Bab 5: Pandas

In [ ]:

```
import pandas as pd
from pandas import Series, DataFrame
```

Series dalam Package Pandas

In [ ]:

```
#Series
#Objek di pandas itu menunjukkan suatu
obj = pd.Series([4, 7, -5, 3])
obj
#Pada bagian kiri menunjukkan index, sedangkan bagian kanan menunjukkan values
```

Out[ ]:

```
0    4
1    7
2   -5
3    3
dtype: int64
```

In [ ]:

```
obj.values
#output yang dihasilkan menunjukkan value yang sama dengan sebelah kanan yang dihasilkan
dari syntax di atas
```

Out[ ]:

```
array([ 4,  7, -5,  3])
```

In [ ]:

```
obj.index #seperti range(4)
```

Out[ ]:

```
RangeIndex(start=0, stop=4, step=1)
```

In [ ]:

```
#Identifikasi setiap data poin
obj2 = pd.Series([4, 7, -5, 3], index =['d', 'b', 'a', 'c'])
obj2
```

Out[ ]:

```
d    4
b    7
a   -5
c    3
dtype: int64
```

In [89]:

```
#bedanya dengan enumerate
for i, a in enumerate(obj2):
  print(i, a)
```

```
0 6
1 7
2 -5
3 3
```

In [ ]:

```
obj2.values
```

Out[ ]:

```
array([ 4,  7, -5,  3])
```

In [ ]:

```
obj2.index
#index nya bukan jadi literator lagi, tapi kalo di define sendiri jadi another array
```

Out[ ]:

```
Index(['d', 'b', 'a', 'c'], dtype='object')
```

In [87]:

```
#memanggil suatu value dengan menggunakan index
obj2['a'], #atau
obj2.a
```

Out[87]:

```
-5
```

In [ ]:

```
#mengganti value pada series menggunakan index dan mengganti nama index berdasarkan uruta
n
obj2['d'] = 6
obj2[['c', 'a', 'd']]
```

Out[ ]:

```
c    3
a   -5
d    6
dtype: int64
```

In [91]:

```
#slice series
obj[:3]
```

Out[91]:

```
0    4
1    7
2   -5
dtype: int64
```

In [94]:

```
#Boolean
obj2[obj2 > 0] #tipe int64, mengeluarkan subhimpunan yang memenuhi pertidaksamaan
```

Out[94]:

```
d    6
b    7
c    3
dtype: int64
```

In [93]:

```
#Boolean
obj2 > 0 #tipe datanya boolean, tetapi output yang dihasilkan adalah bukan subhimpunan ya
ng memenuhi pertidaksamaan
```

Out[93]:

```
d     True
b     True
a    False
c     True
dtype: bool
```

In [ ]:

```
#menggunakan NumPy dalam operasi matematika
obj2 * 2
```

Out[ ]:

```
d    12
b    14
a   -10
c     6
dtype: int64
```

In [ ]:

```
#mengganti series menjadi bentuk eksponensial
import numpy as np
np.exp(obj2)
```

Out[ ]:

```
d     403.428793
b    1096.633158
a       0.006738
c      20.085537
dtype: float64
```

In [ ]:

```
#mapping dengan dict
'b' in obj2
'e' in obj2
```

Out[ ]:

```
False
```

In [ ]:

```
#memanfaatkan dict dalam pandas
sdata = {'Ohio' : 35000, 'Texas' : 71000, 'Oregon' : 16000, 'Utah' : 5000}
obj3 = pd.Series(sdata) #convert dari dict, menjadi Pandas Series
obj3
```

Out[ ]:

```
Ohio      35000
Texas     71000
Oregon    16000
Utah       5000
dtype: int64
```

In [ ]:

```
#membuat data baru/mengganti nama index
states = ['California', 'Ohio', 'Oregon', 'Texas']
obj4 = pd.Series(sdata, index=states)
obj4
```

Out[ ]:

```
California        NaN
Ohio          35000.0
Oregon        16000.0
Texas         71000.0
dtype: float64
```

In [ ]:

```
#Lihat index states California memiliki value NaN
pd.isnull(obj4)
```

Out[ ]:

```
California     True
Ohio          False
Oregon        False
Texas         False
dtype: bool
```

In [ ]:

```
#Lihat index states California memiliki value NaN
pd.notnull(obj4)
```

Out[ ]:

```
California    False
Ohio           True
Oregon         True
Texas          True
dtype: bool
```

In [ ]:

```
#Lihat index states California memiliki value NaN
obj4.isnull()
```

Out[ ]:

```
California     True
Ohio          False
Oregon        False
Texas         False
dtype: bool
```

In [ ]:

```
#Operasi Aritmatika
obj3
```

Out[ ]:

```
Ohio      35000
Texas     71000
Oregon    16000
Utah       5000
dtype: int64
```

In [ ]:

```
#Operasi Aritmatika
obj4
```

Out[ ]:

```
California        NaN
Ohio          35000.0
Oregon        16000.0
Texas         71000.0
dtype: float64
```

In [ ]:

```
#Menjumlahkan value obj3 dan obj4
obj3+obj4
```

Out[ ]:

```
California         NaN
Ohio           70000.0
Oregon         32000.0
```

```
Texas          142000.0
Utah                NaN
dtype: float64
```

In [ ]:

```
#Memberikan nama Pandas Series dan nama indexnya
obj4.name = 'Population'
obj4.index.name = 'state'
obj4
```

Out[ ]:

```
state
California         NaN
Ohio           35000.0
Oregon         16000.0
Texas          71000.0
Name: Population, dtype: float64
```

DataFrame

In [ ]:

```
#DataFrame menunjukkan tabel data berbentuk persegi yang mengandung suatu koleksi perinta
h kolom, tiap kolomnya bisa memiliki tipe value yang berbeda (numeric, string, bool, etc)
#DataFrame memiliki baris dan kolom index
data = {'state' : ['Ohio', 'Ohio', 'Ohio', 'Nevada', 'Nevada', 'Nevada'],
        'year' : [2000, 2001, 2002, 2001, 2002, 2003],
        'pop' : [1.5, 1.7, 3.6, 2.4, 2.9, 3.2]}
frame = pd.DataFrame(data)
frame
```

Out[ ]:

	state	year	pop
0	Ohio	2000	1.5
1	Ohio	2001	1.7
2	Ohio	2002	3.6
3	Nevada	2001	2.4
4	Nevada	2002	2.9
5	Nevada	2003	3.2

In [ ]:

```
#Untuk DataFrame dengan ukuran yang besar, dapat menggunakan metode 'head' hanya mengamb
il 5 baris pertama
frame.head()
```

Out[ ]:

	state	year	pop
0	Ohio	2000	1.5
1	Ohio	2001	1.7
2	Ohio	2002	3.6
3	Nevada	2001	2.4
4	Nevada	2002	2.9

In [ ]:

```
#Jika melakukan spesifikasi kolom, kolom DaraFrame akan mengurutkan dengan:
```

```
pd.DataFrame(data, columns=['year', 'state', 'pop'])
```

Out[ ]:

	year	state	pop
0	2000	Ohio	1.5
1	2001	Ohio	1.7
2	2002	Ohio	3.6
3	2001	Nevada	2.4
4	2002	Nevada	2.9
5	2003	Nevada	3.2

In [ ]:

```
frame2 = pd.DataFrame(data, columns=['year', 'state', 'pop', 'debt'],
                      index=['one', 'two', 'three', 'four', 'five', 'six'])
frame2
#akan menghasilkan index yang berubah nama, dan debt yang tercatat 'NaN' karena value nya belum dimasukkan
```

Out[ ]:

	year	state	pop	debt
one	2000	Ohio	1.5	NaN
two	2001	Ohio	1.7	NaN
three	2002	Ohio	3.6	NaN
four	2001	Nevada	2.4	NaN
five	2002	Nevada	2.9	NaN
six	2003	Nevada	3.2	NaN

In [ ]:

```
frame2.columns
```

Out[ ]:

```
Index(['year', 'state', 'pop', 'debt'], dtype='object')
```

In [ ]:

```
#memanggil suatu value dalam DataFrame dengan menggunakan indexnya
frame2['state']
```

Out[ ]:

```
one        Ohio
two        Ohio
three      Ohio
four     Nevada
five     Nevada
six      Nevada
Name: state, dtype: object
```

In [ ]:

```
frame2.year
#karena value dari index year adalah integer, maka dituliskan tidak seperti jenis string
```

Out[ ]:

```
one       2000
two       2001
three     2002
four      2001
```

```
five      2002
six       2003
Name: year, dtype: int64
```

In [ ]:

```
#atribut 'loc'
frame2.loc['three'] #memanggil dengan index baris 'three'
```

Out[ ]:

```
year      2002
state     Ohio
pop        3.6
debt       NaN
Name: three, dtype: object
```

In [ ]:

```
#kolom bisa diubah dengan suatu assignment, misalnya kolom yang kosong 'debt' diisi deng
an suatu array
frame2['debt']=16.5
frame2
```

Out[ ]:

	year	state	pop	debt
one	2000	Ohio	1.5	16.5
two	2001	Ohio	1.7	16.5
three	2002	Ohio	3.6	16.5
four	2001	Nevada	2.4	16.5
five	2002	Nevada	2.9	16.5
six	2003	Nevada	3.2	16.5

In [ ]:

```
frame2['debt'] = np.arange(6.) #???
frame2
```

Out[ ]:

	year	state	pop	debt
one	2000	Ohio	1.5	0.0
two	2001	Ohio	1.7	1.0
three	2002	Ohio	3.6	2.0
four	2001	Nevada	2.4	3.0
five	2002	Nevada	2.9	4.0
six	2003	Nevada	3.2	5.0

In [ ]:

```
#Jika ingin memasukkan list atau array ke suatu kolom, maka panjang value harus sesuai de
ngan panjang dari DataFrame
#Jika ingin memasukkan Series,

val = pd.Series([-1.2, -1.5, -1.7], index=['two', 'four', 'five']) #value yang diisi ses
uai dengan index yang disebut
frame2['debt'] = val #index 'debt' diisi dengan val
frame2
```

Out[ ]:

	year	state	pop	debt

	year	state	pop	debt
one	2000	Ohio	1.5	NaN
two	2001	Ohio	1.7	-1.2
three	2002	Ohio	3.6	NaN
four	2001	Nevada	2.4	-1.5
five	2002	Nevada	2.9	-1.7
six	2003	Nevada	3.2	NaN

In [ ]:

```
#untuk menghapus kolom menggunakan keyword del

#tambah kolom baru dengan kolom yang berisi boolean
frame2['eastern'] = frame2.state == 'Ohio'
frame2
#atau kolom baru bisa menggunakan frame2.eastern
```

Out[ ]:

	year	state	pop	debt	eastern
one	2000	Ohio	1.5	NaN	True
two	2001	Ohio	1.7	-1.2	True
three	2002	Ohio	3.6	NaN	True
four	2001	Nevada	2.4	-1.5	False
five	2002	Nevada	2.9	-1.7	False
six	2003	Nevada	3.2	NaN	False

In [ ]:

```
#Hapus kolom eastern
del frame2['eastern'] #untuk hapus kolom eastern
frame2.columns #untuk menunjukkan nama index pada setiap kolom
```

Out[ ]:

```
Index(['year', 'state', 'pop', 'debt'], dtype='object')
```

In [ ]:

```
#bentuk lain dari data dict yang di dalamnya juga dict
pop = {'Nevada' : {2001: 2.4, 2002: 2.9,},
       'Ohio': {2000: 1.5, 2001: 1.7, 2002: 3.6}}
#bentuk DataFrame baru
frame3 = pd.DataFrame(pop)
frame3
```

Out[ ]:

	Nevada	Ohio
2001	2.4	1.7
2002	2.9	3.6
2000	NaN	1.5

In [ ]:

```
#Bisa melakukan transpose DataFrame (menukar baris dan kolom) mirip dengan syntax NumPy a
rray
frame3.T #atau bisa .transpose()
```

Out[ ]:

	2001	2002	2000

	2001	2002	2000
Nevada	2.4	2.9	NaN
Ohio	1.7	3.6	1.5

In [95]:

```
frame3.values
```

Out[95]:

```
array([[2.4, 1.7],
       [2.9, 3.6],
       [nan, 1.5]])
```

In [97]:

```
#reindex suatu metode dalam Pandas (tipe objects) dimana membuat objek baru dengan data s
ebelumnya ke index yang baru. ??
obj = pd.Series([4.5, 7.2, -5.3, 3.6], index = ['d', 'b', 'a', 'c' ])
obj
```

Out[97]:

```
d    4.5
b    7.2
a   -5.3
c    3.6
dtype: float64
```

In [98]:

```
#penggunaan axis
#kalo di buku:
obj = pd.Series(np.arange(5.), index=['a', 'b', 'c', 'd', 'e'])
obj
```

Out[98]:

```
a    0.0
b    1.0
c    2.0
d    3.0
e    4.0
dtype: float64
```

In [99]:

```
#penggunaan axis
#kalo di buku:
new_obj = obj.drop('c')
new_obj
```

Out[99]:

```
a    0.0
b    1.0
d    3.0
e    4.0
dtype: float64
```

In [ ]:

```
#penggunaan axis yang disarankan (hemat memori)
```