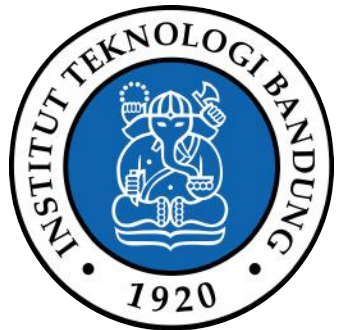
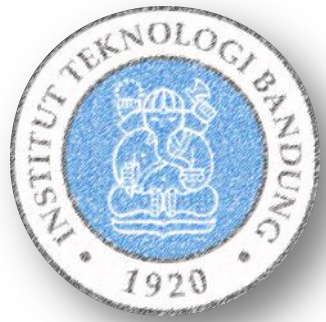


# Descriptive Analytics dengan Python Pandas

Tim Penyusun Materi Pengenalan Komputasi  
Institut Teknologi Bandung © 2019

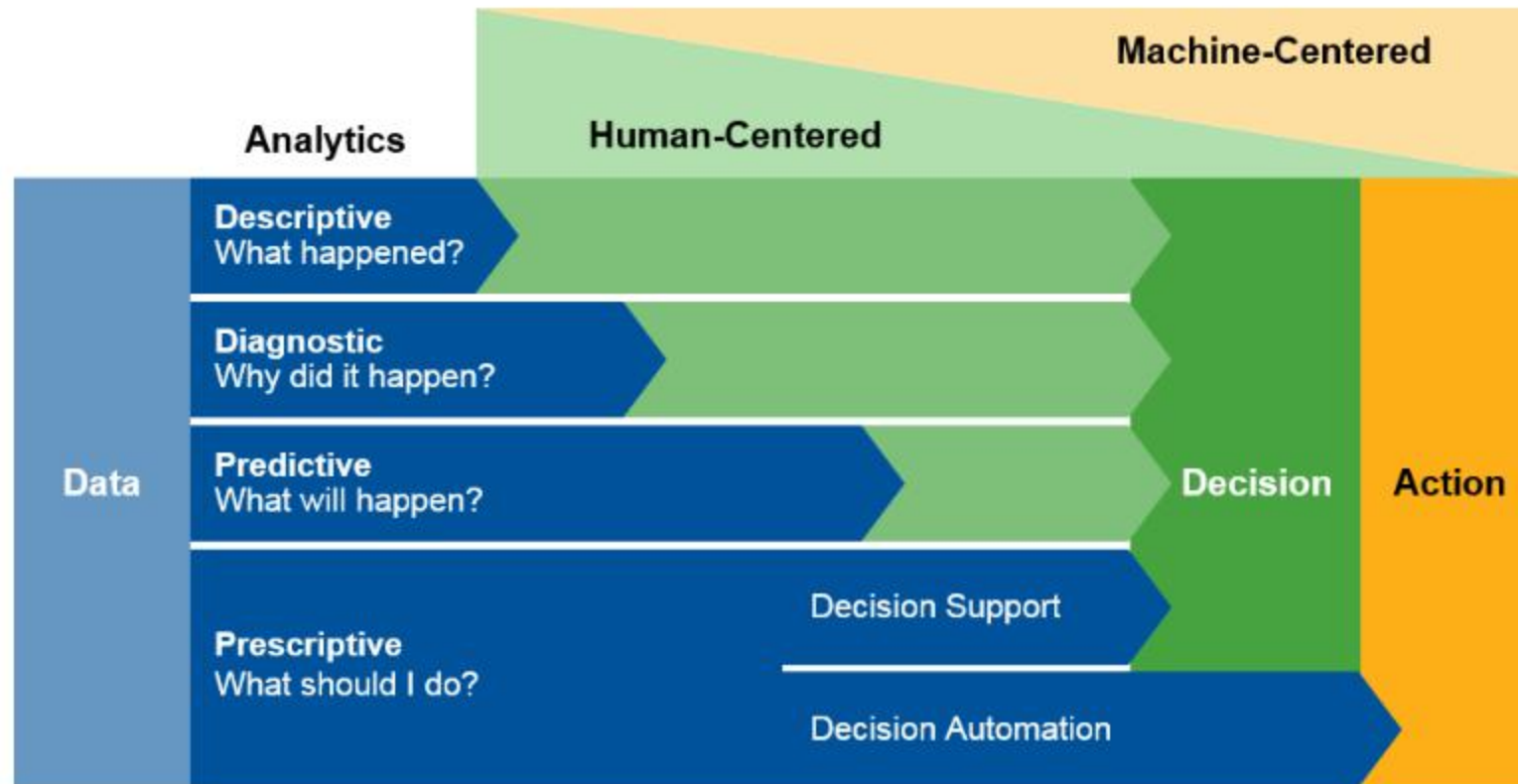




# Objektif

- Mahasiswa dapat mempraktikkan teknik dalam Descriptive Analytics dan EDA untuk melakukan analisis data sederhana dengan menggunakan **Python Pandas**

# Klasifikasi Data Analytics



Source: Gartner (October 2016)

Sumber:  
2017 Planning Guide for Data and Analytics



# Descriptive Analytics

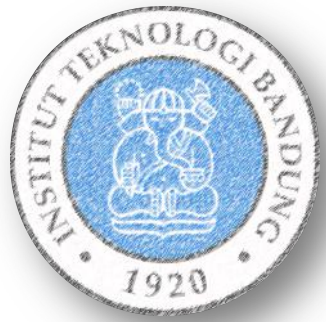
- Jenis paling sederhana dari Data Analytics
- Analisis terhadap data history untuk mendapatkan profil umum dalam bentuk summary dari data atau hubungan antar data untuk menjelaskan situasi yang telah terjadi.
- Contoh hasil analisis:
  - Banyaknya friend, mention, followers, page views
  - Banyaknya page views
  - Perbandingan banyaknya mahasiswa antar prodi di ITB
  - Rata-rata nilai mahasiswa peserta PTIB
  - Hubungan antara banyaknya jam belajar dengan prestasi akademik
  - Ada kecenderungan bahwa orang beli **roti tawar** bersamaan dengan **butter/mentega**
  - dll

# Contoh-contoh kegiatan Data Analysis Descriptive Analytics dan Exploratory DA



- Retrieve Value (Selection)
- Filter
- Compute Derived Value
- Find Extremum
- Sort
- Determine Range
- Characterize Distribution
- Find Anomalies
- Correlation
- Clustering

[https://en.wikipedia.org/wiki/Data\\_analysis](https://en.wikipedia.org/wiki/Data_analysis)



# Tipe Data

- Categorical-Nominal
  - Nama negara, warna kulit, nama program studi, dll
- Categorical-Ordinal
  - Likert scale (“sangat setuju” s.d. “sangat tidak setuju”)
  - Indeks nilai A, B, C, D, E
- Categorical-Binary
  - Jenis kelamin, status mahasiswa (aktif, tidak aktif), dll
- Quantitative-Discrete
  - Banyaknya anak, banyaknya mahasiswa, banyaknya sks lulus
- Quantitative-Continues
  - Usia, berat badan, tinggi, suhu

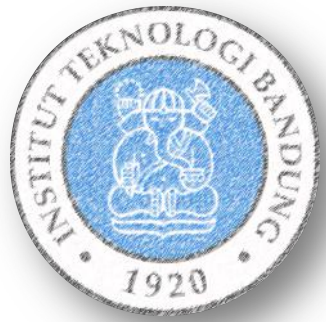


# Pandas

- Pandas = Python Data Analysis Library
- Dikembangkan We McKinney 2008
- Sebuah *library open source* yang menyediakan model data terstruktur dan fungsi manipulasi dan analisis data dalam lingkungan bahasa pemrograman Python
- Instalasi Pandas: dalam paket Anaconda (<http://www.anaconda.com>)
- Model data pada Pandas disebut **DataFrame**: data dalam bentuk tabular (tabel dengan kolom dan baris)

<https://pandas.pydata.org/>

Subjek	Jam belajar	IPK
1	33	3.9
2	32	3.5
3	21	3.2
4	34	3.3
5	34	3.5
6	35	3.8
7	32	3.7
8	21	3.3
9	21	3.2
10	35	3.6



# Create DataFrame

```
import pandas as pd

dataMhs = {"NIM": ["16519001", "16519002", "16519003"],
           "Nama" : ["Bin Bin", "Atung", "Kaka"],
           "NilaiPTIB": [75, 80, 90]}

df1 = pd.DataFrame(data=dataMhs)
print(df)
```

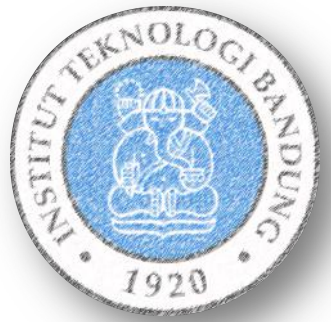
	NIM	Nama	NilaiPTIB
0	16519001	Bin Bin	75
1	16519002	Atung	80
2	16519003	Kaka	90

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

df2 = pd.DataFrame(d)
```

	one	two
a	1.0	1.0
b	2.0	2.0
c	3.0	3.0
d	NaN	4.0





# Baca dan Tampilkan Data

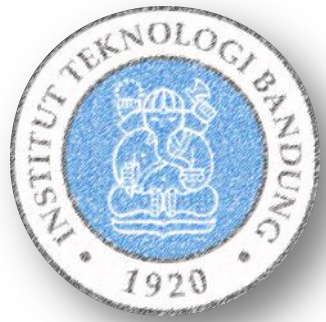
- Baca Data dari File CSV

```
import pandas as pd
df3 = pd.read_csv("D:/medali.csv")
print(df3)
```

- Baca Data dari File Spreadsheet (.xlsx)

```
import pandas as pd
sn = "Sheet1"
df4 = pd.read_excel("D:/medali.xlsx", sheet_name=sn)
print(df4)
```

Sumber data: <https://www.bola.com/pages/perolehan-medali/>



# Tulis Data

- Tulis Data ke File CSV

```
# Asumsi df3 sudah terdefinisi (lihat slide sebelumnya)
df3.to_csv("D:/medali_out.csv")
```

- Tulis Data ke File Spreadsheet (.xlsx)

```
# Asumsi df4 sudah terdefinisi (lihat slide sebelumnya)
writer = pd.ExcelWriter("D:/medali_out.xlsx")
df4.to_excel(writer, "Sheet1")
df4.to_excel(writer, "Sheet2") #tulis data yang sama di sheet lain
writer.save()
```

# Select Data (1): Subset, Filter

- DataFrame menggunakan indeks mulai dengan nilai 0
- Contoh-contoh:

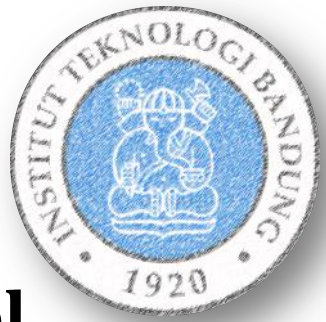
```
#Dapatkan subset mulai dari baris ke-5  
#(dari indeks ke-4)  
df3[4:]
```

```
#dapatkan mulai baris ke-5 (idx=4),  
#s.d. baris ke-8 (idx=7)  
df3[4:8]
```

```
#dapatkan semua baris  
#s.d. baris ke-8 (idx=7)  
df3[:8]
```



	rank	country	gold	silver	bronze	total
4	5	Uzbekistan	21	24	25	70
5	6	IR Iran	20	20	22	62
6	7	Chinese Taipei	17	19	31	67
7	8	India	15	24	30	69



# Select Data (2): Subset, Filter

**DataFrame.loc:** mengakses baris dan kolom berdasarkan **label** atau array of boolean

```
df3.index
```

```
RangeIndex(start=0, stop=46, step=1)
```

```
df3.columns
```

```
Index(['rank', 'country', 'gold', 'silver', 'bronze', 'total'], dtype='object')
```

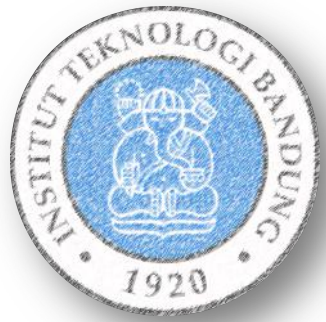
```
#data baris ke-4, kolom country
```

```
df3.loc[3, "country"] #Indonesia
```

```
#data dari baris ke-4 (idx=3) s.d. ke-10 (idx=9);
```

```
#kolom country s.d. bronze
```

```
df3.loc[3:9, "country": "bronze"]
```



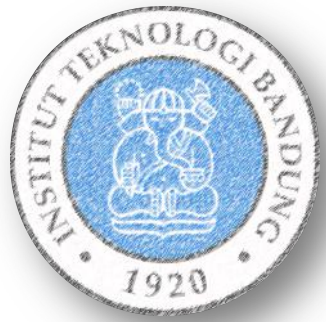
# Select Data (3): Berdasar kondisi

```
#Ambil semua data dengan jumlah medali perunggu = 0  
df3.loc[df3["bronze"] == 0]
```

```
#Cetak semua data dengan total medali >= 50  
print(df3.loc[df3["total"] >= 50])
```

```
#Cetak semua data negara yang hanya mendapat medali perunggu  
print(df3.loc[(df3["gold"] == 0) & (df3["silver"] == 0) &  
              (df3["bronze"] > 0)])
```

```
#Ambil semua data negara yang tidak mendapatkan medali emas atau  
#perak atau perunggu  
df3.loc[(df3["gold"] == 0) | (df3["silver"] == 0) |  
        (df3["bronze"] == 0)]
```



# Select Data (3a)

## Latihan-1

- Tuliskan perintah dan tuliskan hasilnya untuk beberapa perintah berikut:
  1. Tampilkan data 10 negara ranking pertama.
  2. Tampilkan daftar negara rangking 11 s.d. 20 (nama negaranya saja yang ditampilkan)
  3. Tampilkan data negara yang mendapatkan 1 medali emas.
  4. Tampilkan data negara dengan total perolehan medali  $> 20$ , tetapi dengan jumlah perolehan medali emas  $< 5$

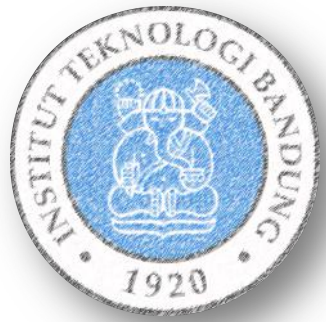
# Select Data (3b)

## Latihan-1

- Contoh perintah dan hasil:
  3. Tampilkan data negara yang mendapatkan 1 medali emas.

```
print(df3.loc[df3["gold"] == 1])
```

	rank	country	gold	silver	bronze	total
24	25	Kingdom of Saudi Arabia	1	2	3	6
25	26	Macau China	1	2	2	5
26	27	Iraq	1	2	0	3
27	28	Korea	1	1	2	4
28	28	Lebanon	1	1	2	4



# Find Extremum

```
#Temukan indeks dengan nilai maksimum
#pada kolom bronze
imax = df3["bronze"].idxmax()
df3[imax:imax+1]
```

	rank	country	gold	silver	bronze	total
1	2	Japan	75	56	74	205

```
#Temukan indeks dengan nilai minimum
#pada kolom bronze
#Jika nilai minimum ada lebih dari
#satu, maka baris yang ditemukan
#pertama kali
imin = df3["bronze"].idxmin()
df3[imin:imin+1]
```

	rank	country	gold	silver	bronze	total
26	27	Iraq	1	2	0	3



# Determine Range

```
#dapatkan nilai minimum dan maximum
#untuk seluruh data
minimum,maximum=(df3.min(),df3.max())
print(minimum); print (maximum)
```

```
#nilai minimum untuk kolom country
minimum["country"]
```

```
#nilai maksimum untuk kolom gold
maximum["gold"]
```

```
print(minimum); print (maximum)
```

```
rank      1
country   Afghanistan
gold      0
silver    0
bronze    0
total     0
dtype: object
rank      39
country   Yemen
gold     132
silver    92
bronze    74
total    289
dtype: object
```

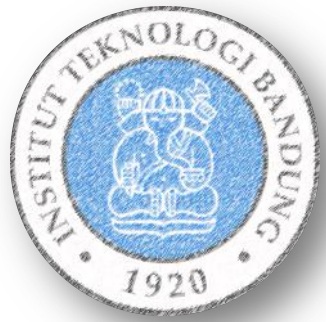
# Latihan-2

- Tampilkan negara(-negara) dengan perolehan perunggu terbanyak

	rank	country	gold	silver	bronze	total
1	2	Japan	75	56	74	205

- Tampilkan negara(-negara) dengan perolehan perunggu paling sedikit, tapi total perolehan medali  $> 0$

	rank	country	gold	silver	bronze	total
26	27	Iraq	1	2	0	3
32	33	Nepal	0	1	0	1
33	33	Oman	0	1	0	1



# Sort

```
#Sort terurut menurun (ascending=0) berdasarkan kolom country  
print(df3.sort_values(["country"], ascending=[0]))
```

```
#Sort terurut menaik (ascending=1) berdasarkan kolom total  
print(df3.sort_values(["total"], ascending=[1]))
```

```
#Sort terurut menaik berdasarkan kolom total dan menurun  
#berdasarkan kolom country  
print(df3.sort_values(["total", "country"], ascending=[1,0]))
```

## Latihan-3

Sort data perolehan medali terurut **menurun** berdasarkan **kolom gold** dan terurut **menaik** berdasarkan **kolom silver**


# Counting Frequency

Menghitung banyaknya kemunculan suatu data item pada suatu kolom → distribusi kemunculan nilai

```
#Counting frequency kolom bronze  
df3["bronze"].value_counts()
```

## Latihan-4

Buatlah distribusi frekuensi untuk **total** perolehan medali.



```
0      11  
2       7  
3       4  
1       2  
12      1  
65      1  
4       1  
5       1  
70      1  
7       1  
9       1  
74      1  
11      1  
46      1  
44      1  
14      1  
15      1  
16      1  
18      1  
20      1  
22      1  
25      1  
30      1  
31      1  
43      1  
13      1  
Name: bronze, dtype: int64
```

# Group By (1)

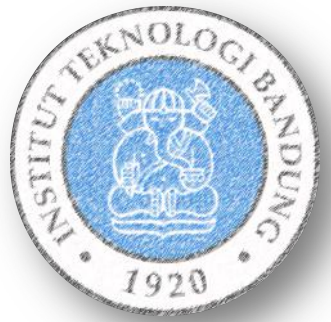
name	age	gender	state	num_children	num_pets
john	23	M	CA	2	5
marry	78	F	DC	0	1
peter	22	M	CA	0	0
jeff	19	M	DC	3	5
bill	45	M	CA	2	2
lisa	33	F	TX	1	2
jose	20	M	TX	4	3

- Group by adalah perintah untuk membagi data dalam kelompok-kelompok berbeda berdasarkan suatu variabel tertentu
- Contoh-1: mengelompokkan data berdasarkan atribut

```
df5 = pd.read_csv("D:/data.csv")
df5.groupby("gender")
```
- Fungsi `sum()`, `max()`, `min()`, `mean()`, `first()`, `last()`, `size()` dapat diberlakukan pada objek yang di-groupby untuk mendapatkan statistik
- Contoh-2: mendapatkan total jumlah anak (`num_children`) per kelompok gender

```
df5.groupby("gender")["num_children"].sum()
```
- Contoh-3: mendapatkan rata-rata jumlah peliharaan (`num_pets`) per kelompok gender per state

```
df5.groupby(["gender", "state"])["num_pets"].mean()
```



# Group By (2)

```
df5.groupby("gender")["num_children"].sum()
```

```
gender
F      1
M     11
Name: num_children, dtype: int64
```

```
df5.groupby(["gender", "state"])["num_pets"].mean()
```

```
gender  state  num_pets
F      DC      1.000000
      TX      2.000000
M      CA      2.333333
      DC      5.000000
      TX      3.000000
Name: num_pets, dtype: float64
```

# Group By (3)

- Membuat tabel pivot
- Contoh: membuat tabel pivot berdasarkan atribut gender (sebagai baris) dan state (sebagai kolom), sel berisi banyaknya data gender per state

```
df5.groupby(["gender", "state"])["name"].size().unstack()
```

state	CA	DC	TX
gender			
F	NaN	1.0	1.0
M	3.0	1.0	1.0





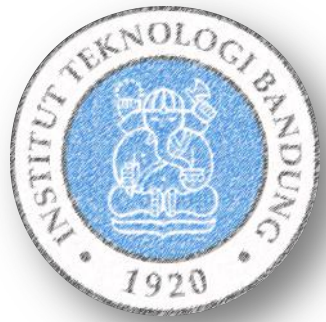
# Statistics Summary, Characterize Distribution

Menampilkan beberapa statistik penting pada data

```
df3.describe()
```

	rank	gold	silver	bronze	total
<b>count</b>	46.000000	46.000000	46.000000	46.000000	46.000000
<b>mean</b>	22.847826	10.108696	10.130435	13.521739	33.760870
<b>std</b>	12.516598	23.073051	17.917848	19.512662	58.264601
<b>min</b>	1.000000	0.000000	0.000000	0.000000	0.000000
<b>25%</b>	12.250000	0.000000	0.000000	1.000000	2.000000
<b>50%</b>	23.500000	2.000000	2.000000	3.000000	6.500000
<b>75%</b>	34.500000	10.250000	15.250000	17.500000	37.750000
<b>max</b>	39.000000	132.000000	92.000000	74.000000	289.000000





# Rata-Rata dan Deviasi Standar

Menghitung rata-rata (*mean*) dan deviasi standar (*std*) pada data

```
#rata-Rata data df3
```

```
df3.mean()
```

```
#deviasi standar data df3
```

```
df3.std()
```

```
#rata-rata total perolehan medali
```

```
df3.mean() ["total"]
```

```
#deviasi standar perolehan medali emas
```

```
df3.std() ["gold"]
```

```
df3.mean()
```

rank	22.847826
gold	10.108696
silver	10.130435
bronze	13.521739
total	33.760870
dtype:	float64

```
df3.std()
```

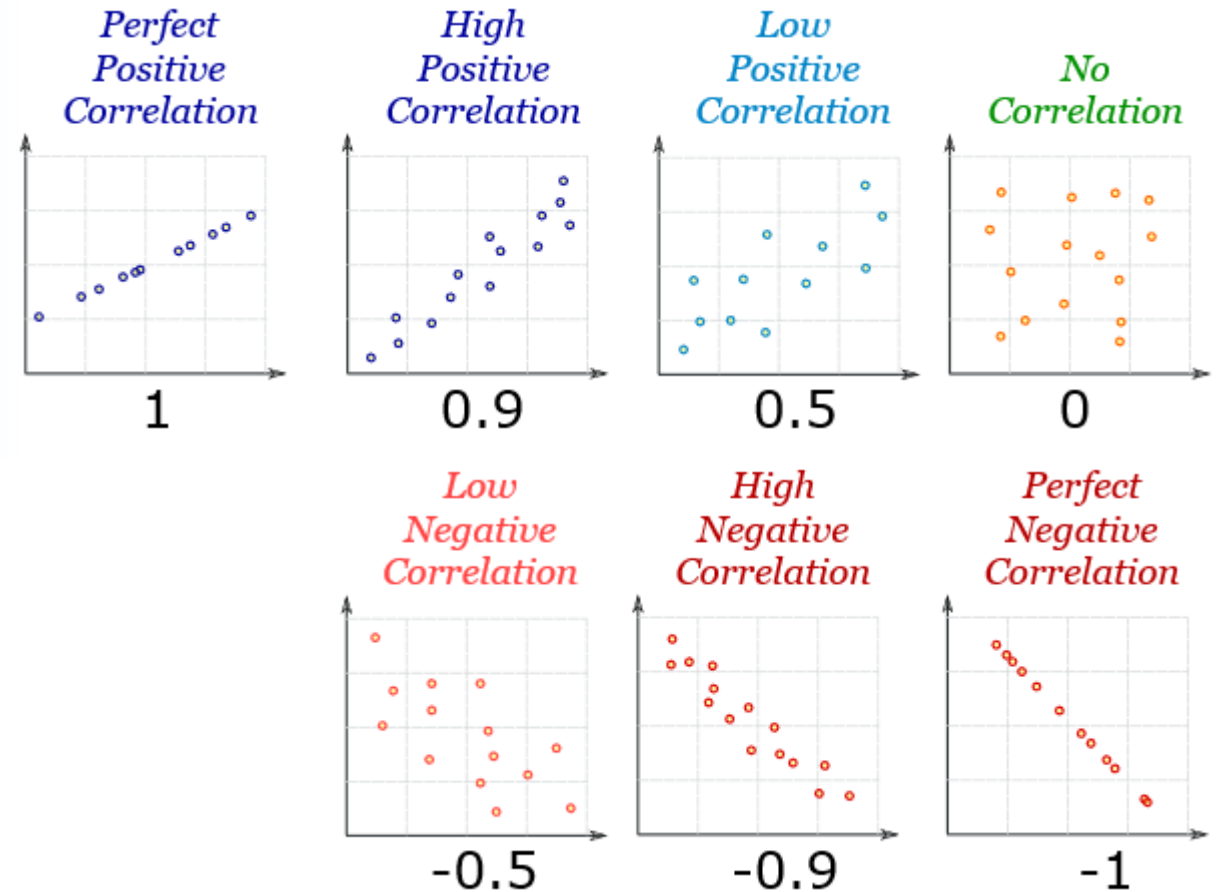
rank	12.516598
gold	23.073051
silver	17.917848
bronze	19.512662
total	58.264601
dtype:	float64

# Correlation (1)

Correlation adalah indikasi hubungan antara dua atau lebih variable, yang dinyatakan dalam ***correlation coefficient***

Sumber:

<https://www.mathsisfun.com/data/correlation.html>



# Correlation (2)

```
#korelasi antara perolehan medali emas dengan total perolehan  
#medali  
df3["gold"].corr(df3["total"])
```

```
df3["gold"].corr(df3["total"])  
0.965646961465112
```

- Dengan nilai mendekati 1, korelasi antara “gold” dan “total” adalah cenderung positif (*high positive correlation*)
- Artinya: semakin banyak perolehan medali emas, total perolehan medali juga cenderung semakin banyak



# Correlation (2)

## Latihan-5

- Hitung dan analisis-lah korelasi antara data-data berikut:
  - Gold vs Silver
  - Silver vs Bronze
  - Gold vs Bronze
  - Silver vs Total
  - Bronze vs Total

	gold	silver	bronze	total
gold	1			
silver	0.96218...	1		
bronze	0.81740...	0.89955...	1	
total	0.96564...	0.98981...	0.93523...	1



# Python For Data Science Cheat Sheet

## Pandas Basics

Learn Python for Data Science Interactively at [www.DataCamp.com](https://www.datacamp.com)



### Pandas

The Pandas library is built on NumPy and provides easy-to-use data structures and data analysis tools for the Python programming language.

**pandas**

Use the following import convention:

```
>>> import pandas as pd
```

### Pandas Data Structures

#### Series

A one-dimensional labeled array capable of holding any data type

a	3
b	-5
c	7
d	4

Index

```
>>> s = pd.Series([3, -5, 7, 4], index=['a', 'b', 'c', 'd'])
```

#### DataFrame

Columns

	Country	Capital	Population
0	Belgium	Brussels	11190846
1	India	New Delhi	1303171035
2	Brazil	Brasilia	207847528

Index

A two-dimensional labeled data structure with columns of potentially different types

```
>>> data = {'Country': ['Belgium', 'India', 'Brazil'],
            'Capital': ['Brussels', 'New Delhi', 'Brasilia'],
            'Population': [11190846, 1303171035, 207847528]}
```

```
>>> df = pd.DataFrame(data,
                      columns=['Country', 'Capital', 'Population'])
```

### I/O

#### Read and Write to CSV

```
>>> pd.read_csv('file.csv', header=None, nrows=5)
>>> df.to_csv('myDataFrame.csv')
```

#### Read and Write to Excel

```
>>> pd.read_excel('file.xlsx')
>>> pd.to_excel('dir/myDataFrame.xlsx', sheet_name='Sheet1')
```

Read multiple sheets from the same file

```
>>> xlsx = pd.ExcelFile('file.xls')
>>> df = pd.read_excel(xlsx, 'Sheet1')
```

### Asking For Help

```
>>> help(pd.Series.loc)
```

### Selection

Also see NumPy Arrays

#### Getting

```
>>> s['b']
-5
```

Get one element

```
>>> df[1:]
      Country      Capital  Population
1      India  New Delhi  1303171035
2      Brazil  Brasilia   207847528
```

Get subset of a DataFrame

#### Selecting, Boolean Indexing & Setting

##### By Position

```
>>> df.iloc[[0],[0]]
'Belgium'
>>> df.iat([0],[0])
'Belgium'
```

Select single value by row & column

##### By Label

```
>>> df.loc[[0], ['Country']]
'Belgium'
>>> df.at[[0], ['Country']]
'Belgium'
```

Select single value by row & column labels

##### By Label/Position

```
>>> df.ix[2]
Country      Brazil
Capital      Brasilia
Population   207847528
```

Select single row of subset of rows

```
>>> df.ix[:, 'Capital']
0      Brussels
1      New Delhi
2      Brasilia
```

Select a single column of subset of columns

```
>>> df.ix[1, 'Capital']
'New Delhi'
```

Select rows and columns

##### Boolean Indexing

```
>>> s[~(s > 1)]
>>> s[(s < -1) | (s > 2)]
>>> df[df['Population'] > 1200000000]
```

Series *s* where value is not >1  
*s* where value is <-1 or >2  
Use filter to adjust DataFrame

##### Setting

```
>>> s['a'] = 6
```

Set index *a* of Series *s* to 6

### Dropping

```
>>> s.drop(['a', 'c'])
>>> df.drop('Country', axis=1)
```

Drop values from rows (axis=0)  
Drop values from columns (axis=1)

### Sort & Rank

```
>>> df.sort_index()
>>> df.sort_values(by='Country')
>>> df.rank()
```

Sort by labels along an axis  
Sort by the values along an axis  
Assign ranks to entries

### Retrieving Series/DataFrame Information

#### Basic Information

```
>>> df.shape
>>> df.index
>>> df.columns
>>> df.info()
>>> df.count()
```

(rows, columns)  
Describe index  
Describe DataFrame columns  
Info on DataFrame  
Number of non-NA values

#### Summary

```
>>> df.sum()
>>> df.cumsum()
>>> df.min()/df.max()
>>> df.idxmin()/df.idxmax()
>>> df.describe()
>>> df.mean()
>>> df.median()
```

Sum of values  
Cumulative sum of values  
Minimum/maximum values  
Minimum/Maximum index value  
Summary statistics  
Mean of values  
Median of values

### Applying Functions

```
>>> f = lambda x: x*2
>>> df.apply(f)
>>> df.applymap(f)
```

Apply function  
Apply function element-wise

### Data Alignment

#### Internal Data Alignment

NA values are introduced in the indices that don't overlap:

```
>>> s3 = pd.Series([7, -2, 3], index=['a', 'c', 'd'])
>>> s + s3
a      10.0
b       NaN
c       5.0
d       7.0
```

#### Arithmetic Operations with Fill Methods

You can also do the internal data alignment yourself with the help of the fill methods:

```
>>> s.add(s3, fill_value=0)
a      10.0
b      -5.0
c       5.0
d       7.0
>>> s.sub(s3, fill_value=2)
>>> s.div(s3, fill_value=4)
>>> s.mul(s3, fill_value=3)
```

