

*Database Programming*

*Introduction to NumPy*

*Introduction to Dataframe  
(Pandas)*

*Learning Progress Review Week 7*

By:  
**MARVEL TEAM**

Fikrie | Natalia | Satria

# Daftar Materi

## Database Programming

- ❖ Reading File To Python
- ❖ Writing File to Python
- ❖ DB Programming in Python

## Introduction to Numpy

- ❖ Apa itu Numpy
- ❖ Penggunaan Numpy
- ❖ Membuat Matrix di Numpy
- ❖ Manipulasi Array di Numpy

## Introduction to Pandas

- ❖ Apa itu Pandas
- ❖ Data apa aja di Pandas
- ❖ Creating Dataframe
- ❖ Read Dataframe
- ❖ Selecting Dataframe

# 1. *Database Programming*

*Introduction Database Programming*

# Reading File to Python

## File

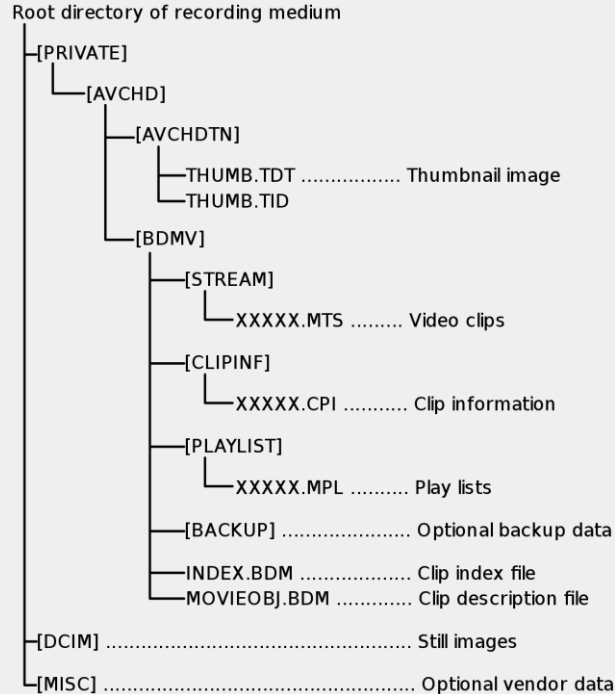
**File** adalah kumpulan susunan dari bite yang bisa digunakan untuk menyimpan data berupa tipe data numeric, character dan binary.

**File** bisa berupa Microsoft Word, Excel, Notepad dll. **File** juga bisa di simpan ke berbagai jenis extension seperti txt, doc, csv dll.

## File Path

Secara sederhana **File Path** dapat diartikan sebagai lokasi di mana file atau folder komputer berada

# Reading File to Python



## File Path

Contoh:

Bila kita ingin mengakses file Thumb.tdt dan kita berada di folder AVCHD, kita dapat menuliskan nya seperti berikut:

(\AVCHD\AVCHDTN\THUMB.TDT)

Dan apabila kita berada di folder AVCDHTN kita dapat langsung mengakses file nya:

(AVCDHTN\THUMB.TDT)

# Reading File to Python

## Open a File Stream

Ada beberapa fungsi mode pada Python yang dapat digunakan untuk open file seperti read, write, update diantaranya adalah :

Character	Meaning
'r'	open for reading (default)
'w'	open for writing, truncating the file first
'x'	open for exclusive creation, failing if the file already exists
'a'	open for writing, appending to the end of the file if it exists
'b'	binary mode
't'	text mode (default)
'+'	open a disk file for updating (reading and writing)

## Reading File

Untuk membaca sebuah file kita dapat membuatnya seperti Variabel pada Python. Fungsi read juga terbagi menjadi 3 bagian di antaranya:

- `read()` = berfungsi untuk membaca semua karakter dari file yang diakses.
- `readline()` = berfungsi untuk membaca satu baris data di dalam file.
- `readlines()` = berfungsi untuk menampilkan list yang berisi baris-baris file dari awal sampai akhir.

Contoh:

```
Output = open("Example.txt")
```

```
Print(output.read())
```

# *Writing File to Python*

## Writing File

Untuk menulis sebuah file kita dapat membuatnya seperti Variabel pada Python.

Contoh:

```
Output = open("Example.txt")
```

```
Output.write("Welcome to  
Digital Skola \n")
```

## Closing File

File yang sudah terbuka perlu ditutup Kembali menggunakan method `.close()`. Bila file yang di akses tidak di tutup Kembali, maka perubahan yang kita lakukan bisa saja hilang dan dapat menghemat memory pada notebook.

Contoh:

```
Output = open("Example.txt")
```

```
Print(output.read())
```

```
Output.close()
```



## **Closing File**

Teknik safer adalah cara yang paling aman apabila kita lupa menggunakan syntax `close()`, karena menggunakan Teknik ini kita tidak perlu lagi menggunakan method `.close()` pada statement.

Contoh:

```
With open("example.txt","r") as testfile:  
    print(testfile.read()).
```

## Introduction

Pada sesi Database Programming In Python ini akan menggunakan Library dari psycopg2.

Psycopg2 adalah adaptor database PostgreSQL paling populer untuk Bahasa Python. Psycopg2 dirancang untuk aplikasi multi-threaded yang mendukung pembuatan dan penghapusan banyak kursor secara bersamaan.

Kelebihan psycopg2 adalah result set hasil query secara otomatis di konversikan ke dalam variable tipe list.

# *DB Programming in Python*

## Install psycopg2 using pip

Install psycopg2 bertujuan untuk menghubungkan Python langsung ke Database. Jadi pada case ini kita tidak perlu menggunakan SQL secara terpisah untuk mengakses database.

```
!pip install psycopg2
```

## Database Connection

Selanjutnya kita akan melakukan connect dari Python ke database yang ingin kita tuju menggunakan fungsi connect() pada psycopg2.

```
import psycopg2

#establishing the connection
conn = psycopg2.connect(host="digitalskoladb.c04me33o8tni.ap-southeast-1.rds.amazonaws.com", port = 5432, database="sandbox", user="group_1", password="12345")

#Creating a cursor object using the cursor() method
cursor = conn.cursor()

#Executing an SQL function using the execute() method
cursor.execute("select version()")

# Fetch a single row using fetchone() method.
data = cursor.fetchone()
print("Connection established to: ",data)

#Closing the connection
conn.close()
```

# *DB Programming in Python*

## Create Table

Untuk membuat tabel di Python kita dapat menggunakan perintah CREATE TABLE dan di simpan pada variable, lalu eksekusi variable tersebut menggunakan method EXECUTE() pada psycopg2.

```
#Creating table as per requirement
sql = '''create table sandbox.batch_2.buses (
    id int,
    origin varchar,
    destination varchar,
    time time
)'''
cursor.execute(sql)
print("Table created successfully.....")
```

## Insert Data

Untuk memasukkan data ke dalam tabel kita dapat menggunakan INSERT INTO di dalam statement variabel yang kita buat. Lalu panggil variable menggunakan method .execute()

```
#Creating a cursor object using the cursor() method
cursor = conn.cursor()

sql = '''insert into sandbox.batch_2.buses values
        (100, 'Munich', 'Rome', '13:00'),
        (200, 'Munich', 'Rome', '15:30'),
        (300, 'Munich', 'Rome', '20:00')'''

cursor.execute(sql)
```

# DB Programming in Python

## Select Data

Pada perintah SELECT pada statement berguna untuk memilih kolom mana yang akan di tampilkan, kita dapat menggunakan method fetch(). Pada method fetch di bagi lagi menjadi 2 method yang di gunakan sesuai kebutuhan.

fetchall() - untuk membaca semua record

fetchone() - untuk memanggil satu record saja.

```
#Creating a cursor object using the cursor() method
cursor = conn.cursor()

sql = '''select * from sandbox.batch_2.buses'''

cursor.execute(sql)

#Fetching 1st row from the table
result = cursor.fetchone();
print(result)

#Fetching the next row from the table/ retrieves all the row in the result set of a query
result = cursor.fetchall();
print(result)

#Commit your changes in the database
conn.commit()

#Closing the connection
conn.close()
```

## Update Data

Untuk modifikasi suatu konten pada tabel di perlukan nya fungsi UPDATE di dalam statement variabel. Bila kita ingin lebih spesifik dalam memodifikasi tabel dapat menggunakan WHERE di dalam statement.

```
#Fetching all the rows before the update
print("Contents of the Employee table: ")
cursor.execute('''select * from sandbox.batch_2.buses''')
print(cursor.fetchall())

sql = '''update sandbox.batch_2.buses
        set id = 100, origin = 'Munich', destination = 'Roma', time = '14:00'
        where id = 100'''

cursor.execute(sql)
print("Table updated..... ")
```



## Delete Data

Untuk menghapus isi tabel seperti kolom atau baris kita dapat menggunakan fungsi DELETE dan bila ingin lebih spesifik kolom mana atau baris mana yang ingin di hapus dapat menambahkan fungsi WHERE

```
#Deleting records
cursor.execute('delete from sandbox.batch_2.buses where id = 100')

#Retrieving data after delete
print("Contents of the table after delete operation ")
cursor.execute("select * from sandbox.batch_2.buses")
print(cursor.fetchall())
```

## Drop Table

Bila kita ingin menghapus tabel dapat menggunakan fungsi DROP di dalam method .execute().

```
#Creating a cursor object using the cursor() method
cursor = conn.cursor()

#Dropping buses table if already exists
cursor.execute("drop table sandbox.batch_2.buses")
print("Table dropped... ")

#Commit your changes in the database
conn.commit()

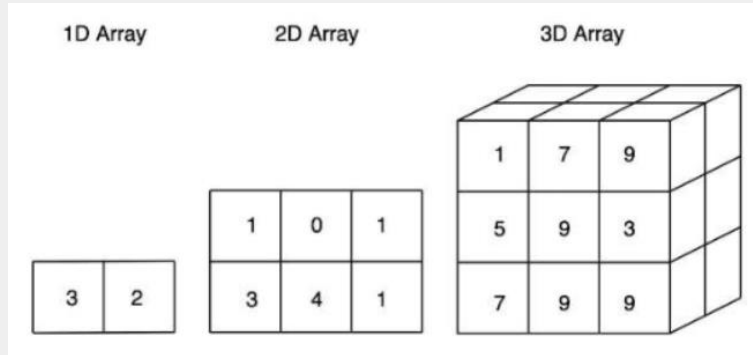
#Closing the connection
conn.close()
```

## 2. *Introduction to NumPy*

NumPy adalah core Library untuk untuk Scientific Computing di Python

## Apa itu Numpy?

- ❖ Numpy adalah *core Library* untuk untuk *Scientific Computing* di Python.
- ❖ Memiliki performa yang cepat untuk multidimensi Array.



- ❖ Scikit-Learn & Pandas dibangun di atas Numpy.
- ❖ *Open Source* sehingga dapat digunakan secara gratis.

## Penggunaan Numpy

- ❖ Setiap Data Scientist mayoritas bekerja di Python menggunakan Numpy.
- ❖ Array seringkali digunakan oleh Data Scientist, karena *speed* dan *resources* yang cepat.

## Numpy Dasar

### 1 Dimensi

```
[ ] a = np.array([1, 2, 3, 4, 5])  
    print(a)
```

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

```
[ ] print(a.ndim)
```

```
1
```

### 2 Dimensi

```
[ ] b = np.array([[1, 2, 3, 4, 5],  
                  [6, 7, 8, 9, 10]  
                  ])  
    print(b)
```

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

```
[ ] print(b.ndim)
```

```
2
```

## Numpy Dasar

### 3 Dimensi

```
[ ] c = np.array([[1, 2, 3, 4, 5],  
                 [6, 7, 8, 9, 10]],  
                 [[11, 12, 13, 14, 15],  
                 [16, 17, 18, 19, 10]])  
  
print(c)
```

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

```
[ ] print(c.ndim)
```

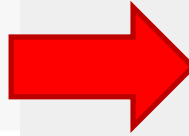
3

## Mendapatkan/Mengganti Elemen, Baris atau Kolom

### 2 Dimensi

```
[ ] d = np.array([[1, 2, 3, 4, 5],  
                  [6, 7, 8, 9, 10]  
                  ])  
  
print(d)
```

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



```
[ ] #Mendapatkan elemen tertentu  
d[1,3]
```

```
9
```

```
[ ] #Mendapatkan baris tertentu  
d[0, 0:3]
```

```
array([1, 2, 3])
```

```
[ ] #Mendapatkan kolom tertentu  
d[:, 3]
```

```
array([89, 90])
```

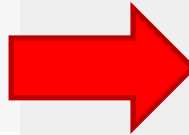


## Mendapatkan/Mengganti Elemen, Baris atau Kolom

### 2 Dimensi

```
[ ] d = np.array([[1, 2, 3, 4, 5],  
                  [6, 7, 8, 9, 10]  
                  ])  
  
print(d)
```

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



```
[ ] #Mengganti elemen tertentu  
d[1, 4] = 88  
print(d)
```

```
[[ 1  2  3  4  5]  
 [ 6  7  8  9 88]]
```

```
[ ] #Mengganti kolom tertentu  
d[:, 3] = [89, 90]  
print(d)
```

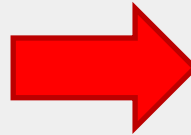
```
[[ 1  2  3 89  5]  
 [ 6  7  8 90 88]]
```

## Mendapatkan/Mengganti Elemen, Baris atau Kolom

### 3 Dimensi

```
[ ] e = np.array([[1, 2, 3, 4, 5],  
                  [6, 7, 8, 9, 10]],  
                  [[11, 12, 13, 14, 15],  
                  [16, 17, 18, 19, 10]])  
  
print(e)
```

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



```
[ ] #Mendapatkan elemen tertentu  
e[1, 1, 4]
```

```
10
```

```
[ ] #Mengganti elemen tertentu  
e[:, 1, :] = [[72, 73, 74, 75, 76]]  
print(e)
```

```
[[[ 1  2  3  4  5]  
  [72 73 74 75 76]]  
  
 [[11 12 13 14 15]  
  [72 73 74 75 76]]]
```

## Inisialisasi Berbagai Jenis Array

```
[2] #Matriks 0  
np.zeros((2,3))
```

```
array([[0., 0., 0.],  
       [0., 0., 0.]])
```

```
[4] #Matriks 1  
np.ones((2, 2, 4))
```

```
array([[[1., 1., 1., 1.],  
        [1., 1., 1., 1.]],  
       [[1., 1., 1., 1.],  
        [1., 1., 1., 1.]])
```

```
[5] #Nomor tertentu  
np.full((3, 4), 88)
```

```
array([[88, 88, 88, 88],  
       [88, 88, 88, 88],  
       [88, 88, 88, 88]])
```

```
[6] #Pengulangan Matriks  
i = np.array([[23, 34, 56]])  
j = np.repeat(i, 4, axis=0)  
print(j)
```

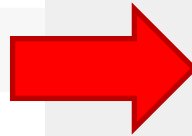
```
[[23 34 56]  
 [23 34 56]  
 [23 34 56]  
 [23 34 56]]
```

## Fungsi Matematika

```
[7] k = np.array([87, 88, 89, 23, 45])
```

```
[8] print(k)
```

```
[87 88 89 23 45]
```



```
[9] k + 5
```

```
array([92, 93, 94, 28, 50])
```

```
[11] k / 4
```

```
array([21.75, 22.  , 22.25,  5.75, 11.25])
```

```
[12] m = np.array([23, 34, 56, 78])
```

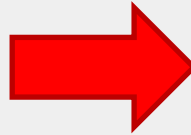
```
n = np.array([90, 98, 87, 76])
```

```
m + n
```

```
array([113, 132, 143, 154])
```

## Fungsi Statistik

```
[14] stat = np.array([[32, 43, 56, 54], [65, 43, 34, 21]])  
stat  
  
array([[32, 43, 56, 54],  
       [65, 43, 34, 21]])
```



```
[15] np.min(stat)
```

21

```
[16] np.sum(stat)
```

348

```
[18] np.median(stat)
```

43.0

```
[19] np.mean(stat)
```

43.5

```
[20] np.average(stat)
```

43.5

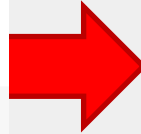
## Reshaping Array

```
[22] x = np.array([[32, 43, 56, 54], [65, 43, 34, 21]])  
      print(x)
```

```
[[32 43 56 54]  
 [65 43 34 21]]
```

```
[23] print(x.shape)
```

```
(2, 4)
```



```
[24] #Mengganti 2x4 ke 4x2  
      y = x.reshape((4, 2))  
      print(y)
```

```
[[32 43]  
 [56 54]  
 [65 43]  
 [34 21]]
```

```
[25] #Mengganti 2x4 ke 2x2x2  
      z = x.reshape((2, 2, 2))  
      print(z)
```

```
[[[32 43]  
  [56 54]]  
  
 [[65 43]  
  [34 21]]]
```

### 3. *Introduction to dataframe* (Pandas)

Pandas adalah salah satu *library* Python untuk analisis dan manipulasi data

## Apa itu Pandas?



**Pandas** adalah salah satu *library* Python untuk analisis dan manipulasi data. Pandas termasuk *open source* yang cepat, *powerful*, fleksibel dan mudah digunakan.



Data apa aja  
di dalam  
Pandas?

**Tipe data dalam Pandas** umumnya ada 2 yaitu:

1. Series **[pd]**
2. Dataframe : data berbentuk table dan pada dasarnya adalah jenis data dictionary.

Dalam membuat dataframe dapat disusun dari 4 tipe data yaitu :

1. List                    [...]
2. Series                [pd]
3. Array                [np]
4. Tuple                (...)
5. Dictionary        {...}

# Creating Dataframe

## Dari List [...]

Menggunakan  
syntax .DataFrame  
untuk membuat  
Dataframe

```
[3] list1 = [1,2,3]
     list2 = ['Lulu', 'Lili', 'Lala']
     column_names = ['number', 'name']
     df = pd.DataFrame(list(zip(list1, list2)), columns = column_names)
     df
```

Dalam List, wajib 1 tipe data.  
Kemudian, jumlah isi data  
harus sama.

zip untuk  
menggabungkan list

untuk nama kolom

	number	name
0	1	Lulu
1	2	Lili
2	3	Lala

# Creating Dataframe

## Dari Array [np]

Membuat Array dari list menggunakan NumPy

```

▶ no_induk = [100, 102, 103]
  nama = ['A', 'B', 'C']
  nilai = [9.4, 7.6, 10.0]
  siswa = np.array([no_induk, nama, nilai])
  siswa

↳ array([[ '100', '102', '103'],
        [ 'A', 'B', 'C'],
        [ '9.4', '7.6', '10.0']], dtype='<U21')
    
```

Dari Array diubah ke  
Dataframe

```

▶ df2 = pd.DataFrame(siswa)
  df2
    
```

```

↳
      0    1    2
0  100  102  103
1    A    B    C
2   9.4   7.6  10.0
    
```

# Creating Dataframe

## Dari Series [pd]

```
no_induk = pd.Series(no_induk)
nama = pd.Series(nama)
nilai = pd.Series(nilai)
df3 = pd.DataFrame(no_induk, columns = ['no induk'])
df3['nama'] = nama
df3['nilai'] = nilai
df3
```



	no induk	nama	nilai
0	100	A	9.4
1	102	B	7.6
2	103	C	10.0

Transform Array  
sebelumnya ke Series

Ketika membuat  
dataframe dari Series,  
harus dari 1 kolom dulu.  
Kemudian digabung  
dengan kolom lainnya  
dengan cara  
memanggil dataframe  
yang sudah jadi.

# Creating Dataframe

## Dari Tuple (...)

```
▶ numbers = tuple([1,2,3,4])
  alphabets = tuple(['A', 'B', 'C', 'D'])
  df4 = pd.DataFrame([numbers, alphabets]).T
  df4
```

```
↳
```

	0	1
0	1	A
1	2	B
2	3	C
3	4	D

} Cara membuat Tuple harus sebut fungsi tuple()  
 → Ketika membuat dataframe dari Tuple, cukup memanggil variabelnya. Hasil default-nya adalah ke samping atau menjadi baris. Untuk mengubahnya, gunakan metode transpose (.T) .

## Creating Dataframe

### Dari Dictionary {...}

```

▶ kota = ['Jakarta', 'Medan', 'Banyuwangi']
  jml_pdd = [15000000, 7000000, 1500000]
  dict_kota = {
      'kota' : kota,
      'jumlah penduduk' : jml_pdd
  }
  df5 = pd.DataFrame(dict_kota)
  df5
    
```

↗

	kota	jumlah penduduk
0	Jakarta	15000000
1	Medan	7000000
2	Banyuwangi	1500000

- Membuat variabelnya dahulu
- Kemudian, membuat dictionary.  
Nama key ('kota', 'jumlah penduduk') otomatis menjadi nama kolom.

# Load Dataset dan Read Dataframe

1. Memasukkan data csv ke dalam google drive, kemudian di Python, data tersebut di-mount menggunakan metode .mount

```
▶ from google.colab import drive
   drive.mount('/content/gdrive')
```

↳ Mounted at /content/gdrive

2. Kemudian, membuat variable untuk path. Lalu, me-loading/memasukkan data csv ke dalam Google Collab. Ingat, delimiter digunakan tergantung isi datanya.

```
▶ path = '/content/gdrive/MyDrive/DigitalSkola/Dataset/sample.csv'
   dataset = pd.read_csv(path, delimiter = ',')
   dataset
```

↳

	order_id	sales	profit	quantity	category	sub_category	Cost
0	AZ-2011-2320734	13	2	1	Office Supplies	Paper	1
1	AZ-2014-1914770	92	42	2	Office Supplies	Art	5
2	BN-2011-5443120	205	-127	8	Office Supplies	Art	33
3	BN-2014-2645644	48	-22	2	Office Supplies	Storage	7

## Selecting Dataframe



```
dataset.head()
```


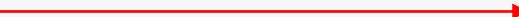



	order_id	sales	profit	quantity	category
0	AZ-2011-2320734	13	2	1	Office Supplies
1	AZ-2014-1914770	92	42	2	Office Supplies
2	BN-2011-5443120	205	-127	8	Office Supplies
3	BN-2014-2645644	48	-22	2	Office Supplies
4	AZ-2012-9764957	75	26	5	Office Supplies

Men-select  
5 data  
teratas

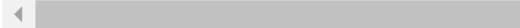


## Selecting Dataframe

 `dataset.tail()` 



	order_id	sales	profit	quantity	category
1488	AZ-2014-5931184	12	1	3	Office Supplies
1489	AZ-2011-8034411	54	12	1	Office Supplies
1490	AZ-2013-8360642	38	6	2	Furniture
1491	AZ-2014-4726682	122	43	7	Office Supplies
1492	AZ-2014-2406788	32	4	1	Furniture



Men-select  
5 data  
terbawah

## Get Column Names

[ ] `print(dataset.columns)` → Menggunakan metode `.columns`

```
Index(['order_id', 'sales', 'profit', 'quantity', 'category', 'sub_category',
      'Cost', 'Total_Cost', 'Total_profit', 'order_date', 'customer_name',
      'city', 'country', 'region', 'segment', 'ship_date', 'ship_mode', 'lon',
      'lat'],
      dtype='object')
```

## Get Spesific Columns

Nama kolomnya

```
[ ] customer_trx = dataset[['order_id', 'sales', 'customer_name']]
```

```
[ ] customer_trx.head()
```

	order_id	sales	customer_name
0	AZ-2011-2320734	13	Charlie Wells
1	AZ-2014-1914770	92	Cooper Burn
2	BN-2011-5443120	205	Kaitlyn Dorsch
3	BN-2014-2645644	48	Jeremy Pilcher
4	AZ-2012-9764957	75	Abbie Perry

## Get Spesific Column(s)

```
dataset['segment'] → [ ]
```

menampilkan kolom sebagai series

0	Home Office
1	Consumer
2	Corporate
3	Corporate
4	Consumer
...	...
1488	Home Office
1489	Home Office
1490	Consumer
1491	Consumer
1492	Consumer

Name: segment, Length: 1493, dtype: object

```
dataset[['segment']] → [ [ ] ]
```

segment	
0	Home Office
1	Consumer
2	Corporate
3	Corporate
4	Consumer
...	...
1488	Home Office
1489	Home Office
1490	Consumer
1491	Consumer

menampilkan kolom sebagai dataframe karena [ ] pertama mewakili kolom, dan [ ] kedua mewakili baris.

## Get Spesific Row(s)



dataset[0:1]



Memanggil  
baris tertentu



	order_id	sales	profit	quantity	cate
0	AZ-2011-2320734	13	2	1	Sup

[... : ...] berarti  
baris mana yang  
ingin diambil



dataset.iloc[:10]



Memanggil baris  
tertentu bisa  
menggunakan  
metode .iloc



	order_id	sales	profit	quantity	ca
0	AZ-2011-2320734	13	2	1	ε
1	AZ-2014-	92	42	2	

## Get Spesific Row(s) and Column(s)

Baris Kolom

`dataset.iloc[100:200, :3]`

	order_id	sales	profit
100	AZ-2012-9553455	30	15
101	AZ-2014-9580273	94	7
102	AZ-2014-1490273	1280	299
103	BN-2013-3350047	1554	730

Menggunakan metode .iloc maka baris terakhir yang disebutkan tidak terpanggil.

`# loc 200 nya termasuk`  
`dataset.loc[100:199, ['segment', 'country', 'Cost']]`

	segment	country	Cost
100	Corporate	Spain	15
101	Consumer	Germany	87
102	Consumer	Germanv	981

Menggunakan metode .loc maka baris terakhir yang disebutkan akan termasuk dalam panggilan

# Tipe Data dan *Statistic Descriptive*

```
dataset.dtypes
```

order_id	object
sales	int64
profit	int64
quantity	int64
category	object
sub_category	object
Cost	int64
Total_Cost	int64
Total_profit	int64
order_date	object
customer_name	object
city	object
country	object
region	object
segment	object
ship_date	object
ship_mode	object
lon	float64
lat	float64
dtype:	object

Cara memanggil  
tipe data

note


int : integer

object : string

# Tipe Data dan *Statistic Descriptive*

Statistic descriptive antara lain:

- Count = menghitung jumlah data
- Mean = rata-rata
- Std = Standard deviasi
- Min/Max = Minimum/Maximum
- Percentile (25%, 50%, 75%)

 dataset.describe()

→ Untuk mendapatkan Statistic descriptive

	sales	profit	quantity	Cost	Total_Cost	Total_profit	lon	lat
<b>count</b>	1493.000000	1493.000000	1493.000000	1493.000000	1493.000000	1493.000000	1493.000000	1493.000000
<b>mean</b>	306.198259	38.541192	4.060281	252.385800	267.657066	193.272605	5.235207	48.782571
<b>std</b>	517.356975	165.200903	2.477320	432.386435	455.831143	1301.324828	6.485921	5.021450
<b>min</b>	4.000000	-1752.000000	1.000000	3.000000	3.000000	-22776.000000	-9.224547	35.889387
<b>25%</b>	52.000000	1.000000	2.000000	39.000000	42.000000	1.000000	-0.127758	45.387638
<b>50%</b>	117.000000	15.000000	3.000000	93.000000	101.000000	44.000000	5.121420	49.258329
<b>75%</b>	319.000000	50.000000	5.000000	255.000000	278.000000	204.000000	9.993682	52.120533
<b>max</b>	5785.000000	1898.000000	16.000000	5380.000000	5380.000000	18980.000000	25.037769	60.794533



# Sampling Dataset

Mau ambil data  
berdasarkan  
Persentase

Mengambil data  
yang diinginkan  
secara berulang

```
sampled_data = dataset.sample(frac=0.5, random_state=12)
sampled_data.head()
```

	order_id	sales	profit	quantity	category	sub_category	Cost	Total_Cos
502	AZ-2011-5463300	21	9	2	Office Supplies	Art	12	1
1180	AZ-2011-6683192	36	8	3	Office Supplies	Labels	28	2
1094	AZ-2011-9119536	112	55	4	Office Supplies	Paper	57	5
649	AZ-2013-	311	-292	2	Office	Annliances	603	60

Special Thanks to :



Slide template by SlideCarnival