

LEARNING PROGRESS REVIEW

WEEK 5

KELOMPOK 7
CITIZEN DATA SCIENTIST

Diaz Jubiary - Hermulia Hadie - Desi Sulistyowati - Farahul Jannah

Basic Programming IV

- Functional Programming
 - Membuat Fungsi
 - Komponen Fungsi
 - Docstring
- Variabel Global vs Lokal
 - Nested Function
 - Returning Function
 - Default Arguments
 - Flexible Arguments
 - Lambda Function
 - Function Map
- Errors and Exception
- Library

Introduction to Numpy

- Numpy
 - Creating a Numpy Array
 - Indexing an Array
 - Array Reshaping
 - Joining Array
 - Splitting Array
 - Sorting Array
 - Filtering Array
 - Matrix Operation

Daftar Isi

Key Presentation Point

Introduction to Pandas Series

- Apa itu Pandas?
- Mengapa Pandas Digunakan?
- Perbedaan List, Tuple, Numpy Array dan Pandas Series
- Dataframe
- Membuat Dataframe
- Membaca File

...

Basic Programming IV

Functional Programming

- Pemrograman Fungsional adalah paradigma pemrograman dengan perangkat lunak yang secara umum terdiri dari fungsi-fungsi yang memproses data selama eksekusinya
- Sederhananya, fungsi itu seperti sebuah pabrik yang memiliki input, urutan proses, dan output.



Membuat Fungsi

```
def square():          # <- Function Header
    new_value = 4 ** 2 # <- Function Body
    print(new_value)
square()

output : 16
```

Parameter Fungsi

```
def square():          # <- Function Header
    new_value = 4 ** 2 # <- Function Body
    print(new_value)
square()
```

output : 16

```
square(5)
```

output : 125

Mendefinisikan fungsi dimulai dengan def, nama, tanda kurung "()", titik dua ":". Setiap proses di dalam fungsi harus dalam tab atau 4 spasi karena Python *space sensitive*. Beberapa layanan seperti Google Cloud Platform lebih memilih 4 spasi untuk mendefinisikan fungsi.

Proses

Komponen Fungsi

Input

```
def wealth_estimator(money):
    if money >= 1000000:
        status = 'You are above 1%'

    elif money < 1000000 and money <= 100:
        status = 'You are in middle class'

    else:
        status = 'You are poor'

    return status
```

Output

Input

- Input disebut juga sebagai parameter.
- Parameter dapat menggunakan berbagai macam tipe data, misalnya string, integer, float, array, dictionary, bahkan function itu sendiri.
- Apabila suatu parameter telah didefinisikan memuat suatu tipe data maka input data selain tipe data yang telah didefinisikan. Misalnya, parameter "Nama" didefinisikan sebagai tipe data string maka jangan memasukkan tipe data integer.
- Parameter adalah variabel di dalam fungsi yang tidak dapat dipanggil dari luar fungsi, kecuali jika kita meletakkannya sebagai output.
- Sebuah fungsi juga dapat memiliki parameter *predefined* atau standar. Jika kita tidak memasukkan nilai apapun, ia akan memanggil nilai defaultnya. Posisi parameter yang telah ditentukan harus setelah yang kosong karena tidak boleh kosong

Proses

- Proses di dalam fungsi seperti urutan langkah-langkah untuk mencapai sesuatu (output).
- Variabel lokal di dalam fungsi hanya bisa tinggal di dalam fungsi dan akan dihapus setelahnya.
- Kita dapat melakukan hampir semua hal di dalam fungsi, seperti looping, if statement, atau bahkan memanggil fungsi lain.
- Setiap proses dari library manapun juga dapat bekerja di dalam fungsi.
- Pastikan untuk memberikan komentar di setiap proses untuk mengingatkan diri Anda atau rekan satu tim Anda di waktu mendatang.

Output

- Keluaran dari fungsi dapat berupa pesan atau nilai. Tergantung bagaimana kita mendefinisikannya.
- Untuk menghasilkan pesan, kita dapat menggunakan "print".
- Untuk menghasilkan nilai, kita dapat menggunakan "return".
- Fungsi Python dapat menghasilkan beberapa nilai sekaligus.
- Jika kita ingin mengabaikan nilai tertentu dari beberapa return, gunakan garis bawah (_).

Docstring

- Menjelaskan apa mengenai fungsi apa yang digunakan
- Berfungsi sebagai dokumentasi
- Ditempatkan di baris setelah header
- Menggunakan tanda """teks"""

```
def square(value):  
    """Return the square of a value"""  
    new_value = value ** 2  
    return new_value
```

Fungsi dengan Parameter Lebih Dari Satu

```
def raise_to_power(value1, value2):  
    """Raise value2 to the power of value2."""  
    new_value = value1 ** value2  
    return new_value
```

Kelebihan Menggunakan Fungsi

1. Ketika memiliki proses berulang, kita dapat menggunakan fungsi untuk menyederhanakan kode.
2. Memisahkan setiap alur kerja sesuai fungsinya (misalnya, membersihkan data, memanggil data dari csv, dll.)
3. Menggunakan fungsi dalam kode akan memudahkan kita untuk fokus pada pemecahan masalah, daripada men-debug proses yang sama berulang kali.
4. Setiap masalah atau kesalahan mudah dilacak melalui fungsi.
5. Relatif berguna untuk mengurangi penggunaan memori.

Scope

- Tidak semua objek dapat diakses di mana saja dalam skrip
- Cakupan: bagian di mana objek/nama dapat diakses
 - Global → didefinisikan di bagian utama skrip
 - Lokal → di dalam suatu fungsi
 - Built-in → nama dalam modul built-in yang telah ditentukan sebelumnya



```
def square(value):
    """Returns the square of a number."""
    new_val = value ** 2
    return new_val
square(3)
```

9

new_val

```
<hr />-----
NameError          Traceback (most recent call last)
<ipython-input-3-3cc6c6de5c5c> in <module>()
<hr />-> 1 new_value
NameError: name 'new_val' is not defined
```

Memunculkan error karena yang dipanggil adalah variabel lokal yang berada di dalam fungsi

```
new_val = 10

def square(value):
    """Returns the square of a number."""
    new_val = value ** 2
    return new_val
square(3)
```

9

new_val

10

Tidak memunculkan error karena yang dipanggil adalah variabel global yang berada di luar fungsi

```
new_val = 10

def square(value):
    """Returns the square of a number."""
    new_value2 = new_val ** 2
    return new_value2
square(3)
```

100

```
new_val = 20
square(3)
```

400

Membuat variabel baru dan mengembalikan parameter yang sama dengan nilai yang berbeda

```
new_val = 10

def square(value):
    """Returns the square of a number."""
    global new_val
    new_val = new_val ** 2
    return new_val
square(3)
```

100

new_val

100

Membuat atau mendeklarasikan variabel global dapat juga ditulis di dalam fungsi dengan menuliskan “global”

Nested Function

```
[9] def mod2plus5(x1, x2, x3):
    """Returns the remainder plus 5 of three values."""
    new_x1 = x1 % 2 + 5
    new_x2 = x2 % 2 + 5
    new_x3 = x3 % 2 + 5

    return(new_x1, new_x2, new_x3)

print(mod2plus5(1, 2, 3))
(6, 5, 6)
```

```
def mod2plus5(x1, x2, x3):
    """Returns the remainder plus 5 of three values."""

    def inner(x):
        """Returns the remainder plus 5 of a value."""
        return x % 2 + 5

    return (inner(x1), inner(x2), inner(x3))

print(mod2plus5(1, 2, 3))
(6, 5, 6)
```

Hasil multiple parameter di sebelah kiri bisa didapatkan menggunakan nested function (fungsi di dalam fungsi) seperti pada hasil di sebelah kanan

Returning Function

The screenshot shows a code editor window with a Python script. The script defines a function `luas_lingkaran` that calculates the area of a circle given its radius (`jari`). The function uses a default value for `phi` (pi) of 3.14. It also includes a docstring explaining its purpose. Below the function definition, a variable `hasil_luas_lingkaran1` is assigned the result of calling the function with a radius of 14. The output of this assignment is 615.44.

```
def luas_lingkaran(jari, phi=3.14):
    """Fungsi ini menghitung luas lingkaran"""
    return phi * jari * jari

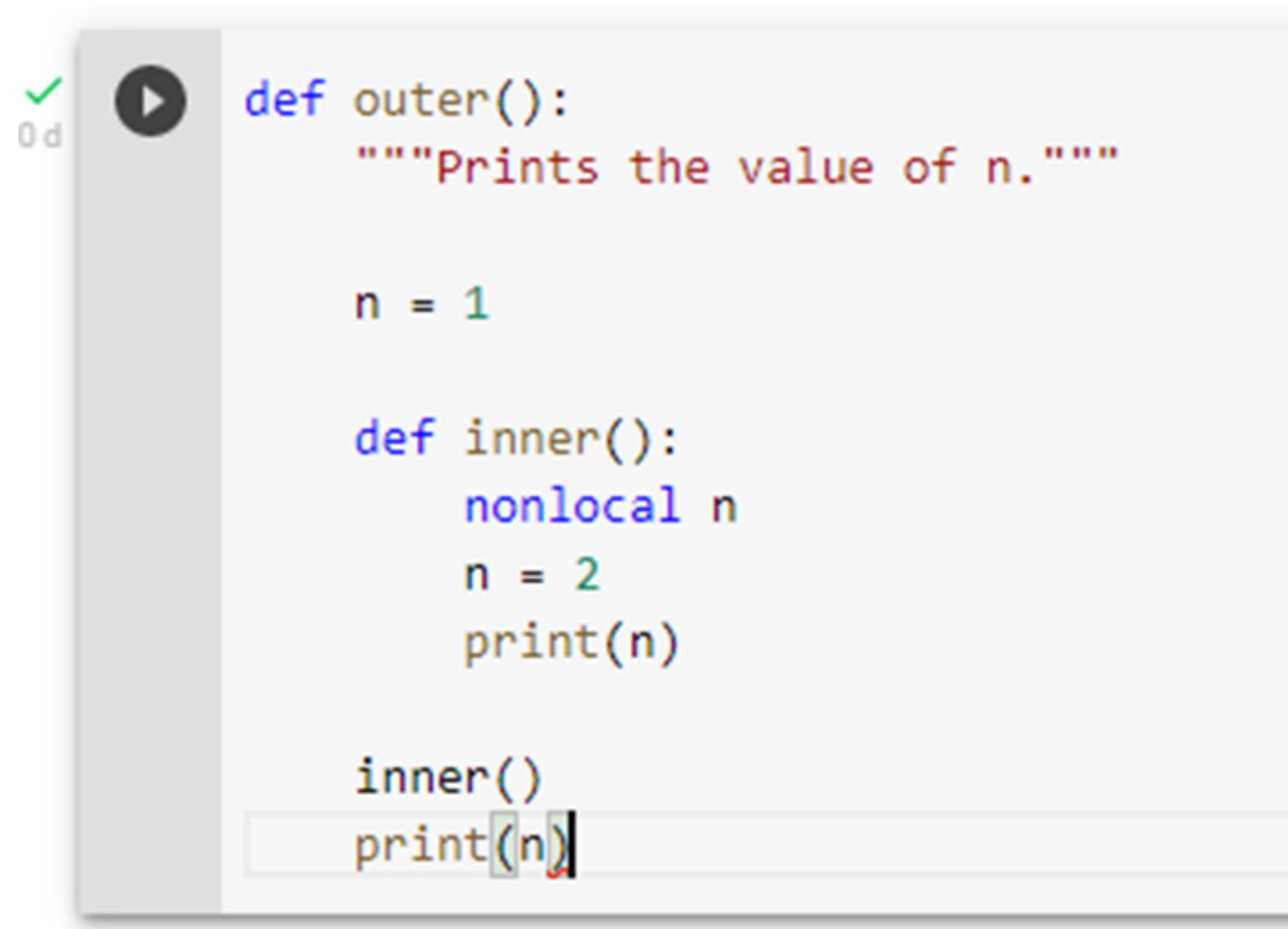
hasil_luas_lingkaran1 = luas_lingkaran(14)
hasil_luas_lingkaran1
```

615.44

Kita dapat memasukan hasil dari fungsi ke dalam suatu variabel sehingga bisa digunakan kembali untuk menghitung atau memproses lagi tanpa mengubah fungsi

9

Using Nonlocal



The screenshot shows a code editor window with the following Python code:

```
def outer():
    """Prints the value of n."""

    n = 1

    def inner():
        nonlocal n
        n = 2
        print(n)

    inner()
    print(n)
```

The code defines an outer function that prints the value of n. Inside outer, there is a nonlocal variable n set to 1. An inner function is defined, which also has a nonlocal variable n set to 2. Both print statements output 2.





Urutan Scopes Searched

⋮

LEGB rules:

- Local scope
- Enclosing functions (if any)
- Global
- Built-in

```
def power(number, pow=1):
    """Raise number to the power of pow."""
    new_value = number ** pow
    return new_value

power(9, 2)
```

```
81
```

```
power(9, 1)
```

```
9
```

```
power(9)
```

```
9
```

Default Argument

Default Argument adalah kita memasukan atau menentukan nilai dari parameter langsung di header fungsi, peletakannya di bagian akhir dari parameter, dalam contoh ini adalah “pow=1”

Flexible More Than One Type Arguments

```
✓ [39] def print_all(**kwargs):
        """print out key-value pairs in **kwargs."""
        # Print out the key-value pairs
        for key, value in kwargs.items():
            print(key + " : " + value)
```

```
✓ 0 d ➔ print_all(name="dumbledore", job="headmaster")
name : dumbledore
job : headmaster
```

Flexible Argument digunakan untuk fungsi dengan parameter yang lebih fleksibel, kita bisa menentukan dan memasukan parameter di luar header fungsi

3

Lambda Functions

```
raise_to_power = Lambda x, y: x ** y  
  
raise_to_power(2, 3)
```

8

Lambda function memungkinkan kita menuliskan fungsi dalam bentuk yang lebih ringkas

```
nums = [48, 6, 9, 21, 1]
```

```
square_all = map(lambda num: num ** 2, nums)
```

```
print(square_all)
```

```
<map object at 0x103e065c0>
```

```
print(list(square_all))
```

```
[2304, 36, 81, 441, 1]
```

Function Map

- Mengambil dua argumen (fungsi, seq)
- Menerapkan fungsi ke SEMUA elemen dalam urutan

```
def sqrt(x):
    """Returns the square root of a number."""
    try:
        return x ** 0.5
    except:
        print('x must be an int or float')

sqrt(4)
```

2.0

sqrt(10.0)

3.1622776601683795

sqrt('hi')

x must be an int or float

```
def sqrt(x):
    """Returns the square root of a number."""
    try:
        return x ** 0.5
    except TypeError:
        print('x must be an int or float')
```

Library

- Python adalah bahasa pemograman yang kuat pada bidang Data Science karena Python mempunyai libraries yang banyak. Libraries tersebut bisa kita import ke dalam projek kita. Kita juga bisa menggunakan beberapa libraries secara bersamaan.
- Library adalah kumpulan kode yang dibuat oleh seseorang, komunitas atau perusahaan.
- Beberapa library yang populer dalam bidang Data Science, yaitu Numpy, Pandas, SQL-Alchemy, Matplotlib, Seaborn, Scikit-Learn, Tensorflow, dll.
- Penggunaan library dapat membuat para Data Scientist lebih fokus terhadap pemecahan masalah.
- Kita dapat membuat library untuk diri kita sendiri atau menyumbangkannya untuk semua orang di seluruh dunia



Introduction to Numpy

Numpy

- Numpy adalah library Python yang fokus pada scientific computing.
- Berfungsi untuk membentuk objek N-dimensional array, seperti list pada Python.
- Konsumsi memory yang lebih kecil serta runtime yang lebih cepat dibanding list pada Python.
- Memudahkan penggerjaan Aljabar Linear, terutama operasi pada Vector (1-d array) dan Matrix (2-d array)
- Array dari Numpy dapat digunakan diberbagai ekosistem seperti data science, machine learning, dan data visualization.

Numpy

1D array

7	2	9	10
---	---	---	----

axis 0 →

shape: (4,)

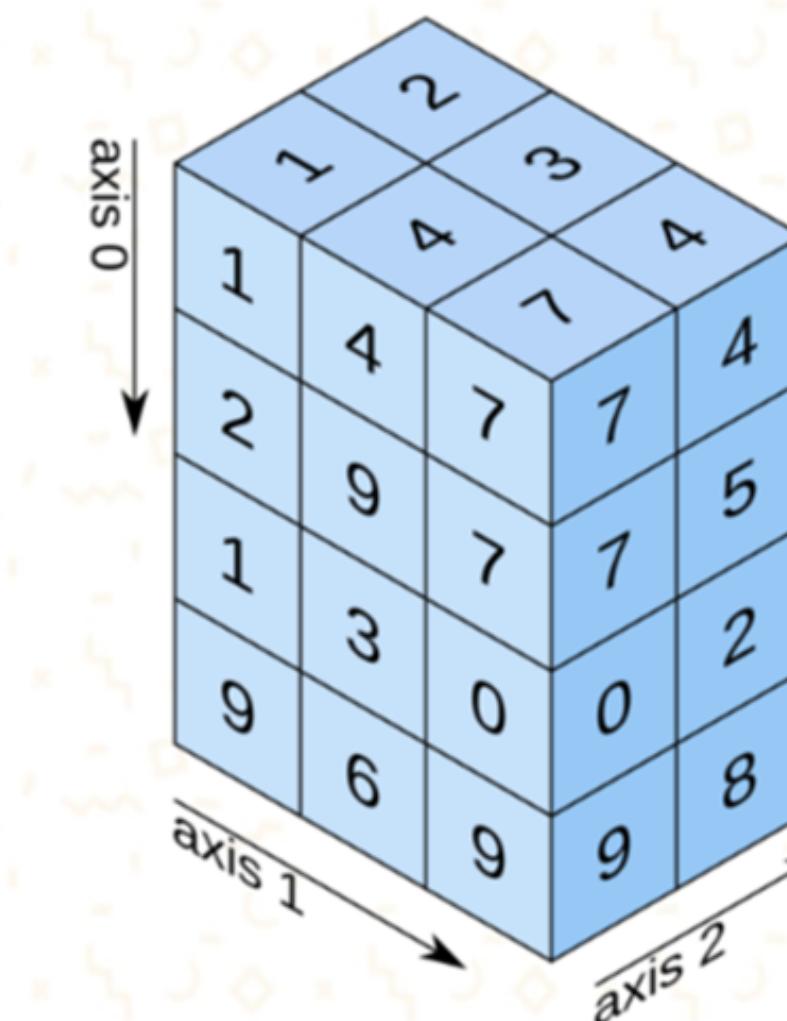
2D array

5.2	3.0	4.5
9.1	0.1	0.3

axis 0 ↓ axis 1 →

shape: (2, 3)

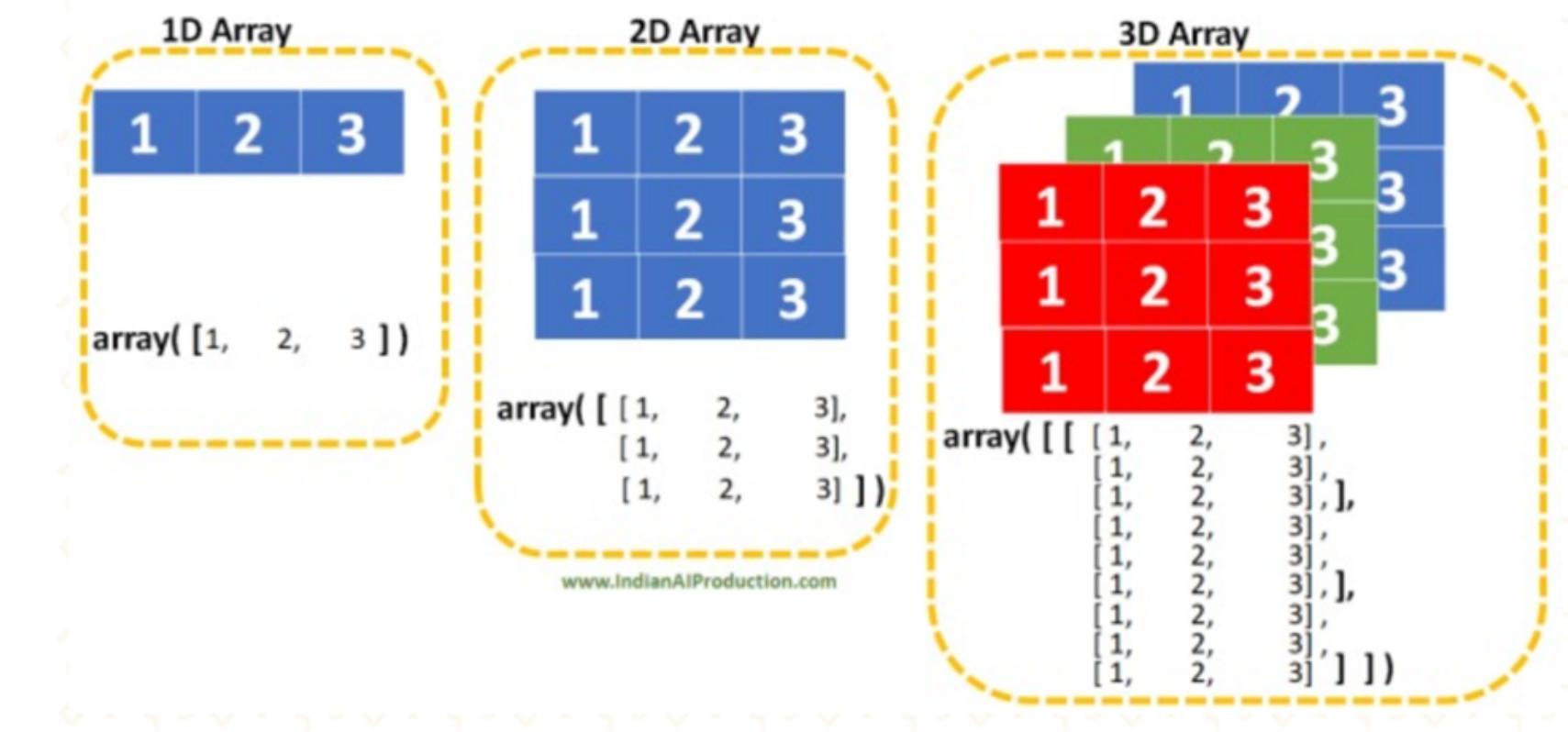
3D array



shape: (4, 3, 2)

Creating a Numpy Array

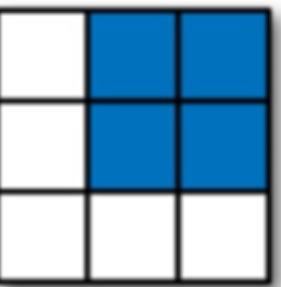
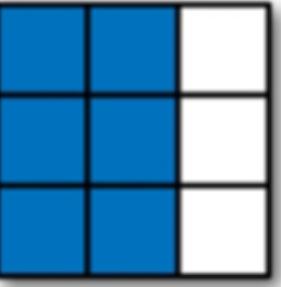
- Lakukan import terlebih dahulu library numpy as np
- Gunakan fungsi array() yang terdapat pada NumPy.
- Pada NumPy terdapat upcasting, yaitu ketika tipe data element array tidak sama, dilakukan penyamaan tipe data pada yang lebih tinggi.



```
In [23]: 1 b = np.array([1, 2, 3.0])
          2 b
```

```
Out[23]: array([1., 2., 3.])
```

Indexing an Array

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

- Indeks pada suatu array dimulai pada indeks ke-0.
- Untuk mengakses element pada array, kita menggunakan indeks sebagai alamat elemen pada array.
- Kita dapat melakukan assign nilai baru pada suatu element berdasarkan alamat indeks

```
In [32]: 1 # mengakses elemen pada indeks ke-0  
2 a = np.array([1, 2, 3])  
3 a[0]
```

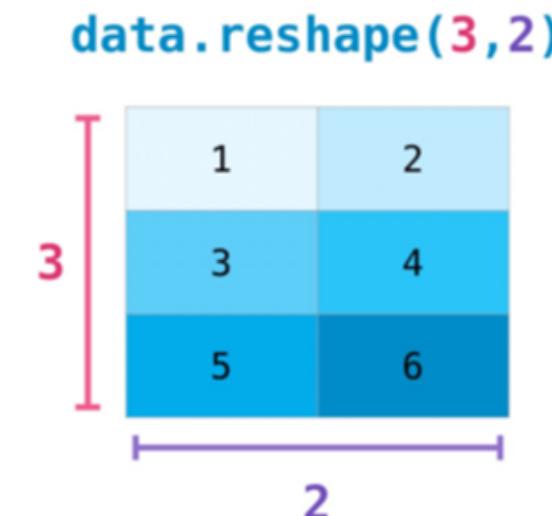
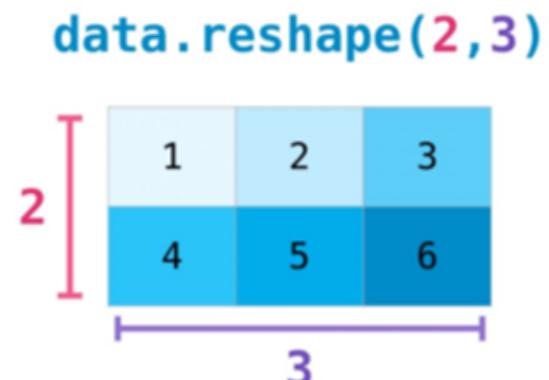
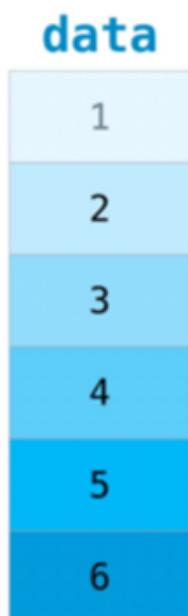
```
Out[32]: 1
```

```
In [33]: 1 # melakukan assign pada element array dengan element baru  
2 a[0] = 10  
3 a
```

```
Out[33]: array([10, 2, 3])
```

Array Reshaping

- Reshape array pada Python berarti mengubah shape array.
- Shape array adalah banyaknya elemen di setiap dimensi.
- Dengan reshape kita dapat menambah atau menghapus dimensi atau mengubah jumlah elemen di setiap dimensi.

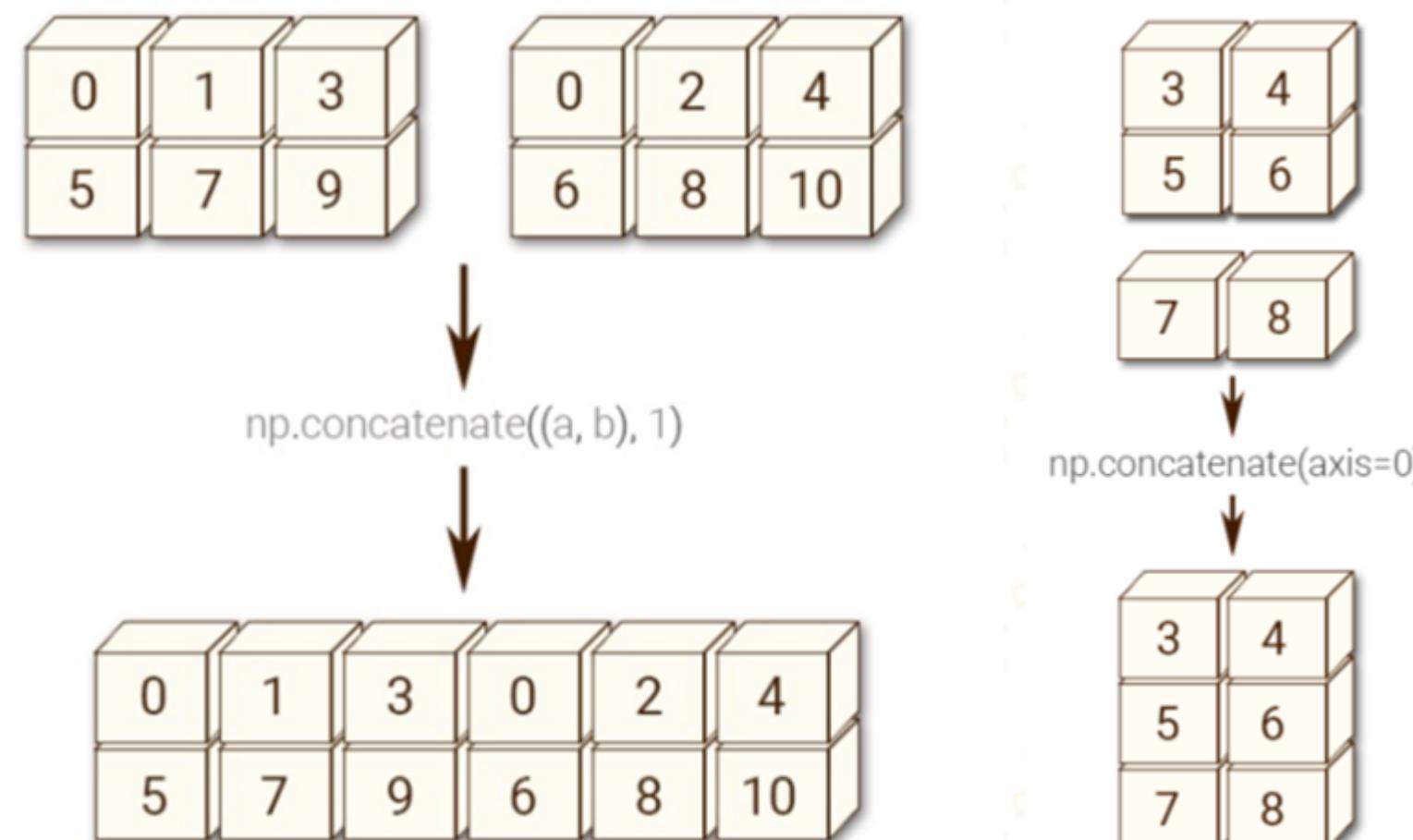


```
import numpy as np  
  
arr = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12])  
print(arr)  
  
print('\n')  
|  
newarr = arr.reshape(4, 3)  
  
print(newarr)  
  
[ 1  2  3  4  5  6  7  8  9 10 11 12]
```

```
[[ 1  2  3]  
 [ 4  5  6]  
 [ 7  8  9]  
 [10 11 12]]
```

Joining Array

- Join array merupakan proses penggabungan array yang satu dengan array lainnya.
- Menggabungkan merujuk kepada proses meletakkan dua atau lebih array dalam satu array.
- Proses join yang paling umum dalam numpy array adalah dengan menggunakan fungsi concatenate()

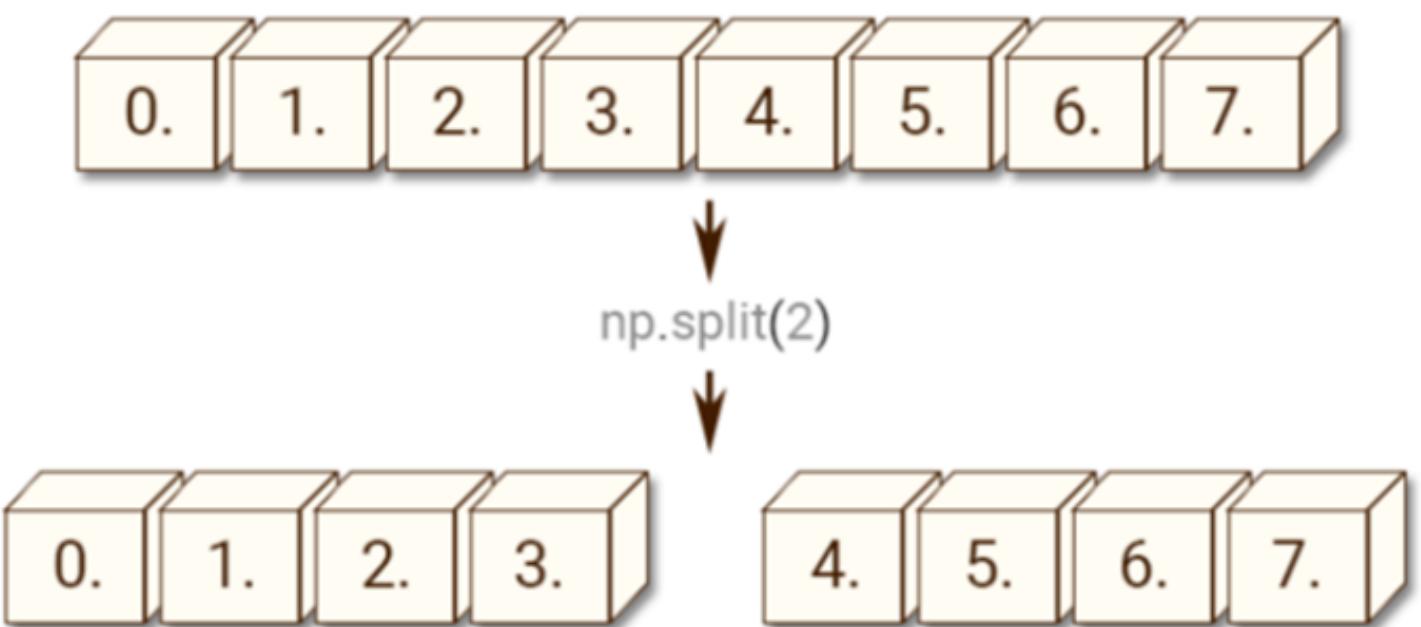


```
arr1 = np.array([1, 2, 3])  
  
arr2 = np.array([4, 5, 6])  
  
arr = np.concatenate((arr1, arr2))  
  
print(arr)
```

[1 2 3 4 5 6]

Splitting Array

- Split berarti memecah satu array menjadi beberapa bagian.
- Kita dapat menggunakan array_split() untuk memisahkan array, kita pass array yang ingin dipisahkan dan jumlah pemisahan.



```
import numpy as np  
  
arr = np.array([1, 2, 3, 4, 5, 6])  
  
newarr = np.array_split(arr, 3)  
  
print(newarr)  
  
[array([1, 2]), array([3, 4]), array([5, 6])]
```

:

Sorting Array

- Menyortir berarti meletakkan elemen dalam urutan yang teratur.
- Untuk mengurutkan array yang ditentukan, digunakan fungsi sort().

Original array:

5 2 8 7 1

Array after sorting:

1 2 5 7 8

```
import numpy as np  
  
arr = np.array([3, 2, 0, 1])  
  
print(np.sort(arr))  
  
[0 1 2 3]
```

Filtering Array

- Mengambil beberapa elemen dari array yang ada dan membuat array baru disebut pemfilteran.
- Di NumPy, kita dapat memfilter array menggunakan daftar indeks boolean.
- Daftar indeks boolean adalah daftar boolean yang sesuai dengan indeks dalam array.

Array →

1	2	3	4
---	---	---	---

completely divisible by 2

New_Array → [2, 4]

```
import numpy as np  
arr = np.array([1, 2, 3, 4, 5, 6, 7])  
  
filter_arr = arr % 2 == 0  
  
newarr = arr[filter_arr]  
  
print(filter_arr)  
  
print(newarr)
```

```
[False  True False  True False  True False]  
[2 4 6]
```

Matrix Operation

- Matriks dapat dikatakan sebagai list dua dimensi dimana suatu list berisi list lagi.
- Untuk merepresentasikan matriks, kita harus menyimpan list dengan panjang yang sama dalam suatu list.
- Bila list berbeda - beda panjangnya, maka list tersebut disebut sebagai sparse matrix.
- Dalam mengolah matriks, ada berbagai operasi yang dapat dilakukan. Mulai dari translasi, rotasi, mencari determinan, operasi baris elementer, dan lainnya

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \times \begin{bmatrix} 10 & 11 \\ 20 & 21 \\ 30 & 31 \end{bmatrix}$$
$$= \begin{bmatrix} 1 \times 10 + 2 \times 20 + 3 \times 30 & 1 \times 11 + 2 \times 21 + 3 \times 31 \\ 4 \times 10 + 5 \times 20 + 6 \times 30 & 4 \times 11 + 5 \times 21 + 6 \times 31 \end{bmatrix}$$
$$= \begin{bmatrix} 10+40+90 & 11+42+93 \\ 40+100+180 & 44+105+186 \end{bmatrix} = \begin{bmatrix} 140 & 146 \\ 320 & 335 \end{bmatrix}$$

Introduction to Pandas Dataframe

Apa itu Pandas?

- Pandas (Python for Data Analysis) adalah library Python yang fokus untuk proses analisis data seperti manipulasi data, persiapan data, dan pembersihan data.
- Pandas menyediakan struktur data dan fungsi high-level untuk membuat pekerjaan dengan data terstruktur/tabular lebih cepat, mudah, dan ekspresif.

Mengapa Pandas digunakan?

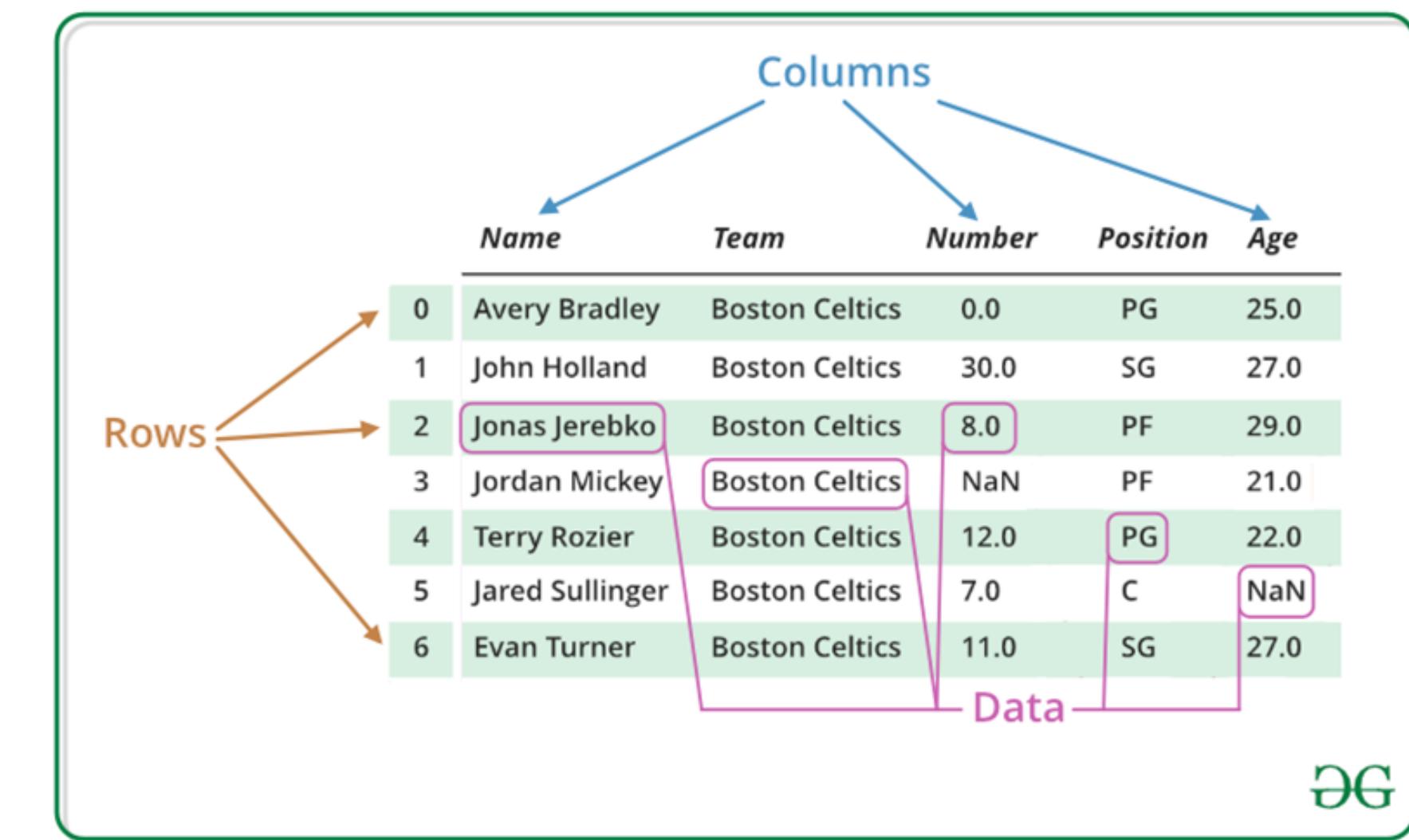
- Mudah untuk melakukan analisis data dengan memanipulasi data yang sesuai kebutuhan
- Dapat membaca beberapa tipe file seperti CSV, Excel, database SQL, dan HDFS file
- Dapat melakukan pivot table seperti pada excel
- Mudah untuk memfilter data yang diinginkan untuk analisa lebih dalam dan spesifik

Perbedaan List, Tuple, Numpy Array dan 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
Bisa memuat berbagai macam tipe data dalam satu list	Bisa memuat berbagai macam tipe data dalam satu tuple	Hanya meyimpan tipe data yang sama dalam satu array	Hanya meyimpan tipe data yang sama dalam satu series

- DataFrame adalah objek yang memiliki struktur data tabular, berorientasi pada kolom dengan label baris dan kolom
- Dataframe adalah struktur data yang berlabel 2 dimensi dengan kolom yang memiliki tipe data yang berbeda, seperti pada di Excel atau SQL
- Komponen Dataframe :
 - Kolom
 - Baris
 - Data

Dataframe



Membuat Dataframe Dari Python List

```
list1 = ["Joy", "Steward", "Nelly"]
list2 = [23, 28, 25]
list3 = ["Medan", "Jakarta", "Surabaya"]

dataframe_list = pd.DataFrame(list(zip(list1, list2, list3)),
                               columns=['Name', 'Age', 'Location'])

dataframe_list
```

	Name	Age	Location
0	Joy	23	Medan
1	Steward	28	Jakarta
2	Nelly	25	Surabaya

Membuat Dataframe Dari Python Tuple

```
tupple1 = ("Harry Potter",8)
tupple2 = ("Jack Bordon", 5)

dataframe_tupple = pd.DataFrame([tupple1, tupple2], columns=["Name", "Rating"])
dataframe_tupple
```

	Name	Rating
0	Harry Potter	8
1	Jack Bordon	5

Membuat Dataframe Dari Python Numpy Array

```
array = np.array([["Meetball",5,7.7], ["Fried Rice",3,8.0],[ "Pizza",10.0,9.0]])  
  
dataframe_array = pd.DataFrame(array, columns=["Food","Price (USD)","Rate"])  
dataframe_array
```

	Food	Price (USD)	Rate
0	Meetball	5	7.7
1	Fried Rice	3	8.0
2	Pizza	10.0	9.0

Membuat Dataframe Dari Python Pandas Series

```
series1 = pd.Series(["House A", "House B", "House C"])
series2 = pd.Series([100000, 250000, 300000])

series_dataframe = pd.DataFrame({"Type of House":series1, "Price":series2})
series_dataframe
```

	Type of House	Price
0	House A	100000
1	House B	250000
2	House C	300000

1. Upload file yang akan dibaca

```
from google.colab import files  
  
uploaded = files.upload()  
  
Choose Files employees.csv  
• employees.csv(application/vnd.ms-excel) - 8894 bytes, last modified: 1/12/2022 - 100% done  
Saving employees.csv to employees.csv
```

2. Membaca file menggunakan Pandas

```
import pandas as pd  
  
csv_file = pd.read_csv("employees.csv")  
csv_file.head()
```

Membaca File Dari Format csv

	employee_id	first_name	last_name	email
0	100	Steven	King	SKING
1	101	Neena	Kochhar	NKOCHHAR
2	102	Lex	De Haan	LDEHAAN
3	103	Alexander	Hunold	AHUNOLD
4	104	Bruce	Ernst	BERNST

1. Upload file yang akan dibaca

```
from google.colab import files  
  
uploaded = files.upload()
```

Choose Files

2. Membaca file menggunakan Pandas

```
excel_file = pd.read_excel("Startups Data.xlsx", sheet_name="Overview")  
excel_file.head()
```

Membaca File Dari Format Excel



Membaca File Dari Format Excel

Output

	ID	Name	Industry	Description	Year Founded	Employees	State	City	Metro Area
0	1	Over-Hex	Software	Provides a Web-based CRM tool that allows hosp...	2006	25	TN	Franklin	Nashville
1	2	Unimattax	IT Services	Helps law firms use Thomson Reuters Elite prac...	2009	36	PA	Newtown Square	Philadelphia
2	3	Lexila	Real Estate	Offers investment, construction, residential, ...	2013	38	IL	Tinley Park	Chicago
3	4	Greenfax	Retail	A Verizon Wireless premium retailer that offer...	2012	320	SC	Greenville	Newberry, SC
4	5	Saoace	Energy	An energy efficiency consulting firm that work...	2009	24	WI	New Holstein	Appleton, WI

• • •



Thank you!

Let us know if you have questions or clarifications.

Kelompok 7
Citizen Data Scientist