# **Day 1 – Hello Python**

Python adalah salah satu bahasa pemrograman yang sedang naik daun. Kepopulerannya terus meroket sejak tahun 2010. Python menjadi bahasa pemrograman yang mudah untuk dipelajari, termasuk bagi orang-orang yang baru belajar bahasa pemrograman.

Kemudian apa yang membuat python begitu populer? Beberapa diantaranya adalah :

1. Python adalah general multipurpose programming. Artinya, dengan bahasa pemrograman python, kita hampir bisa membuat berbagai macam aplikasi, contohnya web, desktop, game, dan lain sebagainya.
2. Python mudah untuk dipelajari. Python didesain dengan penekanan agar mudah untuk dibaca. Kita bisa menuangkan konsep aplikasi kita kedalam bahasa python dengan baris yang relatif sedikit, sehingga kita bisa lebih cepat dan efektif dalam membangun suatu aplikasi.
3. Python open source. Artinya, python gratis dan setiap orang bisa berkontribusi untuk perkembangan python dan membagikannya ke orang lain.

Untuk memahami lebih lanjut, silahkan kunjungi external resource yang telah kami sediakan di link di bawah ini :

## External Source

List Artikel :

* <https://www.petanikode.com/python-pip/>
* <https://www.petanikode.com/python-sintaks/>
* <https://www.petanikode.com/python-variabel-dan-tipe-data/>
* <https://www.petanikode.com/python-operator/>

List Video :

* <https://www.youtube.com/watch?v=SsUF-7X3ehM&list=PLZS-MHyEIRo7cgStrKAMhgnOT66z2qKz1> ( Pengenalan Python )
* <https://www.youtube.com/watch?v=rqt-T1kl4PU&list=PLZS-MHyEIRo7cgStrKAMhgnOT66z2qKz1&index=3> ( Number dan Operasi Matematika )
* <https://www.youtube.com/watch?v=xlWN2fTyQFU&list=PLZS-MHyEIRo7cgStrKAMhgnOT66z2qKz1&index=4> ( Python String )

# **Day 2 – Python List**

List adalah salah satu tipe data untuk mengkoneksikan data di python. Contohnya, kita punya koleksi data suhu semua anggota keluarga kita, untuk menyimpan data ini, tentunya lebih mudah kalau kita menyimpannya dalam satu tempat, dan tempat itu salah satunya bisa bernama list. Mengapa salah satunya? Karena tempat menyimpan koleksi data di python tidak hanya list, ada tipe data lain seperti dictionary, tuple, set dan lain-lain. Namun dalam pembahasan ini kita akan fokus membahas tentang list.

Sampai di sini setidaknya kita sudah dapat sedikit gambaran tentang apa itu list. Selanjutnya kita akan coba pelajari bagaimana cara membuat list di python.. Berikut caranya..

# inisialisasi suhu anggota keluarga

suhu1 = 21

suhu2 = 20

suhu3 = 22

# simpan dalam list

suhu\_keluarga = [suhu1, suhu2, suhu3]

## Akses Data List

Setelah kita bisa membuat data sederhana dengan list. Sekarang kita akan belajar bagaimana caranya mengakses data dalam list. Untuk mengakses data dalam list, python menggunakan sesuatu yang bernama index. Index menunjukan posisi suatu data di dalam list, dan python memulai index dari 0. Perlu diketahui sebelumnya bahwa ada 2 teknik untuk mengakses data di dalam list. Pertama dengan subsetting list, kedua dengan slicing list. Mari kita lihat contohnya :

# Membuat data list

>>> tinggi\_badan = [

162, # index 0

177, # index 1

182, # index 2

150, # index 3

166 # index 4

]

# Subsetting list

>>> print(tinggi\_badan[0]) # posisi pertama

162

>>> print(tinggi\_badan[-2]) # posis kedua dari belakang

150

# Slicinglist

# Mengambil data dengan index 0, 1, 2, 3

>>> print(tinggi\_badan[:4])

[162, 177, 182, 150, 166]

# Mengambil data dengan index 2, 3, 4

>>> print(tinggi\_badan[2:5])

[182, 150, 166]

## Sifat List

Sebelumnya, kita sudah mengenal sedikit tentang list, mari kita kenal lebih dalam lagi tentang list.

1. List berisi koleksi nilai/data.
2. List bisa berisi tipe data apapun dan tidak harus semua data berisi tipe data yang sama.
3. List bisa berubah

Untuk sifat no 1. sudah dijelaskan dan tunjukan di atas. Sekarang kita langsung bahas sifat no.2. Sebelumnya, kita sudah mengenal tentang beberapa tipe data dalam list, ada numerik, string, boolean, dan sebagainya. Nah, nilai-nilai yang bertipe data apapun dapat dimasukan sebagai koleksi di dalam list, bahkan list itu sendiri bisa ada di dalam list. Untuk lebih jelasnya mari kita lihat contoh berikut..

# membuat list

suhu\_keluarga\_ucup = [

'ayah ucup', 19, 'ucup', 19, 'ibu ucup', 20

]

suhu\_keluarga\_boy = [

'istri boy', 20, 'anak boy', 18, 'istri kedua boy', 21

]

# membuat list dalam list dan di campur dengan data boolean

suhu\_keluarga = [

suhu\_keluarga\_ucup, suhu\_keluarga\_boy

]

## 

## 

## Manipulasi List

List adalah “mutable”, artinya dapat diubah.

Salah satu cara untuk mengubah daftar adalah dengan menetapkan indeks atau ekspresi irisan.

Misalnya, katakanlah kita ingin mengubah suhu ucup:

# mengubah suhu ucup

suhu\_keluarga\_ucup[3] = 22

print(suhu\_keluarga\_ucup)

>>> ['ayah ucup', 19, 'ucup', 22, 'ibu ucup', 20]

katakanlah kita ingin mengganti ‘ibu ucup’ dengan ‘mamah ucup’, beserta suhunya :

suhu\_keluarga\_ucup[-2:] = ['mamah ucup', 22]

print(suhu\_keluarga\_ucup)

>>> ['ayah ucup', 19, 'ucup', 22, 'mamah ucup', 22]

kita juga bisa menambahkan elemen di list, yaitu dengan menggunakan ‘+’ operator, katakanlah kita ingin menambahkan adik ucup beserta suhunya, berikut contohnya :

suhu\_keluarga\_ucup = suhu\_keluarga\_ucup + ['adik ucup', 20]

print(suhu\_keluarga\_ucup)

>> ['ayah ucup', 19, 'ucup', 22, 'mamah ucup', 22, 'adik ucup', 20]

terakhir, kita juga bisa menhilangkan element di dalam list, yaitu seperti ini :

del(suhu\_keluarga\_ucup[0])

Perhatian, setelah suatu elemen di dalam list dihapus, maka index dari seluruh elemennya pun akan berubah, contohnya di atas kita mendelete elemen dengan index 0 di list, artinya kita mendelete elemen ‘ayah ucup’, maka elemen yang lain akan berubah menyesuaikan dengan perubahan tersebut, maka ketika kita mengambil index 0 di list tersebut maka hasilnya akan seperti ini :

print(suhu\_keluarga\_ucup[0])

>>>19

## Some Function in List

Python memiliki beberapa fungsi bawaan yang berguna untuk bekerja dengan list. diantaranya adalah :

‘len’ memberikan panjang daftar:

keluarga\_ucup = ['mamah', 'papah', 'ucup', 'adek ucup', 'kaka ucup']

# berapa banyak anggota keluarga ucup

print(len(keluarga\_ucup))

>>>5

‘sorted’ mengurutkan elemen list:

>>>print(sorted(keluarga\_ucup))

>>>['adek ucup', 'kaka ucup', 'mamah', 'papah', 'ucup']

‘sum’, tentu fungsi ini untuk menjumlahkan:

primes = [2, 3, 5, 7]

print(sum(primes))

>>>17

## External Resource

List Artikel/Tutorial :

* <https://www.petanikode.com/python-list/>

List Video :

* <https://www.youtube.com/watch?v=kuLBqkpnKDk&list=PLZS-MHyEIRo7cgStrKAMhgnOT66z2qKz1&index=5> (Python List)
* <https://www.youtube.com/watch?v=NoX5wnjBaHo&list=PLZS-MHyEIRo7cgStrKAMhgnOT66z2qKz1&index=17> (Python List 2)

# **Day 3 – Functions, Method and Getting Help**

Sebelumnya kita telah melihat dan menggunakan fungsi-fungsi seperti print, sum, len dan sebagainya. Python memiliki lebih banyak fungsi, dan membuat fungsi kita sendiri adalah bagian besar dari perjalanan kita belajar pemrograman python.

Dalam bagian ini kita akan belajar lebih banyak tentang menggunakan dan mendefinisikan fungsi.

## Getting Help

Kita melihat fungsi ‘len’ di tutorial sebelumnya, tetapi bagaimana jika kita lupa fungsi apa itu?

Fungsi help () merupakan salah satu fungsi yang sangat penting untuk kita pelajari. Jika kita dapat mengingat cara menggunakan help(), kita memegang kunci untuk memahami sebagian besar fungsi lainnya. Karena fungsi help() adalah suatu fungsi untuk memahami fungsi semua fungsi.

Berikut ini sebuah contoh:

>>>print(help(len))

Help on built-in function len in module builtins:

len(obj, /)

Return the number of items in a container.

help () menampilkan dua hal:

1. header : header adalah bagian pertama dari output help, dari contoh diatas yaitu bagian len(obj, /). Baris ini memberitahu kita bahwa fungsi len ini menerima argument sebuah object.

2. Deskripsi singkat tentang apa fungsinya.

## 

## 

## Mendefinisikan fungsi

Python memiliki banyak sekali fungsi bawaan yang sangat membantu kita. Namun kita juga bisa membuat fungsi kita sendiri untuk melangkah lebih jauh dalam membuat suatu aplikasi. Berikut adalah contoh membuat fungsi di python :

def least\_difference(a, b, c):

diff1 = abs(a - b)

diff2 = abs(b - c)

diff3 = abs(a - c)

return min(diff1, diff2, diff3)

Dari contoh diatas, kita membuat fungsi yang bernama least\_difference, dimana fungsi ini memiliki tiga argumen, yaitu a, b, c

Fungsi dimulai dengan kata kunci ‘def’. Blok kode yang menjorok setelah tanda ‘:’ dijalankan ketika fungsi dipanggil.

‘return’ adalah kata kunci lain juga yang berkaitan dengan fungsi. Kata kunci ‘return’ adalah untuk menentukan apa yang akan dihasilkan dari fungsi tersebut.

## **Default Arguments**

Perhatikan fungsi berikut :

def greet(who="Colin"):

print("Hello,", who)

Dari contoh di atas, kita mendefinisikan fungsi yang memiliki arguments ‘who’. Di dalam fungsi tersebut kita mengassign nilai “Colin” terhadap argumen ‘who’. Ini menunjukan, jika kita tidak memberikan nilai ‘who’ ketika kita memanggil fungsi tersebut.,maka fungsi tersebut akan memiliki nilai argument ‘who’ sebagai “Colion”. Berikut contohnya :

# memanggil fungsi tanpa memberikan nilai untuk argument who

>>>print(greet())

"Hello Colion"

# memanggil fungsi dengan memberikan argument untuk nilai who

>>>print(greet("Fauzan"))

"Hello Fauzan"

## **Method**

Method adalah salah satu hal yang sangat penting dalam python. memahami method akan membuat kita semakin pro dalam menggunakan bahasa python.

Sederhananya, method adalah suatu fungsi yang dimiliki oleh suatu object.

Apa itu object? Segala yang ada di python adalah object. Contohnya kita membuat variabel bertipe string, bertipe list, bertipe numeric dan lain sebagainya, itu semua adalah object, dan setiap object memiliki fungsi yang hanya bisa digunakan oleh object tersebut. Sebagai contoh, object string memiliki method uppercase, dimana method upper ini tidak bisa digunakan oleh object lain seperti list. Tapi list juga memiliki fungsi seperti index, yang dimana fungsi index ini tidak bisa digunakan oleh data dengan tipe objek lain seperti string contohnya. Jadi intinya, fungsi yang dimiliki oleh suatu object dinamakan method. Berikut adalah contohnya :

# membuat object string

>>> huruf\_kecil = 'huruf\_kecil'

# memanggil salah satu method string

>>> print(huruf\_kecil.upper())

'HURUF\_KECIL'

# mendifinisikan object list

>>>keluarga\_ucup = ['mamah', 'papah', 'ucup', 'adek ucup', 'kaka ucup']

# memanggil method yang dimiliki oleh list

>>>print(keluarga\_ucup.index('ucup'))

2

## jika kita menggunakan method upper ketika object list, maka akan error

## karena method tersebut bukan dimiliki object list

>>>print(keluarga\_ucup.upper())

AttributeError

Traceback (most recent call last)

<ipython-input-12-13e588d6418a> in <module>

----> 1 keluarga\_ucup.upper()

AttributeError: 'list' object has no attribute 'upper'

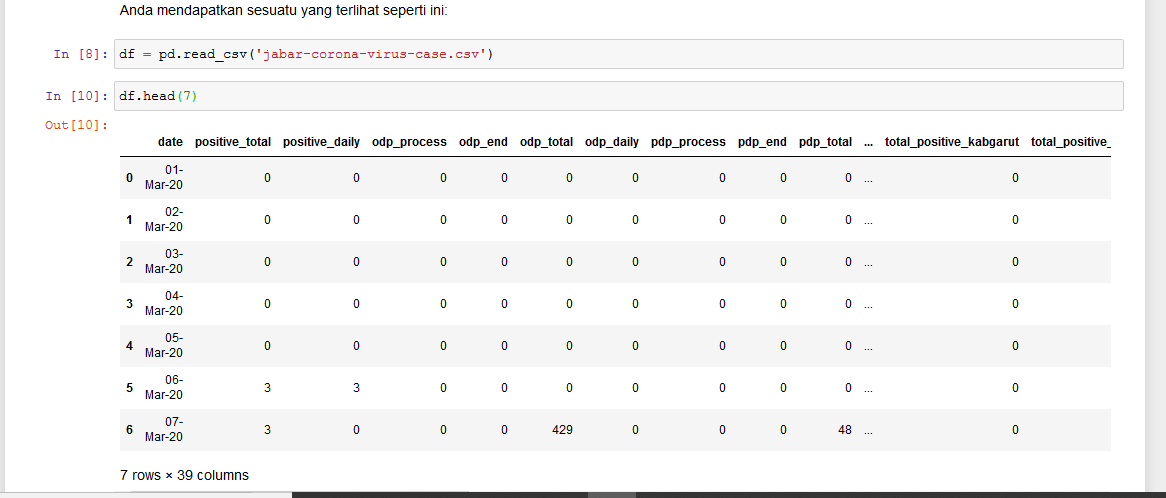
## 

## **Menggunakan External Library**

So far, kita telah mengenal fungsi bawaan dari python. Fungsi bawaan adalah fungsi yang sudah ada di python tanpa kita harus menginstall apapun. Namun di python juga ada berbagai fungsi, method, object yang bukan merupakan bawaan dari python, tapi hasil kerja dari orang lain yang telah membagikannya kepada kita, sesuatu ini kita namakan external library. Kita bisa menggunakan external library tersebut dengan cara menginstallnya dan kemudian menggunakannya di kode kita.

Ada banyak sekali external library di python. Beberapa external library yang sangat penting untuk data science adalah numpy, pandas, skit-learn, dan lain sebagainya.

Cara untuk menginstall external library ini, yang paling mudah adalah dengan menggunakan package manager. Di python package manager ini bernama ‘pip’, contohnya bila kita ingin menginstall library numpy, maka kita cukup membuka command prompt lalu ketikan ‘pip install numpy’, seperti contoh berikut :



Untuk menggunakan external library di code python kita, kita terlebih dahulu memanggil library tersebut menggunakan suatu keyword yaitu ‘import’. Berikut adalah contoh memanggil dan menggunakan external library di python :

# Definition of radius

>>>r = 0.43

# Import the math package

>>>import math

>>>pi=math.pi

# Calculate C

>>>C = 2\*pi\*r

# Calculate A

>>>A = pi\*r\*r

# Build printout

>>>print("Circumference: " + str(C))

Circumference: 2.701769682087222

>>>print("Area: " + str(A))

Area: 0.5808804816487527

Dari kode block di atas kita memanggil library math, kemudian kita menggunakan attribute pi yang ada di dalam library math.

## **External Resource**

Untuk lebih memahami lagi materi, silakan kunjungi artikel dan video di bawah ini :

List Artikel :

* <https://belajarpython.com/tutorial/fungsi-python>
* <https://www.petanikode.com/python-fungsi/>

List Videos :

* <https://www.youtube.com/watch?v=WjM68icSw3s&list=PLZS-MHyEIRo7cgStrKAMhgnOT66z2qKz1&index=12> ( Pengenalan Fungsi )
* <https://www.youtube.com/watch?v=vWuSLG_6rxA&list=PLZS-MHyEIRo7cgStrKAMhgnOT66z2qKz1&index=13> ( Fungsi Argument )
* <https://www.youtube.com/watch?v=23dDEp6WPH4&list=PLZS-MHyEIRo7cgStrKAMhgnOT66z2qKz1&index=14> ( Return Value Function )

# **Day 4 – Logic Control Flow and Loop**

Python memiliki sebuah tipe data bernama boolean, yang hanya memiliki 2 nilai, yaitu True atau False.

>>>x = True

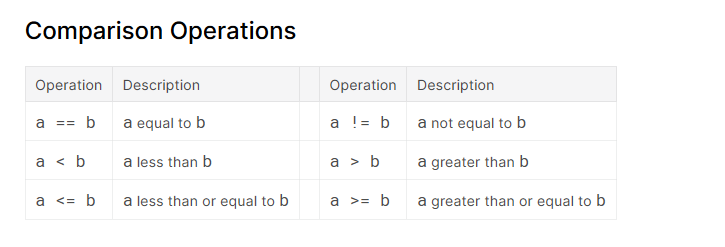
>>>print(x)

>>>print(type(x))

True

<class 'bool'>

Daripada memasukan nilai boolean (True or False) langsung kepada variable, lebih baik mendapatkan nilai dengan melakukan suatu operasi. Operasi itu dinamakan comparison operator. Comparison operator ini yang nantinya akan memberikan jawaban nilai berdasarkan logic yang kita bangun. Ada beberapa dasar comparison operator, yaitu sebagai berikut :



def can\_run\_for\_president(age):

"""Can someone of the given age run for president in the US?"""

# The US Constitution says you must "have attained to the Age of thirty-five Years"

return age >= 35

print("Can a 19-year-old run for president?", can\_run\_for\_president(19))

print("Can a 45-year-old run for president?", can\_run\_for\_president(45))

result :

Can a 19-year-old run for president? False

Can a 45-year-old run for president? True

Comparison cukup pintar juga, perhatikan :

print(3.0 == 3)

True

Tapi tidak terlalu pintar

print('3'==3)

False

Python menyediakan operator untuk menggabungkan nilai boolean menggunakan konsep bahasa standar yang biasa kita pahami seperti “and”, “or”, dan “not”.

Dengan ini, kita dapat membuat fungsi seperti ini :

def can\_run\_for\_president(age, is\_natural\_born\_citizen):

"""Can someone of the given age and citizenship status run for president in the US?"""

# The US Constitution says you must be a natural born citizen \*and\* at least 35 years old

return is\_natural\_born\_citizen and (age >= 35)

print(can\_run\_for\_president(19, True))

print(can\_run\_for\_president(55, False))

print(can\_run\_for\_president(55, True))

result :

False

False

True

Boolean dan conditional operator sangat berguna dalam mengatur alur atau logic di dalam kode kita. Namun Boolean dan conditional akan lebih powerful ketika kita menggunakannya bersama conditional statement, dengan menggunakan keyword if, else, dan elif. Conditional statement biasa disebut dengan if-then.

Conditional Statement, memungkinkan programmer untuk mengeksekusi suatu kode tertentu tergantung pada beberapa kondisi Boolean. Contoh dasar dari pernyataan conditional Python adalah ini:

def inspect(x):

if x == 0:

print(x, "is zero")

elif x > 0:

print(x, "is positive")

elif x < 0:

print(x, "is negative")

else:

print(x, "is unlike anything I've ever seen...")

inspect(0)

inspect(-15)

result :

0 is zero

-15 is negative

## **Loop**

Loop adalah cara untuk berulang kali mengeksekusi beberapa kode. Ini sebuah contoh:

planets = ['Mercury', 'Venus', 'Earth', 'Mars', 'Jupiter', 'Saturn', 'Uranus', 'Neptune']

for planet in planets:

print(planet, end=' ') # print all on same line

result :

Mercury Venus Earth Mars Jupiter Saturn Uranus Neptune

Ketika kita menggunakan loop, ada beberapa hal yang perlu diperhatikan :

1. Kita perlu memberikan nama variabel yang akan digunakan (dalam contoh di atas adalah planet)
2. Serentetan nilai yang ingin kita looping untuk melakukan operasi kepada setiap elemennya ( dalam contoh di atas adalah planets ).

range(), range() adalah suatu fungsi yang menghasilkan suatu baris nilai. Lebih jauhnya lagi ,kita bisa menggunakan fungsi help yang telah kita pelajari sebelumnya untuk lebih paham berbagai macam cara untuk menggunakannya. Berikut merupakan satu contoh sederhana :

for i in range(5):

print("Doing important work. i =", i)

result :

Doing important work. i = 0

Doing important work. i = 1

Doing important work. i = 2

Doing important work. i = 3

Doing important work. i = 4

Di python ada sebuah fungsi bawaan yang bernama enumerate(). Enumerate() memungkinkan kita untuk melakukan loop terhadap suatu object semacam list disertai dengan pengambilan index dari setiap elemennya. Contohnya seperti berikut :

# areas list

areas = [11.25, 18.0, 20.0, 10.75, 9.50]

# Change for loop to use enumerate() and update print()

for index,area in enumerate(areas) :

print("room"+str(index)+": "+str(area))

result :

room0: 11.25

room1: 18.0

room2: 20.0

room3: 10.75

room4: 9.5

Jenis loop lain dalam python adalah while loop, yang terus menerus melakukan looping sampai memenuhi suatu kondisi yang membuat dia harus berhenti, berikut contohnya :

i = 0

while i < 10:

print(i, end=' ')

i += 1

result :

0 1 2 3 4 5 6 7 8 9

Argumen dari while loop dievaluasi setiap looping, dan loop dijalankan sampai hasil evaluasi dari conditional operator bernilai False.

## **External Resource**

List Artikel :

* <https://www.petanikode.com/python-percabangan/>
* <https://www.petanikode.com/python-perulangan/>

List Videos :

* <https://www.youtube.com/watch?v=Hqndpzj0ZFg&list=PLZS-MHyEIRo7cgStrKAMhgnOT66z2qKz1&index=6> ( Percabangan 1)
* <https://www.youtube.com/watch?v=f28RoIcHZhY&list=PLZS-MHyEIRo7cgStrKAMhgnOT66z2qKz1&index=7> ( Percabangan 2)
* <https://www.youtube.com/watch?v=KMmZo_dvmyk&list=PLZS-MHyEIRo7cgStrKAMhgnOT66z2qKz1&index=8> ( Loop 1 )
* <https://www.youtube.com/watch?v=L5GGd1JHqnE&list=PLZS-MHyEIRo7cgStrKAMhgnOT66z2qKz1&index=9> ( Loop 2 )
* <https://www.youtube.com/watch?v=sLxR7vvPemY&list=PLZS-MHyEIRo7cgStrKAMhgnOT66z2qKz1&index=10> ( Loop 3 )
* <https://www.youtube.com/watch?v=S8PxQTcme9k&list=PLZS-MHyEIRo7cgStrKAMhgnOT66z2qKz1&index=11> ( Loop 4)
* <https://www.youtube.com/watch?v=ZnBZWAUusj8&list=PLZS-MHyEIRo7cgStrKAMhgnOT66z2qKz1&index=22> (Optional : Materi Pengayaan)

# **Day 5 – Python Dictionary**

Dictionary adalah suatu topik yang sangat penting dalam python dan data science. Karena dictionary seperti penyusun untuk suatu objek yang lebih kompleks seperti pandas dataframe yang akan kita pelajari nanti.

Jadi, dictionary ini adalah salah satu jenis tipe data di python yang memetakan antara key dan value dari suatu data. Berikut contohnya :

numbers = {'one':1, 'two':2, 'three':3}

Kita dapat mengakses suatu nilai dalam dictionary dengan cara seperti ini :

>>>print(numbers['one'])

1

Kita juga dapat mengubah suatu nilai dalam dictionary dengan cara berikut :

>>>numbers['one'] = 'satu'

>>>print(numbers)

{'one':satu, 'two':2, 'three':3}

Perhatikan value dari key ‘one’ berubah dari awalnya 1 menjadi ‘satu’

Jika kita melakukan loop pada suatu dictionary, maka kita akan me-loop terhadap key pada dictionary tersebut :

or k in numbers:

print("{} = {}".format(k, numbers[k]))

result :

one = satu

two = 2

three = 3

Object dictionary mempunyai suatu method yang bernama items(), dimana dengan fungsi ini kita dapat melakukan loop terhadap suatu dictionary beserta dengan key dan value nya :

or k, v in numbers.items():

print("{} = {}".format(k, v))

result :

one = satu

two = 2

three = 3

Untuk memahami lebih jauh lagi tentang dictionary, kita bisa menggunakan fungsi help sebagai penolong.

## **External Resource**

Untuk lebih memahami lagi tentang materi ini, kunjungi artikel/video dari link-link di bawah ini :

List Artikel/Tutorial :

* <https://belajarpython.com/tutorial/dictionary-python>
* <https://www.petanikode.com/python-dictionary/>

List Video :

* <https://www.youtube.com/watch?v=ARfcEqYpzkk&list=PLZS-MHyEIRo7cgStrKAMhgnOT66z2qKz1&index=21> ( Dictionary )

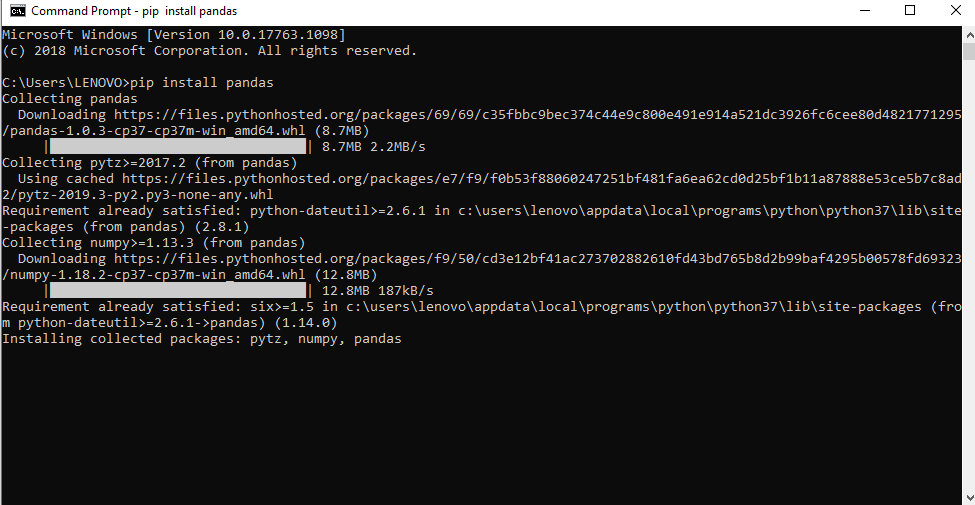
# **Day 6 – Pandas Introduction**

## **Instalasi Pandas**

Kali ini, kamu akan mempelajari semua tentang pandas, pandas adalah suatu liblary di Python yang paling populer untuk analisis data.

Pandas adalah suatu liblary yang open source, menyediakan performance yang sangat baik, struktur data yang mudah digunakan dan alat analisis data untuk Python. Pandas akan menjadi langkah besar kamu dalam perjalanan mempelajari data science.

Untuk menggunakan pandas, kamu terlebih dahulu mesti menginstall nya di command prompt dengan mengetikan pip install pandas.

Installasi Pandas

Tunggu sampai installasi selesai, ketika kamu hendak menginstall pandas, tidak hanya pandas yang akan di install, tapi juga beserta dependency nya, di gambar, ketika menginstall pandas kamu juga menginstall pytz dan numpy.

Pada artikel ini, akan dijelaskan beberapa materi fondasi untuk memahami pandas. diantaranya adalah :

1. Objek Pandas : DataFrame vs Series
2. Membaca data dari File
3. Pemeriksaan sederhana tentang Karakteristik Data

## **Objek Panas : DataFrame vs Series**

Dataframe dan series ini adalah suatu objek tempat kita menyimpan data secara terstruktur. perbedaan dari DataFrame dan Series ini terletak pada struktur nya dan juga attribute dan method-method yang mereka miliki, untuk perbedaan strukturnya adalah sebagai berikut :

1. DataFrame adalah suatu objek 2 dimensi, mirip seperti tabel
2. Series adalah objek 1 dimensi

### **Series**

Series adalah suatu objek satu dimensi yang dapat menyimpan berbagai jenis tipe data seperti integer, string, dan lain sebagainya. tipe data dari objek series ini harus seragam. berikut contoh membuat list :

import pandas as pd

x = pd.Series([6,3,4,6])

print(x)

=============

0 6

1 3

2 4

3 6

dtype: int64

Series memiliki satu sumbu saja, dari contoh di atas sumbu tersebut berada baris baris, yaitu 0, 1, 2, 3. kita bisa mengubah sumbu tersebut dengan sebagai berikut :

# cara pertama

x.index = ['a', 'b', 'c', 'd']

print(x)

===========

a 6

b 3

c 4

d 6

dtype: int64

# cara kedua

x = pd.Series([6,3,4,6], index=['a', 'b', 'c', 'd'])

print(x)

===========

a 6

b 3

c 4

d 6

dtype: int64

# check tipe data

print(type(x))

================

pandas.core.series.Series

Perhatikan : jumlah index harus sama dengan jumlah data.

### **DataFrame**

DataFrame adalah suatu objek 2 dimensi tempat menyimpan data dengan lebih terstruktur. dataframe memiliki 2 index, yaitu index baris dan index columns. Dalam satu column dataframe harus memiliki tipe data yang sama. tapi antar columnnya dataframe bisa memiliki jenis data yang berbeda. untuk lebih jelasnya perhatikan contoh berikut :

import pandas as pd

df = pd.DataFrame({'tipe\_int': [50, 21], 'tipe\_string': ['a', 'b']})

print(df)

==========

tipe\_int tipe\_string

0 50 a

1 21 b

# check tipe data

print(type(df))

==========

<class 'pandas.core.frame.DataFrame'>

## **Membaca data dari File**

Membaca data dari file adalah hal pertama yang dilakukan dalam suatu pekerjaan data science. maka hal ini sangat penting.

Ada beberapa tipe file yang biasa di gunakan untuk menyimpan data, seperti database, excel, csv. disini akan di jelaskan beberapa saja tentang cara membaca file dari berbagai sumber tersebut. akan di jelaskan 2 yaitu csv dan excel karena csv dan excel adalah sumber yang biasa di gunakan untuk menyimpan data karena kemudahannya. ok kita langsung saja.

Pandas menyediakan metode yang berbeda untuk membaca file dengan tipe berbeda. untuk membaca file bertipe csv pandas menggunakan suatu metode read\_csv(), untuk membaca file bertipe excel pandas menggunkan suatu metode bernama read\_excel. perhatikan contoh berikut :

# membaca dari dari csv

df\_from\_csv = pd.read\_csv('jabar-corona-virus-case.csv')

# membaca data dari excel

df\_from\_excel = pd.read\_excel('jabar-corona-virus-case.xlsx')

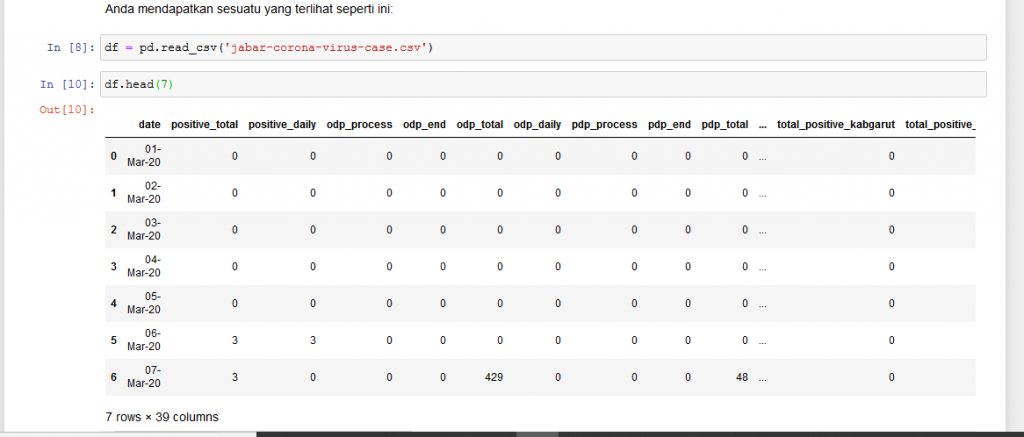
Untuk menggunakan method tersebut kita hanya perlu memasukan argument wajib, yaitu path dari file yang akan kita baca. lebih lanjutnya lagi pandas memiliki beberapa argumen optional, kalian bisa mecarinya dengan menggunakan metode help yang telah kalian pelajari sebelumnya

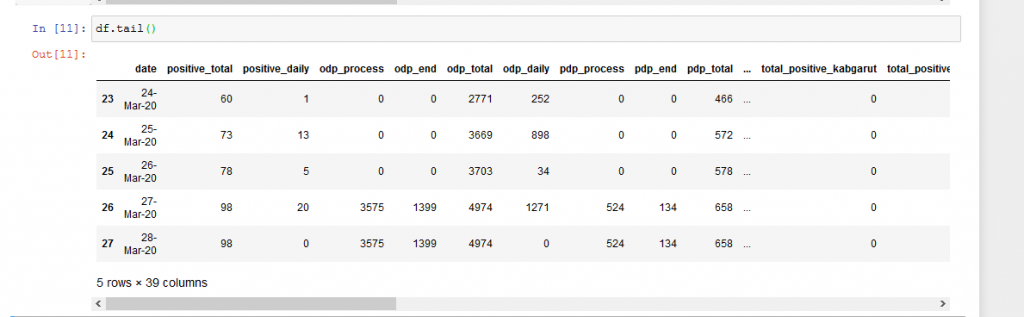
## **Pemeriksaan sederhana tentang Karakteristik Data**

Pandas memiliki beberapa method untuk memahami gambaran besar karakteristik dari data kita. diantaranya adalah :

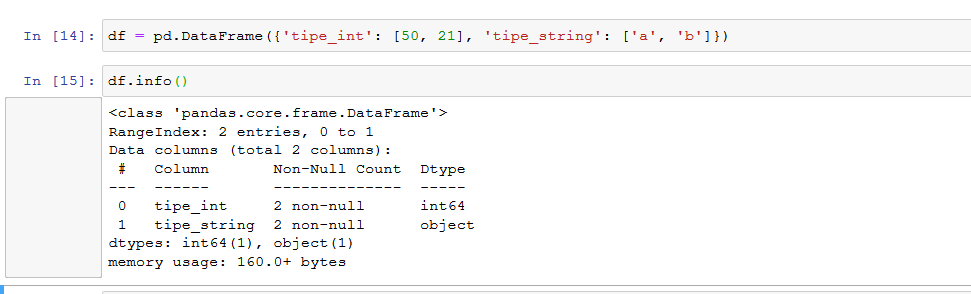
* head()
* tail()
* info()
* describe()

Metode head() adalah secara default adalah untuk melihat 5 pertama data kita, sedangkan metode tail() secara default untuk melihat 5 data kita dari terakhir. kita bisa mengganti jumlah data yang akan di tampilkan dengan memberikan data integer sebagai argument terhadap metode tersebut. contohnya sebagai beriktu :

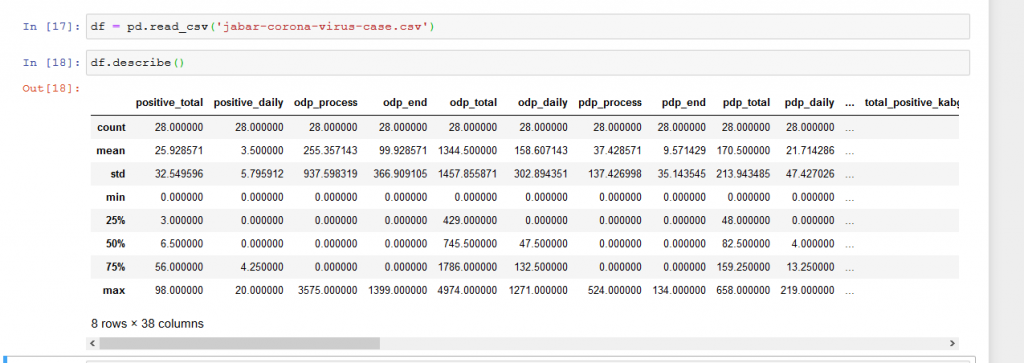
menampilkan tujuh data pertama

Menampilkan 5 data dari terakhir

info() adalah suatu method untuk melakukan summary terhadap data yang memberikan informasi tentang tipe dari index data, tipe dari column data, non-null values pada setiap column dan jumlah memory yang digunakan. berikut adalah contohnya :



descirbe() adalah sutau metode yang menghasilkan kesimpulan deskriptif statistik dari data. kesimpulan deskriptif statistik yang dihasilkan adalah seperti central tendency, dispersi dan bentuk distributsi dari data. untuk mengetahui lebih dalam tentang deskriptik statistik silahkan baca artikel [ini](https://www.yuksinau.id/statistika-deskriptif/). berikut contoh penggunaan metode ini.

Hasil Deskripsi Statistik

## **Video Pembelajaran**

* <https://youtu.be/3sU4Fn9w_ag> (Membuat Series dari list)
* <https://youtu.be/j-o7vsiWafg> ( Perbedaan Series dan List )
* <https://youtu.be/fg4HNG17qgI> ( Perbedaan Series dan Dictionary )
* <https://youtu.be/vKnx8sr9xHo> ( Membaca dan Membuat DataFrame )

[Pandas ReveiwUnduh](https://blog.sanbercode.com/wp-content/uploads/2020/04/Pandas-Reveiw.mp4)

## **External Source**

List Artikel :

* <https://www.yuksinau.id/statistika-deskriptif/> (deskriptif statistik)
* <https://petruknisme.com/2019/04/15/pengenalan-pandas-dan-series/>
* [https://code.tutsplus.com/id/tutorials/introducing-pandas–cms-26514](https://code.tutsplus.com/id/tutorials/introducing-pandas--cms-26514)

List Video :

* <https://www.youtube.com/watch?v=06cjWxfk-Zc&list=PLxBhf17jrfxEFnWyV4nuRZ24MglbAjZTE> ( Numpy )
* <https://www.youtube.com/watch?v=3krwFrozpek&list=PLxBhf17jrfxEFnWyV4nuRZ24MglbAjZTE&index=2> ( Google Colab : Optional)
* <https://www.youtube.com/watch?v=qghFcRSdCSk&list=PLxBhf17jrfxEFnWyV4nuRZ24MglbAjZTE&index=3> ( Pandas Indexing )

# **Day 7 – Akses/Indexing dan Transformasi Data**

## **Indexing DataFrame**

Akses DataFrame sering juga di kenal indexing atau Subset Selection. secara sederhana maknanya adalah kita memilih suatu data dari baris tertentu dan column tertentu. kita bisa saja memilih beberapa baris dan semua kolom dari data, semua baris dan beberapa column dari data, atau beberapa baris dan bebearpa column dari data.

Ada banyak cara-cara kita indexing suatu dataframe. diantranya adalah dengan menggunakan :

* dataframe[] : cara seperti ini dikenal sebagai index operator.
* dataframe.loc[]: operator yang menggunakan label untuk mengakses data, jika misalkan kamu melakukan dataframe.loc[‘jumlah’], berarti kamu hendak mengakses data yang mempunyai index jumlah
* dataframe.iloc[: operator yang menggunakan posisi index untuk mengakses data. jikalau kamu melakukan dataframe.iloc[2] berarti kamu hendak mengakses dataframe dengan index posisi 2

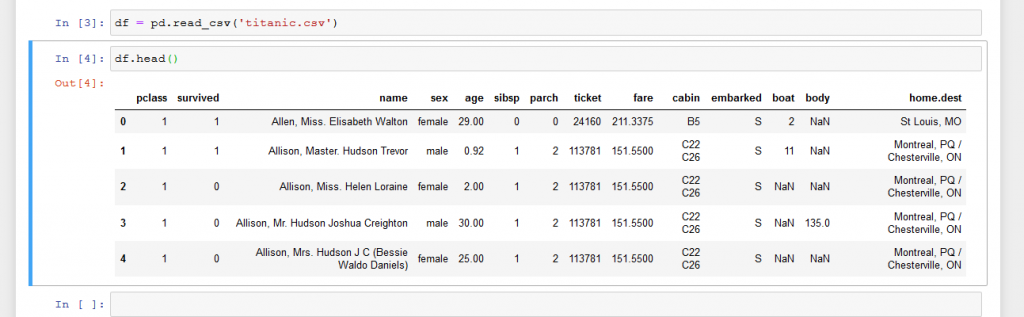
Untuk lebih jelasnya kamu bisa melihat video pembelajaran tentang ‘Mengakses DataFrame dengan index’.

## **Filtering DataFrame with Boolean**

Dapat dibilang kalau filtering dataframe dengan boolean is more human. karena kita mengakses suatu data berdasarkan logika dan dapat kita masukan ke dalam suatu statement, sebagai contoh, ‘Ambilkan data yang memiliki nilai luas lebih dari 50’. ketika mengubahnya kedalam suatu kode python ckup dengan menuliskan df>50. maka akan menghasillkan sua tu data Boolean Series berisi Ya (True) dan Tidak (False).

### **Filtering DataFrame dimulai dari pertanyaan**

Pada tulisan ini saya akan memberikan beberapa contoh mengakses DataFrame dengan Filtering by Boolean. kita akan menggunakan suatu data tentang kejadian titanic. pertama kita import data kita.

Importing and Inspecting Data

Dari gambar dapat kita lihat bahwa data kita memiliki beberapa column seperti tentang status survived, tentang umur, tentang jenis kelamain, dan lain-lain. misalkan boss kita menyuruh kita untuk melakukan beberapa hal sebagai berikut :

1. Tunjukan kepada saya semua data orang-orang yang selamat!!

2. Tunjukan kepada saya semua data laki-laki yang selamat.

3. Tunjukan kepada saya semua data tentang perempuan yang tidak selamat di umur lebih dari 40 tahun atau kurang dari 20 tahun.

Hal pertama yang mesti kita lakukan untuk melakukan operasi yang bisa menjawab permintaan di atas adalah… kita membuat terlebih dahulu suatu Boolean Series. kemudian boolean series tersebut kita masukan kedalam dataframe kita. langsung saja akan saya perlihatkan dengan contoh. dimulai dari pertanyaan pertama.

#### **1. Data orang-orang yang selamat**

# boolean series orang orang yang selamat

survived = df.survived == 1

print(survived)

=================

0 True

1 True

2 False

3 False

4 False

...

1304 False

1305 False

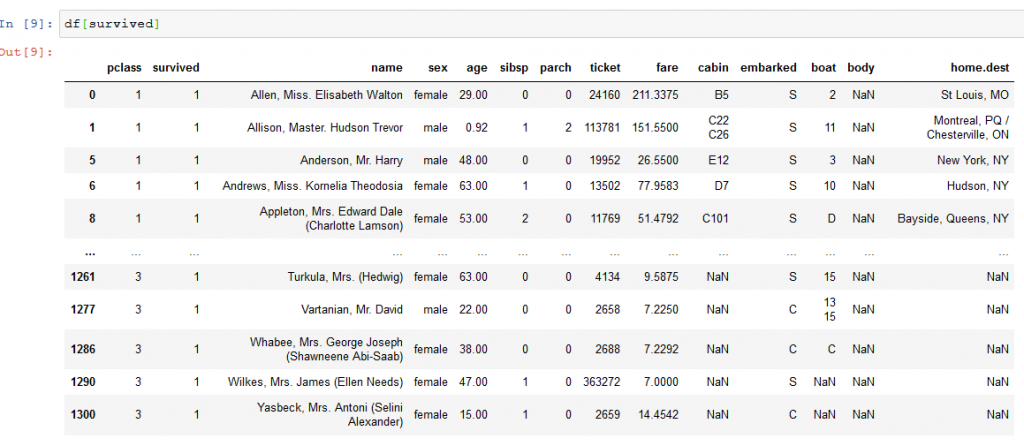
1306 False

1307 False

1308 False

Name: survived, Length: 1309, dtype: bool

Sekarang kita masukan boolean series kita ke dalam dataframe kita.

Hasil filtering data orang yang selamat

#### **2. Data laki-laki yang selamat**

Seperti biasa kita buat terlebih dahulu boolean series nya, kali ini kita membuat 2 boolean series. yaitu boolean series orang yang selamat dan bolean series orang yang berjenis kelamin laki-laki. berikut caranya

# bolean series orang yang selamat

survived = df.survived == 1

# boolean series laki-laki

laki2 = df.sex = 'male'

# tampilkan data boolean series laki2

print(laki2)

=============

0 False

1 True

2 False

3 True

4 False

...

1304 False

1305 False

1306 True

1307 True

1308 True

Name: sex, Length: 1309, dtype: bool

Sekarang kita masukan 2 boolean series ini ke dalam dataframe kita, ada sedikit hal yang perlu kamu lakukan ketika ingin memfilter dataframe dengan lebih dari satu kondisi boolean series, pertama kamu harus memberikan kurung kepada setiap boolean series mu, kedua hubungkan kedua boolean series itu dengan logical operator seperti & (and), | (or), dan lain sebagainya. berikut caranya.

print(df[(survived) & (laki2)])



#### **3. Data Perempuan yang tidak selamat dengan umur lebih dari 40 tahun atau kurang dari 20 tahun.**

Coba perhatikan ada berapa boolean series yang perlu kita buat. disini ada 4 boolean series yang perlu kita buat, yaitu orang yang tidak selamat, orang yang berjnis kelamin perempuan, orang yang berumur lebih dari 40 tahun, dan orang yang kurang dari 20 tahun. jadi mari kita buat boolean series nya.

# orang yang tidak selamat

not\_survived = df.survived == 0

# orang berjnis kelamin perempuan

perempuan = df.sex == 'female'

# orang yang berumur lebih dari 40 tahun

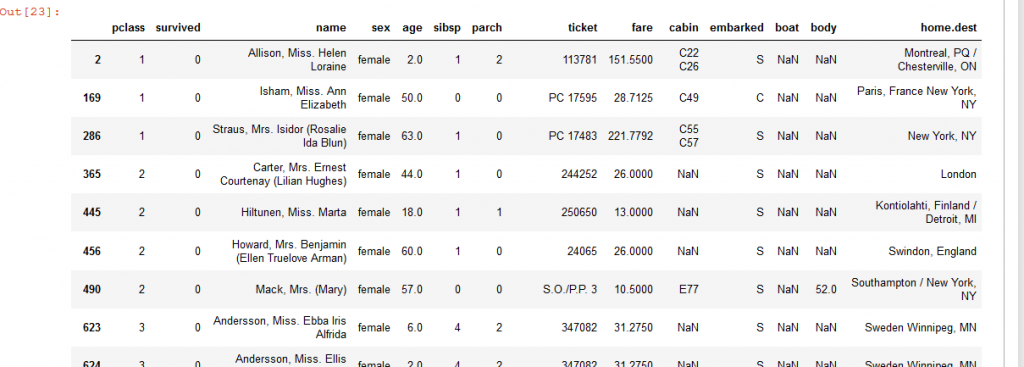
lebih\_40 = df.age > 40

# orang yang berumur kurang dari 20 tahun

kurang\_20 = df.age < 20

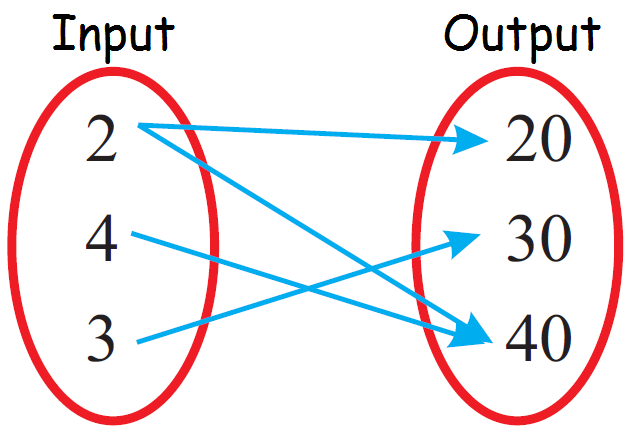
kemudian kita gabungkan boolean series ini dengan seperti ini :

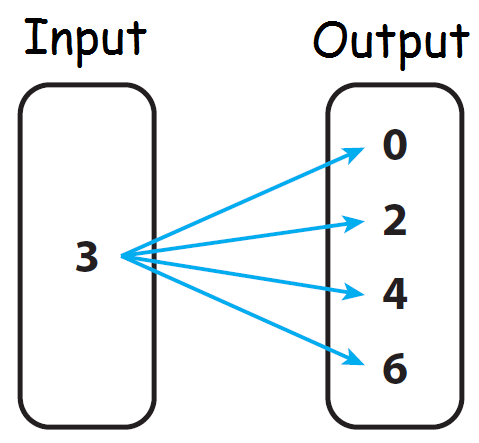
df[(not\_survived) & (perempuan) & (lebih\_40 | kurang\_20)]



## **Data Transformation**

Data transformasi adalah mengubah format data kepada suatu format yang kita inginkan. didalam matematika dikenal dengan istilah mapping. dalam melakukan transformasi kita memerlukan terlebih dahulu suatu fungsi yang dapat memetkan bentuk data awal kita menjadi bentuk data akhir yang kita inginkan.

source : <https://www.onlinemath4all.com/identifying-functions-from-mapping-diagrams.html>

source : https://www.onlinemath4all.com/identifying-functions-from-mapping-diagrams.html

Pandas menyediakan berbagai method bawaan untuk melakukan transformasi terhadap data kita, beberapa diantaranya telah diperkenalkan pada hari pertama di minggu ke dua ini. misalnya describe(), method describe ini tidak lain memetakan data mentah kita menjadi suatu data hasil tentang deskriptif statistik.

Selain itu masih banyak lagi method2 bawaan yang di sediakan pandas untuk melakukan dat transformasi seperti method mean(), dan lain sebagaiinya.

Kita juga bisa melakukan transformasi sesuai keinginan kita sendiri atau custom transformation. untuk melakukan custom transformation terlebih dahulu perlu membuat fungsi untuk melakukan mapping antara data awal hingga menjadi data yang di harapkan. berikut akan di periihatkan satu contoh transformasi data

kita kembali akan menggunakan data tentang titanic. misalkan kita ingin mengubah data kita, dimana kita ingin mengubah column age, semua nilainya di kurangi dengan nilai minimum dari kolumn tersebut. maka pertama kita definiskan fungsi tersebut.

def minus\_minimum(x):

return x - x.min()

setelah kita mendefinisikan fungsinya. sekarang kita akan menggunakan method apply() dari pandas. method apply ini akan menerima suatu fungsi kemudian ‘meng-apply’ fungsi tersebut ke dalam setiap sumbu dari dataframe, secara default sumbu yang ‘apply’ adalah sumbu column. langsung saja kita perlihatkan. sebelum nykita lihat dulu data awal kita seperti apa.

print(df.age)

================

0 29.00

1 0.92

2 2.00

3 30.00

4 25.00

...

1304 14.50

1305 NaN

1306 26.50

1307 27.00

1308 29.00

Name: age, Length: 1309, dtype: float64

Dan nilai minumum dari column age adalah :

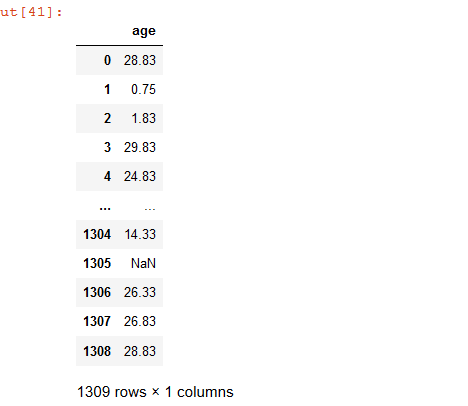
(df.age.min())

==============-

0.17

maka seharusnya apabila kita melakukan transformasi terhadap column age dengan fungsi ‘minus\_minimum’, maka seharusnya semua nilai dari column age akan di kurangi dengan nilai minimumnya. kita buktikan

print(df[['age']].apply(minus\_minimum))



Kalian lihat, hasilnya seperti yang kita harapkan, yaitu semua data dari column age dikurangi nilai minmumnya.

## **Video Pembelajaran**

* <https://youtu.be/Bl1QaerRffE> (Selecting dan Indexing Object Series 1)
* <https://youtu.be/y-MINmOar1s> (Selecting dan Indexing Object Series 2)
* <https://youtu.be/4WvN3RKOLqU> (Selecting dan Indexing Object DataFrame 1)
* <https://youtu.be/I-C2-R1zfok> ( Selecting dan Indexing Object DataFrame 2)
* <https://youtu.be/c3PidtyPnAw> (Filtering DataFrame)
* <https://youtu.be/rzyyDE65w48> (Data Transformasi)

## **External Source**

List artikel :

* <https://www.it-swarm.dev/id/python/memilih-beberapa-kolom-dalam-bingkai-data-panda/1068562049/>
* <https://www.it-swarm.dev/id/python/pilih-baris-dari-dataframe-berdasarkan-nilai-nilai-dalam-kolom-di-panda/1073294319/>
* <https://www.it-swarm.dev/id/python/jatuhkan-kolom-yang-namanya-berisi-string-spesifik-dari-panda-dataframe/1041678799/> (Pandas Filtering)

List Video :

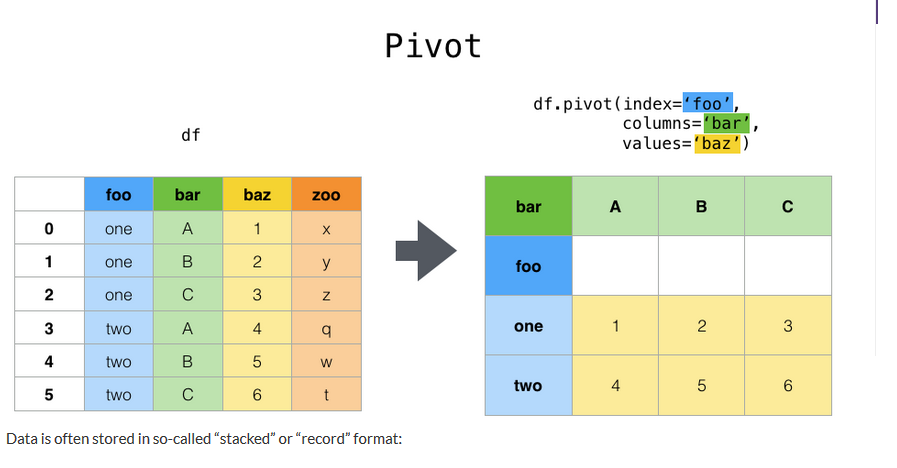
* <https://www.youtube.com/watch?v=OEaNilIlKPY&list=PLxBhf17jrfxEFnWyV4nuRZ24MglbAjZTE&index=4> (Data Transformasi)

# **Day 8 – Reshaping Data**

Dalam beberapa kasus kita perlu melakukan reshaping terhadap data untuk mengubah data yang ‘tidy’ menjadi data yang lebih relevan dan lebih optimal untuk di jadikan bahan analisis.

Ada beberapa teknik reshaping data di pandas, beberapa diantaranya yang akan dijelaskan dalam artikel ini adalah Pivoting dan Melting.

## **Pivoting Data**

****source : https://pandas.pydata.org/docs/user\_guide/reshaping.html#reshaping-by-pivoting-dataframe-objects

Pivoting adalah suatu pengubahan bentuk data dengan memutar data yang terletak di baris menjadi di column. coba perhatikan gambar di atas.

Kita juga dapat mengatakan pivoting tabel adalah melakukan perubahan pada bentuk data dari yang tadinya panjang menjadi lebar.

Ada beberapa metode untuk melakukan pivoting di pandas, yaitu pivot() dan pivot\_table(), namun metode yang bagus adalah pivot\_table(), oleh karena itu di artikel ini akan di tunjukan contoh melakukan perubahan bentuk data dengan metode pivot.

Untuk simplicity, kita akan menggunakan data buatan. misalkan kita mempunyai data seperti yang ada pada gambar diatas

import pandas as pd

data = {

'foo' : ['one']\*3 + ['two']\*3,

'bar' : ['A', 'B', 'C', 'A', 'B', 'C'],

'baz' : [1, 2, 3, 4, 5, 6],

'zoo' : ['x', 'y', 'z', 'q', 'w', 't']

}

df = pd.DataFrame(data)

print(df)

==================

foo bar baz zoo

0 one A 1 x

1 one B 2 y

2 one C 3 z

3 two A 4 q

4 two B 5 w

5 two C 6 t

kemudian kita akan melakukan pivoting tabel seperti yang ada di gambar, maka ada beberapa argumen paling penting di dalam method pivot\_tabel yang perlu kita perhatikan, diantaranya :

1. values : Argumen ini mendefinisikan column mana dari dataframe yang asli yang akan dijadikan values yang ingin kita lihat di dalam hasil pivoting tabel kita.
2. columns : Data dari column mana yang akan dijadikan column baru di dalam hasil pivoting dataframe kita.
3. index : Data dari column mana yang akan dijadikan index di dalam hasil pivoting dataframe kita.

Jika kalian bingung memahami bahasa-bahasa di atas, maka kita langsung praktekan saja dan coba kalian perhatikan baik baik hasilnya.

pivot\_result = df.pivot\_table(values='baz', index='foo', columns='bar')

print(pivot\_result)

======================

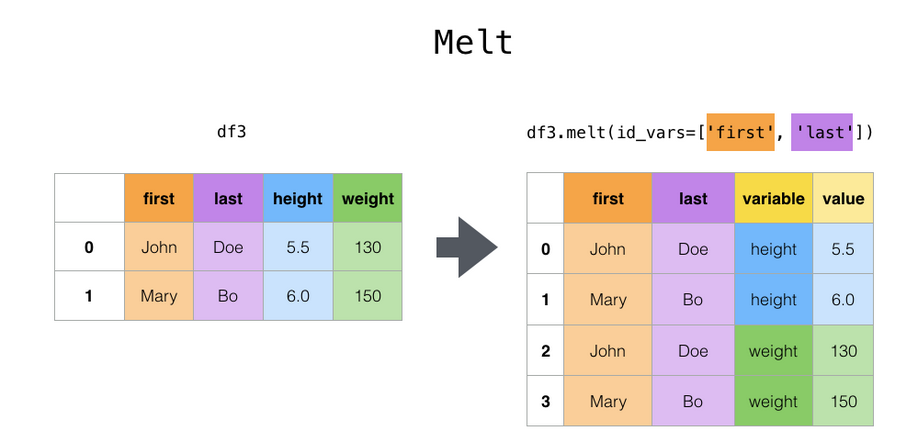
bar A B C

foo

one 1 2 3

two 4 5 6

## **Melting Data**

****source = https://pandas.pydata.org/docs/user\_guide/reshaping.html#reshaping-by-melt

Melting adalah ‘unpivoting’, melting mengubah suatu data dengan memutar dari yang tadinya berada di posisi column menjadi di posisi row.

Melting mengubah data yang tadinya lebar menjadi panjang.

Melting berguna ketika ada suatu kasus dimana terdapat satu atau lebih column yang merupakan identifier variabel, dan column lain adalah measure variabel. Identifier variabel adalah suatu entitas yang melakukan identifikasi terhadap suatu variabel yang sedang diukur.

Apabila kita memiliki suatu besaran yang nilainya selalu berubah, kemudian kita ingin mengamati perubahan itu, berarti kita sedang memperlakukan besaran itu sebagai measure variabel. dan kita perlu sesuatu yang mereferensikan perubahan tersebut. hal itu disebut identifier variable.contohnya nyatanya adalah apabila kita mengamati perubahan suhu setiap waktu. maka perubahan suhu adalah variabel yang di ukur, dan waktu adalah identifier variabel.

Sekarang kita akan mencoba mempraktekan teknik melting terhadap suatu dataframe :

# Membuat datafame sederhana

# importing pandas as pd

import pandas as pd

# membuat dataframe

df = pd.DataFrame({'Name': {0: 'John', 1: 'Bob', 2: 'Shiela'},

'Course': {0: 'Masters', 1: 'Graduate', 2: 'Graduate'},

'Age': {0: 27, 1: 23, 2: 21}})

df

=====================

Name Course Age

0 John Masters 27

1 Bob Graduate 23

2 Shiela Graduate 21

Dalam menggunakan method melt, ada beberapa argumen yang penting untuk diperhatikan, diantaranya adalah :

1. id\_vars : columns mana yang akan dijadikan identifier variabel
2. value\_vars : columns yang akan menjadi value setelah kita melakukan ‘unpivot’ terhadap variabel, jika tidak didefinisikan maka semua column selain id\_vars akan dijadikan value\_vars.

# Unpivot table

pd.melt(df, id\_vars =['Name'], value\_vars =['Course', 'Age'])

======================

Name variable value

0 John Course Masters

1 Bob Course Graduate

2 Shiela Course Graduate

3 John Age 27

4 Bob Age 23

5 Shiela Age 21

# **Day 9 – Grouping Data**

Dalam kesempatan sekarang kita akan membahas teknik-teknik grouping pada data.

Grouping adalah suatu teknik memisahkan data berdasarkan kriteria tertentu. untuk melakukan grouping di pandas kita menggunakan suatu method yang bernama groupby(). definisi abstraknya dari grouping adalah kita melakukan mapping suatu data terhadap suatu group.

ketika melakukan grouping ada beberapa proses yang terjadi secara berurutan, yaitu :

1. Splitting : Memisahkan data kedalam suatu group berdasarkan kriteria tertentu.
2. Applying : Melakukan suatu operasi terhadap sekumpulan data di group-group tersebut.
3. Combining : Menggabungkan data menjadi suatu struktur baru

Langsung saja kita akan melihat bagaimana process split-apply-combine. kita akan menggunakan data sederhana untuk tujuan pembelajaran.

import numpy as np

import pandas as pd

df = pd.DataFrame([('bird', 'Falconiformes', 389.0),

('bird', 'Psittaciformes', 24.0),

('mammal', 'Carnivora', 80.2),

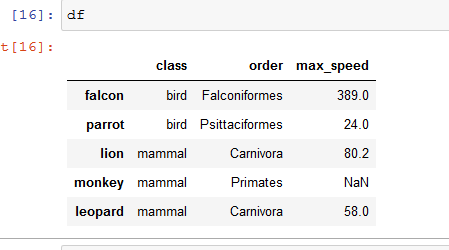
('mammal', 'Primates', np.nan),

('mammal', 'Carnivora', 58)],

index=['falcon', 'parrot', 'lion', 'monkey', 'leopard'],

columns=('class', 'order', 'max\_speed'))

df

dataframes

Mari kita praktikan method groupby terhadap data tersebut. milskan kita group data tersebut berdasarkan class.

df.groupby('class')

====================

<pandas.core.groupby.generic.DataFrameGroupBy object at 0x000002054AEB2808>

Coba kalian perhatikan, pada tahap ini kita baru sampai pada process splitting berdasarkan kategori ‘class’. hasil dari groupby ini hanya sebuah objek GroupBy. objek groupby ini adalah representasi dari suatu hasil penggabungan data berdasarkan group class, namun tidak menghasilkan nilai apapun sebelum kita memberikan suatu operasi terhadap hasil dari setiap grouping tersebut. prosess melakukan operasi terhadap sekumpulan data yang telah di groupkan adalah process applying. kita bisa menggunakan berbagai jenis operasi terhadap objek groupby tersebut. ada operasi yang telah di sediakan sebagai operasi bawaan dari pandas seperti mean(), min(), max(), dan lain sebagainya. dapat pula menggunakan custom operasi tergantung apa yang kita inginkan. mari kita coba praktekan salah satu jenis operasi kepada objek groupby tersebut.

df.groupby('class').mean()

===========================

max\_speed

class

bird 206.5

mammal 69.1

Coba kalian perhatikan. kita telah berhasil menggunakan groupby dan memberikan hasil. jenis operasi yang di apply terhadap setiap group adalah rata-rata. dan hasilnya di combine menjadi struktur data baru. dengan begini kita menhasilkan suatu hasil analisis sederhana yaitu bahwa binatang dengan class bird memiliki kecepatan maksimal rata-rata sebesar 206.5, dan bintang class mamalia sebesar 69.1

Untuk memahami mater lebih lanjut, silahkan kalian tonton video berikut.

## **Video Pembelajaran**

* <https://youtu.be/FQWm_HJAhJ8>
* <https://youtu.be/I1ZTZuZ1dtw>

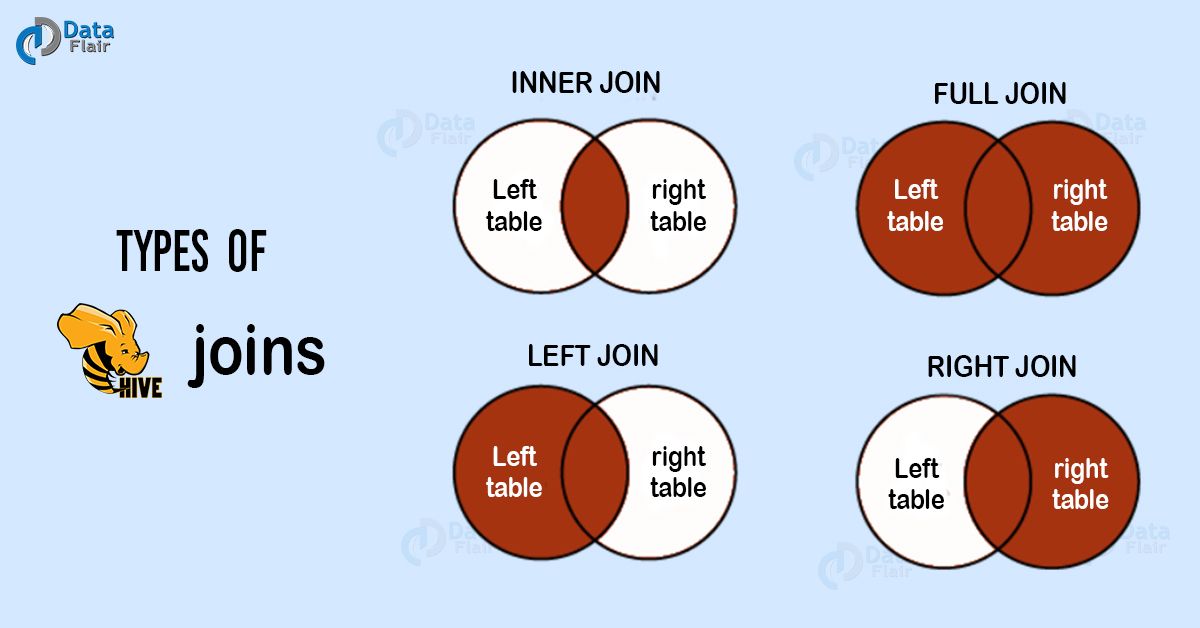
# **Day 10 – Merging DataFrame**

Menggabungkan data adalah salah satu hal yang penting dalam suatu pekerjaan data analyst atau data science. karena dalam banyak kasus data selalu berada dri berbagai file dan sumber. Maka menguasai penggabungan data beserta logika-logika penggabungan data menjadi hal yang esensi untuk di kuasai.

Pandas menyediakan berbagai tool atau method yang memudahkan kita untuk melakukan penggabungan data dengan berbagai macam logik. method-method tersebut diantaranya seperti join(), merge(), concat(). kali ini kita kan mempelejari beberapa teknik tersebut.

Ada beberapa logika penggabungan data, diantaranya adalah :

1. Inner Join
2. Full Outer Join
3. Left Outer Join
4. Right Outer Join
5. dll.

source : https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2018/03/Types-of-Hive-joins.jpg

## **Penggabungan Baris dan Column dengan Metod concat()**

Concat adalah salah satu metode untuk melakukan penggabungan data berdasarkan sumbu tertentu. langsung saja kita lihat.

misalkan kita mempunyai data seperti ini :

import pandas as pd

df1 = pd.DataFrame({'A': ['A0', 'A1', 'A2', 'A3'],

'B': ['B0', 'B1', 'B2', 'B3'],

'C': ['C0', 'C1', 'C2', 'C3'],

'D': ['D0', 'D1', 'D2', 'D3']},

index=[0, 1, 2, 3])

df2 = pd.DataFrame({'A': ['A4', 'A5', 'A6', 'A7'],

'B': ['B4', 'B5', 'B6', 'B7'],

'C': ['C4', 'C5', 'C6', 'C7'],

'D': ['D4', 'D5', 'D6', 'D7']},

index=[4, 5, 6, 7])

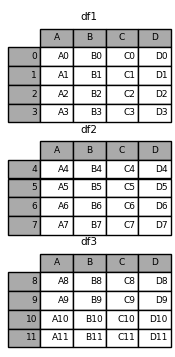
df3 = pd.DataFrame({'A': ['A8', 'A9', 'A10', 'A11'],

'B': ['B8', 'B9', 'B10', 'B11'],

'C': ['C8', 'C9', 'C10', 'C11'],

'D': ['D8', 'D9', 'D10', 'D11']},

index=[8, 9, 10, 11])



Perhatikan, data yang kita punya sangatlah simple, kita mempunyai 3 data yang mempunyai nama column yang sama. apabila kita ingin menggabungkan baris data ini, kita bisa menggunakan suatu method concat(). pertama kita buat terlebih dahulu list dari data yang akan kita gabung, kemudian masukan list tersebut sebagai argument kedalam method concat(). seperti ini.

# list data yang akan di gabung

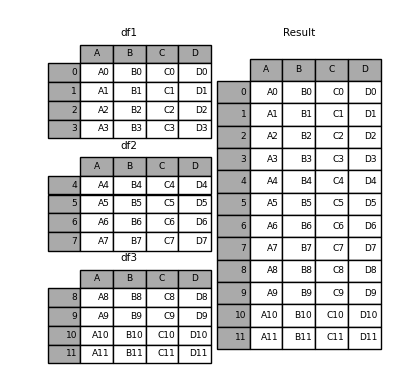
list\_dataframe = [df1, df2, df3]

# gabungkan data dengan metode concat dari pandas

pd.concat(list\_dataframe)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **A** | **B** | **C** | **D** |
| **0** | A0 | B0 | C0 | D0 |
| **1** | A1 | B1 | C1 | D1 |
| **2** | A2 | B2 | C2 | D2 |
| **3** | A3 | B3 | C3 | D3 |
| **4** | A4 | B4 | C4 | D4 |
| **5** | A5 | B5 | C5 | D5 |
| **6** | A6 | B6 | C6 | D6 |
| **7** | A7 | B7 | C7 | D7 |
| **8** | A8 | B8 | C8 | D8 |
| **9** | A9 | B9 | C9 | D9 |
| **10** | A10 | B10 | C10 | D10 |
| **11** | A11 | B11 | C11 | D11 |

Berikut hasilnya, hasil tersebut merupakan gabungan yang seperti ini :



Apabila kita ingin menggabungkan data tersebut berdasarkan column, kita mesti memberikan satu argument lagi kedalam method concat yaitu axis, dengan nilai 1. seperti ini

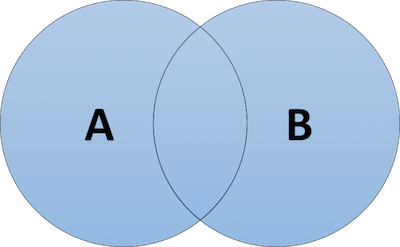
pd.concat(list\_dataframe, axis=1)

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **A** | **B** | **C** | **D** | **A** | **B** | **C** | **D** | **A** | **B** | **C** | **D** |
| **0** | A0 | B0 | C0 | D0 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **1** | A1 | B1 | C1 | D1 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **2** | A2 | B2 | C2 | D2 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **3** | A3 | B3 | C3 | D3 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **4** | NaN | NaN | NaN | NaN | A4 | B4 | C4 | D4 | NaN | NaN | NaN | NaN |
| **5** | NaN | NaN | NaN | NaN | A5 | B5 | C5 | D5 | NaN | NaN | NaN | NaN |
| **6** | NaN | NaN | NaN | NaN | A6 | B6 | C6 | D6 | NaN | NaN | NaN | NaN |
| **7** | NaN | NaN | NaN | NaN | A7 | B7 | C7 | D7 | NaN | NaN | NaN | NaN |
| **8** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | A8 | B8 | C8 | D8 |
| **9** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | A9 | B9 | C9 | D9 |
| **10** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | A10 | B10 | C10 | D10 |
| **11** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | A11 | B11 | C11 | D11 |

Coba kalian perhatikan, penggabungan ini adalah penggabungan sumbu column pada setiap dataframe, dan pada sumbu baris di lakukan penggabungan juga berdasarkan indeks. secara default logika penggabunganya adalah outer join. kita akan melihat beberapa logika penggabungan setelah ini.

## **Logika-Logika Penggabungan Dengan Metode merge**

### **Full Outer Join**

****source : https://static.cdn-cdpl.com/source/998b78e349061b4971c0a2b0e8d6be41/sql6.png

* Full Outer Join menghasilkan gabungan index bari dari setiap data yang di gabung.
* Full Outer Join menggabungan pula column yang berbeda
* Memberikan nilai null untuk nilai column yang tidak di miliki oleh suatu index tertentu.

Misalkan kita mempunyai data seperti ini :

left = pd.DataFrame({'key': ['K0', 'K1', 'K2', 'K3'],

'A': ['A0', 'A1', 'A2', 'A3'],

'B': ['B0', 'B1', 'B2', 'B3']})

right = pd.DataFrame({'key': ['K0', 'K1', 'K4', 'K5'],

'C': ['C0', 'C1', 'C2', 'C3'],

'D': ['D0', 'D1', 'D2', 'D3']})

print('data pertama')

print(=========)

print(left)

print('data kedua')

print(=========)

print(right)

data pertama

=========

key A B

0 K0 A0 B0

1 K1 A1 B1

2 K2 A2 B2

3 K3 A3 B3

data kedua

=========

key C D

0 K0 C0 D0

1 K1 C1 D1

2 K4 C2 D2

3 K5 C3 D3

Kita akan menggabungkan data ini denga menggunakan method merge() dari pandas. dalam penggabungan kita kan gunakan column key dari data kita sebagai variabel identifier dari data kita, atau sebagai column key/referensi.

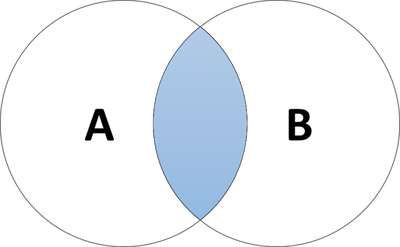
* Untuk melakukan penggabungan dengan menggunakan column key sebagai referensi penggabungan, maka kita akan memberikan argument on dengan nilai ‘key’
* untuk melakukan penggabungan outer join kita akan memberikan argument how dengan nilai ‘outer’

# gabungkan data dengan logika outer join

pd.merge(left, right, on='key', how='outer')

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **key** | **A** | **B** | **C** | **D** |
| **0** | K0 | A0 | B0 | C0 | D0 |
| **1** | K1 | A1 | B1 | C1 | D1 |
| **2** | K2 | A2 | B2 | NaN | NaN |
| **3** | K3 | A3 | B3 | NaN | NaN |
| **4** | K4 | NaN | NaN | C2 | D2 |
| **5** | K5 | NaN | NaN | C3 | D3 |

### **Inner Join**

****source : https://static.cdn-cdpl.com/source/998b78e349061b4971c0a2b0e8d6be41/sql1.png

Inner join ini akan menggabungkan data antara nilai key/referensi yang beririsan.

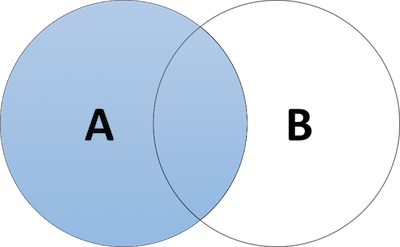
Kita akan mencoba melakukan inner join dengan data yang sama pada outer join, Untuk melakukan inner join dengan metode merge(), kurang lebih sama seperti yang di lakukan pada outer join, kita hanya perlu mengubah argument argument how menjadi inner.

pd.merge(left, right, on='key', how='inner')

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **key** | **A** | **B** | **C** | **D** |
| **0** | K0 | A  0 | B0 | C0 | D0 |
| **1** | K1 | A1 | B1 | C1 | D1 |

Coba perhatikan, bahwa data yang di gabungkan adalah data yang mempunyai nilai key beririsan.

### **Left Join**

****source : https://static.cdn-cdpl.com/source/998b78e349061b4971c0a2b0e8d6be41/sql2.png

Left Outer Join menghasilkan seluruh data dari data yang kiri ditambah data kanan yang memiliki nilai key yang sama dengan data dari kiri.

Kita akan mencoba melakukan left outer join dengan data yang sama pada outer join, untuk melakukan left outer join kita hanya perlu memberikan nilai ‘left’ kepada argument how.

pd.merge(left, right, on='key', how='left')

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **key** | **A** | **B** | **C** | **D** |
| **0** | K0 | A0 | B0 | C0 | D0 |
| **1** | K1 | A1 | B1 | C1 | D1 |
| **2** | K2 | A2 | B2 | NaN | NaN |
| **3** | K3 | A3 | B3 | NaN | NaN |

### **Right Join**

Right outer join ini hanya kebalikan dari left outer join.

Untuk melakukan operasi penggabungan right outer join dengan metode merge, kita hanya perlu memberikan nilai ‘right’ pada argument how.

pd.merge(left, right, on='key', how='right')

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **key** | **A** | **B** | **C** | **D** |
| **0** | K0 | A0 | B0 | C0 | D0 |
| **1** | K1 | A1 | B1 | C1 | D1 |
| **2** | K4 | NaN | NaN | C2 | D2 |
| **3** | K5 | NaN | NaN | C3 | D3 |

## **Video Pembelajaran**

* <https://youtu.be/BeTEJCThNXk>
* <https://youtu.be/n2QuPsIcUW0>

## **External Source**

List Artikel :

* https://www.it-swarm.dev/id/python/penggabungan-panda-101/807882802/

# **Day 11 - Pengenalan Matplotlib**

## **Pengenalan**

Manusia adalah makhluk visual. kita dapat lebih mudah memahami sesuatu lebih baik dengan visualisasi. kita dapat lebih mudah menemukan pola-pola yang mungkin tersembunyi apa bila kita bisa melihatnya secara visual.

Pada artikel ini kita akan membahasa suatu topik tentang Visualisasi menggunakan liblary matplotlib. kita akan belajar bagaimana caranya membuat objek visual yang informatif dan menarik.

Matplotlib adalah suatu liblary yang low-level untuk membangun objek visual. maksudnya low-level bukan jelek ya. tapi low-level ini memberikan kita control sampai ke komponen-komponen kecilnya. kalau misalkan kita buat mobil, kita mengontrol sampai pembuatan ke baut bautnya. Oleh karena itu matplotlib penting di fahami sebagai dasar apabila ingin mendalami bidang visualisasi data.

## **Install Library Matplotlib**

Untuk dapat menggunakan library matplotlib, install terlebih dahulu library tersebut dengan :

pip install matplotlib

atau

conda install matplotlib

## **Komponen Besar : Figure dan Axes**

Kita akan membahas top-level component, yaitu komponen terbesar dalam suatu objek visualisasi, Figure dan Axes.

Sebelum dijelaskan lebih jauh lagi, mari lihat terlebih dahulu seperti apa contoh sederhana visualisasi data.

import random

import numpy as np

import matplotlib.pyplot as plt

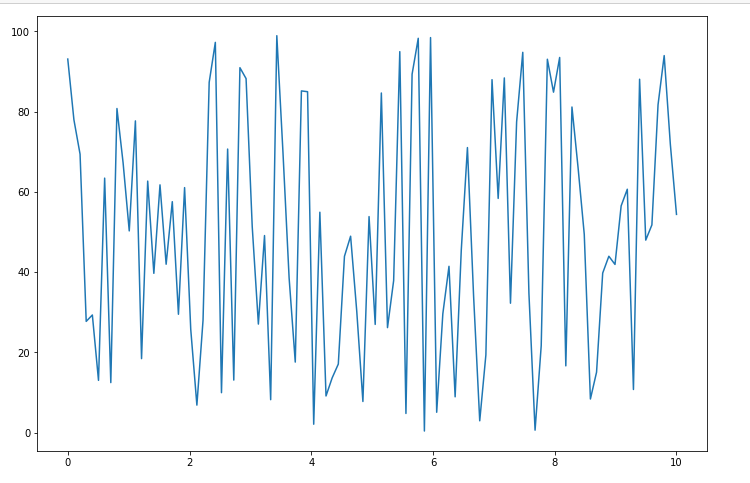
x = np.linspace(0, 10, 100)

y = [random.random()\*100 for i in range(100)]

plt.figure(figsize=(12, 8))

plt.plot(x, y)

plt.show()



Ok, sampai sini setidaknya kalian mempunyai gambaran besar tentang objek visual data. Kita akan lanjut bahas tentang komponen terbesar dalam objek visual ini.

Figure, adalah window atau page atau halaman dalam objek visual. kalau kita ngegambar di kertas, maka kertas tersebutlah yang di namakan figure.

Axis, kedalam Figure kita menambahkan Axis, Axis adalah suatu area di dalam figure dimana data akan di plot. di dalam Axis juga terdapat berbagai macam komponen lagi seperti x-axis, y-axis, dan lain sebagainya. hal ini akan kalian lihat di artikel ke dua.

Setelah memahami komponen terbesar dari matplotlib, berikut akan di tunjukan beberapa macam perintah kode untuk membuat component-component tersebut dan memberikan data kepada komponen Axis.

Beikut adalah method untuk membuat figure :

import matplotlib.pyplot as plt

fig = plt.figure()

print(fig)

===========

<Figure size 432x288 with 0 Axes>

Perhatikan kita mempunyai objek Figure dengan 0 Axis. sekarang kita akan menambahkan axis kedalam objek figure kita dengan method add\_subplot(). kemudian kita tunjukan objek figure kita dengan method show().

# membuat objek figure

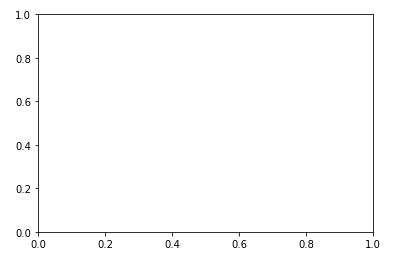
fig = plt.figure()

# membuat objek axes

ax = fig.add\_subplot()

# menampilkan objek visual

plt.show()



Sekarang kita akan tambahkan data kedalam objek tersebut dengan memanggil method plot() dan memberikan data sebagai argument method tersebut :

# membuat objek figure

fig = plt.figure()

# membuat objek axes

ax = fig.add\_subplot()

# membuat data

data\_x = [1, 2, 3, 4, 5]

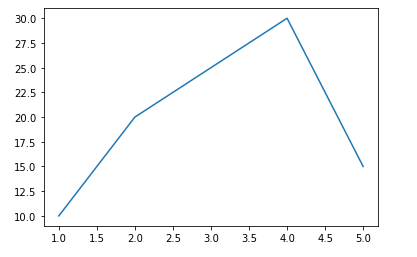
data\_y = [10, 20, 25, 30, 15]

# menambahkan dalam ke objek axes

ax.plot(data\_x, data\_y)

# menampilkan data

plt.show()



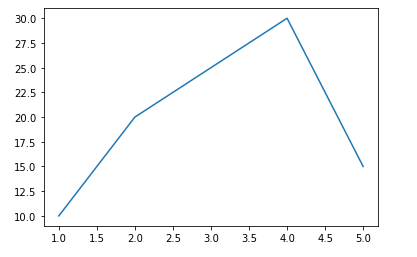
Begitulah cara sederhana untuk membuat objek visualisasi, sekarang akan di tunjukan cara lain yang lebih clean.

# membuat objek visual

plt.plot(data\_x, data\_y)

# menampilkan objek visual

plt.show()



Perhatikan, kedua cara tadi menghasilkan hasil yang sama. tapi cara kedua lebih clean karena tidak memerlukan banyak kode. sebenarnya kedua cara tersebut sama-sama menghasilkan komponen Figure dan Axis. tetapi cara pertama Explicit sedangkan cara kedua Implicit.

Cara yang Explicit direkomendasikan ketika kita ingin membuat visualisasi data yang lebih kompleks. karena kita dapat mengontrol berbagai macam komponen yang terdapat di dalam objek figure dan axes, saat ini kalian belum banyak melihat komponen tersebut, materi ini akan diberikan di materi ke 2.

Namun, kita juga dapat membuat cara visualisasi data lebih simple dengan menggabungkan pembuatan objek Figure dan Axis dengan sekali jalan, melalui pemanggilan method subplots(), seperti ini :

# membuat objek figure dan axes

fig, ax = plt.subplots()

# membuat data

data\_x = [1, 2, 3, 4, 5]

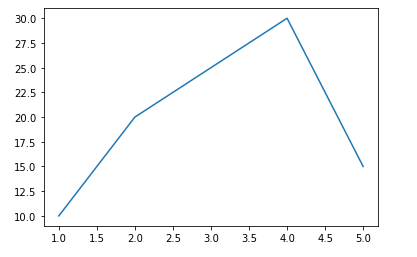
data\_y = [10, 20, 25, 30, 15]

# menambahkan dalam ke objek axes

ax.plot(data\_x, data\_y)

# menampilkan data

plt.show()



Memungkinkan untuk memplot lebih dari satu data kedalam suatu axis dengan cara seperti ini :

fig, ax = plt.subplots()

data\_x1 = [1, 2, 3, 4]

data\_y1 = [10, 20, 25, 30]

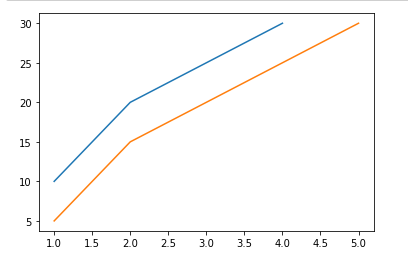
data\_x2 = [1, 2, 3, 4, 5]

data\_y2 = [5, 15, 20, 25, 30]

ax.plot(data\_x1, data\_y1)

ax.plot(data\_x2, data\_y2)

plt.show()



## **Video Belajar**

* <https://youtu.be/5DUjDr_iCQ4> ( Data Science Pengenalan Matplotlib )
* <https://youtu.be/lxpf566gsDY> ( Klarifikasi Video )

# **Day 12 - Visualization : Custom Visualisasi**

Pada materi sebelumnya telah dijelaskan cara membuat objek figure, axes, dan memplot data ke dalam axis tersebut.

Pada artikel ini akan dibahas tentang meng-custom visualisasi kita agar lebih informatif dan menarik.

Berikut adalah raw objek visualisasi data yang akan di custom :

import matplotlib.pyplot as plt

# membuat component figure dan axis

fig, ax = plt.subplots()

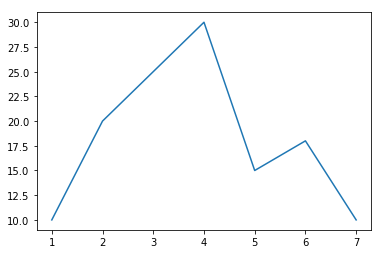
data\_x = [1, 2, 3, 4, 5, 6, 7]

data\_y = [10, 20, 25, 30, 15, 18, 10]

# memberikan data kedalam axis

ax.plot(data\_x, data\_y)

plt.show()



## **Marker Data Point**

Coba perhatikan bahwa bentuk plot dari data adalah berupa line plot, tidak dapat di ketahu dimana tepatnya data poin terletak. terdapat sebuah argument dalam methode plot untuk memberikan marker/tanda terhadap data poin, argument tersebut adalah marker, perhatikan kode berikut :

import matplotlib.pyplot as plt

# membuat component figure dan axis

fig, ax = plt.subplots()

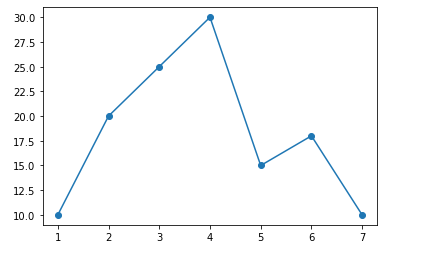
data\_x = [1, 2, 3, 4, 5, 6, 7]

data\_y = [10, 20, 25, 30, 15, 18, 10]

# memberikan data kedalam axis

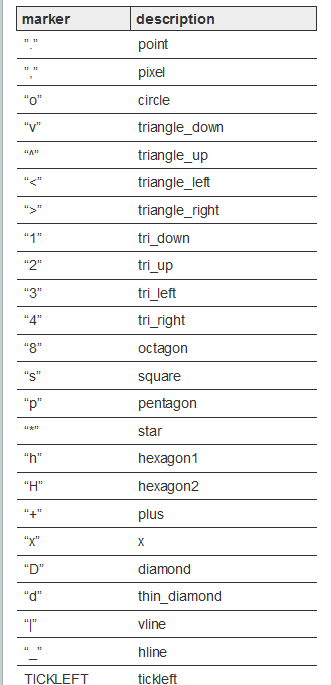
ax.plot(data\_x, data\_y, marker='o')

plt.show()



Di atas telah di berikan nilai ‘o’ terhadap arguent marker, ‘o’ artinya bulat, jadi di berikan marker bulat terhadap setiap data poin, sehingga dapat lebih jelas membedakan antara garis penghubung dengan data poin.

Nilai marker tidak hanya ‘o’ berikut opsi nilai untuk memberikan berbagai jenis bentuk marker terhadap data poin.

source : <https://matplotlib.org/1.4.1/api/markers_api.html>

## **Custom LineStyle**

Garos penghubung antara data poin dapat di ubah dengan memberikan argument linestyle terhadap method plot, perhatikan contoh berikut :

import matplotlib.pyplot as plt

# membuat component figure dan axis

fig, ax = plt.subplots()

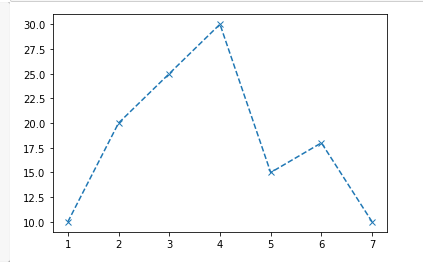
data\_x = [1, 2, 3, 4, 5, 6, 7]

data\_y = [10, 20, 25, 30, 15, 18, 10]

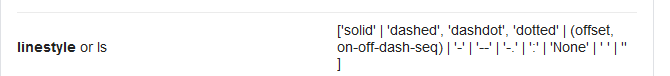
# memberikan data kedalam axis

ax.plot(data\_x, data\_y, marker='x', linestyle='--')

plt.show()



Di atas di berikan nilai ‘–‘ terhadap argument linestyle, terdapat berbagai opsi untuk memberikan berbagai jenis bentuk terhdap linestyle, yaitu sebagai berikut :



## **Memilih Warna**

Dapat di pilih warna plot dengan memberikan argument color terhadap method plot.

Memilhi value untuk argument color bisa dengan berbagai bentuk format warna, seperti format RGB, hex code, atau label warna biasa seperti ‘r’ untuk read, ‘b’ untuk blue dan lain sebagainya. seperti beriktu :

import matplotlib.pyplot as plt

# membuat component figure dan axis

fig, ax = plt.subplots()

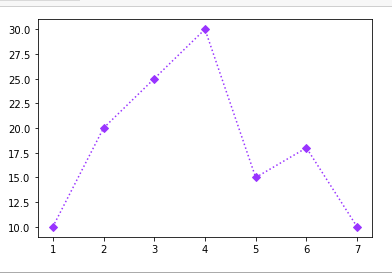
data\_x = [1, 2, 3, 4, 5, 6, 7]

data\_y = [10, 20, 25, 30, 15, 18, 10]

# memberikan data kedalam axis

ax.plot(data\_x, data\_y, marker='D', linestyle='dotted', color='#9934FF')

plt.show()



## **Custom Axis Label and Title**

Component lain yang penting untuk di custom adalah axis label karena axis label menunjukan variabel apa yang sedang di plot.

Axis terdapat 2, yaitu x-axis dan y-axis. untuk mengubah x-axis dapat menggungkan method set\_xlabel(), untuk mengubah y-axis dapat menggungkan set\_ylabel().

Title juga adalah component yang sangat penting untuk menunjukan data apa yang sedang di visualisasikan. untuk mengubah title dapat menggunakan method set\_title().

perhatikan contoh berikut :

import matplotlib.pyplot as plt

# membuat component figure dan axis

fig, ax = plt.subplots()

data\_x = [1, 2, 3, 4, 5, 6, 7]

data\_y = [10, 20, 25, 30, 15, 18, 10]

# memberikan data kedalam axis

ax.plot(data\_x, data\_y, marker='D', linestyle='dotted', color='#9934FF')

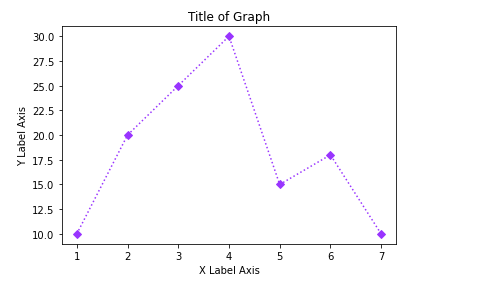
# mengubah axis label dan title

ax.set\_xlabel('X Label Axis')

ax.set\_ylabel('Y Label Axis')

ax.set\_title('Title of Graph')

plt.show()



## **Video Pembelajaran**

* <https://youtu.be/yj-Vdi0t4rk>

# **Day Visualization : Multiple Plots**

Pada hari pertama di minggu ke 3 ini, telah di tunjukan bahwa memungkinkan untuk membuat multiple data plot dalam satu axis. seperti ini :

fig, ax = plt.subplots()

data\_x1 = [1, 2, 3, 4]

data\_y1 = [10, 20, 25, 30]

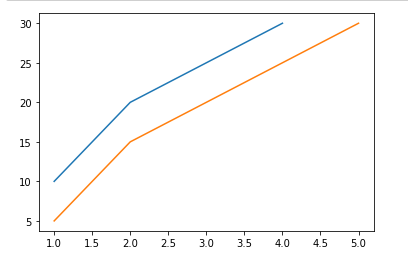
data\_x2 = [1, 2, 3, 4, 5]

data\_y2 = [5, 15, 20, 25, 30]

ax.plot(data\_x1, data\_y1)

ax.plot(data\_x2, data\_y2)

plt.show()



Namun, dalam beberapa kasus, ketika multiple plots di masukan ke dalam satu axes, dapat memungkinkan membuat objek visualisasi menjadi tidak rapi. contohnya seperti ini :

fig, ax = plt.subplots()

data\_x1 = [1, 2, 3, 4]

data\_y1 = [10, 20, 25, 30]

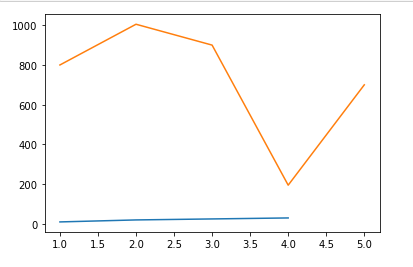
data\_x2 = [1, 2, 3, 4, 5]

data\_y2 = [800, 1005, 900, 195, 700]

ax.plot(data\_x1, data\_y1)

ax.plot(data\_x2, data\_y2)

plt.show()



atau seperti ini :

data\_x1 = [1, 2, 3, 4]

data\_y1 = [10, 20, 25, 30]

data\_x2 = [1, 2, 3, 4, 5]

data\_y2 = [20, 25, 20, 40, 50]

data\_x3 = data\_x2

data\_y3 = [50, 10, 30, 15, 30]

data\_x4 = data\_x2

data\_y4 = [10, 40, 24, 30, 29]

fig, ax = plt.subplots()

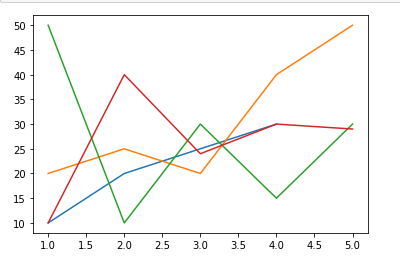
ax.plot(data\_x1, data\_y1)

ax.plot(data\_x2, data\_y2)

ax.plot(data\_x3, data\_y3)

ax.plot(data\_x4, data\_y4)

plt.show()



Terdapat berbagai kasus yang apabila kita memplot data dalam satu axis, dapat membuat data visualisasi kita menjadi berantakan sehingga kita susah u/ menemukan pola.

Pada kasus pertama, perbedaan yang besar nilai y antara data pertama dan kedua membuat data pertama menjadi tidak jelas visualisasinya, tidak jelas terlihat pola grafiknya.

Pada kasus kedua banyaknya plot data membuat visualisasi menjadi berantakan tak karuan. hal ini dapat mengganggu proses analisis dalam menemukan pola.

Oleh karena itu, solusinya adalah membuat multiple axis, dan meletakan data di axis-axis yang berbeda. hal ini bisa dilakukan dengan metode yang sama yaitu subplots().

Metode subplots ini memiliki argumen untuk mengatur jumlah axis yang dihasilkan, argument ini adalah nrows dan ncols. tidak memberikan nilai pada argument ini di contoh-contoh sebelumnya karena secara default, rows dan cols bernilai satu.

Untuk menghasilkan multiple axis, bisa di lakukan dengan memberikan nilai nrows dan ncols lebih dari satu, seperti ini :

fig, ax = plt.subplots(nrows=3, ncols=2)

print('Figure ini memiliki {0} axis di baris dan {1} axis di column'.format(ax.shape[0], ax.shape[1]))

print(ax)

===========================

Figure ini memiliki 3 axis di baris dan 2 axis di column

[[<matplotlib.axes.\_subplots.AxesSubplot object at 0x0000024384AD4248>

<matplotlib.axes.\_subplots.AxesSubplot object at 0x00000243849B3A88>]

[<matplotlib.axes.\_subplots.AxesSubplot object at 0x0000024384A2B708>

<matplotlib.axes.\_subplots.AxesSubplot object at 0x0000024384A23908>]

[<matplotlib.axes.\_subplots.AxesSubplot object at 0x0000024384981D88>

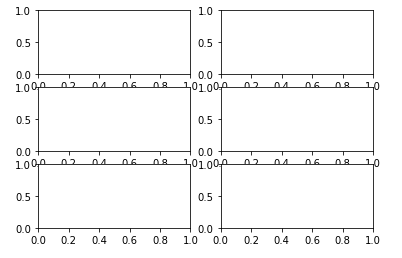
<matplotlib.axes.\_subplots.AxesSubplot object at 0x000002438494E788>]]

Coba perhatikan dengan attribute shape didapatkan bahwa dimensi dari objek axis adalah 3, 2, artinya axis memiliki 3 rows dan 2 columns. dan ketika variable ax di print. dihasilkan sebuah nested list berisi 3 elemen list didlam element list terluar, dan dalam setiap list di dalam list terdapat objek axis berjumlah 2. ini artinya adalah bahwa ax[0] akan menghasilkan objek list berisi objek axis di row pertama. dan setereusnya.

Untuk lebih jelasnya, mari lihat bentuk dari figure ini :

fig, ax = plt.subplots(nrows=3, ncols=2)

plt.show()



Sekarang mari coba membuat visualisasi data 4 plots dengan multiple axis.

import matplotlib.pyplot as plt

data\_x1 = [1, 2, 3, 4]

data\_y1 = [10, 20, 25, 30]

data\_x2 = [1, 2, 3, 4, 5]

data\_y2 = [20, 25, 20, 40, 50]

data\_x3 = data\_x2

data\_y3 = [50, 10, 30, 15, 30]

data\_x4 = data\_x2

data\_y4 = [10, 40, 24, 30, 29]

fig, ax = plt.subplots(nrows=2, ncols=2)

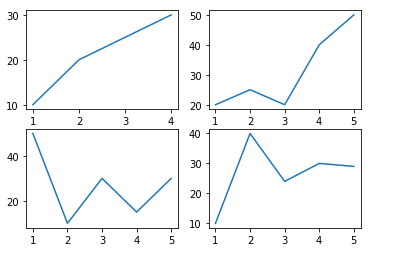
ax[0][0].plot(data\_x1, data\_y1)

ax[0][1].plot(data\_x2, data\_y2)

ax[1][0].plot(data\_x3, data\_y3)

ax[1][1].plot(data\_x4, data\_y4)

plt.show()



Coba perhatikan bahwa axis-y di setiap axis mempunyai interval yang berbeda-beda, begitu pula axis-x nya pada axis baris pertama column pertama hanya pada interval 1-4, sedangkan yang lainnya 1-5. Visualisasi ini bisa di improve agar mendapatkan interval axis yang sama dengan menambahkan argument sharex dan sharey terhadap method subplots. seperti ini :

import matplotlib.pyplot as plt

data\_x1 = [1, 2, 3, 4]

data\_y1 = [10, 20, 25, 30]

data\_x2 = [1, 2, 3, 4, 5]

data\_y2 = [20, 25, 20, 40, 50]

data\_x3 = data\_x2

data\_y3 = [50, 10, 30, 15, 30]

data\_x4 = data\_x2

data\_y4 = [10, 40, 24, 30, 29]

fig, ax = plt.subplots(nrows=2, ncols=2, sharex=True, sharey=True)

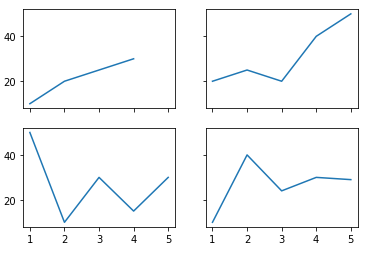
ax[0][0].plot(data\_x1, data\_y1)

ax[0][1].plot(data\_x2, data\_y2)

ax[1][0].plot(data\_x3, data\_y3)

ax[1][1].plot(data\_x4, data\_y4)

plt.show()



## **Video Pembelajaran**

* <https://youtu.be/_Kx4qT0sgGM> (Membuat Beberapa Axes)
* <https://youtu.be/f9jhv8-WvuY> (Axes di Dalam Axes)