>>> ax1 = fig.add_subplot(111, projection='3d')
>>> xpos = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> ypos = [2, 3, 4, 5, 1, 6, 2, 1, 7, 2]
>>> np.zeros(10)
array([ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.])
>>> np.ones(10)
array([ 1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.])
```

```
>>> zpos = np.zeros(10)
>>> dx = np.ones(10)
>>> dy = np.ones(10)
>>> dz = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> ax1.bar3d(xpos, ypos, zpos, dx, dy, dz, color='#00ceaa')
>>> plt.show()
```

Hasil:





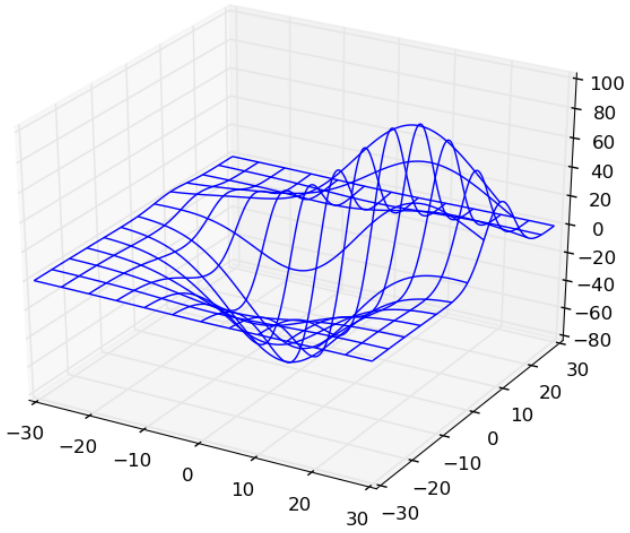
Fungsi	Keterangan
np.zeros	membuat ndarray dengan nilai 0.
np.ones	membuat ndarray dengan nilai 1.

2.6.5 3D Plane Wire Frame

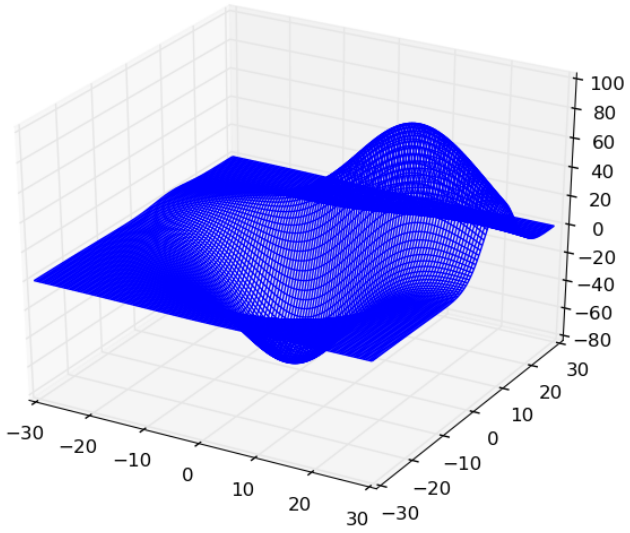
Berikutnya, kita akan mencoba membuat grafik 3d plane wire frame menggunakan `matplotlib`. Perhatikan kode berikut:

```
>>> from mpl_toolkits.mplot3d import axes3d
>>> import matplotlib.pyplot as plt
>>> import numpy as np
>>> fig = plt.figure()
>>> ax = fig.add_subplot(111, projection='3d')
>>> x, y, z = axes3d.get_test_data(0.05)
>>> ax.plot_wireframe(x, y, z, rstride=10, cstride=10)
<mpl_toolkits.mplot3d.art3d.Line3DCollection object at 0xa98048c>
>>> plt.show()
```

Hasil:



Mari kita ubah nilai *parameter* `rstride` dan `cstride` dari `10` ke `1`, dan mari kita lihat hasilnya:



2.7 Contoh Lain

Untuk mendapatkan contoh lebih banyak lagi, bisa berkunjung ke halaman http://matplotlib.org/examples/. Selamat mencoba!

3 Reference

sentdex channel on youtube http://www.youtube.com/user/sentdex