DigitalSkola

LEARNING PROGRESS REVIEW

Week 5

Entropy Team

OUR TEAM

Entropy Team

Adhang Muntaha Muhammad
https://www.linkedin.com/in/adhangmuntaha/

Aziz Fauzi
https://www.linkedin.com/in/aziz-fauzi-a6904711b/

Iwan Wahyu
https://www.linkedin.com/in/iwan-wahyu-setyawan-506809183

Marcellina Alvita F
https://www.linkedin.com/in/marcellina-alvita-faustina-63a284226

Ramadhan Luthfan
https://www.linkedin.com/in/luthfan-mahathir-91369b18b

DigitalSkola

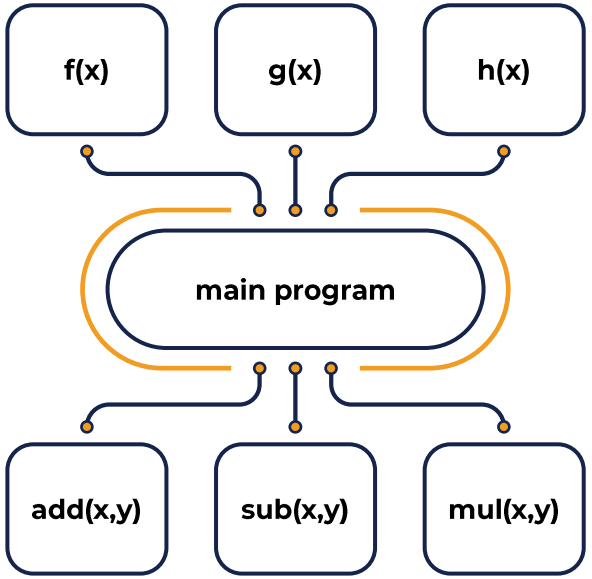
DAFTAR ISI

01

Basic Programming IV: Functions

Pemrograman Dasar IV: Fungsi

Fungsi



- Fungsi adalah **sekumpulan kode** program (*block of code*) yang dapat **berjalan ketika** fungsi tersebut **dipanggil**

- **Keuntungan** menggunakan fungsi:
  - ***Reuse* kode** yang sama
  - Lebih **mudah *troubleshooting*** suatu kode dalam fungsi
  - Lebih **mudah *update*** kode dalam fungsi tanpa memengaruhi kode lain

Pembuatan Fungsi

SYNTAX

```
def function_name():
  statements
```

CONTOH

```
# deklarasi fungsi
def hello_world():
  print('Hello World!')

# pemanggilan fungsi
hello_world()
```

- Pembuatan fungsi menggunakan ***keyword* "def"**
- Perhatikan penggunaan **tanda kurung**, **titik dua**, dan ***indentation***
- Untuk **memanggil fungsi**, dapat menuliskan nama fungsi yang diikuti simbol kurung biasa

Argument

SYNTAX

```
def function_name(expected_val):
  statements

function_name(passed_val)
```

CONTOH

```
def add(x, y):
  print(x + y)

add(97, 19) # 116
add(1.4, 10.25) # 11.65
```

- *Argument* adalah **nilai** yang dimasukkan ke dalam fungsi **ketika fungsi** tersebut **dipanggil**
- *Argument* dapat berjumlah 0 (nol) atau berapa pun

Argument dan Parameter

SYNTAX

```
# pembuatan fungsi
def function_name(parameter):
  statements

# pemanggilan fungsi
function_name(argument)
```

- Kedua istilah tersebut **pada dasarnya** adalah **sama**, yaitu nilai yang menjadi *input* dari suatu fungsi

- Letak yang membedakan:
  - ***Parameter*** adalah variabel yang berada dalam tanda kurung pada **pembuatan fungsi**
  - ***Argument*** adalah nilai yang digunakan ketika **pemanggilan fungsi**

Positional Argument

CONTOH

```
def print_age(name, age):
  if age < 21:
    print(name, 'masih anak-anak')
  else:
    print(name, 'sudah dewasa')

print_age('Entropy', 24)
# Entropy sudah dewasa
```

CONTOH SALAH

```
print_age(24, 'Entropy')
# error
```

- **Jumlah** dan **posisi *argument*** saat pemanggilan fungsi **harus sesuai** dengan *parameter* dalam fungsi

Default Argument

CONTOH

```
def print_power(val, pow=2):
    print(val ** pow)

print_power(2, 3)       # 8
print_power(2)          # 4
```

CONTOH SALAH

```
def print_power(pow=2, val):
    print(val ** pow)
```

- *Default argument* yaitu **nilai dasar**/ bawaan
- Jika pemanggilan fungsi **tidak menyertakan *argument***, maka nilai *default* tersebut yang digunakan
- *Default argument* **harus diletakkan di belakang** *non-default argument*

Arbitrary Argument

CONTOH

```
def print_sum(*args):
  total = 0
  for i in args:
    total += i
  print('total:', total)

print_sum(1,2,3)
# total: 6

print_sum(1,2,3,4,5)
# total: 15
```

- **Jumlah *argument*** saat pemanggilan fungsi **tidak harus sesuai** dengan jumlah *parameter* dalam fungsi
- Ditandai dengan **simbol *asterisk*** di depan *parameter*

Keyword Argument

CONTOH

```
def print_age(name, age):
  if age < 21:
    print(name, 'masih anak-anak')
  else:
    print(name, 'sudah dewasa')

print_age(name='Entropy', age=24)
# Entropy sudah dewasa

print_age(age=24, name='Entropy')
# Entropy sudah dewasa
```

- *Argument* dituliskan dalam bentuk ***key = value***
- ***Key*** adalah ***parameter*** dalam fungsi
- ***Value*** yaitu **nilai** yang akan dimasukkan ke dalam fungsi
- Dengan *keyword argument*, **posisi** *argument* **tidak harus sesuai** dengan *parameter* dalam fungsi

DigitalSkola

Arbitrary Keyword Argument

CONTOH

```
def print_age(**val):
  if val['age'] < 21:
    print(val['name'],\
          'masih anak-anak')
  else:
    print(val['name'],'sudah dewasa')

print_age(name='Entropy', age=24)
# Entropy sudah dewasa

print_age(age=24, name='Entropy')
# Entropy sudah dewasa
```

- *Argument* dituliskan dalam bentuk ***key = value***
- **Jumlah *argument*** saat pemanggilan fungsi **tidak harus sesuai** dengan jumlah *parameter* dalam fungsi
- Ditandai dengan **2 simbol *asterisk*** di depan parameter
- Fungsi akan menerima ***argument* sebagai *dictionary***

DigitalSkola

Variable Scope – Local

CONTOH

```
def print_area(w, l):
  area = w * l
  print('calculated area:', area)

print_area(5, 4)
# calculated area: 20

print('calculated area:', area)
# error
```

- *Variable scope* yaitu **cakupan** dari **variabel**
- ***Local scope*** yaitu **variabel** yang dibuat **dalam fungsi**, dan hanya dapat digunakan dalam fungsi
- Jika *local variable* dipanggil di luar *scope*-nya, maka akan terjadi *error*

Variable Scope – Global

CONTOH

```
def print_area():
  global area
  area = w * l
  print('calculated area:', area)

w = 5
l = 4
print_area()
# calculated area: 20

print('calculated area:', area)
# calculated area: 20
```

- ***Global scope*** yaitu **variabel** yang dibuat dalam **kodingan utama**, dan dapat digunakan di mana saja
- ***Keyword* "global"** digunakan untuk mengubah *local variable* menjadi *global variable*

Return Value

CONTOH

```
def print_area(w, l):
  area = w * l
  return area

val_area = print_area(5, 4)
print('calculated area:', val_area)
```

- *Return value* merupakan **nilai kembalian** dari fungsi
- *Return value* dari sebuah fungsi dapat di-***assign* ke variabel** di luar fungsi
- Salah satu kegunaannya yaitu untuk **mendapatkan *local variable*** dalam fungsi tanpa harus menggunakan *keyword* “global”

Lambda Function

SYNTAX

```
# langsung diberi nilai argument
(lambda arg: expression)(arg_value)

# fungsi disimpan dengan nama
function_name = lambda arg: expression

# fungsi yang ekuivalen
def function_name(arg):
  return expression
```

CONTOH

```
(lambda x: x ** 2)(5) # 25

square = lambda x: x ** 2
square(5) # 25
```

- *Lambda function* adalah “fungsi kecil”, yaitu fungsi yang dapat menerima **beberapa *argument***, tapi hanya mempunyai **satu *expression***

02

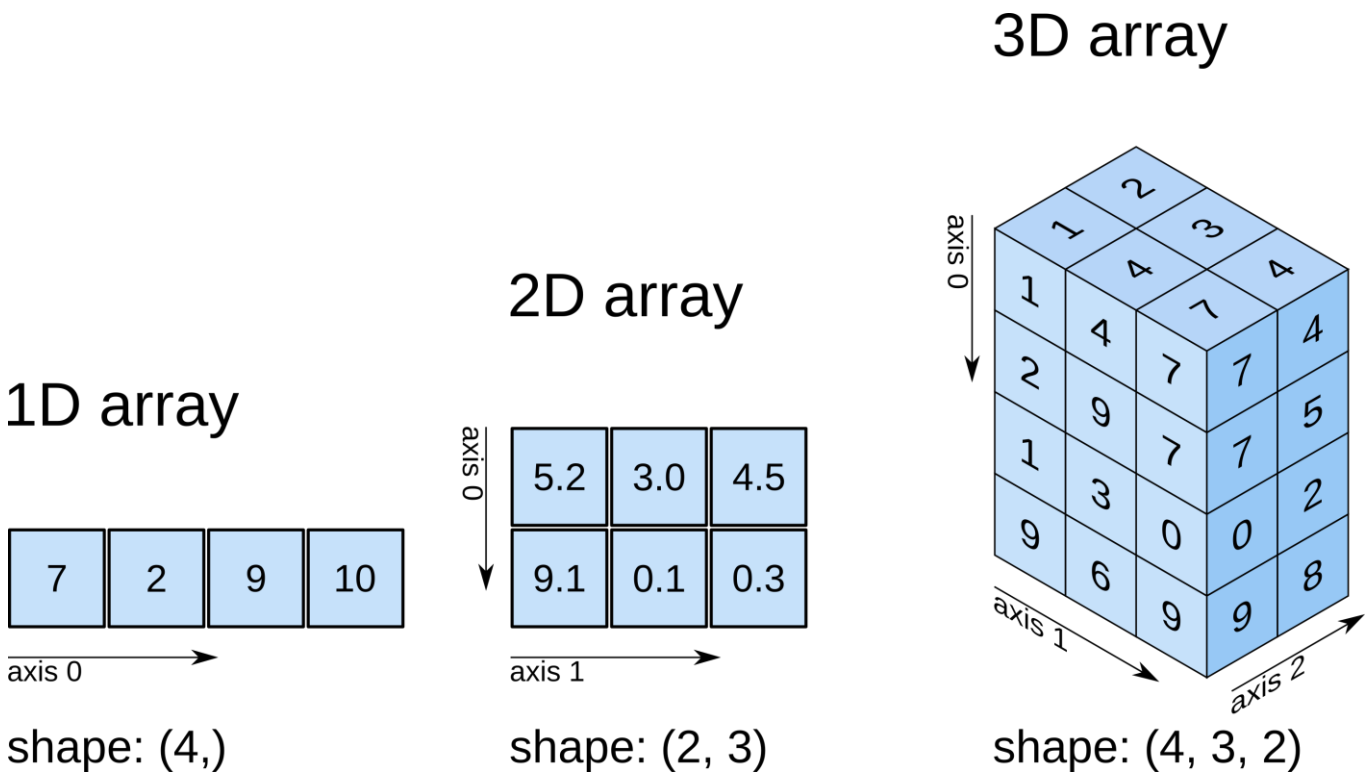
Introduction to Numpy

Pengenalan tentang Numpy

Numpy

- Numpy merupakan library Python yang berfungsi untuk **proses komputasi numerik**
- Numpy dapat digunakan untuk membuat objek **array berdimensi banyak** (*multi-dimensional array*)
- **Kelebihan** numpy array dibanding *built-in* list pada Python:
  - **Memori** lebih kecil
  - ***Runtime*** lebih cepat
  - **Banyak fungsi** untuk operasi aljabar dan matriks

Dimensi Array



- ***Axis* 0** : **baris**
- ***Axis* 1** : **kolom**
- ***Axis* 2** : **kedalaman**

Sumber:
https://github.com/elegant-scipy/elegant-scipy/blob/master/figures/NumPy_ndarrays_v2.png

DigitalSkola

Pembuatan Array – 1D

```
in  | import numpy as np

    | arr = np.array([1,2,3,4])
    | arr
out | array([1, 2, 3, 4])

in  | arr.ndim
out | 1

in  | arr.shape
out | (4,)
```

- arr.ndim: 1
  1 dimensi
- arr.shape: (4,)
  - 4 baris
  - Tidak ada kolom

DigitalSkola

Pembuatan Array – 2D

```
in  | import numpy as np

    | arr = np.array([[1,2,3],[4,5,6]])
    | arr
out | array([[1, 2, 3], [4, 5, 6]])

in  | arr.ndim
out | 2

in  | arr.shape
out | (2,3)
```

- arr.ndim: 2
  2 dimensi
- arr.shape: (2,3)
  - 2 baris
  - 3 kolom

DigitalSkola

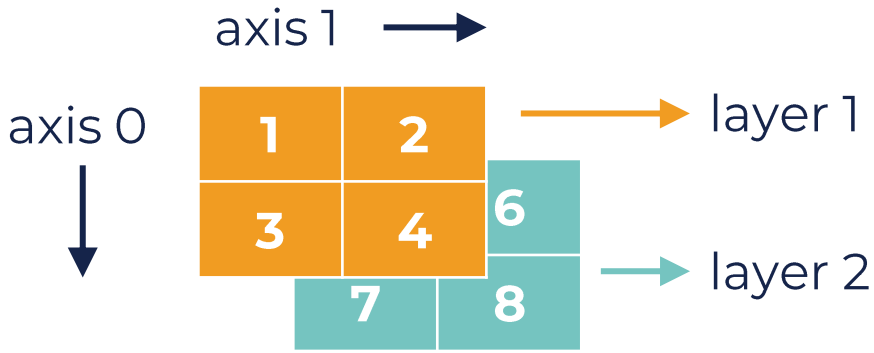
Pembuatan Array – 3D

```
in  | import numpy as np

    | arr = np.array([[[1,2],[3,4]],[[5,6],[7,8]]])
    | arr
out | array([[[1, 2], [3, 4]], [[5, 6], [7, 8]]])

in  | arr.ndim
out | 3

in  | arr.shape
out | (2,2,2)
```



- arr.ndim: 3
  3 dimensi

- arr.shape: (2,2,2)
  - 2 baris
  - 2 kolom
  - 2 kedalaman (layer)

DigitalSkola

Index

```
in  | import numpy as np

    | arr = np.array([1,2,3,4])
    | arr[0]
out | 1

in  | arr[-1]
out | 4

in  | arr[:2]
out | array([1, 2])
```

- Digunakan untuk **mengakses elemen** array

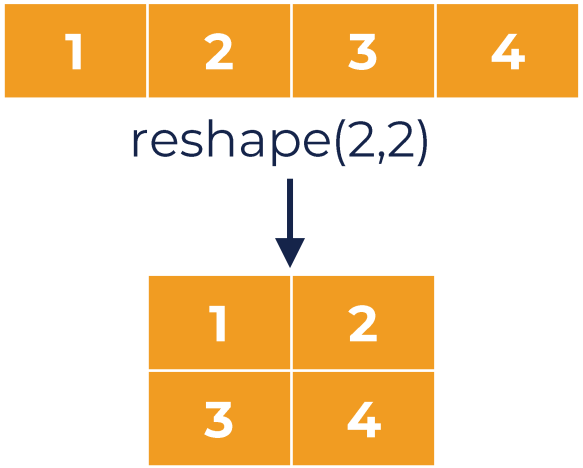
Reshape

```
in  | import numpy as np

    | arr = np.array([1,2,3,4,5,6])
    | arr
out | array([1, 2, 3, 4, 5, 6])

in  | arr.reshape(2,3)
out | array([[1, 2, 3], [4, 5, 6]])

in  | arr.reshape(3,-1)
out | array([[1, 2], [3, 4], [5, 6]])
```



- Digunakan untuk **mengubah bentuk** array
- Gunakan '-1' jika tidak mengetahui panjang dari salah satu *axis*

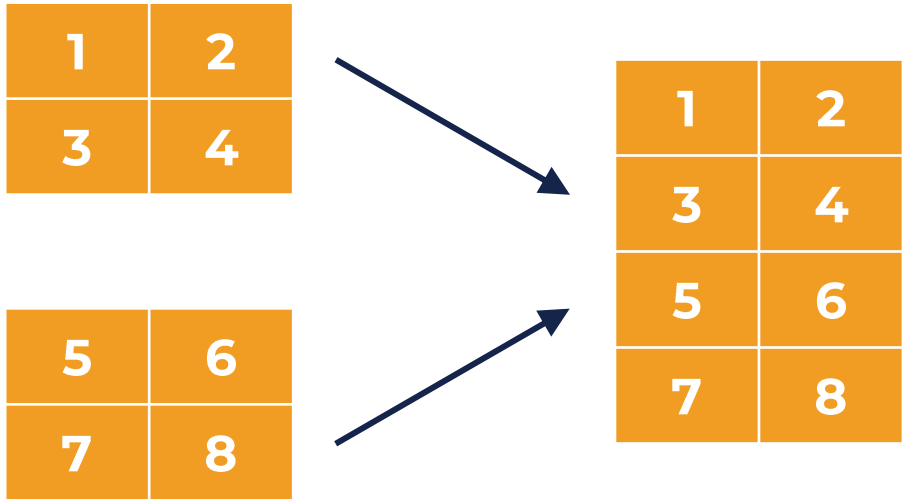
DigitalSkola

Concatenate – Axis 0

```
in  | import numpy as np

    | arr_1 = np.array([[1,2],[3,4]])
    | arr_2 = np.array([[5,6],[7,8]])
out |

in  | np.concatenate((arr_1, arr_2), axis=0)
out | array([[1, 2], [3, 4], [5, 6], [7, 8]])
```



- Digunakan untuk **menggabungkan** array **ke bawah** (axis 0 = baris)

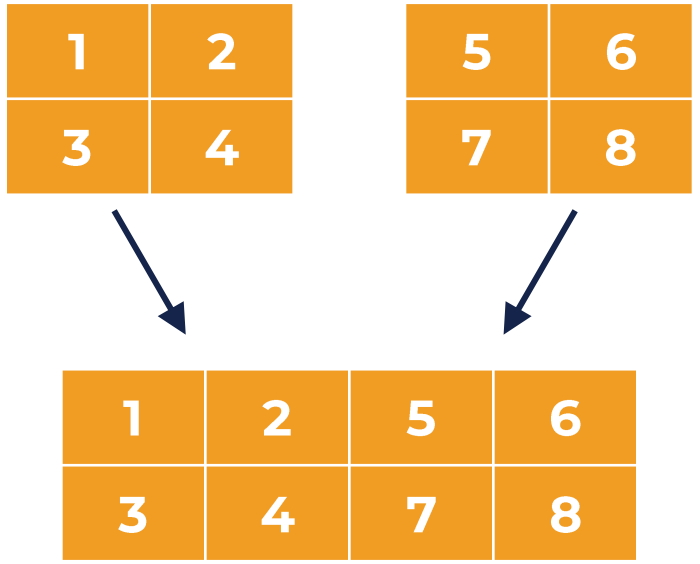
DigitalSkola

Concatenate – Axis 1

```
in  | import numpy as np

    | arr_1 = np.array([[1,2],[3,4]])
    | arr_2 = np.array([[5,6],[7,8]])
out |

in  | np.concatenate((arr_1, arr_2), axis=1)
out | array([[1, 2, 5, 6], [3, 4, 7, 8]])
```



- Digunakan untuk **menggabungkan** array **ke samping** (axis 1 = kolom)

DigitalSkola

Sort

```
in  | import numpy as np

    | arr_1 = np.array([4,2,5,1,3])
    | arr_1
out | array([4, 2, 5, 1, 3])

in  | np.sort(arr_1)
out | array([1, 2, 3, 4, 5])

in  | np.sort(arr_1, )[::-1]
out | array([5, 4, 3, 2, 1])
```

- Digunakan untuk **mengurutkan** array
- *Default* pengurutan adalah ***ascending***

DigitalSkola

Filter

```
in  | import numpy as np

    | arr_1 = np.array([4,2,5,1,3])
    | np.where(arr_1 > 3, True, False)
out | array([ True, False, True, False, False])

in  | arr_2 = arr_1[np.where(arr_1 > 3)]
    | arr_2
out | array([4, 5])

in  | arr_3 = arr_1[arr_1 > 3]
    | arr_3
out | array([4, 5])
```

- Digunakan untuk **mendapatkan elemen** dari array **berdasarkan** suatu **kondisi**
- *Filtering* dapat menggunakan np.where() maupun *masking*

Operasi Matriks

```
in  | import numpy as np

    | arr_1 = np.array([1,2,3])
    | arr_2 = np.array([4,5,6])

    | np.add(arr_1, arr_2)
out | array([5, 7, 9])


in  | arr_1 = np.array([[1,2,3],[4,5,6]])
    | arr_2 = np.array([[7,8],[9,10],[11,12]])

    | np.dot(arr_1, arr_2)
out | array([[ 58, 64], [139, 154]])
```

- Pada Numpy terdapat beberapa fungsi untuk melakukan operasi matriks, seperti:
  - add()
  - subtract()
  - divide()
  - multiply()
  - dot()

03

Introduction to Pandas and Basic Dataframe

Pengenalan tentang
Pandas dan dataframe

Pandas

- Pandas merupakan library Python yang berfungsi untuk **pengolahan, manipulasi,** dan **analisis data *tabular***
- Pandas dapat digunakan untuk **membaca dataset** dan menyajikannya dalam bentuk **dataframe**

Series & Dataframe

Series

	apel
0	3
1	2
2	1

Series

	jeruk
0	4
1	5
2	6

	apel	jeruk
0	3	4
1	2	5
2	1	6

Dataframe

- **Series** adalah struktur data 1 dimensi yang berisikan **data dari 1 kolom**
- **Dataframe** adalah struktur data 2 dimensi yang berisikan **data dari beberapa kolom**

DigitalSkola

Pembuatan Series

```
in  | import pandas as pd

    | series_1 = pd.Series([1,2,3,4])
    | series_1[0]
out | 1

in  | series_2 = pd.Series(['a','b','c','d'])
    | series_2[3]
out | d
```

- **Akses** data pada series dapat menggunakan **nomor indeks**

0	1
1	2
2	3
3	4

0	a
1	b
2	c
3	d

DigitalSkola

Pembuatan Dataframe – List

```
in  | import pandas as pd

    | name = ['Entropy', 'Team']
    | age = [25, 35]
    | city = ['Jakarta', 'Jogja']

    | list_zip = list(zip(name, age, city))
    | list_zip
out | [('Entropy', 25, 'Jakarta'),
    | ('Team', 35, 'Jogja')]

in  | col_name = ['Name','Age','City']
    | df = pd.DataFrame(list_zip, columns=col_name)
out |
```

- Setiap list merupakan **data** dari suatu **kolom**
- Fungsi zip() digunakan untuk **memasangkan** setiap **elemen** list yang berindeks sama

	name	age	city
0	Entropy	25	Jakarta
1	Team	35	Jogja

DigitalSkola

Pembuatan Dataframe – Tuple

```
in  | import pandas as pd

    | tuple_1 = ('Entropy', 25, 'Jakarta')
    | tuple_2 = ('Team', 35, 'Jogja')

    | list_tup = [tuple_1, tuple_2]
    | list_tup
out | [('Entropy', 25, 'Jakarta'),
    | ('Team', 35, 'Jogja')]

in  | col_name = ['Name','Age','City']
    | df = pd.DataFrame(list_tup, columns=col_name)
out |
```

- Setiap tuple merupakan **data** dari suatu **baris**

	name	age	city
0	Entropy	25	Jakarta
1	Team	35	Jogja

DigitalSkola

Pembuatan Dataframe – Array

```
in  | import pandas as pd
    | import numpy as np
    |
    | arr = np.array([['Entropy', 25, 'Jakarta'],
    |                 ['Team', 35, 'Jogja']])
    | arr
out | array([['Entropy', '25', 'Jakarta'],
    | ['Team', '35', 'Jogja']], dtype='<U7')

in  | col_name = ['Name','Age','City']
    | df = pd.DataFrame(arr, columns=col_name)
out |
```

- **Array 2 dimensi** dapat langsung diubah menjadi dataframe

	name	age	city
0	Entropy	25	Jakarta
1	Team	35	Jogja

DigitalSkola

Pembuatan Dataframe – Series

```
in  | import pandas as pd

    | name = pd.Series(['Entropy', 'Team'])
    | age = pd.Series([25, 35])
    | city = pd.Series(['Jakarta', 'Jogja'])
out |

in  | df = pd.DataFrame({'Name':name,
    |                    'Age':age,
    |                    'City':city})
out |
```

- **Nama kolom** dipasangkan dengan **series** menggunakan format dictionary, yaitu **nama_kolom : series**

	name	age	city
0	Entropy	25	Jakarta
1	Team	35	Jogja

DigitalSkola

Akses Dataframe

```
in   df['Age']       # kolom 'Age'
out  0       25
     1       35
     Name: Age, dtype: int64

in   df.loc[1]       # indeks ke-1
out  Name    Team
     Age     35
     City    Jogja
     Name: 1, dtype: object

in   df['Name'][0]   # kolom 'Name', indeks ke-0
out  Entropy
```

- **Kolom** dapat diakses menggunakan **nama kolom**
- **Baris** dapat diakses menggunakan **indeks** (nomor/ label)

	name	age	city
0	Entropy	25	Jakarta
1	Team	35	Jogja

DigitalSkola

Pembacaan Dataset

```
in  import pandas as pd

    data = pd.read_csv('winter_olimpic.csv')

    # melihat 5 baris teratas
    data.head()
```

- Pandas dapat digunakan untuk membaca beberapa format file, seperti **CSV**, **Ms Excel**, **JSON**, dll

	Year	Host_country	Host_city	Country_Name	Country_Code	Gold	Silver	Bronze
0	1924	France	Chamonix	United States	USA	1	2	1
1	1924	France	Chamonix	Great Britain	GBR	1	1	2
2	1924	France	Chamonix	Austria	AUT	2	1	0
3	1924	France	Chamonix	Norway	NOR	4	7	6
4	1924	France	Chamonix	Finland	FIN	4	4	3

THANKS

Entropy Team