



DIGITAL TALENT SCHOLARSHIP 2019

Big Data Analytics



Python (Library and how to Use)

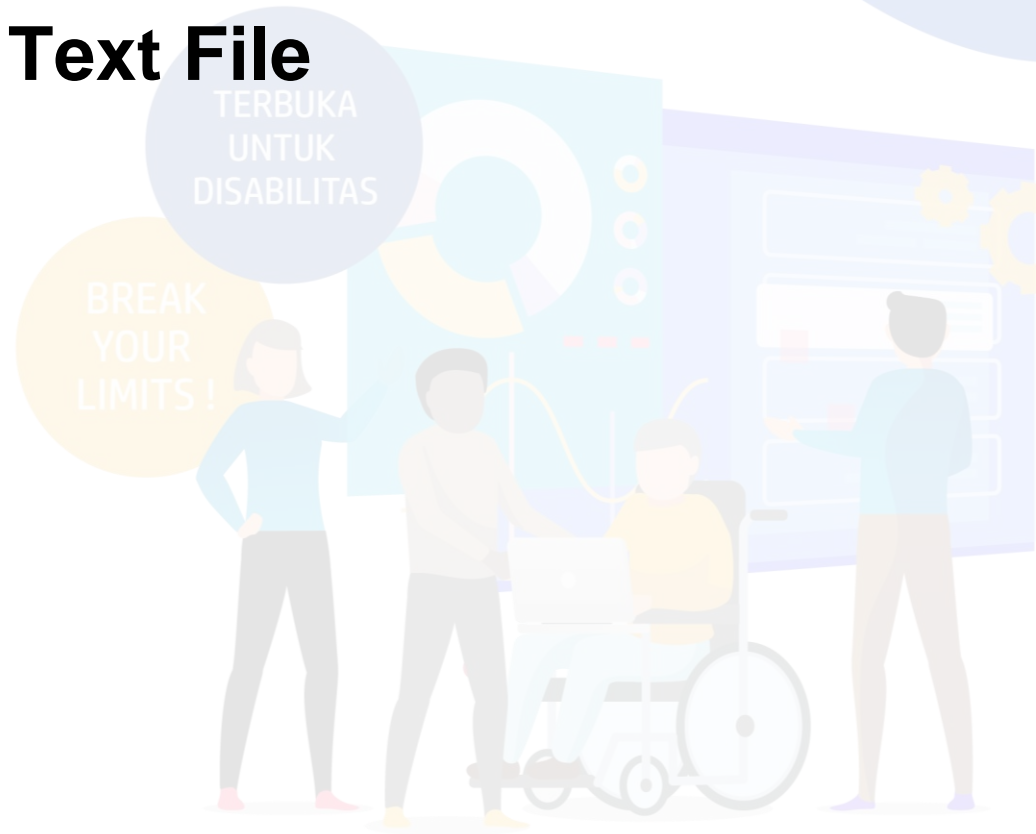
Oleh: Imam Cholissodin | imamcs@ub.ac.id, Putra Pandu Adikara, Sufia Adha Putri

Asisten: Guedho, Sukma, Anshori, Aang dan Gusti

Fakultas Ilmu Komputer (Filkom) Universitas Brawijaya (UB)

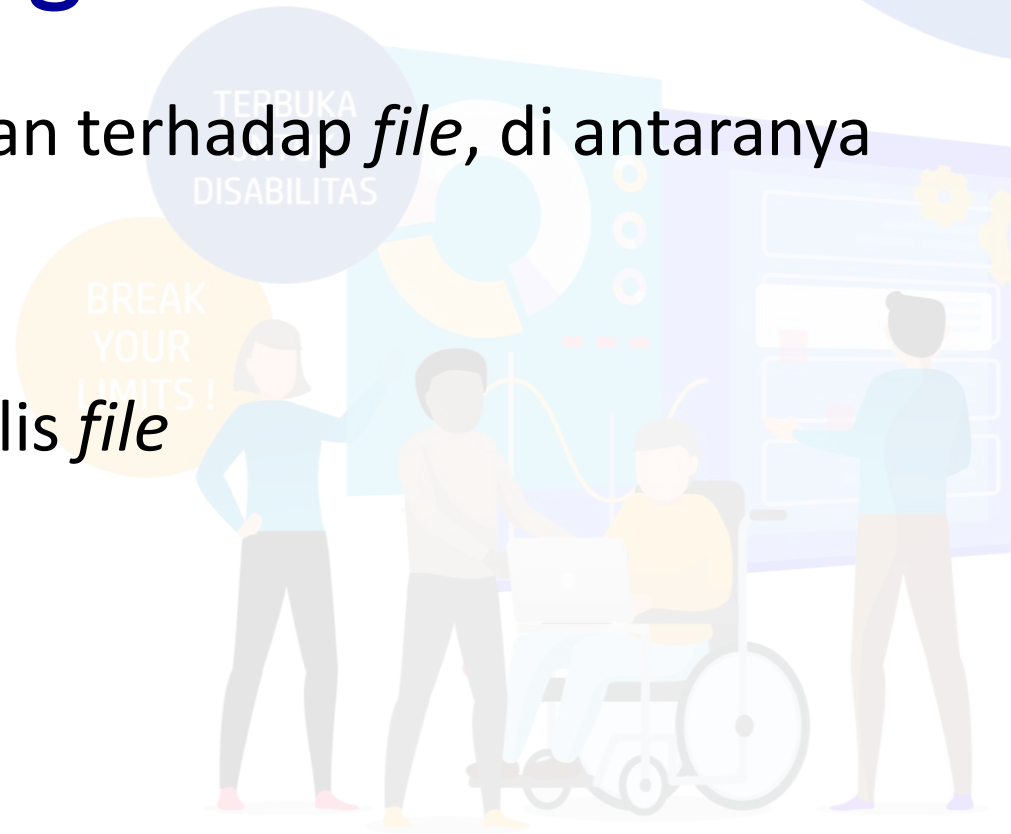
Pokok Bahasan

1. Reading & Writing Text File
2. Library Python
3. Tugas



Reading & Writing Text File

- Operasi yang kita lakukan terhadap *file*, di antaranya adalah:
 1. Membuka *file*
 2. Membaca atau menulis *file*
 3. Menutup *file*



Membuka *File*

- Sebelum kita membuka *file*, tentu harus ada filenya terlebih dahulu
- Buatlah sebuah file teks menggunakan notepad. Simpan di drive. misal C:\ dengan nama test.txt.
- Python memiliki fungsi *open()* untuk membuka file. Fungsi ini mengembalikan sebuah objek file yang digunakan untuk membaca atau mengubah file tersebut. `f = open("C:\\test.txt")`
- **Catatan:** `f = open("C:\\test.txt")` memiliki dua tanda `\` untuk menandai karakter escape. Bisa juga dengan menggunakan karakter raw `r` dengan penulisan `f = open(r"C:\test.txt")`

Membuka *File*

- Kita bisa menyebutkan mode *file* saat membukanya, apakah pakai mode baca 'r' (*read*), tulis 'w' (*write*), atau tambah 'a' (*append*)
- Kita juga bisa membuka *file* dalam mode biner 'b'
- *Default*-nya adalah mode baca. Dalam mode ini, kita bisa memperoleh semua teks atau string yang ada di dalam *file*
- Mode biner kita gunakan saat berhubungan dengan file gambar atau *file* .exe

```
>>> f = open("test.txt")      # sama dengan mode 'r' atau 'rt'  
>>> f = open("test.txt", 'w') # mode tulis  
>>> f = open("img.jpg", 'r+b') # membaca dan menulis dalam mode biner
```

Membuka *File*

Mode	Deskripsi
'r'	Membuka file untuk dibaca. (default)
'w'	Membuka file untuk ditulis. Membuat file baru jika file belum tersedia atau menimpa isi file jika file sudah ada
'x'	Membuka file untuk pembuatan eksklusif. Jika file sudah ada, maka operasi akan gagal
'a'	Membuka file dan menambahkan karakter di ujung file lama (tanpa menghapus isinya). Membuat file baru bila file belum tersedia
't'	Membuka dalam mode teks. (default)
'b'	Membuka file dalam mode biner
'+'	Membuka file untuk diupdate (membaca dan menulis)



Membaca *File*

- *read(n)* berfungsi untuk membaca sebanyak *n* karakter
- *Method* ini akan membaca *file* dan mengembalikan akhir dari *file* yang dibaca tadi

```
>>> f = open("C:\\test.txt")
>>> f.read(4)    # membaca 4 data (karakter) pertama
'This'

>>> f.read(4)    # membaca 4 yang berikutnya lagi
' is '

>>> f.read()      # membaca file sisanya sampai akhir file
'the first line\nThis is the second line\nThis is the third line\n'

>>> r.read()      # sudah tidak ada karakter untuk dibaca
''
```

Menulis *File*

- Untuk menulis ke dalam *file*, kita menggunakan mode 'w' pada saat membuka *file*-nya
- Bisa juga menggunakan mode 'a' untuk menambah isi *file* dari akhir *file* awal, tanpa menghapus atau menimpa isinya terlebih dahulu
- Kita harus berhati-hati dalam menggunakan mode 'w' karena bisa menimpa *file* jika *file*-nya sudah ada
- Semua data di *file* yang tertimpa akan terhapus dan diganti dengan isi baru



DIGITAL
TALENT
SCHOLARSHIP

Menulis *File*

- Kita bisa menulis *file* dengan menggunakan metode *write()*
- Metode ini mengembalikan jumlah karakter yang dituliskan ke dalam *file*

```
with open("C:\\test.txt", 'w') as f:  
    f.write("The new first line\n")  
    f.write("The new second line\n")  
    f.write("The new third line\n")
```

Menutup *File*

- *File* yang sudah terbuka perlu ditutup kembali menggunakan metode *close()*. Bila Anda tidak menutup *file*, maka perubahan yang Anda lakukan bisa saja hilang
- Menutup *file* akan membebaskan memori yang terpakai

```
f.open("C:\\\\test.txt")  
# melakukan berbagai operasi file  
f.close()
```

Menutup *File*

- Cara menutup *file* di atas kurang baik, karena jika ada *error* pada saat operasi *file*, maka program akan keluar sebelum menutup *file*
- Cara yang lebih baik adalah dengan menggunakan blok `try...finally`
- Dengan begitu, kita dapat menjamin bahwa *file* akan tetap ditutup walaupun ada *error* sebelumnya.

```
try:
    f = open("C:\\test.txt")
    # melakukan beberapa operasi file
finally:
    f.close()
```



DIGITAL
TALENT
SCHOLARSHIP

Metode Operasi *File*

Metode	Deskripsi
close()	Menutup file
detach()	Memisahkan buffer biner dari TextIOBase dan mengembalikannya.
fileno()	Mengembalikan integer (file descriptor) file
flush()	Mengosongkan buffer aliran file (filestream)
isatty()	Mengembalikan True jika stream file interaktif
read(n)	Membaca n karakter dari file. Bila tidak ada argumen maka dibaca seluruh isi file.
readable()	Mengembalikan True bila file bisa dibaca
readline()	Membaca dan mengembalikan satu baris file.
readlines()	Membaca dan mengembalikan semua baris sebagai satu list.
seek(offset)	Mengubah posisi kursor file ke offset byte.
seekable()	Mengembalikan True jika stream file mendukung akses random
tell()	Mengembalikan posisi kursor sekarang
truncate(size)	Mengubah ukuran stream file menjadi size byte.
writable()	Mengembalikan True jika stream file bisa ditulis.
write(s)	Menuliskan string s ke file dan mengembalikan jumlah karakter yang dituliskan
writelines(lines)	Menuliskan list lines ke dalam file





DIGITAL
TALENT
SCHOLARSHIP

Library Python



Library Python

- Terdapat beberapa *library* yang banyak digunakan di Python:
 1. Numpy
 2. Pandas
 3. Scipy
 4. Matplotlib
 5. Scikit-Learn



NumPy

- Numpy merupakan package Python untuk **numerical Python**.
- Library yang terdiri dari objek array multidimensi
- Dikembangkan pertama kali oleh Jim Hugunin. Tahun 2005, Travis Oliphant menciptakan package Numpy dengan menggabungkan fitur Numarray kedalam package Numeric.
- *Open source* - Terdapat banyak kontributor

NumPy

- Dengan menggunakan NumPy, developer dapat melakukan operasi berikut:
 - Operasi matematika dan logis pada array
 - Fourier transforms dan *routine* untuk manipulasi bentuk
 - Operasi yang berkaitan dengan aljabar linear. Numpy memiliki fungsi yang build in untuk aljabar linear
 - Numpy sering digunakan Bersama dengan package-package seperti Scipy (Scientific Python) dan Matplotlib (plotting library).

NumPy – Nddarray Object

- Objek yang paling penting yang didefinisikan dalam Numpy adalah array N-dimensi yang disebut nd array
- Ini menggambarkan koleksi item dengan tipe yang sama.
- Setiap item dalam ndarray memiliki ukuran blok yang sama dalam memori
- Setiap elemen dalam ndarray adalah objek dari data-type object (disebut dtype)



Contoh

```
▶ # more than one dimensions
import numpy as np
a = np.array([[1, 2], [3, 4]])
print (a)
```

```
[[1 2]
 [3 4]]
```

TERBUKA
UNTUK
DISABILITAS

```
▶ # dtype parameter
import numpy as np
a = np.array([1, 2, 3], dtype = complex)
print (a)
```

```
[1.+0.j 2.+0.j 3.+0.j]
```



DIGITAL
TALENT
SCHOLARSHIP

NumPy – Tipe Data

```
# using array-scalar type
import numpy as np
dt = np.dtype(np.int32)
print (dt)
```

int32

```
#int8, int16, int32, int64
#can be replaced by equivalent string 'i1', 'i2', 'i4', etc.
import numpy as np

dt = np.dtype('i4')
print (dt)
```

int32

Sr.No.	Data Types & Description
1	bool_ Boolean (True or False) stored as a byte
2	int_ Default integer type (same as C long; normally either int64 or int32)
3	intc Identical to C int (normally int32 or int64)
4	intp Integer used for indexing (same as C ssize_t; normally either int32 or int64)
5	int8 Byte (-128 to 127)
6	int16 Integer (-32768 to 32767)
7	int32 Integer (-2147483648 to 2147483647)
8	int64 Integer (-9223372036854775808 to 9223372036854775807)
9	uint8 Unsigned integer (0 to 255)



DIGITAL
TALENT
SCHOLARSHIP

NumPy – Tipe Data

10	uint16 Unsigned integer (0 to 65535)
11	uint32 Unsigned integer (0 to 4294967295)
12	uint64 Unsigned integer (0 to 18446744073709551615)
13	float_ Shorthand for float64
14	float16 Half precision float: sign bit, 5 bits exponent, 10 bits mantissa
15	float32 Single precision float: sign bit, 8 bits exponent, 23 bits mantissa
16	float64 Double precision float: sign bit, 11 bits exponent, 52 bits mantissa
17	complex_ Shorthand for complex128
18	complex64 Complex number, represented by two 32-bit floats (real and imaginary components)
19	complex128 Complex number, represented by two 64-bit floats (real and imaginary components)

NumPy – Array Attributes

- Beberapa Atribut array Numpy
 - `ndarray.shape`
 - `ndarray.ndim`
 - `numpy.itemsize`
 - `numpy.flags`

TERBUKA
UNTUK
PEMILIK
DISABILITAS

BREAK
YOUR
LIMITS !



NumPy – Array Attributes

- **ndarray.shape** – atribut array yang mengembalikan tuple yang terdiri dimensi array dan juga dapat digunakan untuk mengubah ukuran array

```
▶ import numpy as np  
a = np.array([[1,2,3],[4,5,6]])  
print (a.shape)
```

(2, 3)

```
▶ # this resizes the ndarray  
import numpy as np  
  
a = np.array([[1,2,3],[4,5,6]])  
a.shape = (3,2)  
print (a)
```

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

```
▶ import numpy as np  
a = np.array([[1,2,3],[4,5,6]])  
b = a.reshape(3,2)  
print (b)
```

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

- Reshape function untuk merubah ukuran array

NumPy – Array Attributes

- **ndarray.ndim** – atribut array untuk mengembalikan jumlah dimensi array

```
► # an array of evenly spaced numbers
import numpy as np
a = np.arange(24)
print(a)
```

```
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23]
```

```
► # this is one dimensional array
import numpy as np
a = np.arange(24)
a.ndim
```

```
# now reshape it
b = a.reshape(2,4,3)
print(b)
# b is having three dimensions
```

```
[[[ 0  1  2]
   [ 3  4  5]
   [ 6  7  8]
   [ 9 10 11]]
```

```
[[12 13 14]
 [15 16 17]
 [18 19 20]
 [21 22 23]]]
```

NumPy – Array Attributes

- **numpy.itemsize** – atribut array untuk mengembalikan jumlah dimensi array

```
▶ # dtype of array is int8 (1 byte)
import numpy as np
x = np.array([1,2,3,4,5], dtype = np.int8)
print (x.itemsize)
```

1

```
▶ # dtype of array is now float32 (4 bytes)
import numpy as np
x = np.array([1,2,3,4,5], dtype = np.float32)
print (x.itemsize)
```

4

TERBUKA
UNTUK
DISABILITAS





DIGITAL
TALENT
SCHOLARSHIP

Pandas

- Open-source, Library berlisensi BSD
- Struktur data yang cepat, fleksibel, ekspresif yang didesain untuk membuat pekerjaan dengan data *relational* atau *labeled* mudah dan intuitif
- Merupakan *analysis tool* pada pemograman Python
- Panda dibangun di atas Numpy dan dapat berintegrasi dengan lingkungan komputasi ilmiah dengan banyak 3rd party libraries

Pandas – *Environment Setup*

- Distribusi Python standar tidak di-*bundle* dengan modul Pandas
- **Windows:** Jika menginstall Anaconda Python package, pandas sudah terinstal secara *default*
- **Linux:**
 - **Ubuntu**

```
sudo apt-get install python-numpy python-scipy python-matplotlibpythonipynbpythonnotebook  
python-pandas python-sympy python-nose
```
 - **Fedora**

```
sudo yum install numpy scipy python-matplotlibpython python-pandas sympy  
python-nose atlas-devel
```

Pandas – Struktur Data

Data Structure	Dimensions	Description
Series	1	1D labeled homogeneous array, size immutable.
Data Frames	2	General 2D labeled, size-mutable tabular structure with potentially heterogeneously typed columns.
Panel	3	General 3D labeled, size-mutable array.

- Series : Homogeneous data, ukuran dan nilai bersifat *immutable* (tidak bisa diubah)
- DataFrame: *heterogeneous data*, ukuran dan data bersifat *mutable* (dapat diubah)
- Panel: *heterogeneous data*, ukuran dan data bersifat *mutable* (dapat diubah)

Pandas - Membuat *Series*

- Empty Series

```
▶ import pandas as pd  
s = pd.Series()  
print (s)
```

```
Series([], dtype: float64)
```

TERBUKA
UNTUK
DISABILITAS

BREAK
YOUR
LIMITS!



Pandas - Membuat *Series*

- ndarray

```
▶ import pandas as pd
import numpy as np
data = np.array(['a', 'b', 'c', 'd'])
s = pd.Series(data)
print (s)
```

```
0    a
1    b
2    c
3    d
dtype: object
```

TERBUKA
UNTUK
DISABILITAS

Jika tidak ada index yang di passing, maka akan di assign default index 0 sampai **len(data)-1**

Pandas - Membuat *Series*

- ndarray

```
▶ import pandas as pd
import numpy as np
data = np.array(['a', 'b', 'c', 'd'])
s = pd.Series(data, index=[100, 101, 102, 103])
print (s)
```

```
100    a
101    b
102    c
103    d
dtype: object
```

melewatkan((passed) nilai index

TERBUKA
UNTUK
DISABILITAS



Pandas - Membuat *Series*

- Dict dapat di *passing*kan sebagai input dan jika tidak ada indeks yang ditentukan, maka key *dictionary* diambil dalam urutan untuk membuat index.
- Jika indeks di *passing*kan / dilewatkan, nilai-nilai dalam data yang sesuai dengan label dalam indeks akan ditarik keluar
 - Series dari dict

```
#import the pandas library and aliasing as pd
import pandas as pd
import numpy as np
data = {'a' : 0., 'b' : 1., 'c' : 2.}
s = pd.Series(data)
print (s)
```

```
a    0.0
b    1.0
c    2.0
dtype: float64
```

Pandas - Membuat *Series*

Series dari dict

```
#import the pandas library and aliasing as pd
import pandas as pd
import numpy as np
data = {'a' : 0., 'b' : 1., 'c' : 2.}
s = pd.Series(data,index=['b','c','d','a'])
print (s)
```

```
b    1.0
c    2.0
d    NaN
a    0.0
dtype: float64
```

Urutan indeks bertahan dan element yang missing diisi dengan NaN (Not a Number)

TERBUKA
UNTUK
DISABILITAS



Pandas - Membuat *Series*

Series dari scalar

```
▶ #import the pandas library and aliasing as pd  
import pandas as pd  
import numpy as np  
s = pd.Series(5, index=[0, 1, 2, 3])  
print (s)
```

```
0    5  
1    5  
2    5  
3    5  
dtype: int64
```

Jika data adalah nilai scalar, index harus disediakan. Nilai akan diulang sesuai dengan Panjang index

TERBUKA
UNTUK
DISABILITAS



Mengakses Data dari *Series* dengan Posisi

```
▶ import pandas as pd
s = pd.Series([1,2,3,4,5],index = ['a','b','c','d','e'])

#retrieve the first element
print (s[0])
```

1

```
▶ import pandas as pd
s = pd.Series([1,2,3,4,5],index = ['a','b','c','d','e'])

#retrieve the first three element
print (s[:3])
```

```
a    1
b    2
c    3
dtype: int64
```

Mengambil 3 elemen pertama

Mengakses Data dari *Series* dengan Posisi

```
▶ import pandas as pd
s = pd.Series([1,2,3,4,5],index = ['a','b','c','d','e'])

#retrieve the last three element
print (s[-3:])
```

```
c    3
d    4
e    5
dtype: int64
```

Mengambil 3 elemen terakhir

Mengambil Data Menggunakan Label (index)

```
▶ import pandas as pd
s = pd.Series([1,2,3,4,5],index = ['a','b','c','d','e'])

#retrieve multiple elements
print (s[['a','c','d']])
```

```
a    1
c    3
d    4
dtype: int64
```

Series bisa dikatakan sebagai *fixed-size dict* dimana kita bisa mendapatkan dan menetapkan nilai berdasarkan label index

Mengambil Data Menggunakan Label (index)

TERBUKA

```
import pandas as pd
s = pd.Series([1,2,3,4,5],index = ['a','b','c','d','e'])

#retrieve multiple elements
print (s['f'])
```

```
-----
TypeError                                Traceback (most recent call last)
~\Anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_value(self, series, key)
    4380         try:
-> 4381             return libindex.get_value_box(s, key)
    4382         except IndexError:
```



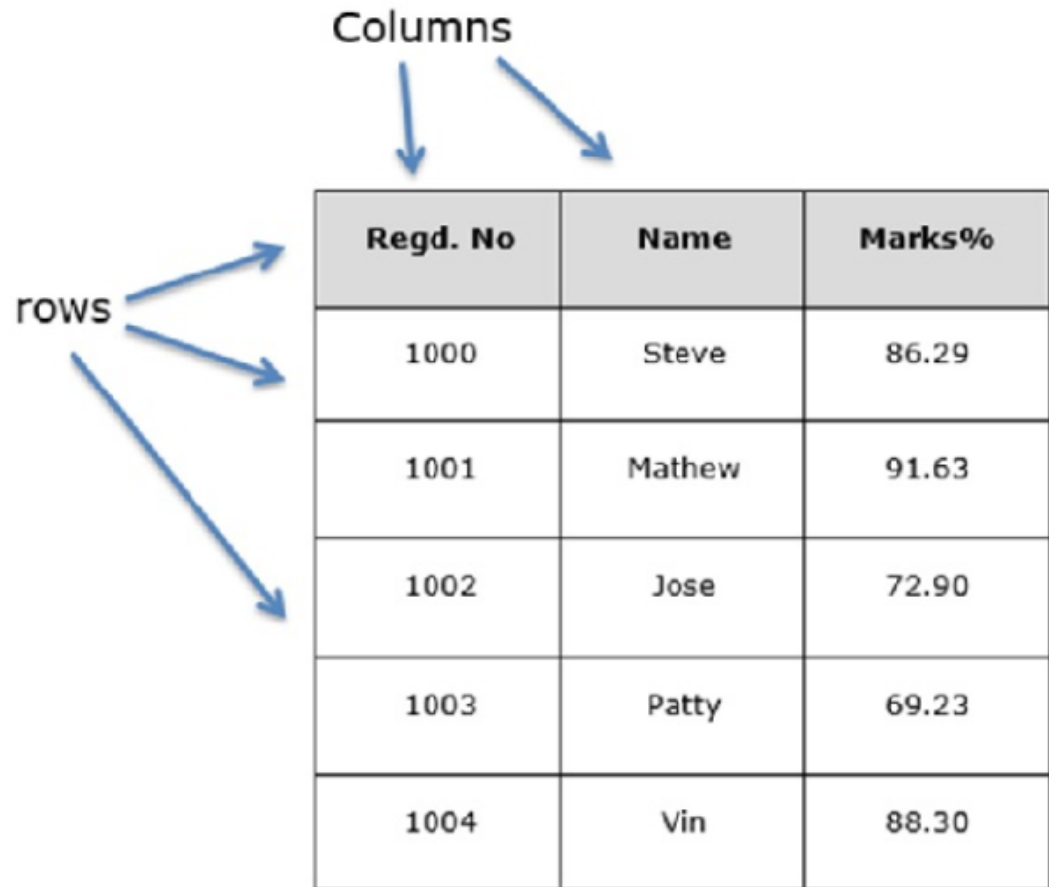
Pandas - DataFrame

- Mari asumsikan kita membuat data frame dengan data siswa
- Anda dapat berpikir ini sebagai tabel SQL atau spreadsheet data

Columns

ROWS

Regd. No	Name	Marks%
1000	Steve	86.29
1001	Mathew	91.63
1002	Jose	72.90
1003	Patty	69.23
1004	Vin	88.30



Pandas - DataFrame

- Membuat pandas DataFrame dapat menggunakan beberapa inputan seperti:
 - Lists
 - Dict
 - Series
 - Numpy ndarrays
 - Another DataFrame



Pandas – Membuat DataFrame

Empty DataFrame

► *#import the pandas library and aliasing as pd*
import pandas **as** pd
df = pd.DataFrame()
print (df)

Empty DataFrame

Columns: []

Index: []

TERBUKA
UNTUK
DISABILITAS

Pandas – Membuat DataFrame

DataFrame dari Lists

```
▶ import pandas as pd  
data = [1,2,3,4,5]  
df = pd.DataFrame(data)  
print (df)
```

	0
0	1
1	2
2	3
3	4
4	5

TERBUKA
UNTUK
DISABILITAS

REAK
OUR
MITS!



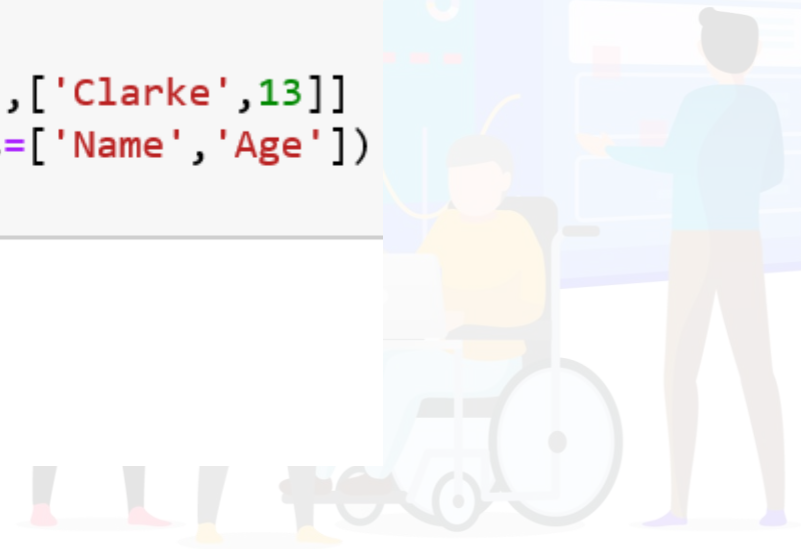
Pandas – Membuat DataFrame

DataFrame dari Lists

```
▶ import pandas as pd  
data = [['Alex',10],['Bob',12],['Clarke',13]]  
df = pd.DataFrame(data,columns=['Name','Age'])  
print (df)
```

	Name	Age
0	Alex	10
1	Bob	12
2	Clarke	13

TERBUKA
UNTUK
DISABILITAS



Pandas – Membuat DataFrame

DataFrame dari Lists

```
▶ import pandas as pd
data = [['Alex',10],['Bob',12],['Clarke',13]]
df = pd.DataFrame(data,columns=['Name','Age'],dtype=float)
print (df)
```

	Name	Age
0	Alex	10.0
1	Bob	12.0
2	Clarke	13.0

Dtype parameter mengubah jenis tipe data **Age** menjadi floating point

TERBUKA
UNTUK
DISABILITAS

Pandas – Membuat DataFrame

DataFrame dari Dict of ndarrays / Lists

```
▶ import pandas as pd
data = {'Name': ['Tom', 'Jack', 'Steve', 'Ricky'], 'Age': [28, 34, 29, 42]}
df = pd.DataFrame(data)
print(df)
```

	Name	Age
0	Tom	28
1	Jack	34
2	Steve	29
3	Ricky	42

- Semua ndarrays harus memiliki Panjang yang sama. Jika tidak ada indeks yang di passing, maka Panjang indeks harus sama dengan Panjang array
- Jika tidak ada indeks yang di passing maka n indeks sama dengan Panjang array

Pandas – Membuat DataFrame

Membuat index data frame menggunakan array

```
▶ import pandas as pd
data = {'Name': ['Tom', 'Jack', 'Steve', 'Ricky'], 'Age': [28, 34, 29, 42]}
df = pd.DataFrame(data, index=['rank1', 'rank2', 'rank3', 'rank4'])
print (df)
```

	Name	Age
rank1	Tom	28
rank2	Jack	34
rank3	Steve	29
rank4	Ricky	42

Pandas – Membuat DataFrame

Membuat DataFrame dari *List of Dicts*

```
import pandas as pd
data = [{'a': 1, 'b': 2}, {'a': 5, 'b': 10, 'c': 20}]
df = pd.DataFrame(data)
print(df)
```

	a	b	c
0	1	2	NaN
1	5	10	20.0

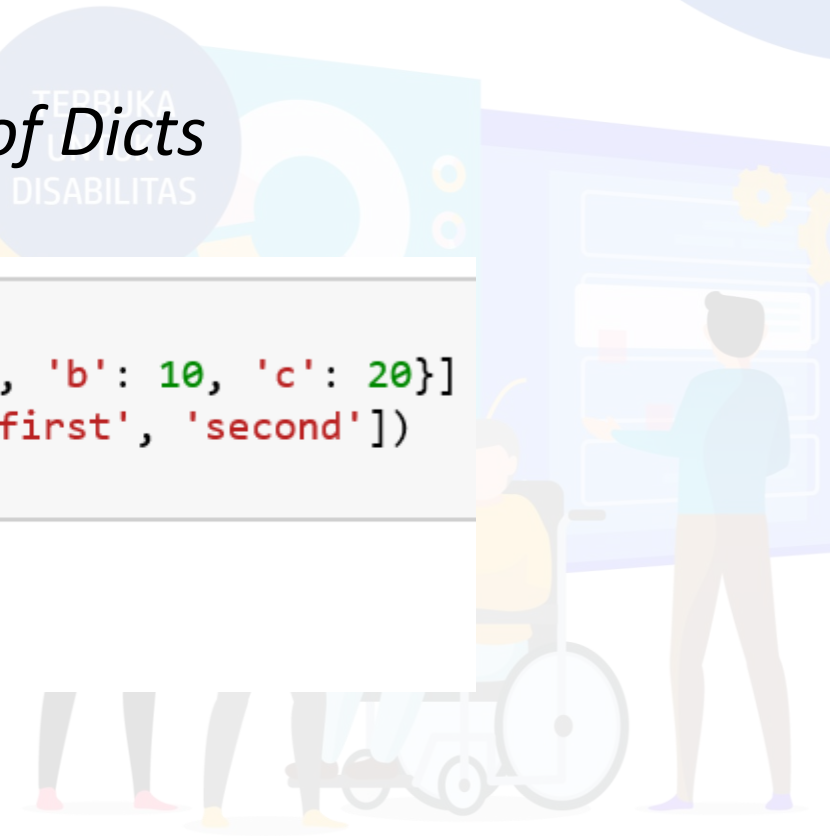
- List dictionaries dapat dipassingkan sebagai input data untuk membuat DataFrame
- Key dictionary secara default diambil sebagai nama kolom
- Nan (Not a Number) ditambahkan ke area yang missing

Pandas – Membuat DataFrame

Membuat DataFrame dari *List of Dicts*

```
▶ import pandas as pd  
data = [{'a': 1, 'b': 2}, {'a': 5, 'b': 10, 'c': 20}]  
df = pd.DataFrame(data, index=['first', 'second'])  
print (df)
```

	a	b	c
first	1	2	NaN
second	5	10	20.0



Pandas – Membuat DataFrame

Membuat DataFrame dari *List of Dicts*

```
import pandas as pd
data = [{'a': 1, 'b': 2}, {'a': 5, 'b': 10, 'c': 20}]

#With two column indices, values same as dictionary keys
df1 = pd.DataFrame(data, index=['first', 'second'], columns=['a', 'b'])

#With two column indices with one index with other name
df2 = pd.DataFrame(data, index=['first', 'second'], columns=['a', 'b1'])
print(df1)
print(df2)
```

	a	b
first	1	2
second	5	10

	a	b1
first	1	NaN
second	5	NaN

Membuat DataFrame dengan list of dictionary, indeks baris, dan indeks kolom

Pandas – Membuat DataFrame

Membuat DataFrame dari *Dict of Series*

```
▶ import pandas as pd

d = {'one' : pd.Series([1, 2, 3], index=['a', 'b', 'c']),
     'two' : pd.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd'])}

df = pd.DataFrame(d)
print (df)
```

	one	two
a	1.0	1
b	2.0	2
c	3.0	3
d	NaN	4

Indeks merupakan semua indeks yang di passing

Pandas – *Column Selection*

Memilih kolom dari DataFrame

TERBUKA
UNTUK
DISABILITAS

```
▶ import pandas as pd
```

```
d = {'one' : pd.Series([1, 2, 3], index=['a', 'b', 'c']),  
     'two' : pd.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd'])}
```

```
df = pd.DataFrame(d)  
print (df ['one'])
```

```
a    1.0  
b    2.0  
c    3.0  
d    NaN
```

```
Name: one, dtype: float64
```

Pandas – Column Addition

Menambah
kolom baru
pada Data
Frame
yang sudah
ada

```
import pandas as pd

d = {'one' : pd.Series([1, 2, 3], index=['a', 'b', 'c']),
     'two' : pd.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd'])}

df = pd.DataFrame(d)

# Adding a new column to an existing DataFrame object with column label by passing new series

print ("Adding a new column by passing as Series:")
df['three']=pd.Series([10,20,30],index=['a','b','c'])
print (df)

print ("Adding a new column using the existing columns in DataFrame:")
df['four']=df['one']+df['three']

print (df)
```

Adding a new column by passing as Series:

	one	two	three
a	1.0	1	10.0
b	2.0	2	20.0
c	3.0	3	30.0
d	NaN	4	NaN

Adding a new column using the existing columns in DataFrame:

	one	two	three	four
a	1.0	1	10.0	11.0
b	2.0	2	20.0	22.0
c	3.0	3	30.0	33.0
d	NaN	4	NaN	NaN

DIGITAL
SCHOOL

Python Pandas – Column Deletion

Menghapus
kolom baru
pada Data
Frame yang
sudah ada

*# Using the previous DataFrame, we will delete a column
using del function*

```
import pandas as pd
```

```
d = {'one' : pd.Series([1, 2, 3], index=['a', 'b', 'c']),  
      'two' : pd.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd']),  
      'three' : pd.Series([10,20,30], index=['a','b','c'])}
```

```
df = pd.DataFrame(d)  
print ("Our dataframe is:")  
print (df)
```

```
# using del function  
print ("Deleting the first column using DEL function:")  
del df['one']  
print (df)
```

```
# using pop function  
print ("Deleting another column using POP function:")  
df.pop('two')  
print (df)
```

Our dataframe is:

	one	two	three
a	1.0	1	10.0
b	2.0	2	20.0
c	3.0	3	30.0
d	NaN	4	NaN

Deleting the first column using DEL function:

	two	three
a	1	10.0
b	2	20.0
c	3	30.0
d	4	NaN

Deleting another column using POP function:

	three
a	10.0
b	20.0
c	30.0
d	NaN

Pandas – *Row Selection, Addition, Deletion*

Selection by Label - Baris dapat dipilih dengan melewati label baris ke fungsi **loc**.

```
➤ import pandas as pd

d = {'one' : pd.Series([1, 2, 3], index=['a', 'b', 'c']),
     'two' : pd.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd'])}

df = pd.DataFrame(d)
print (df.loc['b'])
```

```
one    2.0
two    2.0
Name: b, dtype: float64
```

Pandas – *Row Selection, Addition, Deletion*

Selection by integer location – passing lokasi integer ke fungsi **iloc**

```
import pandas as pd

d = {'one' : pd.Series([1, 2, 3], index=['a', 'b', 'c']),
     'two' : pd.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd'])}

df = pd.DataFrame(d)
print (df.iloc[2])
```

```
one    3.0
two    3.0
Name: c, dtype: float64
```

Pandas – *Row Selection, Addition, Deletion*

Slice Rows – Multiple row dapat dipilih menggunakan operator ':'

```
▶ import pandas as pd

d = {'one' : pd.Series([1, 2, 3], index=['a', 'b', 'c']),
     'two' : pd.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd'])}

df = pd.DataFrame(d)
print (df[2:4])
```

	one	two
c	3.0	3
d	NaN	4

Pandas – *Row Selection, Addition, Deletion*

Addition of Rows– Penambahan baris baru ke DataFrame menggunakan fungsi **append**, ini akan menambahkan baris di bagian akhir

```
import pandas as pd

df = pd.DataFrame([[1, 2], [3, 4]], columns = ['a', 'b'])
df2 = pd.DataFrame([[5, 6], [7, 8]], columns = ['a', 'b'])

df = df.append(df2)
print (df)
```

	a	b
0	1	2
1	3	4
0	5	6
1	7	8

Pandas – *Row Selection, Addition, Deletion*

- Deletion of Rows– Gunakan label indeks untuk menghapus baris dari DataFrame. Jika terdapat label ganda, maka multiple rows akan dihapus
- Jika diperhatikan contoh sebelumnya labelnya duplikat / ganda.

```
➤ import pandas as pd

df = pd.DataFrame([[1, 2], [3, 4]], columns = ['a', 'b'])
df2 = pd.DataFrame([[5, 6], [7, 8]], columns = ['a', 'b'])

df = df.append(df2)

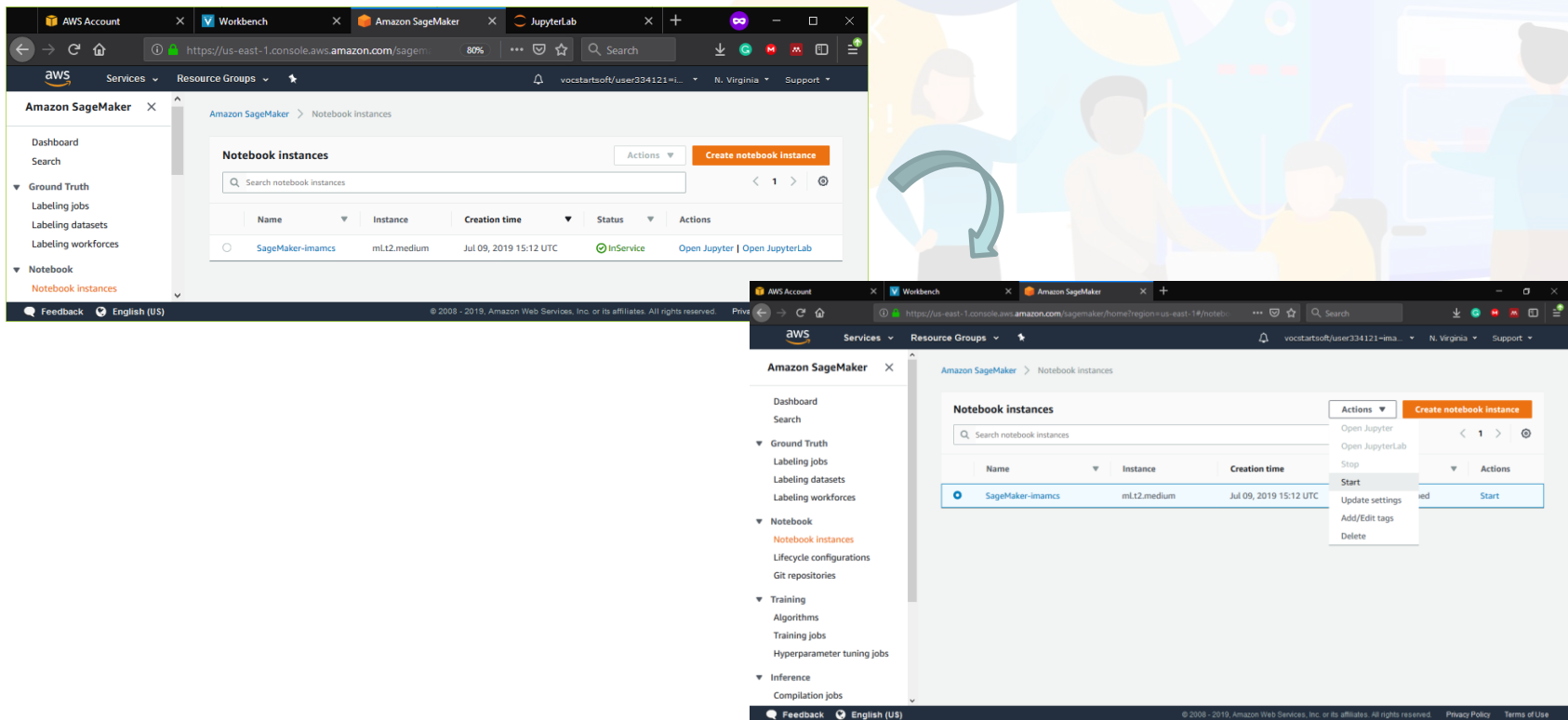
# Drop rows with label 0
df = df.drop(0)

print (df)
```

	a	b
1	3	4
1	7	8

Start Jupyter Notebook

- Silahkan menggunakan Jupyter yang anda install *from scratch* di minggu pertama (di EC2) atau menggunakan SageMaker yang baru dibuat sebelumnya. Dan mohon dicek credit anda, jika menggunakan SageMaker terlalu menguras credit Anda, sebaiknya di-stop setiap kali selesai menggunakan (jgn lupa selalu backup file *.ipynb ke github Anda atau di local).



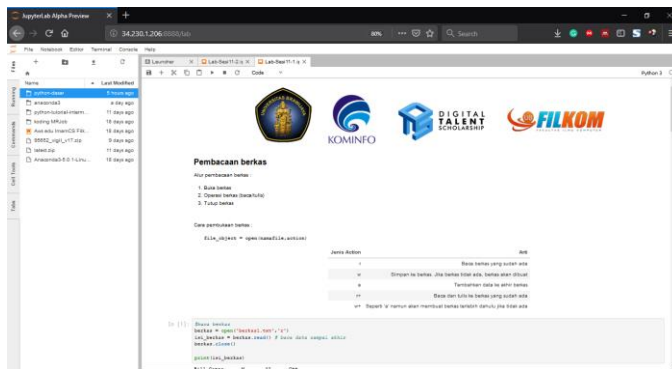


DIGITAL
TALENT
SCHOLARSHIP

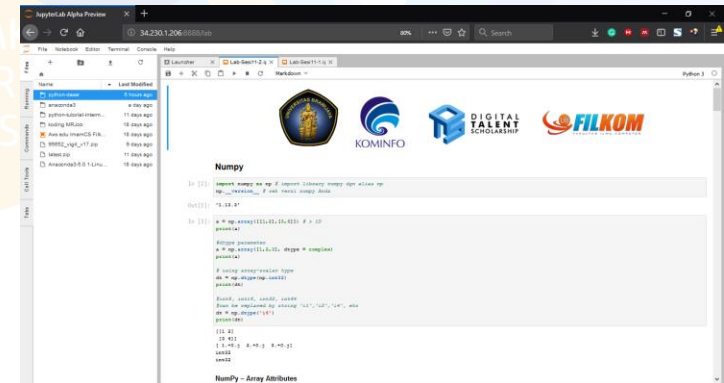
Latihan langsung di Kelas Ke-1 & Pembahasan Link kode “<http://bit.ly/2mOKcFR>”

Silahkan dicoba dijalankan dengan Jupyter notebook yang Anda buat sebelumnya di Ubuntu 16.04 atau dengan SageMaker notebook (JupyterLab) yang baru Anda buat hari ini.

Lab-Sesi10-1



Lab-Sesi10-2



Latihan langsung di Kelas Ke-2 & Pembahasan

- Buatlah program untuk mencari rata-rata dari total pendapatan dari jenis usaha suatu industri diseluruh dunia di setiap harinya (buat data tersebut dengan membuat program **generate data** dengan ukuran, misal sampai diatas satu juta record). Jika diidentifikasi data tersebut sangat besar dan membutuhkan pengolahan secara distribusi (Membagi tugas tiap komputer client secara merata lalu komputer yang ditunjuk sebagai komputer pusat mengambil hasil komputasi tiap komputer yang lainnya untuk dikalkulasi nilai rata-rata akhirnya). Buatlah program tersebut sebagai simulasi pengoalahan Big Data dengan ketentuan:
 - a. Misal dibuat hanya dengan 1 komputer
 - b. Dengan n -komputer, dan n menjadi inputan user

Tugas Individu

1. Buatlah rangkuman materi di atas dengan cara berikut:

- Dari file *.ipynb, berikan penjelasan tambahan lalu *convert* ke pdf (atau cukup dengan pindahkan screenshot kode ke *.doc/x, lalu berikan penjelasan dari koding yg anda buat, lalu convert ke *.pdf) yang refer dari “Latihan langsung di Kelas Ke-1 dan Latihan langsung di Kelas Ke-2”.

lalu simpan dalam satu file PDF dengan nama file, misal
“[Nama Lengkap Mhs]-[Pert. Ke-2.1/..]”

Dan dari semua tugas dalam 1 minggu di-merger, dengan nama file seperti:
“[Nama Lengkap Mhs]-[Minggu Ke-1/2/..]”, lalu cek plagiasi diturnitin dari hasil merger tersebut

> Register ke turnitin

> Masukkan **id class**: 21563495 & **enroll key**: filkomub9302



DIGITAL
TALENT
SCHOLARSHIP



DIGITAL TALENT SCHOLARSHIP 2019

Big Data Analytics



Terimakasih

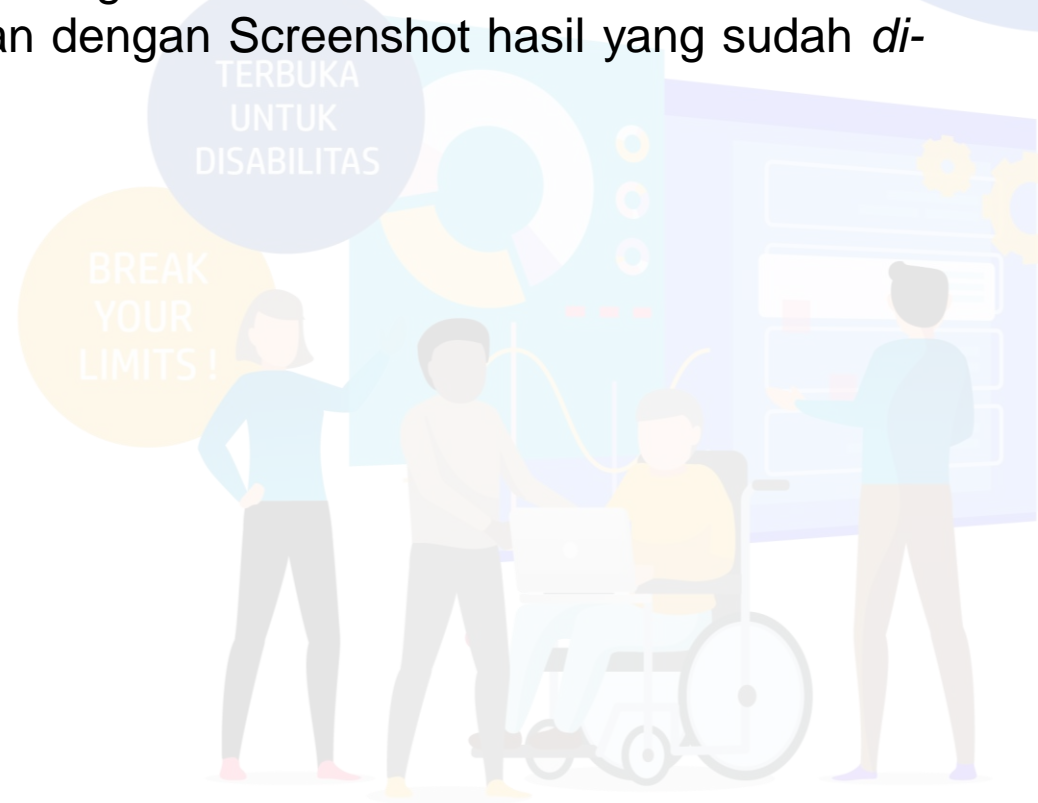
Oleh: Imam Cholissodin | imamcs@ub.ac.id, Putra Pandu Adikara, Sufia Adha Putri

Asisten: Guedho, Sukma, Anshori, Aang dan Gusti

Fakultas Ilmu Komputer (Filkom) Universitas Brawijaya (UB)

Tugas Individu

- Mengerjakan Review Questions Cognitiveclass Module 4 dan Module 5 pada cognitiveclass.ai (Dibuktikan dengan Screenshot hasil yang sudah *di-convert* ke pdf) --> Optional



Materi Tambahan

- Menyelesaikan course “PY0101EN” pada cognitiveclass.ai --> Optional
 - Module 4 Lab – Reading Files
 - Module 4 Lab – Writing Files
 - Module 4 Lab – Loading Data and Viewing Data
 - Module 5 Lab – Working with 1 D-Numpy Arrays
 - Module 5 Lab – Working with 2 D-Numpy Arrays

