Python Pandas untuk Komputasi Sains
Sebuah Pengantar

Kholid Fuadi*

September 28, 2013

Contents

*http://twitter.com/sopier

1 Sekilas Pandas

Sudah bukan rahasia lagi kalau Python banyak digunakan dalam dunia akademis sebagai alat bantu dalam memecahkan persoalan sains atau dalam hal olah data. Kenapa? Karena Python memiliki modul yang cukup lengkap dan sudah teruji dalam bidang ini (mulai berkembang pesat sejak tahun 2000-an hingga sekarang).

Beberapa modul yang akrab kita dengar ketika bersinggungan dengan dunia sains adalah `NumPy`, `matplotlib`, `SciPy` dan `pandas`. Modul terakhir inilah yang akan dikenalkan secara singkat dalam materi *ebook* ini.

`Pandas` dibangun di atas modul `Numpy` yang memiliki beberapa keunggulan, diantaranya menawarkan struktur data yang kaya dan memiliki banyak fungsi siap pakai untuk bekerja dengan data secara cepat, mudah dan gampang diikuti. Juga, penggunaan `API` yang memiliki konsistensi tinggi menjadikkannya lebih mudah dipakai oleh para analis data.

1.1 Instalasi

Berikut ini langkah-langkah instalasi [`Ubuntu`]:

```
$ sudo apt-get install pyton-pandas
$ sudo apt-get install python-matplotlib
$ sudo apt-get install python-numpy
$ sudo apt-get install ipython
```

Untuk pemasangan pada sistem operasi lain, silakan telusuri lebih jauh di Internet.

2 Series Object

`Pandas` memiliki beberapa `object`, diantaranya adalah `Series object`. Berikut ini salah satu contoh data `Series` sederhana:

```
>>> import pandas as pd
>>> l1 = range(10, 101, 10) # start, stop, step
>>> l1
[10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
>>> s1 = pd.Series(l1)
>>> s1
0     10
1     20
2     30
3     40
4     50
5     60
6     70
7     80
8     90
9    100
>>> type(s1)
<class 'pandas.core.series.Series'>
```

```
>>> s1.values
array([ 10,  20,  30,  40,  50,  60,  70,  80,  90, 100], dtype=int64)
>>> s1.index
Int64Index([0, 1, 2, 3, 4, 5, 6, 7, 8, 9], dtype=int64)
```

Meski tidak memasukkan *index* secara eksplisit, `pandas` menggunakan `offset` sebagai *index*. **Penting** untuk dicatat bahwa sistem penomoran *index* dalam Python dimulai dari angka `0`.

Berikut ini contoh lain, objek `Series` dengan *index* dalam bentuk `lists` label:

```
>>> data1 = [33, 19, 15, 89, 11, -5, 9]
>>> index1 = ['Sen', 'Sel', 'Rab', 'Kam', 'Jum', 'Sab', 'Min']
>>> s2 = pd.Series(data1, index=index1)
>>> s2
Sen    33
Sel    19
Rab    15
Kam    89
Jum    11
Sab    -5
Min     9
>>> s2.index
Index([Sen, Sel, Rab, Kam, Jum, Sab, Min], dtype=object)
>>> s2.name = 'Suhu Harian'
>>> s2.index.name = 'Hari'
>>> s2
Hari
Sen     33
Sel     19
Rab     15
Kam     89
Jum     11
Sab     -5
Min      9
Name: Suhu Harian
```

Contoh tipe data campuran *integer* dan *float*:

```
>>> data2 = [33, 19.3, 15, 89, 11, -5, 9]
>>> s3 = pd.Series(data2, index=index1)
>>> s3
Sen    33.0
Sel    19.3
Rab    15.0
Kam    89.0
Jum    11.0
Sab    -5.0
Min     9.0
>>> # output data berupa float
```

Contoh membuat `Series` dengan *dictionary*:

```
>>> dict1 = {'Sen': 33, 'Sel': 19, 'Rab': 15, 'Kam': 89, 'Jum': 15, 'Sab': -5, 'Min': 19}
>>> s4 = pd.Series(dict1)
```

```
>>> s4
Jum    15
Kam    89
Min    19
Rab    15
Sab    -5
Sel    19
Sen    33
```

Operasi `vector` pada objek *Series*:

```
>>> s4 * 2
Jum     30
Kam    178
Min     38
Rab     30
Sab    -10
Sel     38
Sen     66
```

Contoh data `NaN` (*not a number*), penanda *missing data* standar yang digunakan dalam `Pandas`.

```
>>> import numpy as np
>>> np.log(s4)
Jum    2.708050
Kam    4.488636
Min    2.944439
Rab    2.708050
Sab         NaN
Sel    2.944439
Sen    3.496508
```

Karena data yang kita miliki berbentuk *array*, maka kita dapat melakukan operasi pembelahan (*slicing*) seperti halnya tipe data `list` dalam `Python`:

```
>>> s4['Jum': 'Rab']
Jum    15
Kam    89
Min    19
Rab    15
>>> s4[1:4]
Kam    89
Min    19
Rab    15
```

Begitu juga dengan operasi pengambilan (*get*) nilai menggunakan *offset*:

```
>>> s4[3]
15
```

Atau juga memberikan (*set*) nilai menggunakan *offset*:

```
>>> s4[3] = 21
>>> s4
Jum    15
```

```
Kam    89
Min    19
Rab    21
Sab    -5
Sel    19
Sen    33
```

Karena objek `Series` merupakan *subclass* dari `ndarray`, maka kita dapat menerapkan fungsi `NumPy` yang berkaitan dengannya, contoh:

```
>>> s4.median()
19.0
>>> s4.max()
89
>>> s4.cumsum()
Jum     15
Kam    104
Min    123
Rab    144
Sab    139
Sel    158
Sen    191
```

2.1 Looping Series

Kita juga dapat melakukan *looping*:

```
>>> for i, v in enumerate(s4):
...     print i, v
...
0 15
1 89
2 19
3 21
4 -5
5 19
6 33
```

Menggunakan *list comprehension*[1] untuk membuat `list` baru secara cepat:

```
>>> newlist = [x ** 2 for x in s4]
>>> newlist
[225, 7921, 361, 441, 25, 361, 1089]
```

2.2 Series is dict-like

Selain memiliki karakter seperti `list`, objek `Series` juga memiliki karakter seperti `Dictionary` dalam Python.

Cek apakah `key` ada dalam `Series`:

```
>>> 'Jum' in s4
True
```

[1] bagi yang belum familier, silakan telusuri di Internet lebih jauh tentang fitur ini

Mengambil (*get*) nilai menggunakan *key* atau *index*:

```
>>> s4['Rab']
21
```

Memberi (*set*) nilai menggunakan *key*:

```
>>> s4['Rab'] = 25
>>> s4
Jum    15
Kam    89
Min    19
Rab    25
Sab    -5
Sel    19
Sen    33
```

Looping `dictionary` menggunakan *keys* dan *values*:

```
>>> for k, v in s4.iteritems():
...     print k, '=>', v
...
Jum => 15
Kam => 89
Min => 19
Rab => 25
Sab => -5
Sel => 19
Sen => 33
```

3 DataFrames Object

Apa itu *DataFrame*? *DataFrame* adalah objek data *2-dimensional* yang terdiri dari beberapa baris dan kolom. Masing-masing dari baris dan kolom tersebut memiliki *index*, dan bisa memiliki atau tidak memiliki *label*. Tiap kolom memiliki tipe data yang seragam (sama).

3.1 Membuat Objek DataFrame

Contoh membuat objek *DataFrame* menggunakan `pandas`:

```
Python 2.7.4 (default, Apr 19 2013, 18:32:33)
[GCC 4.7.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import pandas as pd
>>> from pylab import randint
>>> temps = randint(100, size=(12, 4))
>>> temps
array([[41, 56, 65, 96],
       [16, 60, 57, 92],
       [63, 36, 49, 71],
       [72, 74, 59, 64],
       [69, 80, 94, 91],
       [72,  3, 55, 37],
```

```
       [61, 18, 52, 40],
       [22, 76, 53, 20],
       [ 2, 46, 49, 31],
       [55, 40, 71, 62],
       [53, 52, 72,  9],
       [20, 41, 63, 73]])
>>> df1 = pd.DataFrame(temps)
>>> df1
     0   1   2   3
0   41  56  65  96
1   16  60  57  92
2   63  36  49  71
3   72  74  59  64
4   69  80  94  91
5   72   3  55  37
6   61  18  52  40
7   22  76  53  20
8    2  46  49  31
9   55  40  71  62
10  53  52  72   9
11  20  41  63  73
```

3.2 Membuat DataFrame dari Python Dictionary

```
>>> mlb_attendance = {'Team':
... ['San Diego Padres', 'Los Angeles Dodgers', 'New York Yankees',
... 'Boston Red Sox', 'Toronto Blue Jays', 'San Diego Padres',
... 'Los Angeles Dodgers', 'New York Yankees', 'Boston Red Sox',
... 'Toronto Blue Jays'],
... 'Year': [2011, 2011, 2011, 2011, 2011, 2012, 2012, 2012, 2012, 2012],
... 'Attendance': [2143018, 2935139, 3653680, 3054001, 1818103, 2123721,
... 3324246, 3542406, 3043003, 2099663]}
>>> mlb_attendance.keys()
['Year', 'Attendance', 'Team']
>>> df_attendance = pd.DataFrame(mlb_attendance)
>>> df_attendance
   Attendance                Team  Year
0     2143018     San Diego Padres  2011
1     2935139  Los Angeles Dodgers  2011
2     3653680     New York Yankees  2011
3     3054001       Boston Red Sox  2011
4     1818103    Toronto Blue Jays  2011
5     2123721     San Diego Padres  2012
6     3324246  Los Angeles Dodgers  2012
7     3542406     New York Yankees  2012
8     3043003       Boston Red Sox  2012
9     2099663    Toronto Blue Jays  2012
```

3.3 Membuat DataFrame dari berkas csv

2

```
>>> mlb_sal = pd.read_csv('mlbsalaries.csv')
>>> mlb_sal
```

[2]Sumber data: https://dl.dropboxusercontent.com/u/5052616/mlbsalaries.csv

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 19543 entries, 0 to 19542
Data columns:
Year        19543  non-null values
Player      19543  non-null values
Salary      19543  non-null values
Position    19543  non-null values
Team        19543  non-null values
dtypes: int64(2), object(3)
>>> mlb_sal.tail()
       Year          Player  Salary        Position              Team
19538  2011  Gustavo Molina  455000         Catcher  New York Yankees
19539  2011       Ivan Nova  432900         Pitcher  New York Yankees
19540  2011    Colin Curtis  420400      Outfielder  New York Yankees
19541  2011   Eduardo Nunez  419300       Shortstop  New York Yankees
19542  2011   Reegie Corona  414000  Second Baseman  New York Yankees
>>> mlb_sal.head()
   Year           Player   Salary           Position                Team
0  1988        Mike Witt  1400000            Pitcher  Los Angeles Angels
1  1988  George Hendrick   989333         Outfielder  Los Angeles Angels
2  1988      Chili Davis   950000         Outfielder  Los Angeles Angels
3  1988    Brian Downing   900000  Designated Hitter  Los Angeles Angels
4  1988        Bob Boone   883000            Catcher  Los Angeles Angels
```

3.4 Memberi Label pada baris dan kolom

```
>>> rowlabels = ['Jan', 'Feb', 'Mar', 'Apr', 'Jun', 'Aug', 'Sep', 'Oct',
... 'Nov', 'Dec']
>>> columnlabels = ['Denver', 'Austin', 'Boston', 'Atlanta']
>>> df2 = pd.DataFrame(temps, index=rowlabels)
>>> df3 = pd.DataFrame(temps, columns=columnlabels)
>>> df4 = pd.DataFrame(temps, index=rowlabels, columns=columnlabels)
>>> df1
     0   1   2   3
0   41  56  65  96
1   16  60  57  92
2   63  36  49  71
3   72  74  59  64
4   69  80  94  91
5   72   3  55  37
6   61  18  52  40
7   22  76  53  20
8    2  46  49  31
9   55  40  71  62
10  53  52  72   9
11  20  41  63  73
>>> df2
      0   1   2   3
Jan  42  16  64  85
Feb  56  32  11  68
Mar  39  91  87  92
Apr  80  34  92  17
May  69  47  19  69
Jun  91  87  56  96
Jul  63  72  15  73
Aug  60  97  64  82
Sep  61  87  98  97
```

```
Oct  40  98  86  96
Nov  19  15  93  82
Dec  60   9  53  51
>>> df3
    Denver  Austin  Boston  Atlanta
0       42      16      64       85
1       56      32      11       68
2       39      91      87       92
3       80      34      92       17
4       69      47      19       69
5       91      87      56       96
6       63      72      15       73
7       60      97      64       82
8       61      87      98       97
9       40      98      86       96
10      19      15      93       82
11      60       9      53       51
>>> df4
     Denver  Austin  Boston  Atlanta
Jan      42      16      64       85
Feb      56      32      11       68
Mar      39      91      87       92
Apr      80      34      92       17
May      69      47      19       69
Jun      91      87      56       96
Jul      63      72      15       73
Aug      60      97      64       82
Sep      61      87      98       97
Oct      40      98      86       96
Nov      19      15      93       82
Dec      60       9      53       51
```

Terlihat bahwa pada `df1` kita belum memberi *label* pada *DataFrame*, pada `df2`, kita memberi *label* pada baris. `df3` kita memberi *label* pada kolom, dan pada `df4` kita memberikan *label* baik pada baris maupun kolom.

3.5 DataFrame Subsets

3.5.1 Subset of Rows

```
>>> df1[5:7]
    0   1   2   3
5  72   3  55  37
6  61  18  52  40
>>> df2['Feb': 'May']
      0   1   2   3
Feb  56  32  11  68
Mar  39  91  87  92
Apr  80  34  92  17
May  69  47  19  69
```

3.5.2 Subset of Columns

```
>>> df3[['Denver']]
    Denver
0       42
```

```
1       56
2       39
3       80
4       69
5       91
6       63
7       60
8       61
9       40
10      19
11      60
>>> df3[['Denver', 'Boston']]
    Denver  Boston
0       42      64
1       56      11
2       39      87
3       80      92
4       69      19
5       91      56
6       63      15
7       60      64
8       61      98
9       40      86
10      19      93
11      60      53
>>> df3.Denver
0     42
1     56
2     39
3     80
4     69
5     91
6     63
7     60
8     61
9     40
10    19
11    60
Name: Denver
```

3.5.3 Subsets of Rows and Columns

Membuat *subsets* baris dan kolom menggunakan fungsi `ix`.

```
>>> df3
    Denver  Austin  Boston  Atlanta
0       42      16      64       85
1       56      32      11       68
2       39      91      87       92
3       80      34      92       17
4       69      47      19       69
5       91      87      56       96
6       63      72      15       73
7       60      97      64       82
8       61      87      98       97
9       40      98      86       96
10      19      15      93       82
```

```
11      60       9      53       51
>>> df3.ix[2:5, 'Denver':'Boston']
   Denver  Austin  Boston
2      39      91      87
3      80      34      92
4      69      47      19
5      91      87      56
>>> df2.ix['Feb': 'Jun', :]
      0   1   2   3
Feb  56  32  11  68
Mar  39  91  87  92
Apr  80  34  92  17
May  69  47  19  69
Jun  91  87  56  96
>>> df2.ix['Feb': 'Jun', 2:4]
      2   3
Feb  11  68
Mar  87  92
Apr  92  17
May  19  69
Jun  56  96
>>> df4
     Denver  Austin  Boston  Atlanta
Jan      42      16      64       85
Feb      56      32      11       68
Mar      39      91      87       92
Apr      80      34      92       17
May      69      47      19       69
Jun      91      87      56       96
Jul      63      72      15       73
Aug      60      97      64       82
Sep      61      87      98       97
Oct      40      98      86       96
Nov      19      15      93       82
Dec      60       9      53       51
>>> df4.ix[['Feb', 'Apr', 'Jun'], ['Boston', 'Austin']]
     Boston  Austin
Feb      11      32
Apr      92      34
Jun      56      87
>>> df1
     0   1   2   3
0   41  56  65  96
1   16  60  57  92
2   63  36  49  71
3   72  74  59  64
4   69  80  94  91
5   72   3  55  37
6   61  18  52  40
7   22  76  53  20
8    2  46  49  31
9   55  40  71  62
10  53  52  72   9
11  20  41  63  73
>>> df1.ix[[3,4,5],[2,3]]
    2   3
3  59  64
```

```
4  94  91
5  55  37
```

3.6 Data Selection

Bagian ini membahas bagaimana kita dapat melakukan *data selection* terhadap *DataFrame*.

Sebagai contoh, mari kita buat *array* `7 x 6`, yang isinya angka *random* antara `0` sampai `100`. Langkah pertama, gunakan fungsi `randrange` dari modul `random` untuk membuat angka secara acak dari 0 sampai 100, masukkan ke dalam *list comprehension* agar menghasilkan tipe data `list` yang memiliki panjang `42`. Kemudian gunakan fungsi `array` dari modul `numpy` untuk mengubah tipe data `list` menjadi `array`. Setelah itu gunakan fungsi `reshape` dari modul `numpy` untuk mengubah *array* menjadi *array* 2 dimensi (`7x6`).[3]

Berikut ini contohnya:

```
Python 2.7.4 (default, Apr 19 2013, 18:32:33)
[GCC 4.7.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import pandas as pd
>>> from random import randrange
>>> import numpy as np
>>> from numpy import reshape
>>> import matplotlib.pyplot as plt
>>> np.array([randrange(0, 100) for i in range(42)]).reshape(7, 6)
array([[71, 86, 40, 44, 48, 46],
       [38, 88, 21, 60, 76, 25],
       [60, 35, 73, 37, 73, 44],
       [68, 58,  6, 56,  3, 51],
       [90, 82, 39, 54, 93,  1],
       [39, 54, 41, 86, 78, 97],
       [83, 86, 43, 63, 83, 21]])
```

Selanjutnya mari kita buat *label* untuk masing-masing baris dan kolom.

```
>>> rowids = ['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun']
>>> colids = ['Boston', 'London', 'Paris', 'Sydney', 'Tokyo', 'Toronto']
```

Sekarang beri *label* `t` untuk *array* acak yang akan kita gunakan:

```
>>> t = np.array([randrange(0, 100) for i in range(42)]).reshape(7, 6)
>>> t
array([[24, 10, 76, 97, 63,  0],
       [ 4,  7, 69, 95, 37, 98],
       [76, 42, 75, 13, 51, 71],
       [57, 94, 43, 73, 13, 88],
       [29, 23, 36, 25, 50,  3],
       [73, 49, 62, 11, 49, 88],
       [42, 14, 40, 40,  4, 63]])
```

Selanjutnya mari kita buat *DataFrame* dan beri *label* `temps`:

[3] karena datanya acak, hasil data bisa jadi berbeda dengan yang penulis dapat

```
>>> temps = pd.DataFrame(t, index=rowids, columns=colids)
>>> temps
     Boston  London  Paris  Sydney  Tokyo  Toronto
Mon      24      10     76      97     63        0
Tue       4       7     69      95     37       98
Wed      76      42     75      13     51       71
Thu      57      94     43      73     13       88
Fri      29      23     36      25     50        3
Sat      73      49     62      11     49       88
Sun      42      14     40      40      4       63
```

Untuk contoh pertama, kita coba menseleksi data temperatur di kota Sydney yang nilainya lebih dari atau sama 60.

Langkah pertama adalah dengan mencari suhu di kolom Sydney dengan kriteria nilai lebih dari 60.

```
>>> temps.Sydney >= 60
Mon     True
Tue     True
Wed    False
Thu     True
Fri    False
Sat    False
Sun    False
Name: Sydney
```

Nilai kembalian (`return`) berupa tipe data `boolean` yang bernilai `True` jika kriteria sesuai dengan yang kita inginkan (`>=60`), dan akan bernilai `False` jika tidak sesuai kriteria. Dari hasil terlihat, bahwa data suhu di kolom Sydney pada hari Mon, Tue, dan Thu bernilai `True`.

Langkah kedua, gunakan nilai *boolean* tersebut sebagai `key` untuk *DataFrame* kita. Hasilnya adalah data di mana hari-hari di kota Sydney dengan suhu sama atau lebih dari 60.

```
>>> temps[temps.Sydney >= 60]
     Boston  London  Paris  Sydney  Tokyo  Toronto
Mon      24      10     76      97     63        0
Tue       4       7     69      95     37       98
Thu      57      94     43      73     13       88
```

Kondisi atau kriteria ini bisa kita asosiasikan dengan nama atau *label* sendiri, dan juga dapat kita gabung dengan kriteria lain.

```
>>> mask = temps.Sydney >= 60
>>> temps[mask]
     Boston  London  Paris  Sydney  Tokyo  Toronto
Mon      24      10     76      97     63        0
Tue       4       7     69      95     37       98
Thu      57      94     43      73     13       88
```

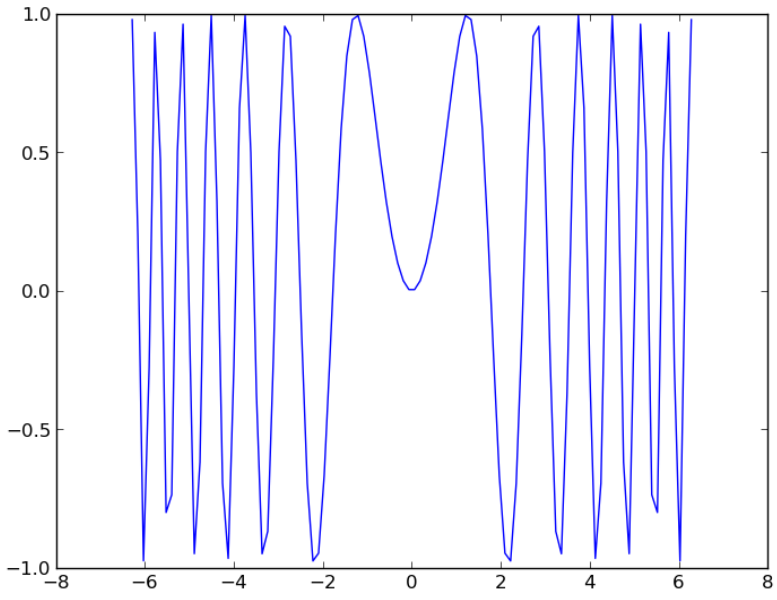
Contoh berikut menggambarkan bagaimana kita dapat menggabung beberapa kriteria sekaligus. Dalam contoh ini kita mencari data hari di kota Sydney dan Toronto yang bersuhu lebih dari atau sama dengan 60. Di sini kita menggunakan *bitwise operator* `&`.

```
>>> mask1 = (temps.Sydney >= 60) & (temps.Toronto >= 60)
>>> mask1
Mon    False
Tue     True
Wed    False
Thu     True
Fri    False
Sat    False
Sun    False
>>> temps[mask1]
     Boston  London  Paris  Sydney  Tokyo  Toronto
Tue       4       7     69      95     37       98
Thu      57      94     43      73     13       88
```

Selain menggunakan *bitwise operator* `&`, kita juga bisa menggunakan *operator* `|` (dibaca `or`). Contoh penggunaan *operator* ini akan disajikan dalam contoh kasus di bagian selanjutnya.

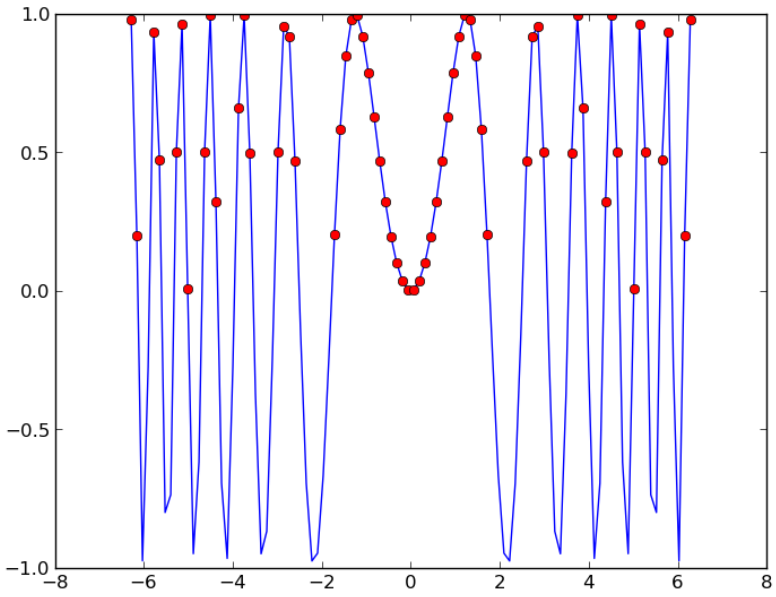
Contoh lain lagi dari proses *data selection*: Misal kita ingin membuat *linearly spaced array* mulai angka `-2*PI` sampai `2*PI`.

```
>>> import numpy as np
>>> x = np.linspace(-2*np.pi, 2*np.pi, 100)
>>> y = np.sin(x**2)
>>> import matplotlib.pyplot as plt
>>> plt.plot(x, y)
[<matplotlib.lines.Line2D object at 0xb3ca1ec>]
>>> plt.show()
```



Sekarang tambahkan kriteria, dalam hal ini hanya y yang bernilai lebih dari atau sama 0:

```
>>> mask = y>=0
>>> plt.plot(x, y)
[<matplotlib.lines.Line2D object at 0xa5c3dcc>]
>>> plt.plot(x[mask], y[mask], 'ro')
[<matplotlib.lines.Line2D object at 0xb0fe66c>]
>>> plt.show()
```



4 Index Object

Objek *Index* ini digunakan sebagai label *axis* dan informasi *metadata* lain seperti nama *axis* atau nama. Perhatikan contoh berikut:

```
>>> import pandas as pd
>>> import numpy as np
>>> from pandas import Series, DataFrame
>>> obj = Series(range(3), index=['a', 'b', 'c'])
>>> index = obj.index
>>> index
Index([a, b, c], dtype=object)
>>> index[1:]
Index([b, c], dtype=object)
```

Index bersifat *immutable* sehingga tidak dapat diubah nilainya:

```
>>> index[1] = 'd'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/usr/lib/python2.7/dist-packages/pandas/core/index.py", line 352, in __setitem__
    raise Exception(str(self.__class__) + ' object is immutable')
Exception: <class 'pandas.core.index.Index'> object is immutable
```

Sifat *immutable* ini penting supaya *index* nilainya tetap terjaga ketika digunakan antar tipe data:

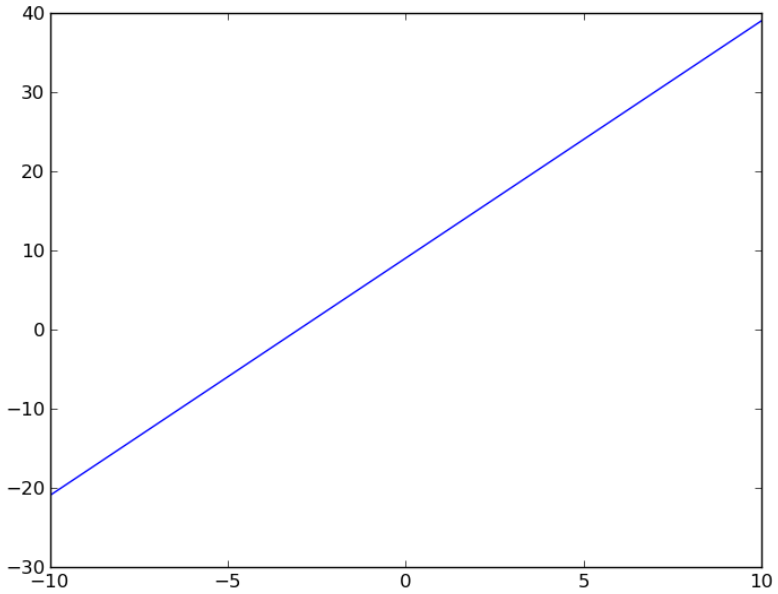
```
>>> index = pd.Index(np.arange(3))
>>> index
Int64Index([0, 1, 2], dtype=int64)
>>> obj2 = Series([1.5, -2.5, 0], index=index)
>>> obj2.index is index
True
```

5 Plotting dan Visualisasi

Bagian ini akan membahas bagaimana kita dapat menggambar *plot* menggunakan fungsi-fungsi dari modul `matplotlib`.

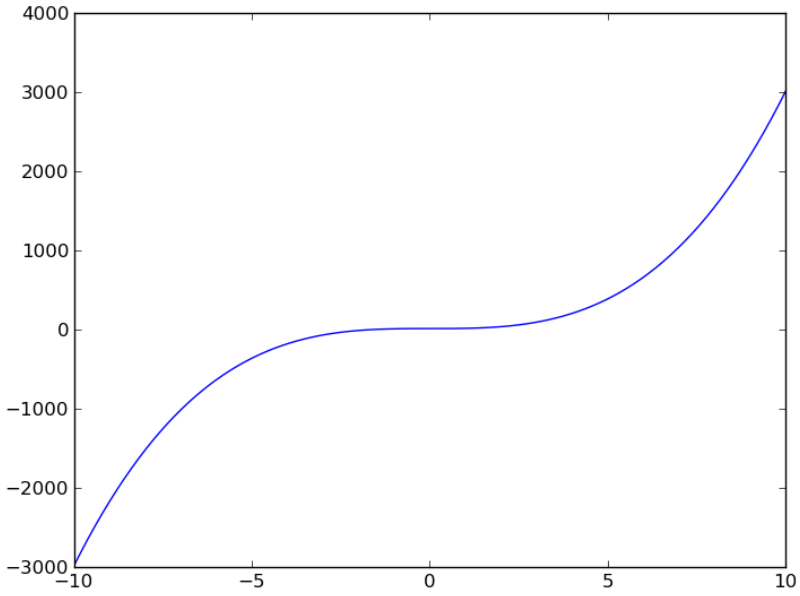
5.1 Dasar Plot

```
>>> from numpy import linspace
>>> x = linspace(-10, 10, 100)
>>> y = 3 * x + 9
>>> plt.plot(x, y)
[<matplotlib.lines.Line2D object at 0xac0d52c>]
>>> plt.show()
```



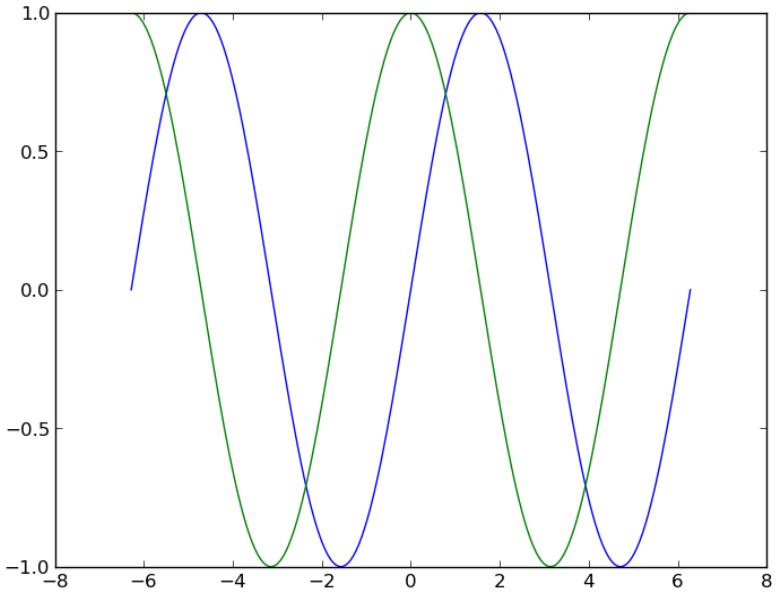
Sekarang mari kita ubah y menjadi seperti berikut:

```
>>> y = 3*x**3 + 9
>>> plt.plot(x, y)
[<matplotlib.lines.Line2D object at 0xabff98c>]
>>> plt.show()
```



Contoh selanjutnya menggambar *plot* untuk fungsi *trigonometry*:

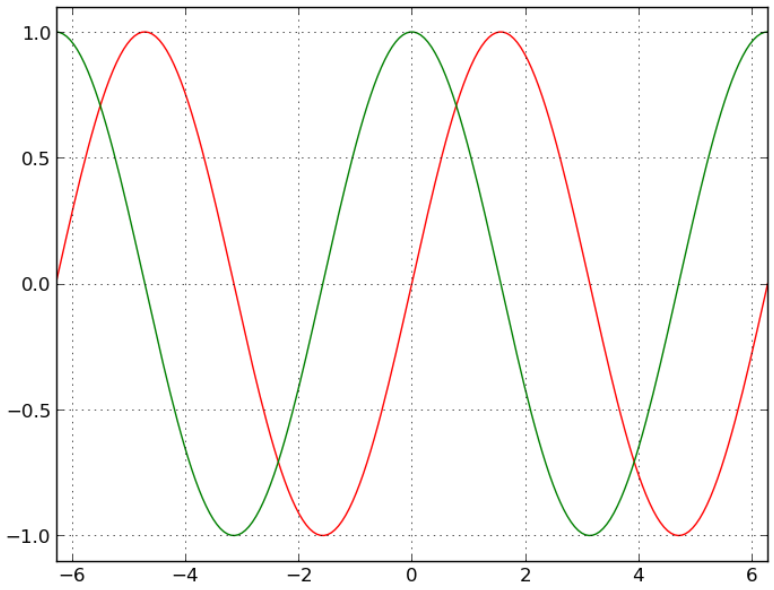
```
>>> from numpy import pi
>>> from numpy import sin
>>> from numpy import cos
>>> x = linspace(-2*pi, 2*pi, 200)
>>> y1 = sin(x)
>>> y2 = cos(x)
>>> plt.plot(x, y1)
[<matplotlib.lines.Line2D object at 0xac1eb4c>]
>>> plt.plot(x, y2)
[<matplotlib.lines.Line2D object at 0xaba512c>]
>>> plt.show()
```



5.2 Menyunting Properti Plot

Selanjutnya mari kita sunting sedikit warna, batas tepi, dan tampilkan garis bantu (*grid*) ke dalam *plot*.

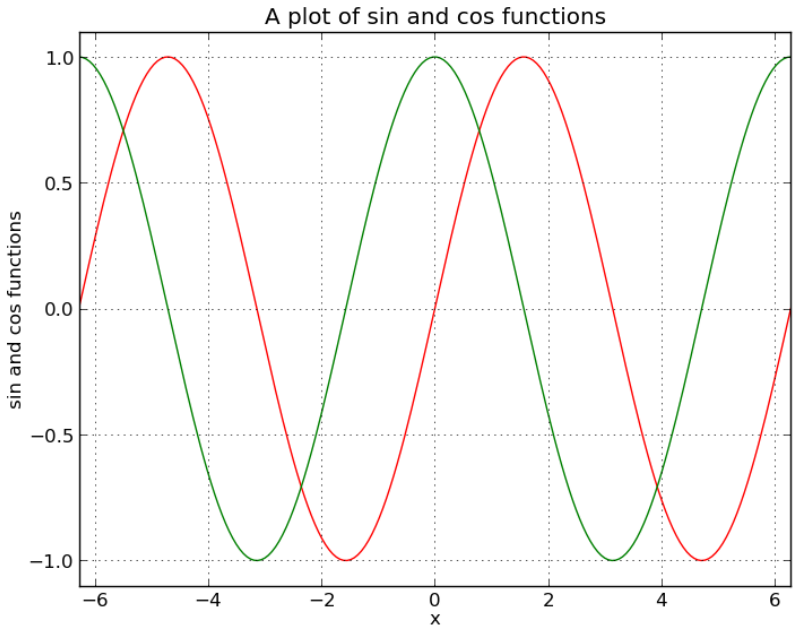
```
>>> plt.plot(x, y1, 'r')
[<matplotlib.lines.Line2D object at 0xae63c2c>]
>>> plt.plot(x, y2, 'g')
[<matplotlib.lines.Line2D object at 0xab426cc>]
>>> plt.grid()
>>> plt.xlim(-2*pi, 2*pi)
(-6.283185307179586, 6.283185307179586)
>>> plt.ylim(-1.1, 1.1)
(-1.1, 1.1)
>>> plt.show()
```



5.3 Memberi Label pada Plot

Berikutnya, mari kita berikan *label* masing-masing untuk *x-axis*, *y-axis* dan judul dari gambar *plot* kita:

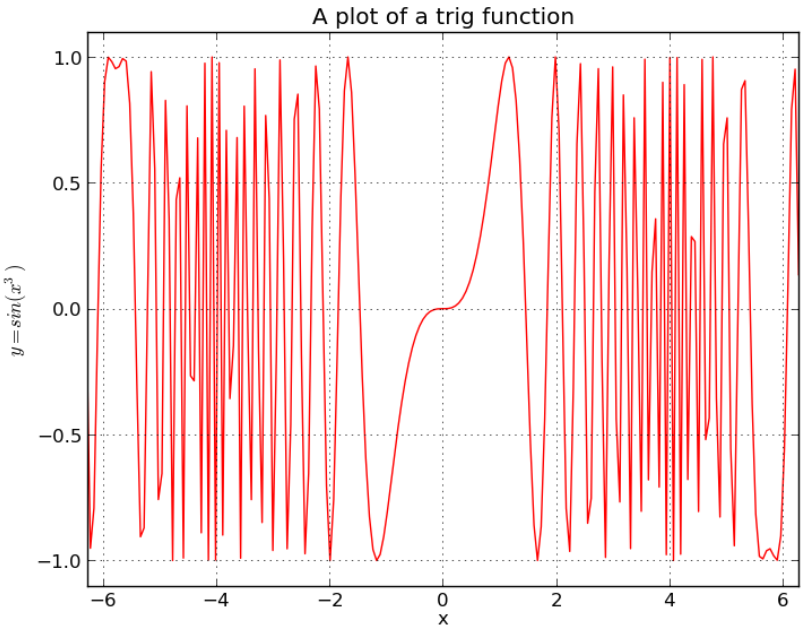
```
>>> plt.plot(x, y1, 'r')
[<matplotlib.lines.Line2D object at 0xac9c5cc>]
>>> plt.plot(x, y2, 'g')
[<matplotlib.lines.Line2D object at 0xaeda9ac>]
>>> plt.grid()
>>> plt.xlim(-2*pi, 2*pi)
(-6.283185307179586, 6.283185307179586)
>>> plt.ylim(-1.1, 1.1)
(-1.1, 1.1)
>>> plt.xlabel('x')
<matplotlib.text.Text object at 0xab9f50c>
>>> plt.ylabel('sin and cos functions')
<matplotlib.text.Text object at 0xab981ac>
>>> plt.title('A plot of sin and cos functions')
<matplotlib.text.Text object at 0xab8680c>
>>> plt.show()
```



5.4 Menggunakan Syntax LaTeX pada Plot

Contoh terakhir, kita menggunakan *syntax* LaTeX untuk menghasilkan *typesetting* yang terlihat profesional. *Syntax* LaTeX dapat disematkan ke dalam modul `matplotlib` dengan menambahkan tanda `$` (*dollar*) ke dalam persamaan matematika. Perhatikan contoh berikut:

```
>>> x = linspace(-2*pi, 2*pi, 200)
>>> y1 = sin(x**3)
>>> plt.plot(x, y1, 'r')
[<matplotlib.lines.Line2D object at 0xab86a4c>]
>>> plt.grid()
>>> plt.xlim(-2*pi, 2*pi)
(-6.283185307179586, 6.283185307179586)
>>> plt.ylim(-1.1, 1.1)
(-1.1, 1.1)
>>> plt.xlabel('x')
<matplotlib.text.Text object at 0xabdb56c>
>>> plt.ylabel('$y = sin(x^3)$') # <= syntax latex
<matplotlib.text.Text object at 0xac9d7ac>
>>> plt.title('A plot of a trig function')
<matplotlib.text.Text object at 0xab98fec>
>>> plt.show()
```



5.5 Menyimpan Plot

Kita dapat menyimpan gambar *plot* menggunakan fungsi `savefig` dari modul `numpy`, berikut contohnya:

```
>>> from matplotlib.pyplot import savefig
>>> savefig('gambar_1.png')
```

6 Contoh Kasus

6.1 Baby Names Data

Mari kita coba lakukan analisis sederhana terhadap data nama bayi yang lahir di Amerika berikut ini.[4]

```
>>> import pandas as pd
>>> import pylab
>>> names = pd.read_csv('baby-names.csv')
>>> names
<class 'pandas.core.frame.DataFrame'>
Int64Index: 258000 entries, 0 to 257999
Data columns:
year       258000  non-null values
name       258000  non-null values
```

[4] data dapat Anda unduh di https://github.com/hadley/data-baby-names

```
percent    258000  non-null values
sex        258000  non-null values
dtypes: float64(1), int64(1), object(2)
```

Syntax di atas adalah perintah untuk mengimpor data `csv`, yang kemudian kita beri label objek tersebut dengan nama `names`. Apabila kita ketik `names`, maka akan muncul deskripsi singkat (emphmetadata) tentang data yang baru saja kita impor.

Dari hasil kita dapat melihat bahwa data ini memiliki 258.000 baris, yang memiliki empat kolom (*year, name, percent, sex*). Mari kita lihat beberapa baris pertama dari data kita:

```
>>> names.head()
   year     name   percent  sex
0  1880     John  0.081541  boy
1  1880  William  0.080511  boy
2  1880    James  0.050057  boy
3  1880  Charles  0.045167  boy
4  1880   George  0.043292  boy
```

Fungsi `head` dapat dimasuki argumen jumlah baris pertama yang akan kita tampilkan, misal kita ingin menampilkan 12 baris pertama dari data:

```
>>> names.head(12)
    year     name   percent  sex
0   1880     John  0.081541  boy
1   1880  William  0.080511  boy
2   1880    James  0.050057  boy
3   1880  Charles  0.045167  boy
4   1880   George  0.043292  boy
5   1880    Frank  0.027380  boy
6   1880   Joseph  0.022229  boy
7   1880   Thomas  0.021401  boy
8   1880    Henry  0.020641  boy
9   1880   Robert  0.020404  boy
10  1880   Edward  0.019965  boy
11  1880    Harry  0.018175  boy
```

Atau, jika ingin lebih fleksibel, bisa menggunakan fungsi *slicing*:

```
>>> names[3:9]
   year     name   percent  sex
3  1880  Charles  0.045167  boy
4  1880   George  0.043292  boy
5  1880    Frank  0.027380  boy
6  1880   Joseph  0.022229  boy
7  1880   Thomas  0.021401  boy
8  1880    Henry  0.020641  boy
```

Dapat juga melakukan *slicing* lintas baris dan kolom (menggunakan fungsi `ix`), misal kita ingin menampilkan data dari baris 2 sampai 4 hanya untuk kolom *year* dan *name.*:

```
>>> names.ix[2:4, ['year', 'name']]
   year     name
2  1880    James
3  1880  Charles
4  1880   George
```

Modul `Pandas` memungkinkan kita membuat *datasets* baru dengan sangat mudah. Misal kita ingin membuat *datasets* nama bayi laki-laki saja atau perempuan saja.

```
>>> names['sex'] == 'girl'
0     False
1     False
2     False
3     False
4     False
5     False
6     False
7     False
8     False
9     False
10    False
11    False
12    False
13    False
14    False
...
257985    True
257986    True
257987    True
257988    True
257989    True
257990    True
257991    True
257992    True
257993    True
257994    True
257995    True
257996    True
257997    True
257998    True
257999    True
Name: sex, Length: 258000
>>> girl_names = names[names['sex'] == 'girl']
>>> boy_names = names[names['sex'] == 'boy']
>>> girl_names
<class 'pandas.core.frame.DataFrame'>
Int64Index: 129000 entries, 129000 to 257999
Data columns:
year       129000  non-null values
name       129000  non-null values
percent    129000  non-null values
sex        129000  non-null values
dtypes: float64(1), int64(1), object(2)
>>> girl_names.head()
        year       name   percent   sex
129000  1880       Mary  0.072381  girl
```

```
129001  1880       Anna  0.026678  girl
129002  1880       Emma  0.020521  girl
129003  1880  Elizabeth  0.019865  girl
129004  1880     Minnie  0.017888  girl
>>> boy_names
<class 'pandas.core.frame.DataFrame'>
Int64Index: 129000 entries, 0 to 128999
Data columns:
year       129000  non-null values
name       129000  non-null values
percent    129000  non-null values
sex        129000  non-null values
dtypes: float64(1), int64(1), object(2)
>>> boy_names.head()
   year     name   percent  sex
0  1880     John  0.081541  boy
1  1880  William  0.080511  boy
2  1880    James  0.050057  boy
3  1880  Charles  0.045167  boy
4  1880   George  0.043292  boy
>>> boy_names.tail() # fungsi untuk menampilkan 5 baris terakhir
        year    name   percent  sex
128995  2008  Kolten  0.000090  boy
128996  2008  Damari  0.000089  boy
128997  2008    Hugh  0.000089  boy
128998  2008  Jensen  0.000089  boy
128999  2008   Yurem  0.000089  boy
```

Menampilkan data anak yang bernama *William* saja:

```
>>> william = names[names['name'] == 'William']
>>> william
<class 'pandas.core.frame.DataFrame'>
Int64Index: 237 entries, 1 to 236957
Data columns:
year       237  non-null values
name       237  non-null values
percent    237  non-null values
sex        237  non-null values
dtypes: float64(1), int64(1), object(2)
>>> william.head()
      year     name   percent  sex
1     1880  William  0.080511  boy
1001  1881  William  0.078712  boy
2001  1882  William  0.076191  boy
3001  1883  William  0.074558  boy
4001  1884  William  0.072475  boy
```

Menampilkan hanya kolom *year* dan *percent* pada data anak yang bernama *William* saja:

```
>>> william = william.ix[0:, ['year', 'percent']]
>>> william.head()
      year   percent
1     1880  0.080511
1001  1881  0.078712
2001  1882  0.076191
```

```
3001  1883  0.074558
4001  1884  0.072475
```

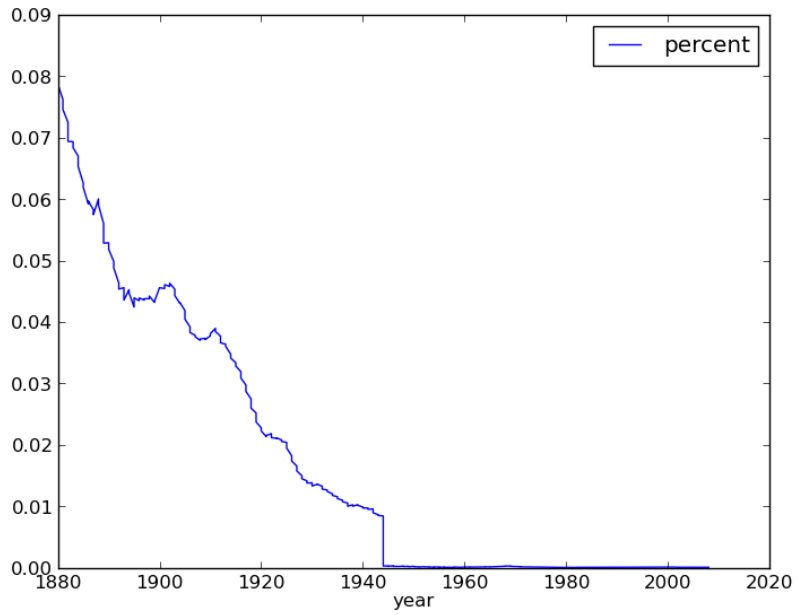
Menggunakan data *year* sebagai *index*:

```
>>> william = william.set_index(['year'])
>>> william.head()
       percent
year
1880  0.080511
1881  0.078712
1882  0.076191
1883  0.074558
1884  0.072475
```

Membuat `plot` sederhana dari data di atas:

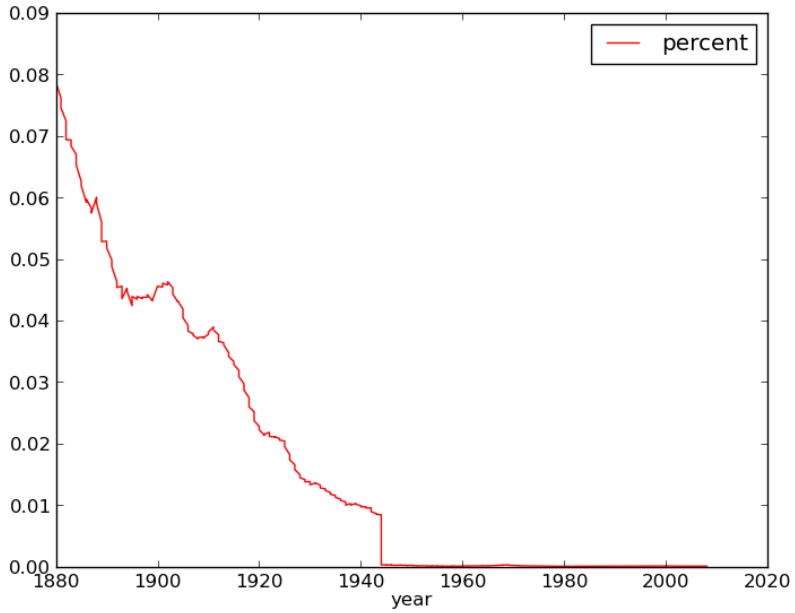
```
>>> import matplotlib.pyplot as plt
>>> from pylab import plot
>>> william.plot()
>>> plt.show()
```

Maka akan muncul *window* grafik dari data kita:



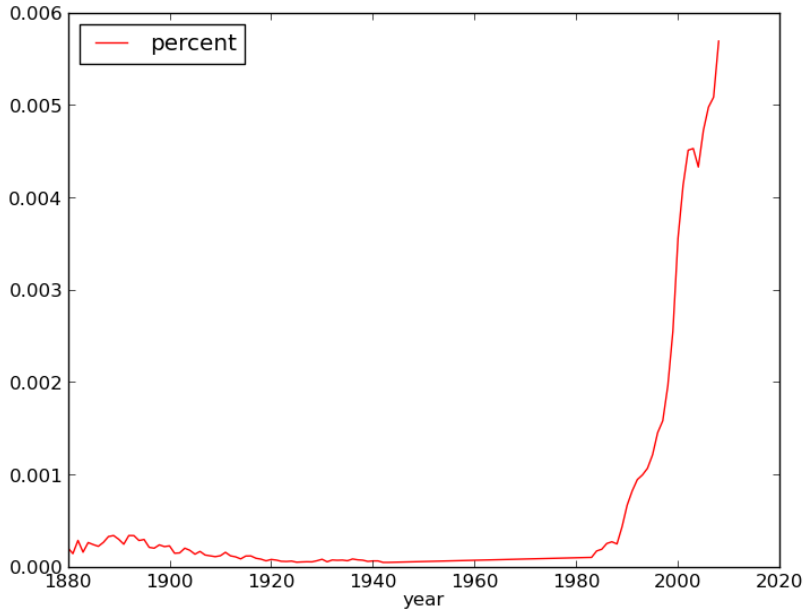
Kita bisa mengubah warna grafik dengan menambahkan *keyword argument* `color` pada fungsi `plot`:

```
>>> william.plot(color='Red')
>>> plt.show()
```



Contoh satu lagi, dengan pola *trend* yang berlawanan dengan contoh sebelumnya:

```
>>> chloe = names[names['name'] == 'Chloe']
>>> chloe = chloe.ix[0:, ['year', 'percent']]
>>> chloe = chloe.set_index(['year'])
>>> chloe.plot(color='Red')
>>> plt.show()
```



6.2 MLB Salaries Data

Data dapat diunduh di tautan berikut.[5]

6.2.1 Impor Data

Mari kita impor lihat informasi *metadata*-nya:

```
>>> import pandas as pd
>>> import matplotlib.pyplot as plt
>>> import pylab
>>> ds = pd.read_csv('mlbsalaries.csv')
>>> ds
<class 'pandas.core.frame.DataFrame'>
Int64Index: 19543 entries, 0 to 19542
Data columns:
Year        19543  non-null values
Player      19543  non-null values
Salary      19543  non-null values
Position    19543  non-null values
Team        19543  non-null values
dtypes: int64(2), object(3)
>>> ds.head()
   Year           Player   Salary           Position                Team
0  1988        Mike Witt  1400000            Pitcher  Los Angeles Angels
1  1988  George Hendrick   989333         Outfielder  Los Angeles Angels
```

[5] https://dl.dropboxusercontent.com/u/5052616/mlbsalaries.csv

```
2  1988      Chili Davis   950000         Outfielder  Los Angeles Angels
3  1988    Brian Downing   900000  Designated Hitter  Los Angeles Angels
4  1988        Bob Boone   883000            Catcher  Los Angeles Angels
>>> ds.tail()
       Year          Player  Salary        Position              Team
19538  2011  Gustavo Molina  455000         Catcher  New York Yankees
19539  2011       Ivan Nova  432900         Pitcher  New York Yankees
19540  2011    Colin Curtis  420400      Outfielder  New York Yankees
19541  2011   Eduardo Nunez  419300       Shortstop  New York Yankees
19542  2011   Reegie Corona  414000  Second Baseman  New York Yankees
```

Dari informasi *metadata* dapat kita lihat data tersebut terdiri dari 19.543 baris dan terdiri dari lima kolom (*Year, Player, Salary, Position* dan *Team*). Saatnya sekarang bermain-main dengan data! `:)`

6.2.2 Filters dan Subsets

Menampilkan baris ke-1000 sampai 1020:

```
>>> ds[1000:1020]
      Year           Player   Salary        Position              Team
1000  1989      Henry Cotto   250000      Outfielder  Seattle Mariners
1001  1989  Mickey Brantley   195000      Outfielder  Seattle Mariners
1002  1989   Scott Bankhead   185000         Pitcher  Seattle Mariners
1003  1989       Jerry Reed   160000         Pitcher  Seattle Mariners
1004  1989     Mike Jackson   155000         Pitcher  Seattle Mariners
1005  1989       Bill Swift   135000         Pitcher  Seattle Mariners
1006  1989      Gene Walter   120000         Pitcher  Seattle Mariners
1007  1989     Rey Quinones   100000       Shortstop  Seattle Mariners
1008  1989    Mike Campbell    85000         Pitcher  Seattle Mariners
1009  1989    Mike Schooler    85000         Pitcher  Seattle Mariners
1010  1989       Mario Diaz    72500  Second Baseman  Seattle Mariners
1011  1989    Rich Renteria    71000   Third Baseman  Seattle Mariners
1012  1989    Dave Cochrane    70000   First Baseman  Seattle Mariners
1013  1989   Edgar Martinez    69000   Third Baseman  Seattle Mariners
1014  1989      Greg Briley    68000      Outfielder  Seattle Mariners
1015  1989  Ken Griffey Jr.    68000      Outfielder  Seattle Mariners
1016  1989      Erik Hanson    68000         Pitcher  Seattle Mariners
1017  1989     Terry Taylor    68000         Pitcher  Seattle Mariners
1018  1989     Omar Vizquel    68000  Second Baseman  Seattle Mariners
1019  1989    Dwight Gooden  2416666         Pitcher     New York Mets
```

Menampilkan kolom *Player* baris ke-160 sampai 175:

```
>>> ds.Player[160:175]
160      Tony Pena Sr.
161      Tom Brunansky
162         Bob Horner
163          Danny Cox
164      Vince Coleman
165    Terry Pendleton
166         Ken Dayley
167      Denny Walling
168        Jose DeLeon
169       Jose Oquendo
170         Steve Lake
```

```
171       Todd Worrell
172        Tom Lawless
173       Greg Mathews
174          Curt Ford
Name: Player
```

Menampilkan data yang memiliki nilai *Year* 2011:

```
>>> ds.Year == 2001
0     False
1     False
2     False
3     False
4     False
5     False
6     False
7     False
8     False
9     False
10    False
11    False
12    False
13    False
14    False
...
19528    False
19529    False
19530    False
19531    False
19532    False
19533    False
19534    False
19535    False
19536    False
19537    False
19538    False
19539    False
19540    False
19541    False
19542    False
Name: Year, Length: 19543
```

Melihat *metadata* dari data dengan nilai *Year* == 2011:

```
>>> ds[ds.Year == 2011]
<class 'pandas.core.frame.DataFrame'>
Int64Index: 843 entries, 18700 to 19542
Data columns:
Year        843  non-null values
Player      843  non-null values
Salary      843  non-null values
Position    843  non-null values
Team        843  non-null values
dtypes: int64(2), object(3)
```

Terdapat 843 baris data dengan nilai *Year* == 2011.

Selanjutnya membuat *subsets* untuk data dengan nilai *Year* == 1988:

```
>>> ds_1988 = ds[ds.Year == 1988]
>>> ds_1988
<class 'pandas.core.frame.DataFrame'>
Int64Index: 686 entries, 0 to 685
Data columns:
Year        686  non-null values
Player      686  non-null values
Salary      686  non-null values
Position    686  non-null values
Team        686  non-null values
dtypes: int64(2), object(3)
```

Sekarang kita coba tampilkan apakah *datasets* sudah benar:

```
>>> ds_2011[30:35]
       Year           Player    Salary    Position                Team
18730  2011    Jordan Walden    414000     Pitcher  Los Angeles Angels
18731  2011       Carlos Lee  19000000  Outfielder      Houston Astros
18732  2011      Brett Myers   8000000     Pitcher      Houston Astros
18733  2011  Wandy Rodriguez   7500000     Pitcher      Houston Astros
18734  2011     Hunter Pence   6900000  Outfielder      Houston Astros
```

Sekarang mari kita asosiasikan *datasets* ini dengan sebuah *label*:

```
>>> ds_1988 = ds[ds.Year == 1988]
>>> ds_2011 = ds[ds.Year == 2011]
```

Sekali lagi mari kita lakukan *filtering* berdasar kolom *Player* yang bernilai *Alex Rodriguez* saja:

```
>>> ds[ds.Player == 'Alex Rodriguez']
       Year          Player    Salary       Position              Team
6493   1996  Alex Rodriguez    442334      Shortstop  Seattle Mariners
7260   1997  Alex Rodriguez   1012500      Shortstop  Seattle Mariners
8068   1998  Alex Rodriguez   2112500      Shortstop  Seattle Mariners
8949   1999  Alex Rodriguez   3112500      Shortstop  Seattle Mariners
9831   2000  Alex Rodriguez   4362500      Shortstop  Seattle Mariners
10903  2001  Alex Rodriguez  22000000      Shortstop     Texas Rangers
11752  2002  Alex Rodriguez  22000000      Shortstop     Texas Rangers
12589  2003  Alex Rodriguez  22000000      Shortstop     Texas Rangers
13665  2004  Alex Rodriguez  22000000  Third Baseman  New York Yankees
14499  2005  Alex Rodriguez  26000000  Third Baseman  New York Yankees
15320  2006  Alex Rodriguez  21680727  Third Baseman  New York Yankees
16168  2007  Alex Rodriguez  22708525  Third Baseman  New York Yankees
17023  2008  Alex Rodriguez  28000000  Third Baseman  New York Yankees
17846  2009  Alex Rodriguez  33000000  Third Baseman  New York Yankees
18675  2010  Alex Rodriguez  33000000  Third Baseman  New York Yankees
19513  2011  Alex Rodriguez  32000000  Third Baseman  New York Yankees
```

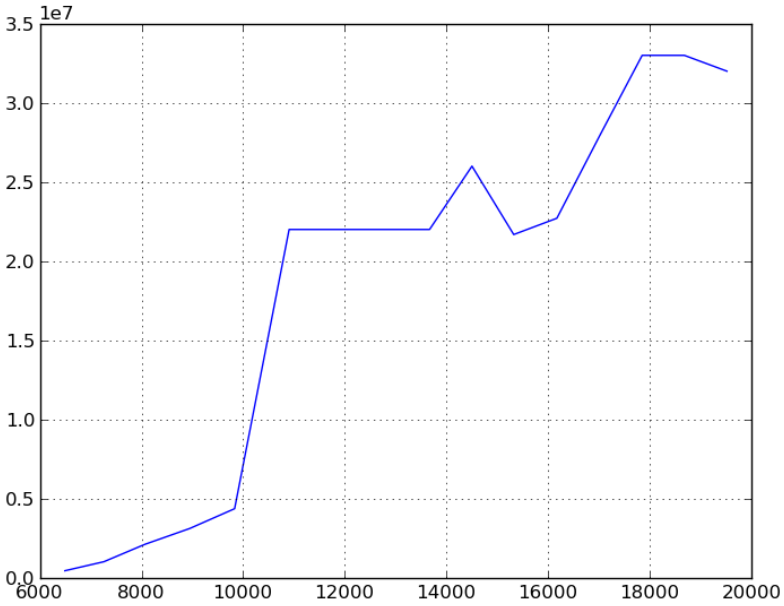
Berikan *label* pada *datasets* di atas dengan nama `ds_alex`:

```
>>> ds_alex = ds[ds.Player == 'Alex Rodriguez']
```

6.2.3 Statistik Dasar dan Menggambar Plot

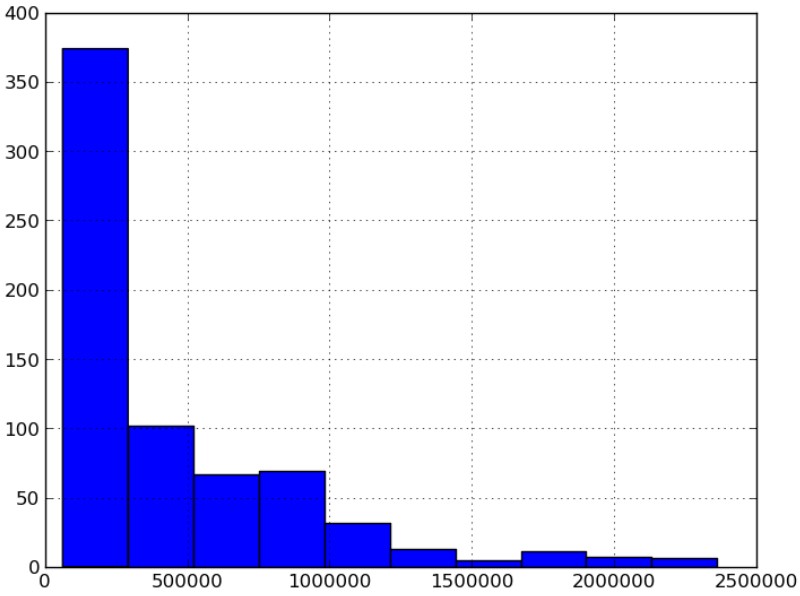
Mari kita tampilkan emphplot dari *datasets* `ds_alex` dari kolom *Salary*:

```
>>> ds_alex.Salary.plot()
<matplotlib.axes.AxesSubplot object at 0x9913eac>
>>> plt.show()
```



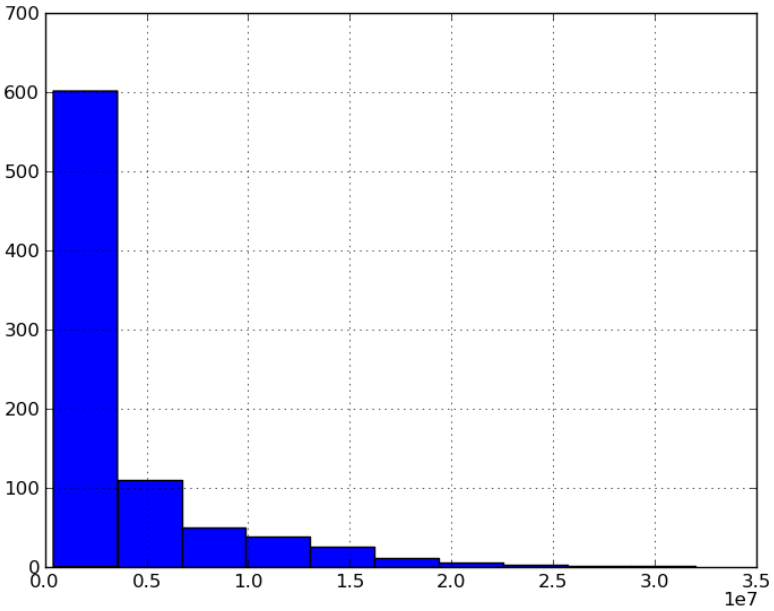
Membuat *histogram* dari `ds_1988`:

```
>>> ds_1988.Salary.hist()
<matplotlib.axes.AxesSubplot object at 0x9913b0c>
>>> plt.show()
```



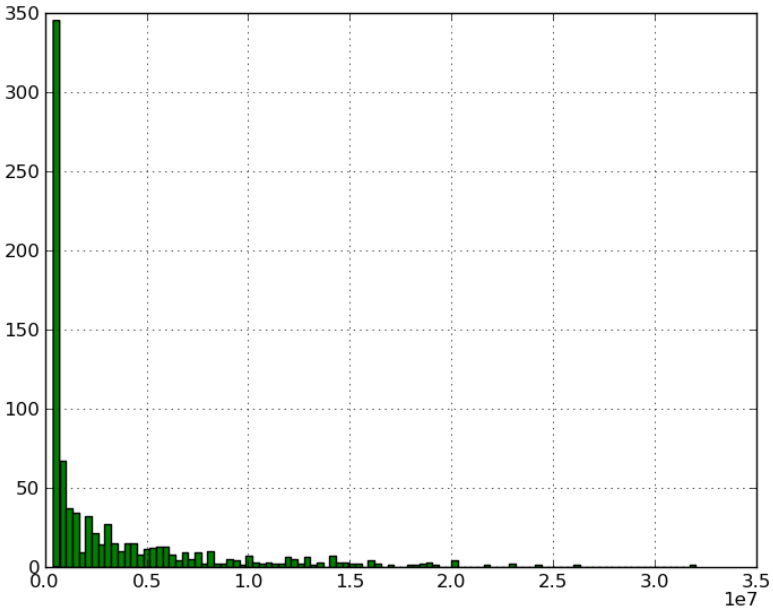
Sekali lagi membuat *histogram* untuk *subset* `ds_2011`:

```
>>> ds_2011.Salary.hist()
<matplotlib.axes.AxesSubplot object at 0x99bf7cc>
>>> plt.show()
```



Mari kita tambahkan *parameter* `bins` dan `fc` pada fungsi `hist()` kita, dan lihat apa yang terjadi:

```
>>> ds_2011.Salary.hist(bins=100, fc='green')
<matplotlib.axes.AxesSubplot object at 0xa22e2ac>
>>> plt.show()
```



Melihat statistik dasar dari *datasets* `ds_1988`:

```
>>> ds_1988.describe()
       Year          Salary
count   686      686.000000
mean   1988   429763.586006
std       0   457506.505135
min    1988    62500.000000
25%    1988    90000.000000
50%    1988   235000.000000
75%    1988   635000.000000
max    1988  2360714.000000
```

Statistik dasar untuk *subsets* `ds_2011`:

```
>>> ds_2011.describe()
       Year           Salary
count   843       843.000000
mean   2011   3305054.912218
std       0   4534742.121161
min    2011    414000.000000
25%    2011    430325.000000
50%    2011   1175000.000000
75%    2011   4306250.000000
max    2011  32000000.000000
```

Kalau ingin menampilkan statistik hanya untuk kolom *Salary* saja, gunakan *syntax* berikut:

```
>>> ds_1988.Salary.describe()
count        686.000000
mean      429763.586006
std       457506.505135
min        62500.000000
25%        90000.000000
50%       235000.000000
75%       635000.000000
max      2360714.000000
>>> ds_2011.Salary.describe()
count         843.000000
mean      3305054.912218
std       4534742.121161
min        414000.000000
25%        430325.000000
50%       1175000.000000
75%       4306250.000000
max      32000000.000000
```

6.2.4 Data Selection

Seperti dibahas pada bagian sebelumnya, kita dapat melakukan *data selection* pada *DataFrame*, sebagai contoh penulis ingin menampilkan data yang berasal dari pemain bernama *Derek Jeter*:

```
>>> import pandas as pd
>>> sal = pd.read_csv('mlbsalaries.csv')
>>> sal.Player == 'Derek Jeter'
0     False
1     False
2     False
3     False
4     False
5     False
6     False
7     False
8     False
9     False
10    False
11    False
12    False
13    False
14    False
...
19528    False
19529    False
19530    False
19531    False
19532    False
19533    False
19534    False
19535    False
19536    False
19537    False
19538    False
19539    False
```

```
19540    False
19541    False
19542    False
Name: Player, Length: 19543
>>> sal[sal.Player == 'Derek Jeter'].head()
       Year       Player    Salary   Position              Team
6943   1996  Derek Jeter    120000  Shortstop  New York Yankees
7715   1997  Derek Jeter    540000  Shortstop  New York Yankees
8560   1998  Derek Jeter    750000  Shortstop  New York Yankees
9465   1999  Derek Jeter   5000000  Shortstop  New York Yankees
10303  2000  Derek Jeter  10000000  Shortstop  New York Yankees
>>> sal[sal.Player == 'Derek Jeter']
       Year       Player    Salary   Position              Team
6943   1996  Derek Jeter    120000  Shortstop  New York Yankees
7715   1997  Derek Jeter    540000  Shortstop  New York Yankees
8560   1998  Derek Jeter    750000  Shortstop  New York Yankees
9465   1999  Derek Jeter   5000000  Shortstop  New York Yankees
10303  2000  Derek Jeter  10000000  Shortstop  New York Yankees
11155  2001  Derek Jeter  12600000  Shortstop  New York Yankees
12006  2002  Derek Jeter  14600000  Shortstop  New York Yankees
12833  2003  Derek Jeter  15600000  Shortstop  New York Yankees
13666  2004  Derek Jeter  18600000  Shortstop  New York Yankees
14500  2005  Derek Jeter  19600000  Shortstop  New York Yankees
15321  2006  Derek Jeter  20600000  Shortstop  New York Yankees
16169  2007  Derek Jeter  21600000  Shortstop  New York Yankees
17025  2008  Derek Jeter  21600000  Shortstop  New York Yankees
17847  2009  Derek Jeter  21600000  Shortstop  New York Yankees
18677  2010  Derek Jeter  22600000  Shortstop  New York Yankees
19518  2011  Derek Jeter  14729364  Shortstop  New York Yankees
```

Contoh berikutnya, misal penulis ingin menampilkan data pemain yang memiliki posisi sebagai *Shortstop*, dan berasal dari tim *Boston Red Sox* atau time *New York Yankees* dan memiliki *Salary* lebih dari atau sama dengan 5000000.

```
>>> criteria = ((sal.Position == 'Shortstop') & ((sal.Team == 'Boston Red Sox')
... | (sal.Team == 'New York Yankees')) & (sal.Salary >= 5000000))
>>> sal[criteria]
       Year              Player    Salary   Position              Team
9465   1999        Derek Jeter   5000000  Shortstop  New York Yankees
10303  2000        Derek Jeter  10000000  Shortstop  New York Yankees
10990  2001  Nomar Garciaparra   7250000  Shortstop    Boston Red Sox
11155  2001        Derek Jeter  12600000  Shortstop  New York Yankees
11841  2002  Nomar Garciaparra   9000000  Shortstop    Boston Red Sox
12006  2002        Derek Jeter  14600000  Shortstop  New York Yankees
12674  2003  Nomar Garciaparra  11000000  Shortstop    Boston Red Sox
12833  2003        Derek Jeter  15600000  Shortstop  New York Yankees
13500  2004  Nomar Garciaparra  11500000  Shortstop    Boston Red Sox
13666  2004        Derek Jeter  18600000  Shortstop  New York Yankees
14335  2005     Edgar Renteria   8000000  Shortstop    Boston Red Sox
14500  2005        Derek Jeter  19600000  Shortstop  New York Yankees
15321  2006        Derek Jeter  20600000  Shortstop  New York Yankees
16004  2007         Julio Lugo   8250000  Shortstop    Boston Red Sox
16169  2007        Derek Jeter  21600000  Shortstop  New York Yankees
16867  2008         Julio Lugo   9250000  Shortstop    Boston Red Sox
17025  2008        Derek Jeter  21600000  Shortstop  New York Yankees
17684  2009         Julio Lugo   9250000  Shortstop    Boston Red Sox
```

```
17847  2009        Derek Jeter  21600000  Shortstop  New York Yankees
18522  2010      Marco Scutaro   5500000  Shortstop    Boston Red Sox
18677  2010        Derek Jeter  22600000  Shortstop  New York Yankees
19367  2011      Marco Scutaro   5500000  Shortstop    Boston Red Sox
19518  2011        Derek Jeter  14729364  Shortstop  New York Yankees
```

6.3 Kurs Dollar Terhadap Rupiah 2001-2013

Data kami peroleh dari situs resmi Bank Sentral Republik Indonesia,[6] yang kemudian dilakukan *parsing* terhadap data tersebut.[7]

Langkah selanjutnya, kita import kemudian kita lihat statistik dari data tersebut kemudian kita gambar *plot*-nya:

```
Python 2.7.4 (default, Apr 19 2013, 18:32:33)
[GCC 4.7.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import pandas as pd
>>> import matplotlib.pyplot as plt
>>> df = pd.read_csv('kurs.csv')
>>> df
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3108 entries, 0 to 3107
Data columns:
Nilai      3108  non-null values
Jual       3108  non-null values
Beli       3108  non-null values
Tanggal    3108  non-null values
dtypes: float64(3), object(1)
>>> df.head()
   Nilai   Jual   Beli      Tanggal
0      1  11492  11378  23 Sep 2013
1      1  11409  11295  20 Sep 2013
2      1  11334  11222  19 Sep 2013
3      1  11549  11435  18 Sep 2013
4      1  11508  11394  17 Sep 2013
>>> df.tail()
      Nilai  Jual  Beli      Tanggal
3103      1  9467  9373  30 Jan 2001
3104      1  9517  9423  29 Jan 2001
3105      1  9472  9378  26 Jan 2001
3106      1  9502  9408  25 Jan 2001
3107      1  9440  9346  24 Jan 2001
>>> df.describe()
       Nilai          Jual          Beli
count   3108   3108.000000   3108.000000
mean       1   9451.216538   9357.180171
std        0    735.435616    728.082895
min        1   8206.000000   8124.000000
25%        1   9037.500000   8947.500000
50%        1   9259.000000   9167.000000
75%        1   9708.000000   9612.000000
max        1  12462.000000  12338.000000
```

[6] http://www.bi.go.id/web/id/Moneter/Kurs+Bank+Indonesia/Kurs+Transaksi/

[7] silakan lihat lampiran untuk melihat bagaimana kami melakukan parsing terhadap data tersebut.

```
>>> df = df.sort_index(ascending=False)
>>> df.head()
      Nilai  Jual  Beli      Tanggal
3107      1  9440  9346  24 Jan 2001
3106      1  9502  9408  25 Jan 2001
3105      1  9472  9378  26 Jan 2001
3104      1  9517  9423  29 Jan 2001
3103      1  9467  9373  30 Jan 2001
>>> df.tail()
   Nilai   Jual   Beli      Tanggal
4      1  11508  11394  17 Sep 2013
3      1  11549  11435  18 Sep 2013
2      1  11334  11222  19 Sep 2013
1      1  11409  11295  20 Sep 2013
0      1  11492  11378  23 Sep 2013
>>> df = df.set_index('Tanggal')
>>> df.head()
             Nilai  Jual  Beli
Tanggal
24 Jan 2001      1  9440  9346
25 Jan 2001      1  9502  9408
26 Jan 2001      1  9472  9378
29 Jan 2001      1  9517  9423
30 Jan 2001      1  9467  9373
>>> jual = df['Jual']
>>> jual.plot()
<matplotlib.axes.AxesSubplot object at 0xa9ab6ac>
>>> plt.show()
```

Berikut tampilan *plot* dari kurs jual menurut data dari BI:

6.4 Analisis Trend IHSG Periode 1997-2013

Data yang digunakan bersumber dari situs *Yahoo Finance*, yang mana datanya sudah berbentuk `.csv` dan siap diunduh.[8]

```
>>> df = pd.read_csv('table.csv', index_col='Date', parse_dates=True)
>>> df.head()
              Open     High      Low    Close      Volume  Adj Close
Date
2013-09-24  4548.55  4574.58  4443.41  4460.41  4098431200    4460.41
2013-09-23  4526.80  4562.86  4512.21  4562.86  3482981600    4562.86
2013-09-20  4655.52  4669.72  4576.32  4583.83  4387711200    4583.83
2013-09-19  4576.57  4791.77  4576.57  4670.73  8323665600    4670.73
2013-09-18  4501.96  4505.08  4448.08  4463.25  4909932800    4463.25
>>> df.tail()
             Open    High     Low   Close  Volume  Adj Close
Date
1997-07-07  736.62  739.83  735.77  738.01       0     738.01
1997-07-04  735.83  736.60  733.89  736.60       0     736.60
1997-07-03  730.48  735.49  729.56  735.49       0     735.49
1997-07-02  731.75  734.02  727.45  730.16       0     730.16
1997-07-01  725.03  732.52  724.99  731.62       0     731.62
>>> df['Adj Close']
Date
2013-09-24    4460.41
2013-09-23    4562.86
2013-09-20    4583.83
2013-09-19    4670.73
2013-09-18    4463.25
2013-09-17    4517.62
2013-09-16    4522.24
2013-09-13    4375.54
2013-09-12    4356.60
2013-09-11    4349.42
2013-09-10    4358.14
2013-09-09    4191.26
2013-09-06    4072.35
2013-09-05    4050.86
2013-09-04    4073.46
...
1997-07-22    711.44
1997-07-21    712.40
1997-07-18    724.00
1997-07-16    723.50
1997-07-15    722.21
1997-07-14    722.50
1997-07-11    723.42
1997-07-10    729.15
1997-07-09    738.14
1997-07-08    740.83
1997-07-07    738.01
1997-07-04    736.60
1997-07-03    735.49
1997-07-02    730.16
1997-07-01    731.62
```

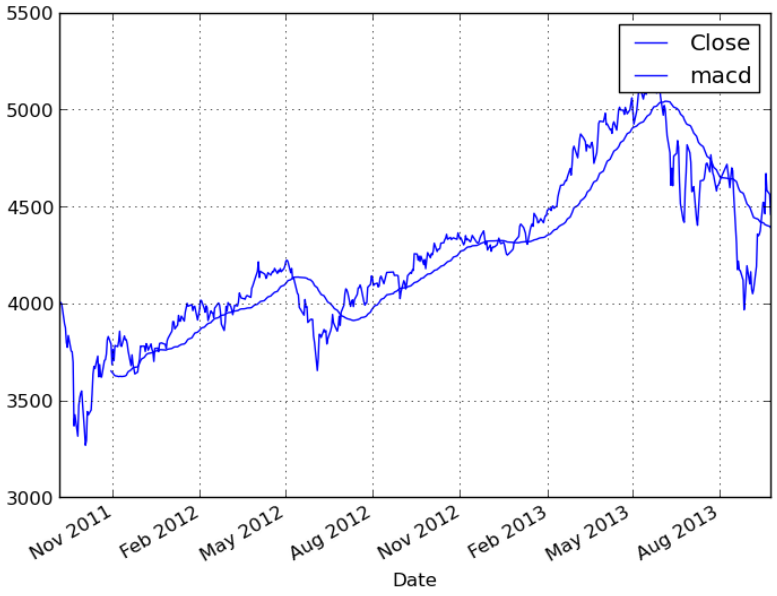
[8] Tautan ke data historis IHSG http://finance.yahoo.com/q/hp?s=%5EJKSE+Historical+Prices

```
Name: Adj Close, Length: 3945
>>> df = df.sort_index(ascending=True)
>>> df.head()
              Open    High     Low   Close  Volume  Adj Close
Date
1997-07-01  725.03  732.52  724.99  731.62       0     731.62
1997-07-02  731.75  734.02  727.45  730.16       0     730.16
1997-07-03  730.48  735.49  729.56  735.49       0     735.49
1997-07-04  735.83  736.60  733.89  736.60       0     736.60
1997-07-07  736.62  739.83  735.77  738.01       0     738.01
>>> df.tail()
               Open     High      Low    Close      Volume  Adj Close
Date
2013-09-18  4501.96  4505.08  4448.08  4463.25  4909932800    4463.25
2013-09-19  4576.57  4791.77  4576.57  4670.73  8323665600    4670.73
2013-09-20  4655.52  4669.72  4576.32  4583.83  4387711200    4583.83
2013-09-23  4526.80  4562.86  4512.21  4562.86  3482981600    4562.86
2013-09-24  4548.55  4574.58  4443.41  4460.41  4098431200    4460.41
>>> df['Adj Close'].plot()
<matplotlib.axes.AxesSubplot object at 0xac13f2c>
>>> plt.show()
```

Sekarang mari kita coba lakukan analisis MACD (*moving average covergence divergence*) terhadap 500 data terakhir dari tanggal sekarang.

```
>>> df = df.tail(500)
>>> df.head()
               Open     High      Low    Close      Volume  Adj Close
Date
2011-09-06  3865.53  3891.03  3829.61  3889.97  2980622400    3889.97
```

```
2011-09-07  3890.52  4002.40  3890.52  4001.43  4395936400    4001.43
2011-09-08  4001.71  4021.76  3978.34  4005.39  3759277600    4005.39
2011-09-09  4004.46  4028.48  3991.58  3998.50           0    3998.50
2011-09-12  3997.22  3997.22  3880.70  3896.12  2723431000    3896.12
>>> df.tail()
              Open     High      Low    Close      Volume  Adj Close
Date
2013-09-18  4501.96  4505.08  4448.08  4463.25  4909932800    4463.25
2013-09-19  4576.57  4791.77  4576.57  4670.73  8323665600    4670.73
2013-09-20  4655.52  4669.72  4576.32  4583.83  4387711200    4583.83
2013-09-23  4526.80  4562.86  4512.21  4562.86  3482981600    4562.86
2013-09-24  4548.55  4574.58  4443.41  4460.41  4098431200    4460.41
>>> macd = pd.rolling_mean(df['Adj Close'], 40)
>>> df['Adj Close'].plot(label='Close')
<matplotlib.axes.AxesSubplot object at 0xabff86c>
>>> macd.plot(label='macd')
<matplotlib.axes.AxesSubplot object at 0xabff86c>
>>> plt.legend()
<matplotlib.legend.Legend object at 0xaa9d50c>
>>> plt.show()
```



7 Lampiran

7.1 Parsing Data Kurs dari Situs BI

Langkah pertama adalah dengan menyalin tabel `HTML` yang berisi kurs, kemudian disimpan dalam berkas `.txt`. Setelah itu, kami hilangkan spasi, ganti

dengan koma dan simpan dalam berkas `.csv` agar mudah dibaca oleh `Pandas`. Berikut kode python yang kami pakai:

```
>>> import re
>>> with open('kurs.txt', 'rb') as f:
...     with open('kurs.csv', 'wb') as w:
...         for l in f:
...             newline = re.sub(re.compile(r"\s+"), ',', l.strip(), count=2)
...             w.write(re.sub(re.compile(r"\s+"), ',', newline, count=1))
...
>>>
```

Tampilan data yang sudah di*parsing*:[9]

```
File Edit Options Buffers Tools Help
Nilai,Jual,Beli,Tanggal
1.00,11492.00,11378.00,23 Sep 2013
1.00,11409.00,11295.00,20 Sep 2013
1.00,11334.00,11222.00,19 Sep 2013
1.00,11549.00,11435.00,18 Sep 2013
1.00,11508.00,11394.00,17 Sep 2013
1.00,11480.00,11366.00,16 Sep 2013
1.00,11452.00,11338.00,13 Sep 2013
1.00,11551.00,11437.00,12 Sep 2013
1.00,11495.00,11381.00,11 Sep 2013
1.00,11236.00,11124.00,10 Sep 2013
1.00,11244.00,11132.00,9 Sep 2013
1.00,11256.00,11144.00,6 Sep 2013
1.00,11181.00,11069.00,5 Sep 2013
1.00,11148.00,11038.00,4 Sep 2013
1.00,11038.00,10928.00,3 Sep 2013
1.00,10977.00,10867.00,2 Sep 2013
1.00,10979.00,10869.00,30 Aug 2013
1.00,10991.00,10881.00,29 Aug 2013
1.00,11005.00,10895.00,28 Aug 2013
1.00,10937.00,10829.00,27 Aug 2013
-UU-:----F1  kurs.csv       Top L1     (Fundamental) --------------------------
Wrote /home/banteng/Dropbox/kurs.csv
```

7.2 Install `matplotlib` Dalam Virtualenv

Memasang modul `matplotlib` dalam sistem *virtualenv* sedikit *tricky*, berikut cara penulis yang sudah dicoba dan berhasil:

```
sudo apt-get install build-essentials
sudo pip install virtualenv
sudo apt-get install libpng++-dev
sudo apt-get install libfreetype6-dev
virtualenv ihsg
. bin/activate
pip install numpy
pip install matplotlib
```

[9] Tautan ke data yang sudah diparsing https://dl.dropboxusercontent.com/u/5052616/kurs.csv

Perlu diingat bahwa *module* `matplotlib` di dalam *virtualenv* tidak mendukung fungsi `show()` (setidaknya penulis belum berhasil), sehingga satu-satunya jalan keluar adalah menyimpan hasil *plot* kedalam berkas menggunakan fungsi `savefig()`.

7.3 Contoh Script CSV Downloader, Analyze and Save the Image

Cara kerja *script* ini adalah pertama mengunduh berkas `.csv`, kemudian melakukan analisis teknikal, dalam hal ini menggunakan *MACD* dan *Bollinger Bands*, kemudian menyimpan hasil gambar *plot* ke dalam berkas.

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

import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
from datetime import datetime, timedelta
import re
import time
import urllib2
import sys

# download data

def analyze(stock):
    #url = 'http://finance.yahoo.com/q/hp?s=%5EJKSE+Historical+Prices'
    url = 'http://finance.yahoo.com/q/hp?s='+stock+'+Historical+Prices'
    try:
        html = urllib2.urlopen(url).read()
    except:
        print 'error accessing the website'

    pattern = re.compile(r"http://ichart.*\.csv\"")
    m = re.search(pattern, html).group()

    with open('table.csv', 'wb') as f:
        f.write(urllib2.urlopen(m).read())

    df = pd.read_csv('table.csv', index_col='Date', parse_dates=True)
    df = df.sort_index(ascending=True)
    df = df.tail(240)

    # analysis
    # macd
    macd = pd.rolling_mean(df['Adj Close'], 12)
    # bollinger bands
    # https://github.com/arvindevo/MachineLearningForTrading/blob/master/bollingerbands.py
    movavg = pd.rolling_mean(df['Adj Close'], 20, min_periods=20)
    movstddev = pd.rolling_std(df['Adj Close'], 20, min_periods=20)
    upperband = movavg + 2*movstddev
    lowerband = movavg - 2*movstddev

    # plot settings
```

```
    matplotlib.rcParams.update({'font.size': 8})
    s = datetime.now()

    # begin plot
    df['Adj Close'].plot(label='Close')
    macd.plot(label='macd', linestyle='--', color='r')
    upperband.plot(color='green')
    lowerband.plot(color='green')

    plt.title('Analisis Teknikal MACD dan Bollinger Bands IHSG')
    plt.legend(['Adjusted Close', 'MACD', 'Upper Bollinger', 'Lower Bollinger'])
    plt.xlim(s - timedelta(days=250), s + timedelta(days=7))
    plt.ylabel('Adjusted Close')
    plt.xlabel('Tanggal')

    # save
    plt.savefig('ihsg.png')

if __name__ == '__main__':
    stock = sys.argv[1]
    analyze(stock)
```

Simpan *script* ini ke dalam sebuah berkas, misal `sahamta.py`, kemudian jalankan dengan mengetikkan `python sahamta.py BBRI`. Kode BBRI ini dapat Anda ganti sesuai dengan kode saham perusahaan yang Anda inginkan sesuai dengan kode yang ada di Yahoo Finance, misal GGRM, dan sebagainya.

```
$ python sahamta.py BBRI
```

7.4 Install TA-Lib Module

Module `TA-Lib` berisi ratusan analisis teknikal yang siap dipakai hanya dengan semudah memanggil sebuah fungsi. Modul ini dapat diunduh dari sourceforge[10], kemudian jalankan perintah berikut:

```
$ tar zxvf ta-lib-0.4.0-src.tar.gz
$ ./configure --prefix=/usr
$ make
$ sudo make install
```

Setelah proses di atas selesai tanpa pesan kesalahan, selanjutnya kita memasang python `wrapper`[11] di direktori `virtualenv` kita:

```
pip install cython
pip install TA-Lib
```

Modul sekarang siap digunakan... Untuk dokumentasi bisa menuju ke tautan berikut:

http://mrjbq7.github.io/ta-lib/index.html

[10] http://prdownloads.sourceforge.net/ta-lib/ta-lib-0.4.0-src.tar.gz
[11] https://github.com/mrjbq7/ta-lib

7.4.1 Contoh Penggunaan

7.5 Modul prettyplotlib

Sesuai dengan namanya, `prettyplotlib`[12] merupakan modul untuk membuat tampilan *plot* menjadi lebih menarik dan lebih mudah dibanding menggunakan modul `matplotlib`.

8 Referensi

Pandas Lighning Tutorials http://www.youtube.com/user/malpas0/videos

Baby Names Data https://github.com/hadley/data-baby-names

TA-Lib Developer http://mrjbq7.github.io/ta-lib/index.html

Contoh Data Kurs IDR/USD https://dl.dropboxusercontent.com/u/5052616/kurs.csv

Data MLB Player Salaries https://dl.dropboxusercontent.com/u/5052616/mlbsalaries.csv

Data Kurs Transaksi BI http://www.bi.go.id/web/id/Moneter/Kurs+Bank+Indonesia/Kurs+Transaksi/

Sentdex Youtube Channel http://www.youtube.com/user/sentdex

[12] http://olgabot.github.io/prettyplotlib/