



LEARNING PROGRESS REVIEW WEEK 5

By
OMICRON

OMICRON

Members of Group 3:

Anugrah Yazid Ghani
Edo Mohammad Hadad Gibran
Fajar Achmad
Muhammad Fikri Fadila

OUTLINE

- BASIC PYTHON PROGRAMMING (FUNCTION)
- INTRODUCTION TO NUMPY
- INTRODUCTION AND BASIC DATAFRAME (PANDAS)

BASIC PYTHON PROGRAMMING (FUNCTION)

OUTLINE:

- Functional Programming
- Component of Function
- Input
- Proses
- Output
- Nested Function
- Docstring
- Keuntungan Fungsi
- Global VS Local Variabel



FUNCTIONAL PROGRAMMING

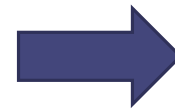
- *Functional programming* adalah sebuah program yang didalamnya secara umum terdiri dari fungsi fungsi untuk memproses data di seluruh eksekusinya.
- Secara singkat, fungsi dapat diibaratkan seperti pabrik yang memiliki input, urutan proses, dan output.



Input



Proses



Output

COMPONENT OF FUNCTION

- Untuk mendefinisikan sebuah fungsi, dapat dimulai dengan : **def**, **function's name**, **parentheses “()”**, **colon “:”**.

```
def component_of_function(succeed):  
    real = succeed  
    print(real)
```

- Setiap proses dalam fungsi python harus menggunakan **tab** atau **spaces 4 kali**. Pilih salah satu. Karena python sensitif terhadap space.

INPUT

- Di dalam python, setiap input disebut dengan **parameter**.
- Kita dapat memasukkan tipe data apapun ke dalam parameter, seperti *string*, *integer*, *float*, *array*, *dictionary*, bahkan sebuah fungsi.
- Walaupun parameter python sangat fleksibel dengan kita bebas memasukkan tipe data apapun kedalamnya. Namun, pastikan ketika memasukkan suatu tipe data sebagai input maka kita harus mendefinsikannya sebagai tipe data yang sama untuk input tersebut. Misal ketika awalnya input parameter 'name' sebagai string, maka jangan menjadikan input parameter tersebut menjadi integer (kecuali integer ke float).
- Paramater adalah variabel didalam fungsi yang tidak dapat digunakan diluar fungsi kecuali kita meletakkannya sebagai output.

PROSES

- Proses didalam fungsi dapat dikatakan sebagai urutan langkah – langkah untuk mencapai sesuatu (output).
- Variabel lokal didalam fungsi hanya berlaku didalam fungsi.
- Kita dapat melakukan hampir semuanya didalam fungsi, seperti *looping*, *conditional statement*, atau bahkan memanggil fungsi lain.
- Proses apapun didalam *library* juga dapat bekerja didalam fungsi.
- Pastikan untuk memberikan **comment** di setiap proses untuk mengingatkan dirimu dan rekan kerjamu.

OUTPUT

- Output pada fungsi dapat berupa *message* atau *value*.
- Untuk menghasilkan message gunakan ***print***.
- Untuk menghasilkan *value* gunakan ***return***.
- Fungsi di python dapat menghasilkan beberapa *value* disaat bersamaan.
- Jika kita ingin mengabaikan *value* tertentu dari beberapa return, gunakan *underscore* (_).

NESTED FUNCTION

Fungsi didalam fungsi

```
def outer(...):  
    x = ...  
    def inner(...):  
        y = x ** 3  
    return ...
```

```
def divide2(x1, x2):  
    D1 = x1 / 2  
    D2 = x2 / 2  
  
    return (D1, D2)
```

DOCSTRING

- Digunakan untuk mendeskripsikan kegunaan suatu fungsi
- Berguna sebagai dokumentasi
- Diletakkan tepat setelah baris *after header*.

```
def component_of_function(succeed):  
    """print real"""  
    real = succeed  
    print(real)
```

KEUNTUNGAN FUNGSI

1. Pada saat kita hendak melakukan proses yang berulang, kita dapat menggunakan fungsi untuk menyederhanakan code.
2. Memisahkan setiap *workflow* kedalam fungsinya.
3. Memiliki fungsi dalam code akan memudahkan kita untuk fokus terhadap *problem solving*, ketimbang *debugging* pada proses yang sama berulang kali.
4. Setiap masalah atau error akan mudah dilacak melalui fungsi.
5. Relatif berguna untuk mengurangi penggunaan *memory*.

GLOBAL VS LOCAL VARIABLE

1. Dapat diakses diseluruh program pada semua fungsi/class.
2. Akan tetap berada di dalam memory kecuali kita menghapusnya atau menutup program python.

1. Hanya dapat diakses didalam fungsi dimana variabel tersebut dimunculkan.
2. Setelah fungsi berhasil menjalankan kerjanya, *local variable* akan otomatis terhapus.

STUDI KASUS

FUNCTION FOR n-th FIBONACCI NUMBER

```
# Function for nth Fibonacci number
def Fibonacci(n):

    # Check if input is 0 then it will
    # print incorrect input
    if n < 0:
        print("Incorrect input")

    # Check if n is 0
    # then it will return 0
    elif n == 0:
        return 0

    # Check if n is 1,2
    # it will return 1
    elif n == 1 or n == 2:
        return 1

    else:
        return Fibonacci(n-1) + Fibonacci(n-2)

# Driver Program
print(Fibonacci(8))
```


INTRODUCTION TO NUMPY

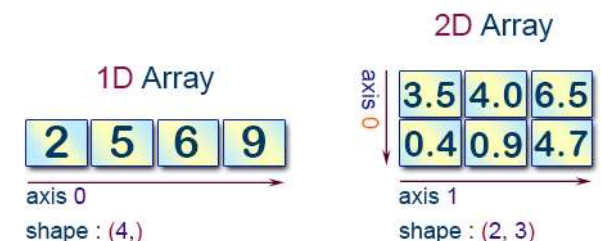
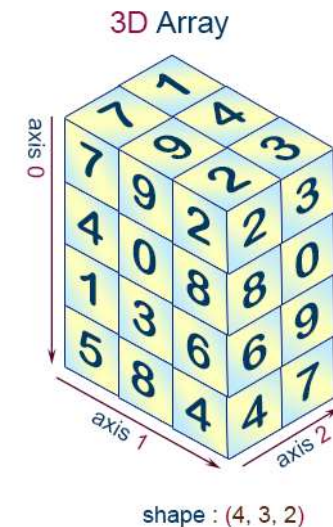
OUTLINE:

- What is Numpy?
- Why Use Numpy?
- Numpy Ability
- Creating a Numpy Array & Indexing an Array
- Array Reshaping & Joining Array
- Splitting Array & Sorting Array
- Filtering Array
- Matrix Operation



WHAT IS NUMPY?

- Numpy merupakan tools dasar yg digunakan dalam komputasi Python
- Tersedia objek array yg dapat menjalankan fungsi dengan kinerja yang tinggi
- Dapat digunakan untuk fungsi array
- Dapat bekerja dengan domain aljabar linier, fourier transform dan matriks.
- Numpy memiliki beberapa dimensi yaitu 1 dimensi, 2 dimensi bahkan hingga 4 dimensi.



WHY USE NUMPY?

- Banyak ilmuwan yang menggunakan Numpy
- Dapat menyediakan objek array hingga 50x lebih cepat dari list Python tradisional
- Array dapat digunakan dalam berbagai ekosistem seperti ilmiah domain, ilmu data, pembelajaran mesin, visualisasi data
- Numpy biasa digunakan dalam aspek kehidupan seperti image processing, geoscience, chemistry, dsb

Quantum Computing



QuTiP
PyQuil
Qiskit

Statistical Computing



Pandas
statsmodels
Xarray
Seaborn

Signal Processing



SciPy
PyWavelets
python-control

Image Processing



Scikit-image
OpenCV
Mahotas

Graphs and Networks



NetworkX
graph-tool
igraph
PyGSP

Astronomy Processes



AstroPy
SunPy
SpacePy

Cognitive Psychology



PsychoPy

Bioinformatics



BioPython
Scikit-Bio
PyEnsembl
ETE

Bayesian Inference



PyStan
PyMC3
ArviZ
emcee

Mathematical Analysis



SciPy
SymPy
cvxpy
FEniCS

Chemistry



Cantera
MDAnalysis
RDKit

Geoscience



Pangeo
Simpeg
ObsPy
Fatiando a Terra

Geographic Processing



Shapely
GeoPandas
Folium

Architecture & Engineering



COMPAS
City Energy Analyst
Sverchok

NUMPY ABILITY

PROOF OF NUMPY IS FASTER THAN LIST

```
list1 = [i for i in range(100000)]
list2 = [i for i in range(100000)]

start = time.time()

list_tot = list1 + list2

end = time.time()

print(f"Runtime of the program is {round(end - start,5)}")

Runtime of the program is 0.00234
```

```
np1 = np.array([i for i in range(100000)])
np2 = np.array([i for i in range(100000)])

start = time.time()

num_con = np.concatenate((np1, np2), axis=0)

end = time.time()

print(f"Runtime of the program is {round(end - start,5)}")

Runtime of the program is 0.00068
```

CREATING A NUMPY ARRAY

1D Array

1 2 3

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

2D Array

1	2	3
1	2	3
1	2	3

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

www.IndianAIProduction.com

3D Array

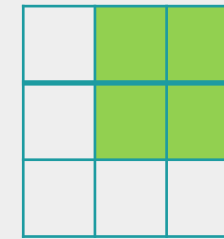
1 2 3

1 2 3

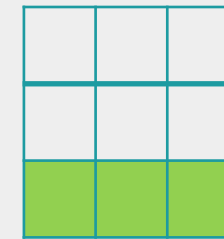
1 2 3

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

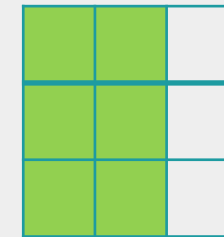
INDEXING AN ARRAY



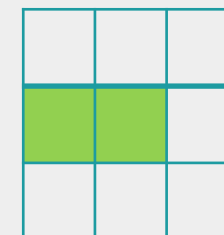
Expression	Shape
<code>arr[:2, 1:]</code>	<code>(2, 2)</code>



<code>arr[2]</code>	<code>(3,)</code>
<code>arr[2, :]</code>	<code>(3,)</code>
<code>arr[2:, :]</code>	<code>(1, 3)</code>



<code>arr[:, :2]</code>	<code>(3, 2)</code>
-------------------------	---------------------



<code>arr[1, :2]</code>	<code>(2,)</code>
<code>arr[1:2, :2]</code>	<code>(1, 2)</code>

CREATING ARRAY

```
[33] # Creating 1 Dimension Array
```

```
array1 = np.array([i for i in range(1,10)])
```

```
print("Dimension:", array1.ndim)  
array1
```

```
Dimension: 1  
array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```



1 DIMENSION

```
[56] # Creating 2 Dimension Array
```

```
array2 = np.array([[i for i in range(1, 10)], [i for i in range(1, 10)]])
```

```
print("Dimension:", array2.ndim)  
array2
```

```
Dimension: 2  
array([[1, 2, 3, 4, 5, 6, 7, 8, 9],  
       [1, 2, 3, 4, 5, 6, 7, 8, 9]])
```



2 DIMENSION

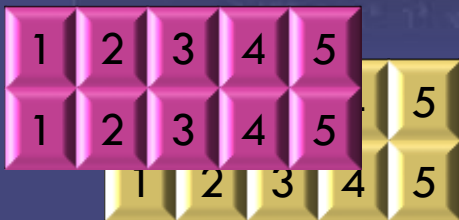
```
[62] # Creating 3 Dimension Array
```

```
array3 = np.array([[[i for i in range(1, 6)], [i for i in range(1, 6)]],  
                  [[i for i in range(1, 6)], [i for i in range(1, 6)]]])
```

```
print("Dimension:", array3.ndim)  
array3
```

```
Dimension: 3  
array([[[1, 2, 3, 4, 5],  
        [1, 2, 3, 4, 5]],
```

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



3 DIMENSION

CREATING ARRAY

1 DIMENSION ARRAY

```
[33] # Creating 1 Dimension Array

array1 = np.array([i for i in range(1,10)])

print("Dimension:", array1.ndim)
array1
```

Dimension: 1
array([1, 2, 3, 4, 5, 6, 7, 8, 9])



axis 0

Shape: (9,)

CREATING ARRAY

2 DIMENSION ARRAY

```
[56] # Creating 2 Dimension Array
```

```
array2 = np.array([[i for i in range(1, 10)], [i for i in range(1, 10)]])
```

```
print("Dimension:", array2.ndim)  
array2
```

```
Dimension: 2
```

```
array([[1, 2, 3, 4, 5, 6, 7, 8, 9],  
       [1, 2, 3, 4, 5, 6, 7, 8, 9]])
```



Shape: (2, 9)

CREATING ARRAY

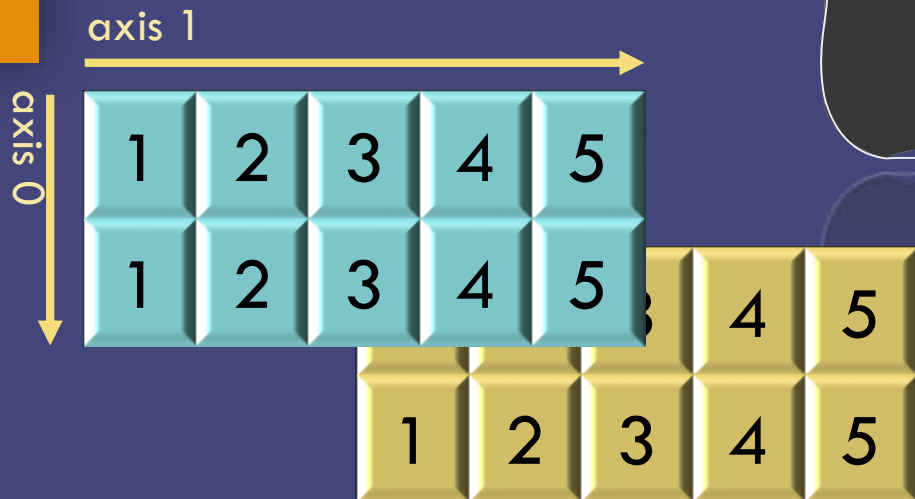
3 DIMENSION ARRAY

```
[62] # Creating 3 Dimension Array
```

```
array3 = np.array([[[i for i in range(1, 6)], [i for i in range(1, 6)]],  
                  [[i for i in range(1, 6)], [i for i in range(1, 6)]]])
```

```
print("Dimension:", array3.ndim)  
array3
```

```
Dimension: 3  
array([[[1, 2, 3, 4, 5],  
        [1, 2, 3, 4, 5]],  
       [[1, 2, 3, 4, 5],  
        [1, 2, 3, 4, 5]]])
```



Shape: (2, 5, 2)

INDEXING ARRAY

1	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18
19	20	21	22	23	24
25	26	27	28	29	30

[3] # Index the RED MARKS

```
print("",arra[:1, 1:], "\n", arra[1:, 1:2])
```

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

```
[[ 8]
```

```
[14]
```

```
[20]
```

```
[26]]
```

```
[3e]]
```

```
[3e]]
```

[4] # Index the YELLOW MARKS

```
arra[2:4, 3:5]
```

```
array([[16, 17],  
       [22, 23]])
```

ARRAY RESHAPING

Data

1
3
5
7
9
11

`data.reshape(2,3)`

1	3	5
7	9	11

`data.reshape(3,2)`

1	3
5	7
9	11

1	2
3	4
7	8

JOINING ARRAY

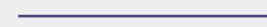
1	2	5	6
3	4	7	8

`np.concatenate((a,b),1)`



1	2	5	6
3	4	7	8

`np.concatenate(axis=0)`



1	2
3	4
7	8

RESHAPING ARRAY

Mengubah bentuk array dari bentuk 1 Dimensi: axis 0 = (10,) menjadi 2 Dimensi: axis 0, 1 = (2, 5)



```
[67] # Array Reshaping
```

```
Array_1 = np.array([i for i in range(10)])  
Array_1
```

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```



```
Array_1.reshape(2, 5)
```

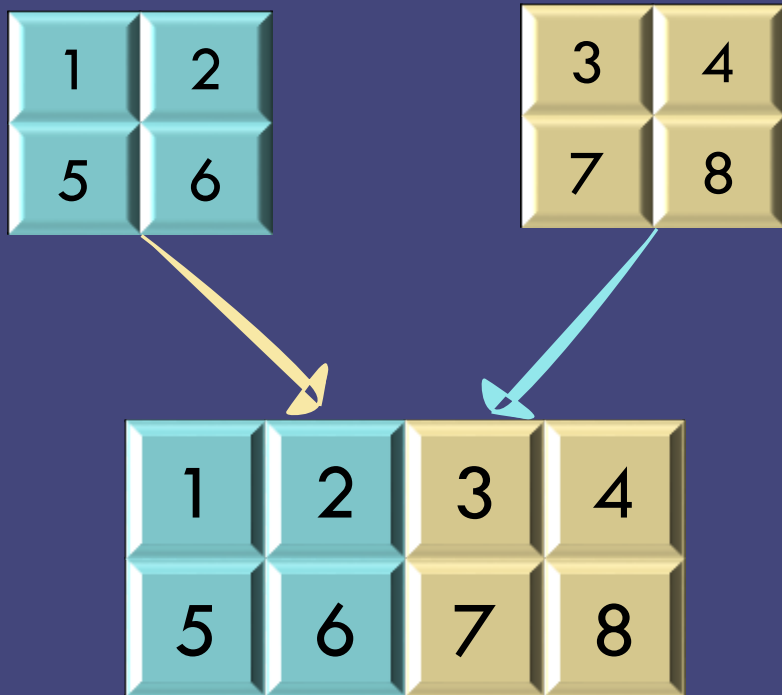
```
array([[0, 1, 2, 3, 4],  
       [5, 6, 7, 8, 9]])
```

5

2

JOINING ARRAY

Menggabungkan 2 array pada axis 1



```
[71] arr1 = np.array([[1, 2], [5, 6]])  
      arr2 = np.array([[3, 4], [7, 8]])  
  
      concat1 = np.concatenate((arr1, arr2), axis=1)  
      concat1  
  
      array([[1, 2, 3, 4],  
             [5, 6, 7, 8]])
```

SPLITTING ARRAY

1	2	5	6
3	4	7	8

np.split(2)



1	2	5	6
3	4	7	8

SORTING ARRAY

Original Array

5	2	11	7	4
---	---	----	---	---

Array after sorting

2	4	5	7	11
---	---	---	---	----

Filtering Array

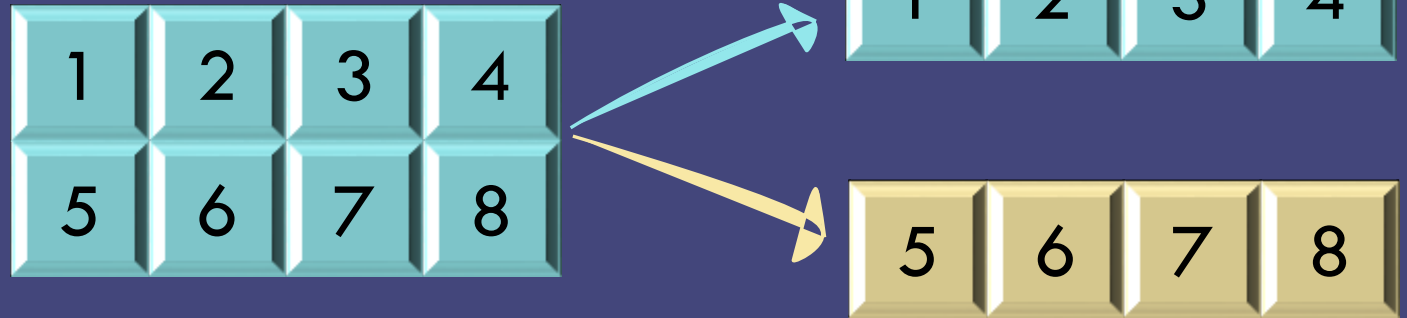
2	4	5	7	11
---	---	---	---	----

completely divisible by 2

2	4
---	---

SPLITTING ARRAY

Memisahkan array pada axis 0



```
[82] Array_11 = ([[1, 2, 3, 4], [5, 6, 7, 8]])
```

```
Array_11
```

```
[[1, 2, 3, 4], [5, 6, 7, 8]]
```

```
▶ np.array_split(Array_11, 4, axis=0 )
```

```
↳ [array([[1, 2, 3, 4]]),  
    array([[5, 6, 7, 8]]),
```

SORTING ARRAY

Menyusun urutan angka secara *descending* dari nilai terbesar

4	2	7	1
8	3	5	9



9	8	5	3
7	4	2	1

```
[89] array_2d = np.array([[4,2,7,1],[8,3,5,9]])  
array_2d
```

```
array([[4, 2, 7, 1],  
       [8, 3, 5, 9]])
```

```
[93] np.sort(array_2d)[::-1, ::-1]
```

```
array([[9, 8, 5, 3],  
       [7, 4, 2, 1]])
```

FILTERING ARRAY

Mengambil angka yg nilainya kurang dari 5

```
[89] array_2d = np.array([[4,2,7,1],[8,3,5,9]])  
array_2d
```

```
array([[4, 2, 7, 1],  
       [8, 3, 5, 9]])
```



```
array_2d[np.where(array_2d < 5)]
```

```
array([4, 2, 1, 3])
```

MATRIX OPERATION

* Can only be done if matrices are equal.

↳ means equal number of rows & columns.

Example:

$$\begin{bmatrix} -3 & 5 \\ 9 & -3 \end{bmatrix} + \begin{bmatrix} 6 & -1 \\ 0 & -5 \end{bmatrix} = \begin{bmatrix} 3 & 4 \\ 9 & -8 \end{bmatrix}$$

$$\begin{bmatrix} 2 & 5 \\ -4 & 3 \\ 0 & 8 \end{bmatrix} - \begin{bmatrix} -3 & 9 \\ 5 & 1 \\ -6 & 1 \end{bmatrix} = \begin{bmatrix} 5 & -4 \\ -9 & 2 \\ 6 & 7 \end{bmatrix}$$

You do these:

$$12a) \begin{bmatrix} -4 & 3 \\ 7 & -6 \end{bmatrix} + \begin{bmatrix} 6 & -3 \\ 2 & -4 \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 9 & -10 \end{bmatrix}$$

INTRODUCTION AND BASIC DATAFRAME (PANDAS)

OUTLINE:

- What is Pandas?
- Why Use Pandas?
- Basic Komponen Pandas
- Komponen Dataframes
- List, Tuple, Numpy Array, Pandas Series
- Making List with Pandas
- Making Tuple with Pandas
- Making Numpy Array with Pandas
- Making Pandas Series
- Reading and Writing with Pandas



WHAT IS PANDAS?

- Library Python untuk pekerjaan yang menggunakan **Data Sets (Kumpulan Data)**.
- **Fungsi:**
 - Analyzing Data
 - Cleaning Data
 - Exploring Data
 - Manipulating Data

Pandas



Nama "**Pandas**" berasal dari "Panel Data", dan "Python Data Analysis".
Dibuat oleh **Wes McKinney** pada tahun **2008**.

WHY USE PANDAS?

“Data yang relevan sangat penting dalam Data Science”

- **Menyajikan** hal-hal pengolahan data yang rumit menjadi sederhana
- **Mempercepat** proses penyajian data dan analisis data.
- Membuat kesimpulan berdasarkan **teori statistik**.
- **Membersihkan** kumpulan data yang berantakan.
- Membuatnya **data dapat dibaca dan relevan**.
- Dapat **membaca beberapa tipe file** (CSV, Excel, database SQL, dan HDFS).
- Dapat **melakukan pivot table** seperti di Excel



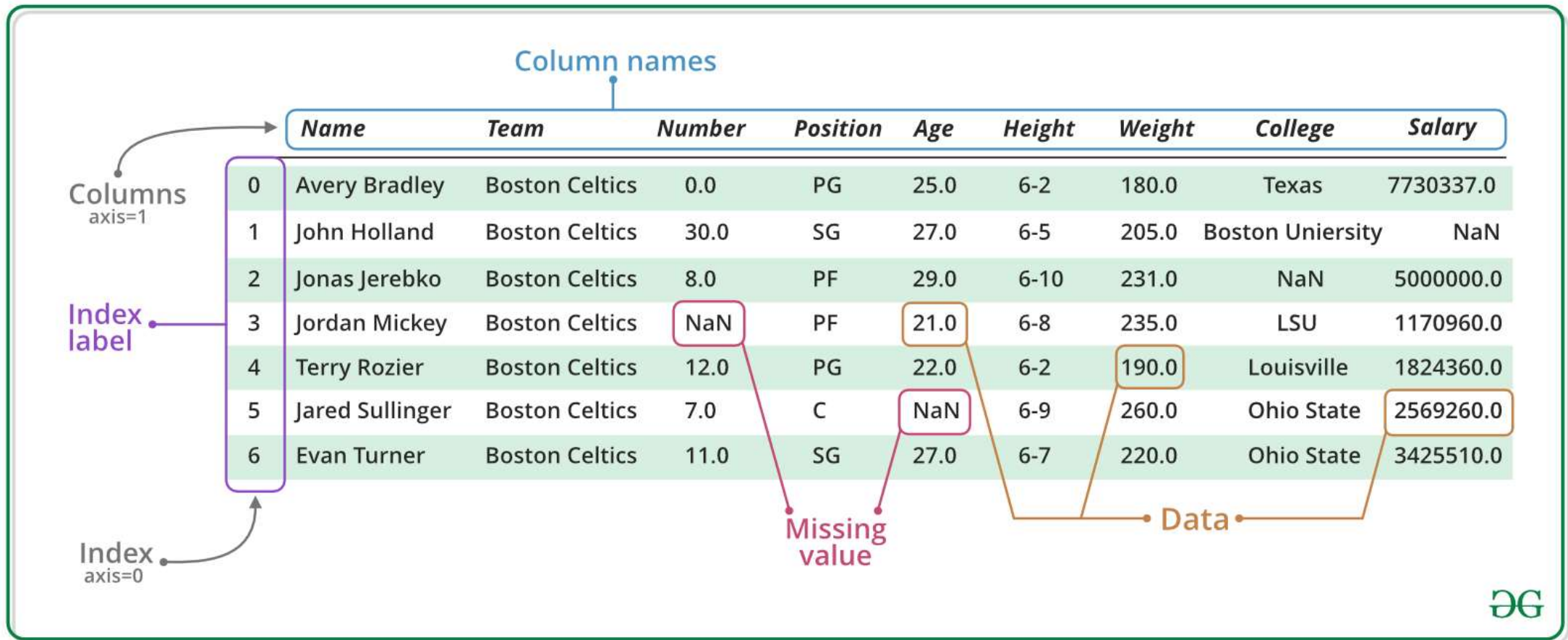
BASIC KOMPONEN PANDAS



- **SERIES** : Sebuah KOLOM
- **DATAFRAMES** : Tabel multi dimensi yang terdiri dari kumpulan kolom.

Series		Series		DataFrame		
apples		oranges				
0	3	0	0	0	3	0
1	2	1	3	1	2	3
2	0	2	7	2	0	7
3	1	3	2	3	1	2

KOMPONEN DATAFRAMES



The diagram illustrates the components of a pandas DataFrame using a table of Boston Celtics players. Annotations include:

- Column names:** Points to the header row of the table.
- Index label:** Points to the index column (axis=0).
- Missing value:** Points to a cell containing 'NaN' in the 'Number' column.
- Data:** Points to the body of the table (axis=1).

	Name	Team	Number	Position	Age	Height	Weight	College	Salary
0	Avery Bradley	Boston Celtics	0.0	PG	25.0	6-2	180.0	Texas	7730337.0
1	John Holland	Boston Celtics	30.0	SG	27.0	6-5	205.0	Boston Uniersity	NaN
2	Jonas Jerebko	Boston Celtics	8.0	PF	29.0	6-10	231.0	NaN	5000000.0
3	Jordan Mickey	Boston Celtics	NaN	PF	21.0	6-8	235.0	LSU	1170960.0
4	Terry Rozier	Boston Celtics	12.0	PG	22.0	6-2	190.0	Louisville	1824360.0
5	Jared Sullinger	Boston Celtics	7.0	C	NaN	6-9	260.0	Ohio State	2569260.0
6	Evan Turner	Boston Celtics	11.0	SG	27.0	6-7	220.0	Ohio State	3425510.0

LIST, TUPLE, NUMPY ARRAY, PANDAS SERIES

List [...]	Tuple (...)	Array np[...]	Series pd[...]
Native Python	Native Python	Numpy	Pandas
Mutable	Immutable	Mutable	Mutable
Anggota list DAPAT diubah dan diganti	Anggota tuple TIDAK DAPAT diubah dan diganti	Anggota np.array DAPAT diubah dan diganti	Anggota pd.series DAPAT diubah dan diganti
Indexed	Indexed	Indexed	Indexed
Memuat berbagai macam tipe data dalam 1 list	Memuat berbagai macam tipe data dalam 1 tuple	Hanya menyimpan tipe data yang sama dalam 1 array	Hanya menyimpan tipe data yang sama dalam 1 series

MAKING LIST WITH PANDAS

Membuat dataframe yang berisikan data anggota omicron berupa ID, Nama, Gender, Usia, Pendidikan, dan Profesi

```
import pandas as pd

ID = ["DS11omicron01", "DS11omicron02", "DS11omicron03", "DS11omicron04"]
Name = ["Anugrah", "Fajar", "Edo", "Fikri"]
Gender = ["Male", "Male", "Male", "Male"]
Age = [ 25, 24, 27, 28]
Graduated = ["Yes", "Yes", "Yes", "Yes"]
Profession = ["Engineer", "Engineer", "Finance", "Agriculture"]

Table = list(zip(ID, Name, Gender, Age, Graduated, Profession ))

pd.DataFrame(Table, columns=["ID", "Name", "Gender",
                             "Age", "Graduated", "Profession"])
```

index	ID	Name	Gender	Age	Graduated	Profession
0	DS11omicron01	Anugrah	Male	25	Yes	Engineer
1	DS11omicron02	Fajar	Male	24	Yes	Engineer
2	DS11omicron03	Edo	Male	27	Yes	Finance
3	DS11omicron04	Fikri	Male	28	Yes	Agriculture

MAKING TUPLE WITH PANDAS

Membuat dataframe yang berisikan data anggota omicron berupa ID, Nama, Gender, Usia, Pendidikan, dan Profesi

```
tdf1 = ("DS11omicron01", "Anugrah", "Male", 25, "Yes", "Engineer")
tdf2 = ("DS11omicron01", "Fajar", "Male", 24, "Yes", "Engineer")
tdf3 = ("DS11omicron01", "Edo", "Male", 27, "Yes", "Finance")
tdf4 = ("DS11omicron01", "Fikri", "Male", 28, "Yes", "Lawyer")

pd.DataFrame([tdf1, tdf2, tdf3, tdf4],
              columns = ["ID", "Name", "Gender",
                        "Age", "Graduated", "Profession"])
```

index	ID	Name	Gender	Age	Graduated	Profession
0	DS11omicron01	Anugrah	Male	25	Yes	Engineer
1	DS11omicron01	Fajar	Male	24	Yes	Engineer
2	DS11omicron01	Edo	Male	27	Yes	Finance
3	DS11omicron01	Fikri	Male	28	Yes	Lawyer

MAKING NUMPY ARRAY WITH PANDAS

Membuat dataframe yang berisikan data anggota omicron berupa ID, Nama, Gender, Usia, Pendidikan, dan Profesi

```
npdf = np.array([["DS11omicron01", "Anugrah", "Male", 25, "Yes", "Engineer"],
                 ["DS11omicron01", "Fajar", "Male", 24, "Yes", "Engineer"],
                 ["DS11omicron01", "Edo", "Male", 27, "Yes", "Finance"],
                 ["DS11omicron01", "Fikri", "Male", 28, "Yes", "Lawyer"]])

numpy_df = pd.DataFrame(npdf, columns = ["ID", "Name", "Gender",
                                         "Age", "Graduated", "Profession" ] )

numpy_df
```

index	ID	Name	Gender	Age	Graduated	Profession
0	DS11omicron01	Anugrah	Male	25	Yes	Engineer
1	DS11omicron01	Fajar	Male	24	Yes	Engineer
2	DS11omicron01	Edo	Male	27	Yes	Finance
3	DS11omicron01	Fikri	Male	28	Yes	Lawyer

MAKING PANDAS SERIES

Membuat dataframe yang berisikan data anggota omicron berupa ID, Nama, Gender, Usia, Pendidikan, dan Profesi

```
import pandas as pd

ID = pd.Series(["DS11omicron01", "DS11omicron02", "DS11omicron03", "DS11omicron04"])
Name = pd.Series(["Anugrah", "Fajar", "Edo", "Fikri"])
Gender = pd.Series(["Male", "Male", "Male", "Male"])
Age = pd.Series([ 25, 24, 27, 28])
Graduated = pd.Series(["Yes", "Yes", "Yes", "Yes"])
Profession = pd.Series(["Engineer", "Engineer", "Finance", "Lawyer"])

Table = list(zip(ID, Name, Gender, Age, Graduated, Profession ))

pd.DataFrame(Table, columns=["ID", "Name", "Gender",
                             "Age", "Graduated", "Profession"])
```

index	ID	Name	Gender	Age	Graduated	Profession
0	DS11omicron01	Anugrah	Male	25	Yes	Engineer
1	DS11omicron02	Fajar	Male	24	Yes	Engineer
2	DS11omicron03	Edo	Male	27	Yes	Finance
3	DS11omicron04	Fikri	Male	28	Yes	Lawyer

READING AND WRITING WITH PANDAS

READ	WRITE
<code>pd.read_csv ('filename.csv')</code>	<code>df.to_csv ('filename or path')</code>
<code>pd.read_excel ('filename.xlsx')</code>	<code>df.to_excel ('filename or path')</code>
<code>pd.read_json ('filename.json')</code>	<code>df.to_json ('filename or path')</code>
<code>pd.read_html ('filename.htm')</code>	<code>df.to_html ('filename or path')</code>
<code>pd.read_sql ('tablename')</code>	<code>df.to_sql ('DB Name')</code>

THANK YOU