

# BAB 1

---

## BERKENALAN DENGAN PYTHON

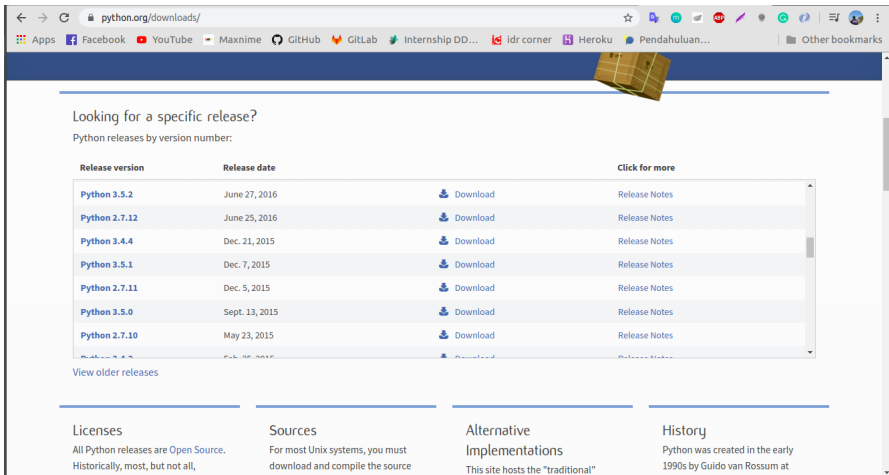
---

### 1.1 Instalasi Python

Tentunya jika ingin bisa menggunakan python, kita perlu memasang terlebih dahulu di komputer PC kita. Python sendiri juga bisa dipasang di berbagai macam sistem operasi seperti Linux, Windows dan Mac OS.

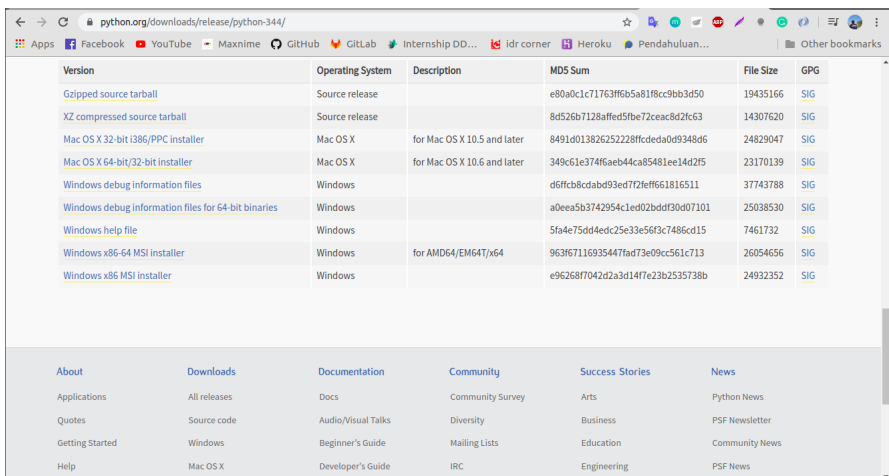
#### 1.1.1 Windows

1. Download terlebih dahulu installer pythonnya di <https://www.python.org/downloads/>. Disini saya memilih python versi 3.4.



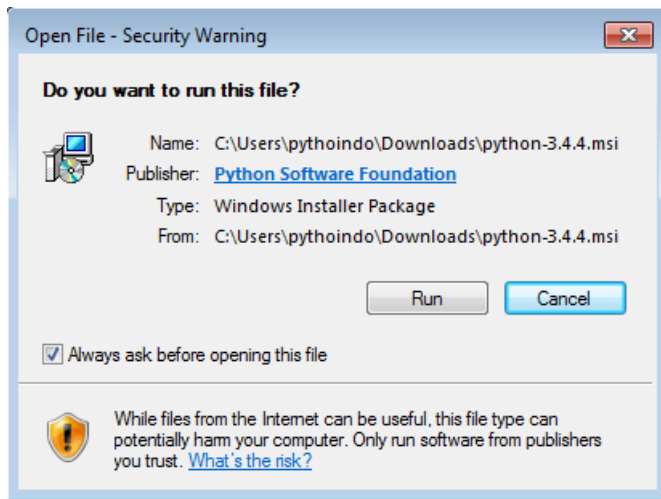
**Gambar 1.1** Pilih Python

Setelah itu saya memilih versi x86 atau 32 bit. Tapi anda bebas mau memilih 32 bit atau 64 bit.



**Gambar 1.2** Download Python

- Setelah selesai mendownload file installer tersebut, buka lokasi file installer tersebut dan tekan 2 kali untuk membukanya.



**Gambar 1.3** Run Installer

Setelah muncul kotak dialog, tekan tombol run untuk menjalankan file tersebut.

3. Lalu tunggu beberapa saat, akan muncul kotak dialog untuk memilih user siapa saja yang bisa akses instalasi tersebut. Disini saya memilih install for all user.



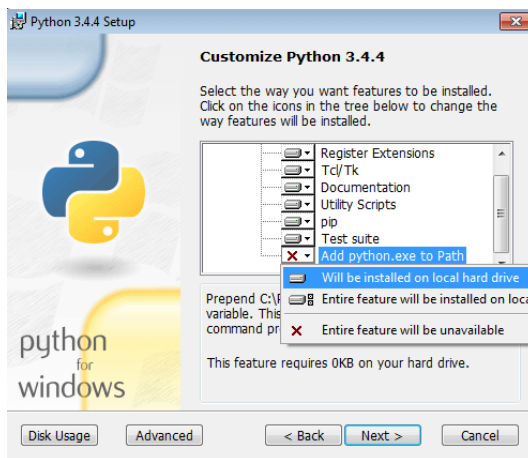
**Gambar 1.4** Pilih User

- Setelah itu, akan muncul kotak dialog tempat instalasi yang akan kita tem-  
patkan.



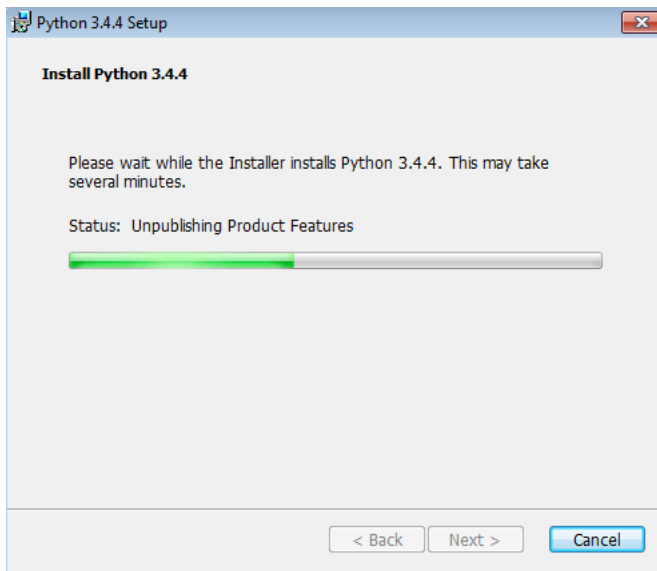
**Gambar 1.5** Pilih Location

- Setelah itu, tambahkan environment python dengan menekan pilihan **Add python.exe to Path**.



**Gambar 1.6** Add Environment

- Setelah itu, tunggu proses instalasi hingga selesai.



**Gambar 1.7** Proses Instalasi

## 7. Proses instalasi selesai.



**Gambar 1.8** Instalasi Selesai

## 1.1.2 Linux

### 1.1.2.1 via apt

1. Buka terminal di linux.
2. Masuk sebagai root di terminal.

```
1 sudo apt update
2 sudo apt install software-properties-common
3
```

**Listing 1.1** Root terminal

Penjelasan pada 1.1, pada baris 1 menjelaskan untuk mengupdate package yang ada serta baris 2 untuk melakukan instalasi prerequisites.

3. Tambahkan deadsnakes PPA ke dalam source list sistem.

```
1 sudo add-apt-repository ppa:deadsnakes/ppa
2
```

**Listing 1.2** Tambah Source

Ketika menjalankan perintah 1.2, pasti nanti diminta untuk tekan **enter** untuk melanjutkan instalasi. Maka tekan saja **enter**.

4. Setelah itu, ketik **sudo apt install python3.7** untuk melakukan instalasi python.
5. Setelah selesai melakukan instalasi, perlu dilakukan pengecekan apakah instalasi tersebut sukses atau tidak. Untuk mengecek ketik **python --version**.

```
(base) newbie@newbie:~$ python --version
Python 3.7.4
(base) newbie@newbie:~$
```

**Gambar 1.9** Cek Instalasi

### 1.1.2.2 from source

1. Buka terminal.
2. Masuk sebagai root di terminal

```
1 sudo apt update
2 sudo apt install build-essential zlib1g-dev libncurses5-
  dev libgdbm-dev libnss3-dev libssl-dev libreadline-dev libffi
  -dev wget
3
```

**Listing 1.3** root

Pada syntax 1.3, pada baris 1 untuk mengupdate list package yang ada, Sedangkan pada baris 2 untuk menginstal package yang dibutuhkan.

3. Download python versi 3.7.4 menggunakan wget.

```
1 wget https://www.python.org/ftp/python/3.7.4/Python-3.7.4.tgz
2
```

4. Ekstrak hasil download.

```
1 tar -xf Python-3.7.4.tgz
2
```

5. Masuk ke direktori hasil ekstraksi dan melakukan konfigurasi.

```
1 ./configure --enable-optimizations
2
```

Konfigurasi disini dijalankan untuk mengecek apakah semua dependency yang dibutuhkan sudah terdapat dalam sistem anda saat ini.

6. Memproses pembangunan python.

```
1 make -j 8
2
```

Disini, merupakan tahap untuk membuild python tersebut. Untuk mempercepat build, maka perlu sebuah modifikasi dengan mengetik **-j** yang relevan dengan jumlah core dari processor anda. Untuk dapat melihat jumlah core dari processor anda, cukup ketika **nproc**.

7. install binary python

```
1 sudo make altinstall
2
```

Setelah build pythonnya selesai, perlu menginstall binary python.

8. Install python sukses, dan untuk mengecek dengan cara berikut.

```
1 python --version
2
```

### 1.1.3 MacOS

## 1.2 Sejarah Python

Pada tahun 1990, Guido van Rossum mengembangkan python di CWI, Amsterdam. Bahasa ini merupakan versi lanjut dari bahasa pemrograman ABC. Versi terakhir python yang dikeluarkan oleh CWI ialah versi 1.2.

Lalu tahun 1995, Guido berpindah dari CWI ke CNRI sambil melanjutkan proses pengembangan python. Versi python terakhir dikeluarkan adalah versi 1.6. Tahun 2000, Guido van Rossum dan tim inti pengembangan python berpindah dari CNRI ke BeOpen.com yang merupakan perusahaan komersial dan telah membentuk BeOpen PythonLabs. Dan BeOpen pun mengeluarkan versi python yang baru yaitu versi 2.0. Setelah Guido dan tim di BeOpen mengeluarkan python versi 2.0, mereka berpindah kembali ke DigitalCreations.

Hingga saat ini, pengembangan python masih terus dilakukan oleh sekumpulan pemrogram yang di koordinir oleh Guido van Rossum dan Python Software Foundation. Python Software Foundation adalah sebuah organisasi non-profit yang dibentuk sebagai hak cipta intelektual atas python sejak python versi 2.1 dan dengan untuk mencegah python dimiliki oleh perusahaan komersial. Saat ini, proses distribusi python sudah mencapai versi 2.6.1 dan versi 3.0.

Guido memilih nama python karena kecintaan Guido van Rossum terhadap sebuah acara televisi bernama Monty Python's Flying Circus.

### 1.3 Pengenalan Python

Python merupakan bahasa pemrograman yang dapat melakukan eksekusi sejumlah instruksi multiguna secara langsung (interpretatif) dengan berorientasi pada objek serta menggunakan semantik dinamis untuk memberikan tingkat keterbacaan kode atau syntax. Sebagian besar mengartikan python sebagai bahasa dengan tingkat kemampuan tinggi, menggabungkan kapabilitas, dan sintaks kode yang sangat jelas dan dilengkapi oleh fungsionalitas dari pustaka dasar yang sangat besar dan komprehensif. Walaupun python ini, digolongkan sebagai bahasa pemrograman tingkat tinggi, namun tetap Python dirancang sedemikian rupa supaya mudah dipahami serta dipelajari.

Python juga dapat berjalan di banyak platform seperti Mac, Linux dan Windows dll. Python bersifat *open source*, sehingga masih banyak orang yang berkontribusi untuk mengembangkan dimana yang hak kekayaan intelektual dipegang oleh PSF. Bahasa Python didukung oleh *library library* yang didalamnya menyediakan fungsi analisis data dan fungsi *machine learning*, *data preprocessing tools*, serta visualisasi data. Hal ini membuat Python menjadi bahasa pemrograman yang populer pada bidang *data science* dan analisis.



**Gambar 1.10** Logo Python



## 1.4 Mengapa harus Python

Alasan mengapa python adalah salah satu bahasa pemrograman yang harus dipelajari adalah sebagai berikut :

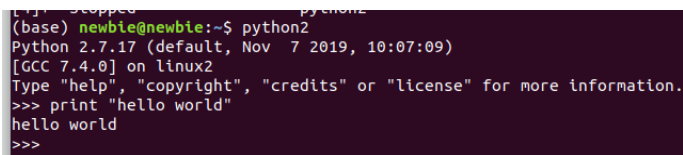
1. Python merupakan bahasa yang mudah dipelajari serta mudah digunakan.
2. Python juga merupakan bahasa yang lebih efisien dibandingkan dengan bahasa pemrograman lain. Contohnya jika dalam bahasa lain bisa sampai 5 baris, maka dengan python cukup 1 baris saja untuk menjalankan perintah tersebut.
3. Python merupakan bahasa multifungsi, dimana python bisa diterapkan dimana saja mulai dari pemrosesan data / teks, membuat website, membuat program jaringan, robotika, sampai dengan kecerdasan buatan.
4. Python juga memiliki dukungan pustaka yang cukup banyak.
5. Python juga bisa melakukan integrasi dengan bahasa pemrograman lainnya.

## 1.5 Mengapa Bisa Muncul Python

### 1.6 Perbedaan Python 2 dan Python 3

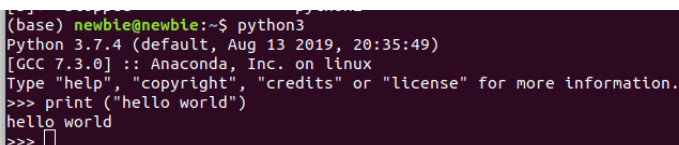
#### 1.6.1 Perintah Print

Dalam python 2, perintah print tidak harus menggunakan tanda kurung, sedangkan dalam python 3, perintah print harus menggunakan tanda kurung.



```
(base) newbie@newbie:~$ python2
Python 2.7.17 (default, Nov 7 2019, 10:07:09)
[GCC 7.4.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> print 'hello world'
hello world
>>>
```

Gambar 1.11 Python 2



```
(base) newbie@newbie:~$ python3
Python 3.7.4 (default, Aug 13 2019, 20:35:49)
[GCC 7.3.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print ('hello world')
hello world
>>> 
```

Gambar 1.12 Python 3

## 1.7 Karakteristik Python

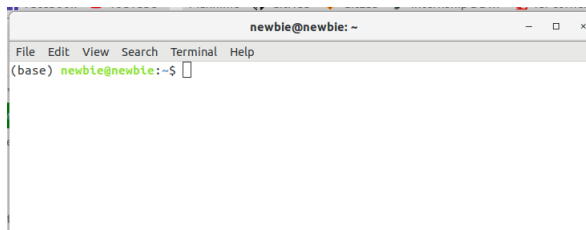
## 1.8 Aturan Dasar

## 1.9 Cara Penggunaan

### 1.9.1 Cara Menjalankan

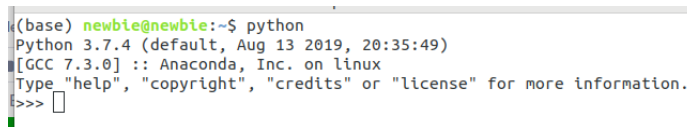
#### 1.9.1.1 Linux

1. Buka terminal linux



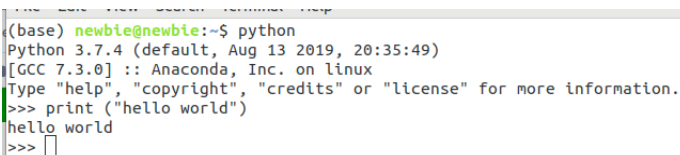
**Gambar 1.13** Terminal

2. Ketik Python di terminal tersebut. Itu digunakan untuk masuk ke dalam sheel python.



**Gambar 1.14** Penggunaan Perintah Python

3. Lalu, tuliskan kode `print("hello world")`. Jika sudah tekan enter.



**Gambar 1.15** Penggunaan Perintah print

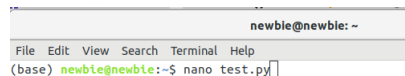
4. Untuk keluar dari sheel python, ketik `exit()`.

```
(base) newbie@newbie:~$ python
Python 3.7.4 (default, Aug 13 2019, 20:35:49)
[GCC 7.3.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print("hello world")
hello world
>>> exit()
(base) newbie@newbie:~$
```

**Gambar 1.16** Penggunaan Perintah exit

Atau bisa menggunakan cara ini, seperti berikut :

1. Menggunakan text editor, seperti nano. Untuk penjelasan tentang nano sendiri akan dijelaskan setelah point ini.



```
newbie@newbie: ~
File Edit View Search Terminal Help
(base) newbie@newbie:~$ nano test.py
```

**Gambar 1.17** Editor Nano

Pada gambar 1.17, digunakan untuk membuat file baru dengan nama test.py.

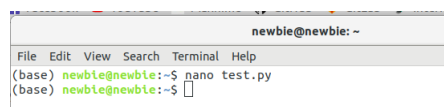
2. Lalu simpan file tersebut dengan menekan Ctrl + O, setelah itu akan muncul konfirmasi. Tekan enter saja.



```
newbie@newbie: ~
File Edit View Search Terminal Help
GNU nano 2.9.3 test.py
print("hello world")
File Name to Write: test.py
^G Get Help ^O DOS Format ^A Append ^B Backup File
^C Cancel ^H Mac Format ^R Prepend ^X To Files
```

**Gambar 1.18** Save File

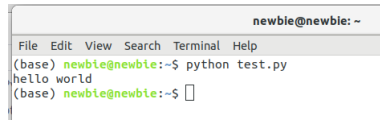
3. Untuk keluar dari editor nano tersebut, tekan Ctrl + X.



```
newbie@newbie: ~
File Edit View Search Terminal Help
(base) newbie@newbie:~$ nano test.py
(base) newbie@newbie:~$
```

**Gambar 1.19** Exit Editor

4. Untuk menjalankan file py tersebut. ketik python namafile.py.



```

newbie@newbie: ~
File Edit View Search Terminal Help
(base) newbie@newbie:~$ python test.py
hello world
(base) newbie@newbie:~$ 

```

**Gambar 1.20** Running File

## 1.9.2 Komentar

Komentar atau comment adalah kode yang berada dalam syntax python yang tidak dieksekusi atau tidak dieksekusi oleh mesin. Komentar biasanya digunakan untuk menandai atau memberikan suatu keterangan pada syntax python yang ada.

Komentar sering digunakan untuk memberikan penjelasan kepada orang lain terhadap syntax yang ada atau bisa digunakan untuk mengingatkan kepada seorang programmer jika ada yang ingin diubah dari syntax tersebut.

Untuk memberikan komentar, cukup dengan memberikan tanda (#) yang diikuti dengan isi komentar tersebut. Berikut Contoh penggunaan komentar di python.

```

1 #ini untuk menampilkan tulisan hello world
2 print("hello world")

```

**Listing 1.4** Penggunaan Komentar

## 1.9.3 Tipe Data

Tipe data merupakan media atau memori pada komputer untuk menampung berbagai informasi sesuai jenisnya.

Python juga memiliki berbagai tipe data yang cukup unik jika dibandingkan dengan bahasa pemrograman lainnya.

Berikut beberapa tipe data dalam python.

1. Boolean, tipe data ini digunakan untuk menentukan dalam pengambilan keputusan. Jika benar atau True akan bernilai 1 dan jika salah atau False akan bernilai 0.
2. String, tipe data ini digunakan untuk menyatakan karakter / kalimat. Dan tipe data ini harus menggunakan tanda “atau ‘ untuk mengapit nilai String tersebut. Contoh implementasinya seperti berikut.

```

1 print("hello world")
2 print('hello world')
3

```

**Listing 1.5** Tipe Data String

```
(base) newbie@newbie:~$ python
Python 3.7.4 (default, Aug 13 2019, 20:35:49)
[GCC 7.3.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print("hello world")
hello world
>>> print('hello world')
hello world
>>> 
```

**Gambar 1.21** String Data

3. Integer, tipe data ini untuk menyatakan bilangan bulat. Contoh implementasinya bisa dilihat seperti berikut.

```
1 print(20)
2 
```

**Listing 1.6** Tipe Data Integer

```
(base) newbie@newbie:~$ python
Python 3.7.4 (default, Aug 13 2019, 20:35:49)
[GCC 7.3.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print(20)
20
>>> 
```

**Gambar 1.22** Integer Data

4. Float, tipe data ini untuk menyatakan bilangan yang memiliki koma. Contoh implementasinya bisa dilihat seperti berikut.

```
1 print(3.14)
2 
```

**Listing 1.7** Tipe Data Float

```
(base) newbie@newbie:~$ python
Python 3.7.4 (default, Aug 13 2019, 20:35:49)
[GCC 7.3.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print(3.14)
3.14
>>> 
```

**Gambar 1.23** Float Data

5. List, tipe data ini untuk menyimpan berbagai jenis tipe data dan isinya bisa diubah-ubah. Contoh implementasi bisa dilihat seperti berikut.

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

**Listing 1.8** Tipe Data List

```
(base) newbie@newbie:~$ python
Python 3.7.4 (default, Aug 13 2019, 20:35:49)
[GCC 7.3.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print([1,2,3,4,5])
[1, 2, 3, 4, 5]
>>>
```

**Gambar 1.24** List Data

6. Tuple, tipe data ini untuk menyimpan berbagai jenis tipe data dan isinya tidak bisa diubah-ubah seperti list. Contoh implementasi bisa dilihat seperti berikut.

```
1 print((1,2,3,4,5))
2
```

**Listing 1.9** Tipe Data tuple

```
(base) newbie@newbie:~$ python
Python 3.7.4 (default, Aug 13 2019, 20:35:49)
[GCC 7.3.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print([1,2,3,4,5])
[1, 2, 3, 4, 5]
>>> print((1,2,3,4,5))
(1, 2, 3, 4, 5)
>>>
```

**Gambar 1.25** Tuple Data

7. Dictionary, tipe data ini untuk menyimpan berbagai tipe data berupa pasangan petunjuk dan nilainya. Contoh implementasi bisa dilihat seperti berikut.

```
1 print({"nama": "Budi", 'umur': 20})
2
```

**Listing 1.10** Tipe Data Dictionary

```
(base) newbie@newbie:~$ python
Python 3.7.4 (default, Aug 13 2019, 20:35:49)
[GCC 7.3.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print({"nama": "Budi", 'umur': 20})
{'nama': 'Budi', 'umur': 20}
>>>
```

**Gambar 1.26** Dictionary Data

## 1.9.4 Variabel

Variabel merupakan lokasi memori yang dicadangkan untuk menyimpan nilai. Variabel menyimpan data yang dilakukan selama program dieksekusi, yang nantinya isi dari variabel bisa dirubah-rubah suatu saat.

Variabel dalam pemrograman python, bersifat dinamis. Artinya tipe data dalam variabel tersebut tidak perlu di deklarasikan dan isi variabel tersebut bisa dirubah ketika menjalankan program.

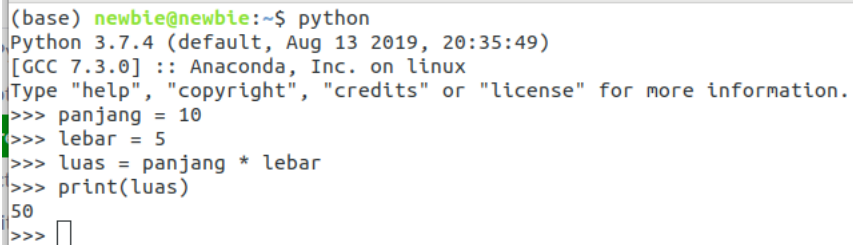
Beberapa aturan dalam penulisan variabel di pemrograman python, sebagai berikut:

1. Karakter utama harus berupa huruf atau garis bawah
2. karakter selanjutnya boleh huruf, angka maupun garis bawah.
3. karakter pada nama variabel bersifat case-sensitif, artinya huruf kecil dan huruf besar memiliki makna yang berbeda. Contoh. variabel **contoh** dan **Contoh** merupakan variabel yang berbeda.

Untuk pembuatan variabel di pemrograman python sangat mudah, cukup ketik nama variabel dengan diikuti tanda `=` dan diikuti dengan isi nilai variabel tersebut.

```
1 panjang = 10
2 lebar = 5
3 luas = panjang * lebar
4 print (luas)
```

**Listing 1.11** Penggunaan Variabel



```
(base) newbie@newbie:~$ python
Python 3.7.4 (default, Aug 13 2019, 20:35:49)
[GCC 7.3.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> panjang = 10
>>> lebar = 5
>>> luas = panjang * lebar
>>> print(luas)
50
>>> □
```

**Gambar 1.27** Variabel

## 1.9.5 Looping

Looping adalah sebuah kondisi dalam bahasa pemrograman yang dieksekusi secara berurutan. Jika pernyataan pertama dijalankan, maka akan diikuti oleh pernyataan yang kedua dan seterusnya. Tetapi terkadang ada dalam suatu kondisi tertentu, kita harus menulis banyak kode. Tentunya jika itu semua dilakukan secara manual akan tidak bisa memberikan performansi yang baik dalam pemrograman tersebut. Oleh karena itu, muncullah looping atau pengolahan.

Pengulangan dalam pemrograman python terbagi menjadi 3 bagian, seperti :

1. While loop  
While loop disini akan dijalankan selama kondisi dalam pemrograman tersebut masih bernilai benar atau True. Contoh implementasi while loop.

```
1     count = 0
2     while (count < 9):
3         print 'The count is:', count
4         count = count + 1
5
6     print ("Good bye!")
```

**Listing 1.12** Penggunaan While Loop

Pada syntax 1.12, dijelaskan pada baris pertama terdapat inisiasi variabel. Lalu baris kedua untuk melakukan looping berdasarkan variabel di baris pertama dengan kondisi jika variabel tersebut akan dilooping sampai dibawah 9. Lalu baris ketiga digunakan untuk menampilkan ini looping ke berapa. Lalu pada baris keempat untuk inisiasi variabel baris pertama dengan kondisi variabel tersebut ditambah 1.

## 2. For loop

Pengulangan dengan menggunakan for memiliki kemampuan untuk mengulang atau me looping item dari urutan yang ada seperti string ata list. Contoh implementasi for loop:

```
1     buah = ["nanas", "apel", "jeruk"]
2     for makanan in buah:
3         print "Saya suka makan", makanan
```

**Listing 1.13** Penggunaan For Loop

Pada syntax 1.13, penjelasan untuk baris pertama ialah untuk inisiasi variabel dengan tipe data list. Lalu pada baris kedua digunakan untuk looping dengan variabel makanan yang isinya di ambil dari variabel di baris pertama. Lalu pada baris ketiga, untuk menampilkan hasil looping.

## 1.9.6 Fungsi

Fungsi dalam pemrograman python merupakan sebuah blok kode yang terorganisir dan dapat digunakan kembali untuk suatu action tertentu di suatu saat. Penggunaan fungsi dapat memberikan tingkat modularitas yang baik terhadap program tersebut serta tingkat penggunaan kode yang tinggi.

Dalam pendeklarasian fungsi dalam pemrograman python, terdapat beberapa aturan yang harus dilakukan, seperti berikut.

1. Pembuatan fungsi dimulai dengan kata **def** lalu diikuti dengan nama fungsi serta tanda kurung ().
2. Setiap parameter masukan harus dimasukkan kedalam tanda kurung (). Dan bisa di atur juga nilai dari parameter tersebut.
3. Setiap fungsi blok kode harus dimulai dengan tanda (:) dan indentasi.
4. Setiap fungsi blok kode harus memiliki pengembalian nilai.



Tentunya kita semua, jika hanya membaca teori mungkin masih kebingungan. Maka dari itu, langsung aja ke contoh implementasinya seperti berikut.

```
1  def printme( str ) :  
2  "This prints a passed string into this function"  
3  print ( str )  
4  return
```

**Listing 1.14** Fungsi Python

Pada syntax 1.14, baris pertama menjelaskan tentang pendefinisian nama fungsi. Lalu baris kedua menjelaskan tentang isi string. Lalu pada baris ketiga menjelaskan untuk menampilkan isi string tersebut. Pada baris keempat menjelaskan untuk fungsi return ketika fungsi tersebut dipanggil.

### 1.9.7 Modul

Modul merupakan sebuah file py yang berisikan sekumpulan kode python. Sebuah file py bisa disebut modul.

Penerapan modul ini biasanya disebut konsep OOP (Object Orientied Programming) dalam pemrograman python. Karena pada dasarnya ini digunakan untuk membagi file-file program yang besar menjadi lebih kecil supaya mudah dalam manage dan diorganisir. Sehingga nantinya bisa di reusable, artinya kode-kode tersebut bisa di gunakan kapan saja.

Contoh penerapan modul tersebut seperti berikut. Saya membuat file dengan nama test.py dengan isi file berikut.

```
1  def jumlah(a, b):  
2  """Fungsi ini menambahkan dua bilangan  
3  dan mengembalikan hasilnya"""  
4  result = a + b  
5  return result
```

**Listing 1.15** Modul

Lalu kita ketik di command line python yang isi kodenya seperti berikut.

```
1  import test  
2  test.jumlah(5,6)
```

**Listing 1.16** Import modul

Pada syntax 1.16, pada baris pertama digunakan untuk mengimport file test.py. Lalu pada baris kedua digunakan untuk memanggil nama fungsi di file test.py.

Dalam python juga, kita bisa menggunakan library-library yang telah disediakan oleh python itu sendiri. Untuk cara import library atau modul ada beberapa cara, seperti berikut.

### 1. cara import standard

```
1 import nama_module
2
```

**Listing 1.17** Import modul

```
(base) newbie@newbie:~$ python
Python 3.7.4 (default, Aug 13 2019, 20:35:49)
[GCC 7.3.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import os
>>> □
```

**Gambar 1.28** Import Module

### 2. cara import dengan alias

```
1 import nama_module as alias
2
```

**Listing 1.18** Import modul

```
(base) newbie@newbie:~$ python
Python 3.7.4 (default, Aug 13 2019, 20:35:49)
[GCC 7.3.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import os as s
>>> □
```

**Gambar 1.29** Import Module Alias

### 3. cara import namun hanya mengambil sebagian dari library tersebut.

```
1 from nama_module import something
2
```

**Listing 1.19** Import module Sebagian

```
(base) newbie@newbie:~$ python
Python 3.7.4 (default, Aug 13 2019, 20:35:49)
[GCC 7.3.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from sklearn.feature_extraction.text import TfidfVectorizer
>>> □
```

**Gambar 1.30** Import Module Sebagian

## 1.10 Contoh Kasus

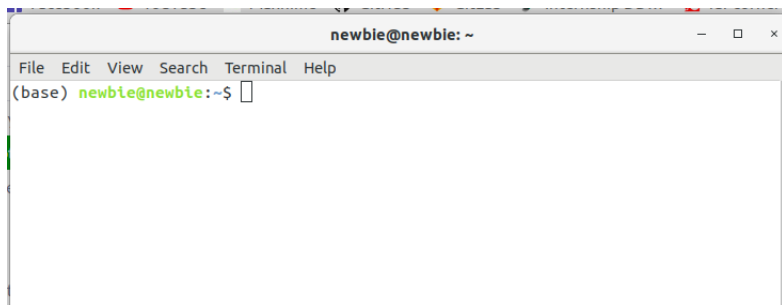
### 1.11 Instalasi Pip

#### 1.11.1 Windows

#### 1.11.2 Linux

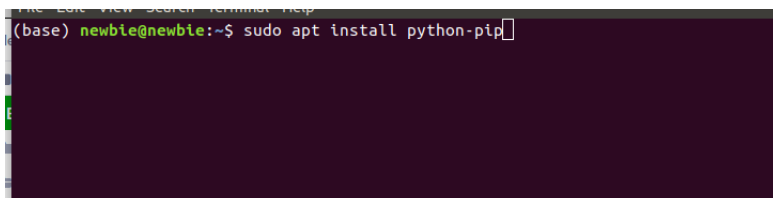
Dalam linux, cukup mudah untuk melakukan instalasi pip.

1. Buka Terminal linux.



**Gambar 1.31** Terminal

2. Ketik **sudo apt install python-pip** untuk python2  
Sedangkan untuk python3, menggunakan perintah **sudo apt install python3-pip**



**Gambar 1.32** Pip

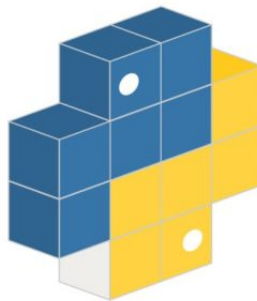
```
File Edit View Search Terminal Help
(base) newbie@newbie:~$ sudo apt install python3-pip
```

**Gambar 1.33** Pip 3

### 1.11.3 MacOS

### 1.12 Pip

PIP adalah sebuah program di Python untuk mengelola, install, mencari dan uninstall package, jika diibaratkan Pip dalam Android itu seperti Google Play Store. Dimana kita mencari, menginstall dan uninstall apapun yang kita inginkan.



**Gambar 1.34** Logo Pip Python

### 1.13 Contoh Penggunaan

Penerapan pip dalam python sangat memberikan kemudahan, dimana kita perlu mendownload source module dengan mencari di internet, menambahkan site-package dari module tersebut yang sudah disediakan oleh package tersebut. Untuk cara penggunaan dari pip itu sendiri menggunakan command line.

**1.13.0.1 Install Package** Untuk melakukan instalasi package, cukup ketik perintah.

```
1 pip install \textbf{nama_package}  
2 pip install \textbf{nama_package}==version_of_package
```

**Listing 1.20** install package

Pada baris 1, itu adalah perintah install package dimana mengambil version package terakhir. Sedangkan baris 2, perintah install package dengan menetapkan version package yang dibutuhkan.

**1.13.0.2 Menampilkan Package** Untuk menampilkan informasi package yang sudah terinstall, cukup menggunakan perintah.

```
1 pip list
```

**Listing 1.21** List package

Sedangkan untuk menampilkan informasi package, version package, lokasi dan dependencies dari package tersebut, cukup menggunakan perintah.

```
1 pip show \textbf{nama_package}
```

**Listing 1.22** Show package

**1.13.0.3 Uninstall Package** Untuk melakukan uninstall package, cukup menggunakan perintah.

```
1 pip uninstall \textbf{nama_package}
```

**Listing 1.23** uninstall package

## 1.14 Ekstensi File

### 1.14.1 .py

File py adalah salah satu file program atau skrip yang ditulis menggunakan python, bahasa pemrograman berorientasi objek. File dapat dibuat dan diedit menggunakan teks editor, tetapi untuk menjalankannya membutuhkan bahasa Python. File PY sering digunakan untuk pemrograman server web dan sistem komputer administrasi lainnya.

### 1.14.2 .ipynb

File IPYNB adalah dokumen notebook yang digunakan oleh Jupyter Notebook, sebuah lingkungan komputasi interaktif yang dirancang untuk membantu para *scientists* bekerja menggunakan bahasa Python dan data mereka. File ini berisi semua konten dari sesi aplikasi web Notebook Jupyter, yang mencakup input dan output perhitungan, matematika, gambar, dan teks penjelasan. selain itu, file IPYNB dapat diekspor ke format .HTML, .PDF, reStructuredText, dan LaTeX. IPYNB menggabungkan 2 komponen, yaitu:

1. Aplikasi web: alat berbasis browser untuk penulisan dokumen interaktif yang menggabungkan teks penjelasan, matematika, perhitungan dan output media yang kaya.
2. Dokumen buku catatan: representasi semua konten yang terlihat dalam aplikasi web, termasuk input dan output dari perhitungan, teks penjelasan, matematika, gambar, dan representasi objek yang kaya media.

### 1.14.3 Konversi File

Sebelum melakukan konversi file, perlu ada beberapa hal yang harus dilakukan seperti menginstal library ipython dan nbconvert.

**1.14.3.1 .py to .ipynb** Cukup menjalankan kode **ipython nbconvert to script abc.py**



**Gambar 1.35** Convert to py

**1.14.3.2 .ipynb to .py** Cukup menjalankan ode **ipython nbconvert to script abc.ipynb**

## 1.15 Mengapa harus ipynb

IPYNB lebih unggul dalam pemrograman yang disebut *literate programming*. *Literate programming* merupakan gaya pengembangan perangkat lunak dipelopori oleh ilmuwan komputer Stanford, Donald Knuth. Jenis pemrograman ini menekankan pendekatan prosa pertama di mana eksposisi dengan teks *human-friendly* diselingi dengan blok kode.

## BAB 2

---

# DATA SCIENCE

---

### 2.1 Apa itu Big Data

Pembuatan data terjadi pada tingkat rekor. Pada 2010, dunia menghasilkan lebih dari 1ZB data; pada 2014, kita akan menghasilkan 7ZB setahun. Sebagian besar ledakan data ini adalah hasil dari peningkatan dramatis dalam perangkat yang terletak di pinggiran jaringan termasuk sensor yang tertanam, *smartphone*, dan komputer tablet. Semua data ini menciptakan peluang baru untuk "mengeksktraksi lebih banyak nilai" dalam genomika manusia, perawatan kesehatan, minyak dan gas, pencarian, pengawasan, keuangan, dan banyak bidang lainnya. Kita memasuki zaman "*Big Data*." [1]

#### 2.1.1 Pengertian

Sebelum memahami '*Big Data*', perlu terlebih dahulu tahu apa itu data. Data merupakan jumlah, karakter, atau simbol tempat operasi dilakukan oleh komputer, yang dapat disimpan dan dikirim dalam bentuk sinyal listrik dan direkam pada media perekaman magnetik, optik, atau mekanis. *Big Data* juga merupakan data tetapi dengan ukuran yang sangat besar. *Big Data* merupakan istilah yang digunakan un-

tuk mendeskripsikan kumpulan data yang berukuran sangat besar namun tumbuh secara eksponensial seiring waktu. Singkatnya, data tersebut sangat besar dan kompleks sehingga tidak ada alat manajemen data tradisional yang dapat menyimpan atau memprosesnya secara efisien.

*Big Data* adalah tentang tantangan yang semakin besar yang dihadapi organisasi ketika mereka berhadapan dengan sumber data atau informasi yang besar dan berkembang pesat yang juga menghadirkan berbagai analisis kompleks dan masalah penggunaan. Ini dapat mencakup [1]:

1. Memiliki infrastruktur komputasi yang dapat menelan, memvalidasi, dan menganalisis volume (ukuran dan / atau tingkat) data yang tinggi.
2. Menilai data campuran (terstruktur dan tidak terstruktur) dari berbagai sumber.
3. Berurusan dengan konten yang tidak dapat diprediksi tanpa skema atau struktur yang jelas.
4. Mengaktifkan pengumpulan, analisis, dan jawaban waktu-nyata-dekat-waktu-nyata.

### 2.1.2 Contoh *Big Data*

Contoh dari *Big Data* misalnya pada *New York Stock Exchange* yang menghasilkan sekitar satu *terabyte* data perdagangan baru per hari. Media sosial seperti Facebook, statistik menunjukkan bahwa setiap harinya lebih dari 500 *terabyte* data baru dapat dicerna ke dalam basis data situs. Data ini terutama dihasilkan dalam hal unggahan foto dan video, pertukaran pesan, memberi komentar, dll. *Big Data* bisa datang dari berbagai sumber, seperti sistem transaksi bisnis, database pelanggan, catatan medis, log clickstream internet, aplikasi mobile, jejaring sosial, repositori penelitian ilmiah, data yang dihasilkan mesin, dan sensor data real-time yang digunakan dalam internet benda (IOT) lingkungan.

Data dapat dibiarkan dalam bentuk mentah dalam sistem data besar atau diproses menggunakan alat penambahan data atau perangkat lunak persiapan data sehingga siap untuk penggunaan analitik tertentu. Menggunakan data pelanggan sebagai contoh, berbagai cabang analitik yang dapat dilakukan dengan informasi yang ditemukan dalam set data besar meliputi yang berikut ini:

1. ***Comparative analysis.*** Ini termasuk pemeriksaan metrik perilaku pengguna dan pengamatan keterlibatan pelanggan waktu nyata untuk membandingkan produk, layanan, dan otoritas merek perusahaan dengan pesaing.
2. ***Social media listening.*** Memuat informasi tentang apa yang orang katakan di media sosial tentang bisnis atau produk tertentu yang melampaui apa yang dapat disampaikan dalam jajak pendapat atau survei. Data ini dapat digunakan untuk membantu mengidentifikasi target audiens untuk kampanye pemasaran dengan mengamati aktivitas seputar topik spesifik di berbagai sumber.



3. **Marketing analysis.** Memuat informasi yang dapat digunakan untuk membuat promosi produk baru, layanan dan inisiatif lebih informatif dan inovatif.
4. **Customer satisfaction and sentiment analysis.** Semua informasi yang dikumpulkan dapat mengungkapkan bagaimana perasaan pelanggan tentang suatu perusahaan atau merek, jika ada masalah potensial yang mungkin timbul, bagaimana loyalitas merek dapat dipertahankan dan bagaimana upaya layanan pelanggan dapat ditingkatkan.
5. Mendorong inovasi.

Penerapan big data juga dapat memberikan rekomendasi kepada anda untuk berinovasi dengan cara mempelajari hubungan sesama manusia dan kemudian menentukan cara baru guna memberikan pengetahuan baru. Dalam kasus ini, kita menggunakan data insight dari pengguna. Dari data tersebut, perlu dianalisis tren apa yang sedang naik dan apa yang sedang diinginkan oleh pengguna. Berdasarkan hasil analisis tersebut, bisa memunculkan inovasi baru. Beberapa perusahaan besar sudah menerapkan big data seperti Netflix, Procter dan Gamble yang menggunakan big data untuk menganalisa keinginan customer. Mereka menggunakan data customer untuk melihat bagaimana customer mereka menggunakan produk mereka. Berdasarkan data tersebut, mereka kemudian mengembangkan sebuah inovasi baru untuk produk atau layanan mereka. Selain itu, perusahaan P & G menggunakan data dan analytic dari berbagai channel dan sosial media.

Salah satu perusahaan e-commerce di indonesia yang menerapkan big data adalah JD.ID.

#### 6. Customer Relationship Management (CRM)

Mungkin kita sering mendengar istilah CRM. CRM biasanya digunakan oleh pebisnis untuk menjaga hubungan dengan customer bisnis tersebut. Dengan adanya data CRM tersebut bisa digunakan untuk menganalisa pelacakan penjualan, history pembelian, jenis customer, dan lain-lain.

### 2.1.3 Cara Kerja *Big Data*

*Big Data* dapat dikategorikan sebagai tidak terstruktur atau terstruktur. Data terstruktur terdiri dari informasi yang sudah dikelola oleh organisasi dalam *database* dan *spreadsheet*; sering bersifat numerik. Data yang tidak terstruktur adalah informasi yang tidak terorganisir dan tidak termasuk dalam model atau format yang ditentukan sebelumnya. Termasuk juga data yang dikumpulkan dari sumber media sosial, yang membantu institusi mengumpulkan informasi tentang kebutuhan pelanggan.

*Big Data* dapat dikumpulkan dari komentar yang dibagikan secara publik di jejaring sosial dan situs *web*, dikumpulkan secara sukarela dari elektronik dan aplikasi pribadi, melalui kuesioner, pembelian produk, dan *check-in* elektronik. Kehadiran sensor dan input lain dalam perangkat pintar memungkinkan data dikumpulkan di

berbagai situasi dan keadaan. *Big data* paling sering disimpan dalam *database* komputer dan dianalisis menggunakan perangkat lunak yang dirancang khusus untuk menangani set data yang besar dan kompleks. Banyak perusahaan perangkat lunak sebagai layanan (SaaS) mengkhususkan diri dalam mengelola jenis data yang kompleks ini.

### 2.1.4 Penggunaan *Big Data*

Analisis data melihat hubungan antara berbagai jenis data, seperti data demografis dan riwayat pembelian, untuk menentukan apakah ada korelasi. Penilaian semacam itu dapat dilakukan sendiri di dalam perusahaan atau secara eksternal oleh pihak ketiga yang berfokus pada pemrosesan data besar ke dalam format yang dapat dicerna. Bisnis sering menggunakan penilaian data besar oleh para ahli untuk mengubahnya menjadi informasi yang dapat ditindaklanjuti.

Hampir setiap departemen di perusahaan dapat memanfaatkan temuan dari analisis data, dari sumber daya manusia dan teknologi hingga pemasaran dan penjualan. Tujuan dari *big data* adalah untuk meningkatkan kecepatan di mana produk sampai ke pasar, untuk mengurangi jumlah waktu dan sumber daya yang dibutuhkan untuk mendapatkan adopsi pasar, target audiens, dan untuk memastikan bahwa pelanggan tetap puas.

### 2.1.5 Jenis *Big Data*

*Big Data* memiliki 3 jenis tipe data, yaitu :

1. *Structured*
2. *Unstructured*
3. *Semi-structured*

#### 2.1.5.1 *Structured data*

Data yang disimpan dalam baris dan kolom, sebagian besar numerik, di mana makna setiap item data didefinisikan. Jenis data ini merupakan sekitar 10% dari total data saat ini dan dapat diakses melalui sistem manajemen basis data. Contoh sumber data terstruktur (atau tradisional) termasuk register resmi yang dibuat oleh lembaga pemerintah untuk menyimpan data pada individu, perusahaan dan real estat; dan sensor di industri yang mengumpulkan data tentang proses. Saat ini, data sensor adalah salah satu area yang tumbuh cepat, khususnya sensor yang dipasang di pabrik untuk memantau pergerakan, suhu, lokasi, cahaya, getaran, tekanan, cairan dan aliran.

Setiap data yang dapat disimpan, diakses dan diproses dalam bentuk format tetap disebut sebagai data 'terstruktur'. Selama periode waktu, bakat dalam ilmu komputer telah mencapai keberhasilan yang lebih besar dalam mengembangkan teknik untuk bekerja dengan data semacam itu (di mana formatnya sudah diketahui sebelumnya) dan terdapat nilai. Namun, saat ini, kami meramalkan masalah ketika ukuran data tersebut tumbuh sangat besar, ukuran tipikal sedang populer di banyak zettabytes.

### 2.1.5.2 *Unstructured data*

Berbagai bentuk data seperti mis. teks, gambar, video, dokumen, dll. Bisa juga dalam bentuk keluhan pelanggan, kontrak, atau *email* internal. Jenis data ini menyumbang sekitar 90% dari data yang dibuat pada abad ini. Faktanya, pertumbuhan vulkanik media sosial (mis. Facebook dan Twitter), sejak pertengahan dekade terakhir, bertanggung jawab atas bagian utama dari data tidak terstruktur yang kita miliki saat ini. Data yang tidak terstruktur tidak dapat disimpan menggunakan database relasional tradisional. Menyimpan data dengan variasi dan kompleksitas seperti itu membutuhkan penggunaan sistem penyimpanan yang memadai, yang biasa disebut sebagai basis data NoSQL, seperti mis. MongoDB dan CouchDB. Pentingnya data yang tidak terstruktur terletak pada hubungan timbal balik yang tertanam yang mungkin tidak ditemukan jika jenis data lain dipertimbangkan. Apa yang membuat data yang dihasilkan di media sosial berbeda dari tipe data lainnya adalah bahwa data di media sosial memiliki selera pribadi.

Setiap data dengan bentuk atau struktur yang tidak dikenal diklasifikasikan sebagai data yang tidak terstruktur. Selain ukurannya yang besar, data yang tidak terstruktur menimbulkan banyak tantangan dalam hal pemrosesan untuk mendapatkan nilai darinya. Contoh khas data tidak terstruktur adalah sumber data heterogen yang berisi kombinasi file teks sederhana, gambar, video dll. Sekarang organisasi saat ini memiliki banyak data yang tersedia dengan mereka tetapi sayangnya, mereka tidak tahu bagaimana mendapatkan nilai dari itu karena data ini dalam bentuk mentah atau format tidak terstruktur.

### 2.1.5.3 *Semi-structured*

Data semi-terstruktur dapat berisi kedua bentuk data. Kita dapat melihat data semi-terstruktur sebagai formulir terstruktur tetapi sebenarnya tidak didefinisikan dengan mis. definisi tabel dalam DBMS relasional. Contoh data semi-terstruktur adalah data yang direpresentasikan dalam file XML.

## 2.1.6 *Tantangan Big Data*

Selain kapasitas pemrosesan dan masalah biaya, merancang arsitektur data besar adalah tantangan umum lainnya bagi pengguna. Sistem big data harus disesuaikan dengan kebutuhan khusus organisasi, sebuah usaha DIY yang mengharuskan tim TI dan pengembang aplikasi untuk mengumpulkan satu set alat dari semua teknologi yang tersedia. Menyebarkan dan mengelola sistem big data juga membutuhkan keterampilan baru dibandingkan dengan yang dimiliki oleh *database administrator (DBA)* dan pengembang yang berfokus pada perangkat lunak relasional.

Kedua masalah tersebut dapat diatasi dengan menggunakan layanan *cloud* yang dikelola, tetapi manajer TI perlu mengawasi penggunaan *cloud* untuk memastikan biaya tidak keluar dari kendali. Selain itu, memigrasikan kumpulan data di tempat dan memproses beban kerja ke *cloud* sering kali merupakan proses yang rumit bagi organisasi.

Membuat data dalam sistem data besar dapat diakses oleh *data scientists* dan analis lain juga merupakan tantangan, terutama di lingkungan terdistribusi yang mencakup campuran berbagai *platform* dan penyimpanan data. Untuk membantu analis menemukan data yang relevan, tim TI dan analitik semakin berupaya untuk membuat katalog data yang menggabungkan fungsi manajemen metadata dan aliran data. Kualitas data dan tata kelola data juga perlu menjadi prioritas untuk memastikan bahwa kumpulan data besar bersih, konsisten dan digunakan dengan benar.

### 2.1.7 Karakteristik *Big Data*

#### 1. *Volume*

Nama *Big Data* sendiri terkait dengan ukuran yang sangat besar. Ukuran data memainkan peran yang sangat penting dalam menentukan nilai dari data. Juga, apakah data tertentu benar-benar dapat dianggap sebagai Data Besar atau tidak, tergantung pada volume data. Oleh karena itu, '*Volume*' adalah salah satu karakteristik yang perlu dipertimbangkan saat berurusan dengan *Big Data*.

#### 2. *Variety*

Aspek *Big Data* selanjutnya adalah keanekaragamannya. Varietas mengacu pada sumber-sumber yang heterogen dan sifat data, baik terstruktur dan tidak terstruktur. Selama hari-hari sebelumnya, *spreadsheet* dan basis data adalah satu-satunya sumber data yang dipertimbangkan oleh sebagian besar aplikasi. Saat ini, data dalam bentuk email, foto, video, perangkat pemantauan, PDF, audio, dll. Juga sedang dipertimbangkan dalam aplikasi analisis. Keragaman data yang tidak terstruktur ini menimbulkan masalah tertentu untuk penyimpanan, penambahan, dan analisis data.

#### 3. *Velocity*

Istilah '*velocity*' mengacu pada kecepatan pembuatan data. Seberapa cepat data dihasilkan dan diproses untuk memenuhi permintaan, menentukan potensi nyata dalam data. *Big Data Velocity* berkaitan dengan kecepatan di mana data mengalir dari sumber-sumber seperti proses bisnis, log aplikasi, jaringan, dan situs media sosial, sensor, perangkat *Mobile*, dll. Aliran data sangat besar dan berkelanjutan.

#### 4. *Variability*

Mengacu pada ketidakkonsistenan yang dapat ditunjukkan oleh data pada waktu tertentu, sehingga menghambat proses untuk dapat menangani dan mengelola data secara efektif.

### 2.1.8 Keuntungan dan Kerugian *Big Data*

Peningkatan jumlah data yang tersedia menghadirkan peluang dan masalah. Secara umum, memiliki lebih banyak data tentang pelanggan seseorang (dan pelanggan

potensial) harus memungkinkan perusahaan untuk menyesuaikan produk dan upaya pemasaran mereka dengan lebih baik untuk menciptakan tingkat kepuasan tertinggi dan mengulangi bisnis. Perusahaan yang mampu mengumpulkan data dalam jumlah besar diberikan kesempatan untuk melakukan analisis yang lebih dalam dan lebih kaya.

Sementara analisis yang lebih baik adalah positif, data besar juga dapat membuat kelebihan dan kebisingan. Perusahaan harus dapat menangani volume data yang lebih besar, sambil menentukan data mana yang mewakili sinyal dibandingkan dengan noise. Menentukan apa yang membuat data relevan menjadi faktor utama.

Selanjutnya, sifat dan format data dapat memerlukan penanganan khusus sebelum ditindaklanjuti. Data terstruktur, yang terdiri dari nilai numerik, dapat dengan mudah disimpan dan disortir. Data yang tidak terstruktur, seperti email, video, dan dokumen teks, mungkin memerlukan teknik yang lebih canggih untuk diterapkan sebelum menjadi berguna.

## 2.2 Menggunakan python

Memilih bahasa pemrograman dalam bidang *Big Data* merupakan hal spesifik dan tergantung pada tujuan proyek. Namun, apa pun yang menjadi tujuannya, *Python* dan *Big Data* adalah kombinasi yang tidak terpisahkan ketika kita mempertimbangkan bahasa pemrograman untuk fase pengembangan *Big Data*. Ini adalah keputusan penting karena sekali Anda mulai mengembangkan proyek, maka akan sulit untuk bermigrasi dalam bahasa lain. Selain itu, tidak semua proyek *big data* memiliki tujuan yang sama. Misalnya, dalam proyek *big data*, tujuannya mungkin hanya memanipulasi data atau membangun analitik sedangkan yang lain bisa untuk *Internet of Things (IoT)*.

Python adalah bahasa pemrograman serba guna yang memungkinkan programmer menulis lebih sedikit baris kode dan membuatnya lebih mudah dibaca. Python memiliki fitur *scripting* dan selain itu menggunakan banyak perpustakaan canggih seperti NumPy, Matplotlib, dan SciPy yang membuatnya berguna untuk komputasi ilmiah. Python adalah alat yang sangat baik dan sangat cocok sebagai kombinasi *big data python* untuk analisis data karena alasan di bawah ini:

### 1. Sumber Terbuka (*Open source*)

Python adalah bahasa pemrograman *open source* yang dikembangkan menggunakan model berbasis komunitas. Itu dapat dijalankan di lingkungan Windows dan Linux. Selain itu, Anda dapat *platformporting* ke *platform* lain karena mendukung banyak *platform*.

### 2. Dukungan Perpustakaan (*Library Support*)

Python banyak digunakan untuk komputasi ilmiah di bidang akademik dan beberapa industri. Python terdiri dari sejumlah besar pustaka analitik yang teruji dengan baik yang mencakup paket-paket seperti :

#### (a) Numerical computing

- (b) Data analysis
- (c) Statistical analysis
- (d) Visualization
- (e) Machine learning

### 3. Kecepatan (*Speed*)

Karena Python adalah bahasa tingkat tinggi, Python memiliki banyak manfaat yang mempercepat pengembangan kode. Ini memungkinkan ide prototyping yang membuat pengkodean cepat sambil menjaga transparansi yang besar antara kode dan pelaksanaannya. Sebagai hasil dari transparansi kode, pemeliharaan kode dan proses menambahkannya ke basis kode dalam lingkungan pengembangan multi-pengguna menjadi mudah.

### 4. Jangkauan (*Scope*)

Python adalah bahasa pemrograman berorientasi objek yang juga mendukung struktur data tingkat lanjut seperti daftar, set, tuple, kamus dan banyak lagi. Ini mendukung banyak operasi komputasi ilmiah seperti operasi matriks, bingkai data, dll. Kemampuan ini dalam Python meningkatkan ruang lingkup untuk menyederhanakan dan mempercepat operasi data.

### 5. Dukungan Pemrosesan Data (*Data Processing Support*)

Python menyediakan dukungan canggih untuk data gambar dan suara karena fitur *inbuilt* untuk mendukung pemrosesan data untuk data tidak terstruktur dan tidak konvensional yang merupakan kebutuhan umum dalam data besar ketika menganalisis data media sosial. Ini adalah alasan lain untuk membuat Python dan data besar bermanfaat satu sama lain.

## 2.3 Tools

Tools dalam penggunaan big data merupakan sebuah software yang digunakan untuk proses big data itu sendiri. Tools itu sendiri ada yang berbayar dan gratis (open-source). Berikut beberapa tools open-source yang bisa digunakan seperti berikut.

1. Hadoop
2. Spark
3. Tableau
4. Jupyter Notebook

### 2.3.1 Hadoop

Hadoop adalah sebuah framework yang memungkinkan proses distribusi data dalam jumlah besar yang dicluster setiap komputer dengan menggunakan algoritma pemrograman yang sederhana.

Hadoop di desain seperti merekayasa satu server, dimana terdapat banyak mesin dan setiap mesin memiliki menyediakan komputasi lokal dan lokasi penyimpanan.



**Gambar 2.1** Logo Hadoop

### 2.3.2 Spark

Spark adalah sebuah sistem komputasi cluster dengan tujuan cepat dan umum. Spark ini mendukung level API seperti java, Scala, Python dan R serta didukung oleh pemvisualisasi data. Spark juga menyediakan seperti Spark sql dan struktur data proses, library untuk machine learning, GraphX untuk pembuatan grafik serta spark streaming.



**Gambar 2.2** Logo Spark

### 2.3.3 Tableau

Tableau adalah sebuah software yang digunakan untuk visualisasi data.



**Gambar 2.3** Logo Tableau

## 2.4 Library



## BAB 3

---

# JUPYTER NOTEBOOK

---

### 3.1 Apakah itu Jupyter Notebook

Jupyter notebook adalah salah satu software berbasis web open-source yang bisa membuat dan berbagi dokumen live interaktif seperti kode, visualisasi dan teks naratif.



**Gambar 3.1** Jupyter

## 3.2 Cara Instalasi

### 3.2.1 Windows

Untuk instalasi jupyter notebook, bisa melalui 2 cara. Pertama, menggunakan pip dan cara kedua menggunakan anaconda.

#### 3.2.1.1 Pip

1. Jika anda menggunakan python 3, ketik perintah berikut.

```
1 python3 -m pip install --upgrade pip
2 python3 -m pip install jupyter
3
```

Sedangkan untuk python 2, ketik perintah berikut.

```
1 python -m pip install --upgrade pip
2 python -m pip install jupyter
3
```

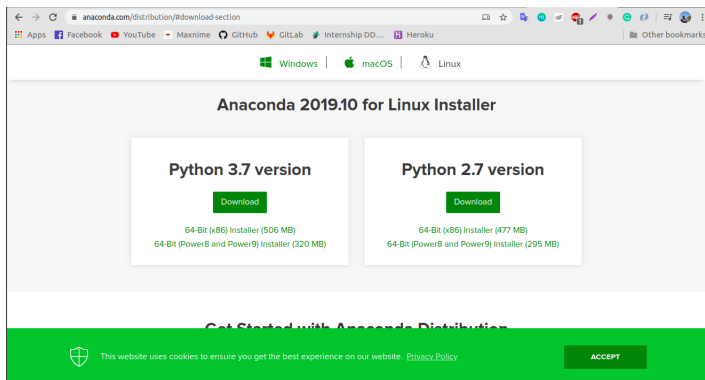
2. Tunggu hingga proses instalasi selesai. Dan Untuk mengecek jupyter notebook nya ketik berikut

```
1 jupyter notebook
2
```

Jika terdapat error **'jupyter' is not recognized as an internal or external command, operable program or batch file.**, coba ketik **python -m notebook**

#### 3.2.1.2 Anaconda

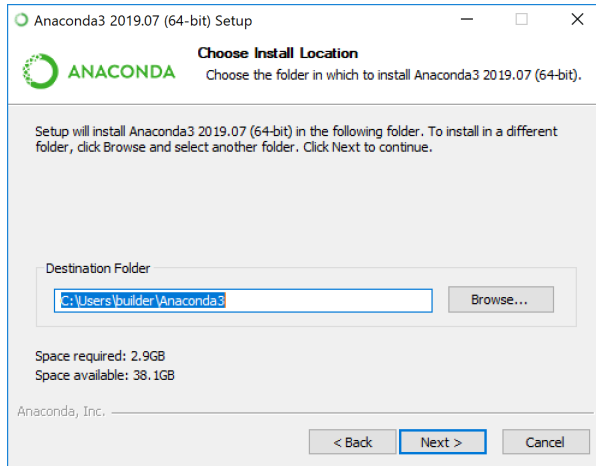
1. Download installer anaconda di <https://www.anaconda.com/distribution/>.



**Gambar 3.2** Download

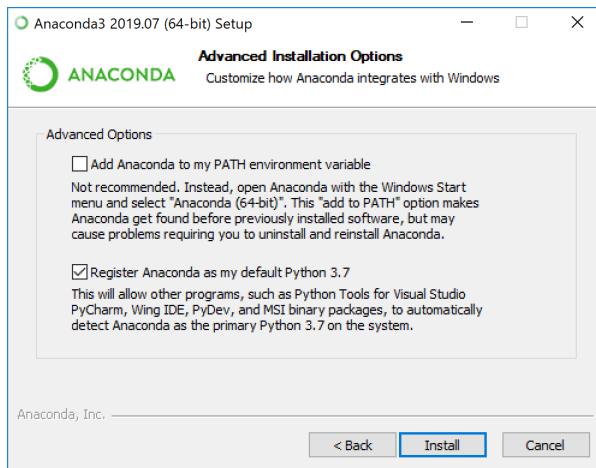
2. Lakukan verifikasi data integrity dengan SHA-256 (direkomendasikan).

3. klik file installer untuk running installer tersebut.
4. Pilih lokasi instalasi.



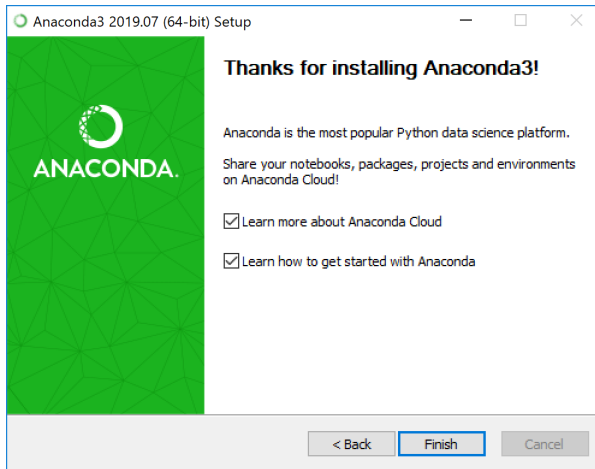
**Gambar 3.3** Select Folder

5. Centang **Add Anaconda to Environment Path** untuk menambahkan anaconda ke dalam environment.



**Gambar 3.4** Add Path Environment

6. Tunggu hingga proses instalasi selesai.



**Gambar 3.5** Instalasi selesai

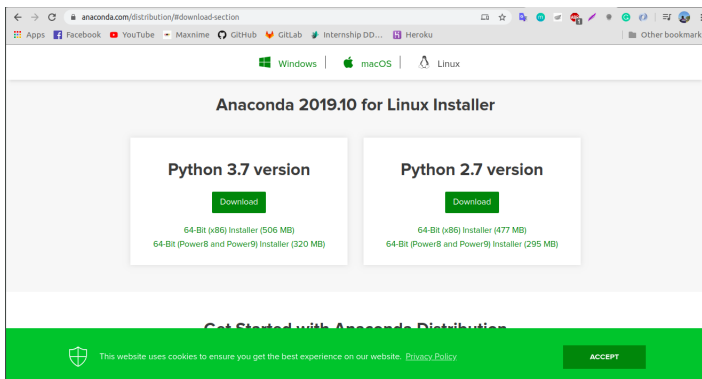
## 3.2.2 Ubuntu

### 3.2.2.1 *pip*

1. Buka Terminal.
2. Lalu ketik **pip install notebook**. Dan tunggu beberapa saat.
3. Instalasi selesai.

### 3.2.2.2 *Anaconda*

1. Download installer anaconda di <https://www.anaconda.com/distribution/>.



**Gambar 3.6** Download

2. Buka Terminal.
3. Lakukan verifikasi data integrity dengan SHA-256 (direkomendasikan).

```
1 sha256sum /path/ filename
2
```

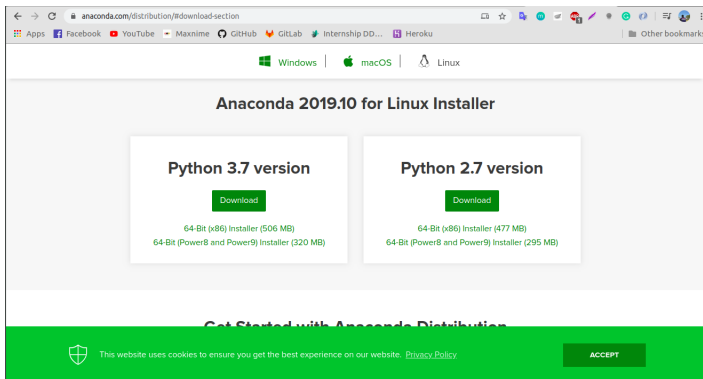
4. Ketik perintah berikut untuk mulai install anaconda.

```
1 bash ~/Downloads/Anaconda3-2019.10-Linux-x86_64.sh
2
```

Ketika diminta untuk agreement, tekan yes saja.

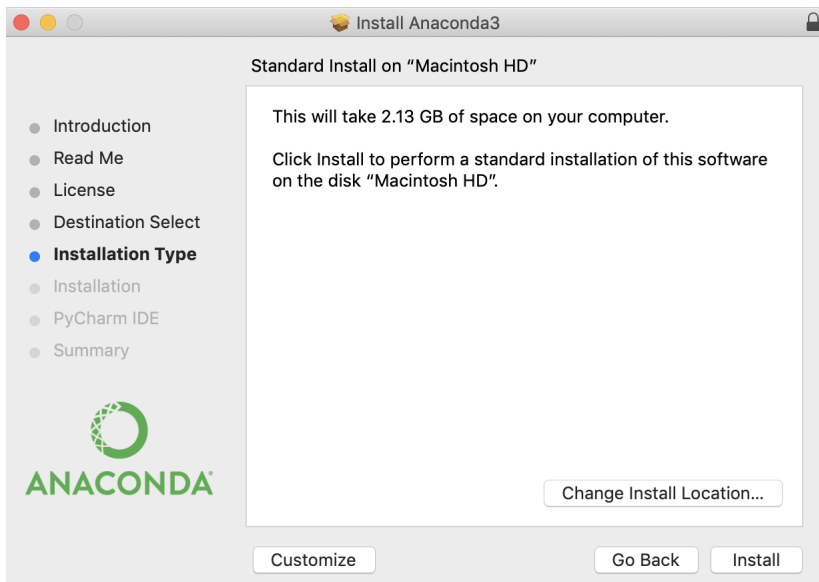
### 3.2.3 MacOS

1. Download installer anaconda di <https://www.anaconda.com/distribution/>.



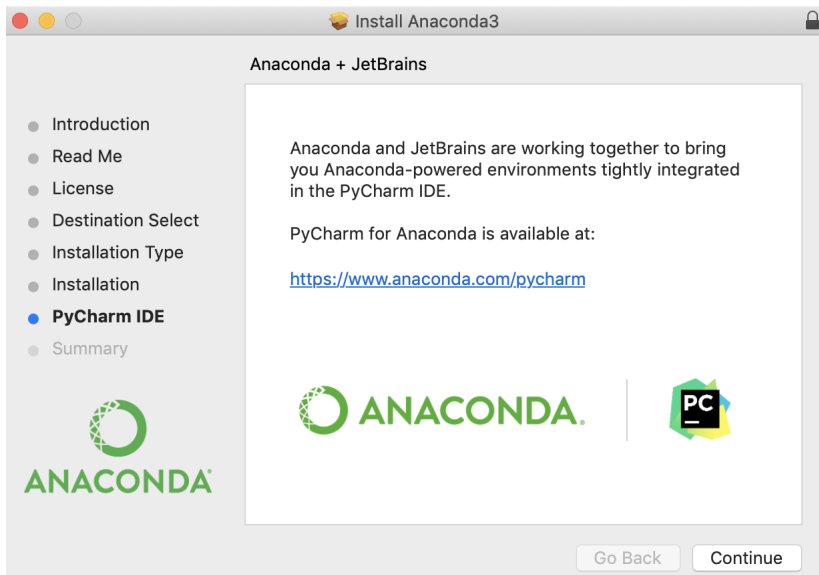
**Gambar 3.7** Download

2. Lakukan verifikasi data integrity dengan SHA 256 (direkomendasikan).
3. klik file installer untuk running installer tersebut.
4. Lalu ikuti yang ada di screen terminal macOS.
5. Pastikan lokasi instalasi anaconda tersebut. Lalu klik tombol install.



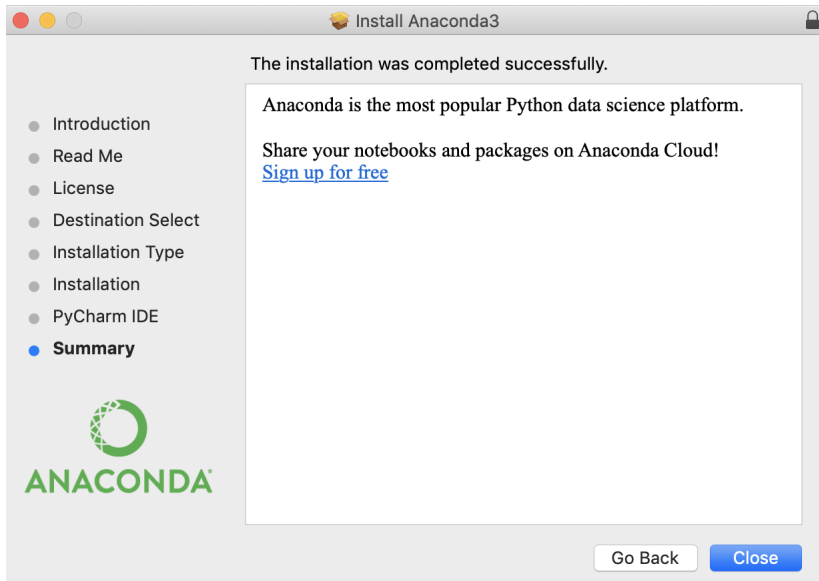
Gambar 3.8 direktori install

6. Setelah itu, terdapat jika kita ingin instal pycharm. Maka klik link yang ada di screen.



Gambar 3.9 Opsi PyCharm

7. Tunggu beberapa saat, Anaconda telah terinstall.



**Gambar 3.10** Instalasi selesai

### 3.2.4 Notebook documents

*Notebook documents* (atau "buku catatan", semuanya huruf kecil) adalah dokumen yang diproduksi oleh Jupyter Notebook App, yang berisi kode komputer (mis. Python) dan elemen teks kaya (paragraf, persamaan, angka, tautan, dll ...). Dokumen Notebook adalah dokumen yang dapat dibaca manusia yang berisi uraian analisis dan hasilnya (angka, tabel, dll.) Serta dokumen yang dapat dieksekusi yang dapat dijalankan untuk melakukan analisis data.

### 3.2.5 Jupyter Notebook

Jupyter Notebook adalah aplikasi *web open-source* yang memungkinkan untuk membuat dan berbagi dokumen yang berisi kode langsung, persamaan, visualisasi, dan teks naratif. Penggunaan meliputi: pembersihan dan transformasi data, simulasi numerik, pemodelan statistik, visualisasi data, pembelajaran mesin, dan banyak lagi.

Jupyter Notebook App adalah aplikasi server-klien yang memungkinkan pengeditan dan menjalankan dokumen notebook melalui browser web. Aplikasi Notebook Jupyter dapat dijalankan pada desktop lokal yang tidak memerlukan akses internet (seperti dijelaskan dalam dokumen ini) atau dapat diinstal pada server jarak jauh dan diakses melalui internet. Selain menampilkan / mengedit / menjalankan dokumen notebook, Aplikasi Notebook Jupyter memiliki "Dasbor" (Dasbor Notebook),

”panel kontrol” yang memperlihatkan file-file lokal dan memungkinkan untuk membuka dokumen notebook atau mematikan kernel mereka.

### 3.2.6 Kernel

Kernel notebook adalah ”mesin komputasi” yang mengeksekusi kode yang terkandung dalam dokumen Notebook. Kernel ipython, dirujuk dalam mengeksekusi kode python. Ketika membuka dokumen Notebook, kernel yang terkait diluncurkan secara otomatis. Ketika notebook dijalankan (baik sel demi sel atau dengan menu Cell -> Run All), kernel melakukan perhitungan dan menghasilkan hasilnya. Bergantung pada jenis perhitungan, kernel dapat mengkonsumsi CPU dan RAM yang signifikan. Perhatikan bahwa RAM tidak dirilis sampai kernel dimatikan.

Ketika Jupyter memulai kernel, ia mengirimkannya file koneksi. Ini menentukan cara mengatur komunikasi dengan *frontend*. Ada dua opsi untuk menulis kernel:

1. Pengguna dapat menggunakan kembali mesin kernel IPython untuk menangani komunikasi, dan cukup jelaskan bagaimana mengeksekusi kode. Hal ini jauh lebih sederhana jika bahasa target dapat didorong dari Python.
2. Pengguna dapat mengimplementasikan mesin kernel dalam bahasa target. Ini lebih banyak bekerja pada awalnya, tetapi orang-orang yang menggunakan kernel pengguna mungkin lebih mungkin untuk berkontribusi jika itu dalam bahasa yang mereka tahu.

### 3.2.7 Notebook Dashboard

*Notebook Dashboard* adalah komponen yang ditampilkan pertama kali ketika membuka Aplikasi Notebook Jupyter. Dasbor Notebook terutama digunakan untuk membuka dokumen notebook, dan untuk mengelola kernel yang berjalan (memvisualisasikan dan mematikan). *Notebook Dashboard* memiliki fitur lain yang mirip dengan manajer file, yaitu menavigasi folder dan mengganti nama / menghapus file.

## 3.3 Perbedaan Jupyter Notebook dan Google Colab

*Google Colaboratory* (Google Colab) merupakan tools baru dari Google Internal Research yang ditujukan membantu para *Researcher* dalam mengolah data, khususnya bidang *Machine Learning*. Google Colab hampir mirip penggunaannya seperti Jupyter Notebook namun tidak memerlukan pengaturan atau setup terlebih dahulu sebelum digunakan dan berjalan sepenuhnya pada Cloud dengan memanfaatkan media penyimpanan Google Drive. *Researcher* dapat menulis dan mengeksekusi kode, menyimpan dan membagikan analisis, serta mengakses sumber daya komputasi yang kuat seperti layanan GPU secara gratis dari browser.

Jupyter Notebook dan Google Colab memiliki perbedaan sebagai berikut:

1. Infrastruktur



Google Colab berjalan di Google Cloud Platform (GCP). Karena itu kuat, fleksibel.

## 2. Perangkat keras

Google Colab baru-baru ini menambahkan dukungan untuk Tensor Processing Unit (TPU) selain GPU dan CPU yang ada. Jadi, ini masalah besar bagi semua orang yang belajar mendalam.

## 3. Harga

Meskipun begitu mahir dalam hal perangkat keras, layanan yang disediakan oleh Google Colab sepenuhnya gratis. Ini membuatnya lebih dahsyat.

## 4. Integrasi dengan Google Drive

Menarik dapat menggunakan drive google sebagai sistem file interaktif dengan Google Colab. Ini membuatnya mudah untuk menangani file yang lebih besar.

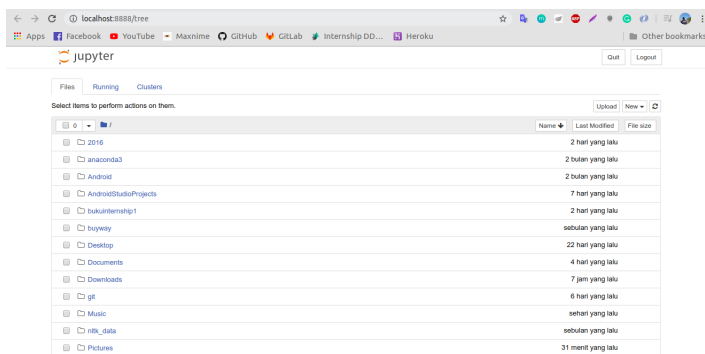
## 5. Hikmah bagi Komunitas Riset dan Startup

Mungkin ini adalah satu-satunya alat yang tersedia di pasar yang menyediakan PaaS yang begitu bagus secara gratis bagi pengguna. Ini sangat membantu bagi startup, komunitas riset dan siswa di ruang belajar yang mendalam.

### 3.3.1 Cara Penggunaan

#### 3.3.1.1 Windows

1. Buka Command Prompt
2. Ketik **jupyter notebook**, dan secara otomatis akan terbuka di browser anda.

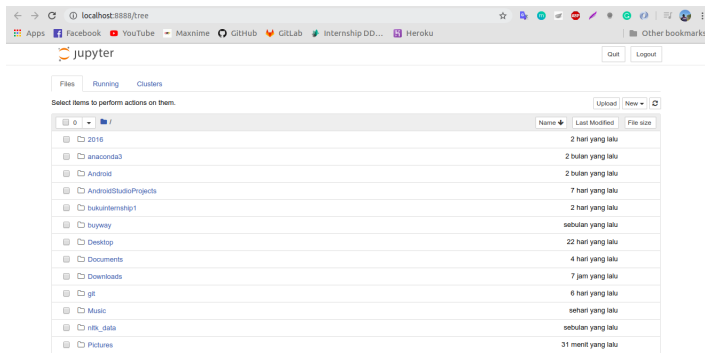


**Gambar 3.11** Jupyter Dashboard

Jika ketika menjalankan perintah tersebut, bisa diganti dengan **python -m notebook**

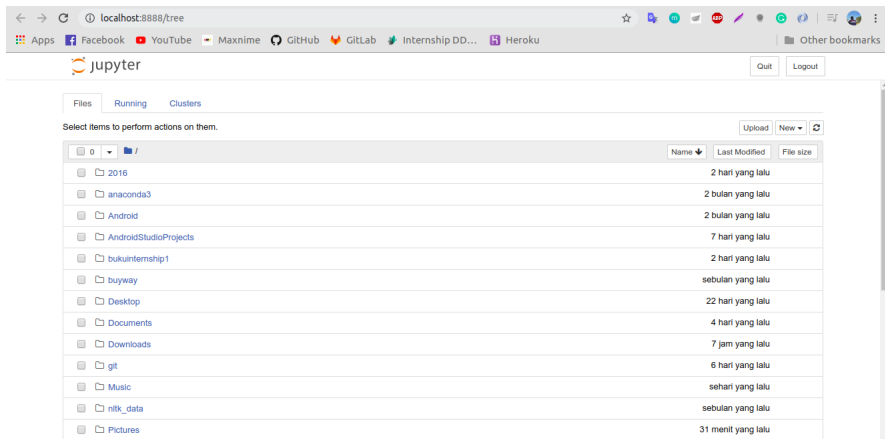
### 3.3.1.2 Linux

1. Buka Terminal
2. Ketik **jupyter notebook**, dan secara default akan terbuka di browser anda.



**Gambar 3.12** Jupyter Dashboard

### 3.3.2 Tampilan Jupyter Notebook



**Gambar 3.13** Struktur Dashboard

Penjelasan gambar 3.13.

1. Tab File  
Disini berisi kumpulan direktori folder yang ada.
2. Tab Running

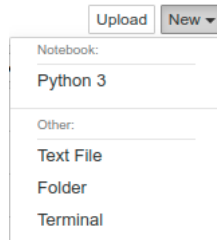
Berisi kumpulan file-file yang sedang dijalankan.

### 3. Tombol Upload

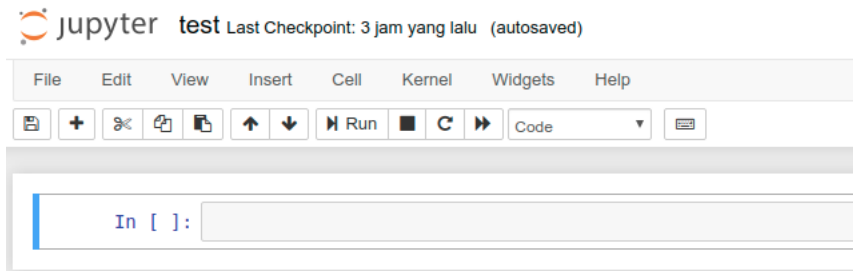
Untuk mengupload file.

### 4. Tombol New

Berisi sub menu seperti Python3, Textfile, Folder dan Terminal.



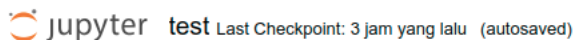
**Gambar 3.14** Struktur Dashboard



**Gambar 3.15** Notebook Document

### 1. Notebook name

Merupakan nama file yang sedang kita gunakan.



**Gambar 3.16** Notebook Name

### 2. Menu Bar

Merupakan sekumpulan menu yang digunakan untuk menyimpan, membuka, dan sebagainya.



**Gambar 3.17** Menu Bar

### 3. Toolbar

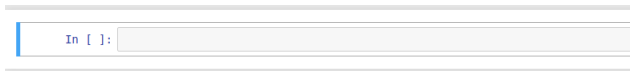
Merupakan sekumpulan icon atau tombol yang bisa menjadi alternatif dari beberapa menu yang ada di menu bar. Letak toolbar biasanya berada dibawah menu bar.



**Gambar 3.18** ToolBar

### 4. Code Cell

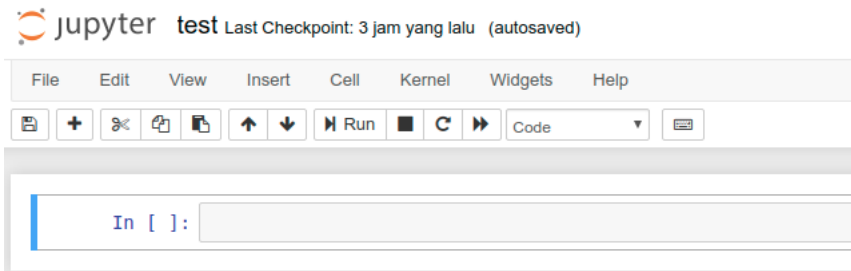
Merupakan shell yang disediakan oleh notebook document untuk membuat program python.



**Gambar 3.19** Code Cell

## 3.3.3 Struktur Notebook Document

Dari tampilan dashboard jupyter notebook pada gambar 3.13, pilih file yang telah dibuat sebelumnya. Tunggu sesaat dan akan berpindah ke halaman editor.

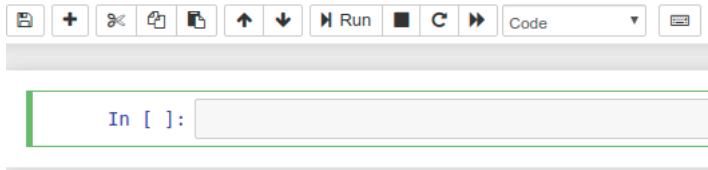


**Gambar 3.20** Notebook Document

### 1. Code Cell

Code cell memungkinkan untuk di edit dan bisa menulis kode baru. Bahasa pemrograman bisa diinputkan dalam code cell ini. Namun default kernel nya adalah kernel python (IPython) untuk menjalankan kode python.

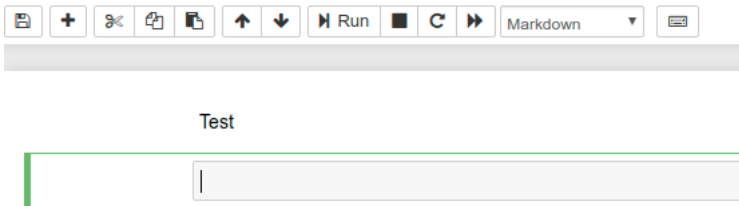
Ketika code cell tersebut dieksekusi atau dijalankan. Hasil dari eksekusi tersebut akan ditampilkan secara langsung dibawah cell yang sedang dijalankan



**Gambar 3.21** Code Cell

## 2. Markdown Cell

Markdown cell merupakan jenis cell yang biasa digunakan untuk penggunaan teks narasi. Dimana markdown cell ini digunakan ketika ingin membuat teks deskripsi.



**Gambar 3.22** Markdown Cell

## 3. Heading

Heading biasanya digunakan untuk membuat judul file. Dimana terdapat 6 tingkatan. Semakin sedikit jumlah pagar (#), maka semakin besar ukuran huruf tersebut dan jika semakin banyak jumlah pagar (#) maka semakin kecil ukuran huruf.

### (a) Heading 1

# Tes

**Tes**

**Gambar 3.23** Heading 1

### (b) Heading 2

`## Tes`**Tes****Gambar 3.24** Heading 2

(c) Heading 3

`### Tes`**Tes****Gambar 3.25** Heading 3

(d) Heading 4

`#### Tes`**Tes ¶****Gambar 3.26** Heading 4

(e) Heading 5

`##### Tes`**Tes****Gambar 3.27** Heading 5

(f) Heading 6

`##### Tes`**Tes****Gambar 3.28** Heading 6

## BAB 4

---

# MANIPULASI DATA

---

### 4.1 Bekerja dengan Data

#### 4.1.1 Numpy

Numpy merupakan kepanjangan dari *Numerical Python*. Numpy adalah sebuah library yang digunakan pemrosesan array. Numpy juga mendukung dalam objek array multidimensi. Numpy merupakan library dasar yang digunakan dalam data science. Numpy juga bisa digunakan untuk wadah data generik multidimensi yang efisien.



**Gambar 4.1** Numpy

### 4.1.2 Cara Instalasi

Hal yang harus diperhatikan ketika ingin melakukan instalasi library adalah harus sudah tersedia pip. Dan untuk yang belum tau cara instalasi pip, bisa dilihat di Bab 1 subbab 1.9.

Sebenarnya pip itu sendiri sudah tersedia ketika melakukan instalasi python itu sendiri. Sehingga kita tidak perlu melakukan instalasi pip. Namun perlu diketahui cara instalasi pip seperti apa.

#### 4.1.2.1 Windows

1. Buka Command Prompt.
2. Lalu ketik **pip install numpy**. Tunggu hingga instalasi selesai.
3. Instalasi Selesai.

#### 4.1.2.2 Linux

1. Buka Terminal.
2. Lalu ketik **pip install numpy**. Tunggu hingga instalasi selesai.
3. Instalasi Selesai.

#### 4.1.2.3 MacOS

1. Buka Terminal.
2. Lalu ketik **pip install numpy**. Tunggu hingga instalasi selesai.
3. Instalasi Selesai.

### 4.1.3 Contoh Penggunaan

**4.1.3.1 Array** Array adalah sekumpulan data dengan tipe data yang sama dan dinyatakan dengan nilai tuple yang sama (nilai positif). Elemen dalam array di python di akses menggunakan tanda kurung siku ([]).

1. Creating Array

Pembuatan array bisa dengan berbagai cara seperti menentukan ukuran array dan lain-lain. Array juga bisa dengan berbagai tipe data seperti list, tupel dan lain-lain. Jenis array yang dihasilkan dari jenis elemen yang diurutkan.



Untuk contoh implementasi pembuatan array seperti berikut.

```
1 import numpy as np
2
3 arr = np.array([1, 2, 3])
4 print("Array with Rank 1: \n", arr)
5
6 arr = np.array([[1, 2, 3],
7                [4, 5, 6]])
8 print("Array with Rank 2: \n", arr)
9
10 arr = np.array((1, 3, 2))
11 print("\nArray created using "
12       "passed tuple:\n", arr)
13
```

**Listing 4.1** Array

Keluaran dari implementasi script tersebut pada gambar 4.1.

```
Array with Rank 1:
[1 2 3]
Array with Rank 2:
[[1 2 3]
 [4 5 6]]

Array created using passed tuple:
[1 3 2]
```

**Gambar 4.2** Array

Penjelasan script 4.2 seperti berikut.

- Pada baris 1, merupakan import module numpy dengan alias np. Pengalihan disini digunakan agar mempermudah pemanggilan nama module.
- Pada baris 3, merupakan penginisialisasian variabel **arr** untuk isian nilai array 1 dimensi.
- Pada baris 4, merupakan perintah untuk menampilkan nilai variabel **arr** di baris 3.
- Pada baris 6, merupakan penginisialisasian variabel **arr** untuk isian nilai array 2 dimensi.
- Pada baris 8, merupakan perintah untuk menampilkan nilai variabel **arr** di baris 6.
- Pada baris 10, merupakan penginisialisasian variabel **arr** untuk isian nilai array dengan tipe data tuple.

- (g) Pada baris 11, merupakan perintah untuk menampilkan nilai variabel **arr** di baris 3.

## 2. Index Array

Dalam array numpy, banyak cara untuk mengakses indeks array tersebut. Salah satunya adalah dengan menampilkan nilai array index yang di iris atau dikecualikan. Dimana nilai tersebut diambil dari elemen nilai array yang asli. Karena nilai irisan array tersebut bisa mengubah nilai array yang asli atau istilah lainnya memodifikasi nilai array.

Untuk contoh implementasi pembuatan array seperti berikut.

```

1      import numpy as np
2
3      arr = np.array([[ -1,  2,  0,  4],
4                      [ 4, -0.5,  6,  0],
5                      [ 2.6,  0,  7,  8],
6                      [ 3, -7,  4,  2.0]])
7      print("Initial Array: ")
8      print(arr)
9
10     Index_arr = arr[[1, 1, 0, 3],
11                     [3, 2, 1, 0]]
12     print ("\nElements at indices (1, 3), "
13           "(1, 2), (0, 1), (3, 0):\n", Index_arr)
14

```

**Listing 4.2** Array Index

Keluaran dari implementasi script tersebut pada gambar 4.3 dan 4.4.

```

Initial Array:
[[-1.  2.  0.  4. ]
 [ 4. -0.5  6.  0. ]
 [ 2.6  0.  7.  8. ]
 [ 3. -7.  4.  2. ]]

```

**Gambar 4.3** Array Index

```

Elements at indices (1, 3), (1, 2), (0, 1), (3, 0):
[ 0.54  2.  3.]

```

**Gambar 4.4** Array Index

Penjelasan script 4.2 seperti berikut.

- (a) Pada baris 1, merupakan import module numpy dengan alias np. Pengalisan disini digunakan agar mempermudah pemanggilan nama module.
- (b) Pada baris 3, merupakan penginisialisasian variabel **arr** untuk isian nilai array.
- (c) Pada baris 8, merupakan perintah untuk menampilkan nilai variabel **arr** di baris 3.
- (d) Pada baris 10, merupakan penginisialisasian variabel **index arr** untuk isian nilai array.
- (e) Pada baris 12, merupakan pengirisan data dengan nilai di baris 1 kolom 3 (1,3) dan seterusnya.

### 3. Array Operations

Array bisa digunakan untuk operasi matematika yang dikombinasikan dengan array tertentu.

Untuk contoh implementasi pembuatan array seperti berikut.

```

1  import numpy as np
2
3  a = np.array([[1, 2],
4               [3, 4]])
5  b = np.array([[4, 3],
6               [2, 1]])
7
8  print ("Adding 1 to every element:", a + 1)
9  print ("\nSubtracting 2 from each element:", b - 2)
10
11 print ("\nSum of all array "
12        "elements: ", a.sum())
13 print ("\nArray sum:\n", a + b)

```

**Listing 4.3** Array Operation

Keluaran dari implementasi script tersebut pada gambar 4.5.

```
Adding 1 to every element:  
[[2 3]  
 [4 5]]  
  
Subtracting 2 from each element:  
[[ 2  1]  
 [ 0 -1]]  
  
Sum of all array elements: 10  
  
Array sum:  
[[5 5]  
 [5 5]]
```

**Gambar 4.5** Array Basic

Penjelasan script 4.3 seperti berikut.

- (a) Pada baris 1, merupakan import module numpy dengan alias np. Pengaliansan disini digunakan agar mempermudah pemanggilan nama module.
- (b) Pada baris 3, merupakan penginisialisasian variabel **a** untuk isian nilai array.
- (c) Pada baris 5, merupakan penginisialisasian variabel **b** untuk isian nilai array.
- (d) Pada baris 7, merupakan perintah untuk menampilkan nilai array di baris 3 dengan kondisi isi nilai array ditambah 1.
- (e) Pada baris 8, merupakan perintah untuk menampilkan nilai array di baris 5 dengan kondisi isi nilai array dikurangi 2.
- (f) Pada baris 11, merupakan perintah untuk menampilkan nilai array di baris 3 dengan kondisi isi nilai array dijumlahkan semua.
- (g) Pada baris 11, merupakan perintah untuk menampilkan nilai array di baris 3 dan baris 5 dengan kondisi isi nilai array dijumlahkan semua sesuai posisi index array tersebut.

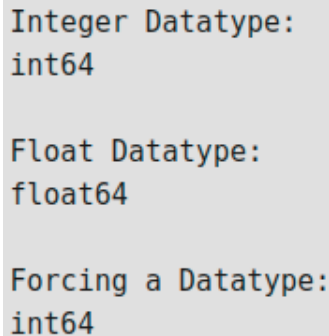
#### 4. DataType Object

Dalam numpy, kita juga mengidentifikasi data yang pada objek tersebut menggunakan tipe data apa. Untuk contoh implementasi seperti berikut.

```
1 import numpy as np
2
3 x = np.array([1, 2])
4 print("Datatype: ")
5 print(x.dtype)
6
7 x = np.array([1.0, 2.0])
8 print("\n Datatype: ")
9 print(x.dtype)
```

**Listing 4.4** Array Identification

Keluaran dari implementasi script tersebut pada gambar 4.6.



```
Integer Datatype:
int64

Float Datatype:
float64

Forcing a Datatype:
int64
```

**Gambar 4.6** Array DataType

Penjelasan script 4.4 seperti berikut.

- Pada baris 1, merupakan import module numpy dengan alias np. Pengaliansan disini digunakan agar mempermudah pemanggilan nama module.
- Pada baris 3, merupakan penginisialisasian variabel **x** untuk isian nilai array.
- Pada baris 4, merupakan perintah untuk menampilkan tulisan Datatype.
- Pada baris 5, merupakan perintah untuk menampilkan jenis tipe data pada object di baris 3.
- Pada baris 7, merupakan penginisialisasian variabel **x** untuk isian nilai array.
- Pada baris 8, merupakan perintah untuk menampilkan tulisan Datatype.
- Pada baris 9, merupakan perintah untuk menampilkan jenis tipe data pada object di baris 7.

## 5. Operation Mathematic

Dalam numpy array, operasi dasar matematika bisa dilakukan pada setiap elemen array tersebut. Beberapa perintah tersebut seperti **sum** untuk menjumlahkan setiap elemen di array tersebut dan **T** untuk mentransformasi setiap elemen yang ada, dan lain-lain.

Untuk contoh implementasi seperti berikut.

```

1  import numpy as np
2
3  arr1 = np.array([[4, 7], [2, 6]],
4                  dtype = np.float64)
5
6  arr2 = np.array([[3, 6], [2, 8]],
7                  dtype = np.float64)
8
9  Sum = np.add(arr1, arr2)
10 print("Addition of Two Arrays: ")
11 print(Sum)
12
13 Sum1 = np.sum(arr1)
14 print("\nAddition of Array elements: ")
15 print(Sum1)
16
17 Sqrt = np.sqrt(arr1)
18 print("\nSquare root of Array1 elements: ")
19 print(Sqrt)
20
21 Trans_arr = arr1.T
22 print("\nTranspose of Array: ")
23 print(Trans_arr)

```

**Listing 4.5** Array Mathematic

Keluaran dari implementasi script diatas seperti gambar 4.7.

```

Addition of Two Arrays:
[[ 7. 13.]
 [ 4. 14.]]

Addition of Array elements:
19.0

Square root of Array1 elements:
[[2.          2.64575131]
 [1.41421356  2.44948974]]

Transpose of Array:
[[4. 2.]
 [7. 6.]]

```

**Gambar 4.7** Array Mathematic

Penjelasan script 4.5 seperti berikut.

- (a) Pada baris 1, merupakan import module numpy dengan alias np. Pengalisan disini digunakan agar mempermudah pemanggilan nama module.
- (b) Pada baris 3, merupakan penginisialisasian variabel **x** untuk isian nilai array.

- (c) Pada baris 4, merupakan pendeklarasian datatype dengan tipe data float.
- (d) Pada baris 6, merupakan penginisialisasian variabel `x` untuk isian nilai array.
- (e) Pada baris 7, merupakan pendeklarasian datatype dengan tipe data float.
- (f) Pada baris 9, penginisialisasian variabel untuk menambahkan setiap elemen di array pada baris 3 dan baris 4.
- (g) Pada baris 10, merupakan perintah untuk menampilkan text.
- (h) Pada baris 11, merupakan perintah untuk menampilkan isi dari variabel di baris 9.
- (i) Pada baris 13, penginisialisasian variabel untuk menambahkan setiap elemen di array pada baris 3.
- (j) Pada baris 14, merupakan perintah untuk menampilkan text.
- (k) Pada baris 15, merupakan perintah untuk menampilkan isi dari variabel di baris 13.
- (l) Pada baris 17, penginisialisasian variabel untuk mencari nilai akar dari setiap elemen di array pada baris 3.
- (m) Pada baris 18, merupakan perintah untuk menampilkan text.
- (n) Pada baris 19, merupakan perintah untuk menampilkan isi dari variabel di baris 17.
- (o) Pada baris 21, penginisialisasian variabel untuk mentransformasi setiap elemen di array pada baris 3.
- (p) Pada baris 22, merupakan perintah untuk menampilkan text.
- (q) Pada baris 23, merupakan perintah untuk menampilkan isi dari variabel di baris 21.

## 6. String Operations

### (a) Lower case.

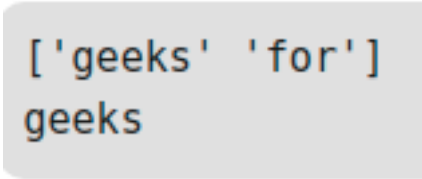
Lower case adalah sebuah fungsi dimana mengkonversi karakter text menjadi huruf kecil semua.

Untuk contoh implementasi seperti berikut.

```
1 import numpy as np
2
3 print(np.char.lower(['GEEKS', 'FOR']))
4
5 print(np.char.lower('GEEKS'))
6
```

**Listing 4.6** Lowercase

Keluaran dari implementasi script diatas seperti gambar 4.8.



```
[ 'geeks' 'for' ]
geeks
```

**Gambar 4.8** Lowercase

Penjelasan script 4.6 seperti berikut.

- i. Pada baris 1, merupakan import module numpy dengan alias np. Pengaliansan disini digunakan agar mempermudah pemanggilan nama module.
- ii. Pada baris 3, merupakan perintah untuk konversi text yang disimpan dalam bentuk array. Jika dalam bentuk array artinya, text yang dikonversi bisa lebih dari 1 text.
- iii. Pada baris 5, merupakan perintah untuk konversi text.

(b) Upper case.


Upper case adalah sebuah fungsi dimana mengkonversi karakter text menjadi huruf kapital semua.

Untuk contoh implementasi seperti berikut.

```
1 import numpy as np
2
3 print(np.char.upper(['geeks', 'for']))
4
5 print(np.char.upper('geeks'))
6
```

**Listing 4.7** Uppercase

Keluaran dari implementasi script diatas seperti gambar 4.9.



```
[ 'GEEKS' 'FOR' ]
GEEKS
```

**Gambar 4.9** Uppercase

Penjelasan script 4.7 seperti berikut.

- i. Pada baris 1, merupakan import module numpy dengan alias np. Pengaliansan disini digunakan agar mempermudah pemanggilan nama module.



- ii. Pada baris 3, merupakan perintah untuk konversi text yang disimpan dalam bentuk array. Jika dalam bentuk array artinya, text yang dikonversi bisa lebih dari 1 text.
- iii. Pada baris 5, merupakan perintah untuk konversi text.

### (c) Split

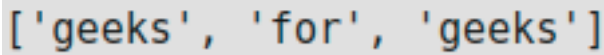
Split adalah sebuah fungsi untuk memisahkan per kata dalam sebuah text.

Untuk contoh implementasinya seperti berikut.

```
1 import numpy as np
2
3 print(np.char.split('geeks for geeks'))
4
```

**Listing 4.8** Split

Keluaran dari implementasi script diatas seperti gambar 4.10.



```
['geeks', 'for', 'geeks']
```

**Gambar 4.10** Split

Penjelasan script 4.8 seperti berikut.

- i. Pada baris 1, merupakan import module numpy dengan alias np. Pengalisan disini digunakan agar mempermudah pemanggilan nama module.
- ii. Pada baris 3, merupakan perintah untuk memisahkan text menjadi per kata.

#### (d) Count

Count adalah sebuah fungsi untuk menghitung text tersebut sejumlah berapa didalam character tersebut. Fungsi ini menerapkan konsep substring text, dimana hanya mengambil sebagian karakter dari text yang ada.

Untuk contoh implementasinya seperti berikut.

```
1 import numpy as np
2
3 a=np.array(['geeks', 'for', 'geeks'])
4
5 print(np.char.count(a, 'geek'))
6
7 print(np.char.count(a, 'fo'))
8
```

**Listing 4.9** Count

Keluaran dari implementasi script diatas seperti gambar 4.11.

The image shows the output of the Python script. It consists of two lines of text, each enclosed in square brackets and separated by commas. The first line is "[1, 0, 1]" and the second line is "[0, 1, 0]". The numbers are displayed in a monospaced font, with the first line in a slightly lighter blue color and the second line in a slightly darker blue color.

**Gambar 4.11** Count

Penjelasan script 4.9 seperti berikut.

- i. Pada baris 1, merupakan import module numpy dengan alias np. Pengaliansan disini digunakan agar mempermudah pemanggilan nama module.
- ii. Pada baris 3, merupakan penginisialisasi variabel **a** untuk elemen array.
- iii. Pada baris 5, merupakan perintah untuk menghitung berjumlah berapa text tersebut dalam setiap elemen pada variabel array di baris 3.
- iv. Pada baris 7, merupakan perintah untuk menghitung berjumlah berapa text tersebut dalam setiap elemen pada variabel array di baris 3.

Dan perintah penggunaan numpy untuk string masih banyak lagi seperti berikut.

(a) `numpy.find()`

`numpy.find()` untuk Mencari nilai boolean dimana jika text substring tersebut ada, maka bernilai 0 dan jika tidak bernilai -1.

(b) `numpy.title()`

`numpy.title` untuk merubah karakter pertama dalam string menjadi huruf besar disetiap kata, sedangkan karakter berikutnya menjadi huruf kecil.

(c) `numpy.capitalize()`

`numpy.capitalize` memiliki fungsi yang seperti `numpy.title`, namun dalam `numpy.capitalize` jika nilai String tersebut di karakter pertama sudah memiliki huruf besar maka akan mengembalikan nilai sesuai nilai string tersebut.

(d) `numpy.islower()`

`numpy.islower` untuk mengecek nilai string tersebut apakah sudah lower-case atau huruf kecil semua karakternya. Jika iya, maka mengeluarkan nilai **True**.

## 7. Sorting

Dalam numpy, kita melakukan sorting data untuk mengurutkan data yang ada. Yang paling banyak digunakan adalah mengurutkan data berdasarkan angka / numerik. Dimana diurutkan dari yang terbesar atau yang terkecil. Library yang bisa digunakan seperti `sort`, `lexsort`, `argsort` dan lain-lain.

(a) `numpy.sort()`

`numpy.sort` merupakan fungsi untuk mengurutkan nilai dengan pengembalian nilai index didalam array. Untuk contoh pengimplementasian seperti berikut.

```

1      import numpy as np
2
3      a = np.array([[12, 15], [10, 1]])
4      arr1 = np.sort(a, axis = 0)
5      print ("Along first axis : \n", arr1)
6
7      a = np.array([[10, 15], [12, 1]])
8      arr2 = np.sort(a, axis = -1)
9      print ("\nAlong first axis : \n", arr2)
10
11     a = np.array([[12, 15], [10, 1]])
12     arr1 = np.sort(a, axis = None)
13     print ("\nAlong none axis : \n", arr1)
14

```

**Listing 4.10** Sort

```
Along first axis :  
[[10  1]  
 [12 15]]  
  
Along first axis :  
[[10 15]  
 [ 1 12]]  
  
Along none axis :  
[ 1 10 12 15]]
```

**Gambar 4.12** Sort

Penjelasan script 4.10 seperti berikut.

- i. Pada baris 1, merupakan import module numpy dengan alias np. Pengaliansan disini digunakan agar mempermudah pemanggilan nama module.
- ii. Pada baris 3, merupakan penginisialisasi variabel **a** untuk elemen array.
- iii. Pada baris 4, merupakan penginisialisasi variabel **arr1** untuk mengurutkan dengan axis 0. Artinya nilai pengembalian dengan nilai mengurut berbentuk kolom.
- iv. Pada baris 5, merupakan perintah untuk menampilkan isi dari variabel di baris 4.
- v. Pada baris 7, merupakan penginisialisasi variabel **a** untuk elemen array.
- vi. Pada baris 8, merupakan penginisialisasi variabel **arr1** untuk mengurutkan dengan axis 1. Artinya nilai pengembalian dengan nilai mengurut berbentuk baris.
- vii. Pada baris 9, merupakan perintah untuk menampilkan isi dari variabel di baris 8.
- viii. Pada baris 11, merupakan penginisialisasi variabel **a** untuk elemen array.
- ix. Pada baris 12, merupakan penginisialisasi variabel **arr1** untuk mengurutkan dengan axis 0. Artinya nilai pengembalian dengan nilai mengurut berbentuk array biasa.
- x. Pada baris 13, merupakan perintah untuk menampilkan isi dari variabel di baris 12.

(b) `numpy.argsort()`

```

1      import numpy as np
2
3      a = np.array([9, 3, 1, 7, 4, 3, 6])
4
5      print('Original array:\n', a)
6
7      b = np.argsort(a)
8      print('Sorted indices of original array->', b)
9
10     c = np.zeros(len(b), dtype = int)
11     for i in range(0, len(b)):
12         c[i] = a[b[i]]
13     print('Sorted array->', c)
14

```

**Listing 4.11** ArgSort

```

Original array:
[9 3 1 7 4 3 6]
Sorted indices of original array-> [2 1 5 4 6 3 0]
Sorted array-> [1 3 3 4 6 7 9]

```

**Gambar 4.13** ArgSort

Penjelasan script 4.11 sebagai berikut.

- i. Pada baris 1, merupakan import module numpy dengan alias np. Pengaliansan disini digunakan agar mempermudah pemanggilan nama module.
- ii. Pada baris 3, merupakan penginisialisasi variabel **a** untuk elemen array.
- iii. Pada baris 5, merupakan perintah untuk menampilkan isi variabel di baris 3.
- iv. Pada baris 7, merupakan penginisialisasi variabel **b** untuk mengurutkan data.
- v. Pada baris 8, merupakan perintah untuk menampilkan isi variabel di baris 7.
- vi. Pada baris 10, merupakan penginisialisasi variabel **c** untuk mengurutkan data.
- vii. Pada baris 11, merupakan perintah untuk melakukan looping.
- viii. Pada baris 12, merupakan perintah untuk menampilkan isi variabel di baris 10.

(c) `numpy.lexsort()`

```

1      import numpy as np
2
3      a = np.array([9, 3, 1, 3, 4, 3, 6])
4
5      b = np.array([4, 6, 9, 2, 1, 8, 7])
6      print('column a, column b')
7      for (i, j) in zip(a, b):
8          print(i, ' ', j)
9
10     ind = np.lexsort((b, a))
11     print('Sorted indices->', ind)
12

```

**Listing 4.12** Lexsort

```

column a, column b
9    4
3    6
1    9
3    2
4    1
3    8
6    7
Sorted indices-> [2 3 1 5 4 6 0]

```

**Gambar 4.14** Lexsort

Penjelasan script 4.12 sebagai berikut.

- i. Pada baris 1, merupakan import module numpy dengan alias np. Pengaliansan disini digunakan agar mempermudah pemanggilan nama module.
- ii. Pada baris 3, merupakan penginisialisasi variabel **a** untuk elemen array.
- iii. Pada baris 5, merupakan penginisialisasi variabel **b** untuk elemen array.
- iv. Pada baris 6, merupakan perintah untuk menampilkan tulisan column a dan column b
- v. Pada baris 7, merupakan perintah untuk perulangan isi variabel di baris 3 dan baris 5.

- vi. Pada baris 8, merupakan perintah untuk menampilkan isi dari parameter di baris 7.
- vii. Pada baris 10, merupakan penginisialisasi variabel pengurutan dengan lexsort.
- viii. Pada baris 11, merupakan perintah untuk menampilkan isi variabel di baris 10.

## 8. Searching

Dalam numpy, juga menyediakan fitur untuk mencari tempat elemen dalam array yang ada. Setiap pencarian elemen dikatakan berhasil bergantung pada elemen yang dicari apakah dapat ditemukan atau tidak. Kita menggunakan beberapa fungsi dalam pencarian seperti `argmax`, `argmin`, `nanaargmax` dll.

### (a) `argmax`

`Argmax` merupakan sebuah fungsi untuk mencari elemen dengan nilai pengembalian indeks maksimal array dalam suatu sumbu tertentu.

```

1      import numpy as geek
2
3      array = geek.arange(12).reshape(3, 4)
4      print("INPUT ARRAY : \n", array)
5
6      print("\nMax element : ", geek.argmax(array))
7
8      print(("Indices of Max element : "
9            , geek.argmax(array, axis=0)))
10     print(("Indices of Max element : "
11           , geek.argmax(array, axis=1)))
12

```

**Listing 4.13** `Argmax`

```

INPUT ARRAY :
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]

Max element : 11

Indices of Max element : [2 2 2 2]

Indices of Max element : [3 3 3]

```

**Gambar 4.15** `Argmax`

Penjelasan script 4.13 seperti berikut.

- i. Pada baris 1, merupakan import module numpy dengan alias np. Pengaliansan disini digunakan agar mempermudah pemanggilan nama module.
- ii. Pada baris 3, merupakan penginisialisasi variabel **array** untuk elemen array.
- iii. Pada baris 4, merupakan perintah untuk menampilkan isi variabel di baris 3.
- iv. Pada baris 6, merupakan perintah untuk menampilkan data dengan fungsi argmax di variabel baris 3.
- v. Pada baris 8, merupakan perintah untuk menampilkan data dengan fungsi argmax dengan nilai axis 0.
- vi. Pada baris 10, merupakan perintah untuk menampilkan data dengan fungsi argmax dengan nilai axis 1.

(b) argmin

Argmin merupakan sebuah fungsi untuk mencari elemen dengan nilai pengembalian nilai minimum sepanjang sumbu tertentu.

```

1      import numpy as geek
2
3      array = geek.arange(8)
4      print("INPUT ARRAY : \n", array)
5
6      print("\nIndices of min element : ", geek.argmin(
7      array , axis=0))

```

**Listing 4.14** Argmin

```

INPUT ARRAY :
[0 1 2 3 4 5 6 7]

Indices of min element : 0

```

**Gambar 4.16** Argmin

Penjelasan script 4.14 seperti berikut.

- i. Pada baris 1, merupakan import module numpy dengan alias np. Pengaliansan disini digunakan agar mempermudah pemanggilan nama module.
- ii. Pada baris 3, merupakan penginisialisasi variabel **array** untuk elemen array.
- iii. Pada baris 4, merupakan perintah untuk menampilkan isi variabel di baris 3.



- iv. Pada baris 6, merupakan perintah untuk menampilkan data dengan fungsi `argmin` dengan nilai `axis 0`.

(c) `nanaargmax`

`nanaargmax` merupakan fungsi untuk mencari elemen dengan nilai pengembalian indeks elemen maksimal array pada sumbu tertentu yang mengabaikan nilai `NaN` yang terdapat pada elemen tersebut. Artinya pada fungsi tidak akan ada nilai `NaN` dalam pengembalian yang ditampilkan.

```

1      import numpy as geek
2
3      array = [geek.nan, 4, 2, 3, 1]
4      print("INPUT ARRAY 1 : \n", array)
5
6      array2 = geek.array([[geek.nan, 4], [1, 3]])
7      print("\nIndices of max in array1 : "
8            , geek.nanargmax(array))
9
10     print("\nINPUT ARRAY 2 : \n", array2)
11     print("\nIndices of max in array2 : "
12           , geek.nanargmax(array2))
13
14     print("\nIndices at axis 1 of array2 : "
15           , geek.nanargmax(array2, axis = 1))
16
17

```

**Listing 4.15** Nanaargmax

```

INPUT ARRAY 1 :
[nan, 4, 2, 3, 1]

Indices of max in array1 : 1

INPUT ARRAY 2 :
[[ nan  4.]
 [ 1.  3.]]

Indices of max in array2 : 1

Indices at axis 1 of array2 : [1 1]

```

**Gambar 4.17** Nanaargmax

Penjelasan script 4.15 seperti berikut.

- i. Pada baris 1, merupakan import module numpy dengan alias np. Pengaliansan disini digunakan agar mempermudah pemanggilan nama module.
- ii. Pada baris 3, merupakan penginisialisasi variabel **array** untuk elemen array.
- iii. Pada baris 4, merupakan perintah untuk menampilkan isi variabel di baris 3.
- iv. Pada baris 6, merupakan penginisialisasi variabel **array** untuk elemen array.
- v. Pada baris 7, merupakan perintah untuk menampilkan isi variabel di baris 3 dengan fungsi `nanaargmax`.
- vi. Pada baris 10, merupakan perintah untuk menampilkan isi dari variabel di baris 6.
- vii. Pada baris 11, merupakan perintah untuk menampilkan isi variabel di baris 6 dengan fungsi `nanaargmax`.
- viii. Pada baris 13, merupakan perintah untuk menampilkan isi variabel di baris 6 dengan fungsi `nanaargmax` dengan nilai axis 1.

## 9. Counting

Counting merupakan sebuah fungsi untuk menghitung nilai elemen tertentu dalam sebuah array. Contoh dari implementasi counting tersebut seperti berikut.

`count_nonzero`, dimana fungsi ini digunakan untuk menghitung nilai elemen yang tidak bernilai 0. Berikut contoh implementasinya.

```

1      # Python Program illustrating
2      # working of count_nonzero()
3
4      import numpy as np
5
6      # Counting a number of
7      # non-zero values
8      a = np.count_nonzero([[0,1,7,0,0],[3,0,0,2,19]])
9      b = np.count_nonzero(([[0,1,7,0,0],[3,0,0,2,19]]
10                          , axis=0))
11
12      print("Number of nonzero values is :",a)
13      print("Number of nonzero values is :",b)
14
```

**Listing 4.16** Nonzero

```

Number of nonzero values is : 5
Number of nonzero values is : [1, 1, 1, 1, 1]
```

**Gambar 4.18** Count NonZero

Penjelasan script 4.16 seperti berikut.

- (a) Pada baris 4, merupakan import module numpy dengan alias np. Pengaliansan disini digunakan agar mempermudah pemanggilan nama module.
- (b) Pada baris 8, merupakan penginisialisasi variabel **a** untuk elemen array.
- (c) Pada baris 9, merupakan penginisialisasi variabel **b** untuk elemen array.
- (d) Pada baris 12, merupakan perintah untuk menampilkan isi variabel di baris 8.
- (e) Pada baris 13, merupakan perintah untuk menampilkan isi variabel di baris 9.

#### 4.1.4 Pandas

Pandas adalah sebuah *library* yang sering digunakan dalam data science atau big data. Dimana *library* ini digunakan untuk menyediakan struktur data dan analisis data yang sangat mudah digunakan dalam bahasa pemrograman python. Atau lebih tepatnya, pandas untuk analisis dan struktur data yang diperlukan untuk membersihkan data dan ditampilkan dalam bentuk tabel.

Pandas juga dapat melakukan beberapa hal seperti menggabungkan data, menghilangkan data yang hilang, dan lain-lain. Oleh karena itu, pandas merupakan *library* yang wajib digunakan dalam pengolahan data tingkat tinggi atau biasa disebut statistik.

Pandas memiliki struktur dasar yaitu DataFrame. DataFrame itu sendiri merupakan sebuah kumpulan data berurutan. Jika diibaratkan DataFrame seperti database, dimana memiliki kolom dan baris serta setiap baris mewakili *record* data. Pandas bisa membaca berbagai macam ekstensi file, seperti **.csv**, **.txt**, **.tsv** dan lainnya.



Gambar 4.19 Pandas

### 4.1.5 Arsitektur DataFrame

	color	director_name	num_critic_for_reviews	duration	actor_2_facebook_likes	imdb_score	aspect_ratio	movie_facebook_likes
0	Color	James Cameron	723.0	178.0	...	936.0	7.9	1.78
1	Color	Gore Verbinski	302.0	169.0	...	5000.0	7.1	2.35
2	Color	Sam Mendes	602.0	148.0	...	393.0	6.8	2.35
3	Color	Christopher Nolan	813.0	164.0	...	23000.0	8.5	2.35
4	NaN	Doug Walker	NaN	NaN	...	12.0	7.1	NaN

**Gambar 4.20** Arsitektur DataFrame

Atribut :

1. Column
2. Column Name
3. More column to display
4. Index label
5. Index
6. Missing Value
7. Data

### 4.1.6 DataFrame

DataFrame itu sendiri merupakan struktur dasar dari pandas ini dan pengertian DataFrame juga sudah dijelaskan sebelumnya. Oleh karena itu disini akan menjelaskan DataFrame penggunaannya bagaimana dan bisa untuk apa saja.

Hal yang harus diperhatikan dalam menggunakan *library* ini adalah melakukan *import library* tersebut dengan cara :

```
1 import pandas as pd
```

**Listing 4.17** Import-Module

### 4.1.7 Understanding Data

**4.1.7.1 dtypes** Dalam DataFrame, kita ingin mengetahui tipe data apa yang digunakan dalam dataFrame tersebut. Fungsi yang digunakan adalah `dtypes`. Untuk pengimplementasiannya menggunakan script 4.23.

```
1 df.dtypes
```

**Listing 4.18** Dtypes

```
A    int64
B    int64
C    int64
dtype: object
```

**Gambar 4.21** dtypes

Berdasarkan gambar 4.21, menjelaskan bahwa pada kolom di dataframe tersebut menggunakan data type `int64` atau integer.

**4.1.7.2 Shape** Shape merupakan fungsi dalam dataframe untuk mengecek dimensi dalam dataframe itu berapa dimensi.

```
1 df.shape
```

**Listing 4.19** Shape

```
df.shape
(3, 3)
```

**Gambar 4.22** shape

Penjelasan script 4.19 seperti berikut.

1. Pada baris 1, merupakan perintah mengecek dimensi dataframe tersebut. Keluaran kode ini seperti gambar 4.22. Dimana arti dari gambar tersebut adalah kolom sebanyak 3 kolom dan sebanyak 3 baris.

## 4.1.8 Conversion

**4.1.8.1 isna** Isna digunakan untuk mendeteksi apakah terdapat nilai *missing value* dalam dataframe tersebut. Jika terdapat *missing value* akan bernilai **True**, sedangkan jika tidak terdapat *missing value* akan bernilai **False**.

Contoh implementasinya seperti berikut.

```
1 import pandas as pd
2 import numpy as np
```

```

3
4     ser = pd.Series([5, 6, np.NaN])
5     ser
6
7     ser.isna()

```

**Listing 4.20** isna

Penjelasan script 4.20 seperti berikut.

1. Pada baris 1, merupakan import module pandas dengan alias pd. Pengaliansan disini digunakan agar mempermudah pemanggilan nama module.
2. Pada baris 2, merupakan import module numpy dengan alias np. Pengaliansan disini digunakan agar mempermudah pemanggilan nama module.
3. Pada baris 4, merupakan penginisialisasi variabel untuk data frame.
4. Pada baris 5, merupakan perintah untuk menampilkan isi variabel di baris 4.

```

0      5.0
1      6.0
2      NaN
dtype: float64

```

**Gambar 4.23** Data Isna

5. Pada baris 7, merupakan penerapa fungsi isna.

```

0      False
1      False
2       True
dtype: bool

```

**Gambar 4.24** Isna

**4.1.8.2 notna** Notna digunakan untuk mendeteksi apakah terdapat nilai *missing value* dalam dataframe tersebut. Jika terdapat *missing value* akan bernilai **True**, sedangkan jika tidak terdapat *missing value* akan bernilai **False**.

Contoh implementasinya seperti berikut.

```

1     import pandas as pd
2     import numpy as np
3

```

```

4     ser = pd.Series([5, 6, np.NaN])
5     ser
6
7     ser.notna()

```

**Listing 4.21** isna

Penjelasan script 4.21 seperti berikut.

1. Pada baris 1, merupakan import module pandas dengan alias pd. Pengaliansan disini digunakan agar mempermudah pemanggilan nama module.
2. Pada baris 2, merupakan import module numpy dengan alias np. Pengaliansan disini digunakan agar mempermudah pemanggilan nama module.
3. Pada baris 4, merupakan penginisialisasi variabel untuk data frame.
4. Pada baris 5, merupakan perintah untuk menampilkan isi variabel di baris 4.

```

0     5.0
1     6.0
2     NaN
dtype: float64

```

**Gambar 4.25** Data Notna

5. Pada baris 7, merupakan penerapan fungsi notna.

```

0     False
1     False
2     True
dtype: bool

```

**Gambar 4.26** notna

**4.1.8.3 copy** Fungsi ini digunakan untuk membuat salinan data yang telah ada.

```

1     DataFrame.copy(self, deep=True)

```

**Listing 4.22** copy

```
ser = pd.Series([5, 6, np.NaN])
ser
```

0	5.0
1	6.0
2	NaN

```
dtype: float64
```

Gambar 4.27 Data Asli

Penjelasan script 4.22 seperti berikut.

1. Deep, Jika bernilai True dan secara default deep bernilai True. Artinya Objek baru akan dibuat dengan salinan data sesuai data yang asli. Dan ketika data hasil salinan ketika di modifikasi tidak akan mempengaruhi data yang asli. Jika bernilai false, ketika data hasil salinan di modifikasi akan mempengaruhi data yang asli.

```
ser_cp = ser.copy()
ser_cp
```

0	5.0
1	6.0
2	NaN

```
dtype: float64
```

Gambar 4.28 Data Copy

### 4.1.9 Indexing Data

**4.1.9.1 at** Dalam DataFrame, kita memanipulasi data dimana ingin menampilkan data di satu kolom tertentu dan baris tertentu. Fungsi tersebut bernama **at**. Mungkin bagi kalian yang sering mendengar **loc**, karena pada dasarnya fungsi kedua hal tersebut memiliki kegunaan yang sama. Untuk contoh implementasinya seperti berikut.

```
1 import pandas as pd
2
3 df = pd.DataFrame([[0, 2, 3], [0, 4, 1], [10, 20, 30]],
4                   index=[4, 5, 6], columns=['A', 'B', 'C'])
5 df
6
```



```
df.at[4, 'B']
```

**Listing 4.23** Data

	A	B	C
4	0	2	3
5	0	4	1
6	10	20	30

**Gambar 4.29** DataFrame

Penjelasan script 4.23 seperti berikut.

1. Pada baris 1, merupakan import module pandas dengan alias `pd`. Pengalisan disini digunakan agar mempermudah pemanggilan nama module.
2. Pada baris 3, merupakan penginisialisasi variabel untuk data frame.
3. Pada baris 4, merupakan pemanggilan variabel di baris 3 untuk menampilkan isi dataFrame tersebut. Hasilnya bisa dilihat pada gambar 4.29.
4. Pada baris 6, merupakan penerapan fungsi `at` untuk menampilkan data di kolom B dengan index / baris 4. Keluaran dari kode ini adalah **2**.

Untuk membuktikan bahwa fungsi `at` dengan fungsi `loc` memiliki kegunaan yang, bisa dilihat pada gambar 4.30.

```
df.at[4, 'B']
```

2

```
df.loc[4].at['B']
```

2

**Gambar 4.30** loc

**4.1.9.2 iat** Dalam DataFrame, kita memanipulasi data dimana ingin menampilkan data di satu kolom tertentu dan baris tertentu dengan inputan parameter tiap kolom dan barisnya adalah integer. Fungsi tersebut bernama **iat**. Mungkin bagi kalian yang sering mendengar **iloc**, karena pada dasarnya fungsi kedua hal tersebut memiliki kegunaan yang sama. Untuk contoh implementasinya seperti berikut.

```
1 import pandas as pd
2
3 df = pd.DataFrame([[0, 2, 3], [0, 4, 1], [10, 20, 30]],
4                   index=[4, 5, 6], columns=['A', 'B', 'C'])
5 df
6
7 df.iat[0, 1]
```

**Listing 4.24** Data

	A	B	C
4	0	2	3
5	0	4	1
6	10	20	30

**Gambar 4.31** DataFrame

Penjelasan script 4.24 seperti berikut.

1. Pada baris 1, merupakan import module pandas dengan alias pd. Pengaliansan disini digunakan agar mempermudah pemanggilan nama module.
2. Pada baris 3, merupakan penginisialisasi variabel untuk data frame.
3. Pada baris 4, merupakan pemanggilan variabel di baris 3 untuk menampilkan isi dataFrame tersebut. Hasilnya bisa dilihat pada gambar 4.31.
4. Pada baris 6, merupakan penerapan fungsi **iat** untuk menampilkan data di kolom index 1 dengan baris index 0. Keluaran dari kode ini adalah **2**.

Untuk membuktikan bahwa fungsi **iat** dengan fungsi **iloc** memiliki kegunaan yang, bisa dilihat pada gambar 4.32.

```
df.iat[0, 1]
2
```

---

```
df.loc[4].iat[1]
2
```

Gambar 4.32 iat

**4.1.9.3 loc** loc merupakan sebuah fungsi untuk mengakses data dengan nilai pengembalian lebih dari 1 nilai. Ini berbeda dengan at yang mengembalikan nilai hanya 1 nilai keluaran.

```
1 import pandas as pd
2
3 df = pd.DataFrame([[0, 2, 3], [0, 4, 1], [10, 20, 30]],
4                   index=[4, 5, 6], columns=['A', 'B', 'C'])
5 df
6
7 df.loc['4']
```

Listing 4.25 Loc

```
df.loc[4]
```

A	0
B	2
C	3

Name: 4, dtype: int64

Gambar 4.33 loc

Penjelasan script 4.25 seperti berikut.

1. Pada baris 1, merupakan import module pandas dengan alias pd. Pengaliansan disini digunakan agar mempermudah pemanggilan nama module.
2. Pada baris 3, merupakan penginisialisasi variabel untuk data frame.

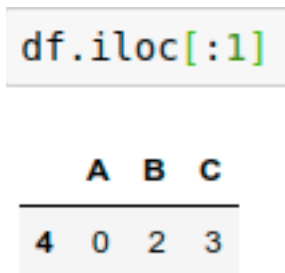
3. Pada baris 4, merupakan pemanggilan variabel di baris 3 untuk menampilkan isi dataFrame tersebut. Hasilnya bisa dilihat pada gambar 4.31.
4. Pada baris 7, merupakan penerapan fungsi **iloc** untuk menampilkan data. Keluaran dari kode ini seperti gambar 4.33.

**4.1.9.4 iloc** *iloc* merupakan sebuah fungsi untuk mengakses data dengan nilai pengembalian lebih dari 1 nilai dengan parameter inputan nilai index. Ini berbeda dengan *at* yang mengembalikan nilai hanya 1 nilai keluaran.

```

1  import pandas as pd
2
3  df = pd.DataFrame([[0, 2, 3], [0, 4, 1], [10, 20, 30]],
4                    index=[4, 5, 6], columns=['A', 'B', 'C'])
5  df
6
7  df.iloc[:1]
```

**Listing 4.26** *iloc*



	A	B	C
4	0	2	3

**Gambar 4.34** *iloc*

Penjelasan script 4.26 seperti berikut.

1. Pada baris 1, merupakan import module pandas dengan alias *pd*. Pengaliansan disini digunakan agar mempermudah pemanggilan nama module.
2. Pada baris 3, merupakan penginisialisasi variabel untuk data frame.
3. Pada baris 4, merupakan pemanggilan variabel di baris 3 untuk menampilkan isi dataFrame tersebut. Hasilnya bisa dilihat pada gambar 4.31.
4. Pada baris 7, merupakan penerapan fungsi **iloc** untuk menampilkan data. Keluaran dari kode ini seperti gambar 4.34.

**4.1.9.5 Head** *Head* merupakan fungsi untuk menampilkan data dengan pengembalian nilai *n* data pertama.

```

1  DataFrame.head(self, n=5)
```

**Listing 4.27** *Head*

Penjelasan script 4.27 seperti berikut.

1. n, n diisi dengan angka. Artinya jika kita mengisi dengan nilai 2, maka keluarannya berupa 2 data pertama.

```
df.head(2)
```

	Animal	Max Speed
0	Falcon	380.0
1	Falcon	370.0

Gambar 4.35 head

#### 4.1.10 Transposing

**4.1.10.1 Transpose** Dalam dataframe, juga terdapat fungsi untuk mentransformasi dari kolom menjadi baris dan sebaliknya. Fungsi yang digunakan itu adalah **Transpose**.

```
1 import pandas as pd
2
3 d1 = {'col1': [1, 2], 'col2': [3, 4]}
4 df1 = pd.DataFrame(data=d1)
5 df1
```

Listing 4.28 Sebelum Transpose

	col1	col2
0	1	3
1	2	4

Gambar 4.36 Sebelum Transpose

Penjelasan script 4.28 seperti berikut.

1. Pada baris 1, merupakan import module pandas dengan alias pd. Pengaliansan disini digunakan agar mempermudah pemanggilan nama module.

2. Pada baris 3, merupakan penginisialisasi variabel untuk data array.
3. Pada baris 4, merupakan penginisialisasi variabel untuk pembuatan dataframe dengan membaca data di variabel baris 3.
4. Pada baris 5, merupakan pemanggilan variabel di baris 3 yang sudah membaca data di baris 4. Dan hasil membaca data akan ditampilkan pada gambar 4.36.

```
1 df1_transposed = df1.T
2 df1_transposed
```

**Listing 4.29** Sesudah Transpose

	0	1
col1	1	2
col2	3	4

**Gambar 4.37** Sesudah Transpose

Penjelasan script 4.29 seperti berikut.

1. Pada baris 1, merupakan penginisialisasi variabel untuk pembuatan tranpose dengan membaca data di variabel baris 3 pada script 4.28.
2. Pada baris 2, merupakan pemanggilan variabel di baris 1. Dan hasil membaca data akan ditampilkan pada gambar 4.37.

**4.1.10.2 pivot** Fungsi ini digunakan untuk membentuk kembali struktur dataframe.

```
1 DataFrame.pivot(self, index=None, columns=None, values=None)
```

**Listing 4.30** Pivot

Penjelasan script 4.30 seperti berikut.

1. index, Secara default akan bernilai None. Index disini digunakan untuk membuat index pada dataframe.
2. columns, secara default bernilai None. columns disini digunakan untuk membuat kolom pada dataframe.
3. values, values digunakan untuk menampilkan elemen yang ada pada dataframe.

Contoh implementasinya seperti berikut.

```

1  import pandas as pd
2
3  df = pd.DataFrame({'foo': ['one', 'one', 'one', 'two', 'two',
4                             'two'],
5                     'bar': ['A', 'B', 'C', 'A', 'B', 'C'],
6                     'baz': [1, 2, 3, 4, 5, 6],
7                     'zoo': ['x', 'y', 'z', 'q', 'w', 't']})
8
9  df
10 df.pivot(index='foo', columns='bar', values='baz')
11
12 df.pivot(index='foo', columns='bar')['baz']
13
14 df.pivot(index='foo', columns='bar', values=['baz', 'zoo'])

```

**Listing 4.31** Kasus Pivot

Penjelasan script 4.31 seperti berikut.

1. Pada baris 1, merupakan import module pandas dengan alias pd. Pengaliansan disini digunakan agar mempermudah pemanggilan nama module.
2. Pada baris 3, merupakan penginisialisasi variabel untuk data frame.
3. Pada baris 8, perintah untuk menampilkan isi variabel di baris 3.
4. Pada baris 10, merupakan perintah menerapkan pivot.

bar	A	B	C
foo			
one	1	2	3
two	4	5	6

**Gambar 4.38** Pivot 1

5. Pada baris 12, merupakan perintah menerapkan pivot.

bar	A	B	C
foo			
one	1	2	3
two	4	5	6

Gambar 4.39 Pivot 2

6. Pada baris 14, merupakan perintah menerapkan pivot.

	baz			zoo		
bar	A	B	C	A	B	C
foo						
one	1	2	3	x	y	z
two	4	5	6	q	w	t

Gambar 4.40 Pivot 2

**4.1.10.3 Pivot table** Fungsi ini digunakan membuat seperti spreadsheet dalam dataframe. Dengan pivot tabel bisa melakukan objek hirarki pada baris dan kolom di dataframe.

```
DataFrame.pivot_table(self, values=None, index=None, columns=None,
                        aggfunc='mean', fill_value=None, margins=False, dropna=True,
                        margins_name='All', observed=False)
```

Listing 4.32 Pivot Table

Penjelasan script 4.32 seperti berikut.

1. index, Secara default akan bernilai None. Index disini digunakan untuk membuat index pada dataframe.
2. columns, secara default bernilai None. columns disini digunakan untuk membuat kolom pada dataframe.
3. values, values digunakan untuk menampilkan elemen yang ada pada dataframe.
4. aggfunc, untuk menentukan isi elemen pada dataframe tersebut apa. Secara default bernilai **mean**.



5. `fill_value`, berfungsi untuk menggantikan nilai yang kosong pada dataframe.
6. `margins`, berfungsi untuk menambahkan kolom atau baris baru. Secara default bernilai **False**
7. `dropna`, berfungsi untuk menghilangkan nilai NaN pada dataframe. Secara default bernilai **True**.
8. `margin_name`, berfungsi untuk memberi nama pada kolom pada dataframe. Secara default bernilai **ALL**.
9. `observed`, ini berlaku jika pengelompokan bersifat kategorial. Artinya jika bernilai **True**, maka hanya menampilkan nilai dalam kelompok tersebut. Sedangkan jika **False**, maka menampilkan semua nilai.

Contoh penerapan `fillna` seperti berikut.

```

1  import pandas as pd
2  import numpy as np
3
4  df = pd.DataFrame({"A": ["foo", "foo", "foo", "foo", "foo",
5                           "bar", "bar", "bar", "bar"],
6                     "B": ["one", "one", "one", "two", "two",
7                           "one", "one", "two", "two"],
8                     "C": ["small", "large", "large", "small",
9                           "small", "large", "small", "small",
10                          "large"],
11                     "D": [1, 2, 2, 3, 3, 4, 5, 6, 7],
12                     "E": [2, 4, 5, 5, 6, 6, 8, 9, 9]})
13
14  df
15
16  table = pd.pivot_table(df, values='D', index=['A', 'B'],
17                          columns=['C'], aggfunc=np.sum)
18  table
19
20  table = pd.pivot_table(df, values='D', index=['A', 'B'],
21                          columns=['C'], aggfunc=np.sum, fill_value=0)
22  table
23
24  table = pd.pivot_table(df, values=['D', 'E'], index=['A', 'C'],
25                          aggfunc={'D': np.mean,
26                                   'E': np.mean})
27  table
28
29  table = pd.pivot_table(df, values=['D', 'E'], index=['A', 'C'],
30                          aggfunc={'D': np.mean,
31                                   'E': [min, max, np.mean]})
32  table

```

**Listing 4.33** Fillna

Penjelasan script 4.33 seperti berikut.

1. Pada baris 1, merupakan import module `pandas` dengan alias `pd`. Pengaliansan disini digunakan agar mempermudah pemanggilan nama module.

2. Pada baris 2, merupakan import module numpy dengan alias np. Pengaliansan disini digunakan agar mempermudah pemanggilan nama module.
3. Pada baris 4, merupakan penginisialisasi variabel untuk data frame.
4. Pada baris 14, merupakan perintah untuk menampilkan isi variabel di baris 4.

	A	B	C	D	E
0	foo	one	small	1	2
1	foo	one	large	2	4
2	foo	one	large	2	5
3	foo	two	small	3	5
4	foo	two	small	3	6
5	bar	one	large	4	6
6	bar	one	small	5	8
7	bar	two	small	6	9
8	bar	two	large	7	9

**Gambar 4.41** Data Pivot

5. Pada baris 16, merupakan penginisialisasi variabel untuk data frame dengan penerapan pivot table.
6. Pada baris 18, merupakan perintah untuk menampilkan isi variabel di baris 16.

		C	large	small
A	B			
bar	one	4.0	5.0	
	two	7.0	6.0	
foo	one	4.0	1.0	
	two	NaN	6.0	

Gambar 4.42 Pivot Table Aggfunc

7. Pada baris 20, merupakan penginisialisasi variabel untuk data frame dengan penerapan pivot table dengan fill\_value.
8. Pada baris 22, merupakan perintah untuk menampilkan isi variabel di baris 20.

		C	large	small
A	B			
bar	one	4	5	
	two	7	6	
foo	one	4	1	
	two	0	6	

Gambar 4.43 Pivot Table Fill\_value

9. Pada baris 24, merupakan penginisialisasi variabel untuk data frame dengan penerapan pivot table.
10. Pada baris 27, merupakan perintah untuk menampilkan isi variabel di baris 20.

		D	E
A	C		
bar	large	5.500000	7.500000
	small	5.500000	8.500000
foo	large	2.000000	4.500000
	small	2.333333	4.333333

Gambar 4.44 Pivot Table Aggfunc2

11. Pada baris 29, merupakan penginisialisasi variabel untuk data frame dengan penerapan pivot table.
12. Pada baris 32, merupakan perintah untuk menampilkan isi variabel di baris 29.

		D	E		
		mean	max	mean	min
A	C				
bar	large	5.500000	9.0	7.500000	6.0
	small	5.500000	9.0	8.500000	8.0
foo	large	2.000000	5.0	4.500000	4.0
	small	2.333333	6.0	4.333333	2.0

Gambar 4.45 Pivot Table Aggfunc2

### 4.1.11 Missing Data

**4.1.11.1 Dropna** Salah satu fungsi yang disediakan oleh pandas untuk mengatasi *missing value* adalah menggunakan **dropna**.

```
1 DataFrame.dropna(self, axis=0, how='any', thresh=None, subset=
  None, inplace=False)
```

Listing 4.34 Dropna

Penjelasan script 4.34 seperti berikut.

1. `axis`, Jika nilai `axis` 0, maka itu akan menghapus baris data. Sedangkan jika `axis` 1, maka akan menghapus kolom data.
2. `how`, secara default bernilai `ANY`. Artinya semua nilai elemen yang bernilai `NA` atau `NaN` pada baris dan kolom akan dihapus. ketika bernilai `ALL`, maka tidak ada baris atau kolom yang dihapus.
3. `thresh`, berfungsi untuk menghapus nilai yang missing. Parameter masukannya berupa angka atau integer. misal memasukkan angka 2, maka akan menghapus nilai elemen sebanyak 2 yang terdapat kolom dan baris di dataframe tersebut.
4. `subset`, berfungsi untuk menghapus data yang missing value yang disertai pemberian nama kolom dan melakukan pengindeksisasi pada keluaran data tersebut.
5. `inplace`, berfungsi untuk melakukan penghapusan data yang missing value dengan kondisi. Jika kondisi `True`, maka tidak ada nilai pengembalian artinya tidak ada data yang ditampilkan walaupun sebenarnya ada data yang tidak missing value. Sedangkan jika kondisi `False`, maka akan ada data yang dikembalikan dan ditampilkan pada keluarannya.

```

1  import pandas as pd
2
3  df = pd.DataFrame({'name': [ 'Alfred', 'Batman', 'Catwoman' ],
4                        "toy": [np.nan, 'Batmobile', 'Bullwhip' ],
5                        "born": [pd.NaT, pd.Timestamp("1940-04-25"),
6                                pd.NaT]})
7
8  df
9
10 df.dropna()
11
12 df.dropna(axis='columns')
13
14 df.dropna(how='all')
15 df.dropna(how='any')
16
17 df.dropna(thresh=2)
18
19 df.dropna(subset=['name', 'born'])
20
21 df.dropna(inplace=True)
22 df.dropna(inplace=False)

```

**Listing 4.35** Kasus Dropna

Penjelasan script 4.35 seperti berikut.

1. Pada baris 1, merupakan import module `pandas` dengan alias `pd`. Pengaliansan disini digunakan agar mempermudah pemanggilan nama module.
2. Pada baris 3, merupakan penginisialisasi variabel untuk data frame.
3. Pada baris 7, merupakan perintah untuk menampilkan isi variabel di baris 3.

	name	toy	born
0	Alfred	NaN	NaT
1	Batman	Batmobile	1940-04-25
2	Catwoman	Bullwhip	NaT

Gambar 4.46 Data Dropna

4. Pada baris 9, merupakan penerapan dropna.

	name	toy	born
1	Batman	Batmobile	1940-04-25

Gambar 4.47 Dropna

5. Pada baris 11, merupakan penerapan dropna dengan axis yang ingin dihapus adalah kolom.

```
df.dropna(axis='columns')
```

	name
0	Alfred
1	Batman
2	Catwoman

Gambar 4.48 Dropna Axis

6. Pada baris 13, merupakan penerapan dropna dengan how yang bernilai all.

```
df.dropna(how='all')
```

	name	toy	born
0	Alfred	NaN	NaT
1	Batman	Batmobile	1940-04-25
2	Catwoman	Bullwhip	NaT

**Gambar 4.49** Dropna How ALL

7. Pada baris 14, merupakan penerapan dropna dengan how yang bernilai any.

```
df.dropna(how='any')
```

	name	toy	born
1	Batman	Batmobile	1940-04-25

**Gambar 4.50** Dropna How ANY

8. Pada baris 16, merupakan penerapan dropna dengan thresh bernilai 2.

```
df.dropna(thresh=2)
```

	name	toy	born
1	Batman	Batmobile	1940-04-25
2	Catwoman	Bullwhip	NaT

**Gambar 4.51** Dropna Thresh

9. Pada baris 18, merupakan penerapan dropna dengan subset.

```
df.dropna(subset=['name', 'born'])
```

	name	toy	born
1	Batman	Batmobile	1940-04-25

**Gambar 4.52** Dropna Subset

10. Pada baris 20, merupakan penerapan dropna dengan kondisi inplace **True**.

```
df.dropna(inplace=True)
```

**Gambar 4.53** Dropna Inplace True

11. Pada baris 21, merupakan penerapan dropna dengan kondisi inplace **False**.

```
df.dropna(inplace=False)
```

	name	toy	born
1	Batman	Batmobile	1940-04-25

**Gambar 4.54** Dropna Inplace False

**4.1.11.2 fillna** Berfungsi untuk mengisi nilai yang missing. Dan biasanya untuk mengisi nilai yang missing dengan nilai rata-rata pada suatu data di dataframe. Jika nilai missing itu sendiri dibiarkan begitu saja, akan mempengaruhi hasil analisis nantinya. Oleh karena itu, fungsi **fillna** dibutuhkan untuk mengatasi masalah tersebut.

```
DataFrame.fillna(self, value=None, method=None, axis=None,
inplace=False, limit=None, downcast=None, **kwargs)
```

**Listing 4.36** fillna

Penjelasan script 4.36 seperti berikut.

1. value, ini digunakan untuk mengisi nilai yang missing dengan memanggil sebuah kolom dalam dataframe atau variabel yang telah dideklarasikan.



2. method, metode untuk mengisi nilai yang missing dalam data dan secara default bernilai **None**.
3. axis, merupakan fungsi untuk mengisi data dengan nilai 0 atau 1. Jika 0, maka mengisi data pada index. Sedangkan 1, mengisi data pada kolom.
4. inplace, berfungsi untuk melakukan mengisi data yang missing value dengan kondisi. Jika kondisi True, maka tidak ada nilai pengembalian artinya tidak ada data yang ditampilkan walaupun sebenarnya ada data yang tidak missing value. Sedangkan jika kondisi False, maka akan ada data yang dikembalikan dan ditampilkan pada keluarannya.
5. limit, ini digunakan untuk membatasi pengisian nilai yang missing dalam data tersebut sampai baris data tertentu. Secara default bernilai **None**.
6. downcast, ini digunakan untuk merubah jenis tipe data atau menurunkan jenis tipe data yang memiliki korelasi yang sama. Seperti Float ke Integer.

Contoh implementasi seperti berikut.

```

1  import pandas as pd
2  import numpy as np
3  df = pd.DataFrame([[np.nan, 2, np.nan, 0],
4                    [3, 4, np.nan, 1],
5                    [np.nan, np.nan, np.nan, 5],
6                    [np.nan, 3, np.nan, 4]],
7                    columns=list('ABCD'))
8
9  df
10
11 df.fillna(0)
12
13 df.fillna(method='ffill')
14
15 values = {'A': 0, 'B': 1, 'C': 2, 'D': 3}
16 df.fillna(value=values)
17
18 df.fillna(value=values, limit=1)

```

**Listing 4.37** Kasus Fillna

Penjelasan script 4.37 seperti berikut.

1. Pada baris 1, merupakan import module pandas dengan alias pd. Pengaliansan disini digunakan agar mempermudah pemanggilan nama module.
2. Pada baris 2, merupakan import module numpy dengan alias np. Pengaliansan disini digunakan agar mempermudah pemanggilan nama module.
3. Pada baris 3, merupakan penginisialisasi variabel untuk data.
4. Pada baris 8, merupakan perintah untuk menampilkan isi variabel di baris 3.

	A	B	C	D
0	NaN	2.0	NaN	0
1	3.0	4.0	NaN	1
2	NaN	NaN	NaN	5
3	NaN	3.0	NaN	4

**Gambar 4.55** Data Missing

5. Pada baris 10, merupakan penerapan fillna dimana nilai yang missing akan di isi dengan nilai 0.

	A	B	C	D
0	0.0	2.0	0.0	0
1	3.0	4.0	0.0	1
2	0.0	0.0	0.0	5
3	0.0	3.0	0.0	4

**Gambar 4.56** Fillna 0

6. Pada baris 12, merupakan penerapan fillna dimana nilai yang missing di isi dengan method ffill.

	A	B	C	D
0	NaN	2.0	NaN	0
1	3.0	4.0	NaN	1
2	3.0	4.0	NaN	5
3	3.0	3.0	NaN	4

Gambar 4.57 Method Ffill

7. Pada baris 14, merupakan penginisialisasi variabel untuk mengisi nilai yang missing.
8. Pada baris 15, merupakan penerapan fillna dengan mengisi nilai yang missing dengan menggunakan variabel di baris 14.

	A	B	C	D
0	0.0	2.0	2.0	0
1	3.0	4.0	2.0	1
2	0.0	1.0	2.0	5
3	0.0	3.0	2.0	4

Gambar 4.58 Fillna Variabel

9. Pada baris 15, merupakan penerapan fillna dengan mengisi nilai yang missing dengan menggunakan variabel di baris 14 dengan kondisi limit bernilai 1.

	A	B	C	D
0	0.0	2.0	2.0	0
1	3.0	4.0	NaN	1
2	NaN	1.0	NaN	5
3	NaN	3.0	NaN	4

**Gambar 4.59** Fillna Limit

## 4.1.12 Grouping

**4.1.12.1 Groupby** Groupby merupakan sebuah fungsi dalam pandas untuk melakukan grouping data berdasarkan suatu parameter tertentu.

```
DataFrame.groupby(self, by=None, axis=0, level=None, as_index=
True, sort=True, group_keys=True, squeeze=False, observed=False,
**kwargs)
```

**Listing 4.38** Groupby Kode

Penjelasan script 4.38 seperti berikut.

1. DataFrame, merupakan nama fungsi dalam module pandas yang sudah kita import sebelumnya.
2. groupby, merupakan fungsi untuk melakukan grouping data.
3. self, merupakan sebuah variabel yang telah di inisialisasi sebelumnya tetapi self disini tidak harus variabel.
4. by, merupakan elemen yang digunakan dalam grouping data dengan memasukkan parameter tertentu sebagai dasar grouping data.
5. axis, merupakan fungsi untuk memisahkan data dengan nilai 0 atau 1. Jika 0, maka memisahkan data dengan index. Sedangkan 1, memisahkan data dengan kolom. Secara default, axis akan bernilai 0.
6. Level, ini digunakan ketika menampilkan data secara hirarki. Secara default, nilai level adalah None.
7. as\_index, ini digunakan untuk membuat dalam dataframe tersebut menggunakan index atau tidak. Secara default bernilai True.

8. Sort, ini digunakan untuk mengurutkan data dan secara default bernilai True. Ini tidak mempengaruhi urutan dalam masing-masing kelompok.
9. group\_key, ini digunakan untuk penambahan kunci dalam mengindeksi untuk proses identifikasi bagian yang ada. Secara default bernilai True.
10. squeeze, Ini bisa mengurangi dimensi pengembalian jika memungkinkan. Jika tidak, maka kembali ke dimensi yang semula. Secara default bernilai False.
11. observed, ini berlaku jika pengelompokan bersifat kategorial. Artinya jika bernilai True, maka hanya menampilkan nilai dalam kelompok tersebut. Sedangkan jika False, maka menampilkan semua nilai.
12. \*\*kwargs, ini hanya opsional saja.

Contoh penerapan dari groupby seperti berikut.

```

1  import pandas as pd
2
3  df = pd.DataFrame({'Animal': ['Falcon', 'Falcon',
4                               'Parrot', 'Parrot'],
5                     'Max Speed': [380., 370., 24., 26.]})
6  df
7
8  df.groupby(['Animal']).mean()
```

**Listing 4.39** Kasus Groupby

Penjelasan script 4.39 seperti berikut.

1. Pada baris 1, merupakan import module pandas dengan alias pd. Pengaliansan disini digunakan agar mempermudah pemanggilan nama module.
2. Pada baris 3, merupakan penginisialisasi variabel untuk data frame.
3. Pada baris 6, merupakan perintah untuk menampilkan isi variabel di baris 3.

	Animal	Max Speed
0	Falcon	380.0
1	Falcon	370.0
2	Parrot	24.0
3	Parrot	26.0

**Gambar 4.60** Data variabel

4. Pada baris 9, merupakan perintah untuk pengelompokkan berdasarkan **Animal** serta mencari nilai rata-rata dari nilai **max speed** di masing-masing kelompok.

Max Speed	
Animal	
Falcon	375.0
Parrot	25.0

Gambar 4.61 Groupby

```

1  import pandas as pd
2
3  arrays = [['Falcon', 'Falcon', 'Parrot', 'Parrot'],
4            ['Captive', 'Wild', 'Captive', 'Wild']]
5
6  index = pd.MultiIndex.from_arrays(arrays, names=('Animal', 'Type'))
7
8  df = pd.DataFrame({'Max Speed': [390., 350., 30., 20.]},
9                    index=index)
10
11 df
12
13 df.groupby(level=0).mean()
14
15 df.groupby(level=1).mean()

```

Listing 4.40 Kasus 2 Groupby

Penjelasan script 4.40 seperti berikut.

1. Pada baris 1, merupakan import module pandas dengan alias pd. Pengalisan disini digunakan agar mempermudah pemanggilan nama module.
2. Pada baris 3, merupakan penginisialisasi variabel untuk data frame.
3. Pada baris 6, merupakan inisiasi variabel untuk mengindeks dataframe.
4. Pada baris 8, merupakan inisiasi variabel untuk dataframe.
5. Pada baris 11, merupakan perintah untuk menampilkan data dari variabel pada baris 8.

Max Speed		
Animal	Type	
Falcon	Captive	390.0
	Wild	350.0
Parrot	Captive	30.0
	Wild	20.0

Gambar 4.62 Data multiindex

6. Pada baris 13, merupakan perintah untuk mengelompokkan data berdasarkan level 0 serta mencari nilai rata-rata di kolom **max speed**.

Max Speed	
Animal	
Falcon	370.0
Parrot	25.0

Gambar 4.63 Groupby Hirarki 1

7. Pada baris 15, merupakan perintah untuk mengelompokkan data berdasarkan level 1 serta mencari nilai rata-rata di kolom **max speed**.

Max Speed	
Type	
Captive	210.0
Wild	185.0

Gambar 4.64 Groupby Hirarki 2

### 4.1.13 Deskripsi

**4.1.13.1 *corr*** Fungsi ini digunakan untuk menghitung korelasi kolom yang berpasangan. Namun disini tidak menghitung elemen yang bernilai NaN atau null.

```
1 DataFrame.corr(self, method='pearson', min_periods=1)
```

**Listing 4.41** Corr

Penjelasan script 4.41 seperti berikut.

1. *method*, Dalam *method* ini terdapat 3 jenis. Pertama, *pearson*. Artinya mencari nilai koefisien korelasi tiap kolom yang secara standart. Kedua, *kendall*. Artinya mencari nilai koefisien korelasi tiap kolom secara *kendall*. Ketiga, *spearman*. Artinya mencari nilai korelasi secara peringkat *spearman* tiap kolomnya.
2. *min\_periods*, merupakan jumlah pengamatan setiap pasang dan ini bersifat opsional saja.

Contoh implementasinya seperti berikut.

```
1 import pandas as pd
2
3 df = pd.DataFrame([(0.2, 0.3), (0, 0.6), (0.6, 0), (0.2, 0.1)],
4                   columns=['dogs', 'cats'])
5 df
6
7 df.corr(method='pearson')
8
9 df.corr(method='kendall')
10
11 df.corr(method='spearman')
```

**Listing 4.42** Corr Kasus

Penjelasan script 4.42 seperti berikut.

1. Pada baris 1, merupakan import module *pandas* dengan alias *pd*. Pengaliansan disini digunakan agar mempermudah pemanggilan nama module.
2. Pada baris 3, merupakan penginisialisasi variabel untuk data frame.
3. Pada baris 5, merupakan perintah untuk menampilkan isi variabel di baris 3.
4. Pada baris 7, merupakan penerapan *corr* dengan metode *pearson*.
5. Pada baris 9, merupakan penerapan *corr* dengan metode *kendall*.
6. Pada baris 11, merupakan penerapan *corr* dengan metode *spearman*.



### 4.1.14 Contoh Kasus

#### 1. Membuat DataFrame.

Pandas yang pertama bisa membuat dataframe, dimana bisa menampilkan data yang ke dalam bentuk tabel.

```
1 test = pd.read_csv('products.csv')
2 test_DF = pd.DataFrame(test)
3 test_DF
4
```

**Listing 4.43** Create DataFrame

Pada kode 4.43, pada baris pertama untuk membaca data yang ada bentuk **.csv**, sedangkan untuk baris kedua untuk membuat data yang telah dibaca pada baris pertama menjadi tabel dan pada baris ketiga untuk menampilkan kode baris kedua. Untuk melihat hasil penerapan kode diatas bisa dilihat pada gambar berikut :

```
In [4]: test = pd.read_csv('products.csv')
test_DF = pd.DataFrame(test)
test_DF
```

Out[4]:

	productid	title
0	1	TOSHIBA Smart HD LED TV 32" - 32L5650VJ Free B...
1	2	TOSHIBA Full HD Smart LED TV 40" - 40L5650VJ ~...
2	3	Samsung 40 Inch Full HD Flat LED Digital TV 40...
3	4	Sharp HD LED TV 24" - LC-24LE175I - Hitam
4	5	Lenovo Ideapad 130-15AST LAPTOP MULTIMEDIA I A...
...	...	...
2495	109487	Flashdisk Hayabusa Toshiba 64GB/ Flash Disk / F...
2496	111362	Flashdisk Toshiba USB [8 GB]
2497	111759	Rimas COD Combo Multi Card Reader + 3 USB HUB ...
2498	112556	Disket/Diskette Floppy Disk Maxell MF2HD Forma...
2499	112852	Toshiba Canvio Basics 2TB - HDD / HD / Hardisk...

2500 rows x 2 columns

**Gambar 4.65** Create DataFrame

#### 2. Menampilkan Kolom yang Terpilih

Pandas juga bisa menampilkan kolom yang terpilih dari data tersedia.

```
1 test_DF[['title']]
2
```

**Listing 4.44** Selection Column

Pada kode 4.44, menjelaskan dimana mengambil variabel diatas lalu untuk mengambil kolom apa yang ditampilkan dengan memasukkan nama kolom yang mau ditampilkan seperti kode tersebut. Untuk menampilkan hasil penerapan nya bisa dilihat digambar berikut :

```
In [6]: test_DF[['title']]
```

```
Out[6]:
```

	title
0	TOSHIBA Smart HD LED TV 32" - 32L5650VJ Free B...
1	TOSHIBA Full HD Smart LED TV 40" - 40L5650VJ ~...
2	Samsung 40 Inch Full HD Flat LED Digital TV 40...
3	Sharp HD LED TV 24" - LC-24LE175I - Hitam
4	Lenovo Ideapad 130-15AST LAPTOP MULTIMEDIA I A...
...	...
2495	Flashdisk Hayabusa Toshiba 64GB/ Flash Disk / F...
2496	Flashdisk Toshiba USB [8 GB]
2497	Rimas COD Combo Multi Card Reader + 3 USB HUB ...
2498	Disket/Diskette Floppy Disk Maxell MF2HD Forma...
2499	Toshiba Canvio Basics 2TB - HDD / HD / Hardisk...

**Gambar 4.66** Selection Column

### 3. Menampilkan baris terpilih dengan loc

Pandas juga bisa menampilkan data hanya untuk baris tertentu berdasarkan karakter string.

```
1 test1 = pd.read_csv("products.csv", index_col="title")
2 y = test1.loc["Flashdisk Toshiba USB [8 GB]"]
3 yy = test1.loc["Flashdisk Toshiba USB [8 GB]"]
4 print(y, "\n\n", yy)
5
```

**Listing 4.45** Selection Row loc

Pada kode 4.45, pada baris pertama untuk membaca data dengan parameter title untuk menampilkan datanya. Lalu pada baris kedua menggunakan perintah **.loc** dimana untuk mengambil data pada baris tersebut dengan memasukkan sesuatu yang unik dari kolom tersebut. Begitupun juga dengan baris 3 pun sama. Sedangkan pada baris ke-empat untuk menampilkan data baris 2 dan 3 dengan memanggil variabel tersebut. Untuk hasil penerapan bisa dilihat digambar berikut :

```
In [25]: test1 = pd.read_csv("products.csv", index_col="title")
        y = test1.loc["Flashdisk Toshiba USB [8 GB]"]
        yy = test1.loc["Flashdisk Toshiba USB [8 GB]"]
        print(y, "\n\n", yy)

productId    111362
Name: Flashdisk Toshiba USB [8 GB], dtype: int64

productId    111362
Name: Flashdisk Toshiba USB [8 GB], dtype: int64
```

**Gambar 4.67** Selection Row loc

#### 4. Menampilkan baris terpilih dengan iloc

Pandas juga bisa menampilkan baris tertentu berdasarkan karakter integer.

```
1 a = test1.iloc[3]
2 a
3
```

**Listing 4.46** Selection Row iloc

Pada kode 4.46, pada baris pertama digunakan untuk mengambil nilai dari data yang ada pada baris urutan ke-3 dan pada baris kedua kode tersebut digunakan untuk menampilkan dari perintah di baris pertama. Untuk hasil penerapan bisa dilihat pada gambar berikut :

```
In [26]: a = test1.iloc[3]
a
Out[26]: productId      4
         Name: Sharp HD LED TV 24" - LC-24LE175I - Hitam, dtype: int64
```

**Gambar 4.68** Selection Row iloc

#### 5. Mengisi Missing Value di DataFrame

Pandas juga mengisi *missing value* dalam sebuah data. Missing Value dalam data ditandai dengan kata NaN. Untuk mengatasi tersebut, maka perlu adanya *library* ini untuk mengatasi permasalahan tersebut.

```
1 final = pd.pivot_table(Rating_avg, values='adg_rating',
2 index='customerId', columns='productId')
3 final.head()
4
5 final_product = final.fillna(final.mean(axis=0))
```

**Listing 4.47** Missing Value

Pada kode 4.47, menjelaskan bahwa pada baris kedua menampilkan data dari baris pertama yang terdapat *missing value*, lalu pada baris keempat digunakan untuk mengisi nilai *missing value* dengan nilai 0. Untuk hasil penerapan bisa dilihat pada gambar berikut :

productId	1	2	3	4	5	6	7	9	10	11
customerId										
316	-0.829457	NaN	NaN	NaN	NaN	NaN	-1.329457	NaN	-0.829457	NaN
320	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
359	1.314526	NaN	NaN	NaN	NaN	1.314526	NaN	NaN	0.314526	0.314526
370	0.705596	0.205596	NaN	NaN	NaN	1.205596	NaN	NaN	NaN	NaN
910	1.101920	0.101920	-0.39808	NaN	-0.39808	-0.398080	NaN	NaN	NaN	0.101920

**Gambar 4.69** Missing Value

```
In [6]: final_product.head()
```

```
Out[6]:
```

	productid	1	2	3	4	5	6	7	9	10	11
customerid											
316	-0.829457	-0.436518	-0.468109	-0.770223	-0.615331	0.320415	-1.329457	-0.690175	-0.829457	-0.094277	
320	0.200220	-0.436518	-0.468109	-0.770223	-0.615331	0.320415	-0.203889	-0.690175	-0.150642	-0.094277	
359	1.314526	-0.436518	-0.468109	-0.770223	-0.615331	1.314526	-0.203889	-0.690175	0.314526	0.314526	
370	0.705596	0.205596	-0.468109	-0.770223	-0.615331	1.205596	-0.203889	-0.690175	-0.150642	-0.094277	
910	1.101920	0.101920	-0.398080	-0.770223	-0.398080	-0.398080	-0.203889	-0.690175	-0.150642	0.101920	

**Gambar 4.70** Fix Missing Value

## 6. Menghapus nilai missing value

Pandas juga bisa menghapus nilai missing value dalam sebuah data, namun ini sangat tidak dianjurkan dalam pengolahan data. Karena bisa mempengaruhi proses yang lainnya. Pandas juga bisa menghapus nilai missing value tersebut

```
1 final.dropna()
```

```
2
```

**Listing 4.48** Drop Missing Value

Pada kode 4.48, digunakan untuk menghapus baris data yang memiliki nilai *missing value* dan untuk hasil penerapan bisa dilihat pada gambar berikut :

```
In [6]: final.dropna()
```

```
Out[6]:
```

```
productid 1 2 3 4 5 6 7 9 10 11
customerid
```

**Gambar 4.71** Drop Missing Value

## 7. Pengulangan Data dengan iterrow

Pandas juga bisa menampilkan semua data yang ada ke dalam bentuk list dengan menggunakan iterrow.

```
1 for i, j in test.iterrows():
2     print(i, j)
3     print()
4
```

**Listing 4.49** Looping Iterrow

Pada kode 4.49, untuk menampilkan data yang ada dalam bentuk list dan untuk hasil penerapan bisa dilihat pada gambar berikut :

## 8. Konversi teks

```
In [30]: for i, j in test.iterrows():
          print(i, j)
          print()

1 title    TOSHIBA Smart HD LED TV 32" - 32L5650VJ Free B...
Name: 1, dtype: object

2 title    TOSHIBA Full HD Smart LED TV 40" - 40L5650VJ -...
Name: 2, dtype: object

3 title    Samsung 40 Inch Full HD Flat LED Digital TV 40...
Name: 3, dtype: object

4 title    Sharp HD LED TV 24" - LC-24LE175I - Hitam
Name: 4, dtype: object
```

**Gambar 4.72** Iterrow Data

Pandas juga bisa melakukan konversi teks menjadi huruf kecil semua dan sebaliknya.

```
1 test_DF["title"] = test_DF["title"].str.lower()
2 test_DF
3
```

**Listing 4.50** Convert Lowercase

Pada kode 4.50, pada baris pertama untuk mengambil data di kolom title lalu di konversi menjadi huruf kecil semua. Sedangkan pada baris kedua untuk menampilkan hasil konversi data tersebut. Untuk hasil penerapan kode diatas bisa dilihat pada gambar :

```
In [3]: test_DF["title"] = test_DF["title"].str.lower()
test_DF

Out[3]:
```

	productid	title
0	1	toshiba smart hd led tv 32" - 32l5650vj free b...
1	2	toshiba full hd smart led tv 40" - 40l5650vj -...
2	3	samsung 40 inch full hd flat led digital tv 40...
3	4	sharp hd led tv 24" - lc-24le175i - hitam
4	5	lenovo ideapad 130-15ast laptop multimedia i a...
...	...	...
2495	109487	flashdisk hayabusa toshiba 64gb/ flash disk /f...
2496	111362	flashdisk toshiba usb [8 gb]
2497	111759	rimas cod combo multi card reader + 3 usb hub ...
2498	112556	disket/diskette floppy disk maxell m12hd forma...
2499	112852	toshiba canvio basics 2tb - hdd / hd / hardisk...

**Gambar 4.73** Lowercase Data

Untuk mengkonversi data ke dalam huruf besar, kita hanya perlu merubah *lower* menjadi *upper*.

```

1 test_DF["title"] = test_DF["title"].str.upper()
2 test_DF
3

```

**Listing 4.51** Convert Uppercase

Pada kode 4.51, pada baris pertama untuk mengambil data di kolom title lalu di konversi menjadi huruf besar semua. Sedangkan pada baris kedua untuk menampilkan hasil konversi data tersebut. Untuk hasil penerapan kode diatas bisa dilihat pada gambar :

```

In [4]: test_DF["title"] = test_DF["title"].str.upper()
test_DF

Out[4]:

```

	productid	title
0	1	TOSHIBA SMART HD LED TV 32" - 32L5650VJ FREE B...
1	2	TOSHIBA FULL HD SMART LED TV 40" - 40L5650VJ -...
2	3	SAMSUNG 40 INCH FULL HD FLAT LED DIGITAL TV 40...
3	4	SHARP HD LED TV 24" - LC-24LE175I - HITAM
4	5	LENOVO IDEAPAD 130-15AST LAPTOP MULTIMEDIA I A...
...	...	...
2495	109487	FLASHDISK HAYABUSA TOSHIBA 64GB/ FLASH DISK /F...
2496	111362	FLASHDISK TOSHIBA USB [8 GB]
2497	111759	RIMAS COD COMBO MULTI CARD READER + 3 USB HUB ...
2498	112556	DISKET/DISKETTE FLOPPY DISK MAXELL MF2HD FORMA...
2499	112852	TOSHIBA CANVIO BASICS 2TB - HDD / HD / HARDISK...

**Gambar 4.74** Uppercase Data

## 9. Mengganti Nilai Data

Pandas juga bisa mengganti nilai yang ada menjadi nilai seperti yang kita inginkan.

```

1 test_DF["productId"] = test_DF["productId"].replace(1, "
One")
2 test_DF
3

```

**Listing 4.52** Replacement Data

Pada kode 4.52, pada baris pertama untuk mengganti nilai yang ada di data menjadi yang kita inginkan. Disini kita mencontohkan nilai 1 menjadi *One* dan baris kedua untuk menampilkannya. Untuk hasil penerapan bisa dilihat pada gambar berikut :

## 10. Ekstraksi Data

Pandas juga bisa melakukan ekstraksi data dimana mengekstrak data yang semula 1 kolom menjadi 2 kolom.

```

1 test_DF["productId"] = test_DF["productId"].replace(1, "
One")

```

```
In [5]: test_DF["productId"] = test_DF["productId"].replace(1, "One")
test_DF
```

```
Out[5]:
```

	productId	title
0	One	TOSHIBA SMART HD LED TV 32" - 32L5650VJ FREE B...
1	2	TOSHIBA FULL HD SMART LED TV 40" - 40L5650VJ ~...
2	3	SAMSUNG 40 INCH FULL HD FLAT LED DIGITAL TV 40...
3	4	SHARP HD LED TV 24" - LC-24LE175I - HITAM
4	5	LENOVO IDEAPAD 130-15AST LAPTOP MULTIMEDIA I A...
...	...	...
2495	109487	FLASHDISK HAYABUSA TOSHIBA 64GB/ FLASH DISK /F...
2496	111362	FLASHDISK TOSHIBA USB [8 GB]
2497	111759	RIMAS COD COMBO MULTI CARD READER + 3 USB HUB ...
2498	112556	DISKET/DISKETTE FLOPPY DISK MAXELL MF2HD FORMA...
2499	112852	TOSHIBA CANVIO BASICS 2TB - HDD / HD / HARDISK...

**Gambar 4.75** Replacement Data

```
2 test_DF
```

```
3
```

**Listing 4.53** Replacement Data





## BAB 5

---

# VISUALISASI DATA

---

### 5.1 Matplotlib



**Gambar 5.1** Matplotlib

#### 5.1.1 Sejarah Matplotlib

Matplotlib merupakan sebuah library dalam python yang bisa digunakan untuk membuat plot atau grafik 2 Dimensi. Walaupun library ini secara asal-usul mengadopsi perintah dalam MATLAB, bukan berarti library ini bergantung pada MATLAB. Meskipun matplotlib ini lebih banyak digunakan dalam penerapan kode python murni, tetapi masih banyak yang menggunakan library lain seperti Numpy serta library lain dalam

meningkatkan kinerja dalam pembuatan visualisasi dengan baik bahkan jika memang membutuhkan array dalam bentuk besar.

Tujuan awal matplotlib itu sendiri adalah bisa membuat plot sederhana cukup dengan beberapa perintah, bahkan jika bisa hanya dengan satu perintah.

Bagi pengguna MATLAB, untuk proses analisis dan visualisasi data menggunakan MATLAB mungkin cukup mudah karena MATLAB itu sendiri memiliki keunggulan dalam pembuatan plot yang mudah dan memiliki kualitas gambar yang bagus. Suatu ketika, John D. Hunter mulai bekerja dengan data EEG. John D. Hunter adalah salah satu penemu dari matplotlib. Dia menemukan sebuah kasus dimana saya harus menulis program yang bisa digunakan untuk berinteraksi dengan data tersebut dan bisa untuk mengembangkan aplikasi yang sudah ada di MATLAB. Lalu saya merasa ada hal membuat dia selalu terpikirkan adalah dengan keterbatasan yang ada pada MATLAB. Oleh sebab itu, dia memutuskan untuk kembali pada pemrograman python. Namun, ia masih mengalami permasalahan visualisasi data dengan plot 2 Dimensi.

Ketika dia ingin mencari library untuk visualisasi data tersebut. Dan ia juga memiliki beberapa persyaratan seperti :

1. Plot harus terlihat bagus. Artinya plot yang ditampilkan harus memiliki kualitas gambar yang bagus serta tulisan terlihat bagus pada gambar tersebut.
2. Bisa ditambahkan dalam pembuatan user interface untuk pengembangan aplikasi.
3. Kode atau perintah yang digunakan harus mudah sehingga bisa dengan mudah untuk memahaminya serta mengembangkan.
4. Membuat plot harus mudah

Namun dalam pencarian library tersebut dalam python, menemui permasalahan dimana tidak menemukan library yang cocok untuk bisa digunakan. Maka dari itu, Dia memutuskan untuk mengadopsi kemampuan MATLAB dalam membuat visualisasi data dengan sangat baik. Dan ini merupakan keuntungan tersendiri bagi pengguna yang sudah biasa menggunakan MATLAB. Karena mereka bisa dengan mudah mempelajari cara implementasinya. Berdasarkan pandangan developers, memiliki user interface yang tetap lebih sangat berguna. Karena isi kode bisa disusun ulang tanpa harus mempengaruhi kode pengguna sebelumnya.

Struktur kode matplotlib secara konseptual terbagi menjadi 3 bagian. **User interface pylab** merupakan sekumpulan fungsi yang disediakan oleh matplotlib.pylab yang bisa digunakan pengguna untuk membuat visualisasi plot dengan kode yang hampir mirip dengan yang ada di MATLAB. **Matplotlib frontend** atau matplotlib adalah sekumpulan kelas yang bisa digunakan untuk mengelola angka, teks, garis, plot dan sebagainya. **Backend** merupakan seperangkat menggambar yang sangat bergantung pada device atau perangkat, dimana bisa mengkonversi hasil presentasi frontend menjadi hardcopy atau file. Seperti SVG, PNG, JPG, dan lain-lain.

Penerapan matplotlib itu sendiri banyak digunakan banyak orang di berbagai konteks. Sebagian orang ingin menghasilkan output file PNG yang bisa dimasukkan pada halaman web menjadi dinamis, dan sebagainya.

## 5.1.2 Instalasi

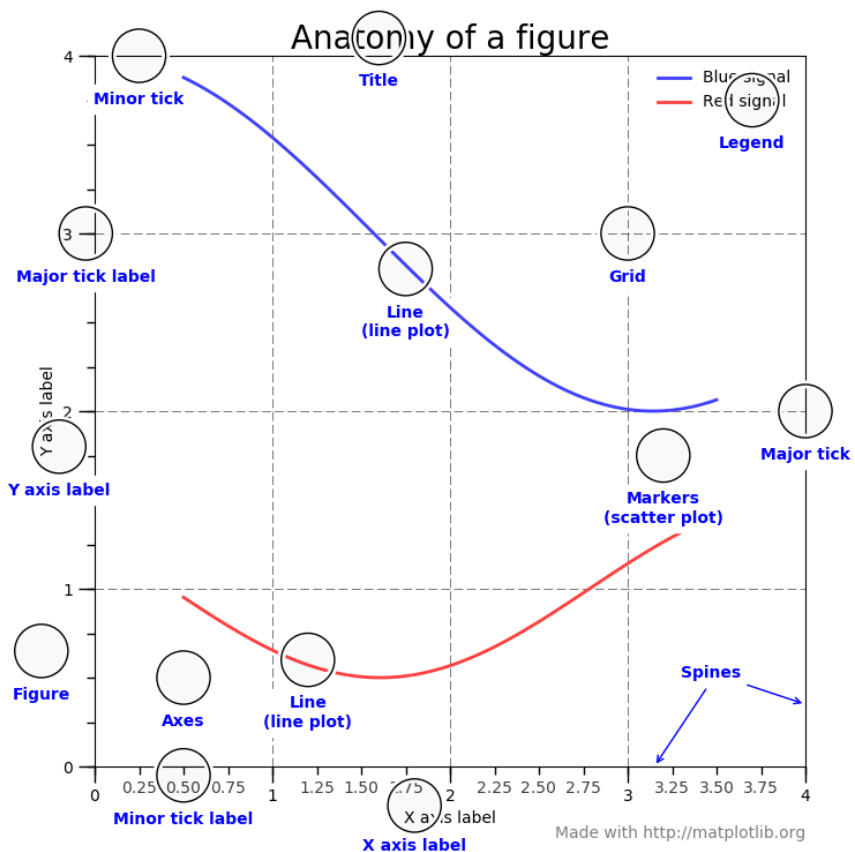
### 5.1.2.1 Windows

### 5.1.2.2 Linux

### 5.1.2.3 MacOS

## 5.1.3 Matplotlib

Matplotlib memiliki sekumpulan kode yang luas dan mungkin bisa menakutkan bagi yang baru menggunakannya. Namun, sebagian besar dari matplotlib tersebut bisa dipahami dengan kerangka kerja yang cukup sederhana dan poin-poin penting.



Gambar 5.2 Anatomy Matplotlib

#### **5.1.3.1 Figure**

#### **5.1.3.2 Axes**

#### **5.1.3.3 Axis**

#### **5.1.3.4 Artis**

#### **5.1.3.5 Legend**

#### **5.1.3.6 Title**

### **5.1.4 Pyplot**

### **5.1.5 Hubungan Matplotlib, pyplot dan pylab**

### **5.1.6 Tipe Plot**

#### **5.1.6.1 Plot Garis**

## **5.2 Seaborn**

## **5.3 Bokeh**



**Gambar 5.3** Bokeh

### **5.3.1 Definisi Bokeh**

Bokeh merupakan salah satu library dalam python yang biasanya digunakan untuk visualisasi data serta menyediakan bagan dan plot dengan kinerja yang tinggi.

Bokeh sendiri juga menyediakan 2 versi kepada pengguna.

1. Bokeh.model

bokeh.model itu sendiri merupakan sebuah antarmuka tingkat rendah yang memiliki fleksibilitas yang tinggi yang biasa digunakan untuk pengembangan aplikasi.

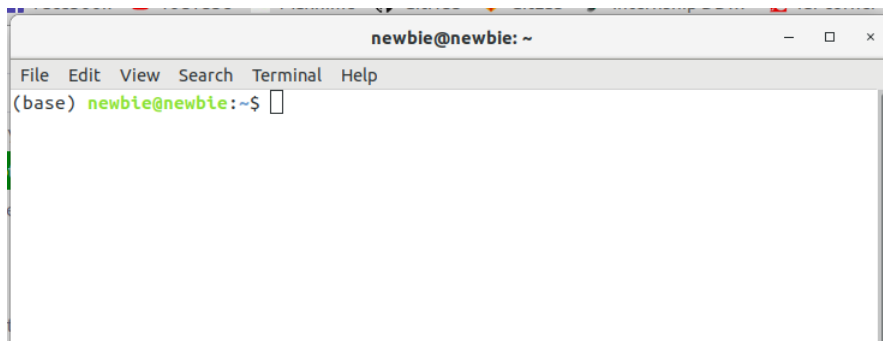
2. Bokeh.plotting

bokeh.plotting itu sendiri merupakan sebuah antarmuka level tinggi yang biasa digunakan untuk pembuatan visualisasi.

### 5.3.2 Instalasi Bokeh

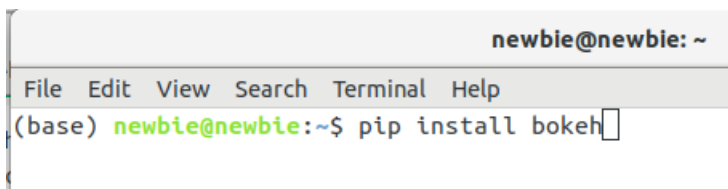
**5.3.2.1 Pip** Untuk menginstal bokeh ini cukup mudah.

1. buka terminal di linux / Mac OS atau Command Prompt di Windows.



**Gambar 5.4** Terminal

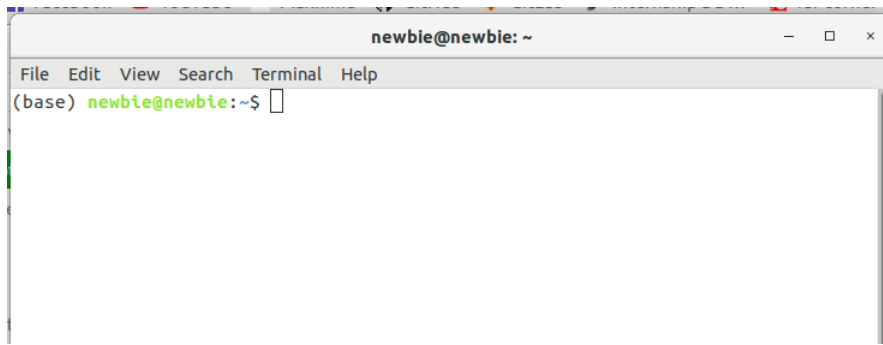
2. Masukkan perintah **pip install bokeh**.



**Gambar 5.5** Bokeh Install

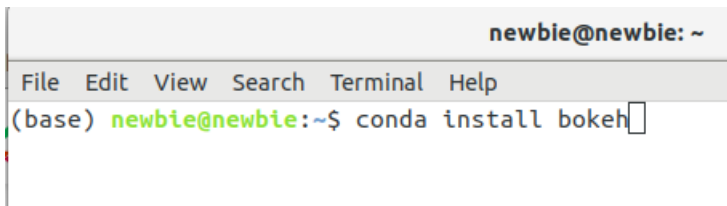
**5.3.2.2 Conda** Untuk menginstal bokeh ini cukup mudah.

1. buka terminal di linux / Mac OS atau Command Prompt di Windows.



Gambar 5.6 Terminal

2. Masukkan perintah **conda install bokeh**.



Gambar 5.7 Bokeh Conda

### 5.3.3 Contoh Implementasi

**5.3.3.1 Scatter Marker** Untuk menyebarkan tanda titik koordinat pada plot gambar.

1. Circle.

```

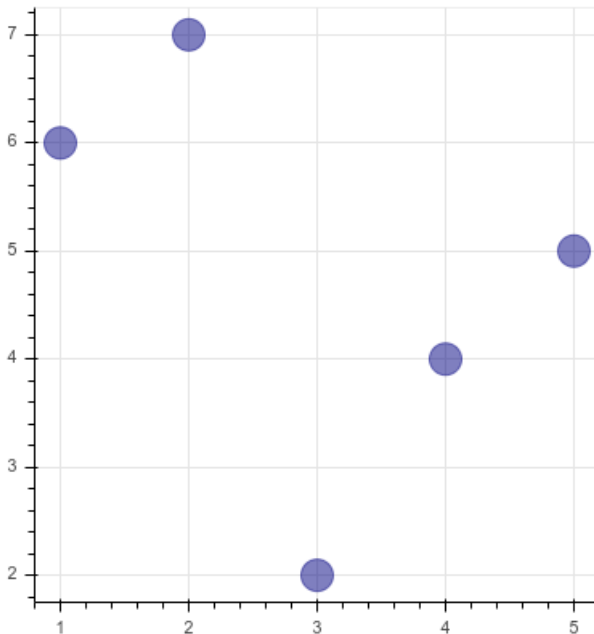
1 from bokeh.plotting import figure, output_file, show
2
3 # output to static HTML file
4 output_file("line.html")
5
6 p = figure(plot_width=400, plot_height=400)
7
8 # add a circle renderer with a size, color, and alpha
9 p.circle([1, 2, 3, 4, 5], [6, 7, 2, 4, 5], size=20, color="navy",
10         alpha=0.5)
11
12 # show the results
13 show(p)

```

Listing 5.1 Scatter Marker Circle

Penjelasan script 5.1 sebagai berikut.

- (a) Pada baris 1, merupakan penerapan dari bokeh.plotting yang import module figure, output\_file, dan show
- (b) Pada baris 4, merupakan perintah untuk mengatur output file.
- (c) Pada baris 6, merupakan perintah untuk inisialisasi pembuatan gambar plot dimana dengan ukuran plot  $400 \times 400$ .
- (d) Pada baris 9, merupakan penerapan metode circle untuk menambahkan titik pada plot gambar dengan warna navy dan size 20.
- (e) Pada baris 12, merupakan penerapan module yang diimport pada baris 1 yaitu show. Dimana untuk menampilkan hasil plot yang di inisialisasi pada baris 6 dan penambahan circle pada baris 9.



**Gambar 5.8** Bokeh Plot Circle

## 2. square

```

1 from bokeh.plotting import figure, output_file, show
2
3 # output to static HTML file
4 output_file("square.html")
5
6 p = figure(plot_width=300, plot_height=300)
7
8 # add a square renderer with a size, color, and alpha
9 p.diamond(x=[1, 2, 3], y=[1, 2, 3], size=20,
10           color="#1C9099", line_width=2)

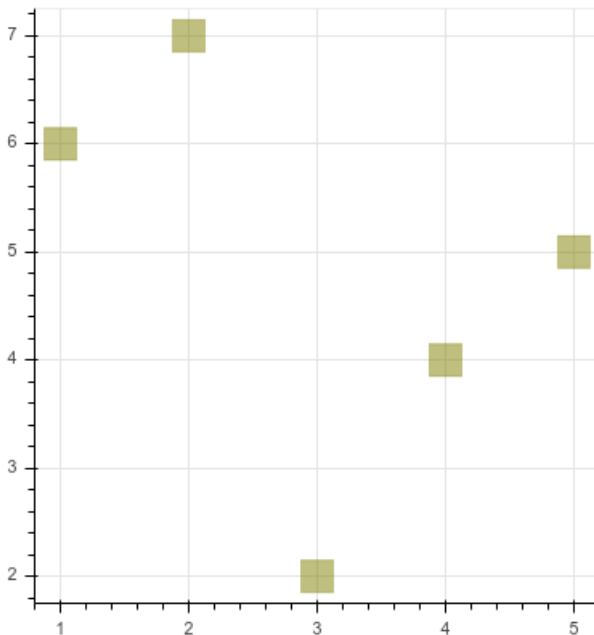
```

```
11  
12 # show the results  
13 show(p)
```

### Listing 5.2 Scatter Marker Square

Penjelasan script 5.2 sebagai berikut.

- (a) Pada baris 1, merupakan penerapan dari bokeh.plotting yang import module figure, output\_file, dan show
- (b) Pada baris 4, merupakan perintah untuk mengatur output file.
- (c) Pada baris 6, merupakan perintah untuk inisialisasi pembuatan gambar plot dimana dengan ukuran plot  $400 \times 400$ .
- (d) Pada baris 9, merupakan penerapan metode square untuk menambahkan titik pada plot gambar dan size 20.
- (e) Pada baris 12, merupakan penerapan module yang dimport pada baris 1 yaitu show. Dimana untuk menampilkan hasil plot yang di inisialisasi pada baris 6 dan penambahan square pada baris 9.



**Gambar 5.9** Bokeh Plot Square

### 3. Diamond

```
1 from bokeh.plotting import figure , output_file , show  
2
```



```

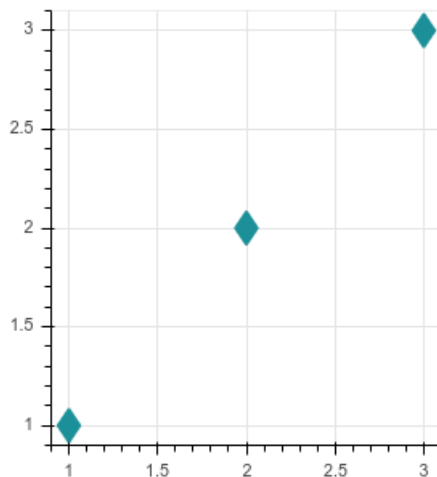
3 # output to static HTML file
4 output_file("square.html")
5
6 p = figure(plot_width=300, plot_height=300)
7
8 # add a square renderer with a size, color, and alpha
9 p.diamond(x=[1, 2, 3], y=[1, 2, 3], size=20,
10          color="#1C9099", line_width=2)
11
12 # show the results
13 show(p)

```

**Listing 5.3** Scatter Marker Diamond

Penjelasan script 5.3 sebagai berikut.

- (a) Pada baris 1, merupakan penerapan dari bokeh.plotting yang import module figure, output\_file, dan show
- (b) Pada baris 4, merupakan perintah untuk mengatur output file.
- (c) Pada baris 6, merupakan perintah untuk inisialisasi pembuatan gambar plot dimana dengan ukuran plot  $300 \times 300$ .
- (d) Pada baris 9, merupakan penerapan metode diamond untuk menambahkan titik pada plot gambar dan size 20.
- (e) Pada baris 13, merupakan penerapan module yang dimport pada baris 1 yaitu show. Dimana untuk menampilkan hasil plot yang di inisialisasi pada baris 6 dan penambahan diamond pada baris 9.



**Gambar 5.10** Bokeh Plot Diamond

#### 4. Diamond Cross

```

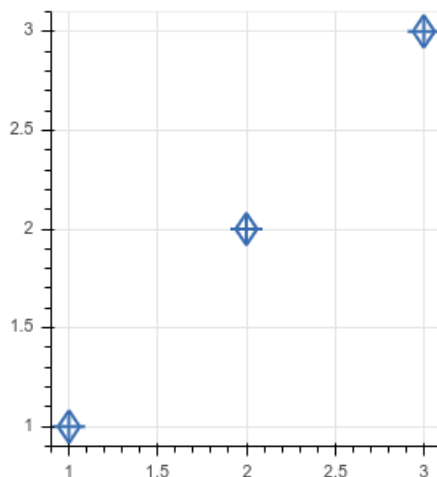
1 from bokeh.plotting import figure, output_file, show
2
3 # output to static HTML file
4 output_file("square.html")
5
6 p = figure(plot_width=300, plot_height=300)
7
8 # add a square renderer with a size, color, and alpha
9 p.diamond_cross(x=[1, 2, 3], y=[1, 2, 3], size=20,
10                color="#386CB0", fill_color=None, line_width=2)
11
12 # show the results
13 show(p)

```

**Listing 5.4** Scatter Marker Diamond Cross

Penjelasan script 5.4 sebagai berikut.

- (a) Pada baris 1, merupakan penerapan dari bokeh.plotting yang import module figure, output\_file, dan show
- (b) Pada baris 4, merupakan perintah untuk mengatur output file.
- (c) Pada baris 6, merupakan perintah untuk inisialisasi pembuatan gambar plot dimana dengan ukuran plot  $300 \times 300$ .
- (d) Pada baris 9, merupakan penerapan metode diamond\_cross untuk menambahkan titik pada plot gambar dan size 20.
- (e) Pada baris 13, merupakan penerapan module yang dimport pada baris 1 yaitu show. Dimana untuk menampilkan hasil plot yang di inisialisasi pada baris 6 dan penambahan diamond pada baris 9.



**Gambar 5.11** Bokeh Plot Diamond Cross

### 5.3.4 Line

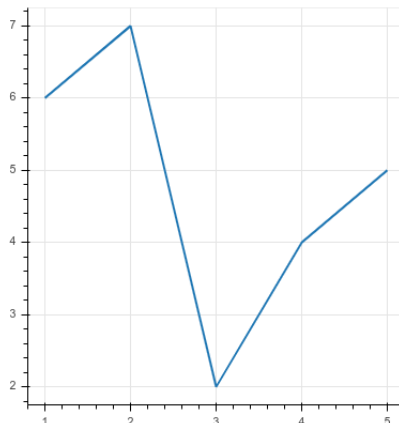
#### 5.3.4.1 Single Line

```
1 from bokeh.plotting import figure, output_file, show
2
3 output_file("line.html")
4
5 p = figure(plot_width=400, plot_height=400)
6
7 # add a line renderer
8 p.line([1, 2, 3, 4, 5], [6, 7, 2, 4, 5], line_width=2)
9
10 show(p)
```

**Listing 5.5** Single Line

Penjelasan script 5.5 seperti berikut.

1. Pada baris 1, merupakan penerapan dari bokeh.plotting yang import module figure, output\_file, dan show
2. Pada baris 3, merupakan perintah untuk mengatur output file.
3. Pada baris 5, merupakan perintah untuk inisialisasi pembuatan gambar plot dimana dengan ukuran plot  $400 \times 400$ .
4. Pada baris 8, merupakan penerapan metode line untuk membuat garis pada plot gambar dan lebar garis sebesar 20.
5. Pada baris 10, merupakan penerapan module yang dimport pada baris 1 yaitu show. Dimana untuk menampilkan hasil plot yang di inisialisasi pada baris 5 dan penambahan diamond pada baris 8.



**Gambar 5.12** Single Line

### 5.3.4.2 Step Line

```

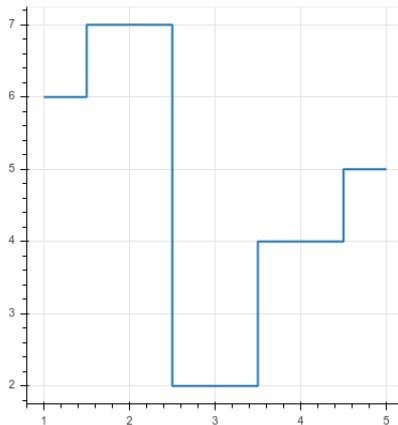
1 from bokeh.plotting import figure, output_file, show
2
3 output_file("line.html")
4
5 p = figure(plot_width=400, plot_height=400)
6
7 # add a steps renderer
8 p.step([1, 2, 3, 4, 5], [6, 7, 2, 4, 5], line_width=2, mode="center")
9
10 show(p)

```

**Listing 5.6** Step line

Penjelasan script 5.6 seperti berikut.

1. Pada baris 1, merupakan penerapan dari bokeh.plotting yang import module figure, output\_file, dan show
2. Pada baris 3, merupakan perintah untuk mengatur output file.
3. Pada baris 5, merupakan perintah untuk inisialisasi pembuatan gambar plot dimana dengan ukuran plot  $400 \times 400$ .
4. Pada baris 8, merupakan penerapan metode step untuk membuat garis pada plot gambar dengan lebar garis sebesar 2.
5. Pada baris 10, merupakan penerapan module yang dimport pada baris 1 yaitu show. Dimana untuk menampilkan hasil plot yang di inisialisasi pada baris 5 dan penambahan diamond pada baris 8.



**Gambar 5.13** Step Line

### 5.3.4.3 Multiple Line

```

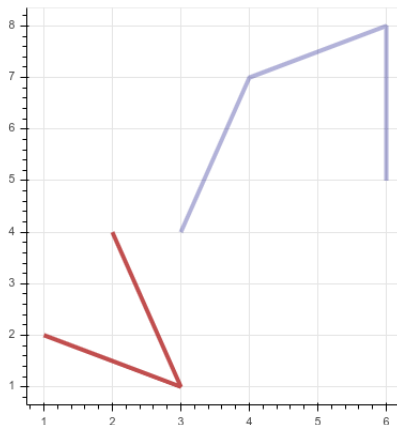
1 from bokeh.plotting import figure, output_file, show
2
3 output_file("patch.html")
4
5 p = figure(plot_width=400, plot_height=400)
6
7 p.multi_line([[1, 3, 2], [3, 4, 6, 6]], [[2, 1, 4], [4, 7, 8, 5]],
8             color=["firebrick", "navy"], alpha=[0.8, 0.3],
9             line_width=4)
10 show(p)

```

**Listing 5.7** Multiple Line

Penjelasan script 5.7 seperti berikut.

1. Pada baris 1, merupakan penerapan dari bokeh.plotting yang import module figure, output\_file, dan show
2. Pada baris 3, merupakan perintah untuk mengatur output file.
3. Pada baris 5, merupakan perintah untuk inialisasi pembuatan gambar plot dimana dengan ukuran plot  $400 \times 400$ .
4. Pada baris 8, merupakan penerapan metode multi\_line untuk membuat garis pada plot gambar dengan lebar garis sebesar 2.
5. Pada baris 10, merupakan penerapan module yang dimport pada baris 1 yaitu show. Dimana untuk menampilkan hasil plot yang di inialisasi pada baris 5 dan penambahan diamond pada baris 8.



**Gambar 5.14** Multiple Line

#### 5.3.4.4 Stacked Line

```

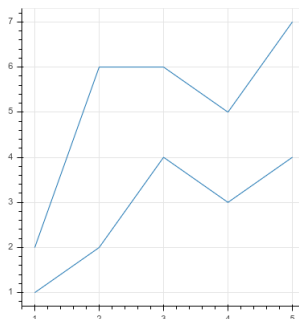
1 from bokeh.models import ColumnDataSource
2 from bokeh.plotting import figure, output_file, show
3
4 output_file("vline_stack.html")
5
6 source = ColumnDataSource(data=dict(
7     x=[1, 2, 3, 4, 5],
8     y1=[1, 2, 4, 3, 4],
9     y2=[1, 4, 2, 2, 3],
10 ))
11 p = figure(plot_width=400, plot_height=400)
12
13 p.vline_stack(['y1', 'y2'], x='x', source=source)
14
15 show(p)

```

### Listing 5.8 Stacked Line

Penjelasan script 5.8 seperti berikut.

1. Pada baris 2, merupakan penerapan dari bokeh.plotting yang import module figure, output\_file, dan show
2. Pada baris 1, merupakan penerapan dari bokeh.model yang import module column data source
3. Pada baris 4, merupakan perintah untuk mengatur output file.
4. Pada baris 6, merupakan penerapan module di baris 1 untuk menginisialisasi data.
5. Pada baris 11, merupakan perintah untuk inisialisasi pembuatan gambar plot dimana dengan ukuran plot  $400 \times 400$ .
6. Pada baris 13, merupakan penerapan metode multiple\_line untuk membuat garis pada plot gambar dengan lebar garis sebesar 2.
7. Pada baris 15, merupakan penerapan module yang dimport pada baris 1 yaitu show. Dimana untuk menampilkan hasil plot yang di inisialisasi pada baris 5 dan penambahan diamond pada baris 8.



Gambar 5.15 Stacked Line

### 5.3.5 Bar

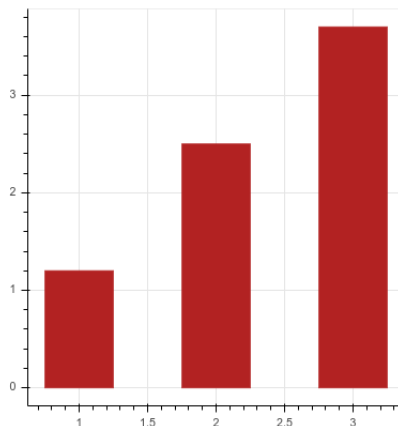
#### 5.3.5.1 Bars

```
1 from bokeh.plotting import figure, show, output_file
2
3 output_file('vbar.html')
4
5 p = figure(plot_width=400, plot_height=400)
6 p.vbar(x=[1, 2, 3], width=0.5, bottom=0,
7        top=[1.2, 2.5, 3.7], color="firebrick")
8
9 show(p)
```

**Listing 5.9** Bar chart

Penjelasan script 5.9 seperti berikut.

1. Pada baris 1, merupakan penerapan dari bokeh.plotting yang import module figure, output\_file, dan show
2. Pada baris 3, merupakan perintah untuk mengatur output file.
3. Pada baris 5, merupakan perintah untuk inisialisasi pembuatan gambar plot dimana dengan ukuran plot  $400 \times 400$ .
4. Pada baris 6, merupakan penerapan metode multiple\_line untuk membuat garis pada plot gambar dengan lebar sebesar 0.5.
5. Pada baris 9, merupakan penerapan module yang dimport pada baris 1 yaitu show. Dimana untuk menampilkan hasil plot yang di inisialisasi pada baris 5.



**Gambar 5.16** Vbar

```

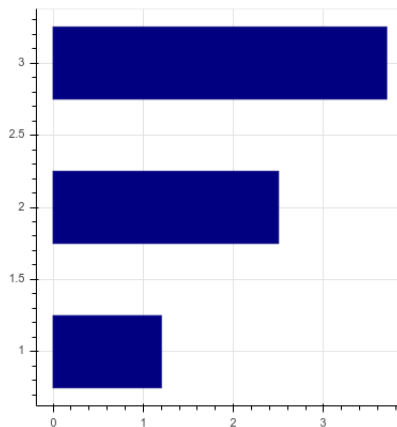
1 from bokeh.plotting import figure, show, output_file
2
3 output_file('hbar.html')
4
5 p = figure(plot_width=400, plot_height=400)
6 p.hbar(y=[1, 2, 3], height=0.5, left=0,
7        right=[1.2, 2.5, 3.7], color="navy")
8
9 show(p)

```

**Listing 5.10** Horizontal Bar

Penjelasan script 5.9 seperti berikut.

1. Pada baris 1, merupakan penerapan dari bokeh.plotting yang import module figure, output\_file, dan show
2. Pada baris 3, merupakan perintah untuk mengatur output file.
3. Pada baris 5, merupakan perintah untuk inisialisasi pembuatan gambar plot dimana dengan ukuran plot  $400 \times 400$ .
4. Pada baris 6, merupakan penerapan metode multiple\_line untuk membuat garis pada plot gambar dengan lebar sebesar 0.5.
5. Pada baris 9, merupakan penerapan module yang dimport pada baris 1 yaitu show. Dimana untuk menampilkan hasil plot yang di inisialisasi pada baris 5.



**Gambar 5.17** Hbar

### 5.3.5.2 Stacked Bar

```

1 from bokeh.models import ColumnDataSource
2 from bokeh.plotting import figure, output_file, show
3
4 output_file("hbar_stack.html")

```



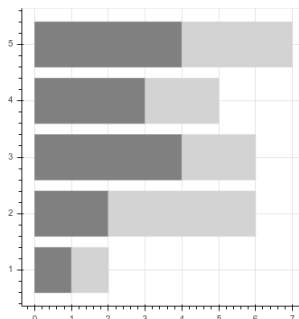
```

5
6 source = ColumnDataSource(data=dict(
7     y=[1, 2, 3, 4, 5],
8     x1=[1, 2, 4, 3, 4],
9     x2=[1, 4, 2, 2, 3],
10 ))
11 p = figure(plot_width=400, plot_height=400)
12
13 p.hbar_stack(['x1', 'x2'], y='y', height=0.8, color=("grey", "
    lightgrey"), source=source)
14
15 show(p)

```

Penjelasan script 5.8 seperti berikut.

1. Pada baris 2, merupakan penerapan dari bokeh.plotting yang import module figure, output\_file, dan show
2. Pada baris 1, merupakan penerapan dari bokeh.model yang import module column data source
3. Pada baris 4, merupakan perintah untuk mengatur output file.
4. Pada baris 6, merupakan penerapan module di baris 1 untuk menginisialisasi data.
5. Pada baris 11, merupakan perintah untuk inisialisasi pembuatan gambar plot dimana dengan ukuran plot  $400 \times 400$ .
6. Pada baris 13, merupakan penerapan metode multiple\_line untuk membuat garis pada plot gambar dengan tinggi garis sebesar 2.
7. Pada baris 15, merupakan penerapan module yang dimport pada baris 1 yaitu show. Dimana untuk menampilkan hasil plot yang di inisialisasi pada baris 5 dan penambahan diamond pada baris 8.



**Gambar 5.18** Stacked Bar

### 5.3.6 Rectangles

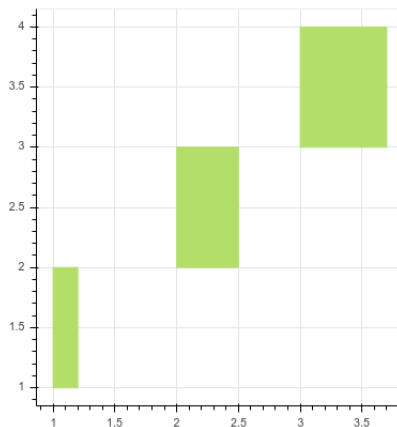
#### 5.3.6.1 Quad Rectangles

```
1 from bokeh.plotting import figure, show, output_file
2
3 output_file('rectangles.html')
4
5 p = figure(plot_width=400, plot_height=400)
6 p.quad(top=[2, 3, 4], bottom=[1, 2, 3], left=[1, 2, 3],
7        right=[1.2, 2.5, 3.7], color="#B3DE69")
8
9 show(p)
```

**Listing 5.11** Quad Rectangles

Penjelasan script 5.9 seperti berikut.

1. Pada baris 1, merupakan penerapan dari bokeh.plotting yang import module figure, output\_file, dan show
2. Pada baris 3, merupakan perintah untuk mengatur output file.
3. Pada baris 5, merupakan perintah untuk inisialisasi pembuatan gambar plot dimana dengan ukuran plot  $400 \times 400$ .
4. Pada baris 6, merupakan penerapan metode multiple\_line untuk membuat garis pada plot gambar dengan lebar sebesar 0.5.
5. Pada baris 9, merupakan penerapan module yang diimport pada baris 1 yaitu show. Dimana untuk menampilkan hasil plot yang di inisialisasi pada baris 5.



**Gambar 5.19** Quad Rectangles

#### 5.3.6.2 Rect Rectangles

```

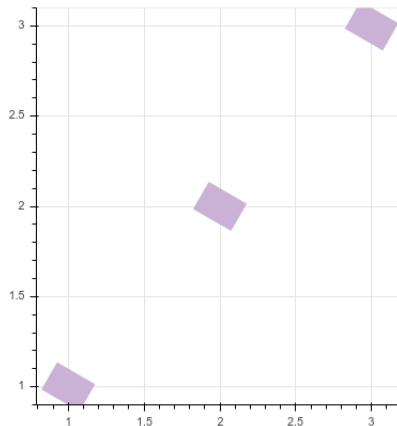
1 from math import pi
2 from bokeh.plotting import figure, show, output_file
3
4 output_file('rectangles_rotated.html')
5
6 p = figure(plot_width=400, plot_height=400)
7 p.rect(x=[1, 2, 3], y=[1, 2, 3], width=0.2, height=40, color="#CAB2D6",
8        ",
9        angle=pi/3, height_units="screen")
10 show(p)

```

**Listing 5.12** Rect Rectangles

Penjelasan script 5.12 seperti berikut.

1. Pada baris 1, merupakan penerapan dari bokeh.plotting yang import module figure, output\_file, dan show
2. Pada baris 3, merupakan perintah untuk mengatur output file.
3. Pada baris 5, merupakan perintah untuk inisialisasi pembuatan gambar plot dimana dengan ukuran plot  $400 \times 400$ .
4. Pada baris 6, merupakan penerapan metode multiple\_line untuk membuat garis pada plot gambar dengan lebar sebesar 0.5.
5. Pada baris 9, merupakan penerapan module yang dimport pada baris 1 yaitu show. Dimana untuk menampilkan hasil plot yang di inisialisasi pada baris 5.



**Gambar 5.20** Rect Rectangles

### 5.3.7 Axes

#### 5.3.7.1 Categorical Axes

```

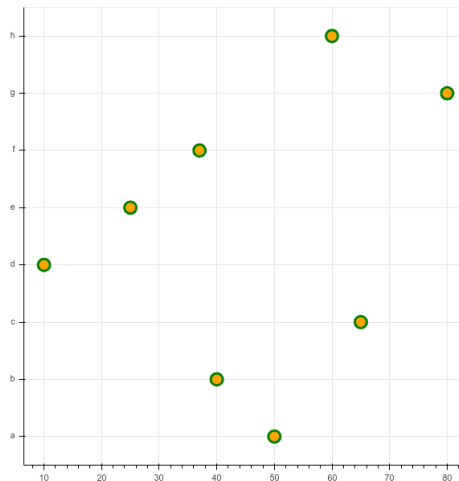
1 from bokeh.plotting import figure, output_file, show
2
3 factors = ["a", "b", "c", "d", "e", "f", "g", "h"]
4 x = [50, 40, 65, 10, 25, 37, 80, 60]
5
6 output_file("categorical.html")
7
8 p = figure(y_range=factors)
9
10 p.circle(x, factors, size=15, fill_color="orange", line_color="green",
11         , line_width=3)
12
13 show(p)

```

**Listing 5.13** Categorical Axes

Penjelasan script 5.13 seperti berikut.

1. Pada baris 1, merupakan penerapan dari bokeh.plotting yang import module figure, output\_file, dan show
2. Pada baris 3, merupakan perintah untuk mengatur output file.
3. Pada baris 5, merupakan perintah untuk inialisasi pembuatan gambar plot dimana dengan ukuran plot  $400 \times 400$ .
4. Pada baris 6, merupakan penerapan metode multiple\_line untuk membuat garis pada plot gambar dengan lebar sebesar 0.5.
5. Pada baris 9, merupakan penerapan module yang dimport pada baris 1 yaitu show. Dimana untuk menampilkan hasil plot yang di inialisasi pada baris 5.



**Gambar 5.21** Categorical Axes

## BAB 6

---

# STUDI KASUS

---

### 6.1 Studi Kasus

#### 6.1.1 Studi Kasus Visualisasi Virus Corona

Virus Corona adalah sebuah virus yang sedang ramai diperbincangkan yang berasal dari Wuhan, China. Jenis virus ini telah banyak menimbulkan banyak ketakutan di berbagai negara karena banyak rumah sakit yang penuh desak. Dataset ini digunakan untuk mengetahui bagaimana persebaran virus corona di seluruh dunia.

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 import pandas as pd
```

**Listing 6.1** Import Module

Penjelasan script 6.1.

1. Pada baris 1, merupakan perintah import module numpy yang dialiaskan menjadi np.

2. Pada baris 2, merupakan perintah import module matplotlib.pyplot yang dialiaskan menjadi plt.
3. Pada baris 3, merupakan perintah import module pandas yang dialiaskan menjadi pd.

```
1 data = pd.read_csv("../input/2019-coronavirus-dataset-01212020-01262020/2019_nCoV_20200121-20200128.csv")
2 data.head()
```

**Listing 6.2** Import Data

	Province/State	Country/Region	Last Update	Confirmed	Suspected	Recovered	Death
0	Hubei	Mainland China	1/28/2020 18:00	3554.0	NaN	80.0	125.0
1	Guangdong	Mainland China	1/28/2020 18:00	207.0	NaN	4.0	NaN
2	Zhejiang	Mainland China	1/28/2020 18:00	173.0	NaN	3.0	NaN
3	Henan	Mainland China	1/28/2020 18:00	168.0	NaN	NaN	1.0
4	Hunan	Mainland China	1/28/2020 18:00	143.0	NaN	NaN	NaN

**Gambar 6.1** Data Corona

Penjelasan script 6.2.

1. Pada baris 1, merupakan perintah untuk membaca file CSV dengan menggunakan library pandas
2. Pada baris 3, merupakan perintah untuk menampilkan isi data file CSV dengan menampilkan 5 data pertama.

```
1 data.info()
```

**Listing 6.3** Detail Data

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 669 entries, 0 to 668
Data columns (total 7 columns):
Province/State      544 non-null object
Country/Region      669 non-null object
Last Update         669 non-null object
Confirmed           639 non-null float64
Suspected           88 non-null float64
Recovered           91 non-null float64
Death              64 non-null float64
dtypes: float64(4), object(3)
memory usage: 36.7+ KB
```

**Gambar 6.2** Data Info

Penjelasan script 6.3.

1. Pada baris 1, merupakan perintah untuk menampilkan isi ringkasan dari data yang pada Dataframe.

```
1 data['Last Update'].unique()
```

**Listing 6.4** Latest

```
array(['1/28/2020 18:00', '1/28/2020 13:00', '1/27/2020 20:30',
      '1/27/2020 19:00', '1/27/2020 9:00', '1/26/2020 23:00',
      '1/26/2020 11:00 AM', '1/25/2020 10:00 PM', '1/25/2020 12:00 PM',
      '1/25/2020 12:00 AM', '1/24/2020 12:00 PM', '1/24/2020 4:00 PM',
      '1/24/2020 12:00 AM', '1/23/20 12:00 PM', '1/22/2020 12:00',
      '1/21/2020'], dtype=object)
```

**Gambar 6.3** Data Array Tanggal

Penjelasan script 6.4.

1. Pada baris 1, merupakan perintah untuk menampilkan pada dataframe di kolom Last Update yang bersifat unik.

```

1 latest_date = '1/28/2020 18:00'
2 data[ data[ 'Last Update' ] == latest_date ]

```

**Listing 6.5** Latest Data

	Province/State	Country/Region	Last Update	Confirmed	Suspected	Recovered	Death
0	Hubei	Mainland China	1/28/2020 18:00	3554.0	NaN	80.0	125.0
1	Guangdong	Mainland China	1/28/2020 18:00	207.0	NaN	4.0	NaN
2	Zhejiang	Mainland China	1/28/2020 18:00	173.0	NaN	3.0	NaN
3	Henan	Mainland China	1/28/2020 18:00	168.0	NaN	NaN	1.0
4	Hunan	Mainland China	1/28/2020 18:00	143.0	NaN	NaN	NaN
5	Chongqing	Mainland China	1/28/2020 18:00	132.0	NaN	NaN	NaN
6	Jiangxi	Mainland China	1/28/2020 18:00	109.0	NaN	3.0	NaN
7	Anhui	Mainland China	1/28/2020 18:00	106.0	NaN	NaN	NaN
8	Shandong	Mainland China	1/28/2020 18:00	95.0	NaN	NaN	NaN
9	Beijing	Mainland China	1/28/2020 18:00	91.0	NaN	4.0	1.0
10	Sichuan	Mainland China	1/28/2020 18:00	90.0	NaN	NaN	NaN
11	Fujian	Mainland China	1/28/2020 18:00	80.0	NaN	NaN	NaN
12	Jiangsu	Mainland China	1/28/2020 18:00	70.0	NaN	1.0	NaN
13	Shanghai	Mainland China	1/28/2020 18:00	66.0	NaN	4.0	1.0
14	Guangxi	Mainland China	1/28/2020 18:00	51.0	NaN	2.0	NaN
15	Shaanxi	Mainland China	1/28/2020 18:00	46.0	NaN	NaN	NaN
16	Yunnan	Mainland China	1/28/2020 18:00	44.0	NaN	NaN	NaN
17	Hainan	Mainland China	1/28/2020 18:00	40.0	NaN	NaN	1.0
18	Liaoning	Mainland China	1/28/2020 18:00	34.0	NaN	NaN	NaN
19	Hebei	Mainland China	1/28/2020 18:00	33.0	NaN	NaN	1.0

**Gambar 6.4** Data Corona Latest

Penjelasan script 6.5.

1. Pada baris 1, merupakan deklarasi variabel dengan nilai statis.
2. Pada baris 2, merupakan perintah menampilkan data dengan parameter nilai yang ada di variabel baris 1.

```

1 unique_countries = data[ data.Confirmed > 0 ][ 'Country / Region' ]. unique ()
2 unique_countries . sort ()
3 unique_countries

```

**Listing 6.6** Cek Data Negara



```
array(['Australia', 'Cambodia', 'Canada', 'France', 'Germany',
      'Hong Kong', 'Ivory Coast', 'Japan', 'Macau', 'Mainland China',
      'Malaysia', 'Nepal', 'Singapore', 'South Korea', 'Sri Lanka',
      'Taiwan', 'Thailand', 'United States', 'Vietnam'], dtype=object)
```

**Gambar 6.5** Data Negara

Penjelasan script 6.6.

1. Pada baris 1, merupakan deklarasi variabel untuk menghitung data negara dengan nilai konfirmasi di atas 0 dan bersifat unik
2. Pada baris 2, merupakan perintah untuk mengurutkan data dengan metode **sort**.
3. Pada baris 3, merupakan perintah untuk menampilkan isi variabel tersebut.

```
1 country_confirmed_cases = []
2 for i in unique_countries:
3     country_confirmed_cases.append(data[data.Confirmed>0][data['Country/Region']==i][data['Last Update']==latest_date].Confirmed.sum())
```

**Listing 6.7** Menghitung Data Negara

```
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:3: UserWarning: Boolean Series key will be reindexed to match DataFrame index.
This is separate from the ipykernel package so we can avoid doing imports until
```

**Gambar 6.6** Output

Penjelasan script 6.7.

1. Pada baris 1, merupakan pendeklarisan variabel untuk data array
2. Pada baris 2, merupakan penerapan pengulangan dengan **for**
3. Pada baris 3, merupakan perintah untuk menambahkan data dengan **append** pada data dengan konfirmasi lebih dari 0 dan data negara/region dengan parameter provinsi yang unik, kolom **latest update** yang terkonfirmasi yang dijumlahkan.

```
1 print(set(zip(unique_countries, country_confirmed_cases)))
```

**Listing 6.8** Menampilkan Data Negara

```
{('Nepal', 1.0), ('United States', 5.0), ('Taiwan', 8.0), ('Ivory Coast', 0.0), ('Sri Lanka', 1.0), ('Vietnam', 2.0), ('Hong Kong', 8.0), ('Mainland China', 5494.0), ('Canada', 2.0), ('Germany', 4.0), ('France', 4.0), ('South Korea', 4.0), ('Japan', 7.0), ('Australia', 5.0), ('Thailand', 14.0), ('Cambodia', 1.0), ('Malaysia', 4.0), ('Singapore', 7.0), ('Macau', 7.0)}
```

**Gambar 6.7** Output

Penjelasan script 6.8.

1. Pada baris 1, merupakan perintah untuk menampilkan isi variabel pada script 6.6 baris 1 dan mengompres file menjadi zip.

```
1 unique_provinces = data['Province / State'][data.Confirmed > 1].unique()
2 unique_provinces
```

**Listing 6.9** Mencari Data Provinsi

```
array(['Hubei', 'Guangdong', 'Zhejiang', 'Henan', 'Hunan', 'Chongqing', 'Jiangxi', 'Anhui', 'Shandong', 'Beijing', 'Sichuan', 'Fujian', 'Jiangsu', 'Shanghai', 'Guangxi', 'Shaanxi', 'Yunnan', 'Hainan', 'Liaoning', 'Hebei', 'Heilongjiang', 'Shanxi', 'Tianjin', 'Gansu', 'Inner Mongolia', 'Ningxia', 'Xinjiang', 'Guizhou', 'Jilin', 'Taiwan', 'Hong Kong', 'Macau', 'Qinghai', 'California', nan, 'New South Wales', 'Bavaria', 'Ontario'], dtype=object)
```

**Gambar 6.8** Data Provinsi

Penjelasan script 6.9.

1. Pada baris 1, merupakan deklarasi variabel untuk menampilkan data provinsi dengan data konfirmasi lebih besar dari 1 serta data yang ditampilkan bersifat unik.

```
1 province_confirmed_cases = []
2 for i in unique_provinces:
3     province_confirmed_cases.append(data[data.Confirmed>0][data['Province / State']==i][data['Last Update']==latest_date].Confirmed.sum())
```

**Listing 6.10** Menghitung Data Provinsi

```
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:3: UserWarning: Boolean Series key will be reindexed to match DataFrame index.
This is separate from the ipykernel package so we can avoid doing imports until
```

**Gambar 6.9** Data Negara

Penjelasan script 6.10.

1. Pada baris 1, merupakan pendeklarisan variabel untuk data array
2. Pada baris 2, merupakan penerapan pengulangan dengan **for**
3. Pada baris 3, merupakan perintah untuk menambahkan data dengan **append** pada data dengan konfirmasi lebih dari 0 dan data negara/region dengan parameter provinsi yang unik, kolom **latest update** yang terkonfirmasi yang dijumlahkan.

```
1 print(set(zip(unique_provinces, province_confirmed_cases)))
```

**Listing 6.11** Menampilkan Data Provinsi

```
{('Shandong', 95.0), ('Hainan', 40.0), ('Beijing', 91.0), ('Taiwan', 8.0), ('Hubei', 3554.0), ('Ningxia', 11.0), ('Sichuan', 90.0), ('Heilongjiang', 33.0), ('New South Wales', 4.0), ('Fujian', 80.0), ('Hunan', 143.0), ('Ontario', 1.0), (nan, 0.0), ('Jiangxi', 109.0), ('Zhejiang', 173.0), ('Shanghai', 66.0), ('Xinjiang', 10.0), ('Yunnan', 44.0), ('Chongqing', 132.0), ('Jiangsu', 70.0), ('Liaoning', 34.0), ('Gansu', 19.0), ('Guizhou', 9.0), ('Guangxi', 51.0), ('Shaanxi', 46.0), ('Hong Kong', 8.0), ('Bavaria', 4.0), ('Hebei', 33.0), ('Anhui', 106.0), ('Shanxi', 27.0), ('California', 2.0), ('Tianjin', 24.0), ('Henan', 168.0), ('Inner Mongolia', 15.0), ('Guangdong', 207.0), ('Jilin', 8.0), ('Qinghai', 6.0), ('Macau', 7.0)}
```

**Gambar 6.10** Data Provinsi

Penjelasan script 6.11.

1. Pada baris 1, merupakan perintah untuk menampilkan isi variabel pada script 6.10 baris 1 dan mengkompres file menjadi zip.

```
1 nan_indices = []
2
3 # handle nan if there is any
4 for i in range(len(unique_provinces)):
5     if type(unique_provinces[i]) == float:
```

```

6         nan_indices.append(i)
7
8 unique_provinces = list(unique_provinces)
9 province_confirmed_cases = list(province_confirmed_cases)
10
11 for i in nan_indices:
12     unique_provinces.pop(i)
13     province_confirmed_cases.pop(i)
14
15 unique_provinces

```

**Listing 6.12** Menghapus nilai NaN

1. Pada baris 1, merupakan pendeklarasian variabel untuk penampung data array.
2. Pada baris 4, merupakan perintah untuk pengulangan
3. Pada baris 5, jika tipe provinsi unik bertipe float
4. Pada baris 6, maka nilai nan akan ditambahkan sesuai variabel di baris 4.
5. Pada baris 8, merupakan pendeklarasian variabel untuk penampung data array.
6. Pada baris 9, merupakan pendeklarasian variabel untuk penampung data array.
7. Pada baris 11, merupakan perintah untuk pengulangan
8. Pada baris 12, merupakan perintah untuk pemanggilan variabel di baris 8 dengan menggunakan fungsi **pop** untuk menghapus nilai tertentu
9. Pada baris 13, merupakan perintah untuk pemanggilan variabel di baris 9 dengan menggunakan fungsi **pop** untuk menghapus nilai tertentu
10. Pada baris 15, merupakan perintah untuk menampilkan isi dari variabel di baris 8.

```

array(['Hubei', 'Guangdong', 'Zhejiang', 'Henan', 'Hunan', 'Chongqing',
      'Jiangxi', 'Anhui', 'Shandong', 'Beijing', 'Sichuan', 'Fujian',
      'Jiangsu', 'Shanghai', 'Guangxi', 'Shaanxi', 'Yunnan', 'Hainan',
      'Liaoning', 'Hebei', 'Heilongjiang', 'Shanxi', 'Tianjin', 'Gansu',
      'Inner Mongolia', 'Ningxia', 'Xinjiang', 'Guizhou', 'Jilin',
      'Taiwan', 'Hong Kong', 'Macau', 'Qinghai', 'California', nan,
      'New South Wales', 'Bavaria', 'Ontario'], dtype=object)

```

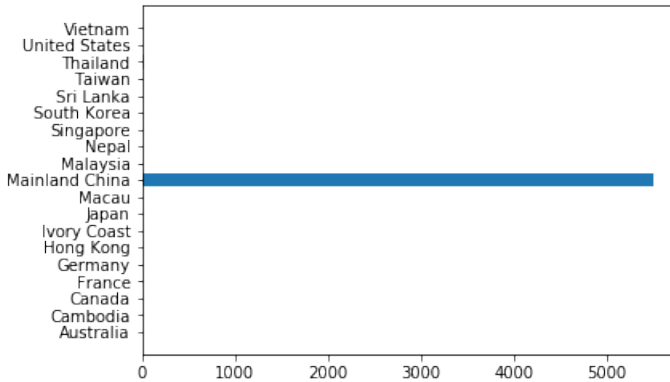
**Gambar 6.11** Data Unik Non Missing

```

1 plt.barh(unique_countries , country_confirmed_cases)
2 plt.show()

```

**Listing 6.13** Visualisasi Bar Horizontal



**Gambar 6.12** Bar Horizontal

```

1 plt.figure(figsize=(20, 10))
2 plt.barh(unique-provinces , province_confirmed_cases)
3 plt.show()

```

**Listing 6.14** Visualisasi Bar Horizontal 2



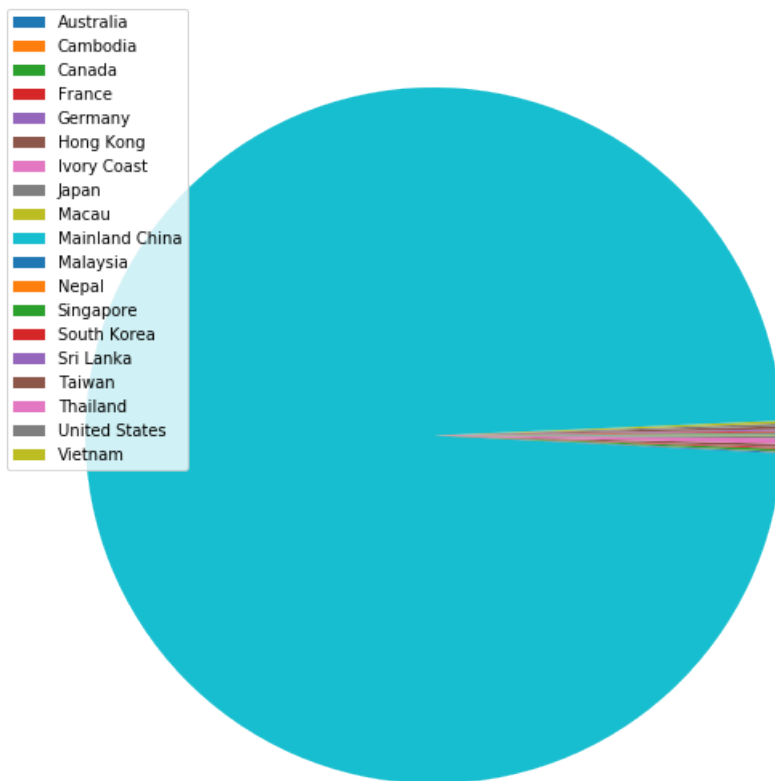
**Gambar 6.13** Bar Horizontal 2

```

1 plt.figure(figsize=(20, 10))
2 plt.barh(unique-provinces , province-confirmed-cases)
3 plt.show()

```

**Listing 6.15** Visualisasi Pie Horizontal



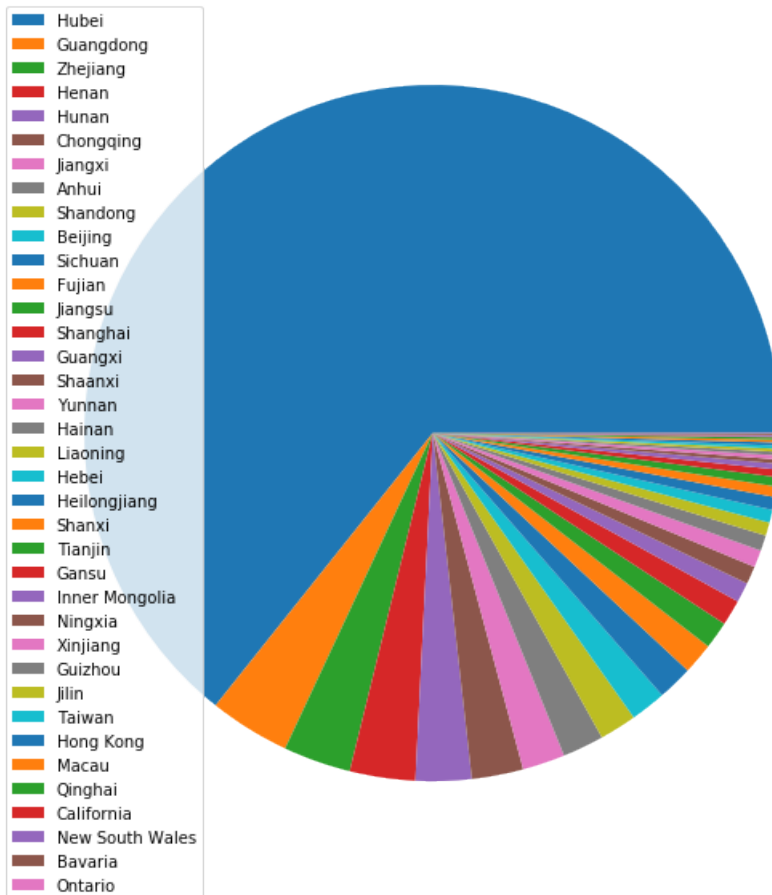
**Gambar 6.14** Pie Chart

```

1 plt.figure(figsize=(10,10))
2 plt.pie(province_confirmed_cases)
3 plt.legend(unique_provinces, loc='best')
4 plt.show()

```

**Listing 6.16** Visualisasi Pie Horizontal 2



**Gambar 6.15** Pie Chart 2

## 6.1.2 Studi Kasus 2

### Deskripsi

```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 import os
5 for dirname, _, filenames in os.walk('/kaggle/input'):
6     for filename in filenames:
7         print(os.path.join(dirname, filename))
```

**Listing 6.17** Import Module

### Penjelasan script 6.17.

1. Pada baris 1, merupakan import library numpy dengan alias nama np untuk mempermudah pemanggilan
2. Pada baris 2, merupakan import library pandas dengan alias nama pd untuk mempermudah pemanggilan
3. Pada baris 3, merupakan import library matplotlib.pyplot dengan alias nama plt untuk mempermudah pemanggilan
4. Pada baris 4, merupakan import library os untuk menggabungkan file yang akan diimport.
5. Pada baris 5, merupakan perintah untuk import file
6. Pada baris 6, merupakan perintah untuk deklarasi variabel dalam for
7. Pada baris 7, merupakan perintah untuk menggabungkan file

```
1 x1 = np.random.normal(25,5,1000)
2 y1 = np.random.normal(25,5,1000)
3
4 x2 = np.random.normal(55,5,1000)
5 y2 = np.random.normal(60,5,1000)
6
7 x3 = np.random.normal(55,5,1000)
8 y3 = np.random.normal(15,5,1000)
```

**Listing 6.18** Data Cluster

```
1 x1 = np.random.normal(25,5,1000)
2 y1 = np.random.normal(25,5,1000)
3
4 x2 = np.random.normal(55,5,1000)
5 y2 = np.random.normal(60,5,1000)
6
```



```

7 x3 = np.random.normal(55,5,1000)
8 y3 = np.random.normal(15,5,1000)

```

**Listing 6.19** Concat Data

```

1 dictionary = {"x":x,"y":y}
2
3 data=pd.DataFrame(dictionary)

```

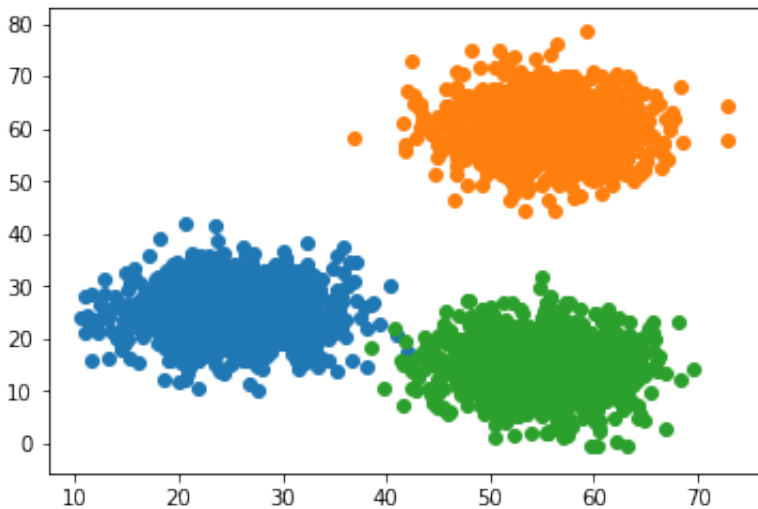
**Listing 6.20** Create Data Frame

```

1 plt.scatter(x1,y1)
2 plt.scatter(x2,y2)
3 plt.scatter(x3,y3)
4 plt.show()

```

**Listing 6.21** Visualisasi Cluster



**Gambar 6.16** Data Cluster

```

1 from sklearn.cluster import KMeans

```

**Listing 6.22** Import Library Cluster

```

1 wcss=[]
2
3 for k in range(1,15):

```

```

4 kmeans=KMeans(n_clusters=k)
5 kmeans.fit(data)
6 wcss.append(kmeans.inertia_)

```

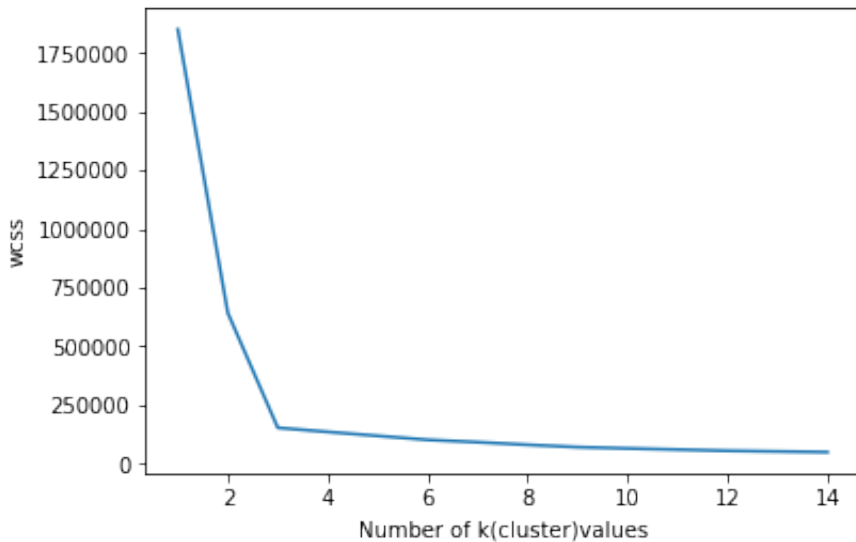
**Listing 6.23** Import Library Cluster

```

1 plt.plot(range(1,15),wcss)
2 plt.xlabel("Number of k(cluster) values")
3 plt.ylabel("wcss")
4 plt.show()

```

**Listing 6.24** Visualisasi Line Chart



**Gambar 6.17** Metode WCSS

```

1 km=KMeans(n_clusters=3)
2 clusters=km.fit_predict(data)
3 data["Label"]=clusters

```

**Listing 6.25** Clustering Data

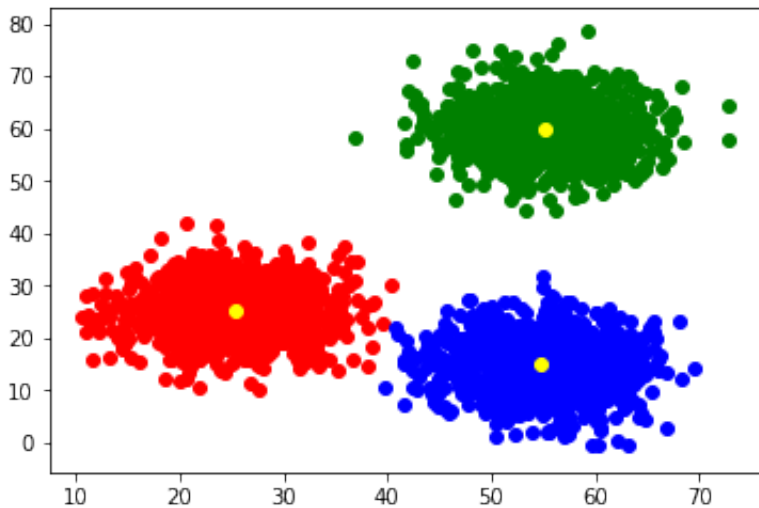
```

1 plt.scatter(data.x[data.Label == 0],data.y[data.Label == 0],color="red")
2 plt.scatter(data.x[data.Label == 1],data.y[data.Label == 1],color="green")
3 plt.scatter(data.x[data.Label == 2],data.y[data.Label == 2],color="blue")

```

```
4 plt.scatter(km.cluster_centers_[0,0],km.cluster_centers_[0,1],color="yellow") #Centroid location
5 plt.show()
```

**Listing 6.26** Visualisasi Clustering Data



**Gambar 6.18** Clustering Data

### 6.1.3 Studi Kasus 3

#### Deskripsi

```

1 import pandas as pd
2 import numpy as np
3 import math
4 import re
5 from scipy.sparse import csr_matrix
6 import matplotlib.pyplot as plt
7 import seaborn as sns
8 from surprise import Reader, Dataset, SVD, evaluate
9 sns.set_style("darkgrid")

```

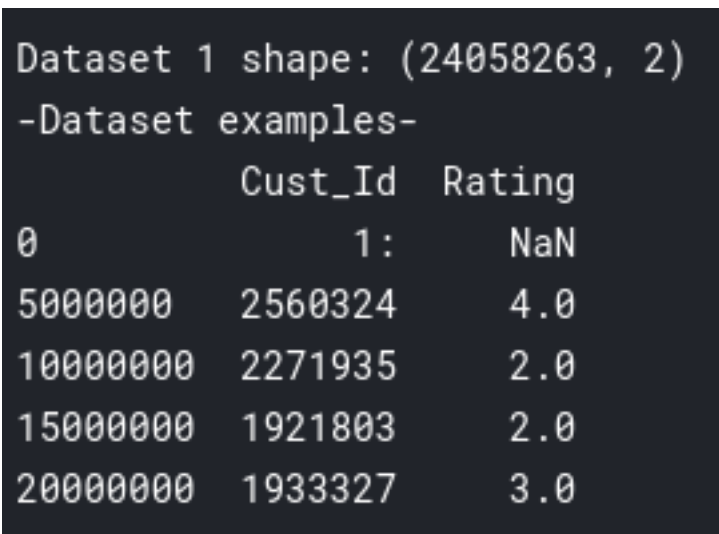
**Listing 6.27** Import Library

```

1 df1 = pd.read_csv('../input/combined_data_1.txt', header = None,
    names = ['Cust_Id', 'Rating'], usecols = [0,1])
2
3 df1['Rating'] = df1['Rating'].astype(float)
4
5 print('Dataset 1 shape: {}'.format(df1.shape))
6 print('-Dataset examples-')
7 print(df1.iloc[:5000000, :])

```

**Listing 6.28** Import Data



Dataset 1 shape: (24058263, 2)  
 -Dataset examples-

	Cust_Id	Rating
0	1:	NaN
5000000	2560324	4.0
10000000	2271935	2.0
15000000	1921803	2.0
20000000	1933327	3.0

**Gambar 6.19** Load Data

```

1 #df = df1.append(df2)
2 #df = df.append(df3)
3 #df = df.append(df4)
4
5 df.index = np.arange(0, len(df))
6 print('Full dataset shape: {}'.format(df.shape))
7 print('-Dataset examples-')
8 print(df.iloc[:5000000, :])

```

**Listing 6.29** Combine Data

```

Full dataset shape: (24058263, 2)
-Dataset examples-

```

	Cust_Id	Rating
0	1:	NaN
5000000	2560324	4.0
10000000	2271935	2.0
15000000	1921803	2.0
20000000	1933327	3.0

**Gambar 6.20** Combine Data

```

1 p = df.groupby('Rating')['Rating'].agg(['count'])
2
3 # get movie count
4 movie_count = df.isnull().sum()[1]
5
6 # get customer count
7 cust_count = df['Cust_Id'].nunique() - movie_count
8
9 # get rating count
10 rating_count = df['Cust_Id'].count() - movie_count
11
12 ax = p.plot(kind = 'barh', legend = False, figsize = (15,10))
13 plt.title('Total pool: {:,} Movies, {:,} customers, {:,} ratings
14         given'.format(movie_count, cust_count, rating_count), fontsize
15         =20)
16 plt.axis('off')
17
18 for i in range(1,6):

```

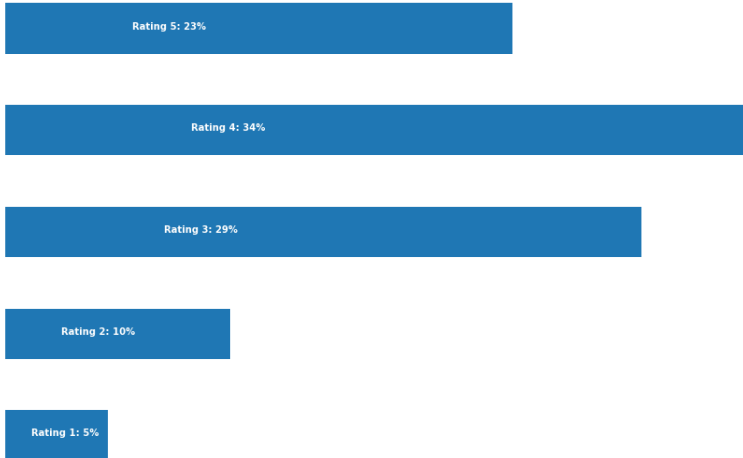
```

17 ax.text(p.iloc[i-1][0]/4, i-1, 'Rating {}: {:.0f}%'.format(i, p.
   iloc[i-1][0]*100 / p.sum()[0]), color = 'white', weight = 'bold')

```

**Listing 6.30** Viewing Data

Total pool: 4,499 Movies, 470,758 customers, 24,053,764 ratings given



**Gambar 6.21** View Data

```

1 df_nan = pd.DataFrame(pd.isnull(df.Rating))
2 df_nan = df_nan[df_nan['Rating'] == True]
3 df_nan = df_nan.reset_index()
4
5 movie_np = []
6 movie_id = 1
7
8 for i,j in zip(df_nan['index'][1:], df_nan['index'][:-1]):
9     # numpy approach
10    temp = np.full((1,i-j-1), movie_id)
11    movie_np = np.append(movie_np, temp)
12    movie_id += 1
13
14 # Account for last record and corresponding length
15 # numpy approach
16 last_record = np.full((1,len(df) - df_nan.iloc[-1, 0] - 1),movie_id)
17 movie_np = np.append(movie_np, last_record)
18
19 print('Movie numpy: {}'.format(movie_np))
20 print('Length: {}'.format(len(movie_np)))

```

**Listing 6.31** Data Cleaning

```
Movie numpy: [1.000e+00 1.000e+00 1.000e+00 ... 4.499e+03 4.499e+03 4.499e
+03]
Length: 24053764
```

**Gambar 6.22** Cleaning Data

```
1 df = df[pd.notnull(df['Rating'])]
2
3 df['Movie_Id'] = movie_np.astype(int)
4 df['Cust_Id'] = df['Cust_Id'].astype(int)
5 print('-Dataset examples-')
6 print(df.iloc[:5000000, :])
```

**Listing 6.32** Remove ID

-Dataset examples-

	Cust_Id	Rating	Movie_Id
1	1488844	3.0	1
5000996	501954	2.0	996
10001962	404654	5.0	1962
15002876	886608	2.0	2876
20003825	1193835	2.0	3825

**Gambar 6.23** Remove Movie ID

```
1 f = ['count', 'mean']
2
3 df_movie_summary = df.groupby('Movie_Id')['Rating'].agg(f)
4 df_movie_summary.index = df_movie_summary.index.map(int)
5 movie_benchmark = round(df_movie_summary['count'].quantile(0.7), 0)
6 drop_movie_list = df_movie_summary[df_movie_summary['count'] <
    movie_benchmark].index
7
8 print('Movie minimum times of review: {}'.format(movie_benchmark))
9
10 df_cust_summary = df.groupby('Cust_Id')['Rating'].agg(f)
```

```

11 df_cust_summary.index = df_cust_summary.index.map(int)
12 cust_benchmark = round(df_cust_summary['count'].quantile(0.7), 0)
13 drop_cust_list = df_cust_summary[df_cust_summary['count'] <
    cust_benchmark].index
14
15 print('Customer minimum times of review: {}'.format(cust_benchmark))

```

**Listing 6.33** Count Minimum Times

```

Movie minimum times of review: 1799.0
Customer minimum times of review: 52.0

```

**Gambar 6.24** Minimum Times

```

1 print('Original Shape: {}'.format(df.shape))
2 df = df[~df['Movie_Id'].isin(drop_movie_list)]
3 df = df[~df['Cust_Id'].isin(drop_cust_list)]
4 print('After Trim Shape: {}'.format(df.shape))
5 print('-Data Examples-')
6 print(df.iloc[:5000000, :])

```

**Listing 6.34** Trim Data

```

Original Shape: (24053764, 3)
After Trim Shape: (17337458, 3)
-Data Examples-

```

	Cust_Id	Rating	Movie_Id
696	712664	5.0	3
6932490	1299309	5.0	1384
13860273	400155	3.0	2660
20766530	466962	4.0	3923

**Gambar 6.25** Trim Data

```

1 df_p = pd.pivot_table(df, values='Rating', index='Cust_Id', columns='
    Movie_Id')

```



```
2
3 print(df_p.shape)
```

**Listing 6.35** Shape Data



(143458, 1350)

**Gambar 6.26** Shape Data

```
1 df_title = pd.read_csv('../input/movie_titles.csv', encoding = "ISO
  -8859-1", header = None, names = ['Movie_Id', 'Year', 'Name'])
2 df_title.set_index('Movie_Id', inplace = True)
3 print(df_title.head(10))
```

**Listing 6.36** Mapping Data

Movie_Id	Year	Name
1	2003.0	Dinosaur Planet
2	2004.0	Isle of Man TT 2004 Review
3	1997.0	Character
4	1994.0	Paula Abdul's Get Up & Dance
5	2004.0	The Rise and Fall of ECW
6	1997.0	Sick
7	1992.0	8 Man
8	2004.0	What the #\$*! Do We Know!?
9	1991.0	Class of Nuke 'Em High 2
10	2001.0	Fighter

**Gambar 6.27** Mapping Data

### 6.1.3.1 With Collaborative Filtering

```
1 reader = Reader()
2
3 # get just top 100K rows for faster run time
4 data = Dataset.load_from_df(df[['Cust_Id', 'Movie_Id', 'Rating'
  ]][:100000], reader)
5 data.split(n_folds=3)
```

```

6
7 svd = SVD()
8 evaluate(svd, data, measures=[ 'RMSE', 'MAE' ])

```

### Listing 6.37 Load Data

```

1 df_785314 = df[(df['Cust_Id'] == 785314) & (df['Rating'] == 5)]
2 df_785314 = df_785314.set_index('Movie_Id')
3 df_785314 = df_785314.join(df_title)['Name']
4 print(df_785314)

```

### Listing 6.38 Show Data

```

Movie_Id
57                                Richard III
175                               Reservoir Dogs
311                                Ed Wood
329                                Dogma
331                               Chasing Amy
395                               Captain Blood
788                                Clerks
798                                Jaws
907                               Animal Crackers
985                               The Mummy
1552                              Black Adder
1905  Pirates of the Caribbean: The Curse of the Bla...
2008                Four Weddings and a Funeral
2122                Being John Malkovich
2342                Super Size Me
2368                Singin' in the Rain
2438                Alien: Collector's Edition
2443                Like Water for Chocolate
2452  Lord of the Rings: The Fellowship of the Ring
2465                This Is Spinal Tap
2554                The Rocky Horror Picture Show
2847                The Mark of Zorro
2848                The Hustler
2862                The Silence of the Lambs
3158  Monty Python: The Life of Python
3168                Evil Dead 2: Dead by Dawn
3198                The Addams Family
3315                The Maltese Falcon
3440                Spirited Away
3489                Time Bandits
3598                Jason and the Argonauts
3648  Who Framed Roger Rabbit?: Special Edition
3798                The Sting
3935                Yellow Submarine
3949  Terminator 2: Extreme Edition: Bonus Material
3962                Finding Nemo (Widescreen)
3984                On the Beach
4088                A Mighty Wind
4227                The Full Monty
4253                Kind Hearts and Coronets
4306                The Sixth Sense
4345                Bowling for Columbine
4356                Road to Perdition
4392                Army of Darkness
4454                To Have and Have Not
Name: Name, dtype: object

```

### Gambar 6.28 Show Data

```

1 user_785314 = df_title.copy()
2 user_785314 = user_785314.reset_index()
3 user_785314 = user_785314[~user_785314['Movie_Id'].isin(
    drop_movie_list)]
4
5 # getting full dataset
6 data = Dataset.load_from_df(df[['Cust_Id', 'Movie_Id', 'Rating']],
    reader)
7
8 trainset = data.build_full_trainset()
9 svd.train(trainset)
10
11 user_785314['Estimate_Score'] = user_785314['Movie_Id'].apply(lambda
    x: svd.predict(785314, x).est)
12
13 user_785314 = user_785314.drop('Movie_Id', axis = 1)
14
15 user_785314 = user_785314.sort_values('Estimate_Score', ascending=
    False)
16 print(user_785314.head(10))

```

**Listing 6.39** Predict Data

### 6.1.3.2 With Pearson R Correlation

```

1 def recommend(movie_title, min_count):
2     print("For movie {}".format(movie_title))
3     print("-- Top 10 movies recommended based on Pearsons'R
    correlation --")
4     i = int(df_title.index[df_title['Name'] == movie_title][0])
5     target = df_p[i]
6     similar_to_target = df_p.corrwith(target)
7     corr_target = pd.DataFrame(similar_to_target, columns = ['
    PearsonR'])
8     corr_target.dropna(inplace = True)
9     corr_target = corr_target.sort_values('PearsonR', ascending =
    False)
10    corr_target.index = corr_target.index.map(int)
11    corr_target = corr_target.join(df_movie_summary)[[
    'PearsonR', 'Name', 'count', 'mean']]
12    print(corr_target[corr_target['count']>min_count][:10].to_string(
    index=False))

```

**Listing 6.40** Definisi Fungsi

```

1 recommend("What the #$*! Do We Know!?", 0)

```

**Listing 6.41** Fungsi Rekomendasi

```

For movie (What the #$*! Do We Know!?)
- Top 10 movies recommended based on Pearsons'R correlation -
PearsonR                                     Name    count    mean
1.000000                                What the #$*! Do We Know!? 14910 3.189805
0.505500                                Inu-Yasha    1883 4.554434
0.452807  Captain Pantoja and the Special Services 1801 3.417546
0.442354                                Without a Trace: Season 1 2124 3.980226
0.384179                                Yu-Gi-Oh!: The Movie    3173 3.331547
0.383959                                Scorched    2430 2.894239
0.381173  All Creatures Great and Small: Series 1 2327 3.938118
0.381112                                As Time Goes By: Series 1 and 2 2249 4.164073
0.373018                                Cowboys & Angels    2368 3.589527
0.371981                                Biggie & Tupac    1866 3.019293

```

**Gambar 6.29** Final Predict

```

1 recommend("X2: X-Men United", 0)

```

**Listing 6.42** Fungsi Rekomendasi 2

```

For movie (X2: X-Men United)
- Top 10 movies recommended based on Pearsons'R correlation -
PearsonR                                     Name    count    mean
1.000000                                X2: X-Men United    98720 3.932202
0.384550                                Batman Beyond: The Movie    2614 3.726855
0.375967                                Justice League    3591 3.710944
0.361393                                Justice League: Justice on Trial    2961 3.718001
0.338025                                Batman Beyond: Return of the Joker    3704 3.604752
0.335256                                Batman Begins    54922 4.236699
0.328229                                Batman: Mask of the Phantasm    2823 3.767977
0.327040  Batman: The Animated Series: Tales of the Dark...    2432 3.583059
0.316666                                Dragon Ball Z: Super Android 13    2426 3.428689
0.316166                                Mortal Kombat: The Movie    7633 3.165466

```

**Gambar 6.30** Final Predict

#### 6.1.4 Studi Kasus 4

## **6.1.5 Studi Kasus 5**