

# BAB 1

---

## PYTHON

---

### 1.1 Sejarah Python

Python merupakan bahasa pemrograman yang sangat populer. Guido van Rossum merupakan yang membuat bahasan pemrograman Python dan dikenalkan pada tahun 1991. Bahasa python terinspirasi dari bahasa pemrograman ABC. Sampai saat ini, Guido masih menjadi penulis utama untuk bahasa pemrograman python, meskipun bersifat open source sehingga ribuan orang juga dapat berkontribusi dalam mengembangkannya. Saat ini pengembangan bahasa pemrograman Python terus dilakukan oleh sekumpulan pemrogram Python Software Foundation. Python Software Foundation merupakan organisasi non-profit yang dibentuk sebagai pemegang hak cipta intelektual Python sejak versi awal. Nama python bukan berasal dari nama ular yang sering kita kenal. Guido memilih nama Python sebagai nama bahasa ciptaannya dikarenakan Guido sangat menyukai acara televisi Monty Python's Flying Circus. Oleh sebab itu seringkali ungkapan dari acara tersebut sering muncul dalam korespondensi antar pengguna Python.

## 1.2 Pembahasan Python

Python merupakan bahasa pemrograman tinggi yang berguna untuk melakukan eksekusi jumlah instruksi multi guna secara interpretatif dengan metode OOP, serta menggunakan semantik dinamis untuk memberikan tingkat keterbacaan syntax. Sebagai bahasa pemrograman tinggi, python dapat dipelajari dengan mudah karena sudah dilengkapi dengan manajemen memori otomatis (*pointer*).



Gambar 1.1 Logo Python

Python bisa digunakan secara bebas, bahkan untuk kepentingan komersial sekalipun. Banyak perusahaan yang mengembangkan bahasa pemrograman python secara komersial untuk memberikan layanan. Misalnya Anaconda Navigator, adalah salah satu aplikasi untuk pemrograman python yang dilengkapi dengan tool-tool pengembangan aplikasi. Python dapat memberikan kualitas dan kecepatan untuk membangun sebuah aplikasi bertingkat atau *Rapid Application Development*. Hal tersebut didukung karena adanya *library* dengan modul-modul baik standar maupun tambahan contohnya NumPy, SciPy, dan sebagainya. Python mempunyai komunitas yang besar sebagai tempat tanya jawab. Syntax pada python dapat dijalankan dan ditulis untuk membangun sebuah aplikasi di berbagai sistem operasi, contohnya seperti:

1. Microsoft Windows : merupakan keluarga sistem operasi. yang dikembangkan oleh Microsoft, dengan menggunakan antarmuka pengguna grafis.
2. Linux atau Unix : adalah proyek perangkat lunak bebas dan sumber terbuka terbesar di dunia.
3. Java Virtual Machine : mesin virtual yang digunakan secara khusus mengeksekusi berkas bytecode java.
4. Android : sistem operasi berbasis Linux yang dirancang untuk perangkat bergenggam layar sentuh seperti telepon pintar dan komputer tablet.

5. Mac OS : ntarmuka grafikal sistem operasi yang dikembangkan dan disebarluaskan oleh Apple Inc.
6. Amiga : merupakan komputer pribadi yang dikembangkan oleh Amiga Corporation
7. Palm : merupakan sistem operasi mobile yang memberikan kemudahan penggunaan dengan layar sentuh berbasis graphical user interface.
8. OS/2 : sistem operasi yang dibuat secara bersama-sama oleh International Business Machine Corporation dan Microsoft Corporation, untuk digunakan pada komputer IBM PS/2, sebagai pengganti sistem operasi DOS yang telah lama digunakan.
9. Symbian OS : merupakan sistem operasi yang dikembangkan oleh Symbian Ltd. yaitu yang dirancang untuk peralatan bergerak.
10. Win 9x/NT/2000 : merupakan sebuah sistem operasi Microsoft yang menjadi leluhur.
11. Unix (Solaris, Linux, FreeBSD, AIX, HP/UX, SunOS, IRIX, dan lain - lain)
12. Macintosh (Intel, PPC, 68K) : merupakan jenis komputer personal berbasis PowerPC yang diproduksi oleh Apple.
13. DOS : merupakan generik bagi setiap sistem operasi yang dimuat dari perangkat penyimpanan berupa disk saat sistem komputer dinyalakan.
14. Nokia mobile phones : merupakan eralatan telekomunikasi ponsel pintar dan ponsel genggam.
15. Windows CE : merupakan perangkat lunak keluaran Microsoft untuk perangkat keras organizer.
16. Acorn/RISC OS : merupakan perangkat lunak milik perorangan dan antarmuka pengguna grafis.
17. BeOS : merupakan sistem operasi untuk komputer pribadi yang dikembangkan pada tahun 1991 oleh Be Inc.
18. VMS/OpenVMS : merupakan server komputer sistem operasi yang berjalan pada VAX, Alfa dan berbasis Itanium keluarga komputer.
19. QNX : adalah sistem operasi komersial realtime mirip UNIX yang khusus ditujukan untuk perangkat embedded.
20. VxWorks : merupakan sistem operasi waktu nyata (real-time operating system RTOS) yang dapat digunakan dalam sistem tertanam.
21. Psion : merupakan komputer mungil atau kecil.

Python juga digunakan di berbagai pada bidang pengembangan. Berikut adalah beberapa contoh aplikasi penggunaan python yang paling populer:

#### 1. Penelitian Ilmiah dan Numerik

Bahasa pemrograman python dapat digunakan untuk mengerjakan riset ilmiah untuk dapat mempermudah perhitungan dalam numerik. Contohnya penerapan pada algoritma *Double Exponential Smoothing*, *Naive Bayes*, *Decision Tree*, *Least Square*, KNN dan sebagainya.

#### 2. Data Science dan Big Data

Python sangat memungkinkan untuk dapat melakukan analisis data dari database big data.

#### 3. Media Dalam Pembelajaran Program

Bahasa pemrograman python bisa digunakan sebagai media pembelajaran di perguruan tinggi atau universitas. Bahasa pemrograman python hemat dan sangat mudah untuk dipelajari sebagai *Object Oriented Programming* dibandingkan bahasa pemrograman lainnya seperti, C++, MATLAB dan C#.

#### 4. Pengembangan Software

Python menyediakan dukungan struktur kode untuk mempermudah pengembangan software.

#### 5. Graphical User Interface (GUI)

Bahasa pemrograman python bisa digunakan untuk membuat interface dari sebuah aplikasi, dan tersedia *library* untuk membuat GUI menggunakan bahasa pemrograman python, contohnya win32extension, Qt dan GTK+.

#### 6. Website dan Internet

Bahasa pemrograman python bisa juga digunakan untuk *server side* yang diintegrasikan dengan berbagai macam internet protokol contohnya yaitu seperti HTML, JSON, FTP, IMAP dan Email Processing. Selain itu, juga python juga mempunyai *library* yang berguna untuk pengembangan internet.

#### 7. Aplikasi Bisnis

Bahasa pemrograman python juga dapat digunakan untuk membuat sistem informasi baik itu untuk bisnis ataupun instansi.

### 1.3 Perbedaan Python2 dan Python3

Bahasa pemrograman python versi 2 dan python versi 3 tidak jauh berbeda. Tetapi, disini akan dijelaskan untuk memberikan sedikit perbedaan - perdeaan dari yang telah ditemukan dan diketahui sebelumnya. Berikut adalah contohnya:

#### 1. Print

Pada Python2, print diperlakukan seperti statemen ketimbang sebuah function.  
print "Esi Vidia Rahcmadani"

Sedangkan pada Python 3, print diperlakukan sebagai function.

```
print ("Esi Vidia Rahcmadani")
```

Perubahan ini membuat sintaksis pada Python lebih konsisten. Penggunaan print() juga kompatibel dengan Python 2.7.

#### 2. Pembagian Pada Integer Pada Python 2, semua tipe data angka yang tidak mengandung desimal akan diperlakukan sebagai integer. Terlihat mudah pada awalnya, ketika mencoba untuk membagi kedua integer akan didapatkan tipe data float.

```
3 / 2 = 1.5
```

Python 2 menggunakan floor division atau dibulatkan ke nilai paling rendah misalnya 1.5 jadi 1, 2.6 jadi 2 dan seterusnya. Pada Python 2.7 akan menjadi seperti ini

```
x = 3 / 2  
print a  
#Output  
1
```

Untuk desimal maka tambahkan jadi seperti ini 3.0 / 2.0 untuk mendapatkan hasil 1.5 pada Python 3, pembagian pada bilangan integer lebih intuitif :

```
a = 3 / 2  
print(a)  
#Output  
1.5
```

Kita juga masih bisa melakukan 3.0 / 2.0 untuk mendapatkan 1.5 namun untuk mendapatkan floor division maka pada Python 3 gunakan // :

```
b = 3 // 2  
print(b)
```

```
#Output
```

```
1
```

Fitur Python 3 ini tidak kompatibel dengan Python 2.7

### 3. Dukungan Unicode

Ketika bahasa pemrograman menangani tipe data string (yang mana merupakan sekumpulan dari beberapa karakter), mereka dapat melakukan beberapa cara berbeda sehingga komputer dapat mengubah angka ke huruf dan simbol lainnya. Python2 menggunakan alfabet ASCII (*American Standard Code for Information Interchange*) secara default, sehingga ketika kita mengetik Hallo! maka Python2 menangani string sebagai ASCII. Terbatas pada beberapa ratus karakter, ASCII mungkin bukan pilihan yang fleksibel untuk menangani proses encoding terutama yang non English. Untuk menggunakan unicode yang lebih luwes, mendukung lebih dari 128,000 karakter maka kita harus mengetik u Halo!, dengan tambahan u di depannya yang mana berarti Unicode. Python3 menggunakan Unicode secara default, yang mana menyelamatkan programmer dari tambahan kode lagi, lebih hemat waktu dan mudah untuk diisikan dan ditampilkan. Karena Unicode mendukung berbagai karakter linguistik yang beragam termasuk menampilkan emoji, penggunaan karakter secara default dengan encoding memastikan perangkat mobile didukung oleh program yang kita buat. Jika kita ingin kode Python3 kita mendukung Python2, tambahkan u di depan string.

### 4. Kelanjutan Pengembangan

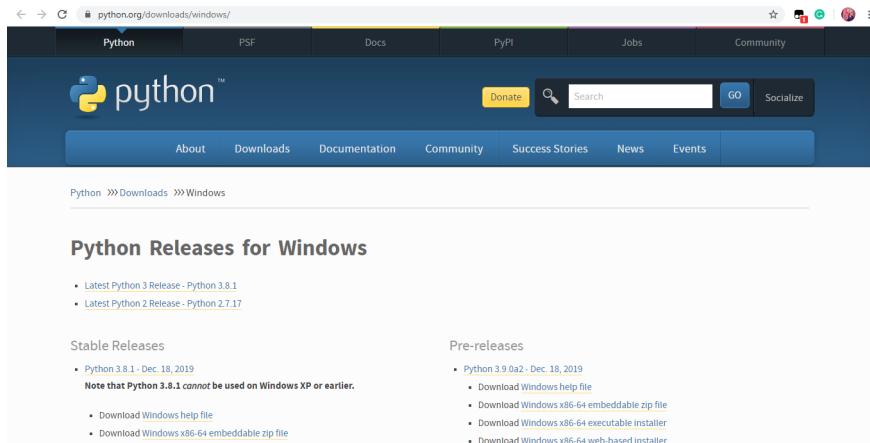
Selain perbedaan mendasar antara Python2 dan Python3 secara sintatikal, pada fakta lainnya adalah Python 2.7 akan dihentikan dukungannya per tahun 2020 dan Python3 akan terus dikembangkan dengan fitur yang lebih banyak lagi kedepannya, dan tentu saja yang lebih penting adalah perbaikan dari bug dan performa. Pengembangan saat ini telah mendukung format *string* secara literal, kustomisasi pembuatan *class* secara sederhana dan sintaksis yang lebih bersih dan rapi untuk menangani perkalian matriks.

Pengembangan Python3 akan terus berlangsung, hal ini berarti pengembangan perangkat lunak akan dengan nyaman menggunakan Python3 karena terus didukung oleh komunitas dengan jangka waktu yang panjang. Tentu saja hal ini meningkatkan kinerja programmer dan kualitas program yang lebih baik lagi.

## 1.4 Instalasi Python pada Windows

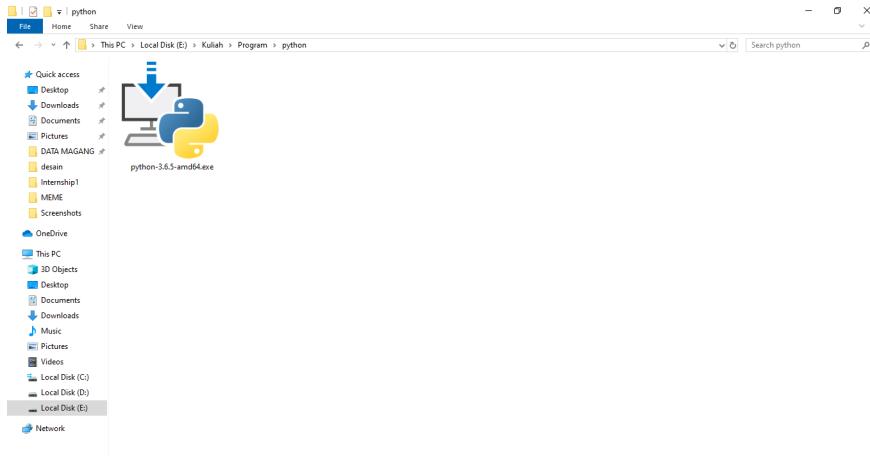
Instalasi python pada Windows sangatlah mudah. Langkah-langkanya yaitu sama seperti menginstall software Windows yang lainnya, next-next dan finish. Tetapi, terdapat konfigurasi yang harus dipilih terlebih dahulu ditengah-tengah proses instalasi, agar perintah pada Python dapat dikenali di CMD. Python yang akan di install

dalam adalah python versi 3. Sebelumnya download terlebih dahulu di situs resmi python yaitu di ([python.org](https://python.org/downloads/windows/)) dengan pilihan 64 bit atau 32 bit.



**Gambar 1.2** Website [python.org](https://python.org)

1. Buka File Python, setelah mendownload aplikasi python selesai, maka selanjutnya kita akan mendapatkan file python. File ini akan melakukan instalasi ke sistem windows. Yang perlu diketahui tidak perlu lagi menginstal pip karena di python versi 3 ini sudah ada pipnya. Yang perlu menginstal pip yaitu python dengan versi 2. Klik dua kali untuk mengeksekusinya.



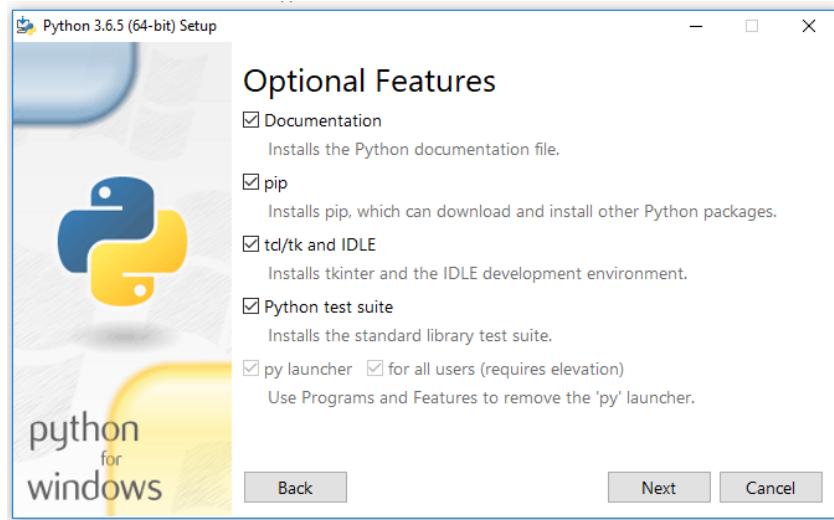
**Gambar 1.3** File Python

2. Setelah membuka aplikasi python, ceklist Add Python 3.6 to PATH dan klik Customize installation.



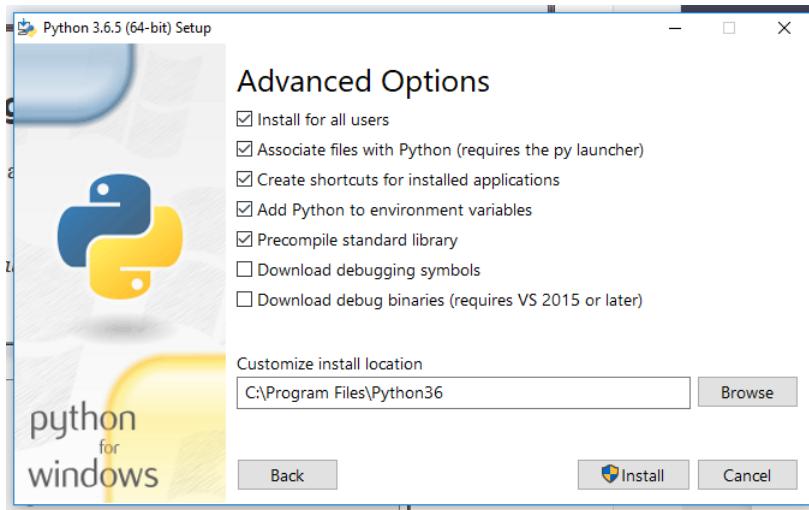
Gambar 1.4    Customize Installation

3. Ceklis atau pilih semua yaitu Documentation, pip, td/tk and IDLE, dan Python test suite. Setelah itu klik next.



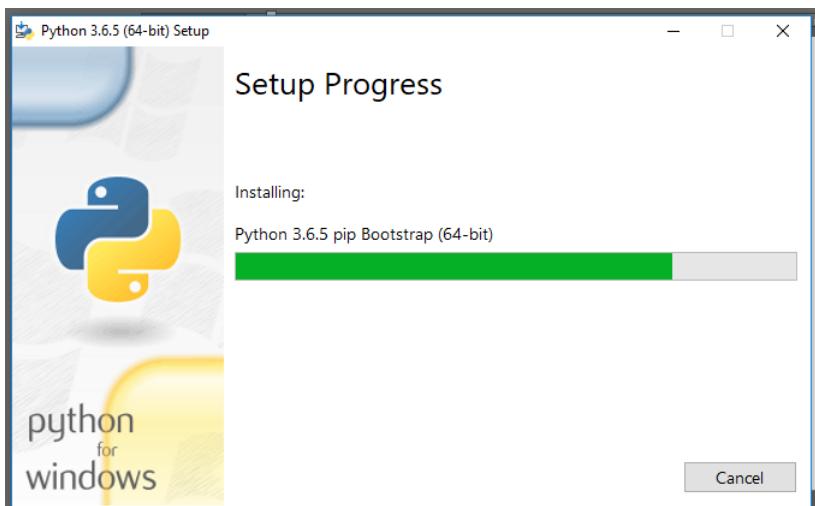
Gambar 1.5    Optional Features

4. Ceklis atau pilih sesuai yang terdapat di gambar 1.6, lalu pilih lokasi penyimpanan dan klik install.



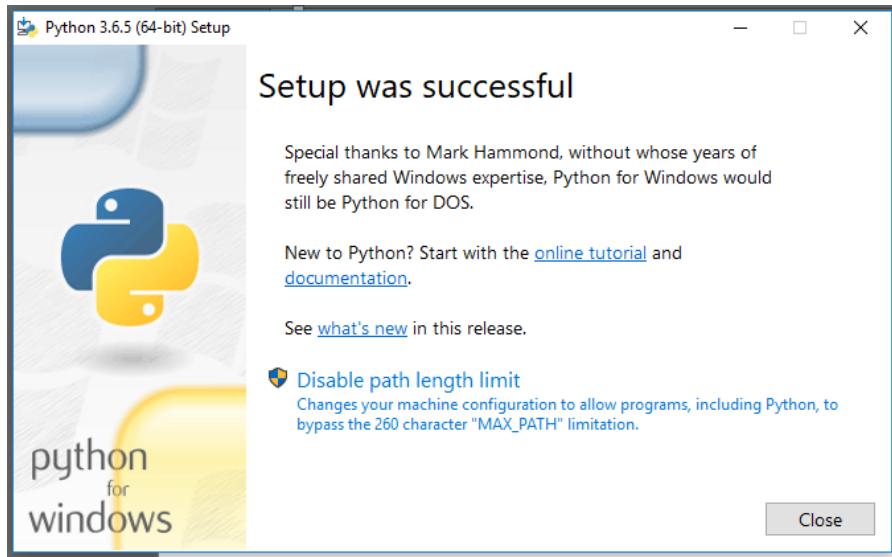
**Gambar 1.6** Advanced Options

5. Setelah di klik install akan muncul pilihan apakah kita akan menginstal atau tidak, dan pilih yes.
6. Setelah itu kita tunggu sampai instalasi selesai



**Gambar 1.7** Setup Progress

7. Instalasi selesai dan setelah itu dapat di close saja



**Gambar 1.8** Setup Was Successful

## 1.5 Syntax Dasar

Selanjutnya akan diberikan contoh fungsi Python yang digunakan untuk mencetak atau *print*. Di Python untuk mencetak cukup gunakan fungsi print(), dimana sesuatu yang akan dicetak harus diletakkan diantara kurung buka dan kurung tutup, bahkan di Python2 tidak harus menggunakan tanda kurung kurawal, tetapi cukup pisahkan saja dengan spasi. Jika hendak mencetak tipe data *String* langsung, harus memasukkannya ke dalam tanda kutip terlebih dahulu.

```
print("Hello World")
```

Saat menjalankan perintah script seperti yang terdapat diatas, akan melihat tampilan output berupa text Hello World.

## 1.6 Python Case Sensitive

Python sangat bersifat *case sensitive*, yang berarti huruf kecil dan huruf besar memiliki perbedaan. Contohnya jika menggunakan fungsi print dengan huruf kecil print() akan berhasil. Tetapi jika menggunakan huruf kapital Print() atau seperti PRINT(), akan muncul pesan *error*, ini berfungsi untuk nama variabel ataupun fungsi-fungsinya.

## 1.7 Komentar Python

Komentar merupakan kode di dalam *script* Python yang tidak akan dijalankan mesin atau tidak akan dieksekusi. Komentar hanya digunakan untuk memberikan keterangan tertulis pada script atau menandai. Komentar dapat digunakan untuk membiarkan orang lain memahami apa yang dilakukan pada script tersebut, dan atau untuk mengingatkan kepada programmer jika hendak mengedit script yang sudah ada. Untuk menggunakan komentar cukup mengetikkan tanda pagar #, dan diikuti dengan komentar yang sesuai dengan fungsi dari *script* tersebut. Komentar tidak akan dapat dieksekusi dan komentar hanya dapat digunakan untuk satu baris saja. Contoh penggunaan penggunaan komentar pada Python:

```
1 # Ini
2 # Adalah
3 # Sebuah
4 # Komentar
5 print("Hai Dunia") #ini juga komentar
6 #print("Hai")
7 #komentar
8 # bisa berisi
9 #spesial karakter $$&*() ,!@./; '[%^]\ \
10 #mencetak
11 #nama
12 print("Vidia")
13 #mencetak
14 #angka/integer
15 print(1701)
```

Script diatas akan menampilkan *output* **Hai Dunia, Vidia, dan 1701.**

## 1.8 Tipe Data pada Python

Tipe data merupakan memori pada komputer atau media yang digunakan untuk menampung sebuah informasi. Python mempunyai tipe data yang cukup unik bila bandingkan dengan bahasa pemrograman yang lainnya. Berikut merupakan beberapa tipe data dari bahasa pemrograman Python:

**Tabel 1.1** Tipe Data

Tipe Data	Contoh	Penjelasan
Float	<b>7.13</b> atau <b>0.17</b>	Menyatakan bilangan yang mempunyai koma
Complex	<b>3 + 4j</b>	Menyatakan pasangan angka real dan imajiner
String	<b>"Semangat Kawan"</b>	Menyatakan karakter/kalimat bisa berupa huruf angka, dll (diapit tanda " atau ')
Tuple	<b>('abc', 836, 5.24)</b>	Data uantaian yang menyimpan berbagai tipe data tapi isinya tidak bisa diubah
Boolean	<b>True</b> atau <b>False</b>	Menyatakan benar(True) yang bernilai 1, atau salah (False) yang bernilai 0
Integer	<b>25</b> atau <b>1209</b>	Menyatakan bilangan bulat
Hexadecimal	<b>9a</b> atau <b>1d3</b>	Menyatakan bilangan dalam format heksa (bilangan berbasis 16)
List	<b>[‘abc’, 836, 5.24]</b>	Data uantaian yang menyimpan berbagai tipe data dan isinya bisa diubah-ubah
Dictionary	<b>'nama': 'esi', 'id': 3</b>	Data uantaian yang menyimpan berbagai tipe data berupa pasangan penunjuk dan nilai

Berikut adalah contoh pengimplementasian macam-macam tipe data dengan *script* Python:

```

1 #tipe data Float
2 print(4.17)
3 #tipe data Complex
4 print(9e)
5 #tipe data String
6 print("Semangat Kawan")
7 print('Kita Pasti Bisa')
8 #tipe data Tuple
9 print((2,4,7,8,9))
10 print(("dua", "empat", "tujuh"))
11 #tipe data Boolean
12 print(False)
13 #tipe data Integer
14 print(98)
15 #tipe data Hexadecimal
16 print(7c)

```

```

17 #tipe data List
18 print([2,4,7,8,9])
19 print(["dua", "empat", "tujuh"])
20 #tipe data Dictionary
21 print({"Nama":"EsiVR", 'Umur':22})
22 #tipe data Dictionary dimasukan ke dalam variabel biodata
23 biodata = {"nama":"Andi", 'umur':21} #proses inisialisasi variabel
   biodata
24 print(biodata) #proses pencetakan variabel biodata yang berisi tipe
   data
25 Dictionary
26 type(biodata) #fungsi untuk mengecek jenis tipe data. akan tampil <
   class
27 'dict'> #yang berarti dict adalah tipe data dictionary

```

## 1.9 Variabel Python

Variabel adalah lokasi memori yang dicadangkan untuk menyimpan nilai-nilai. Ini berarti bahwa ketika Anda membuat sebuah variabel Anda memesan beberapa ruang di memori. Variabel menyimpan data yang dilakukan selama program dieksekusi, yang natinya isi dari variabel tersebut dapat diubah oleh operasi - operasi tertentu pada program yang menggunakan variabel. Variabel dapat menyimpan berbagai macam tipe data. Di dalam pemrograman Python, variabel mempunyai sifat yang dinamis, artinya variabel Python tidak perlu dideklarasikan tipe data tertentu dan variabel Python dapat diubah saat program dijalankan. Penulisan variabel Python sendiri juga memiliki aturan tertentu, yaitu :

1. Karakter pertama harus berupa huruf atau garis bawah/underscore \_
2. Karakter selanjutnya dapat berupa huruf, garis bawah/underscore \_ atau angka
3. Karakter pada nama variabel bersifat sensitif (case-sensitif). Artinya huruf kecil dan huruf besar dibedakan. Sebagai contoh, variabel namaDepan dan namadepan adalah variabel yang berbeda.

Untuk mulai membuat variabel di Python caranya sangat mudah, cukup menuliskan variabel lalu mengisinya dengan suatu nilai dengan cara menambahkan tanda sama dengan = diikuti dengan nilai yang ingin dimasukan.

Contoh penggunaan variabel dalam bahasa pemrograman Python.

```

1 #proses memasukan data ke dalam variabel
2 nama = "Dava Evar"
3 #proses mencetak variabel
4 print(nama)
5 #nilai dan tipe data dalam variabel dapat diubah
6 umur = 17 #nilai awal
7 print(umur) #mencetak nilai umur
8 type(umur) #mengecek tipe data umur
9 umur = "tujuh belas" #nilai setelah diubah
10 print(umur) #mencetak nilai umur

```

```
11 type(umur) #mengecek tipe data umur
12 namaDepan = "Arif"
13 namaBelakang = "Budiman"
14 nama = namaDepan + " " + namaBelakang
15 umur = 29
16 hobi = "Traveling"
17 print("Biodata\n", nama, "\n", umur, "\n", hobi)
18 #contoh variabel lainya
19 inivariabel = "Haaaii"
20 ini_juga_variabel = "Hooo"
21 _inivariabeljuga = "Hi"
22 inivariabel222 = "Haaa"
23 panjang = 7
24 lebar = 9
25 luas = panjang * lebar
26 print(luas)
```

## 1.10 Operator

Operator adalah konstruksi yang dapat memanipulasi nilai dari operan. Sebagai contoh operasi  $3 + 2 = 5$ . Disini 3 dan 2 adalah operan dan + adalah operator. Bahasa pemrograman Python mendukung berbagai macam operator, diantaranya :

- Operator Aritmatika (Arithmetric Operators)
- Operator Perbandingan (Comparison (Relational) Operators)
- Operator Penugasan (Assignment Operators)
- Operator Logika (Logical Operators)
- Operator Bitwise (Bitwise Operators)
- Operator Keanggotaan (Membership Operators)
- Operator Identitas (Identity Operators)

Mari kita membahasnya satu-persatu.

### 1.10.1 Operator Aritmatika

**Tabel 1.2** Tipe Data

Operator	Contoh	Penjelasan
Pengurangan -	<b>7 - 2 = 5</b>	Mengurangi nilai operan di sebelah kiri menggunakan operan di sebelah kanan
Penjumlahan +	<b>3 + 5 = 8</b>	Menjumlahkan nilai dari masing-masing operan atau bilangan
Pembagian /	<b>12 / 3 = 4</b>	Untuk membagi operan di sebelah kiri menggunakan operan di sebelah kanan
Perkalian *	<b>2 * 4 = 8</b>	Mengalikan operan/bilangan
Pangkat **	<b>8 ** 2 = 64</b>	Memangkatkan operan disebelah kiri operator dengan operan di sebelah kanan operator
Sisa Bagi %	<b>11 % 2 = 1</b>	Mendapatkan sisa pembagian dari operan di sebelah kiri operator ketika dibagi oleh operan di sebelah kanan
Pembagian Bulat //	<b>10 // 3 = 3</b>	Sama seperti pembagian. Hanya saja angka dibelakang koma dihilangkan

Contoh penggunaan Operator Aritmatika dalam bahasa pemrograman Python:

```

1 #file /python_dasar/operator_aritmatika.py
2 #OPERATOR ARITMATIKA
3 #Penjumlahan
4 print(17 + 3)
5 apel = 6
6 jeruk = 8
7 buah = apel + jeruk #
8 print(buah)
9 #Pengurangan
10 hutang = 15000
11 bayar = 9000
12 sisaHutang = hutang - bayar
13 print("Sisa hutang Anda adalah ", sisaHutang)
14 #Perkalian
15 panjang = 4
16 lebar = 13
17 luas = panjang * lebar
18 print(luas)
19 #Pembagian
20 kue = 100
21 anak = 4
22 kuePerAnak = kue / anak
23 print("Setiap anak akan mendapatkan bagian kue sebanyak ", kuePerAnak
      )
24 #Sisa Bagi / Modulus
25 bilangan1 = 16

```

```

26 bilangan2 = 7
27 hasil = bilangan1 % bilangan2
28 print("Sisa bagi dari bilangan ", bilangan1, " dan ", bilangan2, " "
      "adalah ", 
29 hasil)
30 #Pangkat
31 bilangan3 = 9
32 bilangan4 = 11
33 hasilPangkat = bilangan3 ** bilangan4
34 print(hasilPangkat)
35 #Pembagian Bulat
36 print(17//3)
37 #17 dibagi 3 adalah 5.6666. Karena dibulatkan maka akan menghasilkan
    nilai 6

```

## 1.10.2 Operator Perbandingan

Operator perbandingan (*comparison operators*) digunakan untuk membandingkan suatu nilai dari masing-masing operan.

**Tabel 1.3** Operator Perbandingan

Operator	Contoh	Penjelasan
Tidak sama dengan <>	<b>4 &lt;&gt; 4</b> bernilai <b>False</b>	Akan menghasilkan nilai kebalikan dari kondisi sebenarnya
Sama dengan ==	<b>2 == 2</b> bernilai <b>True</b>	Akan menghasilkan nilai kebalikan dari kondisi sebenarnya
Tidak sama dengan !=	<b>3 != 3</b> bernilai <b>False</b>	Akan menghasilkan nilai kebalikan dari kondisi sebenarnya
Lebih kecil dari <	<b>7 &lt; 5</b> bernilai <b>True</b>	Jika nilai operan kiri lebih kecil dari nilai operan kanan, maka kondisi menjadi benar
Lebih besar dari >	<b>7 &gt; 5</b> bernilai <b>True</b>	Jika nilai operan kiri lebih besar dari nilai operan kanan, maka kondisi menjadi benar
Lebih kecil atau sama dengan <=	<b>7 &gt; 5</b> bernilai <b>True</b>	Jika nilai operan kiri lebih kecil dari nilai operan kanan, atau sama, maka kondisi menjadi benar
Lebih besar atau sama dengan >=	<b>5 &gt;= 3</b> bernilai <b>True</b>	Jika nilai operan kiri lebih besar dari nilai operan kanan, atau sama, maka kondisi menjadi benar

### 1.10.3 Assignment Operator

Operator penugasan digunakan untuk memberikan atau memodifikasi nilai ke dalam sebuah variabel.

**Tabel 1.4** Assignment Operator

Operator	Contoh	Penjelasan
Tidak sama dengan <>	<b>2 &lt;&gt; 2</b> bernilai <b>False</b>	Akan menghasilkan nilai kebalikan dari kondisi sebenarnya
Sama dengan ==	<b>1 == 1</b> bernilai <b>True</b>	Akan menghasilkan nilai kebalikan dari kondisi sebenarnya
Tidak sama dengan !=	<b>2 != 2</b> bernilai <b>False</b>	Akan menghasilkan nilai kebalikan dari kondisi sebenarnya
Lebih kecil dari <	<b>5 &lt; 3</b> bernilai <b>True</b>	Jika nilai operan kiri lebih kecil dari nilai operan kanan, maka kondisi menjadi benar
Lebih besar dari >	<b>5 &gt; 3</b> bernilai <b>True</b>	Jika nilai operan kiri lebih besar dari nilai operan kanan, maka kondisi menjadi benar
Lebih kecil atau sama dengan <=	<b>5 &gt; 3</b> bernilai <b>True</b>	Jika nilai operan kiri lebih kecil dari nilai operan kanan, atau sama, maka kondisi menjadi benar
Lebih besar atau sama dengan >=	<b>5 &gt;= 3</b> bernilai <b>True</b>	Jika nilai operan kiri lebih besar dari nilai operan kanan, atau sama, maka kondisi menjadi benar

### 1.10.4 Logical Operator

**Tabel 1.5** Logical Operator

Operator	Contoh	Penjelasan
and	a, b = True, True # hasil akan True print a and b	Jika kedua operan bernilai True, maka kondisi akan bernilai True. Selain kondisi tadi maka akan bernilai False
or	a, b = True, False # hasil akan True print a or b print b or a print a or a # hasil akan False print b or b	Jika salah satu atau kedua operan bernilai True maka kondisi akan bernilai True. Jika keduanya False maka kondisi akan bernilai False
not	a, b = True, False # hasil akan True print not a print not b	Membalikkan nilai kebenaran pada operan misal jika asalnya True akan menjadi False dan begitupun sebaliknya

### 1.10.5 Bitwise Operator

**Tabel 1.6** Bitwise Operator

Operator	Contoh	Penjelasan
&	<pre>a, b = 13, 37 # a akan bernilai '0000 1101' print a and b a, b = True, False # hasil akan True # b akan bernilai '0010 0101' c = a &amp; b # c akan bernilai 5 = '0000 0101' print c</pre>	<p>Operator biner AND, memeriksa apakah operan di sebelah kiri dan operan sebelah kanan mempunyai angka biner 1 di setiap bit. Jika operasi akan bernilai 1</p>
	<pre>a, b = 13, 37 # a akan bernilai '0000 1101' # b akan bernilai '0010 0101' c = a   b # c akan bernilai 45 = '0010 1101' print c</pre>	<p>Operator biner OR, memeriksa apakah operan di sebelah kiri dan operan sebelah kanan mempunyai angka biner 1 di setiap bit. Jika salah satunya bernilai 1 maka bit hasil operasi akan bernilai 1</p>
^	<pre>a, b = 13, 37 # a akan bernilai '0000 1101' # b akan bernilai '0010 0101' c = a ^ b # c akan bernilai 40 = '0010 1000'</pre>	<p>Operator biner XOR, memeriksa apakah operan di sebelah kiri dan operan sebelah kanan mempunyai angka biner 1 di setiap bit. Jika keduanya bernilai 1 maka bit hasil</p>
Kali sama dengan *= a	a *= 2	Operator biner Negative, membalik nilai bit. Misal dari 1 menjadi 0, dari 0 menjadi 1
~	<pre>a, b = 13, 37 # a akan bernilai '0000 1101' # b akan bernilai '0010 0101'</pre>	Memberikan nilai variabel dengan nilai variabel itu sendiri dibagi dengan nilai di sebelah kanan
<<	<pre>a, b = 13, 37 # a akan bernilai '0000 1101' # b akan bernilai '0010 0101' # hasil bernilai 52 = "0011 0100" print a &lt;&lt; 2 # hasil bernilai 148 = '1001 0100' print b &lt;&lt; 2</pre>	Memberikan nilai variabel dengan nilai variabel itu sendiri dibagi dengan nilai di sebelah kanan. Yang diambil nantinya adalah sisa baginya

**Tabel 1.7** Lanjutan Tabel Bitwise Operator

>>	a, b = 13, 37 # a akan bernilai '0000 1101' # b akan bernilai '0010 0101' # hasil bernilai 3 = '0000 0011' print a >> 2 # hasil bernilai 9 = '0000 1001' print b >> 2	Operator penggeser biner ke kiri, deret bit akan digeser ke kiri sebanyak n kali
----	---	--

### 1.10.6 Membership Operator

**Tabel 1.8** Membership Operator

in	sebuah_list = [1, 2, 3, 4, 5] print 5 in sebuah_list	Memeriksa apakah nilai yang dicari berada pada list atau struktur data python lainnya. Jika nilai tersebut ada maka kondisi akan bernilai True
not in	sebuah_list = [1, 2, 3, 4, 5] print 10 not in sebuah_list	Memeriksa apakah nilai yang dicari tidak ada pada list atau struktur data python lainnya. Jika nilai tersebut tidak ada maka kondisi akan bernilai True

### 1.10.7 Identity Operator

**Tabel 1.9** Identity Operator

is	a, b = 10, 10 # hasil akan True print a is b	Memeriksa apakah nilai di sebelah kiri operan memiliki identitas memori yang sama dengan nilai di sebelah kanan operan. Jika sama maka kondisi bernilai True.
is not	a, b = 10, 5 # hasil akan True print a is not b	Memeriksa apakah nilai di sebelah kiri operan memiliki identitas memori yang berbeda dengan nilai di sebelah kanan operan. Jika berbeda maka kondisi bernilai True

## 1.11 Konfisi If

Pengambilan keputusan (kondisi if) digunakan untuk mengantisipasi kondisi yang terjadi saat jalanya program dan menentukan tindakan apa yang akan diambil sesuai dengan kondisi. Pada python ada beberapa statement/kondisi diantaranya adalah if, else dan elif Kondisi if digunakan untuk mengeksekusi kode jika kondisi bernilai benar. Jika kondisi bernilai salah maka statement/kondisi if tidak akan di-eksekusi. Dibawah ini adalah contoh penggunaan kondisi if pada Python

```

1 #Kondisi if adalah kondisi yang akan dieksekusi oleh program jika
2 bernilai
3 benar atau TRUE
4 nilai = 9
5 #jika kondisi benar/TRUE maka program akan mengeksekusi perintah
6 dibawahnya
7 if(nilai > 7):
8     print("Selamat Anda Lulus")
9 #jika kondisi salah/FALSE maka program tidak akan mengeksekusi
10    perintah
11 dibawahnya
12 if(nilai > 10):
13     print("Selamat Anda Lulus")

```

Dari contoh diatas, jika program dijalankan maka akan mencetak string "Selamat Anda Lulus Ujian" sebanyak 1 kali yaitu pada if pertama. Di if kedua statement bernilai salah, jadi perintah print("Selamat Anda Lulus") tidak akan dieksekusi.

## 1.12 If Else

Pengambilan keputusan (kondisi if else) tidak hanya digunakan untuk menentukan tindakan apa yang akan diambil sesuai dengan kondisi, tetapi juga digunakan untuk menentukan tindakan apa yang akan diambil/dijalankan jika kondisi tidak sesuai. Pada python ada beberapa statement/kondisi diantaranya adalah if, else dan elif Kondisi if digunakan untuk mengeksekusi kode jika kondisi bernilai benar. Kondisi if else adalah kondisi dimana jika pernyataan benar (true) maka kode dalam if akan dieksekusi, tetapi jika bernilai salah (false) maka akan mengeksekusi kode di dalam else. Contoh penggunaan kondisi if else pada Python:

```

1 #Kondisi if else adalah jika kondisi bernilai TRUE maka akan
2 dieksekusi pada
3 if , tetapi jika bernilai FALSE maka akan dieksekusi kode pada else
4 nilai = 3
5 #Jika pernyataan pada if bernilai TRUE maka if akan dieksekusi ,
6     tetapi jika
7 FALSE kode pada else yang akan dieksekusi .
8 if(nilai > 7):
9     print("Selamat Anda Lulus")
10 else:
11     print("Maaf Anda Tidak Lulus")

```

Pada contoh diatas, jika program dijalankan maka akan mencetak string "Maaf Anda Tidak Lulus" karena pernyataan pada if bernilai FALSE

## 1.13 Kondisi Elif

Pengambilan keputusan (kondisi if elif) merupakan lanjutan/percabangan logika dari "kondisi if". Dengan elif kita bisa membuat kode program yang akan menyeleksi beberapa kemungkinan yang bisa terjadi. Hampir sama dengan kondisi "else", banyaknya kondisi "elif" bisa banyak dan tidak hanya satu. Contoh penggunaan kondisi elif pada Python:

```
1 #Contoh penggunaan kondisi elif
2 hari_ini = "Minggu"
3 if(hari_ini == "Senin"):
4     print("Saya akan kuliah")
5 elif(hari_ini == "Selasa"):
6     print("Saya akan kuliah")
7 elif(hari_ini == "Rabu"):
8     print("Saya akan kuliah")
9 elif(hari_ini == "Kamis"):
10    print("Saya akan kuliah")
11 elif(hari_ini == "Jumat"):
12    print("Saya akan kuliah")
13 elif(hari_ini == "Sabtu"):
14    print("Saya akan kuliah")
15 elif(hari_ini == "Minggu"):
16    print("Saya akan libur")
```

Pada contoh diatas, jika program dijalankan maka akan mencetak string "Saya akan libur".

## 1.14 Pengulangan "Loop"

Secara umum, pernyataan pada bahasa pemrograman akan dieksekusi secara berurutan. Pernyataan pertama dalam sebuah fungsi dijalankan pertama, diikuti oleh yang kedua, dan seterusnya. Tetapi akan ada situasi dimana Anda harus menulis banyak kode, dimana kode tersebut sangat banyak. Jika dilakukan secara manual maka Anda hanya akan membuang-buang tenaga dengan menulis beratus-ratus bahkan berribu-ribu kode. Untuk itu Anda perlu menggunakan pengulangan di dalam bahasa pemrograman Python.

Di dalam bahasa pemrograman Python pengulangan dibagi menjadi 3 bagian, yaitu :

- While Loop

- For Loop

- Nested Loop

### 1.14.1 Pengulangan While

Pengulangan While Loop di dalam bahasa pemrograman Python dieksekusi statement berkali-kali selama kondisi bernilai benar atau True. Contoh penggunaan pengulangan While Loop.

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

### 1.15 Pengulangan For

Pengulangan For pada Python memiliki kemampuan untuk mengulangi item dari urutan apapun, seperti list atau string. Contoh penggunaan pengulangan While Loop:

```
1 #Contoh pengulangan for sederhana
2 angka = [1,2,3,4,5]
3 for x in angka:
4     print(x)
5 #Contoh pengulangan for
6 buah = ["nanas", "apel", "jeruk"]
7 for makanan in buah:
8     print("Saya suka makan", makanan)
```

### 1.16 Pengulangan Bersarang (Nested Loop)

Bahasa pemrograman Python memungkinkan penggunaan satu lingkaran di dalam loop lain. Bagian berikut menunjukkan beberapa contoh untuk menggambarkan konsep tersebut. Contoh penggunaan Nested Loop.

```
1 #Contoh penggunaan Nested Loop
2 i = 2
3 while(i < 100):
4     j = 2
5     while(j <= (i/j)):
6         if not(i%j): break
7         j = j + 1
8     if (j > i/j) : print i, " is prime"
9     i = i + 1
10 print "Good bye!"
```

### 1.17 Number Python

Number adalah tipe data Python yang menyimpan nilai numerik. Number adalah tipe data yang tidak berubah. Ini berarti, mengubah nilai dari sejumlah tipe data

akan menghasilkan objek yang baru dialokasikan. Objek Number dibuat saat Anda memberikan nilai pada-nya. Sebagai contoh : angkaPertama = 1 angkaKedua = 33

Python mendukung beberapa tipe data Number diantaranya :

- Int
- Float
- Complex

Berikut ini adalah beberapa contoh dari Tipe data Number pada Python :

**Tabel 1.10** Tipe Data Number Pada Python

Int	Float	Complex
20	0.1	3.14j
300	1.20	35.j
-13	-41.2	3.12e-12j
020	32.23+e123	.873j
-0103	-92.	-.123+0J
-0x212	-32.52e10	3e+123J
0x56	60.2-E13	4.31e-4j

### 1.17.1 Konversi Tipe Data Number Python

Pada Python Anda bisa mengkonversi tipe data dengan menggunakan fungsi. Dibawah ini adalah beberapa fungsi untuk mengkonversi tipe data number Python.

- `int(x)`  
untuk meng-konversi x menjadi plain integer.
- `long(x)`  
untuk meng-konversi x menjadi long integer.
- `float(x)`  
untuk meng-konversi x menjadi floating point number.
- `complex(x)`  
untuk meng-konversi x menjadi complex number dengan real part x dan imaginary part zero.
- `complex(x, y)`  
untuk meng-konversi x dan y menjadi complex number dengan real part x dan imaginary part y. x dan numeric expressions y.

### 1.17.2 Fungsi Matematika

Pada bahasa pemrograman Python terdapat fungsi untuk melakukan perhitungan matematis, berikut adalah daftarnya :

**Tabel 1.11** Fungsi Matematika

Nama	Pengguna	Penjelasan
Absolute	abs(x)	Nilai absolut dari x:(positive) jarak antara x and 0.
Ceiling	ceil(x)	Ceiling dari x: integer terkecil yang kurang dari x.
Cmp	cmp(x, y)	-1 if $x < y$ , 0 if $x == y$ , or 1 if $x > y$ . Tidak berlaku lagi dengan Python 3. Sebaliknya gunakan return $(x>y)-(x$
Eksponen	exp(x)	Nilai eksponen dari x: $e^x$
Fabs	fabs(x)	Nilai absolut dari x.
Floor	floor(x)	Nilai dasar dari x: internet terbesar tidak lebih besar dari x
Log	log(x)	Logaritma dari x, untuk $x > 0$ .
Log 10	log10(x)	Basis 10 logaritma dari x, untuk $x > 0$
Max	max(x1, x2,...)	Argumen terbesar: Nilai terdekat dengan tak terhingga positif
Min	min(x1, x2,...)	Argumen terkecil: nilai yang paling mendekati tak berhingga negatif.
Modf	modf(x)	Bagian pecahan dan bilangan bulat dari x dalam tupel dua item. Kedua bagian memiliki tanda yang sama dengan x. Bagian integer dikembalikan sebagai float.
Pow	pow(x, y)	Nilai $x ^ y$ .
Round	round(x [,n])	X dibulatkan menjadi n digit dari titik desimal. Putaran Python jauh dari nol sebagai tie-breaker: round (0.5) adalah 1.0 dan round (-0.5) adalah -1.0.
Akar Kuadrat	sqrt(x)	Akar kuadrat x untuk $x > 0$ .

## 1.18 Fungsi Nomor Acak

Nomor acak digunakan untuk aplikasi permainan, simulasi, pengujian, keamanan, dan privasi. Python mencakup fungsi berikut yang umum digunakan. Berikut adalah daftarnya :

**Tabel 1.12** Fungsi Nomor Acak

Nama	Penggunaan	Penjelasan
Choice	<b>choice(seq)</b>	Item acak dari list, tuple, atau string.
RandRange	<b>randrange ([start,] stop [,step])</b>	Elemen yang dipilih secara acak dari jangkauan (start, stop, step).
Random	<b>random()</b>	A random float r, sehingga 0 kurang dari atau sama dengan r dan r kurang dari 1
Seed	<b>seed([x])</b>	Menetapkan nilai awal integer yang digunakan dalam menghasilkan bilangan acak. Panggil fungsi ini sebelum memanggil fungsi modul acak lainnya. Tidak ada pengembalian
Shuffle	<b>shuffle(lst)</b>	Mengacak daftar dari daftar di tempat. Tidak ada pengembalian
Floor	<b>floor(x)</b>	The floor of x: the largest integer not greater than x.
Uniform	<b>uniform(x, y)</b>	Sebuah float acak r, sedemikian rupa sehingga x kurang dari atau sama dengan r dan r kurang dari y.

### 1.18.1 Fungsi Trigonometri

Python mencakup fungsi berikut yang melakukan perhitungan trigonometri. Berikut adalah daftarnya :

**Tabel 1.13** Fungsi Trigonometri

Nama	Penggunaan	Penjelasan
Acos	<b>acos(x)</b>	Kembalikan kosinus x, di radian.
Asin	<b>asin(x)</b>	Kembalikan busur sinus x, dalam radian.
Atan	<b>atan(x)</b>	Kembalikan busur singgung x, di radian.
Atan 2	<b>atan2(y, x)</b>	Kembali atan ( $y / x$ ), di radian
Kosinus	<b>cos(x)</b>	Kembalikan kosinus x radian.
Hypot	<b>hypot(x, y)</b>	Kembalikan norma Euclidean, $\sqrt{x * x + y * y}$ .
Sin	<b>sin(x)</b>	Kembalikan sinus dari x radian.
Tan	<b>tan(x)</b>	Kembalikan tangen x radian.
Derajat	<b>degrees(x)</b>	Mengonversi sudut x dari radian ke derajat.
Radian	<b>radians(x)</b>	Mengonversi sudut x dari derajat ke radian.

### 1.18.2 Konstanta Matematika

Modul ini juga mendefinisikan dua konstanta matematika. Berikut adalah daftarnya :

**Tabel 1.14** Konstanta Matematika

Nama	Penggunaan	Penjelasan
Pi	<b>pi</b>	Konstanta Pi matematika
e	<b>e</b>	Konstanta e matematika

## 1.19 String

String adalah jenis yang paling populer di bahasa pemrograman. Kita bisa membuatnya hanya dengan melampirkan karakter dalam tanda kutip. Python memperlakukan tanda kutip tunggal sama dengan tanda kutip ganda. Membuat string semudah memberi nilai pada sebuah variabel.

Dibawah ini adalah contoh sederhana dari sebuah string pada bahasa pemrograman Python.

```
1 print("Hello World")
```

Python tidak menggunakan tipe karakter titik koma ; Ini diperlakukan sebagai string dengan panjang satu, sehingga juga dianggap sebagai substring.

Untuk mengakses substring, gunakan tanda kurung siku untuk mengiris beserta indeks atau indeks untuk mendapatkan substring Anda. Sebagai contoh :

```
1 name = 'Esi Vidia' message = "Esi harus tetap semangat dalam belajar"
2 print ("name[0]: ", name[0])
3 print ("message[1:4]: ", message[1:4])
```

Bila kode diatas dieksekusi, maka akan menghasilkan hasil sebagai berikut :

**name[0]: E**

**message[1:4]: si**

### 1.19.1 Mengupdate String

Anda dapat ”memperbarui” string yang ada dengan (kembali) menugaskan variabel ke string lain. Nilai baru dapat dikaitkan dengan nilai sebelumnya atau ke string yang sama sekali berbeda sama sekali. Sebagai contoh

```
1 message = 'Hello World'
2 print ("Updated String :- ", message[:6] + 'Python')
```

Bila kode diatas dieksekusi, maka akan menghasilkan hasil sebagai berikut :

**Updated String:- Hello Python**

### 1.19.2 Escape Character

Dibawah ini adalah tabel dari daftar karakter escape atau karakter non-printable yang dapat diwakili/ditulis dengan awalan notasi backslash.

**Tabel 1.15** Escape Character

Notasi Backslash	Karakter Hexadecimal	Penjelasan
a	0x07	Bell atau alert
b	0x08	Backspace
cx		Control-x
C-x		Control-x
e	0x1b	Escape
f	0x0c	Formfeed
M- C-x		Meta-Control-x
n	0x0a	Newline
nnn		Octal notation, dimana n berada di range 0..7
r	0x0d	Carriage return
s	0x20	Space
t	0x09	Tab
v	0x0b	Vertical tab
x		Character x
xnn	asdafsdfsdf	Notasi Hexadecimal, dimana n berada di range 0..9, a..f, atau A..F

### 1.19.3 Operator Spesial String

Asumsikan variabel string adalah 'Belajar' dan variabel b adalah 'Python', lalu dibawah ini adalah operator yang bisa dipakai pada kedua string di variabel tersebut.

**a = "Belajar"**

**b = "Python"**

Berikut adalah daftar operator spesial string pada Python :

**Tabel 1.16** daftar operator spesial string

Operator	Contoh	Penjelasan
<b>+</b>	a + b akan menghasilkan BelajarPython	Concatenation - Menambahkan nilai pada kedua sisi operator
<b>*</b>	a*2 akan menghasilkan BelajarBelajar	Pengulangan - Membuat string baru, menggabungkan beberapa salinan dari string yang sama
<b>[]</b>	a[1] akan menghasilkan e	Slice - Memberikan karakter dari indeks yang diberikan
<b>[ : ]</b>	a[1:4] akan menghasilkan ela	Range Slice - Memberikan karakter dari kisaran yang diberikan
<b>in</b>	B in a akan menghasilkan 1	Keanggotaan - Mengembalikan nilai true jika ada karakter dalam string yang diberikan
<b>not in</b>	Z not in a akan menghasilkan 1	Keanggotaan - Mengembalikan nilai true jika karakter tidak ada dalam string yang diberikan
<b>r/R</b>	print r'n' prints \n dan print R'	Raw String - Menekan arti aktual karakter Escape. Sintaks untuk string mentah sama persis dengan string biasa kecuali operator string mentah, huruf "r", yang mendahului tanda petik. "R" bisa berupa huruf kecil (r) atau huruf besar (R) dan harus ditempatkan tepat sebelum tanda kutip pertama.
<b>%</b>		Format - Melakukan format String

#### 1.19.4 Operator Format String

Salah satu fitur Python yang paling keren adalah format string operator %. Operator ini unik untuk string dan membuat paket memiliki fungsi dari keluarga printf C () C. Berikut adalah contoh sederhananya :

```
print ("My name is %s and weight is %d kg!" % ('Zara', 21))
```

Berikut adalah daftar lengkap simbol yang bisa digunakan bersamaan dengan % :

**Tabel 1.17** daftar simbol

<b>%c</b>	character
<b>%s</b>	Konversi string melalui str () sebelum memformat
<b>%i</b>	Dianggap sebagai bilangan bulat desimal
<b>%d</b>	Dianggap sebagai bilangan bulat desimal
<b>%u</b>	Unsigned decimal integer
<b>%o</b>	Bilangan bulat oktal
<b>%x</b>	Bilangan bulat heksadesimal (huruf kecil)
<b>%X</b>	Bilangan bulat heksadesimal (huruf besar)
<b>%e</b>	Notasi eksponensial (dengan huruf kecil 'e')
<b>%E</b>	Notasi eksponensial (dengan huruf besar 'E')
<b>%f</b>	Bilangan real floating point
<b>%g</b>	Yang lebih pendek dari% f dan% e
<b>%G</b>	Lebih pendek dari% f dan% E

#### 1.19.5 Triple Quote

Python triple quotes digunakan dengan membiarkan string untuk ditulis dalam beberapa baris, termasuk kata kerja NEWLINEs, TABs, dan karakter khusus lainnya. Sintaks untuk triple quotes terdiri dari tiga tanda kutip tunggal atau ganda ditulis berturut-turut.

Berikut adalah contohnya :

```
1 kutipantiga = """this is a long string that is made up of
2 several lines and non-printable characters such as
3 TAB (\t) and they will show up that way when displayed.
4 NEWLINEs within the string , whether explicitly given like
5 this within the brackets [ \n ], or just a NEWLINE within
6 the variable assignment will also show up.
7 """
8 print (kutipantiga)
```

### 1.19.6 String Unicode

Pada Python 3, semua string diwakili dalam Unicode. Sedangkan pada Python 2 disimpan secara internal sebagai 8-bit ASCII, maka diperlukan lampiran 'u' untuk membuatnya menjadi Unicode. Tetapi hal ini tidak lagi diperlukan sekarang.

#### Metode String Built-in

Python menyertakan metode built-in berikut untuk memanipulasi string

**Tabel 1.18** Metode built-in

Metode	Penjelasan
<b>capitalize()</b>	Meng-kapitalkan huruf pertama string
<b>center(width, fillchar)</b>	Mengembalikan string yang dilapisi dengan fillchar dengan string asli yang dipusatkan pada total width kolom.
<b>count(str, beg = 0,end = len(string))</b>	Menghitung berapa kali str yang terjadi dalam string atau dalam substring string jika memulai indeks <b>beg</b> dan end index <b>end</b> diberikan.
<b>decode(encoding = 'UTF 8',errors = 'strict')</b>	Dekode string menggunakan codec yang terdaftar untuk pengkodean. Encoding default ke pengkodean string default.
<b>encode(encoding = 'UTF 8',errors = 'strict')</b>	Mengembalikan versi string yang dikodekan string; Pada kesalahan, default adalah menaikkan ValueError kecuali jika kesalahan diberikan dengan 'ignore' atau 'replace'.
<b>endswith(suffix, beg = 0, end = len(string))</b>	Menentukan apakah string atau substring string (jika memulai indeks memohon dan mengakhiri akhir indeks diberikan) berakhir dengan akhiran; Mengembalikan nilai true jika benar dan salah.
<b>expandtabs(tabsize = 8)</b>	Memperluas tab dalam string ke banyak ruang; Default ke 8 spasi per tab jika tabsize tidak tersedia.
<b>find(str, beg = 0 end = len(string))</b>	Tentukan jika str terjadi dalam string atau dalam substring string jika memulai indeks beg dan end index end diberikan return index jika ditemukan dan -1 sebaliknya
<b>index(str, beg = 0, end = len(string))</b>	Sama seperti find (), namun menimbulkan pengecualian jika str tidak ditemukan.
<b>isalnum()</b>	Mengembalikan true jika string memiliki minimal 1 karakter dan semua karakternya alfanumerik dan false sebaliknya.

**Tabel 1.19** Lanjutan Tabel Metode built-in

<b>isalpha()</b>	Mengembalikan true jika string memiliki minimal 1 karakter dan semua karakter adalah abjad dan false sebaliknya.
<b>isdigit()</b>	Mengembalikan true jika string hanya berisi digit dan false sebaliknya.
<b>islower()</b>	Mengembalikan true jika string memiliki setidaknya 1 karakter casing dan semua karakter casing dalam huruf kecil dan false sebaliknya
<b>isnumeric()</b>	Mengembalikan true jika string unicode hanya berisi karakter numerik dan false sebaliknya.
<b>isspace()</b>	Mengembalikan true jika string hanya berisi karakter spasi dan false sebaliknya.
<b>istitle()</b>	Mengembalikan true jika string benar "titlecased" dan false sebaliknya.
<b>isupper()</b>	Mengembalikan true jika string memiliki setidaknya satu karakter casing dan semua karakter casing ada dalam huruf besar dan false sebaliknya.
<b>join(seq)</b>	Merges (concatenates) representasi string elemen dalam urutan seq menjadi string, dengan string pemisah.
<b>len(string)</b>	Mengembalikan panjang string
<b>ljust(width[, fillchar])</b>	Mengembalikan string berlapis ruang dengan string asli dibiarkan dibenarkan ke kolom lebar total
<b>lower()</b>	Mengonversi semua huruf besar dalam bentuk string menjadi huruf kecil.
<b>lstrip()</b>	Menghapus semua spasi utama dalam string.
<b>maketrans()</b>	Mengembalikan tabel terjemahan untuk digunakan dalam fungsi terjemahan.
<b>max(str)</b>	Mengembalikan karakter alfabetik dari string str
<b>min(str)</b>	Mengembalikan min karakter abjad dari string str.
<b>replace(old, new [, max])</b>	Menggantikan semua kemunculan lama dalam string dengan kejadian baru atau paling maksimal jika max diberikan.
<b>rfind(str, beg = 0,end = len(string))</b>	Sama seperti find (), tapi cari mundur dalam string.
<b>rindex( str, beg = 0, end = len(string))</b>	Sama seperti index (), tapi cari mundur dalam string.

**Tabel 1.20** Lanjutan Tabel Metode built-in

<b>rjust(width,[, fillchar])</b>	Mengembalikan string berlapis ruang dengan senar asli benar-dibenarkan untuk total kolom lebar.
<b>rstrip()</b>	Menghapus semua spasi spasi string.
<b>split(str="", num=string.count(str))</b>	Membagi string sesuai dengan pemisah str (ruang jika tidak disediakan) dan mengembalikan daftar substring; Terpecah menjadi paling banyak substring jika diberikan.
<b>splitlines( num=string.count("") )</b>	Membagi string sama sekali (atau num) NEWLINEs dan mengembalikan daftar setiap baris dengan NEWLINEs dihapus
<b>startswith(str, beg=0,end=len(string))</b>	Determines if string or a substring of string (if starting index beg and ending index end are given) starts with substring str; returns true if so and false otherwise.
<b>strip([chars])</b>	Lakukan kedua lstrip () dan rstrip () pada string
<b>swapcase()</b>	Kasus invers untuk semua huruf dalam string.
<b>title()</b>	Mengembalikan versi string "titlecased", yaitu, semua kata diawali dengan huruf besar dan sisanya huruf kecil.
<b>translate(table, deletechars="")</b>	Menerjemahkan string sesuai dengan tabel terjemahan str (256 karakter), menghapus string del.
<b>upper()</b>	Mengonversi huruf kecil dalam bentuk string ke huruf besar.
<b>zfill (width)</b>	Mengembalikan string asli yang tertinggal dengan angka nol ke total karakter lebar; Dimaksudkan untuk angka, zfill () mempertahankan tanda apapun yang diberikan (kurang satu nol).
<b>isdecimal()</b>	Mengembalikan nilai true jika string unicode hanya berisi karakter desimal dan false sebaliknya.

## 1.20 List

Dalam bahasa pemrograman Python, struktur data yang paling dasar adalah urutan atau lists. Setiap elemen-elemen berurutan akan diberi nomor posisi atau indeksnya. Indeks pertama dalam list adalah nol, indeks kedua adalah satu dan seterusnya.

Python memiliki enam jenis urutan built-in, namun yang paling umum adalah list dan tuple. Ada beberapa hal yang dapat Anda lakukan dengan semua jenis list. Operasi ini meliputi pengindeksan, pengiris, penambahan, perbanyak, dan pengecekan keanggotaan. Selain itu, Python memiliki fungsi built-in untuk menemukan panjang list dan untuk menemukan elemen terbesar dan terkecilnya.

### 1.20.1 Membuat List Python

List adalah tipe data yang paling serbaguna yang tersedia dalam bahasa Python, yang dapat ditulis sebagai daftar nilai yang dipisahkan koma (item) antara tanda kurung siku. Hal penting tentang daftar adalah item dalam list tidak boleh sama jenisnya.

Membuat list sangat sederhana, tinggal memasukkan berbagai nilai yang dipisahkan koma di antara tanda kurung siku. Dibawah ini adalah contoh sederhana pembuatan list dalam bahasa Python.

```
1 #Contoh sederhana pembuatan list pada bahasa pemrograman python
2 list1 = ['kimia', 'fisika', 1993, 2017]
3 list2 = [1, 2, 3, 4, 5]
4 list3 = ["a", "b", "c", "d"]
```

### 1.20.2 Akses Nilai Dalam List

Untuk mengakses nilai dalam list python, gunakan tanda kurung siku untuk mengeiris beserta indeks atau indeks untuk mendapatkan nilai yang tersedia pada indeks tersebut.

Berikut adalah contoh cara mengakses nilai di dalam list python :

```
1 #Cara mengakses nilai di dalam list Python
2 list1 = ['fisika', 'kimia', 1993, 2017]
3 list2 = [1, 2, 3, 4, 5, 6, 7]
4 print ("list1[0]: ", list1[0])
5 print ("list2[1:5]: ", list2[1:5])
```

Setelah Anda mengeksekusi kode diatas, hasilnya akan seperti dibawah ini :

**list1[0]: fisika**

**list2[1:5]: [2, 3, 4, 5]**

### 1.20.3 Update Nilai Dalam List

Anda dapat memperbarui satu atau beberapa nilai di dalam list dengan memberikan potongan di sisi kiri operator penugasan, dan Anda dapat menambahkan nilai ke dalam list dengan metode `append()`. Sebagai contoh :

```

1 list = ['fisika', 'kimia', 1993, 2017]
2 print ("Nilai ada pada index 2 : ", list[2])
3 list[2] = 2001
4 print ("Nilai baru ada pada index 2 : ", list[2])

```

### 1.20.4 Hapus Nilai Dalam List

Untuk menghapus nilai di dalam list python, Anda dapat menggunakan salah satu pernyataan `del` jika Anda tahu persis elemen yang Anda hapus. Anda dapat menggunakan metode `remove()` jika Anda tidak tahu persis item mana yang akan dihapus. Sebagai contoh :

```

1 #Contoh cara menghapus nilai pada list python
2 list = ['fisika', 'kimia', 1993, 2017]
3 print (list)
4 del list[2]
5 print ("Setelah dihapus nilai pada index 2 : ", list)

```

## 1.21 Operasi Dasar

List Python merespons operator + dan \* seperti string; Itu artinya penggabungan dan pengulangan di sini juga berlaku, kecuali hasilnya adalah list baru, bukan sebuah String. Sebenarnya, list merespons semua operasi urutan umum yang kami gunakan pada String di bab sebelumnya. Dibawah ini adalah tabel daftar operasi dasar pada list python.

Tabel 1.21 Operasi Dasar

Python Expression	Hasil	Penjelasan
<code>len([1, 2, 3, 4])</code>	4	Length
<code>[1, 2, 3] + [4, 5, 6]</code>	<code>[1, 2, 3, 4, 5, 6]</code>	Concatenation
<code>'Halo!' * 4</code>	<code>'Halo!', 'Halo!', 'Halo!', 'Halo!'</code>	Repetition
<code>2 in [1, 2, 3]</code>	True	Membership
<code>for x in [1,2,3] :</code>	<code>1 2 3</code>	Iteration
<code>print (x,end = ' ')</code>		

### 1.21.1 Indexing, Slicing dan Matrix pada List Python

Karena list adalah urutan, pengindeksan dan pengiris bekerja dengan cara yang sama untuk list seperti yang mereka lakukan untuk String.

Dengan asumsi input berikut :

**L = ['C++', 'Java', 'Python']**

**Tabel 1.22** Indexing, Slincing dan Matrix pada List Python

Python Expression	Hasil	Penjelasan
<b>L[2]</b>	<b>'Python'</b>	Offset mulai dari nol
<b>L[-2]</b>	<b>'Java'</b>	Negatif: hitung dari kanan
<b>L[1:]</b>	<b>['Java', 'Python']</b>	Slicing mengambil bagian

### 1.21.2 Method dan Fungsi Build-in pada List Python

Python menyertakan fungsi built-in sebagai berikut

**Tabel 1.23** Fungsi built-in

Python Function	Penjelasan
<b>cmp(list1, list2)</b>	# Tidak lagi tersedia dengan Python 3
<b>len(list)</b>	Memberikan total panjang list.
<b>max(list)</b>	Mengembalikan item dari list dengan nilai maks
<b>min(list)</b>	Mengembalikan item dari list dengan nilai min.
<b>list(seq)</b>	Mengubah tuple menjadi list.

Python menyertakan methods built-in sebagai berikut:

**Tabel 1.24** Methods built-in

Python Method	Penjelasan
<b>list.append(obj)</b>	Menambahkan objek obj ke list
<b>list.count(obj)</b>	Jumlah pengembalian berapa kali obj terjadi dalam list
<b>list.extend(seq)</b>	Tambahkan isi seq ke list
<b>list.index(obj)</b>	Mengembalikan indeks terendah dalam list yang muncul obj
<b>list.insert(index, obj)</b>	Sisipkan objek obj ke dalam list di indeks offset
<b>list.pop(obj = list[-1])</b>	Menghapus dan mengembalikan objek atau obj terakhir dari list
<b>list.remove(obj)</b>	Removes object obj from list
<b>list.reverse()</b>	Membalik list objek di tempat
<b>list.sort([func])</b>	Urutkan objek list, gunakan compare func jika diberikan

## 1.22 Tuple

Sebuah tupel adalah urutan objek Python yang tidak berubah. Tupel adalah urutan, seperti daftar. Perbedaan utama antara tupel dan daftarnya adalah bahwa tupel tidak dapat diubah tidak seperti List Python. Tupel menggunakan tanda kurung, sedangkan List Python menggunakan tanda kurung siku.

Membuat tuple semudah memasukkan nilai-nilai yang dipisahkan koma. Secara opsional, Anda dapat memasukkan nilai-nilai yang dipisahkan koma ini di antara tanda kurung juga. Sebagai contoh :

```
1 #Contoh sederhana pembuatan tuple pada bahasa pemrograman python
2 tup1 = ('fisika', 'kimia', 1993, 2017)
3 tup2 = (1, 2, 3, 4, 5 )
4 tup3 = "a", "b", "c", "d"
```

Tupel kosong ditulis sebagai dua tanda kurung yang tidak berisi apa-apa, contohnya :

**tup1 = ()**;

Untuk menulis tupel yang berisi satu nilai, Anda harus memasukkan koma, meskipun hanya ada satu nilai, contohnya :**tup1 = (50,**

Seperti indeks String, indeks tuple mulai dari 0, dan mereka dapat diiris, digabungkan, dan seterusnya.

### 1.22.1 Akses Nilai Dalam Tuple

Untuk mengakses nilai dalam tupel, gunakan tanda kurung siku untuk mengiris bersama indeks atau indeks untuk mendapatkan nilai yang tersedia pada indeks tersebut. Sebagai contoh :

```

1 #Cara mengakses nilai tuple
2 tup1 = ('fisika', 'kimia', 1993, 2017)
3 tup2 = (1, 2, 3, 4, 5, 6, 7)
4 print ("tup1[0]: ", tup1[0])
5 print ("tup2[1:5]: ", tup2[1:5])

```

Setelah Anda mengeksekusi kode diatas, hasilnya akan seperti dibawah ini :

**tup1[0]: fisika**  
**tup2[1:5]: (2, 3, 4, 5)**

### 1.23 Update Nilai Dalam Tuple

Tupel tidak berubah, yang berarti Anda tidak dapat memperbarui atau mengubah nilai elemen tupel. Anda dapat mengambil bagian dari tupel yang ada untuk membuat tupel baru seperti ditunjukkan oleh contoh berikut.

```

1 tup1 = (12, 34.56)
2 tup2 = ('abc', 'xyz')
3 # Aksi seperti dibawah ini tidak bisa dilakukan pada tuple python
4 # Karena memang nilai pada tuple python tidak bisa diubah
5 # tup1[0] = 100;
6 # Jadi, buatlah tuple baru sebagai berikut
7 tup3 = tup1 + tup2
8 print (tup3)

```

### 1.24 Menghapus Nilai Dalam Tuple

Menghapus elemen tuple individual tidak mungkin dilakukan. Tentu saja, tidak ada yang salah dengan menggabungkan tupel lain dengan unsur-unsur yang tidak diinginkan dibuang.

Untuk secara eksplisit menghapus keseluruhan tuple, cukup gunakan del statement. Sebagai contoh

```

1 tup = ('fisika', 'kimia', 1993, 2017);
2 print (tup)
3 del tup;
4 print "Setelah menghapus tuple : "
5 print tup

```

## 1.25 Operasi Dasar Pada List Tuple

Tupel merespons operator + dan \* sama seperti String; Mereka berarti penggabungan dan pengulangan di sini juga berlaku, kecuali hasilnya adalah tupel baru, bukan string.

Sebenarnya, list merespons semua operasi urutan umum yang kami gunakan pada String di bab sebelumnya. Dibawah ini adalah tabel daftar operasi dasar pada list python.

**Tabel 1.25** daftar operasi dasar pada python

Python Expression	Hasil	Penjelasan
<code>len((1, 2, 3))</code>	3	Length
<code>(1, 2, 3) + (4, 5, 6)</code>	<code>(1, 2, 3, 4, 5, 6)</code>	Concatenation
<code>('Halo!',) * 4</code>	<code>('Halo!', 'Halo!', 'Halo!', 'Halo!')</code>	Repetition
<code>3 in (1, 2, 3)</code>	True	Membership
<code>for x in (1,2,3) : print (x, end = ' ')</code>	1 2 3	Iteration

## 1.26 Indexing, Slicing dan Matrix

Karena tupel adalah urutan, pengindeksan dan pengiris bekerja dengan cara yang sama untuk tupel seperti pada String, dengan asumsi masukan berikut

Dengan asumsi input berikut :

`T = ('C++', 'Java', 'Python')`

**Tabel 1.26** Indexing, Slicing dan Matrix

Python Expression	Hasil	Penjelasan
<code>T[2]</code>	<code>'Python'</code>	Offset mulai dari nol
<code>T[-2]</code>	<code>'Java'</code>	Negatif: hitung dari kanan
<code>T[1:]</code>	<code>('Java', 'Python')</code>	Slicing mengambil bagian

## 1.27 Fungsi Build-in

Python menyertakan fungsi built-in sebagai berikut

**Tabel 1.27** fungsi built-in

Python Function	Penjelasan
<b>cmp(tuple1, tuple2)</b>	# Tidak lagi tersedia dengan Python 3
<b>len(tuple)</b>	Memberikan total panjang tuple.
<b>max(tuple)</b>	Mengembalikan item dari tuple dengan nilai maks.
<b>min(tuple)</b>	Mengembalikan item dari tuple dengan nilai min.
<b>tuple(seq)</b>	Mengubah tuple menjadi tuple.

## 1.28 Dictionary Python

Dictionary Python berbeda dengan List ataupun Tuple. Karena setiap urutannya berisi key dan value. Setiap key dipisahkan dari value-nya oleh titik dua (:), item dipisahkan oleh koma, dan semuanya tertutup dalam kurung kurawal. Dictionary kosong tanpa barang ditulis hanya dengan dua kurung kurawal, seperti ini: .

Nilai kamus bisa berupa tipe apa pun, namun key harus berupa tipe data yang tidak berubah seperti string, angka, atau tupel.

**Akses Nilai** Untuk mengakses elemen Dictionary, Anda dapat menggunakan tanda kurung siku yang sudah dikenal bersama dengan key untuk mendapatkan nilainya. Berikut adalah contoh sederhananya :

```

1 #Contoh cara membuat Dictionary pada Python
2 dict = {'Name': 'Dava', 'Age': 12, 'Class': 'SMP'}
3 print ("dict['Name']: ", dict['Name'])
4 print ("dict['Age']: ", dict['Age'])
```

**Update Nilai** Anda dapat memperbarui Dictionary dengan menambahkan entri baru atau pasangan nilai kunci, memodifikasi entri yang ada, atau menghapus entri yang ada seperti ditunjukkan pada contoh sederhana yang diberikan di bawah ini.

```

1 #Update dictionary python
2 dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
3 dict['Age'] = 8; # Mengubah entri yang sudah ada
4 dict['School'] = "DPS School" # Menambah entri baru
5 print ("dict['Age']: ", dict['Age'])
6 print ("dict['School']: ", dict['School'])
```

**Hapus Nilai** Anda dapat menghapus elemen Dictionary individual atau menghapus keseluruhan isi Dictionary. Anda juga dapat menghapus seluruh Dictionary dalam satu operasi. Untuk menghapus seluruh Dictionary secara eksplisit, cukup gunakan del statement. Berikut adalah contoh sederhana :

```

1 #Contoh cara menghapus pada Dictionary Python
2 dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
3 del dict['Name'] # hapus entri dengan key 'Name'
4 dict.clear() # hapus semua entri di dict
5 del dict # hapus dictionary yang sudah ada
6 print ("dict['Age']: ", dict['Age'])
7 print ("dict['School']: ", dict['School'])

```

## 1.29 Tanggal dan Jam

Program Python yang dapat menangani tanggal dan waktu dalam beberapa cara. Mengkonversi antara format tanggal adalah tugas umum untuk komputer. Modul Python's waktu dan kalender membantu melacak tanggal dan waktu.

Interval waktu adalah angka floating-point dalam satuan detik. Instants tertentu dalam waktu dinyatakan dalam satu detik sejak 12:00 am, 1 Januari 1970(epoch).

Ini adalah waktu yang populer modul yang tersedia di Python yang menyediakan fungsi untuk bekerja dengan waktu, dan untuk mengkonversi antara pernyataan. Fungsi time.time() mengembalikan sistem saat ini waktu sejak 12:00 am, 1 Januari 1970.

```

1 import time; # harus menginclude modul time
2 ticks = time.time()
3 print "Number of ticks since 12:00am, January 1, 1970:", ticks

```

Bila kode diatas dieksekusi, maka akan menghasilkan hasil sebagai berikut :

**Number of ticks since 12:00am, January 1, 1970: 7186862.73399**

## 1.30 Fungsi

Fungsi adalah blok kode terorganisir dan dapat digunakan kembali yang digunakan untuk melakukan sebuah tindakan/action. Fungsi memberikan modularitas yang lebih baik untuk aplikasi Anda dan tingkat penggunaan kode yang tinggi.

**Mendefinisikan Fungsi Python** Anda dapat menentukan fungsi untuk menyediakan fungsionalitas yang dibutuhkan. Berikut adalah aturan sederhana untuk mendefinisikan fungsi dengan Python.

- Fungsi blok dimulai dengan def kata kunci diikuti oleh nama fungsi dan tanda kurung (()) .
- Setiap parameter masukan atau argumen harus ditempatkan di dalam tanda kurung ini. Anda juga dapat menentukan parameter di dalam tanda kurung ini.
- Pernyataan pertama dari sebuah fungsi dapat berupa pernyataan opsional string dokumentasi fungsi atau docstring.
- Blok kode dalam setiap fungsi dimulai dengan titik dua (:) dan indentasi.

- Pernyataan kembali [ekspresi] keluar dari sebuah fungsi, secara opsional menyampaikan kembali ekspresi ke pemanggil. Pernyataan pengembalian tanpa argumen sama dengan return None.

Contoh fungsi:

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



## BAB 2

---

# BIG DATA

---

### 2.1 Sejarah Big Data

Sejarah Big Data yaitu yang bermula dari sejarahnya Apache Hadoop. Doug Cutting adalah orang yang iseng-iseng membuat search engine. Doug Cutting menemukan beberapa permasalahan karena search engine milik crawler (untuk ngumpulin web page di internet), indexer, storage dan search algo storage nya harus besar dan tidak bisa hanya disimpan di dalam satu node saja karena disetiap saat terdapat kemungkinan network failure, power failure, node failure dan segala macam failure lain nya yang ada di distributed system. Doug Cutting stuck, bagaimana cara nya agar menyimpan data sebesar itu dan juga cara processing nya, dan bagaimana agar efisien dari data yang tersebar itu sampai akhirnya Doug Cutting menemukan solusi dari permasalahan itu karena google mengeluarkan 2 buah paper Google File System dan juga MapReduce. Fenomena Big Data, dimulai pada tahun 2000-an ketika seorang analis industri Doug menyampaikan konsep Big Data yang terdiri dari tiga bagian penting yaitu 3V (Volume, Velocity, Variety).

- 1944 Fremont Rider, Pustakawan universitas Wesleyan. Dia memperkirakan bahwa Perpustakana yang ada di amerika serikat ukurannya meningkat dua kali lipat setiap 16 tahun.
- 1961 Derek Price. Dia mendiagramkan pertumbuhan pengetahuan ilmiah dengan cara melihat jumlah pertumbuhan jurnal ilmiah dan makalah.
- 1967 B.A. Marron dan P.A.D. De Maine menerbitkan Automatic data compression dalam Komunikasi dari ACM, yang menyatakan bahwa ledakan informasi tercatat dalam beberapa tahun terakhir membuatnya penting bahwa persyaratan penyimpanan untuk semua informasi harus dijaga agar tetap minimum.
- 1971 Arthur Miller Menulis dalam The Assault on Privacy menyatakan, Terlalu banyak informasi. pengurus tampaknya mengukur seorang pria seimbang dengan jumlah bit kapasitas penyimpanan berkas itu akan mengisi.
- 1975 Departemen Pos dan Telekomunikasi di Jepang mulai melakukan Arus Informasi Sensus, pelacakan volume informasi yang beredar di Jepang (ide pertama kali diusulkan dalam makalah 1969).
- 1980 I.A. Tjomsland memberikan ceramah berjudul Where do we go from here? Di IEEE Keempat Symposium on Mass Storage Systems, dia mengatakan Mereka yang terkait dengan perangkat penyimpanan lama menyadari bahwa Hukum parkinson Pertama dapat diparafrasekan untuk menggambarkan Industry kami Data mengembang untuk mengisi ruang yang tersedia.
- 1981 Kantor Pusat statistic hungaria memulai proyek penelitian untuk menjelaskan informasi indsutri negara, termasuk mengukur Volume informasi dalam bit.
- 1983 Ithiel de Sola Pool menerbitkan Pelacakan Arus Informasi di Science. Melihat tren pertumbuhan di 17 Media komunikasi utama 1960-1977, ia menyimpulkan bahwa kata-kata yang tersedia untuk Amerika (di atas usia 10) melalui media ini tumbuh pada tingkat 8,9 persen per tahun, kata-kata benar-benar hadir untuk dari media tersebut tumbuh hanya 2,9 persen per tahun, Pada periode pengamatan, sebagian besar pertumbuhan arus informasi adalah karena pertumbuhan penyiaran, Tapi menjelang akhir periode [1977] situasi berubah: media point-to-point yang tumbuh lebih cepat dari penyiaran.

## 2.2 Gambaran Big Data

Berikut merupakan pengertian big data diruji dari pendapat para ahli,

- Menurut (Eaton, Dirk, Tom, George, & Paul) Big Data merupakan istilah yang berlaku untuk informasi yang tidak dapat diproses atau dianalisis menggunakan alat tradisional.

- Menurut (Dumbill, 2012) , Big Data adalah data yang melebihi proses kapasitas dari kovensi sistem database yang ada. Data terlalu besar dan terlalu cepat atau tidak sesuai dengan struktur arsitektur database yang ada. Untuk mendapatkan nilai dari data, maka harus memilih jalan altenatif untuk memprosesnya.

Dipandang dari sudut pandang ilmu pengetahuan, media penyimpanan yang terdapat pada hardware yang dapat digunakan adalah HDD, FDD, dan juga yang sejenisnya. Sedangkan media penyimpanan pada jaringan biologi, pada diri kita dikaruniai otak oleh Sang Pencipta. Seberapa penting mengolah data-data yang kecil kemudian berkumpul menjadi data yang besar atau bisa disebut dengan Big Data tersebut.

# BIG DATA



Gambar 2.1 Big Data

Terdapat beberapa elemen penting dalam big data diantaranya:

1. Data (Facts, a description of the World)
2. Information (Captured Data and Knowledge): Merekam atau mengambil Data dan Knowledge pada satu waktu tertentu (at a single point). Sedangkan Data dan Knowledge dapat terus berubah dan bertambah dari waktu ke waktu.
3. Knowledge (Our personal map/model of the world): apa yang kita ketahui (not the real world itself) Anda saat ini tidak dapat menyimpan pengetahuan dalam diri anda dalam apa pun selain otak, dan untuk membangun pengetahuan perlu informasi dan data.

Menurut McKinsey Global (2011), Big Data bisa didefinisikan dengan data yang memiliki skala (volume), distribusi (velocity), keragaman (variety) yang sangatlah besar, dan atau abadi, sehingga sangat membutuhkan penggunaan arsitektur teknikal

dan metode analitik yang inovatif untuk mendapatkan wawasan yang dapat memberikan nilai bisnis baru (informasi yang bermakna). Dan pada pengembangannya ada yang menyebut (7V) termasuk Volume, Velocity, Variety, Variability, Veracity, Value, dan Visualization, atau 10V bahkan lebih dari itu.



**Gambar 2.2** 10V Karakteristik Big Data

Big data adalah sebuah istilah dari sekumpulan data yang begitu besar atau kompleks dimana tidak bisa ditangani lagi dengan menggunakan sistem teknologi komputer konvensional.

### 2.3 Karakteristik Big Data

#### 1. Volume

Volume data yang akan terus meningkat dari waktu ke waktu. Serta banyak faktor yang mendukung meningkatnya volume data secara sangat pesat, diantaranya yaitu hampir semua transaksi bisnis melibatkan data, meningkatnya jumlah unstructured data yang mengalir dari media sosial, dan meningkatnya jumlah data yang dihasilkan dari mesin serta perangkat dari mobile.

- Facebook menghasilkan 10TB data baru setiap hari, Twitter 7TB
- Sebuah Boeing 737 menghasilkan 240 terabyte data penerbangan selama penerbangan dari satu wilayah bagian AS ke wilayah yang lain
- Microsoft kini memiliki satu juta server, kurang dari Google, tetapi lebih dari Amazon, kata Ballmer (2013).

Big Data mengarah pada managemen informasi skala yang besar (large-scale) dan teknologi analisis yang melebihi kapabilitas teknologi data secara tradisional. Ada perbedaan yang paling utama dalam tiga hal antara Tradisional dengan Big Data, yaitu the rate of data generation, amount of data (volume) and transmission (velocity), dan the types of structured and unstructured data (variety).



Gambar 2.3 Volume

Teknologi pada Big Data dibagi menjadi 2 kelompok, yaitu batch processing yang mana digunakan untuk menganalisis data yang sudah settle (data at rest) pada satu waktu tertentu. Dan streaming processing yang mana digunakan untuk menganalisis data yang terus menerus terupdate setiap waktu (data in motion).

## 2. Velocity

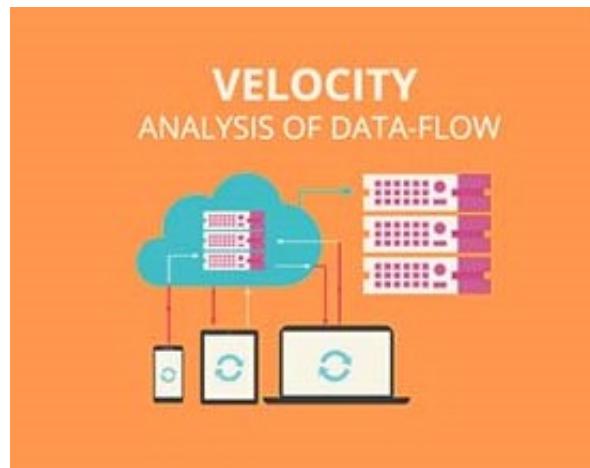
Velocity merupakan kecepatan data yang masuk baik dalam per jam, per detik, dan lainnya. Clickstreams (web log) dan transfer data asynchronous yang dapat menangkap apa saja yang dilakukan oleh jutaan atau lebih pengguna yang lakukan saat ini. Dimana clickstream atau web log merupakan salah satu sumber data yang menarik. Sebuah clickstream meliputi suatu rekaman untuk setiap permintaan halaman dari setiap pengunjung dari website. Oleh karena itu, suatu clickstream merekam setiap gesture yang dibuat oleh pengunjung dan gesture memiliki potensi untuk memberikan deskripsi mengenai kebiasaan dari pengunjung yang bersangkutan. Diharapkan bahwa clickstream akan mengidentifikasi sesi yang berhasil dan tidak berhasil, menentukan apakah pengunjung puas atau tidak puas, dan menemukan bagian dari website yang secara efektif menarik perhatian pengunjung.



Gambar 2.4 Velocity

### 3. Variety

Variety adalah sekumpulan dari berbagai macam-macam data, baik data yang terstruktur, semi terstruktur maupun data tidak terstruktur (bisa dipastikan lebih mendominasi). Tampilan data semakin komprehensif (lengkap dan menyeluruh).



Gambar 2.5 Variety

Saat ini data memiliki banyak format dan tipe. Data tersebut ada yang berupa structured data dan unstructured data. Informasi dihasilkan dari aplikasi bisnis yang digunakan oleh organisasi dan melibatkan structured data ataupun unstructured data. Mengelola, menggabungkan dan mengatur data yang memiliki beragam format dan tipe merupakan suatu tantangan yang harus diselesaikan

dan dijawab oleh suatu organisasi. Selain itu untuk menghasilkan pengetahuan dari data yang berjumlah besar tersebut perlu untuk menghubungkan semua data dari beragam tipe dan format.

#### 4. Value

Value atau Nilai Data. Big data mempunyai nilai data yang rendah terhadap kuantitasnya. Data engineering akan melakukan penataan data mentah sehingga data tersebut lebih bernilai. Kemudian data yang sudah ditata, dapat dilakukan analisis oleh data scientist.

#### 5. Veracity

Veracity atau Kejujuran. Karakter veracity mengarah kepada seberapa akurat dan dapat dipercaya suatu data. Melanjutkan satu contoh di poin value di atas, bisa jadi di dalam file MP3 tersebut IDv1 tag-nya sudah dimodif sehingga keaslian file MP3 tersebut dipertanyakan, perubahan IDv1 tag tersebut bisa jadi karena hasil output aplikasi pengolah suara atau converter file MP3. Data yang tidak memiliki karakter kejujuran atau keaslian tidak akan tersaring ke dalam sistem analisis.



**Gambar 2.6** Veracity

#### 6. Variability

Arus data dalam periode tertentu kadang tidak konsisten sehingga variabilitas menjadi salah satu bagian penting. Contohnya ada pada media sosial. Di media sosial pasti akan ada tren tertentu yang muncul dadakan. Tren tersebut periodenya berbeda-beda, bisa harian, mingguan, maupun bulanan. Beban puncak data yang seperti itu sangat memerlukan analisis big data.

#### 7. Validity

Data yang diambil adalah harus benar dan akurat sesuai dengan yang ingin di gunakan. Terdapat frasa yang cukup populer di Big Data yaitu Garbage In

Garbage Out, yang berarti apabila banyak data yang sampah, maka hasil akhir akan berupa sampah juga. Data yang valid merupakan salah satu kunci di dalam pengambilan keputusan yang tepat.

#### 8. Vulnerability

Bagaimanapun, pelanggaran data dengan data besar adalah pelanggaran besar. Sayangnya ada banyak pelanggaran data besar. Contoh lain, seperti yang dilaporkan oleh CRN: pada bulan Mei 2016 "seorang hacker bernama Peace memposting data di web gelap untuk dijual, yang diduga termasuk informasi tentang 167 juta akun LinkedIn dan ... 360 juta email dan kata sandi untuk pengguna MySpace."

#### 9. Volatility

How old does your data need to be before it is considered irrelevant, historic, or not useful any longer? How long does data need to be kept for?

Before big data, organizations tended to store data indefinitely – a few terabytes of data might not create high storage expenses; it could even be kept in the live database without causing performance issues. In a classical data setting, there might not even be data archival policies in place.

Due to the velocity and volume of big data, however, its volatility needs to be carefully considered. You now need to establish rules for data currency and availability as well as ensure rapid retrieval of information when required. Make sure these are clearly tied to your business needs and processes – with big data the costs and complexity of a storage and retrieval process are magnified.

#### 10. Visualization

Another characteristic of big data is how challenging it is to visualize.

Current big data visualization tools face technical challenges due to limitations of in-memory technology and poor scalability, functionality, and response time. You can't rely on traditional graphs when trying to plot a billion data points, so you need different ways of representing data such as data clustering or using tree maps, sunbursts, parallel coordinates, circular network diagrams, or cone trees.

Combine this with the multitude of variables resulting from big data's variety and velocity and the complex relationships between them, and you can see that developing a meaningful visualization is not easy.

### 2.4 Ekosistem Big Data Analytics

Keterangan:

1. Data Devices
2. Data Collectors



Gambar 2.7 Ekosistem Big Data

3. Data Aggregators: penggabungan beberapa informasi dari database dengan memiliki tujuan untuk mempersiapkan dataset gabungan untuk pengolahan data.
4. Data Users/ Buyers

Sebuah titik awal untuk memahami Analytics adalah cara untuk mengeksplorasi atau menyelidiki atau juga memahami secara mendalam suatu objek sampai ke akarnya. Hasil analytics biasanya tidak menyebabkan banyak kebingungan, karena konteksnya biasanya membuat makna yang jelas. Perkembangan analytics dimulai dari DSS kemudian berkembang menjadi Bussines Intelligence baru kemudian menjadi analytics yang ditunjukkan oleh Gambar 2.8 berikut:

Bussines Intelligence bisa dilihat sebagai suatu istilah umum untuk semua aplikasi yang mendukung DSS, dan bagaimana hal itu dijelaskan dalam industri dan semakin meluas sampai di kalangan akademisi. Bussines Intelligence berevolusi dari DSS, dan orang dapat berargumentasi bahwa Analytics berevolusi dari Bussines Intelligence. Dengan demikian, Analytics merupakan istilah umum untuk aplikasi analisis data.

Big Data Analytics adalah Alat dan teknik analisis yang akan sangat membantu dalam memahami big data dengan syarat algoritma yang menjadi bagian dari alat-



**Gambar 2.8** Perkembangan Analytics

alat ini harus mampu bekerja dengan jumlah besar pada kondisi real-time dan pada data yang berbeda-beda.

Bidang Pekerjaan baru Big Data Analytics:

1. Data Savvy Professionals: Seseorang yang tahu bagaimana untuk berpikir tentang data, bagaimana mengajukan jenis pertanyaan yang tepat sesuai dengan kebutuhan lembaga atau perusahaan atau lainnya dan mampu memahami dan mengklarifikasi jawaban hasil analisis yang mereka terima.
2. Technology and data enablers: Seseorang dapat memberikan dukungan integrasi antara data dengan teknologi yang sesuai, dan yang paling berkembang saat ini.
3. Data scientists (Memiliki bakat analitik yang mendalam/ Ilmuwan Data): orang-orang yang memiliki latar belakang yang kuat dalam algoritma-algoritma sistem cerdas, atau matematika terapan, atau ekonomi, atau ilmu pengetahuan lainnya melalui inferensi data dan eksplorasi.

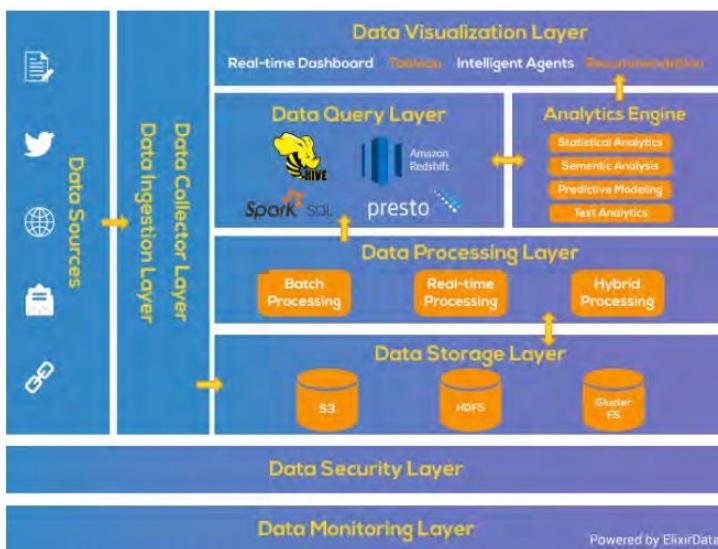
## 2.5 Arsitektur Big Data

Cara Terbaik untuk mendapatkan solusi dari Permasalahan Big Data (Big Data Solution) adalah dengan "Membagi Masalahnya". Big Data Solution dapat dipahami dengan baik menggunakan Layered Architecture. Arsitektur Layered dibagi ke dalam Lapisan yang berbeda dimana setiap lapisan memiliki spesifikasi dalam melakukan fungsi tertentu. Arsitektur tersebut membantu dalam merancang Data Pipeline (jalur data) dengan berbagai mode, baik Batch Processing System atau Stream Processing System. Arsitektur ini terdiri dari 6 lapisan yang menjamin arus data yang optimal dan aman.

Data Ingestion Layer - Lapisan ini merupakan langkah awal untuk data yang berasal dari sumber tertentu dan akan memulai perjalannya. Data disini akan dilakukan pemrioritasan dan pengkategorian, sehingga data dapat diproses dengan mudah diteruskan ke lapisan lebih lanjut. Tool yang dapat digunakan, yaitu Apache Flume, Apache Nifi (Pengumpulan dan Penggalian Data dari Twitter menggunakan

Apache NiFi untuk Membangun Data Lake), Elastic Logstash. Data masuk ke dalam Data Lake dalam bentuk mentah, dan semua datanya disimpan, tidak hanya data yang digunakan saja, tapi juga data yang mungkin digunakan di masa depan. Di Data Lake semua data tersimpan dalam bentuk aslinya.

Lapisan Kolektor Data (Data Collector Layer) - Di Lapisan ini, lebih fokus pada penyaluran data dari lapisan penyerapan atau pengambilan data awal (ingestion) ke jalur data yang lainnya. Pada Lapisan ini, data akan dipisahkan sesuai dengan kelompoknya atau komponen-komponennya (Topik: kategori yang ditentukan pengguna yang pesannya dipublikasikan, Produser - Produsen yang memposting pesan dalam satu topik atau lebih, Konsumen berlangganan topik dan memproses pesan yang diperlukan, Brokers - Pialang yang tekun dalam mengelola dan replikasi data pesan) sehingga proses analitik bisa dimulai. Tool yang dapat digunakan, yaitu Apache Kafka.



Gambar 2.9 Arsitektur Big Data

Lapisan Pengolahan Data (Data Processing Layer) - fokus utama lapisan ini adalah untuk sistem pemrosesan data pipeline atau dapat kita katakan bahwa data yang telah kita kumpulkan di lapisan sebelumnya akan diproses di lapisan ini. Di sini kita melakukan ekstraksi dan juga learning dari data untuk diarahkan ke tujuan yang bermacam-macam, mengklasifikasikan arus data yang seharusnya diarahkan dan ini adalah titik awal di mana analitik telah dilakukan. Data pipeline merupakan komponen utama dari Integrasi Data. Data pipeline mengalirkan dan mengubah data real-time ke layanan yang memerlukannya, mengotomatiskan pergerakan dan transformasi data, mengolah data yang berjalan di dalam aplikasi Anda, dan mentransformasikan semua data yang masuk ke dalam format standar sehingga bisa digunakan untuk analisis dan visualisasi. Jadi, Data pipeline adalah rangkaian langkah

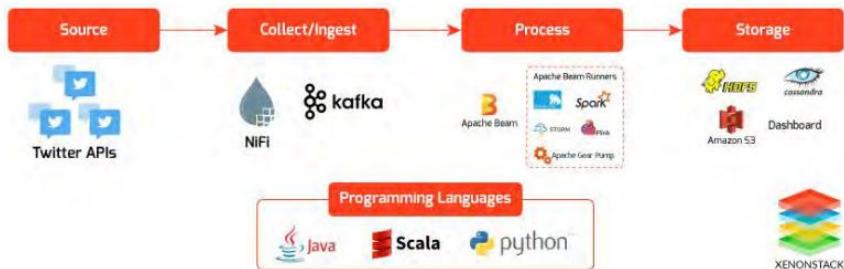
yang ditempuh oleh data Anda. Output dari satu langkah dalam proses menjadi input berikutnya. Langkah-langkah dari Data pipeline dapat mencakup pembersihan, transformasi, penggabungan, pemodelan dan banyak lagi, dalam bentuk kombinasi apapun. Tool yang dapat digunakan, yaitu Apache Sqoop, Apache Storm, Apache Spark, Apache Flink.

Lapisan Penyimpanan Data (Data Storage Layer) - Media penyimpanan menjadi tantangan utama, saat ukuran data yang digunakan menjadi sangat besar. Lapisan ini berfokus pada "tempat menyimpan data yang begitu besar secara efisien". Tool yang dapat digunakan, yaitu Apache Hadoop (HDFS), Gluster file systems (GFS), Amazon S3.

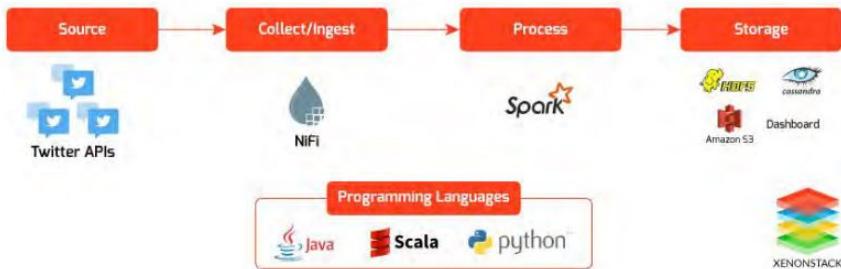
Lapisan Query Data (Data Query Layer) - lapisan ini merupakan tempat berlangsungnya pemrosesan secara analitik yang sedang dalam keadaan aktif. Di sini, fokus utamanya adalah mengumpulkan data value sehingga dapat dibuat lebih bermanfaat dan mudah digunakan untuk lapisan berikutnya. Tool yang dapat digunakan, yaitu Apache Hive, Apache (Spark SQL), Amazon Redshift, Presto.

Lapisan Visualisasi Data (Data Visualization Layer) - Proses Visualisasi, atau tahapan merepresentasikan data dalam bentuk visual, kemungkinan ini adalah tingkat yang paling bergengsi, di mana pengguna data pipeline dapat merasakan hasil laporan yang mendetail dan mudah dipahami dari data value yang telah divisualisasikan. Kita membutuhkan sesuatu yang akan menarik perhatian orang dari visualisasi data, sehingga membuat temuan Anda mudah dipahami dengan baik oleh mereka melalui visualisasi tersebut. Tool yang dapat digunakan, yaitu Tableau, Kibana sebagai Real-Time Dashboards, Angular.js sebagai Intelligence Agents misalnya agen dapat mengingat hal-hal yang mungkin Anda sudah lupa, dengan cerdas meringkas data yang kompleks, belajar dari Anda dan bahkan membuat rekomendasi untuk Anda, menemukan dan memfilter informasi saat Anda melihat data perusahaan atau berse-lancar di Internet dan tidak tahu di mana informasi yang tepat, React.js sebagai sistem recommender untuk memprediksi tentang kriteria pengguna, yaitu menentukan model penggunanya seperti apa.

Apache Spark digunakan secara luas untuk pengolahan Big Data. Spark bisa mengolah data di kedua mode yaitu Pengolahan Batch Mode dan Streaming Mode. Apache NiFi ke Apache Spark melakukan transmisi data menggunakan komunikasi situs ke situs. Dan output port-nya digunakan untuk mempublikasikan data dari sum-



**Gambar 2.10** Data Integration Using Apache NiFi dan Apache Kafka



**Gambar 2.11** Integrating Apache Spark and NiFi for Data Lakes

bernya (source). Apache Spark adalah mesin pemrosesan data dalam memori, yang cepat dan ringkas dengan mode pengembangan API yang elegan dan ekspresif, yang memungkinkan pengguna melakukan proses secara streaming, menggunakan pembelajaran mesin (machine learning), atau SQL yang memerlukan akses berulang-ulang secara cepat terhadap kumpulan data. Dengan Spark yang berjalan di Apache Hadoop YARN, developer sekarang dapat membuat aplikasi dengan memanfaatkan kehandalan dari Spark, untuk memperoleh wawasan, dan memperkaya data sains mereka dengan memasukkan data dalam satu kumpulan data besar di Hadoop.

## 2.6 Key Roles Kunci Sukses Proyek Analitik

### 1. Bussines User

**Business User:** Seseorang yang memahami wilayah domain (kondisi existing) dan dapat mengambil manfaat besar dari hasil proyek analitik, dengan cara konsultasi dan menyarankan tim proyek pada scope proyek, hasil, dan operasional output (terkait dengan cara mengukur suatu variabel). Biasanya yang memenuhi peran ini adalah analis bisnis, manajer lini, atau ahli dalam hal pokok yang mendalam.

### 2. Project Sponsor

**Project Sponsor:** Bertanggung jawab terkait asal proyek. Memberi dorongan, persyaratan proyek dan mendefinisikan masalah core bisnis. Umumnya menyediakan dana dan konsep pengukur tingkat nilai output akhir dari tim kerja. Menetapkan prioritas proyek dan menjelaskan output yang diinginkan.

### 3. Project Manager

**Project Manager:** Memastikan bahwa pencapaian utama projek dan tujuan terpenuhi tepat waktu dan sesuai dengan kualitas yang diharapkan.

### 4. Business Intelligence Analyst

**Business Intelligence Analyst:** Menyediakan keahlian dalam domain bisnis berdasarkan pemahaman yang mendalam mengenai data, indikator kinerja utama (KPI), metrik kunci, dan intelijen bisnis dari perspektif pelaporan. Analis Business Intelligence umumnya membuat dashboard (panel kontrol) dan laporan dan memiliki pengetahuan tentang sumber data dan mekanismenya.

### 5. Database Administrator (DBA)

**Database Administrator (DBA):** Set up dan mengkonfigurasi database untuk mendukung kebutuhan analitik. Tanggung jawab ini mungkin termasuk menye-

diakan akses ke database keys atau tabel dan memastikan tingkat keamanan yang sesuai berada di tempat yang berkaitan dengan penyimpanan data.

## 6. Data Engineer

Data Engineer: Memiliki keterampilan teknis yang mendalam untuk membantu penyetelan query SQL untuk pengelolaan data dan ekstraksi data, dan mendukung untuk konsumsi data ke dalam sandbox analitik. Data Engineer melakukan ekstraksi data aktual dan melakukan manipulasi data yang cukup besar untuk memfasilitasi kebutuhan proyek analitik. Insinyur data (Data Engineer) bekerja sama dengan ilmuwan data (Data Scientist) untuk membantu membentuk data yang sesuai dengan cara yang tepat untuk analisis .

## 7. Data Scientist

Data Scientist (Ilmuwan Data): Menyediakan keahlian untuk teknik analitis, pemodelan data, dan menerapkan teknik analitis yang valid untuk masalah yang diberikan. Memastikan melalui keseluruhan analitik tujuannya dapat terpenuhi. Merancang dan mengeksekusi metode analitis dan melakukan pendekatan lainnya dengan data yang tersedia untuk proyek tersebut.

## 2.7 Konsep Pengolahan Big Data

Bayangkan ada sekumpulan data yang sangat besar (Big Data), bagaimana kita dapat melakukan Parallel atau Distributed Processing.

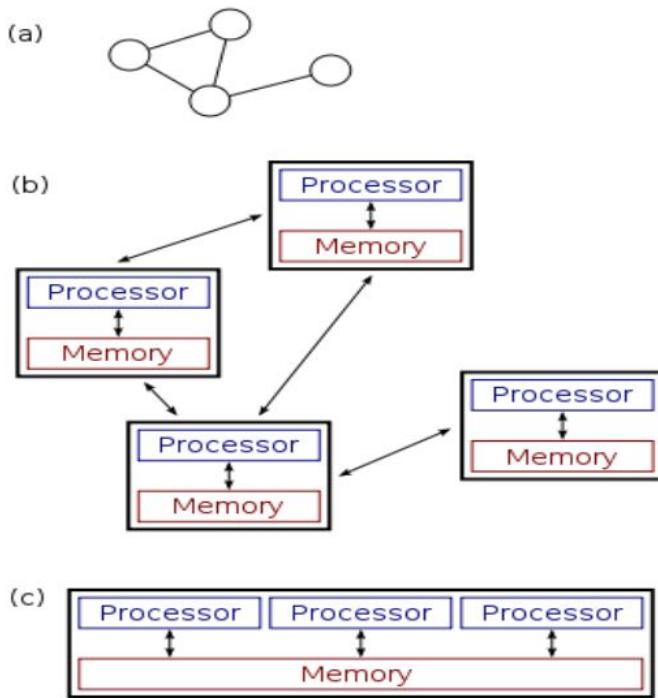
File Sistem Terdistribusi (Distributed File System, atau disingkat dengan DFS) adalah file sistem yang mendukung sharing files dan resources dalam bentuk penyimpanan persistent (tetap) untuk tujuan tertentu di sebuah network.

## 2.8 Introduction to Hadoop

Hadoop: Suatu software framework (kerangka kerja perangkat lunak) open source berbasis Java di bawah lisensi Apache untuk aplikasi komputasi data besar secara intensif.

Hadoop File System dikembangkan menggunakan desain sistem file yang terdistribusi. Tidak seperti sistem terdistribusi, HDFS sangat faulttolerant dan dirancang menggunakan hardware low-cost. Atau dalam arti lain, Hadoop adalah Software platform (platform perangkat lunak) sebagai analytic engine yang memungkinkan seseorang dengan mudah untuk melakukan pembuatan penulisan perintah (write) dan menjalankan (run) aplikasi yang memproses data dalam jumlah besar, dan di dalamnya terdiri dari: HDFS Hadoop Distributed File System dan MapReduce offline computing engine.

Dalam komputasi, platform menggambarkan semacam (hardware architecture) arsitektur perangkat keras atau (software framework) kerangka kerja perangkat lunak (termasuk kerangka kerja aplikasi), yang memungkinkan perangkat lunak dapat berjalan.



**Gambar 2.12** Distributed System (a) dan Paralel System (b)

Ciri khas dari platform meliputi arsitekturnya komputer, sistem operasi, bahasa pemrograman dan runtime libraries atau GUI yang terkait.

## 2.9 Hadoop Distributed File System (HDFS)

Hadoop terdiri dari HDFS (Hadoop Distributed file System) dan Map Reduce. HDFS sebagai direktori di komputer dimana data hadoop disimpan. Untuk pertama kalinya, direktori ini akan di format agar dapat bekerja sesuai spesifikasi dari Hadoop. HDFS sebagai file system, tidak sejajar dengan jenis file system dari OS seperti NTFS, FAT32. HDFS ini menumpang diatas file system milik OS baik Linux, Mac atau Windows.

Data di Hadoop disimpan dalam cluster. Cluster biasanya terdiri dari banyak node atau komputer/server. Setiap node di dalam cluster ini harus terinstall Hadoop untuk bisa jalan.

Hadoop versi 1.x ada beberapa jenis node di dalam cluster:

- Name Node: Ini adalah node utama yang mengatur penempatan data di cluster, menerima job dan program untuk melakukan pengolahan dan analisis data misal melalui Map Reduce. Name Node menyimpan metadata tempat data di cluster dan juga replikasi data tersebut.
- Data Node: Ini adalah node tempat data ditempatkan. Satu block di HDFS/data node adalah 64 MB. Jadi sebaiknya data yang disimpan di HDFS ukurannya minimal 64 MB untuk memaksimalkan kapasitas penyimpanan di HDFS.
- Secondary Name Node: Bertugas untuk menyimpan informasi penyimpanan data dan pengolahan data yang ada di name node. Fungsinya jika name node mati dan diganti dengan name node baru maka name node baru bisa langsung bekerja dengan mengambil data dari secondary name node.
- Checkpoint Node dan Backup Node: Checkpoint node melakukan pengecekan setiap interval waktu tertentu dan mengambil data dari name node. Dengan check point node maka semua operasi perubahan pada data terekam. Namun, secondary name node hanya perlu menyimpan check point terakhir. Backup Node juga berfungsi sama, hanya bedanya data perubahan yang disimpan dilakukan di memory bukan di file seperti checkpoint dan secondary node.

Kelemahan HDFS di hadoop versi 1.x adalah jika name node mati. Maka seluruh cluster tidak bisa digunakan sampai name node baru dipasang di cluster.

Hadoop versi 2.x ada beberapa jenis node di dalam cluster:

- Lebih dari satu name nodes. Hal ini berfungsi sebagai implementasi dari High Availability. Hanya ada satu name node yang berjalan di cluster (aktif) sedangkan yang lain dalam kondisi pasif. Jika name node yang aktif mati/rusak, maka name node yang pasif langsung menjadi aktif dan mengambil alih tugas sebagai name node.

- Secondary name node, checkpoint node dan backup node tidak lagi diperlukan. Meskipun ketiga jenis node tersebut menjadi optional, tetapi kebanyakan tidak lagi ada di cluster yang memakai hadoop versi 2.x. Hal ini karena selain fungsi yang redundant, juga lebih baik mengalokasikan node untuk membuat tambahan name node sehingga tingkat High Availability lebih tinggi.
- Data node tidak ada perubahan yang signifikan di versi hadoop 2.x dari versi sebelumnya.

Meskipun konteks yang kita bicarakan disini adalah dalam cluster, Hadoop juga bisa dijalankan dalam single node. Dalam single node maka semua peran diatas berada dalam satu komputer. Biasanya single node ini digunakan hanya untuk training atau development. Bukan untuk produksi skala enterprise.

## **BAB 3**

---

### ***GOOGLE COLAB***

---

#### **3.1 *Google Colab***

*Google Colab* adalah tools yang dikeluarkan oleh *Google Internal Research*. *Google Colab* merupakan salah satu produk Google berbasis *cloud* yang bisa digunakan secara gratis. *Google Colab* dibuat khusus untuk para researcher atau programmer yang kesulitan untuk mendapatkan akses komputer dengan spek tinggi. *Google Colab* merupakan coding environment bahasa pemrograman Python dengan format *notebook*. Tools pada *Google Colab* menyediakan layanan GPU gratis kepada pengguna sebagai *backend* komputasi dan dapat digunakan selama 12 jam pada suatu waktu. *Google Colab* berjalan diatas cloud milik Google dan menyimpan berkas kedalam Google Drive, serta dapat menjalankan command line langsung pada cell notebook dengan diawali tanda “!”.

### 3.2 Manfaat Menggunakan *Google Colab*

#### 1. Free GPU

*Google Colab* memudahkan untuk menjalankan program pada komputer dengan spek tinggi (GPU Tesla, RAM 12GB, Disk 300GB yang masih bisa disambungkan dengan Google Drive, akses internet cepat untuk download file besar) dan running dalam waktu yang lama.

#### 2. Colaborate

Memudahkan untuk berkolaborasi dengan orang lain dengan cara membagikan kodingan secara online. Dapat lebih mudah bereksperimen secara bersamaan, atau sekadar menggunakan fitur ini untuk mempelajari codingan milik orang lain.

#### 3. Mudah berintegrasi

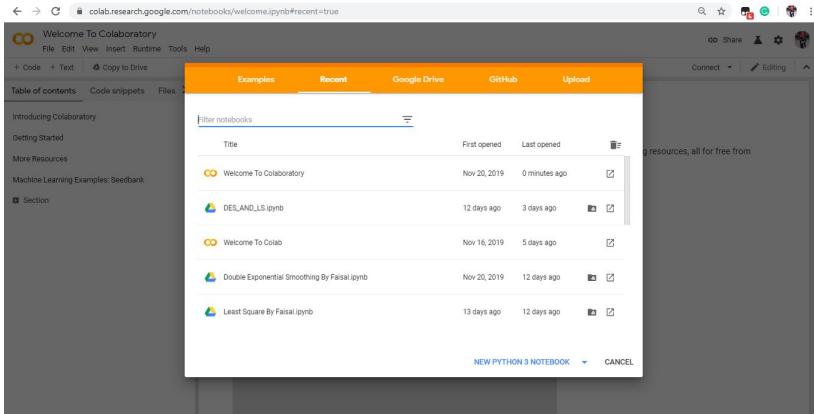
Dapat dengan mudah menghubungkan *Google Colab* dengan jupyter notebook di komputer dengan *local runtime*, menghubungkan dengan *Google Drive*, atau dengan Github.

#### 4. Fleksibel

Bisa dengan mudah merunning *deep learning* program melalui handphone, karena pada *Google Colab* hanya perlu *running* di browser, dapat mengawasi via browser smartphone selama smartphone terhubung dengan *Google Drive* yang sama.

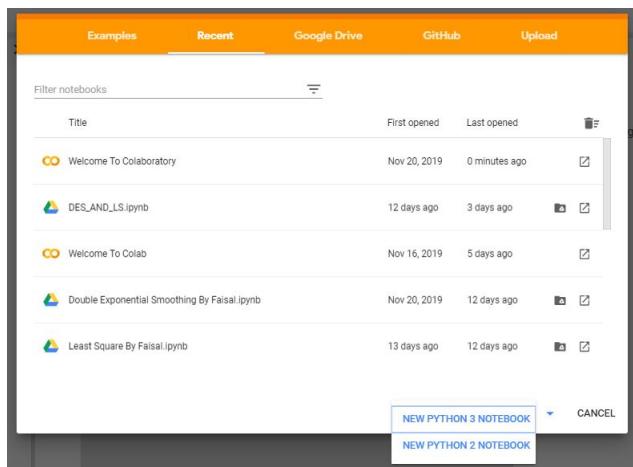
### 3.3 Cara Penggunaan *Google Colab*

1. Sebelumnya butuhkan terlebih dahulu akun Google dan selanjutnya kunjungi ke link <https://colab.research.google.com/>. Setelah itu akan menampilkan tampilan sebagai berikut:



**Gambar 3.1** Tampilan Awal *Google Colab*

2. Untuk membuat notebook baru, cukup klik New Python 3 Notebook atau Python 2 tergantung dari apa yang akan digunakan.



**Gambar 3.2** Tampilan Membuat Notebook Baru

3. Setelah itu akan menampilkan tampilan ke halaman yang mirip dengan Jupyter Notebook. Nantinya, setiap notebook yang kita buat akan disimpan di *Google Drive*.

The screenshot shows the Google Colab interface. At the top, there's a navigation bar with back, forward, and search icons, followed by the URL 'colab.research.google.com/drive/1InA...'. Below the URL is a toolbar with file operations like 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', 'Help', and a 'Comment' button. A status bar at the bottom shows 'RAM' and 'Disk' usage.

The main area contains a code cell with the following content:

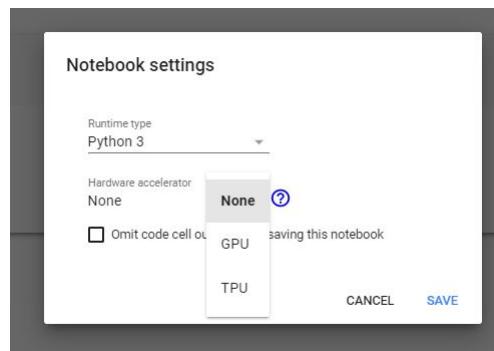
```
[1]: print("Hello Word!")
Hello Word!
[2]: y = 7
for i in range(y):
    print(i)
```

The output of the second cell is a list of numbers from 0 to 6, each on a new line. The code cell itself is collapsed, indicated by a minus sign icon.

**Gambar 3.3** Tampilan *Google Colab*

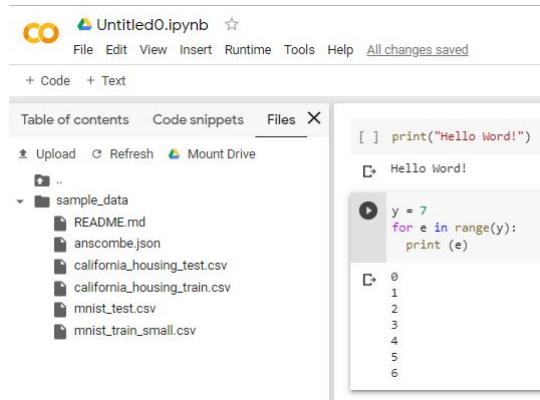
#### 4. Pengaturan GPU

Jika menjalankan program Python menggunakan GPU atau TPU, cukup pilih atau klik Edit > Notebook Settings. Setelah itu pada bagian *Hardware Accelerator* pilih GPU.



**Gambar 3.4** Tampilan Pengaturan *Hardware Accelerator*

5. Selanjutnya mengupload data yang akan diolah di dalam *Google Colab* dengan mengupload data yang berformat csv. Caranya klik pada tulisan Upload, lalu pilih file mana yang akan diupload lalu klik Open.

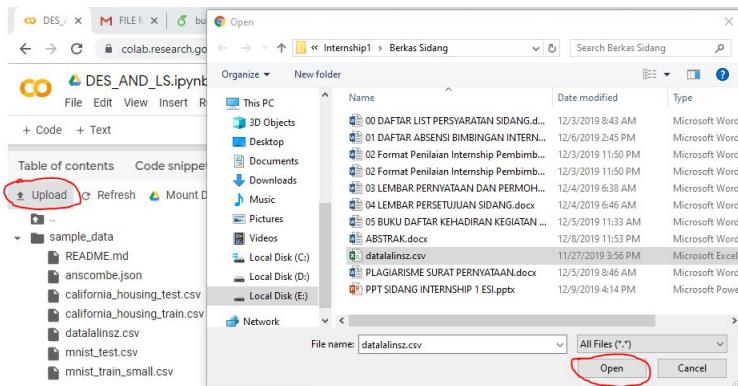


The screenshot shows the Google Colab interface. On the left, there's a sidebar with a 'Files' tab selected. It lists a folder named 'sample\_data' containing several files: README.md, anscombe.json, california\_housing\_test.csv, california\_housing\_train.csv, mnist\_test.csv, and mnist\_train\_small.csv. On the right, there's a code editor window with the following content:

```
[ ] print("Hello Word!")
D: Hello Word!
```

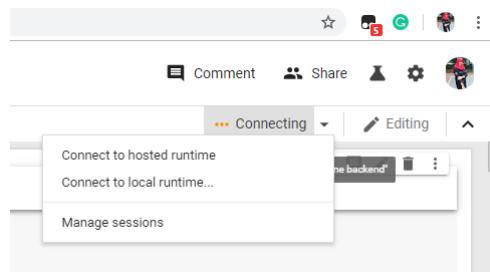
```
[ ] y = 7
for e in range(y):
    print (e)
D: 0
1
2
3
4
5
6
```

**Gambar 3.5** Tampilan File Data csv di Goole Colab

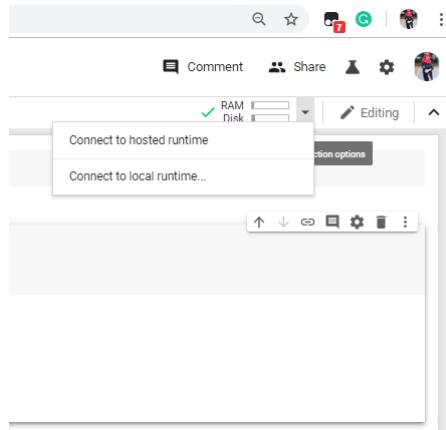


**Gambar 3.6** Tampilan Upload Data csv

6. Ketika kita membuat file baru di *Google Colab* tentu belum langsung terkoneksi ke komputing di google seperti pada gambar 3.7. Lalu setelah itu pilih *Connect to hosted runtime*. Maka setelah itu akan terkoneksi seperti gambar 3.8.

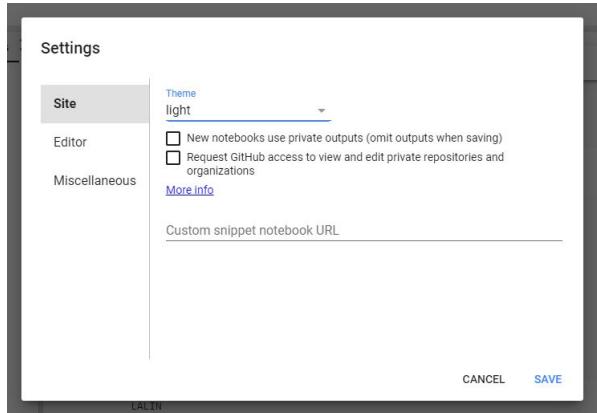


**Gambar 3.7** Tampilan Sebelum Konek

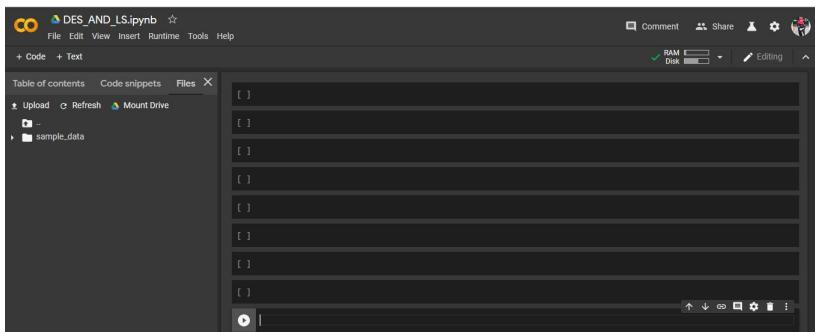


**Gambar 3.8** Tampilan Setelah Konek

7. Tampilan notebook dapat diubah sesuai keinginan. Terdapat pilihan *night mode* yang membuat tema notebook menjadi gelap. Langkah yang dilakukan dengan Tools > Settings > Site.



**Gambar 3.9** Tampilan Pengaturan Notebook



**Gambar 3.10** Tampilan Berhasil Merubah Warna

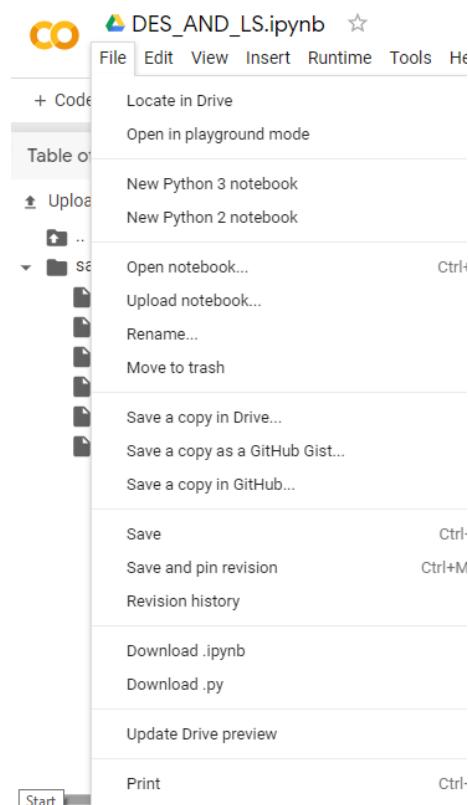
### 3.4 Tools Google Colab

Pada *Google Colab* terdapat beberapa *tools* yang dapat digunakan, berikut di bawah ini akan di jelaskan fungsi-fungsi dari setiap *tools* yang terdapat di *Google Colab*.



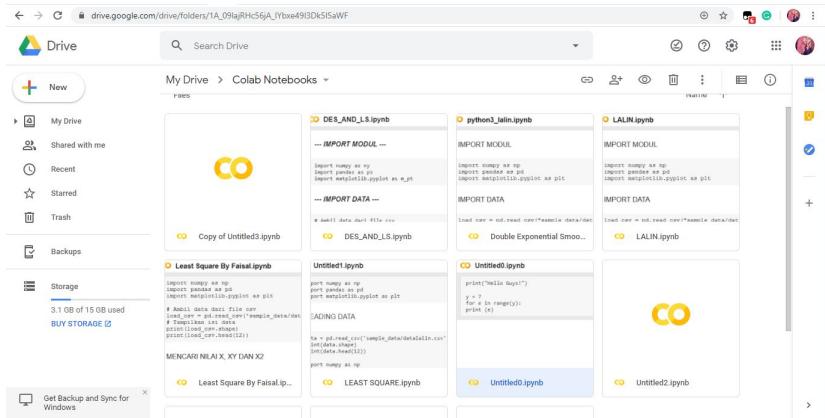
**Gambar 3.11** Tools *Google Colab*

#### 1. File :



**Gambar 3.12** Tools *File*

- *Locate in Drive* : Untuk menemukan file yang sudah tersimpan di dalam *Google Drive*, selanjutnya akan di arahkan menuju tampilan di google drive masing-masing.



Gambar 3.13 *Locate in Drive*

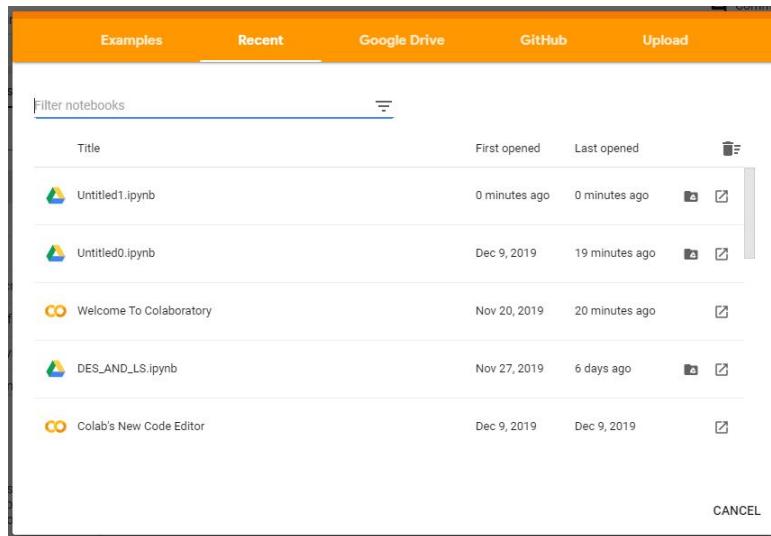
- *Open in playground mode* : Tidak terdapat *output* yang disimpan dalam mode tersebut, tetapi tidak dapat menggunakan fungsi *tools rename* dan *save and pin revision*.



Gambar 3.14 *Open in playground mode*

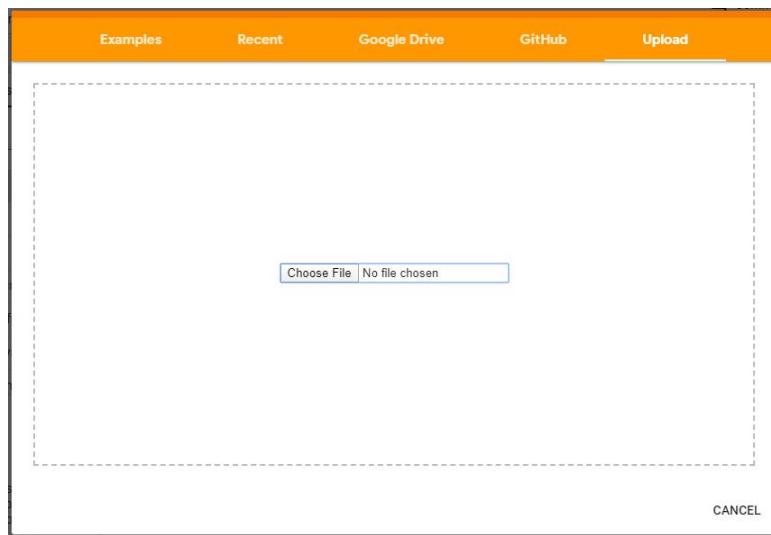
- *New Python 3 notebook* : Membuat file baru di tab baru dengan format Python 3 *notebook*.
- *New Python 2 notebook* : Membuat file baru dengan di tab baru dengan format Python 2 *notebook*.

- *Open notebook* : Untuk membuka file lain yang sudah tersimpan sebelumnya.



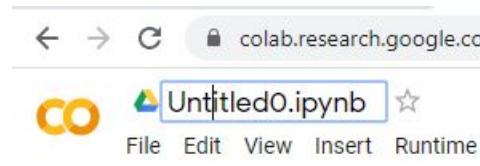
Gambar 3.15 *Open notebook*

- *Upload notebook* : Untuk mengupload file Python yang ada di PC untuk di tampilkan di dalam *Google Colab*.



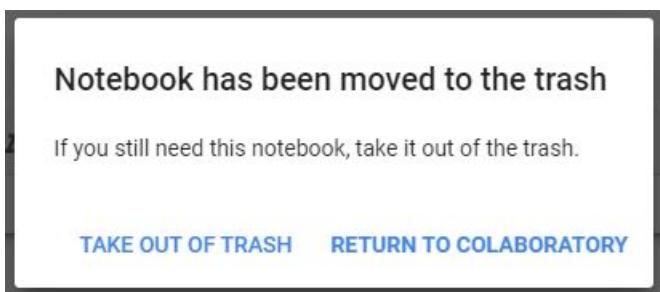
Gambar 3.16 *Upload notebook*

- *Rename* : Untuk mengubah nama dari file Python sesuai yang di inginkan, setelah itu akan diarahkan seperti gambar 3.17.



Gambar 3.17 *Rename*

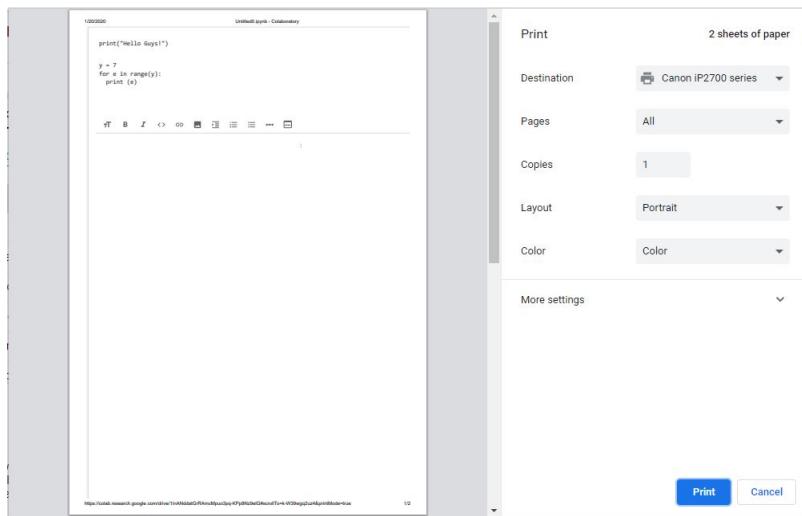
- *Move to trash* : Untuk menghapus file yang sedang dibuka dan terdapat dua pilhan yaitu keluarkan dari tempat sampah atau kembali ke kolaboratori.



Gambar 3.18 *Move to trash*

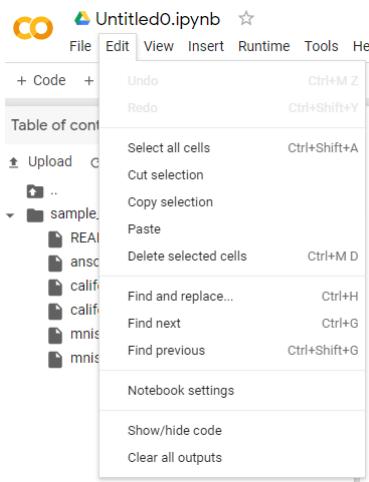
- *Save a copy in Drive* : Untuk menyimpan file *notebook* yang sudah dibuat kedalam *Google Drive*.
- *Save a copy as a GitHub Gist* : Untuk menyimpan file *notebook* yang sudah dibuat kedalam bentuk *Github Gist*.
- *Save a copy in GitHub* : Untuk menyimpan file *notebook* yang sudah dibuat kedalam *GitHub*.
- *Save* : Berfungsi sebagai menyimpan file *notebook* yang sudah dibuat ke dalam *Google Colab*.
- *Save and pin revision* : Menyimpan file yang telah di revisi atau yang telah diubah dengan cara di pin.
- *Revision history* : Melihat data yang sudah di revisi sebelumnya di dalam *Google Colab*.
- *Download .ipynb* : Untuk menyimpan file *notebook* ke dalam PC yang sudah dibuat dalam format *.ipynb*.
- *Download .py* : Untuk menyimpan file *notebook* ke dalam PC yang sudah dibuat dalam format *.py*.

- *Update Drive preview* : Untuk menyimpan file *notebook* yang di simpan di dalam *Google Drive*.
- *Print* : Untuk mencetak hasil keseluruhan yang terdapat di *file notebook Google Colab*.



Gambar 3.19 Print

## 2. Edit :



Gambar 3.20 Tools Edit

- *Undo* : Perintah untuk membatalkan suatu perintah yang sudah dilakukan sebelumnya.
- *Redo* : Cara untuk mengulang sesuatu yang telah dibatalkan sebelumnya.
- *Select all cells* : Berfungsi untuk memblok seluruh sel yang berisikan source code di dalam *notebook Google Colab*.

```
[ ] print("Hello Guys!")

[ ] y = 7
      for e in range(y):
          print (e)

[ ]

Double-click (or enter) to edit
```

**Gambar 3.21** *Select all cells*

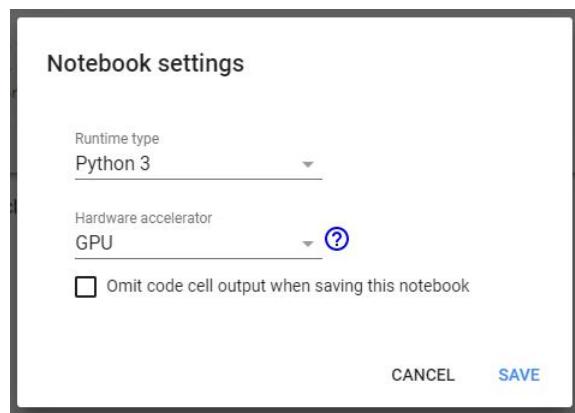
- *Cut selection* : Befungsi untuk mengcut atau memotong *source code* yang sudah di pilih yang ada di dalam *notebook Google Colab*.
- *Copy selection* : Berguna untuk mencopy kata-kata atau *source code* yang sudah terseleksi atau terpilih.
- *Paste* : Yang digunakan untuk menempelkan kata, paragraf, kelompok kata, tabel, gambar, dan object apapun yang sebelumnya telah di Copy atau di Cut.
- *Delete selected cells* : Berfungsi untuk menghapus sel yang telah dipilih.
- *Find and replace* : Untuk mencari kata pada yang di inginkan dan mengganti kata tersebut sesuai dengan keinginan. Akan tampil pada bagian pojok kanan bawah pada google colab.



**Gambar 3.22** *Find and replace*

- *Find text* : Untuk mencari kata pada lembar kerja yang di inginkan dan mengganti kata tersebut sesuai dengan kemauan. Tampilannya sama dengan *Find and replace*.
- *Find previous* : Untuk menemukan kata-kata sebelumnya. Tampilannya sama dengan *Find and replace*.

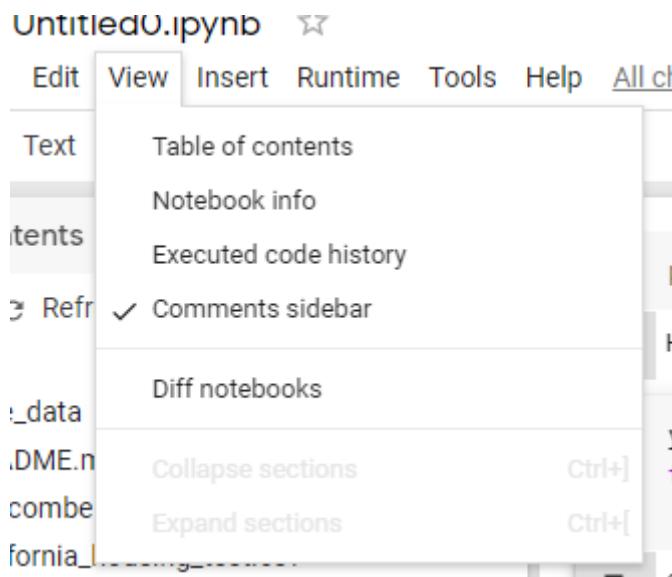
- *Notebook settings* : Untuk mengatur pengaturan yang ada di dalam *notebook* seperti mengatur runtime type, hardware accelerator.



Gambar 3.23 *Notebook settings*

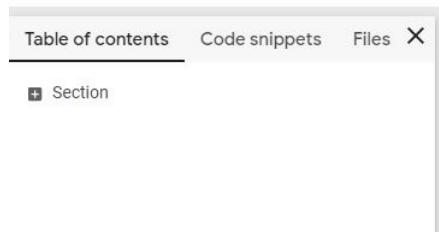
- *Show/hide code* : Berfungsi untuk menampilkan dan menyembunyikan *code*.
- *Clear all outputs* : Berfungsi untuk menghapus semua *output* yang ada.

### 3. View :



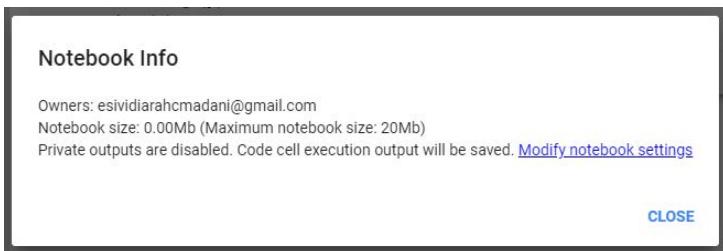
Gambar 3.24 *Tools View*

- *Table of contents* : Salah satu fasilitas yang digunakan untuk pembuatan daftar isi secara otomatis.



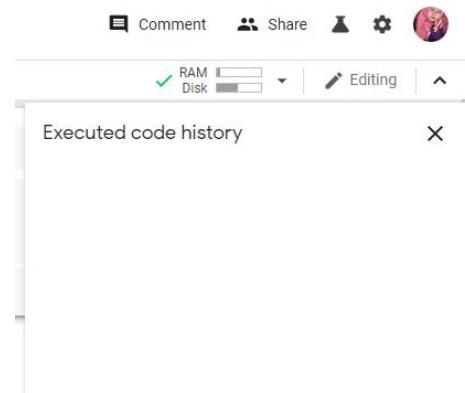
Gambar 3.25 *Table of contents*

- *Notebook info* : Menu yang berisi menampilkan informasi terkait owener, notebook size dan private outputs.



Gambar 3.26 *Notebook info*

- *Executed code history* : Riwayat kode yang dieksekusi.



Gambar 3.27 *Executed code history*

- *Comments sidebar* : Untuk menambahkan komentar di dalam *source code* di sampingnya.
- *Diff notebooks* : Perintah atau menu yang berfungsi untuk menampilkan *notebook* yang baru atau yang berbeda.

The screenshot shows two versions of a notebook, 'Untitled0.ipynb', side-by-side. The left version is the original code, and the right version is the modified code with red highlights indicating changes. The code consists of several code cells and a text cell.

```

Raw source | Inline diff
/drive/1InANddatGrRAmcMpuo3pq-KPp8Nz9elG
Untitled0.ipynb

Code cell <HVEY_-LK3sd>
#% [code]
1 print("Hello Guys!")

Code cell <TNPbeh6aKjq2>
#% [code]
1 y = 7
2 for e in range(y):
3     print(e)

Code cell <8NI87Xwpndr>
#% [code]
1

Text cell <k-W39wgq2uz4>
#% [markdown]
1

```

```

/drive/1InANddatGrRAmcMpuo3pq-KPp8Nz9elG
Untitled0.ipynb

Code cell <HVEY_-LK3sd>
#% [code]
1 print("Hello Guys!")

Code cell <TNPbeh6aKjq2>
#% [code]
1 y = 7
2 for e in range(y):
3     print(e)

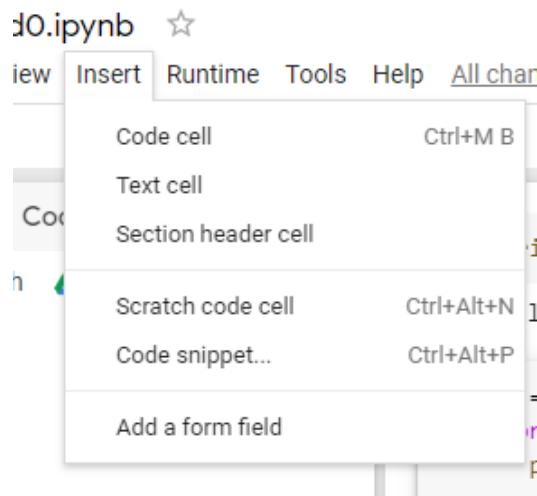
Code cell <8NI87Xwpndr>
#% [code]
1

Text cell <k-W39wgq2uz4>
#% [markdown]
1

```

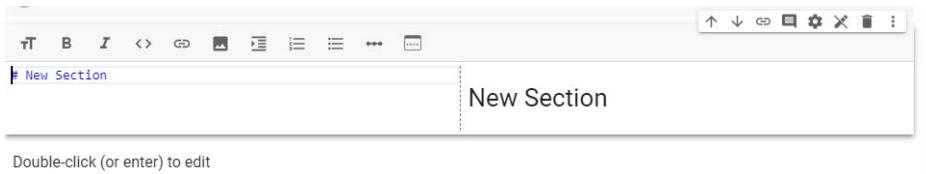
Gambar 3.28 Diff notebooks

#### 4. Insert :



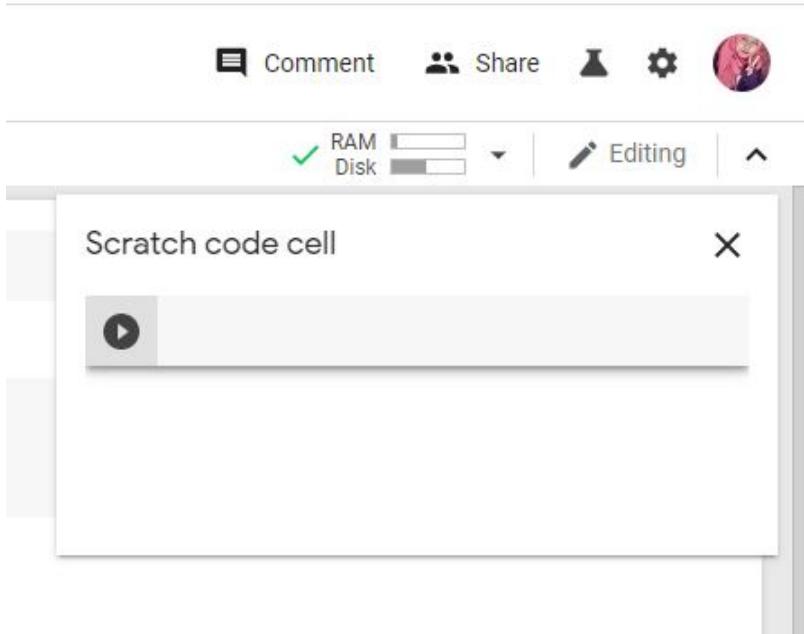
Gambar 3.29 Tools Insert

- *Code cell* : Untuk menambah sel baru untuk membuat baris *source code* baru.
- *Text cell* : Untuk menambahkan sel baru yang berisi hanya *text* biasanya untuk menambahkan keterangan atau judul sebelum *source code*.
- *Section header cell* : Untuk menambahkan *section* baru di dalam *notebook* yang sedang di kerjakan.



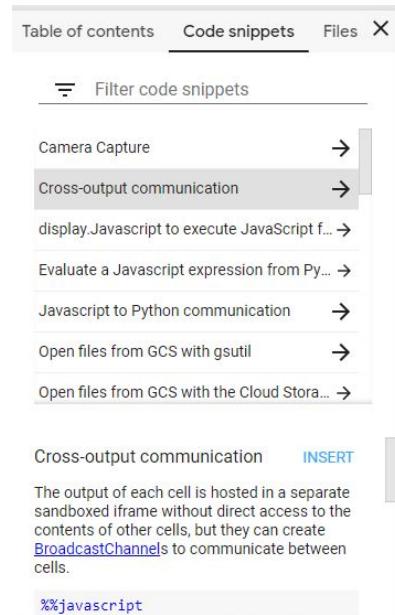
Gambar 3.30 Section header cell

- *Scratch code cell* : untuk mengoreskan kode sel atau membuat dan merunging *script* di dalam *Scratch code cell* seperti pada 3.31.



Gambar 3.31 Scratch code cell

- *Code snippet* : untuk cuplikan kode atau menampilkan kode-kode sesuai dengan yang dicari.



Gambar 3.32 *Code snippet*

- *Add a from field* : untuk menambahkan field baru dengan harus menginputkan form field type, variable name dan variable type.

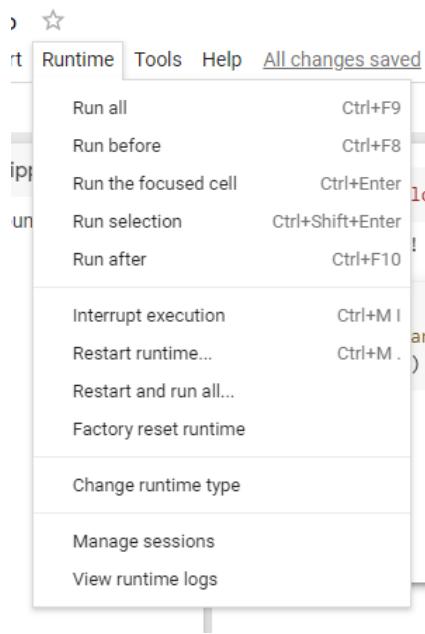
The screenshot shows a modal dialog titled 'Add new form field'. It contains three input fields: 'Form field type' set to 'input', 'Variable type' set to 'string', and 'Variable name' set to 'variable\_name'. At the bottom right of the dialog are 'CANCEL' and 'SAVE' buttons.

Field Type	Type	Name
input	string	variable_name

CANCEL    **SAVE**

Gambar 3.33 *Add a from field*

## 5. Runtime :



Gambar 3.34 Tools Runtime

- **Run all** : Perintah untuk menjalankan semua yang ada di dalam *notebook*.

A screenshot of a Jupyter Notebook cell. The cell contains the following Python code:

```
print("Hello Guys!")  
...  
y = 7  
for e in range(y):  
    print (e)  
...
```

Gambar 3.35 Run all

- *Run before* : Perintah untuk menjalankan *source code* sebelumnya atau yang dimulai dari awal.

The screenshot shows a Jupyter Notebook interface. A cell containing the code `print("Hello Guys!")` is at the top. Below it, there is an ellipsis (...). A second cell is highlighted with a grey background, containing the code `y = 7` followed by a for loop `for e in range(y): print (e)`. The output of this cell is a list of integers from 0 to 6, each preceded by a right-pointing arrow (⇒). The code cell itself has a play button icon to its left.

Gambar 3.36 *Run before*

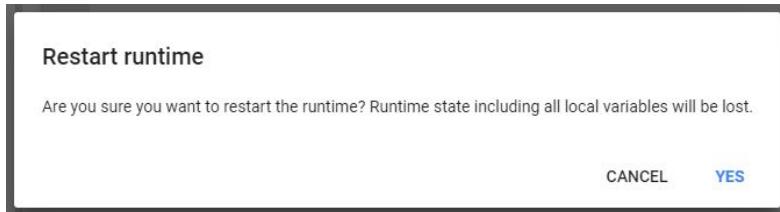
- *Run the focused cell* : Perintah yang hanya akan merunning jika satu cell yang di fokuskan.

The screenshot shows a Jupyter Notebook interface. At the top, a cell with index [15] contains the code `print("Hello Guys!")`. The output is "Hello Guys!", preceded by a right-pointing arrow (⇒). Below it is another cell containing the same code as in Gambar 3.36: `y = 7` and a for loop. This second cell is highlighted with a grey background and has a play button icon to its left. An ellipsis (...) is shown below the second cell.

Gambar 3.37 *Run the focused cell*

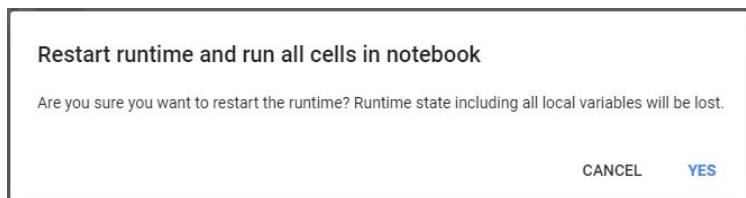
- *Run selection* : Perintah untuk menjalankan *source code* yang di seleksi saja.
- *Run after* : Perintah untuk menjalankan pada *source code* sebelumnya.

- *Interrupt execution* : Perintah untuk menjalankan eksekusi interupsi yang sudah di perintah.
- *Restart runtime* : Perintah yang berfungsi atau berguna untuk memulai kembali menjalankan *source code*.



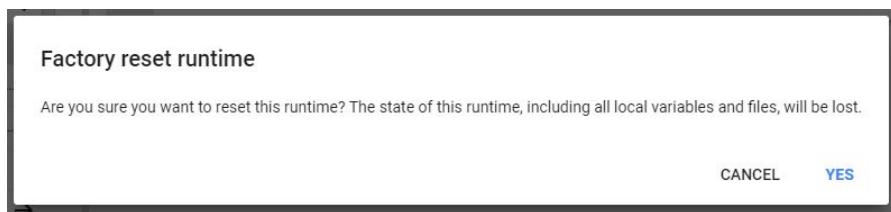
**Gambar 3.38** *Popup Restart runtime*

- *Restart and run all* : Perintah untuk memulai ulang dan setelah itu menjalankan semua atau merunning *source code* yang ada di dalam notebook.



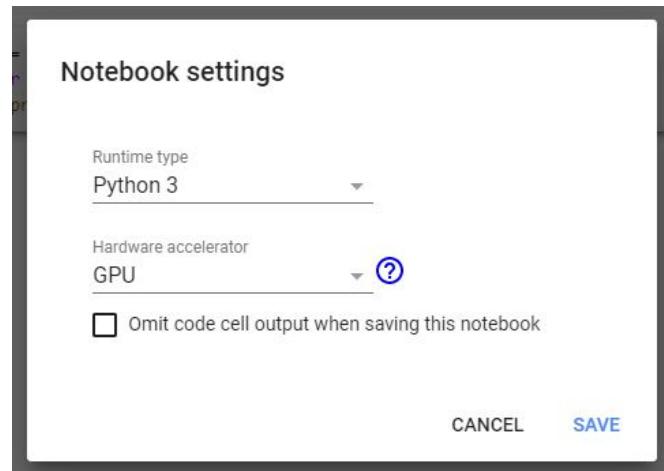
**Gambar 3.39** *Popup Restart and run all*

- *Factory reset runtime* : Perintah untuk mengatur ulang runtime termasuk semua variabel dan file lokal, akan hilang.



**Gambar 3.40** *Popup Factory reset runtime*

- *Change runtime type* : Perintah yang berguna untuk mengubah jenis runtime type dan hardware accelerator.

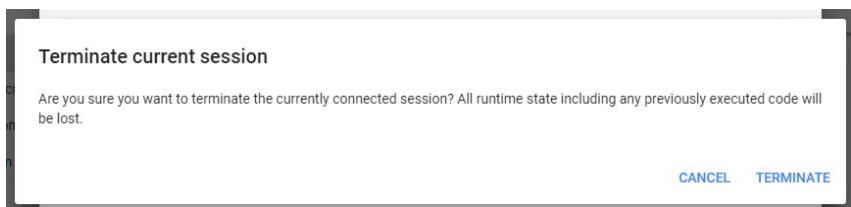


**Gambar 3.41** *Change runtime type*

- *Manage sessions* : Perintah untuk mengakhiri sesi yang sudah terhubung dan semua status runtime termasuk kode yang dieksekusi akan hilang.

Active sessions				
Title	Last execution	RAM used		
Untitled0.ipynb Current session	GPU	in 1 minute	0.14 GB	<a href="#">TERMINATE</a>
				<a href="#">CLOSE</a>

**Gambar 3.42** *Manage sessions*



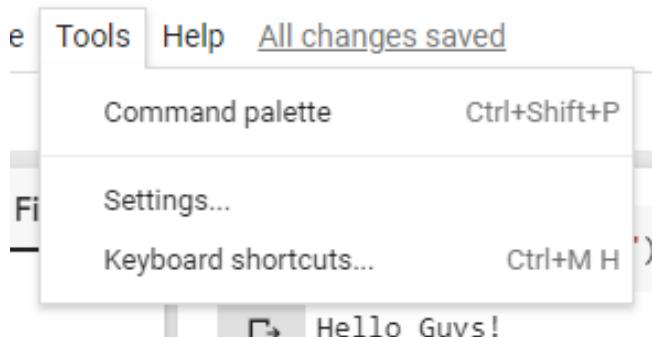
**Gambar 3.43** Setelah Terminate

- *View runtime logs* : Perintah untuk merunning atau menjalankan *source code* yang ada di notebook melalui colab-jupyter.log.

Timestamp	Level	Message
Jan 20, 2020, 3:12:43 AM	INFO	Adapting to protocol v5.1 for kernel 64611f7a-2eb2-4d2f-838c-446abfad30e0
Jan 20, 2020, 2:55:52 AM	INFO	Kernel restarted: 64611f7a-2eb2-4d2f-838c-446abfad30e0

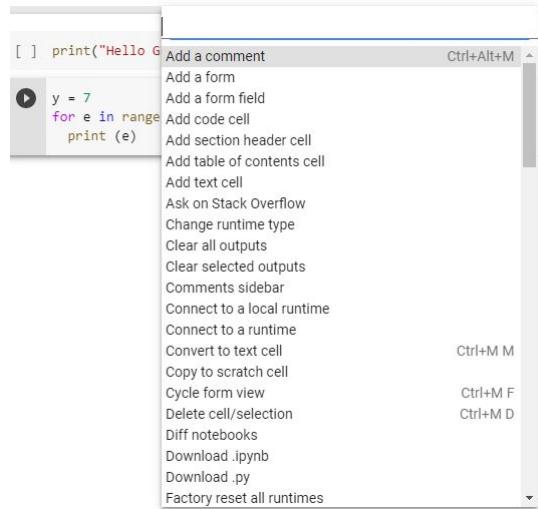
**Gambar 3.44** View runtime logs

## 6. Tools :



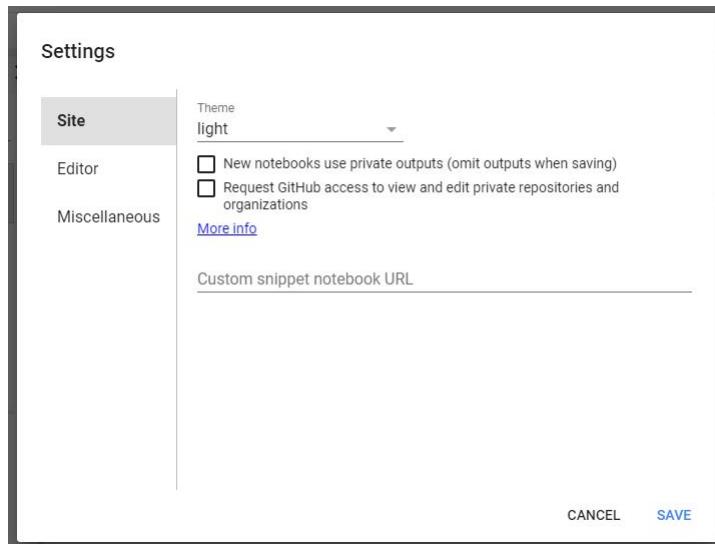
**Gambar 3.45** Tools

- *Command palette* : Palet perintah mencakup daftar item atau perintah yang sering digunakan.



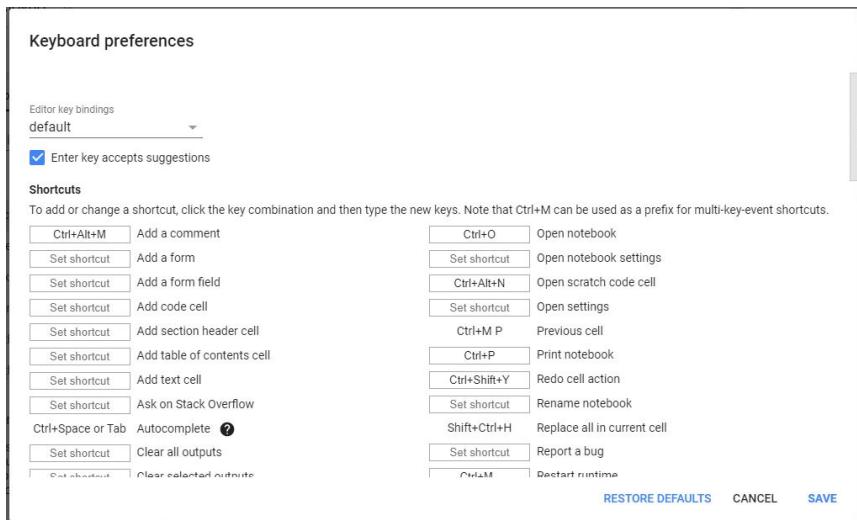
**Gambar 3.46** *Command palette*

- *Settings* : Perintah untuk mengatur pengaturan pada google colab seperti mengatur site, editor, dan Miscellaneous.



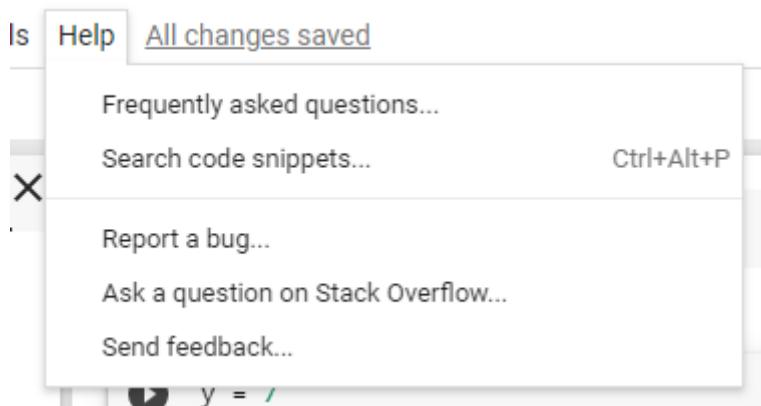
**Gambar 3.47** *Settings*

- **Keyboard shortcuts** : Yaitu perintah atau petunjuk *Keyboard preferences* yang dapat di gunakan di dalam google colab agar mempermudah pengguna dalam memakai google colab.



**Gambar 3.48** *Keyboard shortcuts*

## 7. Help :



**Gambar 3.49** *Tools Help*

- *Frequently asked questions* : Perintah yang sering digunakan untuk pertanyaan yang sering diajukan sehingga akan terjadi saling tanya jawab.

The screenshot shows a web browser displaying the 'Frequently Asked Questions' section of the Google Colaboratory website. The page has a light gray background with a white header containing the Google logo. Below the header, the word 'Colaboratory' is written in a large, dark font. Underneath it, the heading 'Frequently Asked Questions' is displayed in bold black text. A sub-section titled 'The Basics' follows, with a question 'What is Colaboratory?' and its detailed answer. Other sections visible include 'Is it really free to use?', 'Seems too good to be true. What are the limitations?', and 'What is the difference between Jupyter and Colab?'. The page ends with a '... more' ellipsis.

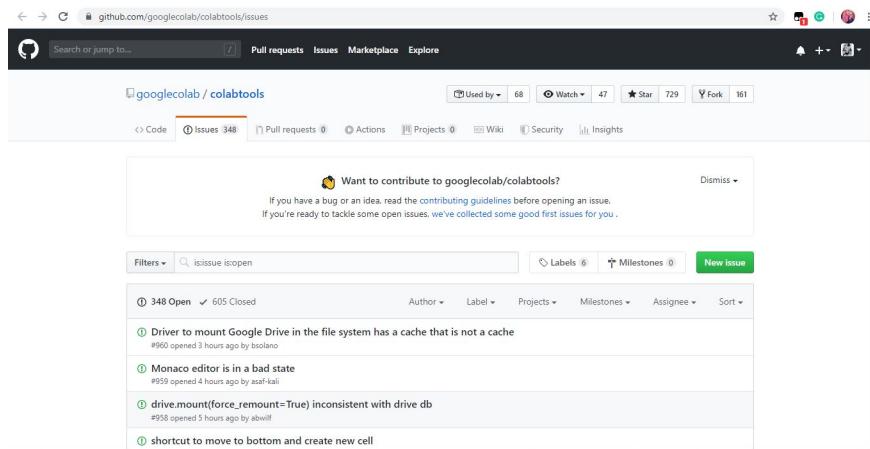
**Gambar 3.50** *Frequently Asked Questions*

- *Search code snippets* : Perintah untuk membantu pengguna google colab dalam mencari cuplikan dari *source code* yang nantinya akan menampilkan full hanya bagian dari *source code* saja seperti pada gambar 3.51.

The screenshot shows a Jupyter Notebook interface with a light gray background. At the top, there's a toolbar with various icons for file operations like 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help'. Below the toolbar, the title 'Untitled0.ipynb' is shown along with a star icon and a lock icon. The main area contains a code cell with the following Python code:[ ] print("Hello Guys!")  
y = 7  
for e in range(y):  
 print (e) To the right of the code cell is a panel with several small icons for file operations like 'Comment', 'Share', 'Edit', and 'Delete'. Below the code cell, there are buttons for 'Code' and 'Text' and a dropdown menu for 'RAM' and 'Disk' settings. The bottom right corner of the interface has a set of small control icons.

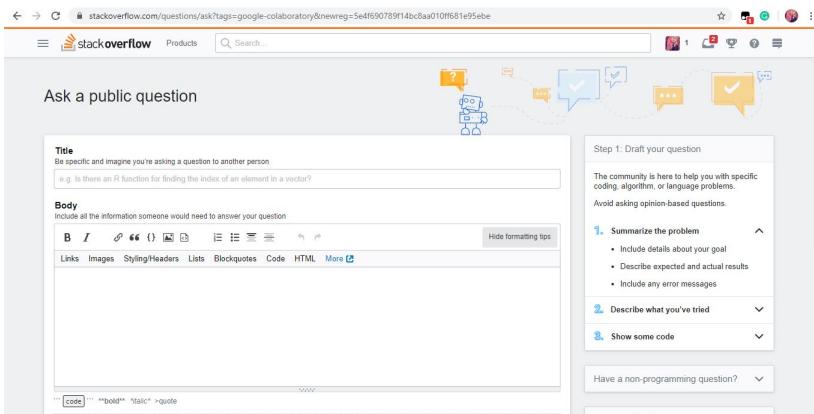
**Gambar 3.51** *Search code snippets*

- *Report a bug* : Perintah untuk melaporkan bug yang ada pada google colab yang setalah itu akan muncul pada tampilan github seperti pada gambar 3.52.



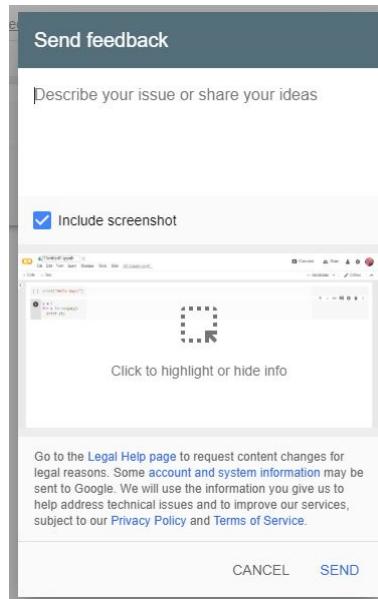
**Gambar 3.52 Report a bug**

- *Ask a question on Stack Overflow* : Perintah untuk mencari pertanyaan yang di tanyakan pada *platform Stack Overflow*, jika belum memiliki akun pada *Stack Overflow* dapat membuat dulu akun terlebih dahulu atau dapat juga langsung login dengan menggunakan akun google yang sudah ada.



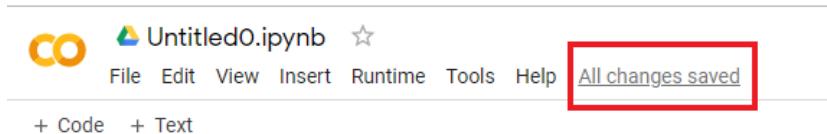
**Gambar 3.53 Ask a question on Stack Overflow**

- *Send feedback* : Perintah untuk melakukan meminta umpan balik dari apa yang telah di buat.



**Gambar 3.54** *Send feedback*

8. All changes saved : Perintah untuk melihat semua history yang dikerjakan.



**Gambar 3.55** *All changes saved*

The screenshot shows a Google Colab notebook interface. On the left, there are two code cells. The first cell, titled 'Untitled0.ipynb' (Tue Dec 24 2019 14:08:51 GMT-0700 (Indochina Time)), contains Python code to print 'Hello Guys!' and a range loop from 0 to 6. The second cell, also titled 'Untitled0.ipynb' (Mon Jan 20 2020 02:51:54 GMT-0700 (Indochina Time)), contains similar code. To the right, a sidebar displays a list of saved versions, each with a timestamp, author, and three-dot menu icon.

```

Code cell <HLVEY_-LKJ5d>
#%%
[code]
1 print("Hello Guys!")
Execution output from Dec 18, 2019 9:02 AM
Stream
Hello Guys!

Code cell <TNPbeh6aKjq2>
#%%
[code]
1 y
2 for e in range(y):
3 | print (e)
Execution output from Dec 9, 2019 12:39 PM
Stream
0
1
2
3
4
5
6

```

- Only show named versions
- Jan 20, 2020 2:51 AM  
Esi Vidia Rahmadani
- Dec 24, 2019 2:08 PM  
Esi Vidia Rahmadani
- Dec 18, 2019 3:15 PM  
Esi Vidia Rahmadani
- Pinned version  
Dec 18, 2019 8:29 AM  
Esi Vidia Rahmadani
- Pinned version  
Dec 18, 2019 8:23 AM  
Esi Vidia Rahmadani
- Pinned version  
Dec 17, 2019 6:12 PM  
Esi Vidia Rahmadani
- Dec 9, 2019 1:31 PM

**Gambar 3.56** Tampilan All changes saved

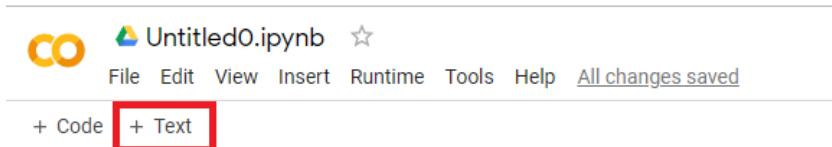
9. Code : Perintah untuk menambahkan code baru dalam mengerjakan di dalam notebook google colab.

The screenshot shows the top navigation bar of a Google Colab notebook. The title 'Untitled0.ipynb' is followed by a star icon. Below the title, the menu bar includes File, Edit, View, Insert, Runtime, Tools, Help, and 'All changes saved'. At the bottom of the bar, there are two buttons: '+ Code' and '+ Text', with '+ Code' being highlighted with a red box. The main workspace below the bar is currently empty.

**Gambar 3.57** Code

**Gambar 3.58** Tampilan Code

10. Text : Perintah untuk menambahkan text baru dalam mengerjakan di dalam notebook google colab.



**Gambar 3.59** *Text*



**Gambar 3.60** *Tampilan Text*

## BAB 4

---

# LIBRARY PYTHON

---

### 4.1 Konsep Penting untuk Dimengerti tentang Library Python

Sebelum masuk ke penjelasan berbagai macam library Python, kita harus memahami beberapa konsep. Misalnya, deep learning (pembelajaran secara dalam) adalah sebuah proses dari machine learning (pembelajaran mesin). Apakah kamu tahu bagaimana orang dapat belajar dari kesalahan mereka? Hal yang sama berlaku untuk komputer. Deep learning bertujuan untuk membuat mesin belajar melalui contoh.

Istilah lain yang relevan adalah neural network atau jaringan saraf, yang menyerupai otak manusia. Dengan cara bagaimana? Neural network adalah kombinasi dari algoritma yang bertujuan untuk meniru cara manusia yang mampu mengidentifikasi berbagai pola. Oleh karena itu, konsep ini mengambil biologi manusia dan merapkannya ke dunia pemrograman untuk memperkenalkan pengenalan gambar dan ucapan (hanya salah satu opsi).

Pertama, kamu harus memahami bahwa library untuk Python tidak jauh berbeda dari perpustakaan biasa yang kamu kunjungi untuk mencari dan memilih buku yang menarik. Keduanya merupakan kumpulan sumber informasi.

Python libraries

Namun, alih-alih buku, kamu memilih modul yang akan kamu terapkan selama proses coding kamu. Semua developer profesional memanfaatkan modul yang ditulis dengan baik. Jika ada cara yang mudah untuk melakukan sesuatu, mengapa kamu tidak mengambil cara ini?

Ketika kamu mulai meneliti library Python, kamu akan dihujani oleh sejumlah library asli maupun library pihak ketiga. Ada banyak koleksi modul yang tersedia. Karena itu, kamu mungkin akan merasa bingung ketika harus memutuskan yang mana untuk dijelajahi. Jika kamu adalah seorang programmer yang ingin menjadi unggul dalam beberapa domain yang berbeda, mungkin akan sulit untuk memilih library yang paling cocok.

Kamu sebaiknya sudah tahu bahwa Python adalah bahasa yang sangat fleksibel. Ia adalah permata di dunia pemrograman karena penggunaannya bervariasi dari ilmu data/data science, pengembangan web, dan bahkan pembelajaran mesin. Kalau kamu adalah seorang programmer Python pemula, kami menganjurkan untuk mengambil kursus ini untuk memperdalam pengetahuan kamu.

Secara keseluruhan, berbagai library Python mencakup modul untuk area tertentu. Siapkah kita untuk memulai ekspedisi untuk mencari tahu apa itu TensorFlow, PyTorch, Numpy, Sklearn, dan perpustakaan populer lainnya?

Namun sebelumnya, apakah kamu kesulitan mencari pekerjaan sebagai programmer Python? Dalam kasus seperti itu, kami sangat menyarankan untuk membaca beberapa pertanyaan dan jawaban wawancara Python yang biasanya ditanyakan oleh para pengusaha dan perusahaan. Jika kamu tidak bisa menjawabnya, kamu mungkin tampaknya tidak siap untuk posisi tersebut. Katakanlah salah satu pertanyaan wawancara Python mengharuskan kamu untuk berbicara tentang library Python.

Banyaknya kelebihan pada pemrograman python seperti efisiensi, keterbacaan kode dan kecepatan telah membuat python menjadi bahasa pemrograman yang banyak digunakan oleh para data scientist. Python menjadi pilihan untuk para data scientist dan machine learning engineer untuk mengembangkan model dan berbagai aplikasi terkait data science.

Karena penggunaannya yang luas, Python memiliki banyak library yang memudahkan para ilmuwan data / data scientist untuk menyelesaikan tugas-tugas rumit tanpa banyak gangguan pengkodean. Berikut adalah 3 library Python yang paling banyak digunakan untuk data science.

## 4.2 Library Terbaik untuk Dipertimbangkan

PI adalah singkatan untuk application programming interface (antarmuka pemrograman aplikasi). Ia membuka jendela untuk interaksi antara aplikasi melalui komunikasi mesin-ke-mesin. Python memiliki framework (kerangka kerja) yang mempercepat proses pembuatan API. Oleh karena itu, misi kita adalah membahas secara singkat library paling umum untuk Python yang dapat kamu pilih:

1. Flask adalah framework web yang berkembang pesat, dirancang untuk proses perancangan API yang lebih efisien. Ini hanya salah satu dari banyak kemungkinan penggunaan Flask. Secara umum, ini adalah framework untuk pengem-

bangun aplikasi web. Flash ringan, menawarkan dukungan untuk pengujian unit dan mengamankan cookie untuk sesi di sisi klien. Para developer memuji framework ini karena didokumentasikan dengan baik, artinya kamu akan menemukan banyak contoh penggunaan untuk dipelajari.

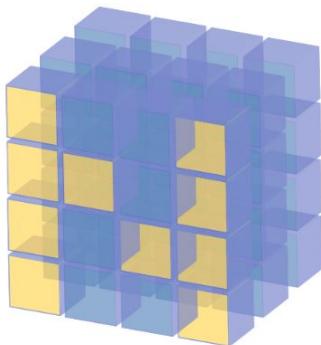
2. Django adalah salah satu framework web pihak ketiga berbasis Python. Di antara library Python lainnya, tujuan utama Django dari framework ini adalah untuk menyederhanakan proses pengembangan situs web kompleks yang digerakkan oleh database. Perpustakaan Django menyediakan banyak alat (tool) manajemen. Oleh karena itu, para developer akan dapat menghasilkan bagian-bagian kode tanpa harus beralih ke alat lain. Django REST adalah framework untuk membuat API Web dengan kode yang minimal.
3. Falcon adalah framework web yang ringan dan sesuai dengan SWGI, yang dirancang untuk membangun RESTful APIs. Para pemula menghargai tutorial yang terdokumentasi dengan baik yang menyediakan banyak panduan untuk pembuatan proyek pertama. Falcon berjalan pada hardware apa saja dan hanya bergantung pada dua dependensi pihak ketiga.
4. Eve adalah framework REST API berbasis Python gratis, ditenagai oleh Flask dan Cerberus. Ia memungkinkan layanan web RESTful yang unik dan kaya fitur untuk dikembangkan secara cepat. Framework ini mendukung MongoDB dan sangat kompatibel karena ekstensi.

### 4.3 library NumPy

NumPy (kependekan dari Numerical Python) adalah salah satu library teratas yang dilengkapi dengan sumber daya yang berguna untuk membantu para data scientist mengubah Python menjadi alat analisis dan pemodelan ilmiah yang kuat. Library Open source terpopuler ini tersedia di bawah lisensi BSD. Ini adalah pustaka Python dasar untuk melakukan tugas dalam komputasi ilmiah. NumPy adalah bagian dari ekosistem berbasis Python yang lebih besar dari tool open source yang disebut SciPy.

Perpustakaan memberdayakan Python dengan struktur data substansial untuk mudah melakukan perhitungan multi-dimensi (multi-dimensional arrays) dan perhitungan matrik. Selain penggunaannya dalam menyelesaikan persamaan aljabar linier (linear algebra equations) dan perhitungan matematis lainnya, NumPy juga digunakan sebagai wadah multi-dimensi serbaguna untuk berbagai jenis data generik.

Lebih hebatnya, NumPy terintegrasi dengan bahasa pemrograman lain seperti C / C ++ dan Fortran. Fleksibilitas perpustakaan NumPy memungkinkannya untuk dengan mudah dan cepat bergabung dengan berbagai database dan tools. Sebagai contoh, mari kita lihat bagaimana NumPy (disingkat np) dapat digunakan untuk mengalikan dua matriks.



# NumPy

**Gambar 4.1** NumPy

List pada Python tidak mendukung penuh kemudahan scientific computing, sebagai contoh kita akan lakukan operasi penjumlahan pada 2 list.

```
In [1]: 1 # list pada python
2 a = [1, 2, 3]
3 b = [4, 5, 6]
4 a + b
```

```
Out[1]: [1, 2, 3, 4, 5, 6]
```

```
In [2]: 1 # penjumlahan dilakukan iterasi
2 result = []
3 for first, second in zip(a, b):
4     result.append(first + second)
5 result
```

```
Out[2]: [5, 7, 9]
```

**Gambar 4.2**

Ketika kita ingin menjumlahkan tiap elemen pada list a dan list b, hasilnya dengan operator + adalah penggabungan (concat) keduanya. Tentu tidak sesuai yang diharapkan, maka kita harus menggunakan perulangan for untuk menambahkan tiap elemen pada list a dan list b. Proses penjumlahan list yang menggunakan perulangan for membutuhkan waktu yang lama dan tidak efisien dari sisi penulisan code.

\*catatan, disini di beberapa tempat akan dijelaskan code menggunakan Ipython. Misal In[10] artinya adalah input code oleh user pada baris input ke 10. Out[10] adalah output code yang ditulis pada In[10]. Kalau mau mengcopy paste code ini, silahkan hilangkan symbol tsb

Contoh pembuatan array 1 dimensi:

```
1 Import numpy as np
2 data=[5, 7.2, 8 0, 1]
3 arr=np.array(data)
```

```
4 arr2=np.array([5, 7.2, 8, 0, 1])
```

note\* arr dan arr2 akan berisi sama. Kita dapat mendefinisikan list angkanya dahulu seperti pada arr atau langsung memasukkan list ke dalam fungsi array

Contoh pebuatan array 2 dimensi:

```
1 Import numpy as np
2 data=[[1,2,3,4,5], [6,7,8,9,10]]
3 arr=np.array(data)
```

Contoh untuk menghitung dimensi suatu array dapat menggunakan fungsi .dim

```
1 In[1]: arr.ndim
2 out[1]: 2
```

note\* arr pada contoh 2 adalah array 2 dimensi, sehingga arr.ndim akan bernilai 2

Contoh untuk menghitung ukuran suatu arrau dapat menggunakan fungsi .shape

```
1 Import numpy as np
2 data=[[1,2,3,4,5], [6,7,8,9,10]]
3 arr=np.array(data)
4 ukuran=arr.shape
```

note\* maka disini variable ukuran akan bernilai [2,5] yaitu punya baris 2 dan kolom 5

Contoh membuat array berisi bilangan zero 0 menggunakan fungsi .zeros, membuat array berisi bilangan one 1 menggunakan fungsi .ones

```
1 Import numpy as np
2 zero1=np.zeros(10)
3 zero2=np.zeros((2,3))
4 one1=np.ones(8)
5 one2=np.ones((3,2))
6 rand1=np
```

note\* disini zero1 adalah array berdimensi 1 dengan panjang 10 yang berisi semuanya bilangan 0. zero2 berisi array berdimesni 2 dengan ukuran 2x3 yang berisi semuanya bilangan 0. sama seperti itu, one1 adalah array berdimensi 1 dengan panjang 8 yang berisi semuanya bilangan 1, sedang one2 adalah array berdimensi 2 dengan ukuran 3x2 yang berisi semuanya bilangan 1.

Contoh menggunakan fungsi ones\_like dan zeros\_like untuk membuat array ones atau zeros seperti contoh 5, dengan ukuran yang sama seperti suatu array tertentu

```
1 Import numpy as np
2 data1=[[1,2,3,4,5], [6,7,8,9,10]]
3 data2=[1,2,3,4]
4 arr1=np.array(data1)
5 arr2=np.array(data2)
6 zeros1=np.zeros_like(arr1)
7 ones1=np.ones_like(arr2)
```

note\* zeros1 akan membuat array zeros (berisi semuanya angka 0) dengan ukuran yang sama dengan ukuran aray arr1 yaitu 2x5. Sedang ones1 akan membuat array ones (berisi semuanya angka 1) dengan ukuran array arr2 yaitu berdimensi 1 dengan panjang 4.

Contoh membuat matriks identitas (matriks dua dimensi NxN dengan semuanya bernilai 0 kecuali pada diagonalnya yaitu bernilai 1), coba google matriks identitas. dapat menggunakan fungsi .eye atau .identity, keduanya sama.

```

1 Import numpy as np
2 arr1=eye(10)
3 arr2=identity(10)
```

note\* baik arr1 maupun arr2 akan sama-sama berisi matriks identitas dengan besar 10x10

Contoh untuk membuat array dengan format data tertentu (misal int32, atau float64)

```

1 Import numpy as np
2 arr1=np.array([1,2,3,4], dtype=np.float64)
3 arr1=np.array([1,2,3,4], dtype=np.int32)
```

\*note: terdapat beberapa format yaitu: int8, uint8, int16, uint16, int32, uint32, int64, uint64, float16, float32, float64, float128, complex64, complex128, complex256, bool, object, string\_, unicode\_. int adalah bilangan bulat, uint adalah bilangan bulat positif. float adalah bilangan desimal. complex adalah bilangan kompleks. bool adalah boolean (nilainya 0 atau 1). Bilangan setelah int,uint,float,complex adalah jumlah bit yang dibutuhkan. Semakin besar niali bitnya, semakin besar bilang yang bisa disimpan.

Contoh untuk melihat tipe data suatu array dapat menggunakan fungsi ,dtype

```

1 In [1]: arr1.dtype
2 out[1]: dtype('float64')
3 In [2]: arr2.dtype
4 out[2]: dtype('int32')
```

Contoh untuk mengubah suatu tipe data array dapat menggunakan fungsi .astype(np.typedata)

```

1 Import numpy as np
2 arr1=np.array([1,2,3,4], dtype=np.float64)
3 arr2=arr1.astype(np.int16)
```

\*note: disini arr1 bertipe float64, dan arr2 adalah arr1 akan teapi tipe datanya diubah menjadi int16

Contoh membuat array dengan angka yang berurutan dari 0-N-1 menggunakan fungsi np.arange(N)

```

1 Import numpy as np
2 arr=np.arange(10)
```

\*note : disini arr akan berisi arr=[0,1,2,3,4,5,6,7,8,9]

Contoh operasi antar array: dot product \*, tambah +, kurang -, pembagian /, pangkat \*\*

```
1 Import numpy as np
2 arr=np.arange(10)
3 arr2=arr*arr
```

Contoh indexing, slicing array PENTING

```
1 Import numpy as np
2 arr=np.array([1,2,3,4,5,6,7,8,9,10])
3 arr2=arr[4:6]
4 arr2[1]=8080
```

\*note: python memulai index dengan 0, sehingga arr2=[5,6,7]. arr2[1] =5, dengan begini, sekarang tebak apa isi arr? : CAUTION arr=[1,2,3,4,8080,6,7,8,9,10] !!!!!!!!. tidak seperti bahasa program yang lain. arr2 bukannya meng-copy data dari arr, tapi arr2 merupakan representasi dari arr itu sendiri. untuk mengcopy array, gunakan fungsi .copy() spt pada contoh 13

Contoh mengcopy data array PENTING (PASTIKAN SUDAH BACA CONTOH 12)

```
1 Import numpy as np
2 arr=np.array([1,2,3,4,5,6,7,8,9,10])
3 arr2=arr[4:6].copy()
4 arr2[1]=8080
```

\*note: "arr" akan tetap

Contoh indexing multidimensi array

```
1 Import numpy as np
2 arr=np.array([[1,2,3,4],[4,5,6,7],[8,9,10,11]])
3 arr2=arr[1,3] #baris 1, kolom 3=[6]
4 arr3=arr[2,0] #baris 1, kolom 0=[8]
5 arr4=arr[:2,1] #baris 0 sampai 2. kolom 1= [[2],[5],[9]]
6 arr5=arr[2,:,:] #baris 2 sampai terahir , kolom semuanya = [8,9,10,11]
```

Contoh dan membuat array berisi bilangan random 0-1 menggunakan fungsi randn dalam fungsi numpy.random

```
1 Import numpy as np
2 from numpy.random import randn
3 data=randn(7,4) #:membuat bilangan random baris dengan ukuran 7x4
```

### 4.3.1 Pengenalan NumPy Arrays

#### Membuat Array

Lakukan import terlebih dahulu library numpy as np. Penggunaan as disini, artinya kita menggantikan pemanggilan numpy dengan prefix np untuk proses berikutnya.

```
In [3]: 1 import numpy as np  
In [4]: 1 # membuat array  
2 a = np.array([1, 2, 3])  
3 a  
Out[4]: array([1, 2, 3])
```

Gambar 4.3

Untuk membuat sebuah array, kita menggunakan fungsi array() yang terdapat pada NumPy. Pada NumPy, terdapat upcasting, yaitu ketika tipe data element array tidak sama, dilakukan penyamaan tipe data pada yang lebih tinggi. Misalkan kita membuat array numeric dengan semua element bertipe integer, kecuali 1 element bertipe float, maka otomatis akan dilakukan upcasting menjadi tipe float pada semua element array.

Untuk melakukan pengecekan tipe pada array menggunakan fungsi type().

NumPy array merupakan sebuah objek ndarray, yang merupakan singkatan dari n-dimensional array. Tipe Data untuk Element Pengecekan tipe data element pada array menggunakan fungsi dtype.

Dalam membuat sebuah array, kita dapat menetapkan tipe data dengan menambahkan parameter dtype.

Array a memiliki tipe data int32 dan int64 yang keduanya sama-sama bertipekan integer. Perbedaan keduanya pada kapasitas penyimpanan data. Pada int32 mampu menampung hingga (-2,147,483,648 to +2,147,483,647) sedangkan int64 mampu menampung hingga (-9,223,372,036,854,775,808 to +9,223,372,036,854,775,807). Penting bagi kita memperhatikan tipe data beserta kapasitas penyimpanannya agar dapat mengalokasikan memory penyimpanan dengan baik. Beberapa tipe data standar yang terdapat pada NumPy adalah sebagai berikut:

NumPy array memiliki keunggulan mendukung operasi pada data dimensional seperti Vektor dan Matriks. Untuk mengetahui jumlah dimensi pada data menggunakan fungsi ndim.

Array a memiliki jumlah dimensi 1. Jika kita membentuk array dengan 2-dimensi, maka jumlah dimensinya adalah 2, begitu juga dengan dimensi yang lebih besar. Array Shape Pada fungsi shape menghasilkan sebuah tuple yang berisikan panjang sebuah array pada tiap dimensi.

Array a memiliki shape 3, yaitu panjang array pada 1-dimensi array. Operasi pada Array NumPy memudahkan kita untuk operasi elementwise pada Vektor dan Matriks seperti penjumlahan, perkalian, pangkat, dan operasi lainnya.

Pada NumPy telah disediakan universal functions (ufunc) yaitu fungsi yang dapat melakukan operasi pada NumPy array dengan eksekusi elemen-demi-elemen. ufunc

merupakan sebuah vectorized wrapper untuk sebuah fungsi yang memiliki input dan output yang spesifik. Contoh dari ufunc misalkan fungsi sin dan cos.

Element pada Array Array Indexing Indeks pada suatu array dimulai pada indeks ke-0. Untuk mengakses element pada array, kita menggunakan indeks sebagai alamat elemen pada array.

Kita dapat melakukan assign nilai baru pada suatu element berdasarkan alamat indeks. Maka, setelah dilakukan assign nilai element pada indeks ke-0 berganti menjadi 10. Perlu kita perhatikan jika dtype pada array adalah integer, misalkan int32. Kemudian kita assign nilai baru pada salah satu elemen dengan tipe data float. Maka, nilai baru tersebut akan dipaksa menjadi tipe int32 sesuai dengan tipe data pada array.

Multi-Dimensional Arrays NumPy array memudahkan kita untuk membuat array multi-dimensi.

Fungsi shape pada array multi-dimensi menghasilkan tuple berisikan (jumlah baris, jumlah kolom).

Untuk mengetahui jumlah elemen pada array multi-dimensi menggunakan fungsi size.

Untuk mengetahui jumlah dimensi dari array multi-dimensi menggunakan fungsi ndim. Jumlah dimensi pada array a adalah 2, karena merupakan array 2-dimensi.

Pada array multi-dimensi kita dapat mengakses nilai elemen berdasarkan indeks dengan pemisah tanda koma [ , ]. Misalkan kita ingin mendapatkan data pada baris ke-1 dan kolom ke-3 dari array a, maka dilakukan perintah a[1, 3].

Misalkan, kita akan mengganti data pada baris ke-1 dan kolom ke-3 pada array a dengan nilai -1, kita dapat langsung assign nilai baru tersebut pada indeks.

Slicing Selain kita dapat mengambil sebuah element array dengan cara indexing seperti cara di atas, kita dapat melakukan slicing, yaitu melakukan ekstraksi elemen atau beberapa elemen pada array, dengan menggunakan tanda [ : ]. var[lower:upper:step] Perlu kita perhatikan, elemen pada lower akan dimasukkan hasil slicing, sedangkan element pada upper TIDAK dimasukkan hasil slicing, dan step adalah jarak antar elemen.

Kita dapat mendefinisikan indeks dengan nilai negatif. Misalkan indeksnya adalah [-1], maka indeks terdapat pada elemen ke-1 yang dari lokasi akhir elemen suatu array. NumPy masih menyediakan banyak sekali fungsi yang sangat memudahkan kita untuk scientific computing. Untuk mempelajari lebih jauh dapat mengakses dokumentasi NumPy.

## 4.4 Pandas

Pandas adalah library hebat lain yang dapat meningkatkan keterampilan Python Anda untuk data science. Sama seperti NumPy, Pandas milik keluarga perangkat lunak open source SciPy dan tersedia di bawah lisensi perangkat lunak bebas BSD.

Pandas menawarkan alat serbaguna dan kuat untuk struktur data dan melakukan analisis data yang luas. Library ini berfungsi dengan baik dengan data dunia nyata yang tidak lengkap, tidak terstruktur, dan tidak teratur dan dilengkapi dengan tool untuk membentuk, menggabungkan, menganalisis, dan memvisualisasikan datasets.

Pandas (Python for Data Analysis) adalah library Python yang fokus untuk proses analisis data seperti manipulasi data, persiapan data, dan pembersihan data. Pandas menyediakan struktur data dan fungsi high-level untuk membuat pekerjaan dengan data terstruktur/tabular lebih cepat, mudah, dan ekspresif. Dalam pandas terdapat dua objek yang akan dibahas pada tutorial ini, yaitu DataFrame dan Series. DataFrame adalah objek yang memiliki struktur data tabular, berorientasi pada kolom dengan label baris dan kolom. Sedangkan Series adalah objek array 1-dimensi yang memiliki label.

Pandas memadukan library NumPy yang memiliki kemampuan manipulasi data yang fleksibel dengan database relasional (seperti SQL). Sehingga memudahkan kita untuk melakukan reshape, slice dan dice, agregasi data, dan mengakses subset dari data. Menurut penulis buku Python for Data Analysis sekaligus pembuat pandas, Wes McKinney, nama pandas berdasarkan dari panel data, yaitu istilah ekonometrik untuk data multi-dimensi terstruktur, dan berdasarkan dari kata yang merupakan fungsional library itu sendiri yaitu Python data analysis.

#### 4.4.1 Struktur Data Pandas

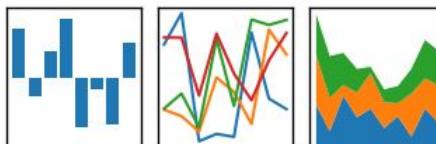
Saya telah menyebutkan salah satu struktur data pandas di atas, DataFrame. Saya akan menjelaskan struktur data ini di dalam section ini sebagai tambahan untuk struktur data pandas lainnya, Series. Ada struktur data lainnya bernama Panel, namun saya tidak akan menjelaskan itu di dalam tutorial ini karena itu tidak sering digunakan, seperti yang disebutkan di dalam dokumentasi. DataFrame adalah struktur data 2D, Series adalah struktur data 1D, dan Panel adalah struktur data 3D dan lebih tinggi.

Ada tiga jenis struktur data di library ini:

- Series: single-dimensional, array homogen
- DataFrame: two-dimensional dengan kolom yang diketik secara heterogen
- Panel: three-dimensional, array size-mutable

**pandas**

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



Gambar 4.4 Pandas

#### DataFrame

DataFrame adalah struktur data tabular yang disusun pada kolom dan baris berurut. Untuk membuatnya lebih jelas, mari lihat contoh pembuatan sebuah DataFrame

(tabel) dari kamus sebuah daftar. Contoh berikut menunjukkan sebuah kamus berisi dua kunci, Name dan Age, dan daftar nilainya.

```
1 import pandas as pd
2 import numpy as np
3
4 name_age = { 'Name' : [ 'Ali', 'Bill', 'David', 'Hany', 'Ibtisam' ],
5   'Age' : [ 32, 55, 20, 43, 30] }
6 data_frame = pd.DataFrame(name_age)
7 print data_frame
```

Jika kamu menjalankan script di atas, kamu harusnya mendapatkan sebuah input mirip dengan di bawah ini:

	Age	Name
0	32	Ali
1	55	Bill
2	20	David
3	43	Hany
4	30	Ibtisam

Gambar 4.5 DataFrame

Perhatikan bahwa constructor DataFrame mengurutkan kolom secara alfabetis. Jika kamu ingin mengubah urutan kolom, kamu dapat mengetikkan hal berikut di bawah data\_frame di atas:

```
1 data_frame_2 = pd.DataFrame(name_age, columns = [ 'Name' , 'Age' ])
```

Untuk melihat hasilnya, cukup ketik:

```
1 print data_frame_2
```

Katakan kamu tidak ingin menggunakan label default 0,1,2..., dan ingin menggunakan a, b, c,... sebagai gantinya. Dalam kasus itu, kamu dapat menggunakan index di dalam script di atas sebagai berikut:

```
1 data_frame_2 = pd.DataFrame(name_age, columns = [ 'Name' , 'Age' ] ,  
2 index = [ 'a' , 'b' , 'c' , 'd' , 'e' ])
```

Itu sangat bagus, bukan? Dengan menggunakan DataFrame, kita dapat melihat data kita tertata dalam sebuah bentuk tabular.

### Series

Series adalah objek 1-dimensi yang berisi sequence nilai dan berasosiasi dengan label data, yang disebut indeks. Untuk membuat sebuah Series, kita dapat membenaknya dari sebuah array dengan memanggil fungsi Series pada pandas.

Series adalah struktur data pandas kedua yang akan saya bicarakan. Series adalah object satu dimensi (1D) yang serupa dengan kolom di dalam tabel. Jika kita ingin membuat sebuah Series untuk daftar nama, kita dapat melakukan di bawah ini:

```
1 series = pd.Series([ 'Ali' , 'Bill' , 'David' , 'Hany' , 'Ibtisam' ] ,  
2 index = [ 1 , 2 , 3 , 4 , 5 ])  
3 print series
```

Output script ini akan berupa sebagai berikut:

```
1           Ali  
2           Bill  
3          David  
4           Hany  
5          Ibtisam  
dtype: object
```

Gambar 4.6 Series

Perhatikan bahwa kita menggunakan index untuk melabeli data. Jika tidak, label default akan mulai dari 0,1,2...

#### 4.4.2 Function Pandas

Dalam section ini, saya akan menunjukkan contoh beberapa function yang dapat kita gunakan dengan DataFrame dan Series.

##### Head dan Tail

Function head() dan tail() mengijinkan kita untuk melihat sebuah sampel data, khususnya ketika kita memiliki jumlah entri yang besar. Jumlah default dari elemen yang ditampilkan adalah 5, namun kamu dapat mengkustomasi angkanya sesukamu.

Mari katakan kita memiliki sebuah Series yang disusun dari 20,000 item (angka) secara acak:

```
1 import pandas as pd
2 import numpy as np
3 series = pd.Series(np.random.randn(20000))
```

engan menggunakan method head() dan tail() untuk mengamati lima item pertama dan lima item terakhir, kita dapat melakukan di bawah ini:

```
1 print series.head()
2 print series.tail()
```

Output script ini harusnya serupa dengan di bawah (perhatikan bahwa kamu mungkin memiliki nilai yang berbeda karena kita membentuk nilai acak):

##### ADD

Mari ambil contoh function add(), dimana kita akan berusaha untuk menambahkan dua data frames sebagai berikut:

```
1 import pandas as pd
2
3 dictionary_1 = {'A' : [5, 8, 10, 3, 9],
4 'B' : [6, 1, 4, 8, 7]}
5 dictionary_2 = {'A' : [4, 3, 7, 6, 1],
6 'B' : [9, 10, 10, 1, 2]}
7 data_frame_1 = pd.DataFrame(dictionary_1)
8 data_frame_2 = pd.DataFrame(dictionary_2)
9 data_frame_3 = data_frame_1.add(data_frame_2)
10 print data_frame_1
11 print data_frame_2
12 print data_frame_3
```

Output dari script di atas adalah:

Kamu dapat juga melakukan proses penambahan ini dengan cukup menggunakan operator +: data\_frame\_3 = data\_frame\_1 + data\_frame\_2.

##### Describe

```
0      -1.250659
1      -0.961649
2      -0.399952
3      1.853716
4      -0.309293
dtype: float64
19995      0.374166
19996      1.014064
19997      -0.225456
19998      -1.751164
19999      0.303555
dtype: float64
```

**Gambar 4.7** Head Tail

	A	B
0	5	6
1	8	1
2	10	4
3	3	8
4	9	7

	A	B
0	4	9
1	3	10
2	7	10
3	6	1
4	1	2

	A	B
0	9	15
1	11	11
2	17	14
3	9	9
4	10	9

Gambar 4.8 Add

Sebuah function pandas yang sangat bagus adalah describe(), yang membuat berbagai ringkasan statistik data kita. Sebagai contoh dalam section terakhir, mari lakukan berikut ini:

```
1 print data_frame_3.describe()
```

Output dari operasi ini akan berupa:

	A	B
<b>count</b>	5.00000	5.000000
<b>mean</b>	11.20000	11.600000
<b>std</b>	3.34664	2.792848
<b>min</b>	9.00000	9.000000
<b>25%</b>	9.00000	9.000000
<b>50%</b>	10.00000	11.000000
<b>75%</b>	11.00000	14.000000
<b>max</b>	17.00000	15.000000

Gambar 4.9 Add

Dapat kita lihat, pada Series obj ditampilkan dua buah kolom, bagian kiri adalah index dan bagian kanan adalah values. Pada index, karena kita tidak mendefinisikan index pada data, maka secara default dimulai dari integer 0 hingga N-1 (dimana N adalah panjang data). Kita dapat mendefinisikan index dengan menambahkan parameter index saat membuat objek Series.

Kita dapat mengambil index dari objek Series dengan menggunakan fungsi index dan mengambil values dari objek Series dengan menggunakan fungsi values.

Untuk mengakses suatu value pada objek Series, kita dapat menggunakan index sebagai alamat value tersebut. Hal ini memungkinkan untuk melakukan assign nilai baru pada objek Series.

Hasil operasi aritmatika tambah pada 2 objek Series mirip dengan operasi join pada pengolahan database. Indeks yang tidak memiliki kesamaan pada 2 objek Series akan memiliki value NaN.

DataFrame merupakan tabel data yang terdapat kolom dan baris, dimana nilai-nilai yang terdapat di dalamnya dapat berupa tipe berbeda seperti numeric, string, boolean, dll. DataFrame mirip dengan data 2-dimensi dengan adanya baris dan kolom. Selain itu, DataFrame bisa dikatakan gabungan dari dictionary objek Series yang memiliki indeks yang sama. Terdapat berbagai macam cara untuk memben-

tuk objek DataFrame. Salah satu cara yang biasa dilakukan untuk membentuk objek DataFrame dengan menggunakan data masukan berupa dictionary.

Kita dapat menggunakan fungsi shape untuk mengetahui jumlah baris dan kolom dari DataFrame. Fungsi shape pada DataFrame memiliki hasil keluaran yang sama dengan fungsi shape pada NumPy, dimana menghasilkan keluaran sebuah tuple dari jumlah baris dan jumlah kolom.

fungsi info() sangat berguna untuk mengetahui keterangan dari objek DataFrame yang kita buat seperti index dari DataFrame lengkap dengan range dari index, jumlah kolom beserta informasi tiap kolom untuk null data dan tipe data, dan jumlah total penggunaan memory pada tiap kolom dalam satuan bytes. Saat kita input data dictionary pada DataFrame, kita tidak perlu mendefinisikan tipe data untuk masing-masing kolom, karena secara otomatis pandas akan memberikan tipe data sesuai dengan values untuk tiap kolom, meskipun kita juga bisa mendefinisikan tipe data secara manual.

Misalkan kita memiliki objek DataFrame yang memiliki baris hingga jutaan, kita tidak ingin menampilkan data secara keseluruhan karena akan menghabiskan memory. Kita dapat menggunakan fungsi head() dan tail() untuk menampilkan data secara default untuk 5 data teratas dan 5 data terbawah.

Seperti pada Series, kita juga dapat mengakses kolom pada DataFrame menggunakan fungsi columns dan untuk mengakses indeks dari DataFrame menggunakan fungsi index. Jika ingin mendapatkan data pada DataFrame secara keseluruhan menggunakan fungsi values yang akan menghasilkan output berupa array 2-dimensi sesuai dengan jumlah baris dan kolom.

ataFrame menyediakan fungsi describe() untuk mengetahui statistika data untuk data numeric seperti count, mean, standard deviation, maximum, minum, dan quartile. Untuk data string, misalkan data tersebut adalah kategori, kita dapat menggunakan fungsi value\_counts() untuk mengetahui jumlah tiap kategori pada data.

### Mengakses Data pada DataFrame

Terkadang kita butuh mengakses kolom tertentu atau lebih spesifik elemen tertentu pada DataFrame. Untuk mengakses suatu data, alamat data tersebut adalah pada nama kolom sebagai petunjuk lokasi kolom dan indeks sebagai petunjuk lokasi baris. Misalkan untuk mengakses semua data pada kolom populasi, maka kita menggunakan perintah frame[populasi]. Perlu diingat karena nama kolom adalah tipe data string, maka kita menggunakan petik.

Sedangkan mengakses data pada baris tertentu, kita menggunakan fungsi loc[indeks]. Kita juga bisa mendapatkan lebih dari 1 baris dengan menggunakan titik dua :, misalkan kita ingin mengakses indeks 23, maka menggunakan perintah loc[2:3].

Elemen pada DataFrame dapat diakses dengan mendefinisikan nama kolom dan indeks baris secara bersamaan. Misalkan kita ingin mendapatkan data populasi pada indeks ke-2, maka digunakan perintah frame[populasi][2].

Pandas masih menyediakan banyak sekali fungsi yang sangat memudahkan kita untuk analisis data seperti mengisi data kosong dengan fungsi fillna(), menerapkan suatu fungsi pada DataFrame menggunakan apply(), mengubah format DataFrame dari wide ke long menggunakan melt(), dan masih banyak lagi. Untuk mempelajari lebih jauh dapat mengakses dokumentasi pandas.

### 4.4.3 Menginstall Pandas

Untuk dapat menginstall pandas, kita bisa menjalankan perintah berikut :

```
| pip install pandas
```

Atau jika teman-teman menggunakan library Anaconda, kita bisa menginstallnya dengan perintah berikut :

```
| conda install pandas
```

### 4.4.4 Mencoba Series

Series merupakan sebuah array satu dimensi yang memiliki label dan digunakan untuk menyimpan beragam tipe data seperti integer, string, float, bahkan objek, dan lain sebagainya. Label pada series disebut dengan index. Bagaimana cara membuat series ? Perintah dasar untuk membuat sebuah Series dengan pandas adalah

```
| pandas.Series( data , index , dtype , copy )
```

Penjelasan Kode : **data** : parameter ini diisi dengan data yang akan dibuat series **index** : parameter ini diisi dengan index dari series. Jumlah index harus sama dengan jumlah data. Jika kita tidak mengisi parameter index, maka series akan memiliki index integer seperti halnya array biasa. **dtype** : parameter ini diisi dengan tipe data dari series, sebenarnya kita tidak perlu untuk mengisi parameter ini, karena secara otomatis python akan menyimpulkan tipe data yang kita masukkan. **copy** : parameter untuk copy data, secara default akan bernilai false.

Contoh:

```
| import pandas as pd
2 import numpy as np
3 data = np.array(['a','b','c','d'])
4 karakter = pd.Series(data)
5 print(karakter)
```

Pada contoh diatas kita membuat sebuah Series yang berisi karakter a,b,c, dan d. Jika kode diatas dijalankan maka hasilnya adalah:

```
print(karakter)
```

```
0 a
```

```
1 b
```

```
2 c
```

```
3 d
```

```
dtype: object
```

Series yang kita buat memiliki index default, lalu bagaimana jika kita ingin membuat custom index ? Yup benar, kita harus menambahkan parameter kedua yaitu index. Berikut contoh sederhananya:

```
1 import pandas as pd
2 import numpy as np
3 data = np.array(['a','b','c','d'])
4 karakter = pd.Series(data, index=['satu','dua','tiga','empat'])
5 print(karakter)
```

Jika kita jalankan, maka hasilnya adalah sebagai berikut:

```
print(karakter)
satu a
dua b
tiga c
empat d
dtype: object
```

Sekarang kita berhasil membuat custom index pada Series kita. Sekarang bagaimana caranya untuk mengakses series berdasarkan indexnya. Caranya sama seperti kita mengakses sebuah array.

```
1 print(karakter['satu'])
```

Jika dijalankan hasilnya adalah:

```
print(karakter['satu'])
a
```

Lalu apa bedanya Series dengan array biasa pada python? Seperti yang sudah kita bahas sebelumnya, pandas menyediakan beragam fungsi operasi untuk mengolah data. Contoh jika kita menggunakan series kita bisa mencari nilai max, min, dan mean secara langsung, bahkan kita juga bisa melakukan operasi perpangkatan pada nilai Series secara langsung. Biar gak bingung yuk mari kita coba.

Sekarang kita coba membuat sebuah Series baru yang akan berisi nilai float. Berikut adalah kode nya:

```
1 angka = pd.Series(np.array([60.5,70.45,80.00,90.64]));
```

Sekarang mari kita coba mencari tahu berapa nilai minimum, maximum dan nilai rata-ratanya. Caranya cukup mudah, kita tinggal menggunakan saja method yang sudah disediakan oleh pandas. Berikut contoh sederhana :

```
1 angka.max()
2 angka.min()
3 angka.mean()
```

Hasilnya adalah sebagai berikut:

```
angka.mean()
75.39749999999994
angka.max()
90.64000000000001
angka.min()
60.5
angka.mean()
```

```
75.39749999999999
```

Contoh yang lain kita akan mencoba memangkatkan setiap nilai pada Series angka dengan pangkat 2. Bagaimana caranya? yuk langsung kita coba.

```
| angka .pow(2 , axis=0)
```

Jika perintah diatas dijalankan, maka seluruh nilai yang ada pada Series angka akan dipangkatkan 2. Hasilnya adalah sebagai berikut:

```
angka.pow(2, axis=0)
0 3660.2500
1 4963.2025
2 6400.0000
3 8215.6096
dtype: float64
```

## 4.5 Matplotlib

Matplotlib adalah modul python untuk menggambar plot 2D dengan kualitas tinggi. matplotlib dapat digunakan dalam script python, interpreter python dan ipython, server, dan 6 GUI toolkit. matplotlib berusaha untuk membuat segalanya jadi mudah, dan yang tadinya seperti tidak menjadi mungkin untuk dilakukan. Dengan matplotlib, Anda dapat membuat plots, histograms, spectra, bar charts, errorcharts, scatterplots, dan masih banyak lagi.

Pembuat matplotlib bernama John D. Hunter yang pada 28 Agustus 2012 lalu meninggal dunia setelah bergelut dengan komplikasi kanker yang diidap beliau. Jasa beliau untuk Python Community sungguh sangat luar biasa (khususnya python untuk science).

Jika Anda merasa mendapatkan manfaat dari modul matplotlib yang sudah beliau buat, tidak ada salahnya untuk ikut melakukan kontribusi dengan melakukan donasi ke John Hunter Memorial Fund. Donasi ini nantinya akan diberikan langsung kepada keluarga yang sudah beliau tinggalkan, Miriam (istri), Clara, Ava dan Rahel (anak).

Matplotlib juga merupakan bagian dari paket inti SciPy dan ditawarkan di bawah lisensi BSD. Ini adalah library ilmiah Python populer yang digunakan untuk menghasilkan visualisasi yang sederhana dan kuat. Anda dapat menggunakan kerangka kerja Python untuk ilmu data untuk menghasilkan grafik, chart, histogram, dan bentuk dan gambar lain yang kreatif tanpa perlu khawatir menulis banyak baris kode. Sebagai contoh, mari kita lihat bagaimana perpustakaan Matplotlib dapat digunakan untuk membuat bar chart sederhana.

Matplotlib adalah library Python yang fokus pada visualisasi data seperti membuat plot grafik. Matplotlib pertama kali diciptakan oleh John D. Hunter dan sekarang telah dikelola oleh tim developer yang besar. Awalnya matplotlib dirancang untuk menghasilkan plot grafik yang sesuai pada publikasi jurnal atau artikel ilmiah. Matplotlib dapat digunakan dalam skrip Python, Python dan IPython shell, server aplikasi web, dan beberapa toolkit graphical user interface (GUI) lainnya.



**Gambar 4.10** Matplotlib

Visualisasi dari matplotlib adalah sebuah gambar grafik yang terdapat satu sumbu atau lebih. Setiap sumbu memiliki sumbu horizontal (x) dan sumbu vertikal (y), dan data yang direpresentasikan menjadi warna dan glyphs seperti marker (contohnya bentuk lingkaran) atau lines (garis) atau poligon.

Gambar di bawah menunjukkan bagian-bagian dari visualisasi matplotlib dibuat oleh Nicolas P. Rougier.

Hal yang penting dalam visualisasi data adalah penentuan warna, tekstur, dan style yang menarik untuk dilihat dan representatif terhadap data. Seorang Cartographer yaitu Jacques Bertin mengembangkan rekomendasi berikut untuk pemilihan informasi visual yang cocok, dan kita dapat menerapkannya menggunakan matplotlib.

Untuk memulai menggunakan matplotlib, lakukan import terlebih dahulu library matplotlib.pyplot as plt. Penggunaan as disini, artinya kita menggantikan pemanggilan fungsi pyplot pada matplotlib dengan prefix plt untuk proses berikutnya. Disini terdapat magic command % matplotlib inline, untuk pengaturan pada backend matplotlib agar setiap grafik ditampilkan secara inline, yaitu akan ditampilkan langsung pada cell notebook.

Line plot berguna untuk melacak perubahan pada periode waktu pendek dan panjang. Ketika terdapat perubahan kecil, line plot lebih baik dalam melakukan visualisasi dibandingkan grafik bar.

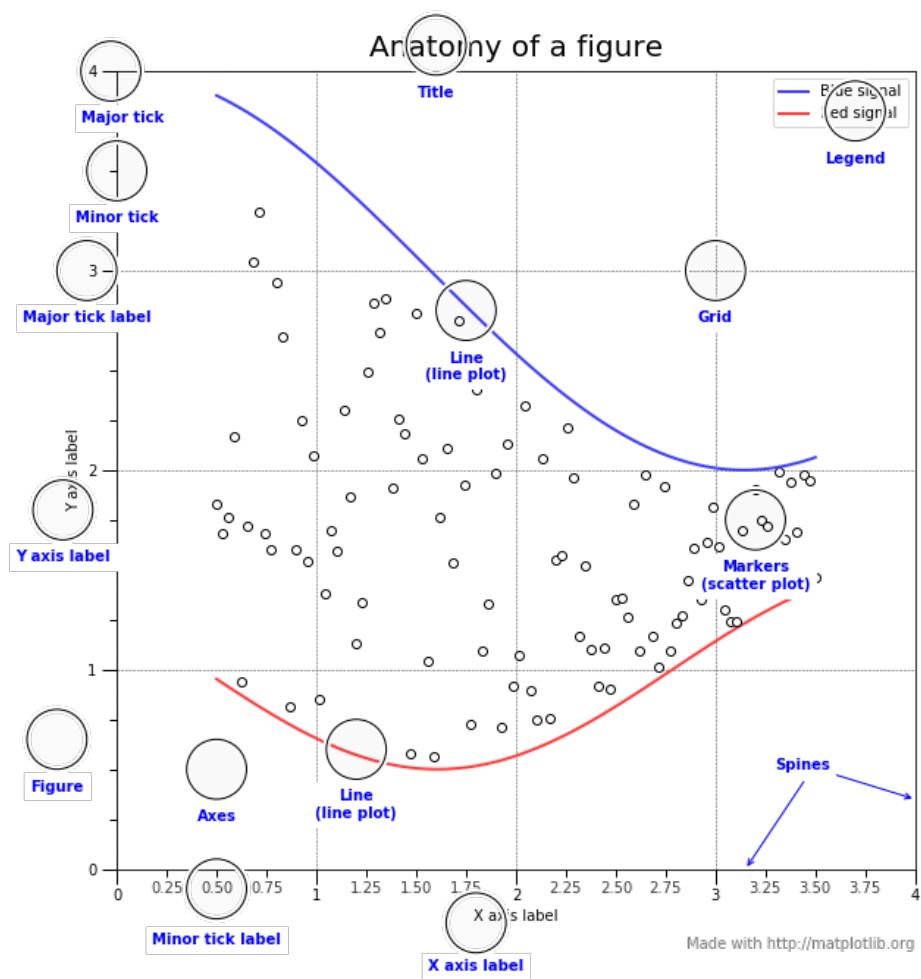
#### 4.5.1 Pemasangan Modul matplotlib

Jika Anda menggunakan sistem operasi ubuntu, Anda dapat memasang modul matplotlib dengan:

```
1 $ sudo apt-get install python-matplotlib
```

Karena tutorial ini nantinya akan banyak bersinggungan dengan modul numpy dan pandas, maka silakan dipasang juga kedua modul tersebut dengan:

```
1 $ sudo apt-get install python-numpy  
2 $ sudo apt-get install python-pandas  
3 $ sudo apt-get install python-scipy
```



**Gambar 4.11** Bagian Visualisasi Matplotlib

	Points	Lines	Areas	Best to show
Shape		<i>possible, but too weird to show</i>	<i>cartogram</i>	<i>qualitative differences</i>
Size			<i>cartogram</i>	<i>quantitative differences</i>
Color Hue				<i>qualitative differences</i>
Color Value				<i>quantitative differences</i>
Color Intensity				<i>qualitative differences</i>
Texture				<i>qualitative &amp; quantitative differences</i>

Gambar 4.12 Style Visual

## 4.5.2 Membuat Plot

Bagian ini nanti akan berisi contoh kode untuk menggambar plot dengan matplotlib.

## 4.5.3 Dasar-dasar Plot

Sesuai dengan namannya, fungsi plot berguna untuk menggambar garis atau penanda pada bidang gambar. Mari kita lihat dokumentasi untuk fungsi ini:

```

1 >>> import matplotlib.pyplot as plt
2 >>> help(plt.plot)
3 Help on function plot in module matplotlib.pyplot:
4
5     plot(*args, **kwargs)
6         Plot lines and/or markers to the
7             :class:`~matplotlib.axes.Axes` . *args* is a variable
8             length
9                 argument, allowing for multiple *x*, *y* pairs with an
10                optional format string. For example, each of the following is
11                legal::
12                    plot(x, y) # plot x and y using default line style and
13                    color
14                    plot(x, y, 'bo') # plot x and y using blue circle
15                    markers
16                    plot(y) # plot y using x as index array 0..N-1
17                    plot(y, 'r+') # ditto, but with red plusses
18
19 ....

```

Contoh menggambar plot:

```

1 >>> import matplotlib.pyplot as plt
2 >>> x = [1, 2, 3, 4]
3 >>> y = [13, 17, 19, 33]
4 >>> plt.plot(x, y)
5 [<matplotlib.lines.Line2D object at 0x94d1b4c>]
6 >>> plt.show()

```

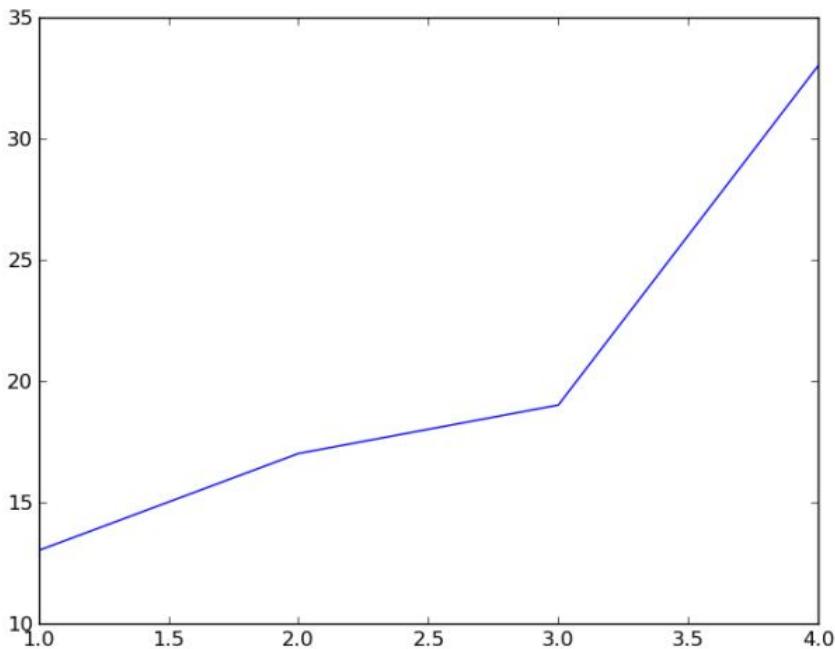
Tampilan:

Anda dapat menambahkan judul, label untuk garis horisontal dan vertikal dengan memanggil fungsi `xtitle`, `xlabel`, dan `ylabel` seperti pada contoh berikut:

```

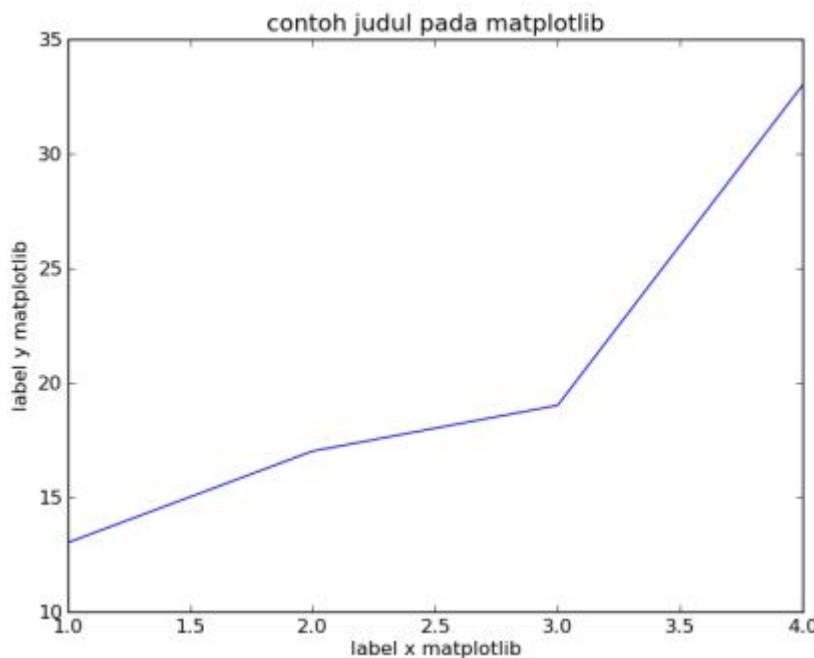
1 >>> import matplotlib.pyplot as plt
2 >>> plt.plot([1, 2, 3, 4], [13, 17, 19, 33])
3 [<matplotlib.lines.Line2D object at 0xa57a46c>]
4 >>> plt.title('contoh judul pada matplotlib')
5 <matplotlib.text.Text object at 0xa54fbec>
6 >>> plt.xlabel('label x matplotlib')
7 <matplotlib.text.Text object at 0xa54bc4c>
8 >>> plt.ylabel('label y matplotlib')

```



**Gambar 4.13** Penggambaran Plot

```
9 <matplotlib.text.Text object at 0xa5475ac>
10 >>> plt.show()
```



Gambar 4.14 Hasil Plot

Tabel ringkasan fungsi:

#### 4.5.4 Membaca Data dari Berkas

Pada contoh di atas, kita menggunakan tipe data list yang dideklarasikan langsung pada python. Bagaimana jika data kita berada pada sebuah berkas? Tentu data harus kita baca terlebih dahulu, sebelum ditampilkan dalam bentuk grafik.

**Tabel 4.1** Ringkasan Fungsi

Fungsi	Keterangan
plot	menggambar plot
title	memberi judul pada gambar plot
xlabel	memberi nama label untuk garis x
ylabel	memberi nama label untuk garis y
show	menampilkan gambar plot

#### 4.5.5 Menggunakan Fungsi Read (Python File I/O)

Sebagai contoh, kita memiliki data berikut pada berkas data.txt:

```
1366671909,5
1366671914,6
1366671920,3
1366671937,7
1366671942,1
1366671947,8
1366671955,4
1366671976,5
1366671981,7
1366671986,3
```

Selanjutnya kita baca berkas tersebut:

```
1 >>> open( data.txt ).read().strip()
2 1366671909,5\n1366671914,6\n1366671920,3\n1366671937,7\n1366671942
3 ,1
3 \n1366671947,8\n1366671955,4\n1366671976,5\n1366671981,7\n1366671986
3 ,3
```

Terlihat bahwa kita menggunakan 3 fungsi dari python untuk membaca berkas tersebut, mari kita baca penggalan dokumentasi dari masing-masing fungsi tersebut:

```
1 >>> help(open)
2 Help on built-in function open in module __builtin__:
3
4 open(...
5     open(name[, mode[, buffering]]) -> file object
6
7         Open a file using the file() type, returns a file object.
8         This is the
9             preferred way to open a file. See file.__doc__ for further
10            information.
11
10 >>> help(open( data.txt ).read)
11 Help on built-in function read:
```

```

13 read(...)
14     read([size]) -> read at most size bytes, returned as a string.
15
16     If the size argument is negative or omitted, read until EOF is
17     reached.
18     Notice that when in non-blocking mode, less data than what was
19     requested
20     may be returned, even if no size parameter was given.
21
22 >>> help(open( data.txt ).read().strip)
23 Help on built-in function strip:
24
25 strip(...)
26     S.strip([chars]) -> string or unicode
27
28     Return a copy of the string S with leading and trailing
29     whitespace removed.
30     If chars is given and not None, remove characters in chars
31     instead.
32     If chars is unicode, S will be converted to unicode before
33     stripping

```

Dari dokumentasi, kita ketahui bahwa open berfungsi untuk membuka berkas dengan nilai kembalian berupa file object, read berfungsi untuk membaca file object, dengan nilai kembalian (return) berupa string, sedangkan strip berguna untuk menghilangkan tanda whitespace di depan dan dibelakang string.

**Tabel 4.2 Fungsi**

Fungsi	Keterangan
open	membuka berkas
read	membaca file object
strip	menghilangkan whitespace di depan atau belakang string

Berikut urutan perintah yang kita jalankan untuk membaca dan menampilkan grafik plot dari data tersebut:

```

1 >>> with open( data.txt ) as f:
2 ... l = f.read().strip().split( '\n' )
3 ...
4 >>> l
5 [ 1366671909 ,5 , 1366671914 ,6 , 1366671920 ,3 ,
6 1366671937 ,7 ,
7 1366671942 ,1 , 1366671947 ,8 , 1366671955 ,4 ,
8 1366671976 ,5 ,
9 1366671981 ,7 , 1366671986 ,3 ]
10 >>> x = []
11 >>> y = []
12 >>> for i in l:
13 ... xny = i.split( ' ' )
14 ... x.append(int(xny[0]))
15 ... y.append(int(xny[1]))

```

```

14 ...
15 >>> x
16 [1366671909, 1366671914, 1366671920, 1366671937, 1366671942,
17 1366671947, 1366671955, 1366671976, 1366671981, 1366671986]
18 >>> y
19 [5, 6, 3, 7, 1, 8, 4, 5, 7, 3]
```

Pada kode di atas, kita menggunakan satu fungsi baru lagi, yakni split, mari kita lihat dokumentasi:

```

1 >>> help(aku, suka, python .split)
2 Help on built-in function split:
3
4 split(...)
5     S.split([sep [,maxsplit]]) -> list of strings
6
7     Return a list of the words in the string S, using sep as the
8     delimiter string. If maxsplit is given, at most maxsplit splits
9     are done. If sep is not specified or is None, any whitespace
10    string is a separator and empty strings are removed from the
11    result.
```

Fungsi split berguna untuk memecah string menjadi list yang berisi string.

**Tabel 4.3 Fungsi Split**

Fungsi	Keterangan
split	memecah string menjadi list of string

Setelah mendapatkan nilai x dan y, sekarang saatnya menampilkan data tersebut ke dalam grafik plot:

```

1 >>> plt.plot(x, y)
2 [<matplotlib.lines.Line2D object at 0xa7d32ec>]
3 >>> plt.title( matplotlib example title )
4 <matplotlib.text.Text object at 0xae946cc>
5 >>> plt.xlabel( matplot x label )
6 <matplotlib.text.Text object at 0xa5cf6ec>
7 >>> plt.ylabel( matplot y label )
8 <matplotlib.text.Text object at 0xa5e3e0c>
9 >>> plt.show()
```

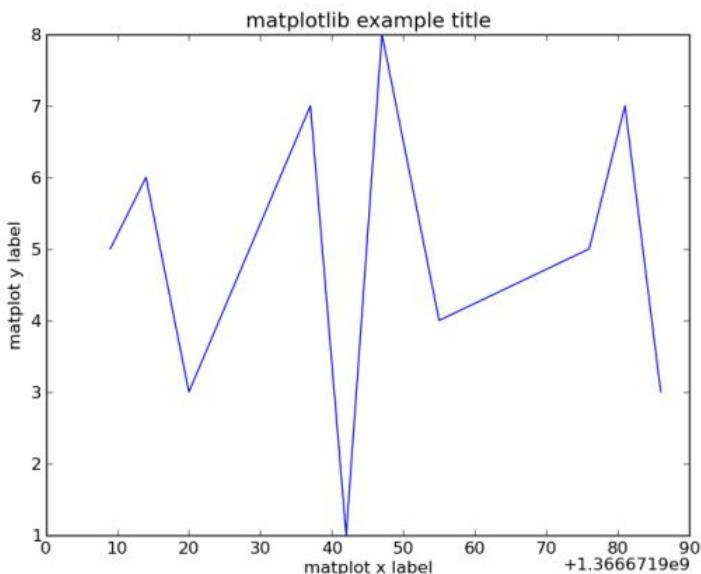
Tampilan:

#### 4.5.6 Menggunakan Modul numpy

Selain menggunakan cara di atas, kita juga dapat memanfaatkan salah satu fungsi dari modul numpy, yakni loadtxt yang sangat powerful untuk urusan membaca input dari berkas.

Sebagai contoh kita memiliki berkas data.csv dengan isi sebagai berikut:

2013-02-22,22



**Gambar 4.15** Penggambaran Plot

2013-03-28,11  
 2013-04-15,24  
 2013-05-03,45  
 2013-05-15,26  
 2013-06-20,12  
 2013-07-14,16  
 2013-08-04,23  
 2013-09-12,21  
 2013-10-02,31

Mari kita baca dan impor data tersebut ke dalam python menggunakan fungsi loadtxt. Ketik baris kode berikut dan simpan ke dalam berkas matplotlib\_5.py

```

1 #!/usr/bin/python
2 # matplotlib_5.py
3
4 import numpy as np
5 import matplotlib.pyplot as plt
6 import matplotlib.dates as mdates
7
8 def graph():
9     date, value = np.loadtxt(  data.csv , delimiter= , ,
10     converters = {0: mdates.strptime2num( %Y-%m-%d )} )
11
12

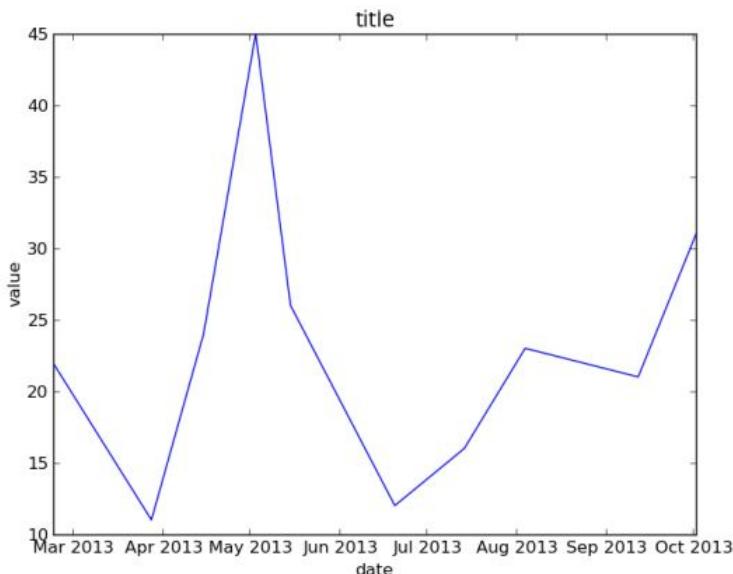
```

```

13 fig = plt.figure()
14 ax1 = fig.add_subplot(1,1,1, axisbg= white )
15 ax1.plot_date(x=date ,y=value ,fmt= '-' )
16 ax1.set_title( title )
17 ax1.set_ylabel( value )
18 ax1.set_xlabel( date )
19 plt.show()
20
21 if __name__ == __main__ :
22 graph()

```

Jalankan dan berikut hasilnya:



**Gambar 4.16** Penggambaran Plot

Sebenarnya penggunaan fungsi `graph` di atas kurang begitu pas, karena terlalu statis, seharusnya sebuah fungsi harus bersifat dinamis. Apabila Anda ingin lebih dinamis, kita dapat ubah fungsi di atas menjadi:

```

1 #!/usr/bin/python
2 # matplotlib-din.py
3
4 import numpy as np
5 import matplotlib.pyplot as plt
6 import matplotlib.dates as mdates
7
8 def graph(thefile , delimiter , title , ylabel , xlabel):
9     date , value = np.loadtxt(thefile , delimiter=delimiter , unpack=
10         True ,
11         converters = {0: mdates.strptime2num( '%Y-%m-%d' )})

```

```

11     )
12
13     fig = plt.figure()
14     ax1 = fig.add_subplot(1,1,1, axisbg= white)
15     ax1.plot_date(x=date, y=value, fmt= -)
16     ax1.set_title(title)
17     ax1.set_ylabel(ylabel)
18     ax1.set_xlabel(xlabel)
19     plt.show()
20
21 if __name__ == __main__:
22 graph(data.csv, , title, value, date)

```

Terlihat kita sudah berhasil melakukan abstraksi terhadap fungsi yang kita buat, sehingga sekarang kita dapat memanggil fungsi tersebut secara berulangulang hanya dengan mengganti parameter yang tersedia.

Namun perlu diingat, bahwa fungsi graph() di atas hanya berlaku untuk data csv dengan isi 2 kolom saja, jika Anda ingin lebih fleksibel lagi, Anda dapat melakukan abstraksi lagi sesuai keinginan.

**Tabel 4.4 Fungsi**

Fungsi	Keterangan
plot_date	Membuat plot dari data yang mengandung date
	(x, atau y, atau keduanya)
set_title	memberi judul figure
set_ylabel	memberi label y
set_xlabel	memberi label x

#### 4.5.7 Kustomisasi Tampilan Grafik Plot

Sekarang saatnya belajar membuat tampilan custom dari grafik kita.

#### 4.5.8 Atribut Background, Foreground dan Garis

Agar lebih mudah dipahami, mari kita simpan kode-kode di atas ke dalam sebuah berkas dan kemudian menambahkan beberapa fungsi baru untuk mengubah tampilan dari grafik yang kita buat. Simpan kode berikut ke dalam berkas matplotlib1.py

```

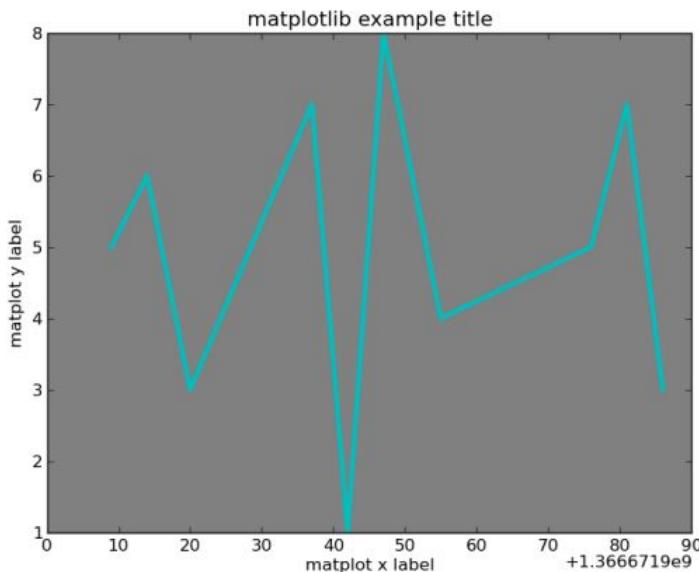
1#!/usr/bin/python
2
3import matplotlib.pyplot as plt
4with open(data.txt, r) as f:
5    l = f.read().strip().split( \n )
6
7x = []
8y = []

```

```
9
10 for i in l:
11     xny = i.split(   ,    )
12     x.append(int(xny[0]))
13     y.append(int(xny[1]))
14
15 # sebelum di plot, mari kita ubah tampilan
16 fig = plt.figure()
17 rect = fig.patch
18 rect.set_facecolor( "#31312e" )
19 ax1 = fig.add_subplot(1, 1, 1, axisbg= "grey" )
20 ax1.plot(x, y, "c", linewidth=3.3)
21
22 # plot dimulai
23 ax1.set_title( "matplotlib example title" )
24 ax1.set_xlabel( "matplotlib x label" )
25 ax1.set_ylabel( "matplotlib y label" )
26 plt.show()
```

Jalankan script dengan menulis perintah berikut pada Terminal:

```
$ python matplotlib_1.py
```



**Gambar 4.17** Penggambaran Plot

Terlihat warna dan ketebalan garis sekarang sudah berubah. Perlu diingat bahwa warna facecolor tidak bisa ikut ditampilkan, tapi kalau Anda menjalankan dari Terminal, pasti Anda bisa melihat warna di sekitar grafik berubah menjadi abu-abu gelap.

Seperti biasa, jika Anda masih bingung dengan fungsi, baik itu built-in maupun 3rd-party, Anda dapat memanggil dokumentasi menggunakan fungsi help seperti pada contoh sebelumnya.

**Tabel 4.5** Fungsi

Fungsi	Keterangan
int	mengubah string menjadi integer
figure	mengakses figure
patch	menyunting setting figure
set_facecolor	mengubah warna di sekitar grafik, di luar kotak grafik
add_subplot	mengakses subplot tertentu

#### 4.5.9 Atribut Warna

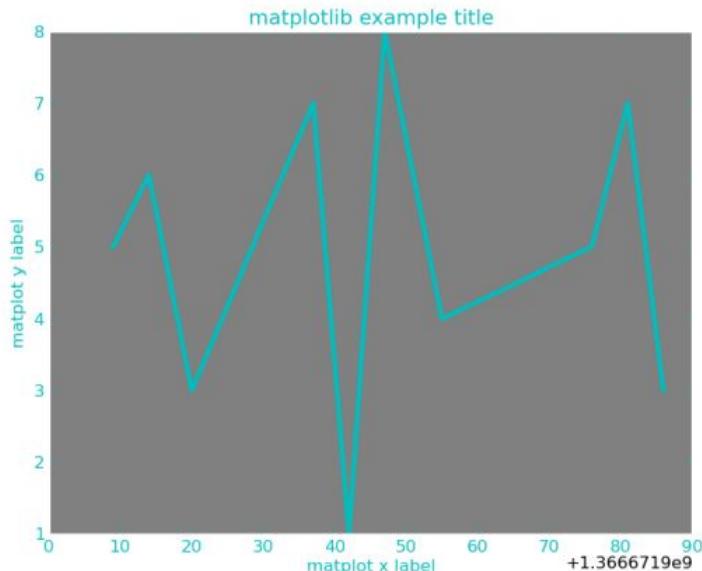
Mari kita salin kode pada berkas `matplotlib_1.py`, dan sunting menjadi seperti berikut:

```

1 #!/usr/bin/python
2
3 import matplotlib.pyplot as plt
4
5 with open('data.txt', 'r') as f:
6     l = f.read().strip().split('\n')
7
8 x = []
9 y = []
10
11 for i in l:
12     xny = i.split(',')
13     x.append(int(xny[0]))
14     y.append(int(xny[1]))
15
16 # sebelum di plot, mari kita ubah tampilan
17 fig = plt.figure()
18 rect = fig.patch
19 rect.set_facecolor('#31312e')
20
21 ax1 = fig.add_subplot(1, 1, 1, axisbg='grey')
22 ax1.plot(x, y, 'c', linewidth=3.3)
23
24 # customize start
25 ax1.tick_params(axis='x', color='c')
26 ax1.tick_params(axis='y', color='c')
27
28 ax1.spines['bottom'].set_color('w')
29 ax1.spines['top'].set_color('w')
30 ax1.spines['left'].set_color('w')
31 ax1.spines['right'].set_color('w')
32 ax1.yaxis.label.set_color('c')
```

```
33 ax1.xaxis.label.set_color( c )
34 # customize end
35
36 # plot dimulai
37 ax1.set_title( matplotlib example title )
38 ax1.set_xlabel( matplotlib x label )
39 ax1.set_ylabel( matplotlib y label )
40 plt.show()
```

Simpan ke dalam berkas `matplotlib_2.py` dan jalankan. Berikut hasil tampilan layarnya:



**Gambar 4.18** Penggambaran Plot

**Tabel 4.6** Fungsi

Fungsi	Keterangan
tick_params	mengubah warna huruf tick
spines	mengubah warna border grafik
label.set_color	mengubah warna huruf pada label

#### 4.5.10 Multiple Graph Same Figure

Contoh berikut ini menunjukkan bagaimana kita dapat membuat 2 atau lebih grafik dalam satu figure. Sebelumnya, mari kita buat berkas baru, beri nama data2.txt dan isikan data berikut:

```
1366671909,2  
1366671914,1  
1366671920,4  
1366671937,5  
1366671942,2  
1366671947,1  
1366671955,6  
1366671976,2  
1366671981,1  
1366671986,2
```

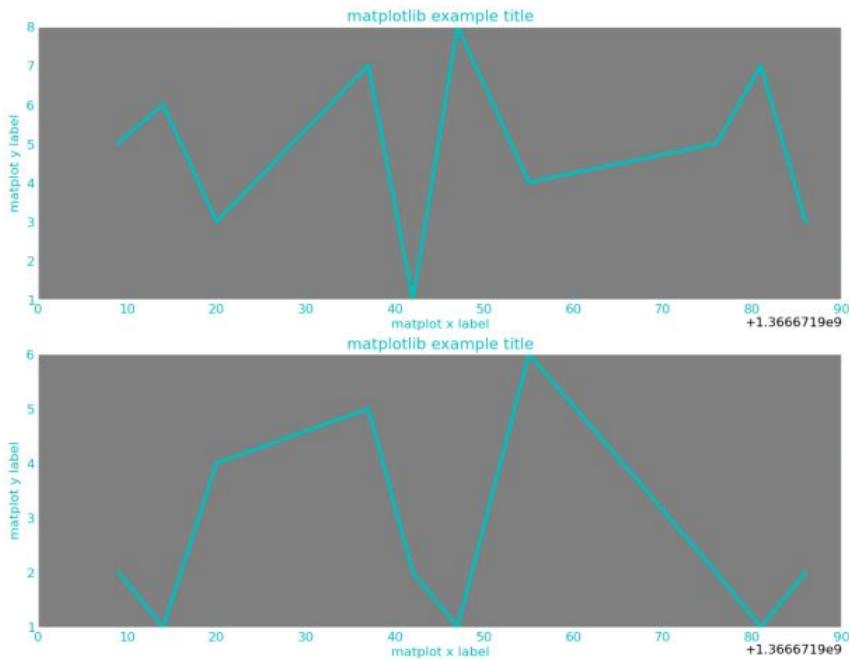
Kemudian buat berkas python baru, misal matplotlib\_3.py, dan isikan baris kode berikut:

```
1 #!/usr/bin/python  
2 import matplotlib.pyplot as plt  
3  
4 # baca berkas data.txt  
5 with open('data.txt', 'r') as f:  
6     l = f.read().strip().split('\n')  
7  
8 x = []  
9 y = []  
10  
11 # baca berkas data2.txt  
12 with open('data2.txt', 'r') as f2:  
13     l2 = f2.read().strip().split('\n')  
14  
15 x2 = []  
16 y2 = []  
17  
18 for i in l:  
19     xny = i.split(',')  
20     x.append(int(xny[0]))  
21     y.append(int(xny[1]))  
22  
23 for i in l2:  
24     xny2 = i.split(',')  
25     x2.append(int(xny2[0]))  
26     y2.append(int(xny2[1]))  
27  
28 # sebelum di plot, mari kita ubah tampilan  
29 fig = plt.figure()  
30 rect = fig.patch  
31 rect.set_facecolor('#31312e')  
32  
33 # set atribut untuk graph1  
34 ax1 = fig.add_subplot(2, 1, 1, axisbg='grey')
```

```
35 ax1.plot(x, y, c , linewidth=3.3)
36
37 # customize start
38
39 # mengubah warna angka pada x dan y
40 ax1.tick_params(axis= x , colors= c )
41 ax1.tick_params(axis= y , colors= c )
42
43 # mengubah border tepi grafik
44 ax1.spines[ bottom ].set_color( w )
45 ax1.spines[ top ].set_color( w )
46 ax1.spines[ left ].set_color( w )
47 ax1.spines[ right ].set_color( w )
48
49 # mengubah warna label x dan y
50 ax1.yaxis.label.set_color( c )
51 ax1.xaxis.label.set_color( c )
52 ax1.set_title( matplotlib example title , color= c )
53 ax1.set_xlabel( matplot x label )
54 ax1.set_ylabel( matplot y label )
55
56 # customize end
57
58 # set atribut untuk graph2
59 ax2 = fig.add_subplot(2, 1, 2, axisbg= grey ) # height x width
      and the chart #
60 ax2.plot(x2, y2, c , linewidth=3.3)
61
62 # customize start
63
64 # mengubah warna angka pada x dan y
65 ax2.tick_params(axis= x , colors= c )
66 ax2.tick_params(axis= y , colors= c )
67
68 # mengubah border tepi grafik
69 ax2.spines[ bottom ].set_color( w )
70 ax2.spines[ top ].set_color( w )
71 ax2.spines[ left ].set_color( w )
72 ax2.spines[ right ].set_color( w )
73
74 # mengubah warna label x dan y
75 ax2.yaxis.label.set_color( c )
76 ax2.xaxis.label.set_color( c )
77
78 # customize end
79
80 # plot dimulai
81 ax2.set_title( matplotlib example title , color= c )
82 ax2.set_xlabel( matplot x label )
83 ax2.set_ylabel( matplot y label )
84
85 # show the graph!
86 plt.show()
```

Dan berikut tampilannya:

Tabel parameter dari fungsi add\_subplot:



**Gambar 4.19** Penggambaran Plot

**Tabel 4.7** Parameter

Parameter	Keterangan
2, 1, 1	h=2, w=1, chart number 1
2, 1, 2	h=2, w=1, chart #2

Yang agak membingungkan di sini adalah 3 parameter pertama dari fungsi sub-plot. Mari kita buat satu lagi grafik agar lebih mudah dalam memahaminya.

Buat berkas baru, misal matplotlib\_4.py, dan isikan baris berikut:

```
1 #!/usr/bin/python
2 import matplotlib.pyplot as plt
3
4 # baca berkas data.txt
5 with open( data.txt , r ) as f:
6     l = f.read().strip().split( '\n' )
7
8 x = []
9 y = []
10
11 # baca berkas data2.txt
12 with open( data2.txt , r ) as f2:
13     l2 = f2.read().strip().split( '\n' )
14
15 x2 = []
16 y2 = []
17
18 for i in l:
19     xny = i.split( ' ' )
20     x.append( int(xny[0]) )
21     y.append( int(xny[1]) )
22
23 for i in l2:
24     xny2 = i.split( ' ' )
25     x2.append( int(xny2[0]) )
26     y2.append( int(xny2[1]) )
27
28 # sebelum di plot, mari kita ubah tampilan
29 fig = plt.figure()
30 rect = fig.patch
31 rect.set_facecolor( "#31312e" )
32
33 # set atribut untuk graph1
34 ax1 = fig.add_subplot(2, 2, 1, axisbg= "grey" )
35 ax1.plot(x, y, c= "red", linewidth=3.3)
36
37 # customize start
38
39 # mengubah warna angka pada x dan y
40 ax1.tick_params(axis= "x" , colors= "red" )
41 ax1.tick_params(axis= "y" , colors= "red" )
42
43 # mengubah border tepi grafik
44 ax1.spines[ "bottom" ].set_color( "white" )
45 ax1.spines[ "top" ].set_color( "white" )
46 ax1.spines[ "left" ].set_color( "white" )
47 ax1.spines[ "right" ].set_color( "white" )
48
49 # mengubah warna label x dan y
50 ax1.yaxis.label.set_color( "red" )
51 ax1.xaxis.label.set_color( "red" )
```

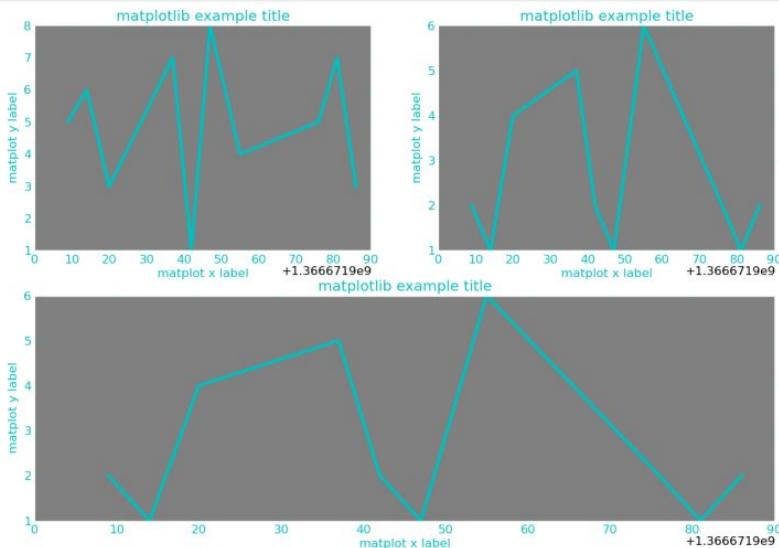
```
52 ax1.set_title( matplotlib example title , color= c )
53 ax1.set_xlabel( matplot x label )
54 ax1.set_ylabel( matplot y label )
55
56 # customize end
57
58 # set atribut untuk graph2
59 ax2 = fig.add_subplot(2, 2, 2, axisbg= grey ) # 2x2 grid for the
      chart number 2
60 ax2.plot(x2, y2, c , linewidth=3.3)
61
62 # customize start
63
64 # mengubah warna angka pada x dan y
65 ax2.tick_params(axis= x , colors= c )
66 ax2.tick_params(axis= y , colors= c )
67
68 # mengubah border tepi grafik
69 ax2.spines[ bottom ].set_color( w )
70 ax2.spines[ top ].set_color( w )
71 ax2.spines[ left ].set_color( w )
72 ax2.spines[ right ].set_color( w )
73
74 # mengubah warna label x dan y
75 ax2.yaxis.label.set_color( c )
76 ax2.xaxis.label.set_color( c )
77
78 # customize end
79
80 # plot dimulai
81 ax2.set_title( matplotlib example title , color= c )
82 ax2.set_xlabel( matplot x label )
83 ax2.set_ylabel( matplot y label )
84
85 # set atribut untuk graph3
86 ax2 = fig.add_subplot(2, 1, 2, axisbg= grey ) # height x width
      and the chart #
87 ax2.plot(x2, y2, c , linewidth=3.3)
88
89 # customize start
90
91 # mengubah warna angka pada x dan y
92 ax2.tick_params(axis= x , colors= c )
93 ax2.tick_params(axis= y , colors= c )
94
95 # mengubah border tepi grafik
96 ax2.spines[ bottom ].set_color( w )
97 ax2.spines[ top ].set_color( w )
98 ax2.spines[ left ].set_color( w )
99 ax2.spines[ right ].set_color( w )
100
101 # mengubah warna label x dan y
102 ax2.yaxis.label.set_color( c )
103 ax2.xaxis.label.set_color( c )
104
105 # customize end
```

```

106
107 # plot dimulai
108 ax2.set_title( matplotlib example title , color= c )
109 ax2.set_xlabel( matplot x label )
110 ax2.set_ylabel( matplot y label )
111
112 plt.show()

```

Dan berikut tampilannya:



**Gambar 4.20** Penggambaran Plot

Berikut rangkuman paremeter dari fungsi `add_subplot` beserta keterangannya:

**Tabel 4.8** Parameter

Parameter	Keterangan
2, 2, 1	h=2, w=2, chart number 1
2, 2, 2	h=2, w=2, chart #2
2, 1, 2	h=2, w=1, chart #2

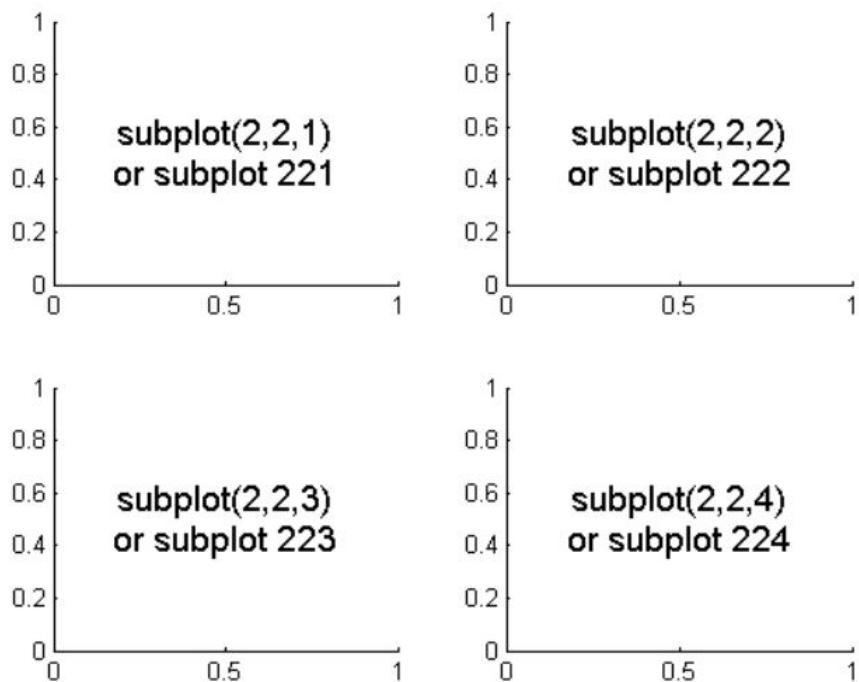
Untuk dapat lebih menjelaskan mengenai `add_subplot`, mari kita lihat gambar berikut:

Kode python untuk membuat grafik di atas:

```

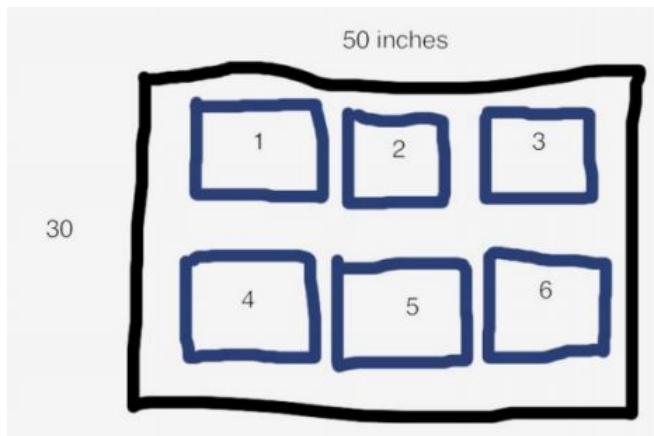
1 fig1.add_subplot(221) #top left
2 fig2.add_subplot(222) #top right
3 fig3.add_subplot(223) #bottom left
4 fig4.add_subplot(224) #bottom right

```



**Gambar 4.21** Penggambaran Plot

Dan berikut satu gambar lagi:



**Gambar 4.22 Penggambaran Plot**

Kode python untuk membuat grafik di atas:

```

1 fig1.add_subplot(221) #top left
2 fig2.add_subplot(222) #top center
3 fig3.add_subplot(223) #top right
4 fig4.add_subplot(224) #bottom left
5 fig5.add_subplot(225) #bottom center
6 fig6.add_subplot(226) #bottom right

```

#### 4.5.11 Histogram

Berikut contoh menggambar histogram:

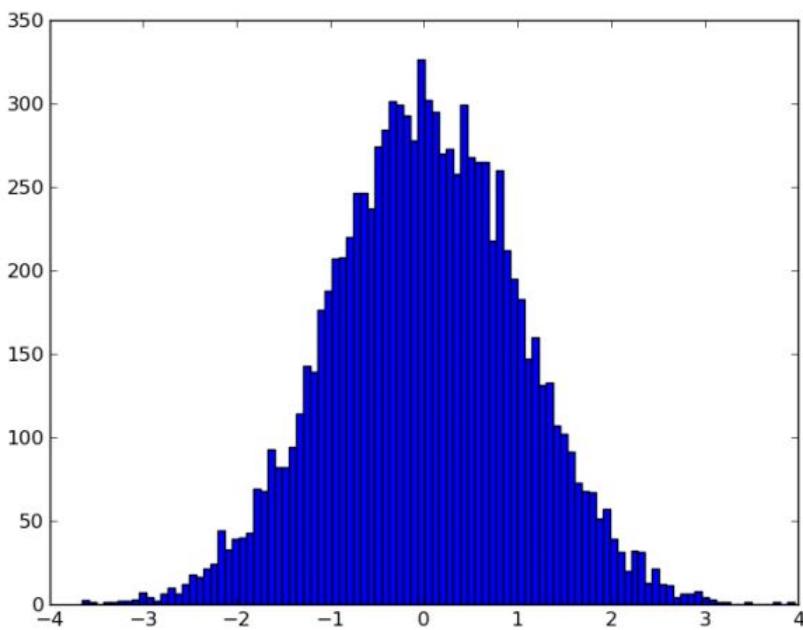
```

1 >>> import matplotlib.pyplot as plt
2 >>> import numpy as np
3 >>> x = np.random.randn(10000)
4 >>> plt.hist(x, 100)
5 (array([-2, 1, 0, 1, 1, 2, 2, 3, 7, 4, 2, 6, 10,
6 6, 12, 18, 16, 21, 24, 44, 33, 39, 40, 43, 69, 68,
7 93, 82, 82, 94, 114, 143, 139, 176, 188, 207, 208, 220, 246,
8 246, 237, 274, 284, 301, 299, 293, 278, 326, 302, 295, 270, 273,
9 258, 299, 268, 265, 265, 218, 260, 212, 195, 183, 147, 160, 131,
10 133, 107, 102, 91, 73, 68, 67, 51, 57, 39, 31, 20, 32,
11 31, 13, 21, 12, 11, 4, 6, 6, 8, 4, 3, 1, 1,
12 0, 0, 1, 0, 0, 1, 0, 1]), array([
13 -3.64645879, -3.57037555, -3.49429231, -3.41820907, -3.34212584,
14 -3.2660426 , -3.18995936, -3.11387613, -3.03779289, -2.96170965,
15 -2.88562641, -2.80954318, -2.73345994, -2.6573767 , -2.58129347,
16 -2.50521023, -2.42912699, -2.35304376, -2.27696052, -2.20087728,
17 -2.12479404, -2.04871081, -1.97262757, -1.89654433, -1.8204611 ,
18 -1.74437786, -1.66829462, -1.59221138, -1.51612815, -1.44004491,
19 -1.36396167, -1.28787844, -1.2117952 , -1.13571196, -1.05962873,

```

```
20 -0.98354549, -0.90746225, -0.83137901, -0.75529578, -0.67921254,
21 -0.6031293 , -0.52704607, -0.45096283, -0.37487959, -0.29879635,
22 -0.22271312, -0.14662988, -0.07054664, 0.00553659, 0.08161983,
23 0.15770307, 0.2337863 , 0.30986954, 0.38595278, 0.46203602,
24 0.53811925, 0.61420249, 0.69028573, 0.76636896, 0.8424522 ,
25 0.91853544, 0.99461867, 1.07070191, 1.14678515, 1.22286839,
26 1.29895162, 1.37503486, 1.4511181 , 1.52720133, 1.60328457,
27 1.67936781, 1.75545105, 1.83153428, 1.90761752, 1.98370076,
28 2.05978399, 2.13586723, 2.21195047, 2.2880337 , 2.36411694,
29 2.44020018, 2.51628342, 2.59236665, 2.66844989, 2.74453313,
30 2.82061636, 2.8966996 , 2.97278284, 3.04886608, 3.12494931,
31 3.20103255, 3.27711579, 3.35319902, 3.42928226, 3.5053655 ,
32 3.58144873, 3.65753197, 3.73361521, 3.80969845, 3.88578168,
33 3.96186492]), <a list of 100 Patch objects>
34
35 >>> plt.show()
```

Dan berikut hasilnya:



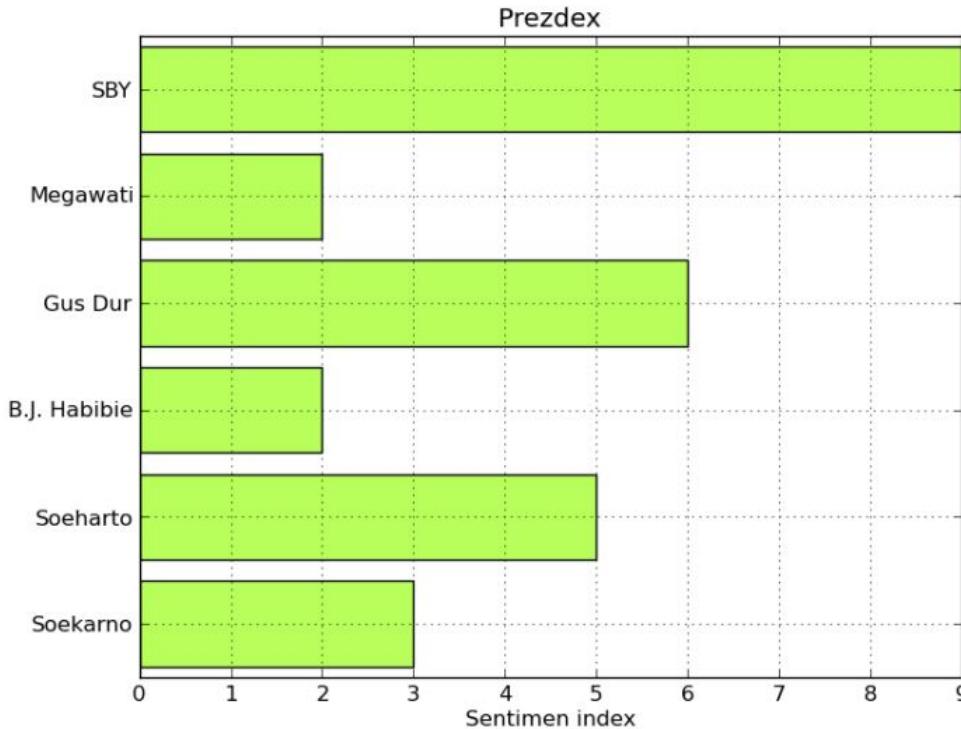
**Gambar 4.23** Penggambaran Plot

## 4.6 Bar Charts

Bagian ini akan membahas tentang bar charts. Berikut ini contoh sederhananya:

```
>>> from pylab import *
```

```
2 >>> pos = arange(6) + .5
3 >>> barh(pos, (3, 5, 2, 1, 2, 9), align= center , color= "#b8ff5c")
4 <Container object of 6 artists>
5 >>> yticks(pos, ( Soekarno , Soeharto , B.J. Habibie ,
6 ( Gus Dur , Megawati , SBY ))
6 ([<matplotlib.axis.YTick object at 0xac7396c>, .. list of 6 Text
7 yticklabel objects>)
7 >>> xlabel( Sentimen index )
8 <matplotlib.text.Text object at 0xac7356c>
9 >>> ylabel( candidate )
10 <matplotlib.text.Text object at 0xac774ec>
11 >>> title( Prezdex )
12 <matplotlib.text.Text object at 0xac0f98c>
13 >>> grid(True)
14 >>> show()
```



**Gambar 4.24** Penggambaran Plot

barh sendiri menerima beberapa parameter seperti height (dalam contoh di atas pos), width (lebar bar), align (secara default nilainya edge, namun bisa kita ganti center seperti pada contoh di atas, dan color (warna dari bar)).

**Tabel 4.9** Fungsi

Parameter	Keterangan
arange	membuat deret angka
barh	membuat bar horisontal
yticks	keterangan setiap titik pada y
grid	memunculkan garis bantu

Contoh satu lagi menggunakan data di atas, namun kali ini kita tidak akan menggunakan parameter align (menggunakan default).

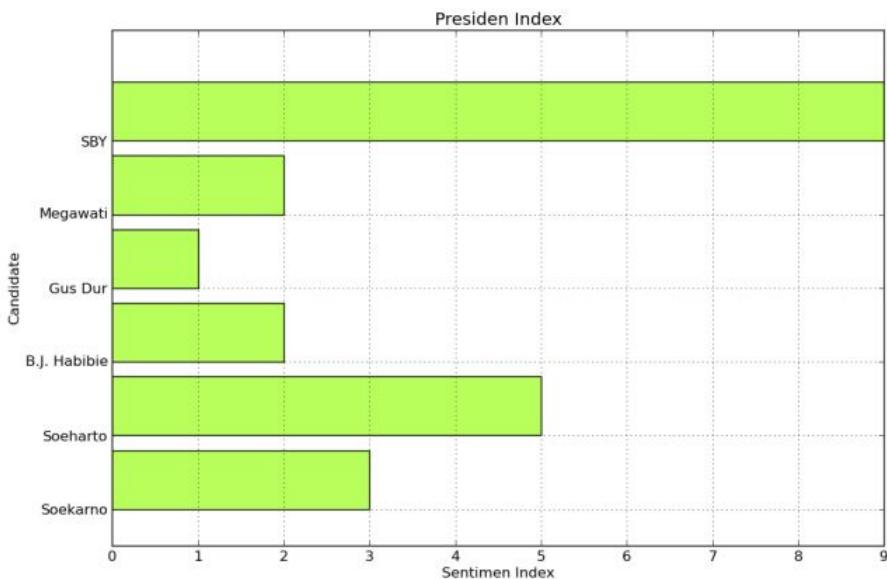
```

1 >>> from pylab import *
2 >>> pos = arange(6) + .5
3 >>> barh(pos, (3, 5, 2, 1, 2, 9), color= "#b8ff5c")
4 <Container object of 6 artists>
5 >>> yticks(pos, ("Soekarno", "Soeharto", "B.J. Habibie",
6 "Gus Dur", "Megawati", "SBY"))
6 ([<matplotlib.axis.YTick object at 0xaa2f4cc> .. list of 6 Text
7 yticklabel objects>)
7 >>> xlabel("Sentimen Index")
8 <matplotlib.text.Text object at 0xaaleecc>
9 >>> ylabel("Candidate")
10 <matplotlib.text.Text object at 0xaa2f66c>
11 >>> title("Presiden Index")
12 <matplotlib.text.Text object at 0xacbde2c>
13 >>> grid(True)
14 >>> show()

```

Terlihat bahwa sekarang posisi bar menempel pada edge dari angka 0.5. Contoh berikutnya, misalkan kita memiliki data dalam berkas csv seperti berikut:

Nixon;-1.4013  
 Bloomberg;-0.7981  
 Rand Paul;-0.6930  
 Hillary Clinton;-0.6216  
 Gillibrand;-0.3818  
 Biden;-0.0999  
 Palin;-0.0979  
 Cuccinelli;0.1184  
 Ted Cruz;0.1354  
 Paul Ryan;0.2464  
 John Kerry;0.2477  
 Patrick;0.2974  
 Jindal;0.4317  
 Gingrich;0.5241  
 Santorum;0.5854  
 Christie;0.6704  
 Warren;0.7332  
 Sebelius;0.8494



**Gambar 4.25 Penggambaran Plot**

Ayotte;1.0317

Booker;1.0882

Mari kita olah dan tampilkan dalam bentuk bar horisontal

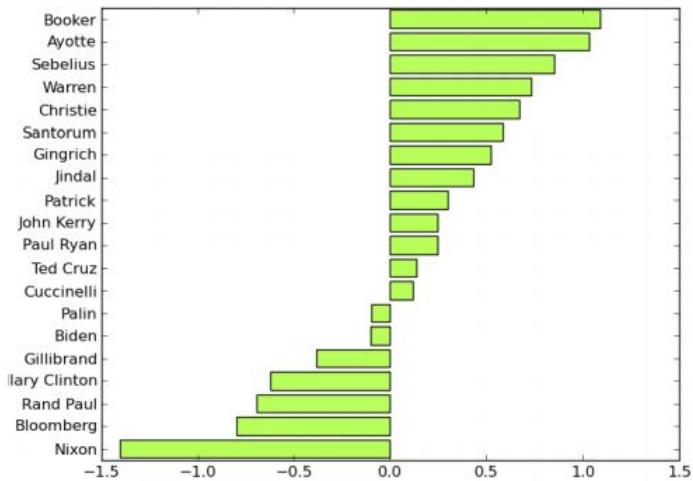
```

1 >>> from pylab import *
2 >>> name = []
3 >>> value = []
4 >>> with open( presdex.csv ) as f:
5 ... listline = f.read().split( '\n' )
6 ...
7 >>> for line in listline:
8 ... split = line.split( ';' )
9 ... name.append( split[0] )
10 ... value.append( float( split[1] ) )
11 ...
12 >>> pos = arange( len(name) ) + 0.5
13 >>> barh( pos, value, align= center, color= "#b8ff5c" )
14 <Container object of 20 artists>
15 >>> yticks( pos, name )
16 [<matplotlib.axis.YTick object at 0xadb112c> ... <a list of 20 Text
     ticklabel objects>]
17 >>> show()

```

Dan berikut hasilnya:

Terlihat dari grafik, ada satu nama yang terpotong yakni calon presiden Hillary Clinton karena terlalu panjang, kita dapat menyesuaikan agar nama tersebut masuk ke dalam grafik. Sekalian kita akan lengkapil grafik di atas dengan title, xlabel, ylabel, grid dan pernak-pernik lain.



**Gambar 4.26** Penggambaran Plot

Simpan kode berikut ke dalam berkas python

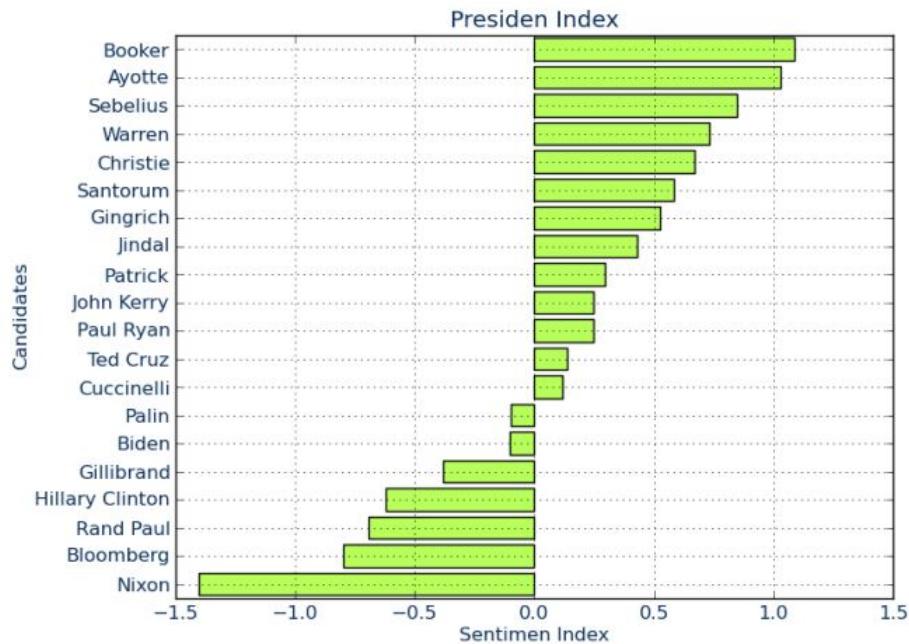
```

1 #!/usr/bin/python
2 # matplot_6.py
3
4 from pylab import *
5
6 name = []
7 value = []
8
9 # membaca berkas dan menyimpan tiap baris ke dalam object
10 with open( presdex.csv ) as f:
11     listline = f.read().split( '\n' )
12
13 # split tiap baris dengan ;
14 for line in listline:
15     split = line.split( ';' )
16     name.append( split[0] )
17     value.append( float( split[1] ) )
18 # sesuaikan jarak kiri biar semua nama dan atribut tampil
19 subplots_adjust(left=0.20, right=0.97)
20 tick_params(axis= 'x' , colors= '#072 b 57 ')
21 tick_params(axis= 'y' , colors= '#072 b 57 ')
22 # beri label
23 xlabel( Sentimen Index , color= '#072 b 57 ')
24 ylabel( Candidates , color= '#072 b 57 ')
25 title( Presiden Index , color= '#072 b 57 ')
26 pos = arange(len(name)) + 0.5
27 barh(pos, value, align= center , color= '#b 8 ff5c ')
28 yticks(pos, name)
29 grid(True)

```

```
30 show()
```

Jalankan dan berikut hasilnya:



**Gambar 4.27** Penggambaran Plot

**Tabel 4.10** Fungsi

Fungsi	Keterangan
subplots_adjust	menyesuaikan jarak tepi grafik, left, right, top, bottom
tick_params	parameter axis x dan y

#### 4.6.1 Grafik 3D

Sebelumnya, pastikan Anda sudah memasang modul mpl\_toolkits pada sistem operasi Anda. Jika Anda pengguna Ubuntu, cukup ketikkan:

```
$ sudo apt-get install python-mpltoolkits.basemap
```

#### 4.6.2 3D Line

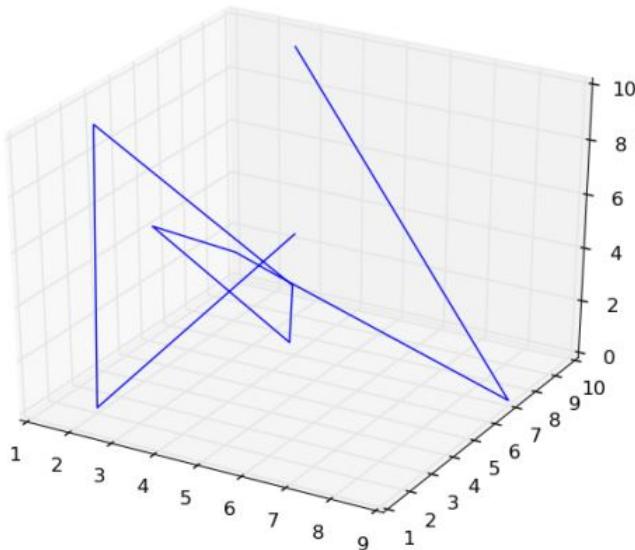
Berikut contoh script python untuk membuat grafik 3d:

```

1 >>> from mpl_toolkits.mplot3d import axes3d
2 >>> import matplotlib.pyplot as plt
3 >>> from random import randint
4 >>> X, Y, Z = [randint(0, 10) for i in range(10)],
5 ... [randint(0, 10) for i in range(10)],
6 ... [randint(0, 10) for i in range(10)]
7 >>> fig = plt.figure()
8 >>> ax = fig.add_subplot(111, projection= '3d')
9 >>> ax.plot_wireframe(X, Y, Z)
10 <mpl_toolkits.mplot3d.art3d.Line3DCollection object at 0x9c738cc>
11 >>> plt.show()

```

Dan berikut hasilnya:



**Gambar 4.28** Penggambaran Plot

Anda dapat memperbesar grafik di atas dengan menekan klik kanan perangkat mouse, kemudian gerakkan maju (zoom out) atau mundur (zoom in).

#### 4.6.3 3D Scatter Plot

Berikut ini contoh satu lagi untuk membuat grafik 3d scatter plot:

```

1 >>> from mpl_toolkits.mplot3d import axes3d
2 >>> import matplotlib.pyplot as plt
3 >>> from random import randint
4 >>> fig = plt.figure()

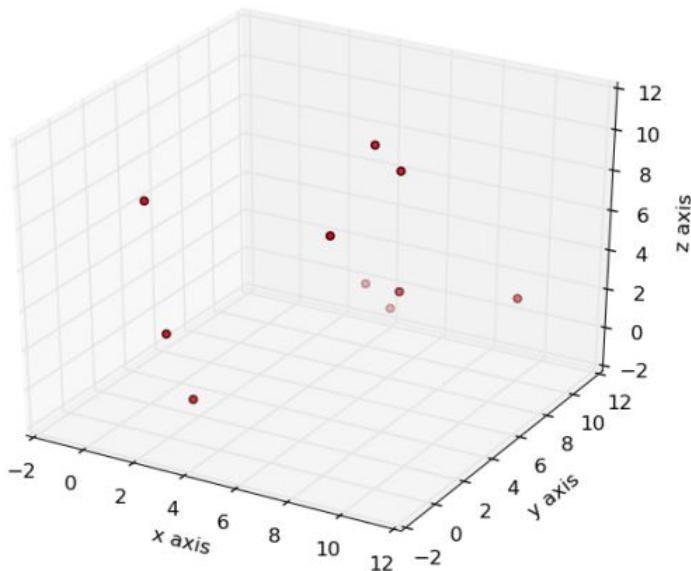
```

```

5 >>> ax = fig.add_subplot(111, projection= '3d' )
6 >>> X = [ randint(0, 10) for i in range(10) ]
7 >>> Y = [ randint(0, 10) for i in range(10) ]
8 >>> Z = [ randint(0, 10) for i in range(10) ]
9 >>> ax.scatter(X, Y, Z, c= 'r' , marker= 'o' )
10 <mpl_toolkits.mplot3d.art3d.Patch3DCollection object at 0x9c9488c>
11 >>> ax.set_xlabel( 'x axis' )
12 <matplotlib.text.Text object at 0x9a1206c>
13 >>> ax.set_ylabel( 'y axis' )
14 <matplotlib.text.Text object at 0x9b118cc>
15 >>> ax.set_zlabel( 'z axis' )
16 <matplotlib.text.Text object at 0x9b1cc8c>
17 >>> plt.show()

```

Dan berikut hasilnya:



**Gambar 4.29** Penggambaran Plot

Bila diperhatikan, titik akan berwarna semakin gelap ketika jarak titik dengan pengamat semakin dekat, begitu pun sebaliknya.

**Tabel 4.11** Fungsi

Fungsi	Keterangan
scatter	membuat grafik scatter
randint	membuat bilangan random integer

#### 4.6.4 3D Scatter Plot with Multiple Datasets

Pada contoh sebelumnya kita hanya menggunakan satu sumber data, berikut ini contoh penggunaan scatter plot menggunakan lebih dari satu sumber data. Ketikkan kode berikut ke dalam python interpreter Anda:

```

1 >>> from mpl_toolkits.mplot3d import axes3d
2 >>> import matplotlib.pyplot as plt
3 >>> from random import randint
4 >>> fig = plt.figure()
5 >>> ax = fig.add_subplot(111, projection= '3d' )
6 >>> X = [randint(0, 10) for i in range(10)]
7 >>> Y = [randint(1, 13) for i in range(10)]
8 >>> Z = [randint(2, 12) for i in range(10)]
9 >>> Xs = [randint(-10, 0) for i in range(10)]
10 >>> Ys = [randint(-14, 9) for i in range(10)]
11 >>> Zs = [randint(-8, 12) for i in range(10)]
12 >>> ax.scatter(X, Y, Z, c= 'r' , marker= 'o' )
13 <mpl_toolkits.mplot3d.art3d.Patch3DCollection object at 0x98edf2c>
14 >>> ax.scatter(Xs, Ys, Zs, c= 'b' , marker= '^' )
15 <mpl_toolkits.mplot3d.art3d.Patch3DCollection object at 0x9b8950c>
16 >>> ax.set_xlabel( 'x axis' )
17 <matplotlib.text.Text object at 0x98c8a8c>
18 >>> ax.set_ylabel( 'y axis' )
19 <matplotlib.text.Text object at 0x98d3bac>
20 >>> ax.set_zlabel( 'z axis' )
21 <matplotlib.text.Text object at 0x98dbb2c>
22 >>> plt.show()
```

Hasil:

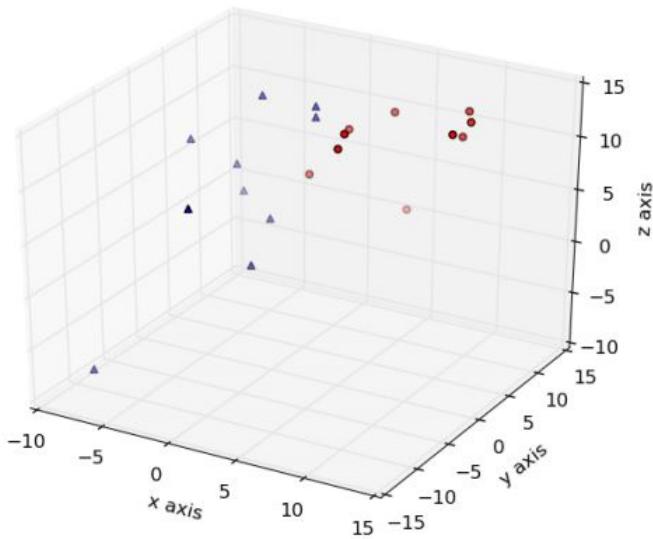
#### 4.6.5 3D Bar Charts

Berikut ini contoh membuat 3d bar charts:

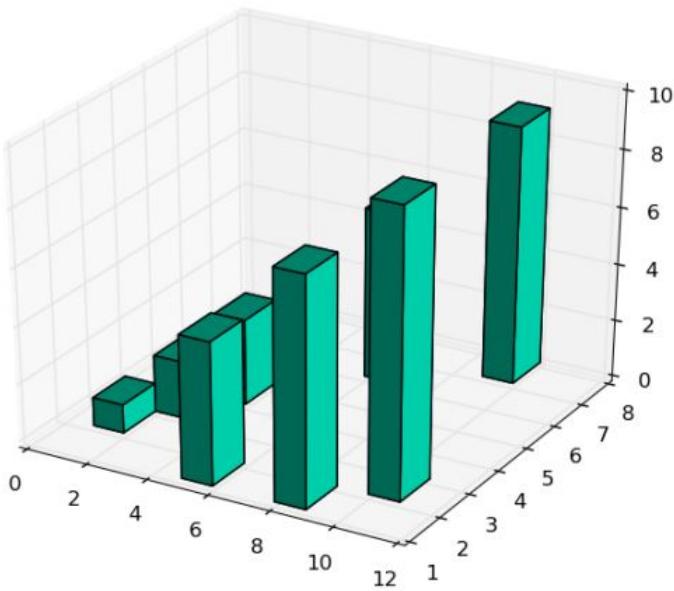
```

1 >>> from mpl_toolkits.mplot3d import axes3d
2 >>> import matplotlib.pyplot as plt
3 >>> import numpy as np
4 >>> fig = plt.figure()
5 >>> ax1 = fig.add_subplot(111, projection= '3d' )
6 >>> xpos = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
7 >>> ypos = [2, 3, 4, 5, 1, 6, 2, 1, 7, 2]
8 >>> np.zeros(10)
9 array([ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.])
10 >>> np.ones(10)
11 array([ 1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.])
12 >>> zpos = np.zeros(10)
13 >>> dx = np.ones(10)
14 >>> dy = np.ones(10)
15 >>> dz = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
16 >>> ax1.bar3d(xpos, ypos, zpos, dx, dy, dz, color= '#00ceaa' )
17 >>> plt.show()
```

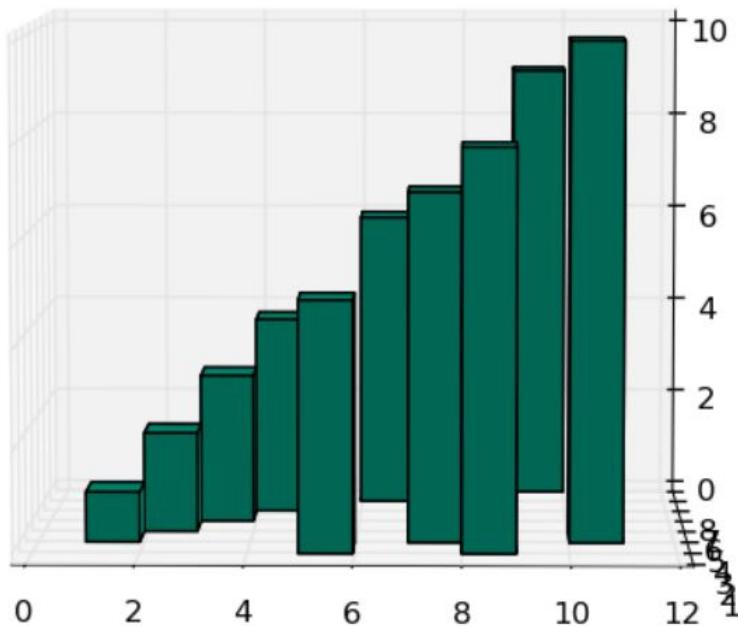
Hasil:



Gambar 4.30 Penggambaran Plot



Gambar 4.31 Penggambaran Plot



Gambar 4.32 Penggambaran Plot

Tabel 4.12 Fungsi

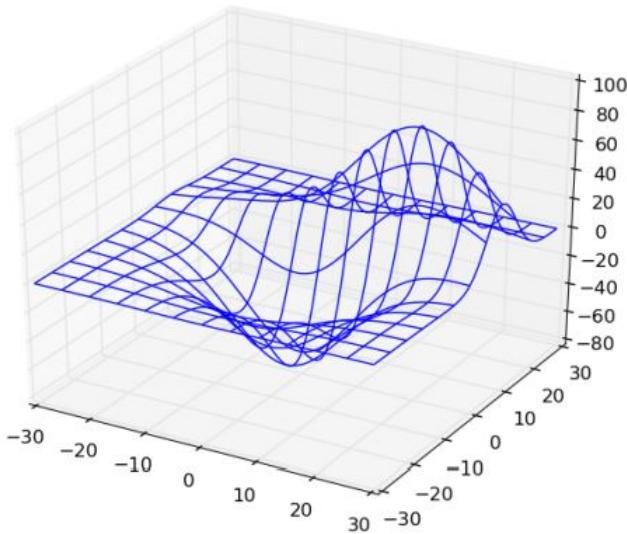
Fungsi	Keterangan
np.zeros	membuat ndarray dengan nilai 0.
np.ones	membuat ndarray dengan nilai 1.

#### 4.6.6 3D Plane Wire Frame

Berikutnya, kita akan mencoba membuat grafik 3d plane wire frame menggunakan matplotlib. Perhatikan kode berikut:

```
1 >>> from mpl_toolkits.mplot3d import axes3d
2 >>> import matplotlib.pyplot as plt
3 >>> import numpy as np
4 >>> fig = plt.figure()
5 >>> ax = fig.add_subplot(111, projection='3d')
6 >>> x, y, z = axes3d.get_test_data(0.05)
7 >>> ax.plot_wireframe(x, y, z, rstride=10, cstride=10)
8 <mpl_toolkits.mplot3d.art3d.Line3DCollection object at 0xa98048c>
9 >>> plt.show()
```

Hasil:



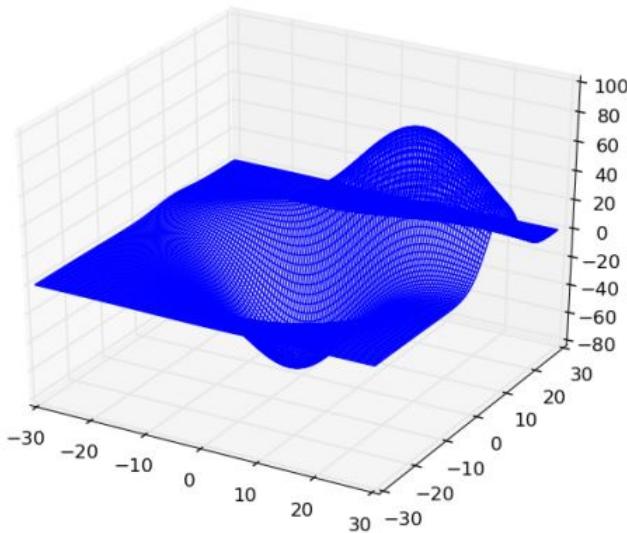
**Gambar 4.33** Penggambaran Plot

Mari kita ubah nilai parameter rstride dan cstride dari 10 ke 1, dan mari kita lihat hasilnya:

#### 4.6.7 Plot Scatter Dengan plt.plot

Di bagian sebelumnya, kami melihat plt.plot / ax.plot untuk membuat plot garis. Ternyata fungsi yang sama ini bisa menghasilkan plot scatter juga:

```
1 x = np.linspace(0, 10, 30)
```



**Gambar 4.34** Penggambaran Plot

```

2 y = np.sin(x)
3
4 plt.plot(x, y, 'o', color='black');
```

Argumen ketiga dalam pemanggilan fungsi adalah karakter yang mewakili jenis simbol yang digunakan untuk plotting. Anda juga dapat menentukan opsi seperti '-' , '-' untuk mengontrol model garis, model penanda memiliki kumpulan kode string singkatnya sendiri. Daftar lengkap simbol yang tersedia dapat dilihat dalam dokumentasi plt.plot, atau dalam dokumentasi online Matplotlib. Sebagian besar kemungkinannya cukup intuitif, dan kami akan menampilkan sejumlah yang lebih umum di sini:

```

1 rng = np.random.RandomState(0)
2 for marker in ['o', '.', ',', 'x', '+', 'v', '^', '<', '>', 's', 'd']
3     plt.plot(rng.rand(5), rng.rand(5), marker,
4               label="marker='{0}'".format(marker))
5 plt.legend(numpoints=1)
6 plt.xlim(0, 1.8);
```

Untuk lebih banyak kemungkinan, kode-kode karakter ini dapat digunakan bersama dengan kode garis dan warna untuk memplot titik-titik bersama dengan garis yang menghubungkannya:

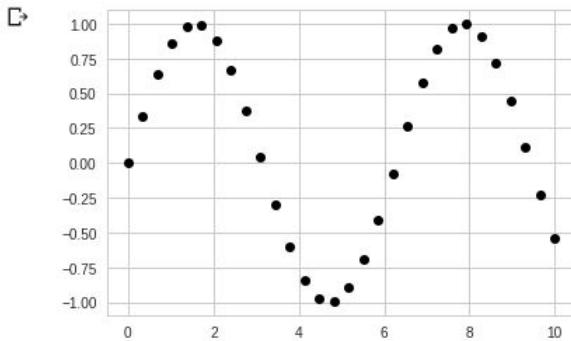
```

1 plt.plot(x, y, '-ok');
```

```
%matplotlib inline
import matplotlib.pyplot as plt
plt.style.use('seaborn-whitegrid')
import numpy as np

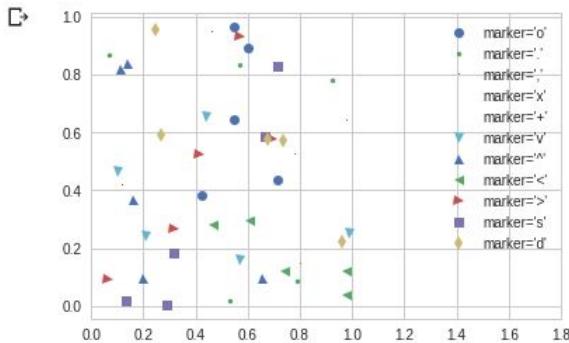
[2] x = np.linspace(0, 10, 30)
y = np.sin(x)

plt.plot(x, y, 'o', color='black');
```

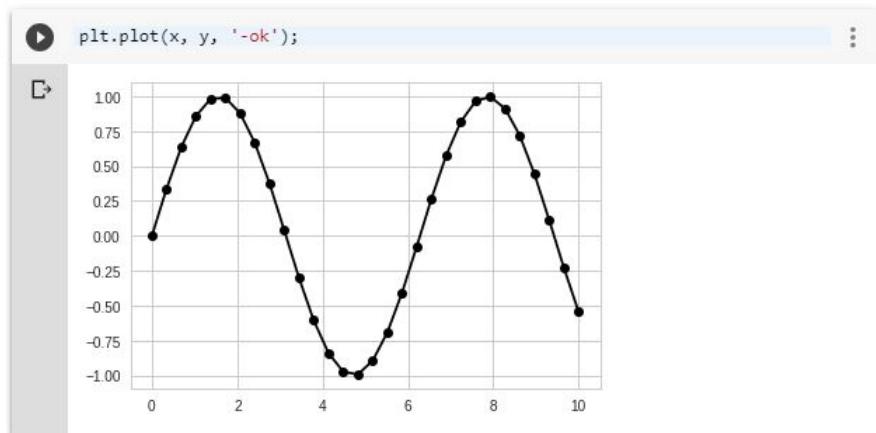


Gambar 4.35 Penggambaran Plot

```
rng = np.random.RandomState(0)
for marker in ['o', '.', ',', 'x', '+', 'v', '^', '<', '>', 's', 'd']:
    plt.plot(rng.rand(5), rng.rand(5), marker,
              label="marker='{0}'".format(marker))
plt.legend(numpoints=1)
plt.xlim(0, 1.8);
```



Gambar 4.36 Penggambaran Plot

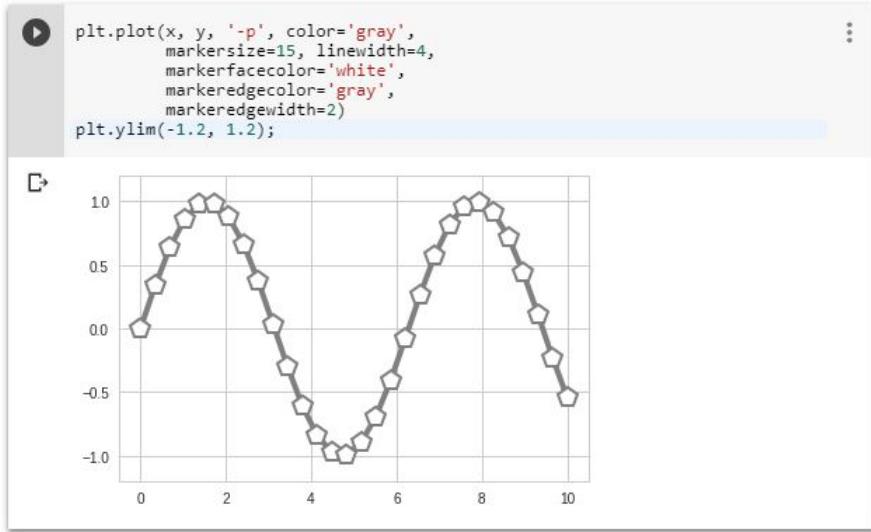


Gambar 4.37 Penggambaran Plot

Argumen kata kunci tambahan untuk plt.plot menentukan berbagai properti garis dan marker:

```
1 plt.plot(x, y, '-p', color='gray',
2           markersize=15, linewidth=4,
3           markerfacecolor='white',
4           markeredgecolor='gray',
5           markeredgewidth=2)
6 plt.ylim(-1.2, 1.2);
```

Fleksibilitas jenis ini dalam fungsi plt.plot memungkinkan berbagai kemungkinan opsi visualisasi. Untuk keterangan lengkap tentang opsi yang tersedia, lihat dokumentasi plt.plot.



Gambar 4.38 Penggambaran Plot

