Modul 5
Pengenalan Dasar Data Processing dengan Numpy dan Pandas

Pada praktikum kali ini, kita akan mengenal beberapa tool yang sangat berguna untuk data processing. Ada dua tool yang akan dijelaskan yaitu: **Numpy** dan **Pandas**.

Overview:

1. **Deskripsi Numpy dan Pandas**
2. **Contoh Penggunaan Numpy**
    A. Operasi matematik
    B. Operasi Vektor/Matriks
    C. Slicing Array/Matrix
    D. Array Manipulation
3. **Contoh Penggunaan Pandas**
    A. Membaca dan Menulis file csv
    B. Indexing, Selecting, dan Assigning
    C. Data type
    D. Renaming
4. **Post Test Praktikum**

1. Deskripsi Numpy dan Pandas

Numpy

Numpy merupakan suatu library dalam bahasa Python yang berfungsi untuk mengolah data numerik. Kepanjangan dari Numpy adalah Numerical Python. Dengan Numpy, banyak sekali yang bisa kita lakukan, misalnya: operasi matematik, operasi pada vektor/matriks, slicing data, manipulasi array, dan sebagainya.

Pandas

Sederhananya, pandas dipakai untuk pengolahan **data terstuktur**. Apa saja yang bisa kita lakukan dengan pandas? Jawabnya ya banyak sekali. Diantaranya adalah eksplorasi data (data exploration), membersihkan data (data cleansing), dan data processing. Pandas memiliki struktur data table yang dinamakan sebagai **Data Frame**.

2. Contoh Penggunaan Numpy

A. Operasi Matematik

In [1]:

```
import numpy as np

# penjumlahan
a, b = 3, 4
c = np.add(a,b)
print("Jumlah dari a + b adalah",c)

# pengurangan
```

```
c = np.subtract(a,b)
print("Pengurangan dari a - b adalah",c)

# perkalian
c = np.multiply(a,b)
print("Perkalian dari a * b adalah",c)

# pembagian
c = np.divide(3,4)
print("Pembagian dari a / b adalah",c)

# eksponensial
c = np.exp(a) # exp adalah bilangan natural Euler. Nilai dari e adalah 2.718...
print("Hasil e^a adalah",c)

# logaritmik alami atau ln
c = np.log(b)
print("Ln b adalah",c)

# logaritmik basis 10
c = np.log10(100)
print("Log 100 adalah",c)
```

```
Jumlah dari a + b adalah 7
Pengurangan dari a - b adalah -1
Perkalian dari a * b adalah 12
Pembagian dari a / b adalah 0.75
Hasil e^a adalah 20.085536923187668
Ln b adalah 1.3862943611198906
Log 100 adalah 2.0
```

B. Operasi Vektor/Matriks

In [2]:

```
# membuat vektor
A = np.array([1,2,3]) # vektor baris 1x3
print("Vektor A",A)

# membuat matriks 2 dimensi dengan ukuran 3x3
B = np.array([
    [1,2,1],
    [2,1,3],
    [3,2,2]
])

print("Matriks B",B)

# penjumlahan vektor/matriks
C = np.add(A,B)
print(C)

# inisiasi array
A = np.zeros((3,4)) # membuat matriks dengan ukuran 3x4 di mana setiap elemen bern
ilai nol
print(A)
A = np.ones((3,4)) # membuat matriks dengan ukuran 3x4 di mana setiap elemen berni
lai 1
print(A)
```

```
# Matriks Tranpose
BT = np.transpose(B)
print(BT)
# atau
BT = B.T
print(BT)
```

```
Vektor A [1 2 3]
Matriks B [[1 2 1]
 [2 1 3]
 [3 2 2]]
[[2 4 4]
 [3 3 6]
 [4 4 5]]
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]
[[1. 1. 1. 1.]
 [1. 1. 1. 1.]
 [1. 1. 1. 1.]]
[[1 2 3]
 [2 1 2]
 [1 3 2]]
[[1 2 3]
 [2 1 2]
 [1 3 2]]
```

C. Slicing Array/Matrix

In [3]:

```
# mengakses elemen dari vektor/matriks
A = np.array([1,2,3])
a = A[2] # angka 2 adalah indeks. Indeks dimulai dari 0
# print("Elemen ke-2 dari vektor A adalah",a)

B = np.array([
    [1,2,1],
    [2,1,3],
    [3,2,2]
])

# karena B adalah matriks 2 dimensi, maka untuk mengakses suatu elemen
# membutuhakan 2 parameter yaitu baris dan kolom
x = B[0,2] # baris ke-0 berisi [1,2,1], kolom ke-2 dari baris ke-0 adalah 1
# print(x)

# Slicing array
x = A[0:2]
# print(x)

# memilih semua elemen sebanyak pada baris ke 0
x = B[:1]
print(x)
```

```
[[1 2 1]]
```

D. Array Manipulation

In [4]:

```
# mengubah matriks 2 dimensi ke vektor
vB = B.ravel()
print(vB)

D = np.array([
    [1,2,1],
    [2,1,3],
    [3,2,2],
    [4,5,6]
])

# mengubah bentuk array
vC = D.reshape(6, 2) # mengubah bentuk array dari 3x4 menjadi 6x2
print(vC)

# mengubah bentuk array ke vektor 1D
vC = D.reshape(1, -1) # sama dengan ravel()
print(vC)
```

```
[1 2 1 2 1 3 3 2 2]
[[1 2]
 [1 2]
 [1 3]
 [3 2]
 [2 4]
 [5 6]]
[[1 2 1 2 1 3 3 2 2 4 5 6]]
```

3. Penggunaan Pandas

Pandas merupakan library opensource dengan BSD licence, menawarkan high-performance, easy-to-use data structure dan data analysis tools untuk bahasa pemrograman Python. Pandas memungkinkan datascientist untuk melakukan import data dari berbagai sumber mulai dari CSV, Excel, SQL bahkan sampai binary HDF5. Pandas memiliki banyak variasi operasi data manipulation sepery groupby, join, merge, melt, concatenation dan juga untuk keperluan data cleansing seperti filtering dan replacing.

Data Frame

Pandas merepresentasikan data sebagai Data Frame. Data Frame sendiri merupakan struktur data berbentuk tabel dua dimensi dengan ukuran yang dapat berubah-ubah dan dapat menampung data heterogen (integer, text, decimal, dll.). Data Frame disusun oleh Baris, Kolom dan Data.

Advantages Pandas Library

1. Kemampuan untuk menghandle data yang sangat besar
   Ketika membaca file mengunakan Pandas, kita dapat memilah file tersebut menjadi beberapa bagian kecil. Setelah itu kita bisa melakukan preprocessing data untuk mendapatkan data yang sudah bersih dan hanya diperlukan.
2. Memiliki Feature yang sangat banyak

Pandas memiliki banyak fitur yang diperlukan untuk data analisis seperti untuk melakukan filtering dengan beragam kondisi atau segmentasi dan segregasi data sesuai dengan preferensi.

3. Handling missing data
   Pandas dapat menghandle missing data dengan menggunakan function fillna(). Nilai yang hilang dapat diganti dengan nilai khusus atau nilai bertambah seperti rata-rata, median.
4. Cleaning up
   Pandas dapat mendeteksi data yang hilang, tidak sesuai, atau tidak standar dan menggantinya dengan format yang umum atau umumnya diisi dengan median data. Function isnull() pada pandas dapat mengidentifikasi data pada baris kolom tertentu apakah kosong atau tidak, dan dengan mengunakan function fillna() kita bisa mengisinya dengan nilai yang standar.
5. Input and output tools dengan support banyak format file
   Pandas dapat membaca dan menulis dari dan ke berbagai jenis file mulai dari CSV, Excel, SQL, JSON, bahkan binary data seperti HDF5.
6. Merging and joining of Dataframe
   Pandas memiliki kemampuan untuk melakukan merging dan joining data sama seperti pada RDBMS SQL [9]. Jenis Merging dan joining data frame yang bisa dilakukan oleh Pandas, sebagai berikut :
   a) Inner Merge / Inner join
   b) Left Merge / Left outer join
   c) Right Merge / Right outer join
   d) Outer Merge / Full outer join
7. A lot of time series
   Pandas memberikan kemudahan dalam analisis data time series, yang dapat menjadikan index sebagai date time value, konversi antar time zone, konversi dari string date time menjadi date time, advance time series filtering, timeseries data plotting, rolling_data by average, sum, max, min dan lain sebagainya.
8. Visualization
   Pandas menyediakan fitur plot data yang sangat mudah digunakan. mulai dari histogram, line chart, bar chart, box plot, scatter plot, kernel density plot sampai plotting data untuk data timeseries.

A. Membaca dan menulis file csv, excel, JSON

Sebelum menggunakan Pandas terlebih dahulu lakukan proses import library pandas dengan perintah :

```
import pandas as pd
```

untuk cek versi pandas yang digunakan, jalankan command berikut,

```
pd.__version__
```

membaca file CSV yang telah disiapkan dengan menggunakan fungsi **pd.read_csv()**.

```
pd.read_csv("harga_rumah.csv")
```

Kita dapat menyimpan Pandas Data Frame kedalam variable untuk kebutuhan analisa selanjutnya,

```
data_rumah = pd.read_csv("harga_rumah.csv")
# cetak Dataframe
data_rumah
```

Menampilkan data teratas dengan menggunakan perintah : **pd.head(),** jika ingin menampilkan jumlah data tertentu maka didalam kurung tambahkan jumlah data yang akan ditampilkan: **pd.head(7)** untuk menampilkan 7 data teratas/pertama. Sedangkan untuk menampilkan data terakhir gunakan perintah : **pd.tail()** untuk jumlah data tertentu, masukkan jumlah data dalam kurung: **pd.tail(7)** untuk menampilkan 7 data terakhir.

In [5]:

```
import pandas as pd

# membaca file
data = pd.read_csv('../data/harga_rumah.csv') # membaca file csv
data.head() # .head() berfungsi untuk menampilkan top 5 data
```

Out[5]:

	jml_lantai	ruangan	luas (m2)	harga
0	2	2	40.5	235000.0
1	1	3	53.0	760000.0
2	10	1	28.0	320000.0
3	1	3	64.0	375000.0
4	9	2	47.8	450000.0

Kita juga dapat melakukan random sampling dari data yang diambil dengan menggunakan command berikut,

```
data_rumah.sample(5)
```

Melakukan sampling data acak sebanyak 5 baris data pada Data Frame.

Kita dapat menampilkan rangkuman dari Data Frame yang sedang diproses sehingga kita dapat melihat banyaknya kolom, tipe-data tiap kolom, banyaknya baris data sampai alokasi memori.

```
data_rumah.info()
```

In [6]:

```
# menulis file
df = pd.DataFrame({
    'nama': ['Budi', 'Siti'],
    'jenis': ['Laki-laki', 'Perempuan'],
    'usia': [22, 21]
})

# sebaiknya dikerjakan di komputer/PC local
# df.to_csv('../data/fileku.csv',index=False)
```

B. Indexing, Selecting, dan Assigning

In [7]:

```
# membaca salah satu kolom berdasarkan nama kolom
df = pd.DataFrame(data)
print(df.jml_lantai)
print(df['jml_lantai'])

# membaca baris pertama saja
print(df.iloc[0])

# membaca semua baris pada kolom tertentu berdasarkan index
print(df.iloc[:,1]) # iloc[:,1] -> tanda : artinya semua baris, 1 artinya index ko
lom ke-1

# membuat kolom tertentu untuk mengambil sisa kolom yang lain
# axis = 1 artinya hanya mengambil kolom
# bentuk dari X adalah matriks 2D
X = df.drop(['harga'],axis=1)
print(X.head())
```

```
0         2
1         1
2        10
3         1
4         9
        ..
23759     0
23760     0
23761     0
23762     1
23763     8
Name: jml_lantai, Length: 23764, dtype: int64
0         2
1         1
2        10
3         1
4         9
        ..
23759     0
23760     0
23761     0
23762     1
23763     8
Name: jml_lantai, Length: 23764, dtype: int64
jml_lantai         2.0
ruangan            2.0
luas (m2)         40.5
harga         235000.0
Name: 0, dtype: float64
0        2
1        3
2        1
3        3
4        2
       ..
23759    2
23760    3
23761    3
```

```
23762    1
23763    3
Name: ruangan, Length: 23764, dtype: int64
   jml_lantai  ruangan  luas (m2)
0           2        2       40.5
1           1        3       53.0
2          10        1       28.0
3           1        3       64.0
4           9        2       47.8
```

C. Data type

Di sini, kita akan belajar beberapa hal, yaitu:

1. Melihat tipe data
2. Konversi tipe data
3. Mencari data NaN. NaN singkatan dari Not a Number. NaN akan menyebabkan error saat data preprocessing jika dalam suatu kolom tipe datanya tidak seragam
4. Mehandle NaN

In [8]:

```
# melihat tipe data dari suatu kolom
print(df.jml_lantai.dtype)
print(df.harga.dtype)

# melihat tipe data semua kolom/atribut
print(df.dtypes)

# konvert tipe data pada kolom tertentu
# tipe data jml_lantai sebelumnya adalah int64
# di sini akan diconvert ke float64
print(df.jml_lantai.astype('float64'))
```

```
int64
float64
jml_lantai      int64
ruangan         int64
luas (m2)     float64
harga         float64
dtype: object
0         2.0
1         1.0
2        10.0
3         1.0
4         9.0
         ...
23759     0.0
23760     0.0
23761     0.0
23762     1.0
23763     8.0
Name: jml_lantai, Length: 23764, dtype: float64
```

In [9]:

```
# mencari data yang NaN
# NaN kepanjangan dari Not a Number
tmp = pd.DataFrame({
    'nama': ['Budi', 'Siti','Agus'],
```

```
    'jenis': ['Laki-laki', 'Perempuan','Laki-laki'],
    'usia': [22, 21,np.nan]
})

# cek yang mengandung NaN pada kolom usia
print(tmp[pd.isnull(tmp.usia)])
```

```
   nama      jenis  usia
2  Agus  Laki-laki   NaN
```

D. Renaming Column

Seringkali, kita akan mendapati dataset yang memiliki header label. Akan tetapi, kadang juga kita mendapati dataset yang tidak memiliki header label. Untuk kasus data tanpa label, di sini kita bisa **memberi label/header** pada dataset yang kita miliki.

In [10]:

```
# mengubah nama kolom

df = df.rename(columns={'jml_lantai':'lantai'})
df.head()
```

Out[10]:

	lantai	ruangan	luas (m2)	harga
0	2	2	40.5	235000.0
1	1	3	53.0	760000.0
2	10	1	28.0	320000.0
3	1	3	64.0	375000.0
4	9	2	47.8	450000.0

Latihan

1. **Petunjuk**: Jawab soal-soal berikut ini dengan membuat program Python:
   Diketahui:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \text{ dan } B = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

   a) Berapakah A + B? Munculkan hasilnya (poin 25)
   b) Berapakah A x B? Munculkan hasilnya
   c) Baca file harga_rumah.csv dan dapatkan kolom harga saja. Munculkan hasilnya
   d) Baca file harga_rumah.csv dan buang kolom harga. Munculkan hasilnya.

2. **Menggunakan Pandas**

   a) **Menggunakan Pandas**

In [1]:

```
# Import numpy dan pandas
import numpy as np
import pandas as pd
```

b) Menambahkan dan menampilkan tabel

In [2]:

```
opo_nama = {'Names' : ['Antasia','Dima','Katherine','James','Emily','Michael','Matthew','Laura','Kevin','Jonas'],
        'Scores' : [12.5,9,16.5,np.nan,9,20,14.5,np.nan,8,19], 'Attempts ': [1,3,2,3,2,3,1,1,2,1],
        'Qualify' : ['Yes','No','Yes','No','No','Yes','Yes','No','No','Yes']}
df_opo = pd.DataFrame(opo_nama)
df_opo
```

Out[2]:

	Names	Scores	Attempts	Qualify
0	Antasia	12.5	1	Yes
1	Dima	9.0	3	No
2	Katherine	16.5	2	Yes
3	James	NaN	3	No
4	Emily	9.0	2	No
5	Michael	20.0	3	Yes
6	Matthew	14.5	1	Yes
7	Laura	NaN	1	No
8	Kevin	8.0	2	No
9	Jonas	19.0	1	Yes

c) keluar kan yang Qualify = Yes

In [3]:

```
df_opo.loc[(df_opo['Qualify'] == 'Yes'), :]
```

Out[3]:

	Names	Scores	Attempts	Qualify
0	Antasia	12.5	1	Yes
2	Katherine	16.5	2	Yes
5	Michael	20.0	3	Yes
6	Matthew	14.5	1	Yes
9	Jonas	19.0	1	Yes

d) buat kolom baru Grade

In [4]:

```
df_opo['Grade'] = np.random.random(len(df_opo))
df_opo
```

Out[4]:

	Names	Scores	Attempts	Qualify	Grade
0	Antasia	12.5	1	Yes	0.856866
1	Dima	9.0	3	No	0.083202
2	Katherine	16.5	2	Yes	0.624952
3	James	NaN	3	No	0.143228
4	Emily	9.0	2	No	0.754052
5	Michael	20.0	3	Yes	0.443985
6	Matthew	14.5	1	Yes	0.476147
7	Laura	NaN	1	No	0.629983
8	Kevin	8.0	2	No	0.802167
9	Jonas	19.0	1	Yes	0.834177

e) tambahkan baris baru = Nopal

In [5]:

```
df_opo.append({'Names':'Nopal'}, ignore_index=True)
```

Out[5]:

	Names	Scores	Attempts	Qualify	Grade
0	Antasia	12.5	1.0	Yes	0.856866
1	Dima	9.0	3.0	No	0.083202
2	Katherine	16.5	2.0	Yes	0.624952
3	James	NaN	3.0	No	0.143228
4	Emily	9.0	2.0	No	0.754052
5	Michael	20.0	3.0	Yes	0.443985
6	Matthew	14.5	1.0	Yes	0.476147
7	Laura	NaN	1.0	No	0.629983
8	Kevin	8.0	2.0	No	0.802167
9	Jonas	19.0	1.0	Yes	0.834177
10	Nopal	NaN	NaN	NaN	NaN

In [6]:

```
new_row = pd.Series(['Nopal',100,2,'Yes',np.random.random(1)[0]],index=df_opo.colu
mns,name=10)
df_opo.append(new_row)
```

Out[6]:

	Names	Scores	Attempts	Qualify	Grade
0	Antasia	12.5	1	Yes	0.856866
1	Dima	9.0	3	No	0.083202
2	Katherine	16.5	2	Yes	0.624952
3	James	NaN	3	No	0.143228
4	Emily	9.0	2	No	0.754052
5	Michael	20.0	3	Yes	0.443985

	Names	Scores	Attempts	Qualify	Grade
6	Matthew	14.5	1	Yes	0.476147
7	Laura	NaN	1	No	0.629983
8	Kevin	8.0	2	No	0.802167
9	Jonas	19.0	1	Yes	0.834177
10	Nopal	100.0	2	Yes	0.468220

f) buat kolom Pass isinya 1 dan 0, dari Qualify

In [7]:

```
df_opo['Pass'] = np.where(
    df_opo['Qualify']=='yes',
    1,
    0
)
df_opo
```

Out[7]:

	Names	Scores	Attempts	Qualify	Grade	Pass
0	Antasia	12.5	1	Yes	0.856866	0
1	Dima	9.0	3	No	0.083202	0
2	Katherine	16.5	2	Yes	0.624952	0
3	James	NaN	3	No	0.143228	0
4	Emily	9.0	2	No	0.754052	0
5	Michael	20.0	3	Yes	0.443985	0
6	Matthew	14.5	1	Yes	0.476147	0
7	Laura	NaN	1	No	0.629983	0
8	Kevin	8.0	2	No	0.802167	0
9	Jonas	19.0	1			

3. Buatlah program untuk melakukan konstruksi matriks 1 x 100, isilah dengan bilangan random, kemudian carilah :

a) Nilai maksimum
b) Nilai minimum
c) Perbedaan nilai maksimum dan minimum
d) Media (Nilai Tegah)
e) Mean (rata-rata)
f) Standar deviasi
g) Variable variaance
h) Koefisien korelasi.
i) Jumlah total

4. Buatlah array dengan dimensi 1 x 100 dengan isi bilangan random, lakukan pengambilan pada:

a) 10 data pertama
b) 20 data pertama
c) 10 data terakhir
d) 20 data terakhir