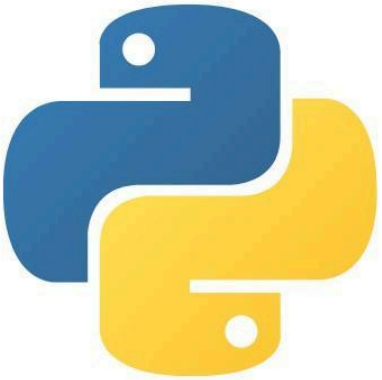
PROGRAM STUDI
TEKNIK INFORMATIKA

AKREDITASI

Kampus Merdeka
INDONESIA JAYA



MODUL PRAKTIKUM DATA MINING

Teknik Informatika
Universitas Pelita Bangsa

DAFTAR ISI

MODUL 1
(MINGGU PERTAMA)

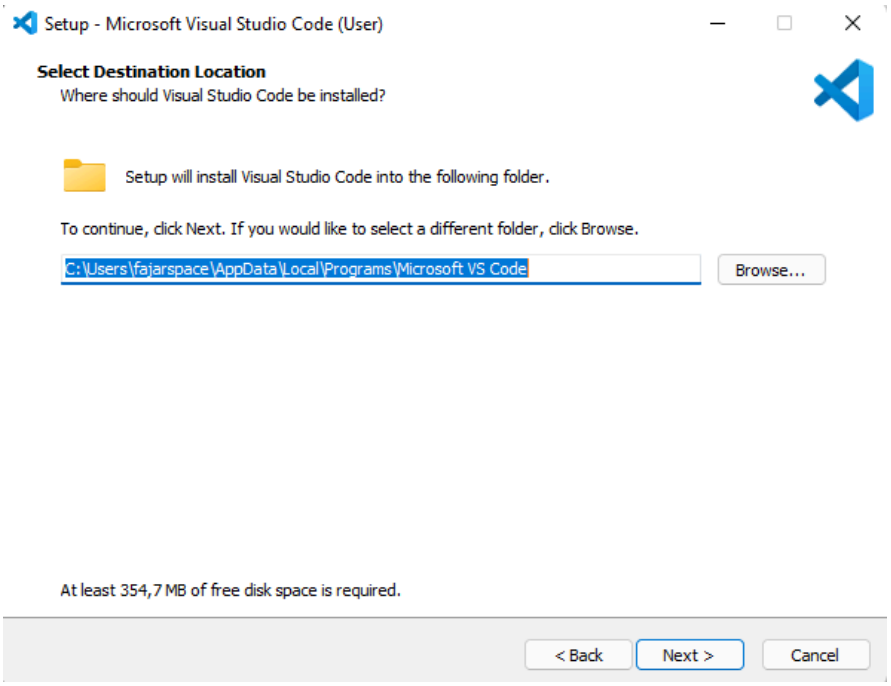
PRAKTIKUM 1

JUDUL : Instalasi VSCode, Python, Library

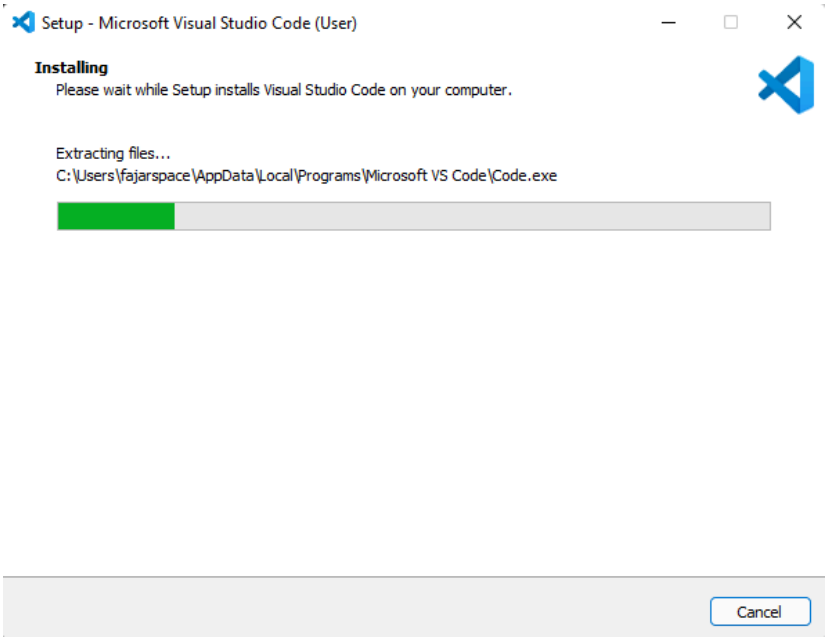
DESKRIPSI : Tujuan dari praktikum ini adalah

INSTALASI

1. Download VSCode di https://code.visualstudio.com/
2. lalu diklik, Kemudian klik next saja.
3. Sesuaikan dengan direktori masing-masing



4. Kemudian klik next saja, lalu tunggu hingga selesai


Setup - Microsoft Visual Studio Code (User)
Installing
Please wait while Setup installs Visual Studio Code on your computer.
Extracting files...
C:\Users\fajarspace\AppData\Local\Programs\Microsoft VS Code\Code.exe
Cancel

INSTALASI PYTHON
TUJUAN

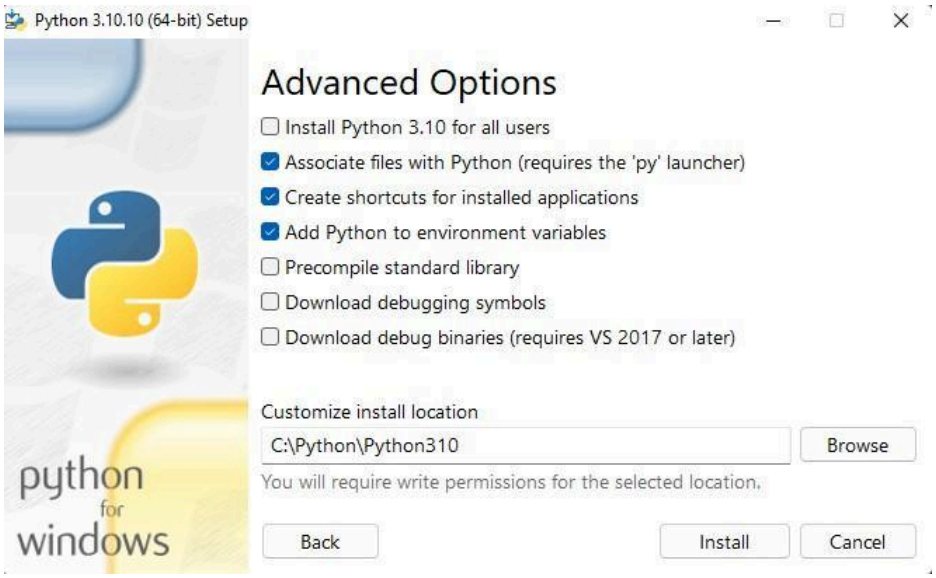
1. Mahasiswa Mampu Melakukan Instalasi Python
2. Mahasiswa Dapat Mengerti Dasar Pemrograman Python

INSTALASI

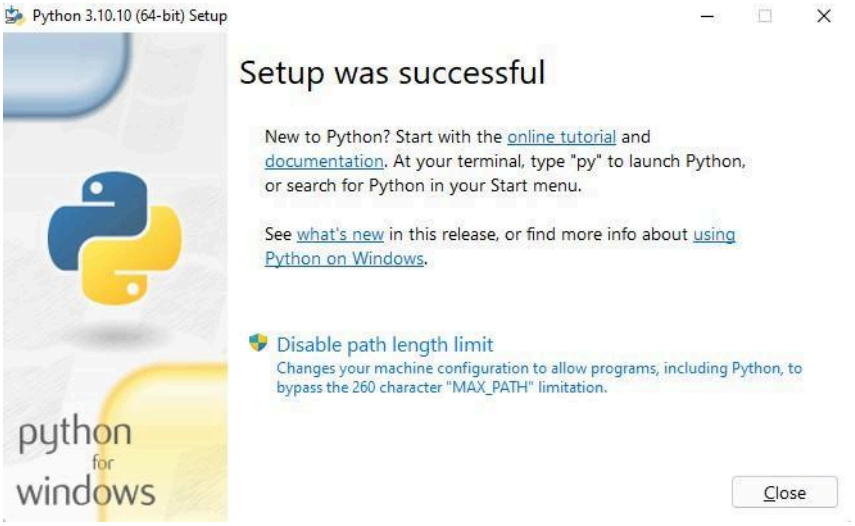
1. Download Python di https://www.python.org/downloads/windows/
   Pilih versi Python sesuai dengan tipe sistem operasi Anda (32/64

   bit)

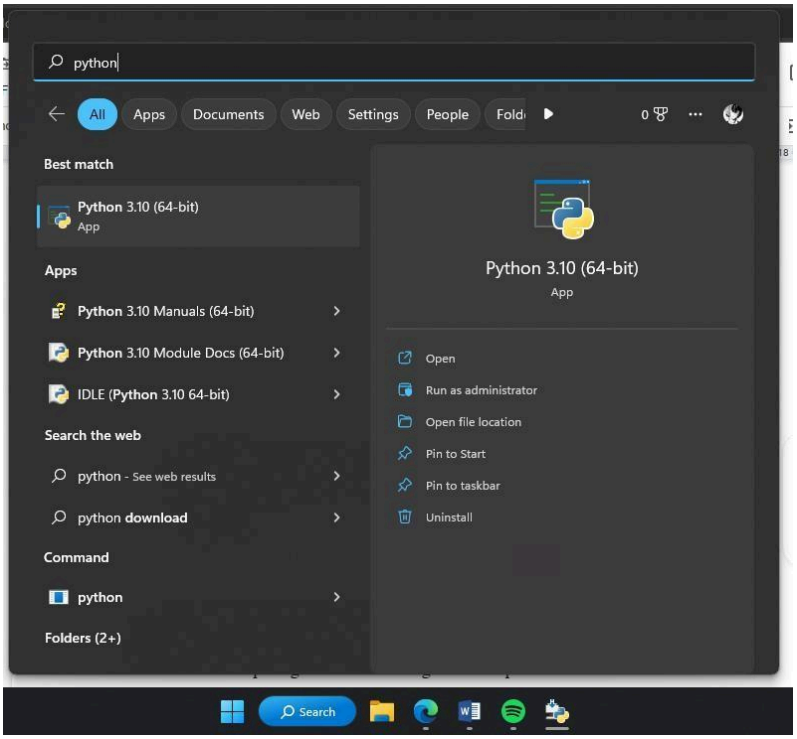
   lalu diklik. Setelah muncul kotak dialog seperti dibawah, pilih **Customize installation**.
2. Kemudian klik next saja.
3. Pilih direktori dimana Anda ingin menginstal Python. Lokasi yang saya pilih **C:\Python\Python310** agar mudah diakses, Centang **'Add python to environment variables'** Kemudian klik Install.



4. Tunggu hingga selesai
5. Setelah sukses, klik Close

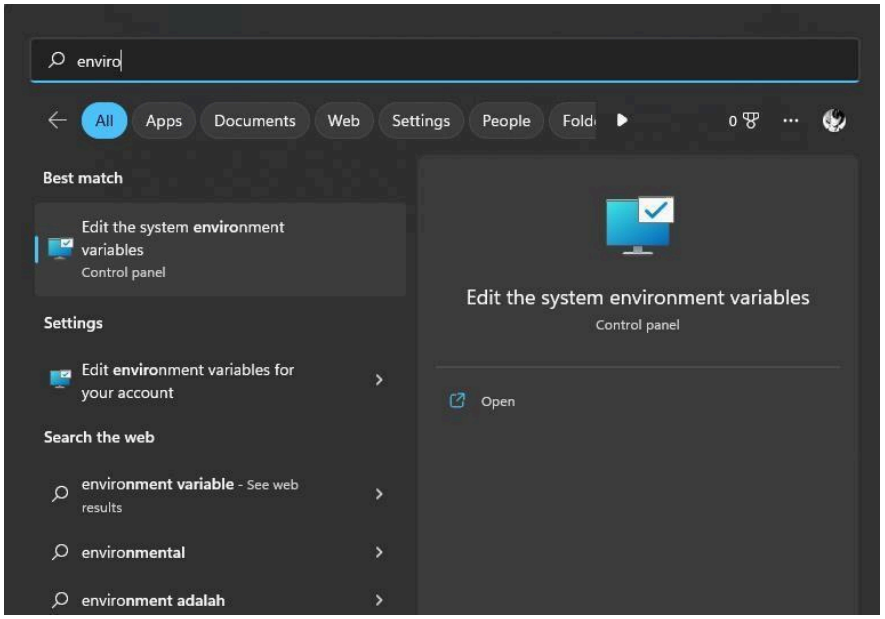


6. Anda bisa mengecek apakah Python sudah diinstal dengan melakukan searching di Windows.

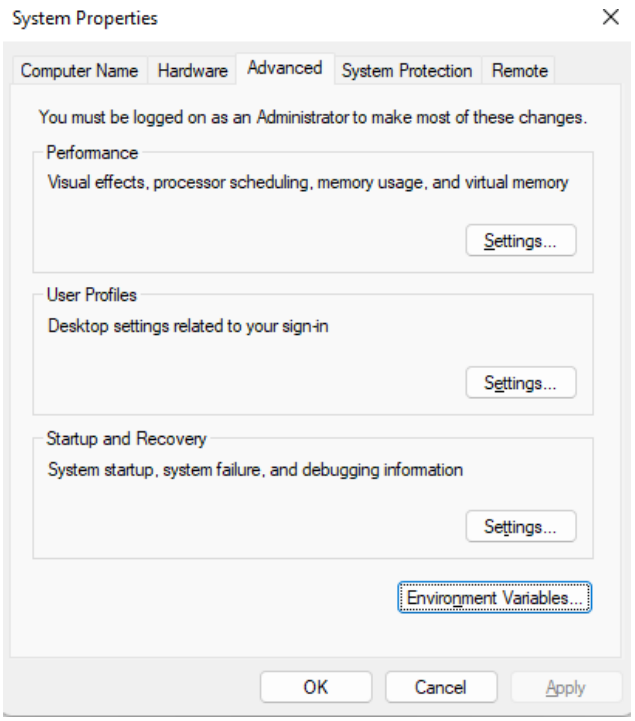


7. Kemudian copy alamat direktori Python ***C:\python\python310***

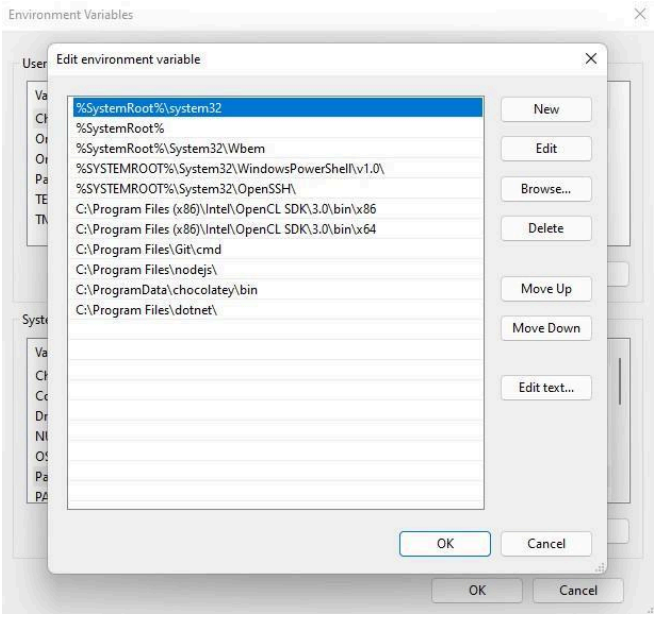
8. Buka System Environment Variables untuk men-setting Path. Buka **Control Panel** lalu searching dengan keyword atau bisa langsung search melalui **Search Box Windows** seperti gambar dibawah agar lebih cepat.



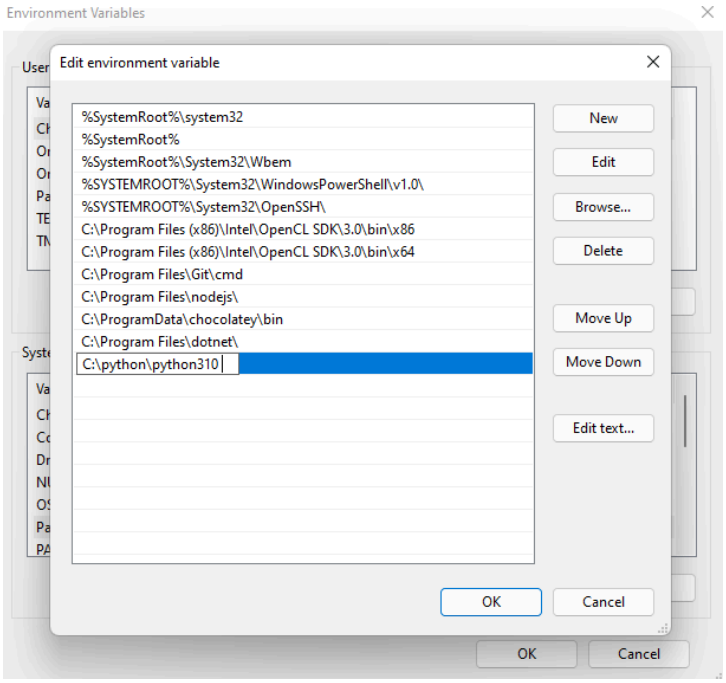
Setelah muncul kotak dialog seperti gambar dibawah klik **Environment Variables.**



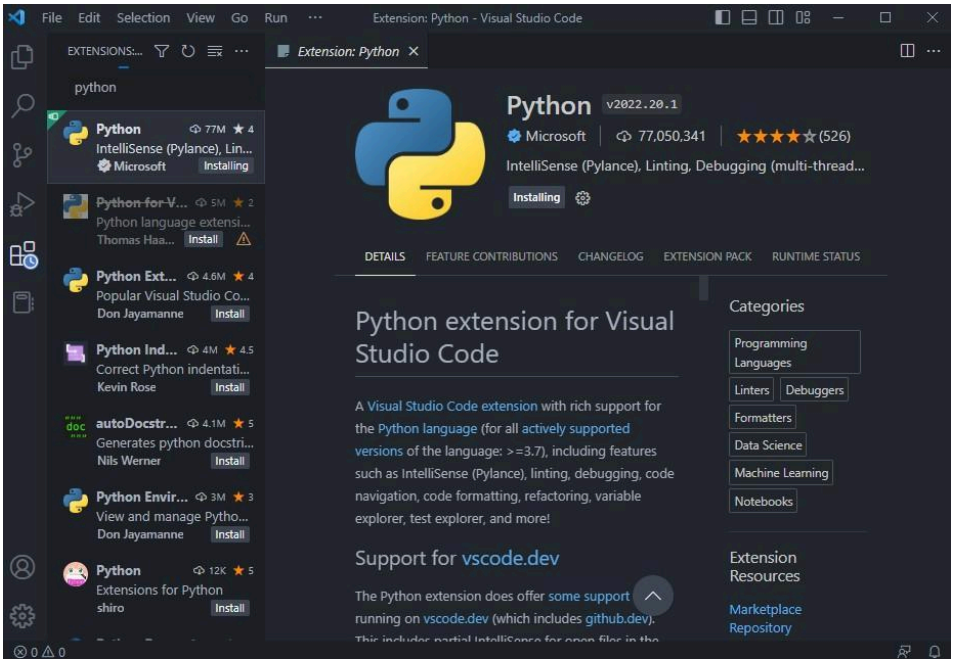
9. Pilih bagian **System variables > Path** lalu klik Edit.



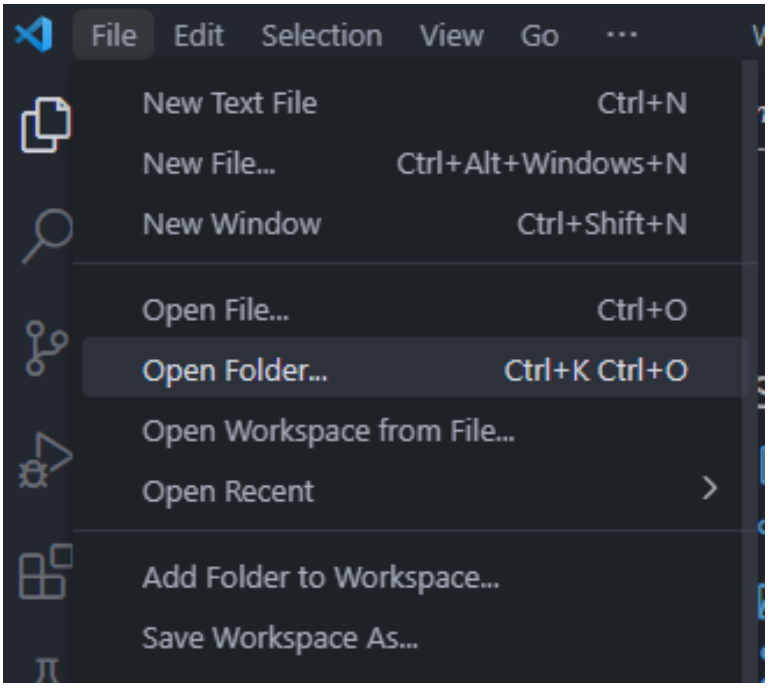
10. Klik tombol **New** kemudian paste alamat **C:\Python\Python310** yang telah di-copy tadi. Setelah selesai klik OK.



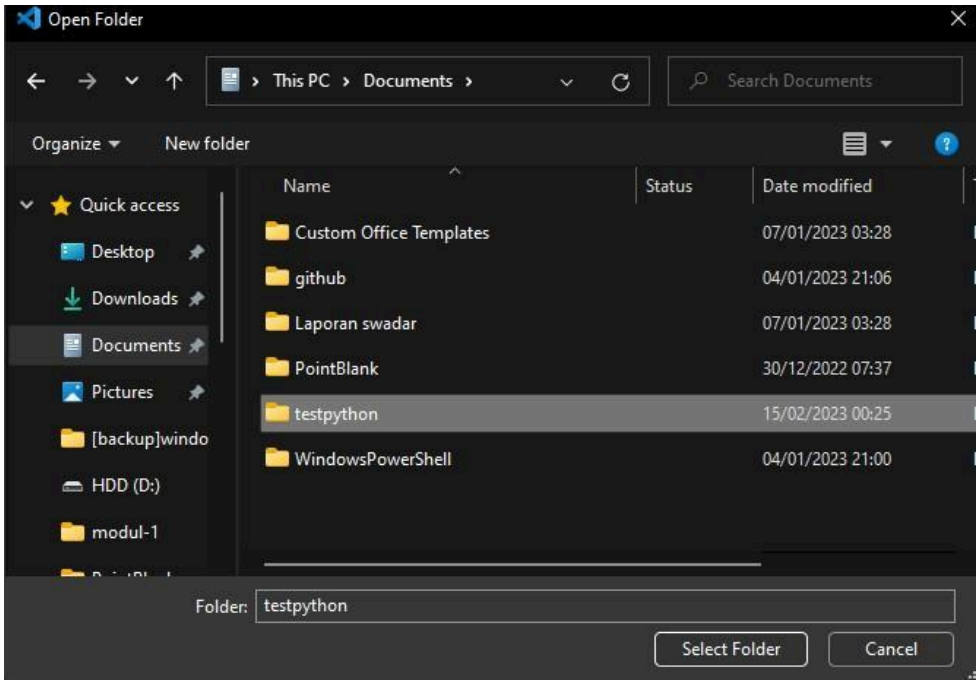
11. Buka VS Code yang sudah terinstal, Lalu klik menu **Extensions** pada **Activity Bar** (sebelah kiri). Ketik **python**, kemudian **Install** dan reload VS Code Anda.



12. Pada bagian **Top Bar**, klik **File > Open Folder**. Mulai dari sini kita akan melakukan tes running project Python.

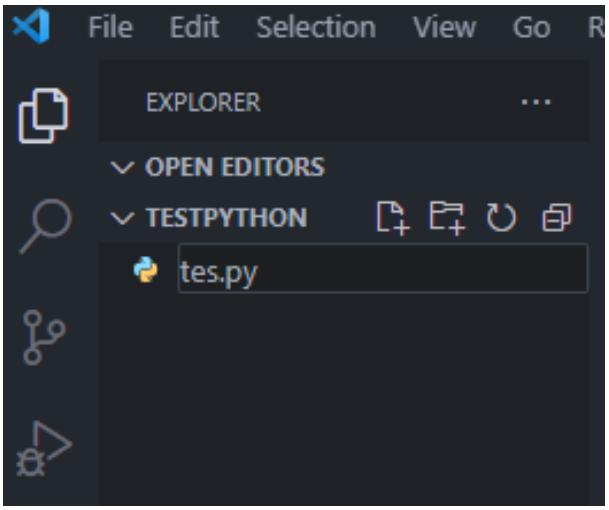


13. Pilih lokasi folder project. Anda bisa membuat folder dimana saja. Pada kasus ini saya membuatnya di **Documents** dengan nama **tespython**. Kemudian klik Select Folder.

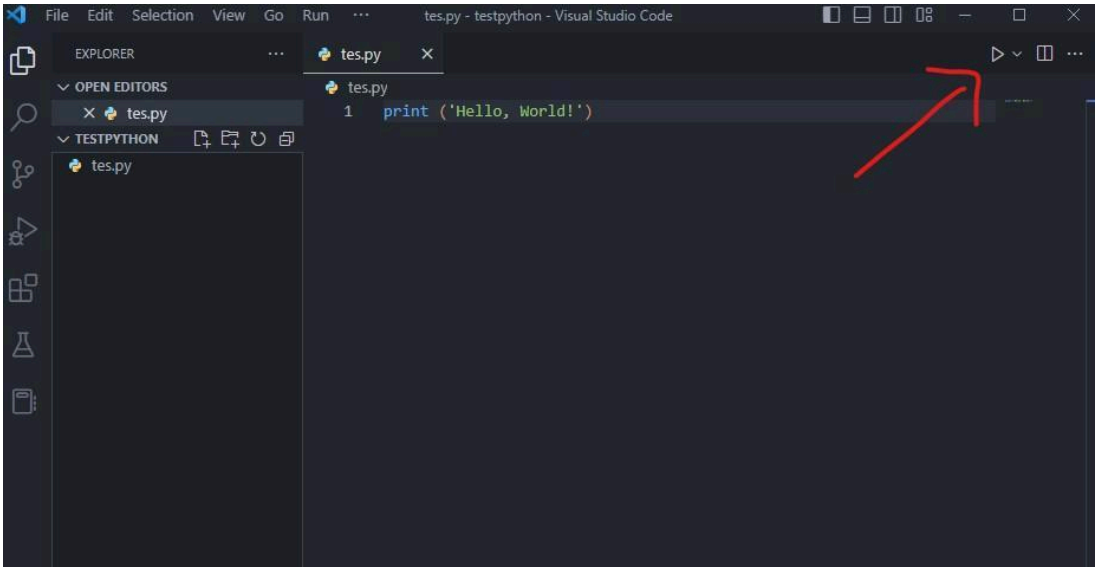


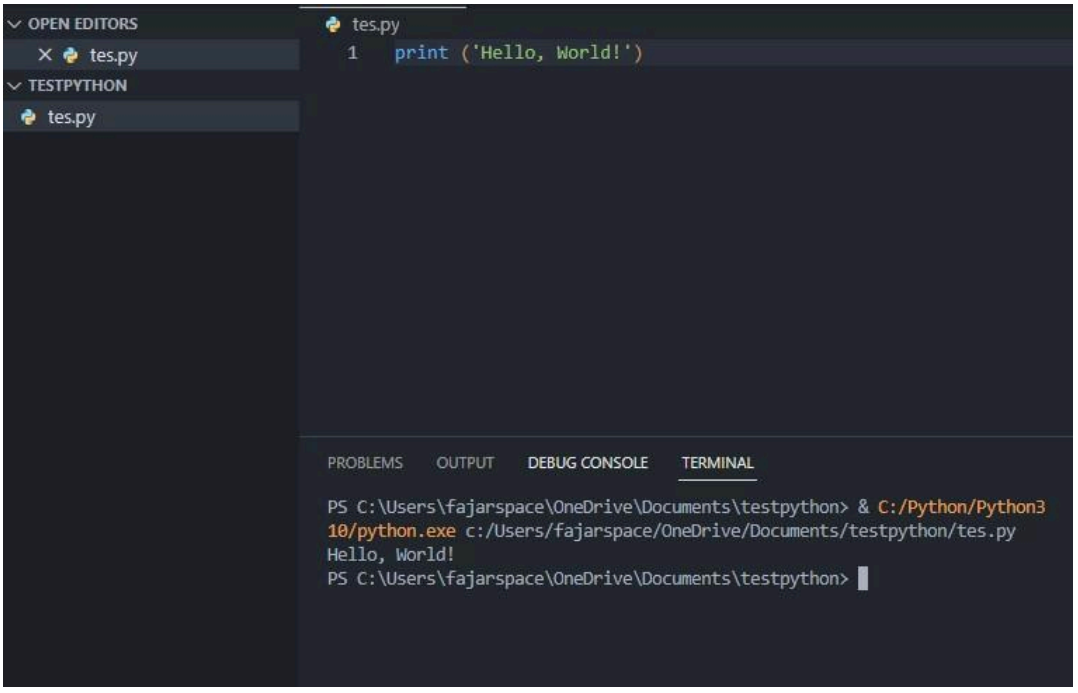
14. Setelah tampil di **Explorer**. Klik **New File** untuk membuat file Python.

15. Ketik nama file yang Anda inginkan. Jangan lupa menambahkan ekstensi **.py** dibelakang nama file. Misalnya **tes.py**



16. Contoh dibawah adalah menampilkan teks Tes Python, kemudian klik tombol **Run** pada pojok kanan atas.





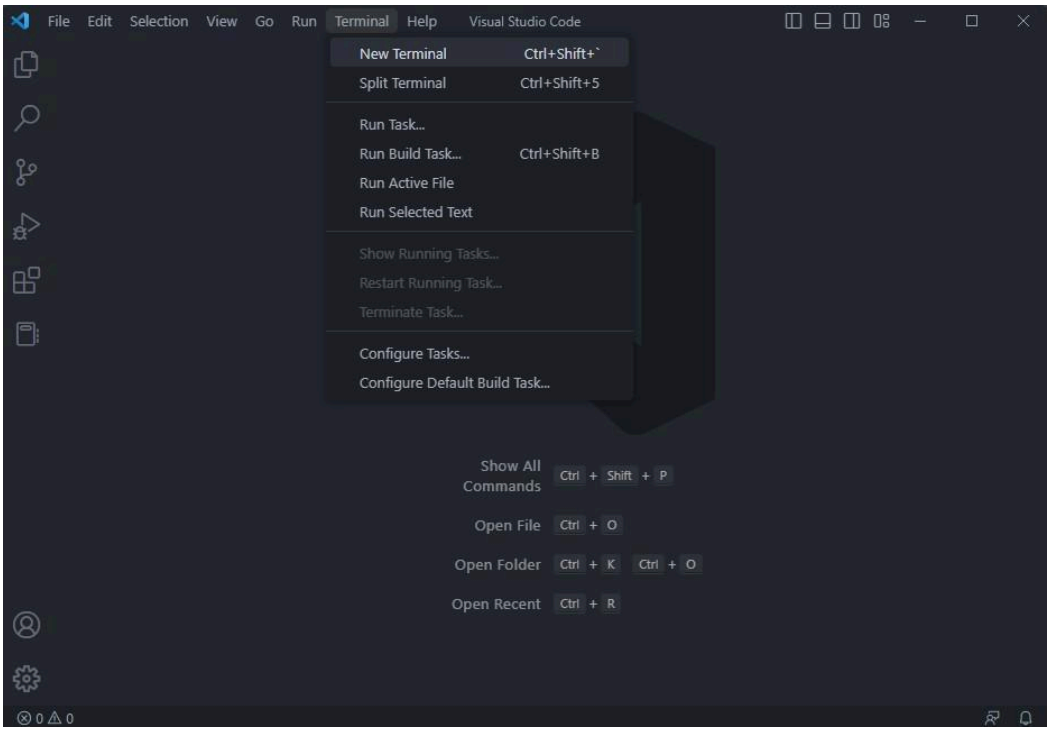
INSTALASI LIBRARY PANDAS, MATPLOTLIB

TUJUAN

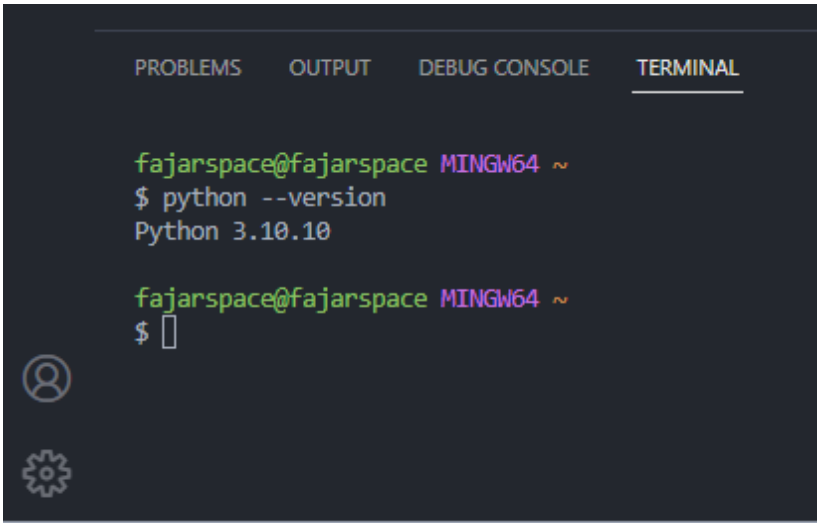
1. Mahasiswa Mampu Melakukan Instalasi Library pandas dan matplotlib

INSTALASI

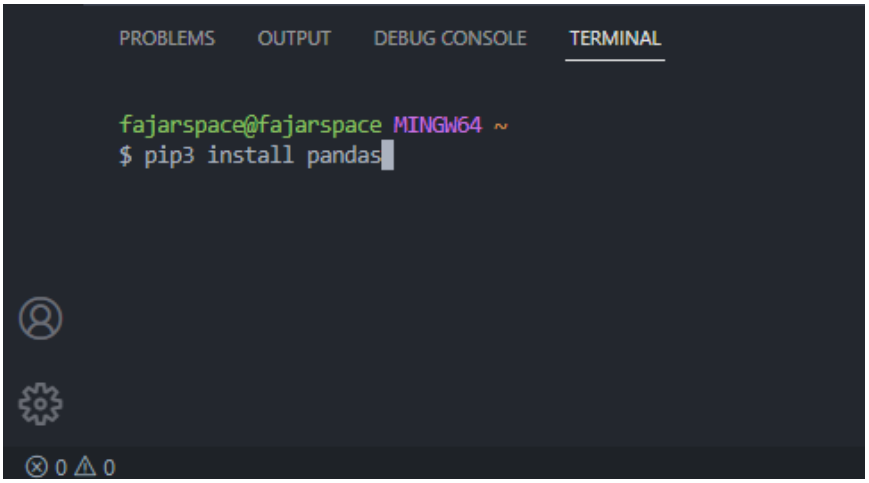
1. Buka VSCode yang sebelumnya sudah di install, lalu klik **terminal > new terminal**



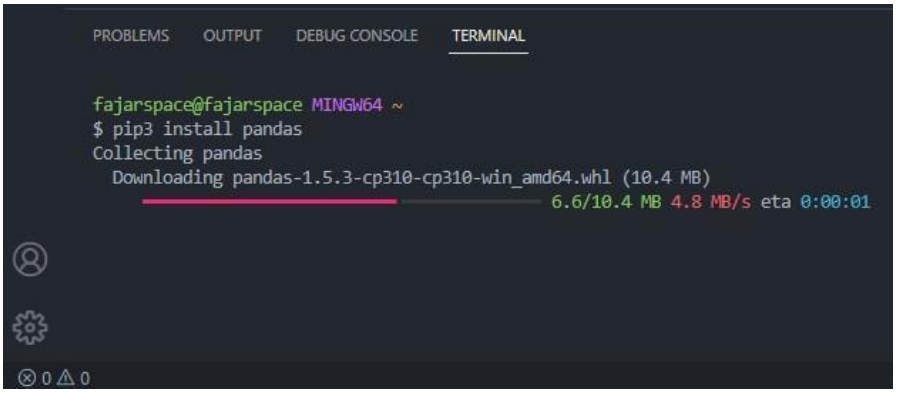
2. Untuk mengecek python yang sudah kita install ketik **python –version** disitu tercetak **python 3.10.10**, yang artinya versi python yang terinstal adalah versi 3.10



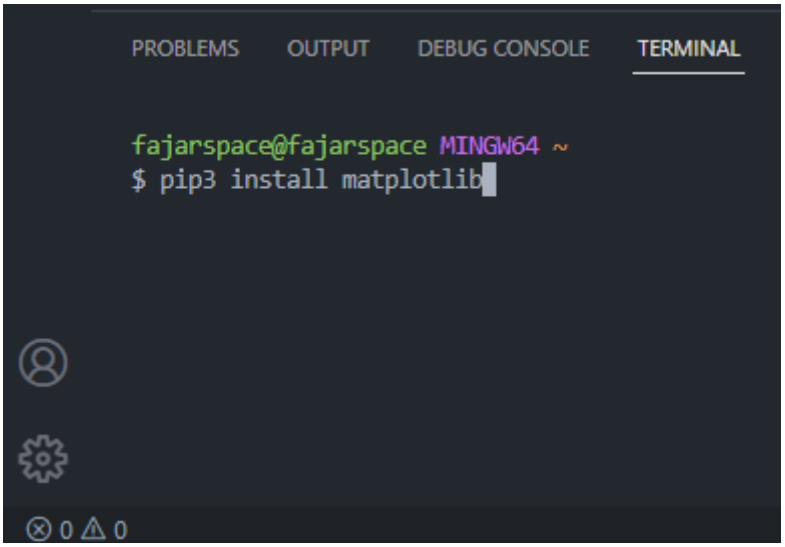
3. Untuk dapat menginstall **pandas**, kita bisa menjalankan perintah berikut :



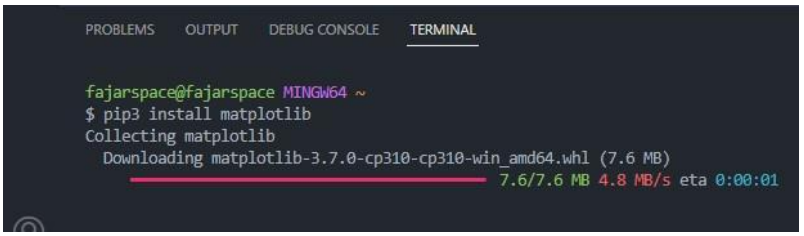
Lalu klik **enter**, tunggu hingga proses selesai.



4. Lalu untuk dapat menginstall **matplotlib**, kita bisa menjalankan perintah berikut :



Lalu klik **enter**, tunggu hingga proses selesai.



PRAKTIKUM 2

JUDUL : Python Dasar

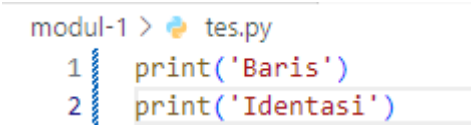
Sintaks Dasar Python

Identifier

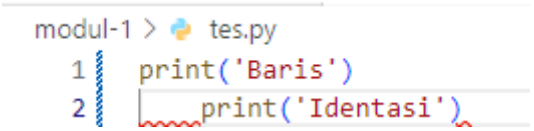
Identifier adalah identitas atau nama yang telah diberikan kepada function, variabel, obyek, class, namespace dan lain-lain.

1.1. Baris dan Indentasi

Python tidak menggunakan tanda { } untuk menandai blok/grup kode. Blok kode di python menggunakan tanda indentasi (spasi). Jumlah spasi untuk setiap baris yang ada dalam satu blok kode harus sama.



Gambar 1: Contoh benar



Gambar 2: Contoh Salah

2. Tipe Data

Beberapa tipe data built-in atau bawaan yang cukup lengkap dan tidak sulit untuk digunakan. Beberapa tipe data yang wajib Anda ketahui di Python3 antara lain:

- int, tipe data yang dapat Anda isi dengan bilangan bulat.
- float, tipe data yang dapat Anda isi dengan bilangan koma.
- string, tipe data yang dapat Anda isi dengan sebuah karakter atau kalimat.
- complex, tipe data bilangan kompleks atau bilangan imajiner, seperti 5j, 54j, 1j.
- long, tipe data yang dapat Anda isi dengan bilangan yang sangat besar. Bisa lebih dari 1 milyar.

- boolean, tipe data yang dapat Anda isi untuk mendukung operasi bool. Nilainya hanya True dan False.
- List, adalah tipe data yang berisi item yang berurut. List bisa berisi anggota dengan tipe yang sama maupun berbeda. Untuk mendeklarasikan list, digunakan tanda kurung [ ] dan masing-masing anggotanya dipisahkan oleh tanda koma.
- Tuple, adalah jenis data lain yang mirip dengan list. Perbedaannya dengan list adalah anggotanya tidak bisa diubah (immutable). Tuple dideklarasikan dengan menggunakan tanda kurung ( ). dan anggotanya dipisahkan oleh tanda koma.
- Set, adalah salah satu tipe data di Python yang tidak berurut (unordered).
- Dictionary, adalah tipe data yang tiap anggotanya terdiri dari pasangan kunci- nilai (key-value). Dictionary dideklarasikan dengan menggunakan tanda kurung kurawal

  { }, dimana anggotanya memiliki bentuk kunci:nilai atau key:value dan tiap anggota dipisah tanda koma. Kunci dan nilainya bisa memiliki tipe sembarang.

3. Variabel, Operator, dan Ekspresi 3.1.Variabel

Variabel adalah lokasi di memori yang digunakan untuk menyimpan nilai. Pada saat kita membuat sebuah variabel, kita ‘memesan’ tempat di dalam memori. Tempat tersebut bisa diisi dengan data atau objek, baik itu bilangan bulat (integer), pecahan (float), karakter (string), dan lain - lain.

Di python, variabel tidak perlu dideklarasikan secara eksplisit. Deklarasi atau pembuatan variabel terjadi secara otomatis pada saat kita memberi (menugaskan) suatu nilai ke variabel. Tanda sama dengan (=) digunakan untuk memberikan nilai ke variabel. Misal *a = ‘Hello, world!’*

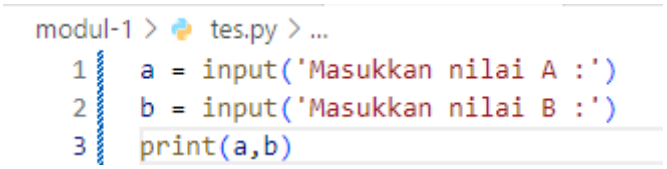
3.2. Operator & Ekspresi

Hampir semua statemen (baris logis) yang Anda tulis akan mengandung ekspresi. Contoh sederhana dari ekspresi adalah 2+3. Sebuah ekspresi dapat diturunkan menjadi operator dan operand. Operator adalah simbol tertentu yang digunakan untuk melakukan operasi aritmatika maupun logika. Nilai yang padanya dilakukan operasi disebut operand. Misalnya adalah 2 + 3. Di sini tanda + adalah operator penjumlahan. 2 dan 3 adalah operand.

4. Input dan Output

4.1. Input

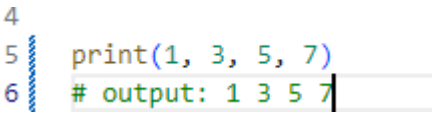
Input adalah masukan yang kita berikan ke program. Program akan memprosesnya dan menampilkan hasil outputnya. Input, proses, dan output adalah inti dari semua program komputer. Fungsi untuk melakukan operasi input adalah fungsi input()



Gambar 2: Contoh input

4.2. Output

Fungsi bawaan untuk melakukan operasi output adalah print(). Seperti yang sudah sering kita praktekkan, kita menggunakan fungsi print() untuk menampilkan data ke perangkat keluaran standar (layar).



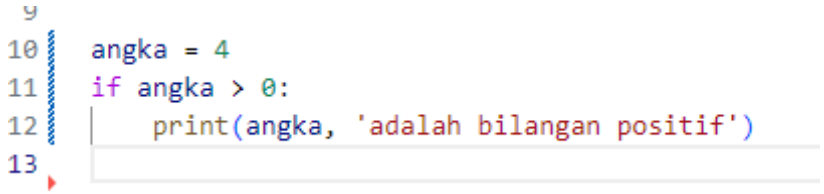
Gambar 3: Contoh output

5. Percabangan (if, if else, if elif else)

Percabangan adalah cara yang digunakan untuk mengambil keputusan apabila di dalam program dihadapkan pada kondisi tertentu. Jumlah kondisinya bisa satu, dua atau lebih. Percabangan mengevaluasi kondisi atau ekspresi yang hasilnya benar atau salah. Kondisi atau ekspresi tersebut disebut ekspresi boolean. Hasil dari pengecekan kondisi adalah True atau False. Bila benar (True), maka pernyataan yang ada di dalam blok kondisi tersebut akan dieksekusi. Bila salah (False), maka blok pernyataan lain yang dieksekusi.

5.1.1. Pernyataan if

Pernyataan if menguji satu buah kondisi. Bila hasilnya benar maka pernyataan di dalam blok if tersebut dieksekusi. Bila salah, maka pernyataan tidak dieksekusi. Sintaksnya adalah seperti berikut:



Gambar 4: Contoh if

5.1.2. Pernyataan if...else

Pernyataan if...else menguji 2 kondisi. Kondisi pertama kalau benar, dan kondisi kedua kalau salah. Sintaksnya adalah seperti berikut :

```
14  bilangan = -1
15  if bilangan >= 0:
16      print('Positif atau nol')
17  else:
18      print('Bilangan negatif')
```

Gambar 5: Contoh if else

5.1.3. Pernyataan if...elif...else...

Pernyataan if...elif...else digunakan untuk menguji lebih dari 2 kondisi. Bila kondisi pada if benar, maka pernyataan di dalamnya yang dieksekusi. Bila salah, maka masuk ke pengujian kondisi elif.

```
20  bilangan = 5.5
21  if bilangan > 0:
22      print('Positif atau nol')
23  elif bilangan == 0:
24      print('Nol')
25  else:
26      print('Bilangan negatif')
```

Gambar 6: Contoh if elif else

5.1.4. Tambahan : if Bersarang

Sebuah kondisional dapat disimpan di dalam if lain. Berikut ini adalah contoh kode if bersarang di Python :

percabangan_if_bersarang.py › ...

```
1   gaji = 10000000
2   berkeluarga = True
3   punya_rumah = True
4
5   if gaji > 3000000:
6       print ("Gaji sudah diatas UMR")
7       if berkeluarga:
8           print ("Wajib ikut asuransi dan menabung untuk pensiun")
9       else:
10          print ("Tidak perlu ikut asuransi")
11
12      if punya_rumah:
13          print ("Wajib bayar pajak rumah")
14      else:
15          print ("Tidak wajib bayar pajak rumah")
16  else:
17      print ("Gaji belum UMR")
```

6. Perulangan

Secara umum, Python mengeksekusi program baris perbaris. Mulai dari baris satu, dua, dan seterusnya. Ada kalanya, kita perlu mengeksekusi satu baris atau satu blok kode program beberapa kali.

6.1.1. Perulangan dengan Menggunakan for

Perulangan dengan menggunakan for memiliki sintaks seperti berikut :

for var in sequence:
body of for

var adalah variabel yang digunakan untuk penampung sementara nilai dari sequence pada saat terjadi perulangan. **Sequence** adalah tipe data berurut seperti string, list, dan tuple.

```
28  nomor = [5, 5, 2]
29  jumlah = 0
30  for tampung in nomor:
31      jumlah = jumlah + tampung
32  print ('jumlah semuanya :', jumlah)
```

Gambar 7: Contoh for

Perulangan for dengan range

Fungsi range() dapat digunakan untuk menghasilkan deret bilangan. range(10) akan menghasilkan bilangan dari 0 sampai dengan 9 (10 bilangan).

```
34  for hitung in range(5):
35      print('Hitung :', hitung)
```

Gambar 8: Contoh for dengan range()

6.1.2. Perulangan Menggunakan while

Perulangan menggunakan while akan menjalankan blok pernyataan terus menerus selama kondisi bernilai benar.

Adapun sintaks dari perulangan menggunakan while adalah :

while expression:
statement (s)

Di sini, **statement (s)** bisa terdiri dari satu baris atau satu blok pernyataan. **Expression** merupakan ekspresi atau kondisi apa saja, dan untuk nilai selain nol dianggap True.

```
37  hitung = 0
38  while (hitung < 5):
39      print('hitung :', hitung)
40      hitung = hitung + 1
```

Gambar 9: Contoh while

Contoh Program Kelipatan Bilangan Genap

Ketentuan : Program pengulangan dengan for. Tampilkan bilangan genap dari 0 hingga batas terakhir bilangan input. Misalnya, apabila diinput 10, maka yang tampil adalah : 0 2 4 6 8.

```
42  i = 0
43  n = int(input('Masukkan batas :'))
44  for i in range(n):
45      if i%2 == 0:
46          print("Bilangan :", i)
47      i = i + 1
```

Gambar 10: Contoh program kelipatan genap

6.1.3. Latihan

Buatlah program kelipatan bilangan genap dengan menampilkan banyaknya jumlah. Misalnya, apabila diinput 10, maka yang tampil adalah 0 2 4 6 8 10 12 14 16 18 **(10 bilangan).**

7. Fungsi

Fungsi adalah grup/blok program untuk melakukan tugas tertentu yang berulang. Fungsi membuat kode program menjadi reusable, artinya hanya di definisikan sekali saja, dan kemudian bisa digunakan berulang kali dari tempat lain di dalam program.

7.1.1. Mendefinisikan Fungsi

Berikut adalah sintaks yang digunakan untuk membuat fungsi :

def function_name(parameters):

"""function_docstring"""

statement(s)

return [expression]

Penjelasannya dari sintaks fungsi di atas :

- Kata kunci **def** diikuti oleh **function_name** (nama fungsi), tanda kurung dan tanda titik dua (:) menandai header (kepala) fungsi.
- **Parameter**/ argumen adalah input dari luar yang akan diproses di dalam tubuh fungsi.

- **"function_docstring"** bersifat opsional, yaitu sebagai string yang digunakan untuk dokumentasi atau penjelasan fungsi. “function_doctring” diletakkan paling atas setelah baris def.
- Setelah itu diletakkan baris-baris pernyataan **(statements)**. Jangan lupa indentasi untuk menandai blok fungsi.
- bersifat opsional. Gunanya adalah untuk mengembalikan suatu nilai expression dari fungsi.

```
49  def sapa(nama):
50      print('hai,' + nama +  '. Apa kabar?')
51      return nama
52  sapa('Anna')
```

Gambar 11: Contoh fungsi

7.1.2. Contoh Program Luas Persegi Panjang dengan Fungsi

```
54  def persegipanjang(panjang, lebar):
55      luas = panjang * lebar
56      print('Luasnya :', luas)
57      return luas
58
59  print('menghitung luas persegi panjang')
60  a = int(input('Masukkan panjang :'))
61  b = int(input('Masukkan lebar :'))
62  persegipanjang(a, b)
```

Gambar 12: Contoh Program Luas Persegi Panjang dengan Fungsi

7.1.3. Latihan

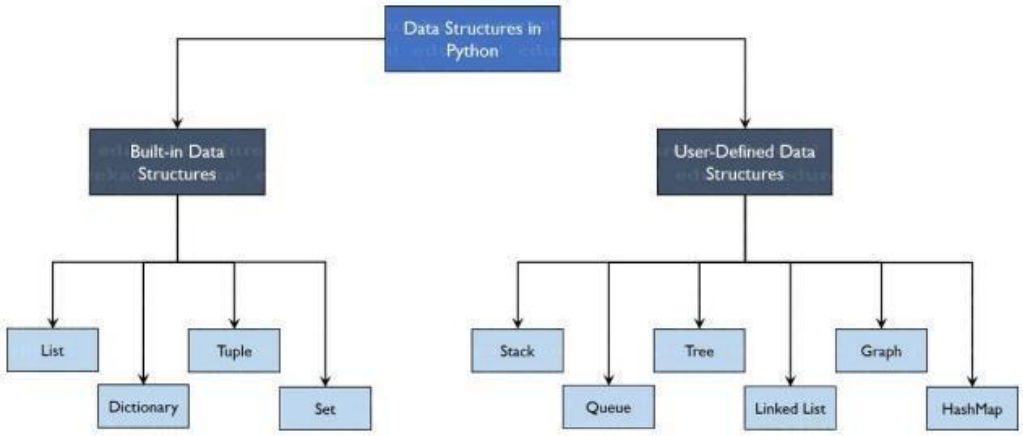
Buatlah program dinamis menghitung luas persegi panjang dan persegi dengan menggunakan 1 fungsi. Misalnya, apabila diinput panjang = 4 dan lebar 3, maka tampil luas persegi panjang = 12. Dan apabila diinput sisi persegi = 3, maka tampil luas persegi = 9. Contoh tampilan terminal seperti gambar dibawah.

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

lient' '--host' 'localhost' '--port' '54851
Menghitung Luas Persegi Panjang & Persegi
Persegi Panjang
Masukkan Panjang : 4
Masukkan Lebar : 3
Luasnya : 12

Persegi
Masukkan Sisi : 3
Luasnya : 9
PS C:\Python Projects\sintaks_ok>
s_ok)
```

MODUL 2
(MINGGU KEDUA)

Praktikum 3

Judul : Struktur Data dalam Python



Gambar 3. 1 Strukur data dalam Python

Struktur data adalah cara yang digunakan untuk mengorganisir dan menyimpan data dalam suatu program. Python menyediakan beberapa jenis struktur data yang dapat digunakan untuk berbagai keperluan, seperti list, tuple, set, dictionary, dan lain-lain. Berikut adalah penjelasan tentang struktur data dalam Python:

LIST

List adalah struktur data yang paling umum digunakan di Python. List adalah kumpulan nilai yang terurut dan dapat diubah-ubah. Setiap nilai dalam list dapat diakses menggunakan indeksnya, yang dimulai dari nol. List dibuat menggunakan ***tanda kurung siku***:

Contoh penggunaan list:

list.py

```
1   thislist = ["apple", "banana", "cherry"]
2   print(thislist)
```

Output :

```
PS D:\praktikum\data mining\python> python .\list.py
['apple', 'banana', 'cherry']
```

Untuk menentukan berapa banyak item yang dimiliki daftar, gunakan fungsi len():

```
1   thislist = ["apple", "banana", "cherry"]
2   print(len(thislist))
```

Output :

```
PS D:\praktikum\data mining\python> python .\list.py
3
```

Item list dapat berupa tipe data apa pun, baik itu numerik, string, boolean ataupun lainnya.

```
1   list1 = ["apple", "banana", "cherry"]
2   list2 = [1, 5, 7, 9, 3]
3   list3 = [True, False, False]
4   print(list1)
5   print(list2)
6   print(list3)
```

Output

```
PS D:\praktikum\data mining\python> python .\list.py
['apple', 'banana', 'cherry']
[1, 5, 7, 9, 3]
[True, False, False]
```

Dalam satu variable list dapat berisi tipe data yang berbeda:

```
1   list1 = ["abc", 34, True, 40, "male"]
2   print(list1)
```

Output :

```
PS D:\praktikum\data mining\python> python .\list.py
['abc', 34, True, 40, 'male']
```

Item list diindeks dengan aturan item pertama memiliki index [0], item kedua memiliki indeks [1],

```
1   thislist = ["apple", "banana", "cherry"]
2   print(thislist[1])
```

Output

```
PS D:\praktikum\data mining\python> python .\list.py
banana
```

Pengindeksan negatif berarti mulai dari akhir. -1 mengacu pada item terakhir, -2 mengacu pada item terakhir kedua dll.

```
1   thislist = ["apple", "banana", "cherry"]
2   print(thislist[-1])
```

Output

```
PS D:\praktikum\data mining\python> python .\list.py
cherry
```

Anda dapat menentukan rentang indeks dengan menentukan di mana harus memulai dan di mana harus mengakhiri rentang. Saat menentukan rentang, nilai yang dikembalikan (hasilnya) akan menjadi daftar baru dengan item yang ditentukan.

```
1   thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
2   print(thislist[2:5])
```

Output :

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
['cherry', 'orange', 'kiwi']
```

Untuk kasus rentang, dengan mengabaikan nilai awal, rentang akan dimulai pada item pertama:

```
1   thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
2   print(thislist[:4])
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
['apple', 'banana', 'cherry', 'orange']
```

Masih pada kasus rentang, dengan meninggalkan nilai akhir, rentang nilai akhir akan diset ke akhir daftar:

```
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
print(thislist[4:])
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
['kiwi', 'melon', 'mango']
```

Untuk menentukan apakah item tertentu ada dalam list, gunakan **in** :

```
thislist = ["apple", "banana", "cherry"]
if "apple" in thislist:
    print("Ya, 'apple' ada di dalam list")
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
Ya, 'apple' ada di dalam list
```

Latihan 3.1

1. Buat list yang isinyanya adalah nama depan dan nama belakang masing-masing praktikan, kemudian hitung huruf vocal yang ada di nama masing-masing !

MENGUBAH NILAI LIST

Untuk mengubah nilai list, tinggal tentukan saja pada indeks yang diinginkan. Perhatikan contoh berikut:

```
thislist = ["apple", "banana", "cherry"]
thislist[1] = "blackcurrant"
print(thislist)
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
['apple', 'blackcurrant', 'cherry']
```

Perhatikan, pada indeks 1, yang awalnya adalah **banana**, diganti nilainya dengan blackcurrent.

APPEND / MENAMBAHKAN ITEM

Untuk menambahkan item ke akhir list, kita dapat gunakan metode append() :

```
thislist = ["apple", "banana", "cherry"]
thislist.append("orange")
print(thislist)
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
['apple', 'banana', 'cherry', 'orange']
```

Untuk menyisipkan item baru di tengah-tengah / tidak di akahir, kita dapat menggunakan metode insert(). Metode insert() ini menyisipkan item pada indeks yang ditentukan:

```
thislist = ["apple", "banana", "cherry"]
thislist.insert(2, "watermelon")
print(thislist)
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
['apple', 'banana', 'cherry', 'orange']
```

Perhatikan pada kode thislist.insert(2, “watermelon”). Kode ini artinya adalah kita akan menyisipkan item baru “watermelon” pada tepat indeks 2. Sehingga indeks 2 (cherry) yang awalnya akan mundur menjadi indeks ke-3. Ingat kembali indeks pertama dinamakan indeks ke-0.
Untuk menambahkan elemen dari list lain ke list saat ini, kita dapat gunakan metode extend().

```
thislist = ["apple", "banana", "cherry"]
tropical = ["mango", "pineapple", "papaya"]
thislist.extend(tropical)
print(thislist)
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
['apple', 'banana', 'cherry', 'mango', 'pineapple', 'papaya']
```

Perhatikan pada list thislist telah ditambahkan konten dari list tropical.

MENGHAPUS ITEM

Untuk menghapus item di dalam list, kita dapat menggunakan metode remove() terhadap item dengan nilai yang sama sepertiyang diinginkan.

```
thislist = ["apple", "banana", "cherry"]
thislist.remove("banana")
print(thislist)
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
['apple', 'cherry']
```

Untuk menghapus item berdasarkan nomor indeks, kita dapat menggunakan metode pop().

```
thislist = ["apple", "banana", "cherry"]
thislist.pop(1)
print(thislist)
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
['apple', 'cherry']
```

Perhatikan pada kode thislist.pop(1). Kode ini artinya adalah kita akan menghapus item indeks ke-1. Lihat pada hasilnya, indeks ke-1 yaitu banana telah terhapus.

Sedang untuk menghapus isi dari list atau mengosongkannya dapat menggunakan metode clear()

```
thislist = ["apple", "banana", "cherry"]
thislist.clear()
print(thislist)
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
[]
```

Perhatikan, dengan metode clear(), maka list akan menjadi kosong total, namun varibel list tersebut tetap ada.
Untuk menghapus variabel list, dari memory, kita dapat menggunakan metode del().

FOR LOOP PADA LIST

Anda dapat membuat perintah berulang untuk setiap item pada list. Untuk agar lebih memahami perhatikan cotoh berikut:

```
1   thislist = ["apple", "banana", "cherry"]
2   for x in thislist:
3       print(x)
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
apple
banana
cherry
```

Perhatikan pada kode for x in thislist:, pada kode ini memberitahu mesin bahwa kita ingin melakukan perulangan perintah untuk setiap item yang ada di list thislist dan disimbolkan dengan x. Kemudian pada bagian menjorok kedepan ada kode print(x), kode ini artinya kita memberitahu mesin untuk memprint-out ke layar yaitu varible x. Dimana variable x adalah item pada list thislist.

Perlu dipahami, jika di bahasa pemrograman lain untuk menunjukkan isi dari kumpulan statement adalah dengan kurung kurawa { ….. }, jika di phyton adalah dengan indent (kode yang agak menjorok ke depan dari parent-nya)

SORTING / PENGURUTAN LIST

Untuk mengurutkan berdasarkan alfanumerik list, dapat menggunakan metode sort():

```
thislist = ["orange", "mango", "kiwi", "pineapple", "banana"]
thislist.sort()
print(thislist)
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
['banana', 'kiwi', 'mango', 'orange', 'pineapple']
```

Perhatikan contoh lagi dibawah ini untuk mengurutkan angka.

```
1   thislist = [100, 50, 65, 82, 23]
2   thislist.sort()
3   print(thislist)
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
[23, 50, 65, 82, 100]
```

Secara default, metode **sort**() akan mengurukan secara naik yaitu dari yang terkecil ke terbesar. Untuk mengurutkan sebaliknya / menurun kita dapat menggunakan metode **sort**(**reverse=True**).

```
1   thislist = ["orange", "mango", "kiwi", "pineapple", "banana"]
2   thislist.sort(reverse = True)
3   print(thislist)
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
['pineapple', 'orange', 'mango', 'kiwi', 'banana']
```

Phyton juga mempunyai fungsi reverse() untuk membalikkan urutan dari list. Artinya yang paling belakang akan menjadi yang paling depan dan sebaliknya. Perhatikan contoh berikut:

```
1   thislist = ["banana", "Orange", "Kiwi", "cherry"]
2   thislist.reverse()
3   print(thislist)
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
['cherry', 'Kiwi', 'Orange', 'banana']
```

MENYALIN LIST (COPY)

Untuk mengopy variable list ke variable lain, di phyton tidak bisa sesederhana dengan mengetik *list2 = list1*, karena: *list2* hanya akan menjadi referensi ke *list1*, dan perubahan yang dibuat *list1* secara otomatis juga akan dibuat di *list2*. Mungkin ini sedikit membingungkan, tapi nanti anda akan terbiasa, ini seperti pointer jika di bahasa pemrograman lain.

Ada cara untuk membuat salinan, salah satu caranya adalah dengan menggunakan metode bawaan copy().

```
1   thislist = ["apple", "banana", "cherry"]
2   mylist = thislist.copy()
3   print(mylist)
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
['apple', 'banana', 'cherry']
```

Cara lain untuk membuat salinan adalah dengan menggunakan metode bawaan list().

```
thislist = ["apple", "banana", "cherry"]
mylist = list(thislist)
print(mylist)
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
['apple', 'banana', 'cherry']
```

Latihan 3.2

1. Buat sebuah list yang isinya adalah nama masing-maisng praktikan, kemudian balik nama tersebut menggunakan fungsi yang sudah didefiniskan sendiri.
   a. Contoh : “Susi Susanti” menjadi “Susanti Susi”

MENGGABUNGKAN BEBERAPA LIST (JOIN)

Ada beberapa cara untuk menggabungkan dua atau lebih daftar dengan Python. Salah satu cara termudah adalah dengan menggunakan operator +.

```
list1 = ["a", "b", "c"]
list2 = [1, 2, 3]
list3 = list1 + list2
print(list3)
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
['a', 'b', 'c', 1, 2, 3]
```

Kita juga dapat menggunakan metode **extend**(), yang tujuannya adalah untuk menambahkan elemen dari satu daftar ke daftar lain:

```
list1 = ["a", "b" , "c"]
list2 = [1, 2, 3]
list1.extend(list2)
print(list1)
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
['a', 'b', 'c', 1, 2, 3]
```

Tuple

Tuple digunakan untuk menyimpan beberapa item dalam satu variabel. Berbeda dengan list yang bisa kita ubah-ubah, Tuple adalah kumpulan yang dipesan dan *tidak dapat diubah.* Ingat kembali bab sebelumnya**,**

jika list ditulis dengan kurung siku, tuple ditulis dengan kurung bulat.

```
1   thistuple = ("apple", "banana", "cherry")
2   print(thistuple)
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
('apple', 'banana', 'cherry')
```

Sama seperti list, Item tuple diindeks, item pertama memiliki index [0], item kedua memiliki indeks [1], dst

Tuple tidak dapat diubah, artinya kita tidak dapat mengubah, menambah, atau menghapus item setelah tuple dibuat. Urutannya pun tidak dapat diubah, diurutkan, dan lainnya.
Untuk menentukan berapa banyak item yang dimiliki sebuah tuple, gunakan fungsi len():

```
1   thistuple = ("apple", "banana", "cherry")
2   print(len(thistuple))
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
3
```

Untuk membuat Tuple dengan hanya satu item, Anda harus menambahkan koma setelah item, jika tidak Python tidak akan mengenalinya sebagai Tuple.

```
1   thistuple = ("apple",)
2   print(type(thistuple))
3   #NOT a tuple
4   thistuple = ("apple")
5   print(type(thistuple))
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
<class 'tuple'>
<class 'str'>
```

Item tuple dapat berupa tipe data apa pun baik itu strig, numerik, boolean, dan lain sebagainya. Dalam satu tuple juga bisa mengizinkan tipe data yang berbeda-beda.

Kita dapat mengakses/memanggil item Tuple dengan mengacu pada nomor indeks, di dalam tanda kurung siku, sama seperti list:

```
1   thistuple = ("apple", "banana", "cherry")
2   print(thistuple[1])
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
banana
```

Setelah tuple dibuat, Anda tidak dapat mengubah nilainya. Tuple tidak dapat diubah , atau tidak dapat diubah seperti yang juga disebut. Tapi ada solusi. Anda dapat mengonversi Tuple menjadi list, mengubah list, dan mengonversi list kembali menjadi Tuple.

```
1   x = ("apple", "banana", "cherry")
2   y = list(x)
3   y[1] = "kiwi"
4   x = tuple(y)
5   print(x)
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
('apple', 'kiwi', 'cherry')
```

Anda diperbolehkan untuk menambahkan tupel ke tupel, jadi jika Anda ingin menambahkan satu item, (atau banyak), buat tupel baru dengan item tersebut, dan tambahkan ke tupel yang ada:

Sama seperti list, anda dapat mengulang item Tuple dengan menggunakan forloop in.

Latihan 3.3

1. Buat tuple dengan isi adalah nim masing-masing praktikan, kemudian tambahkan tanggal lahir masing-masing dalam Tuple tersebut !

Set

Set digunakan untuk menyimpan beberapa item dalam satu variabel. Berbeda dengan list dan tuple, **set tidak diindeks yang artinya tidak bisa diurutkan, tidak bisa diubah**. Kita hanya bisa menghapus dan menambahkan saja. **Jika list dituliskan dengan kurung siku, tuple kurung bulat, set dituliskan dengan kurung kurawa.**

Perhatikan contoh berikut

```
1   thisset = {"apple", "banana", "cherry"}
2   print(thisset)
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
{'apple', 'cherry', 'banana'}
```

Set tidak boleh memiliki dua item dengan nilai yang sama. Perhatikan contoh berikut:

```
1   thisset = {"apple", "banana", "cherry", "apple"}
2   print(thisset)
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
{'apple', 'cherry', 'banana'}
```

Perhatikan contoh diatas, oleh karena pada item terakhir (apple) sama dengan item yang pertama, maka secara otomatis python akan menghapus item yang lebih awal.

Untuk menentukan berapa banyak item yang dimiliki satu set, gunakan fungsi len(). Item pada set dapat berupa tipe data apa pun, sama seperti list ataupun tuple.

Dan satu set dapat berisi tipe data yang berbeda.

```
1   set1 = {"abc", 34, True, 40, "male"}
2   print(set1)
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
{True, 34, 40, 'male', 'abc'}
```

Jangan kaget jika saat dirun, urutan yang keluar berbeda dari saat awal kita deklarasikan. Karena memang set tidak menyimpan urutan item.

Kita tidak dapat mengakses item dalam set dengan mengacu pada indeks atau kunci. Tetapi Kita dapat mengulang item yang ditetapkan menggunakan for loop.

```
1   thisset = {"apple", "banana", "cherry"}
2   for x in thisset:
3       print(x)
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
apple
cherry
banana
```

Kita juga bisa menanyakan apakah nilai yang ditentukan ada dalam satu set, dengan menggunakan kata kunci **in**

```
1   thisset = {"apple", "banana", "cherry"}
2   print("banana" in thisset)
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
True
```

Untuk menambahkan satu item ke set gunakan fungsi add().

```
1   thisset = {"apple", "banana", "cherry"}
2   thisset.add("orange")
3   print(thisset)
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
{'apple', 'cherry', 'orange', 'banana'}
```

MENGHAPUS ITEM

Untuk menghapus item dalam set dapat menggunakan remove(), discard(), atau clear(). Kita akan bahasa satu per satu.
Pertama, remove(), coba perhatikan contoh berikut:

```python
thisset = {"apple", "banana", "cherry"}
thisset.remove("banana")
print(thisset)
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
{'cherry', 'apple'}
```

Fungsi remove() akan menimbulkan error jika kata kunci yang mau kita hapus tidak ditemukan di dalam set yang terkait, perhatikan cotoh berikut:

```python
thisset = {"apple", "banana", "cherry"}
thisset.remove("kiwi")
print(thisset)
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
Traceback (most recent call last):
  File "tes.py", line 2, in <module>
    thisset.remove("kiwi")
KeyError: 'kiwi'
```

Berbeda dengan remove(), dengan fungsi discard(), saat kata kunci yang kita tentukan untuk dihapus tidak ditemukan di set maka python tidak akan error.

Untuk menghapus keseluruhan isi dari set kita dapat menggunakan fungsi clear().

INTERAKSI DUA SET

Untuk menambahkan item dari set lain ke set saat ini, gunakan fungsi update().

```python
thisset = {"apple", "banana", "cherry"}
tropical = {"pineapple", "mango", "papaya"}
thisset.update(tropical)
print(thisset)
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
{'pineapple', 'mango', 'banana', 'papaya', 'cherry', 'apple'}
```

Untuk mendapatan nilai irisan, kita dapat menggunaka, Metode intersection_update().

Dengan fungsi ini, kita hanya akan menyimpan item yang ada di kedua set (irisan).

```
x = {"apple", "banana", "cherry"}
y = {"google", "microsoft", "apple"}
x.intersection_update(y)
print(x)
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
{'apple'}
```

Untuk mendapatkan nilai yang tidak ada di kedua set (berkebalikan dengan irisan), maka kita dapat menggunakan metode **symmetric_difference_update()**. Dengan fungsi ini hanya akan menyimpan elemen yang tidak ada di kedua set.

2. Dictionary

Dictionary (kamus) digunakan untuk menyimpan nilai data dalam pasangan **key:value** atau dalam bahasa Indonesia **kunci:nilai**.

Dictionary adalah kumpulan yang dipesan*, dapat diubah dan tidak memungkinkan duplikat.

Dictionary ditulis dengan tanda kurung kurawal, dan memiliki kunci dan nilai,

perhatikan contoh dibawah ini

```
thisdict = {
"brand": "Ford",
"model": "Mustang",
"year": 1964 }
print(thisdict)
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
```

Item dictionary disajikan dalam pasangan **kunci:nilai,** dan dapat dirujuk dengan menggunakan nama kuncinya, lihat contoh dibawah ini:

```
1   thisdict = {
2   "brand": "Ford",
3   "model": "Mustang",
4   "year": 1964
5   }
6   print(thisdict["brand"])
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
Ford
```

Dictionary tidak boleh memiliki dua item dengan kunci yang sama:

```
1   thisdict = {
2   "brand": "Ford",
3   "model": "Mustang",
4   "year": 1964,
5   "year": 2020
6   }
7   print(thisdict)
```

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
{'brand': 'Ford', 'model': 'Mustang', 'year': 2020}
```

Perhatikan contoh diatas, oleh karena ada dua kata kunci yang sama yaitu year, maka yang akan dieksekusi oleh python adalah yag didefinisikan yang paling terakhir.
Untuk menentukan berapa banyak item yang dimiliki kamus, gunakan fungsi len():

```
1   thisdict = {
2   "brand": "Ford",
3   "model": "Mustang",
4   "year": 1964,
5   "year": 2020
6   }
7   print(len(thisdict))
```

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
3
```

Nilai dalam item kamus dapat berupa tipe data apa pun, bahkan berupa list, tuple, atau set.

```
thisdict = {
"brand": "Ford",
"electric": False,
"year": 1964,
"colors": ["red", "white", "blue"]
}
print(thisdict)
```

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
{'brand': 'Ford', 'electric': False, 'year': 1964, 'colors': ['red', 'white', 'blue']}
```

Kita dapat mengakses item dictionary dengan mengacu pada nama kuncinya, di dalam tanda kurung siku

```
thisdict = {
"brand": "Ford",
"model": "Mustang",
"year": 1964
}
x = thisdict["model"]
print(x)
```

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
Mustang
```

Ada juga metode yang disebut get() yang akan memberikan hasil yang sama:

```
thisdict = {
"brand": "Ford",
"model": "Mustang",
"year": 1964
}
x = thisdict.get("model")
print(x)
```

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
Mustang
```

Metode keys(), akan mengembalikan daftar semua kunci dalam dictionary.

```python
thisdict = {
"brand": "Ford",
"model": "Mustang",
"year": 1964
}
x = thisdict.keys()
print(x)
```

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
dict_keys(['brand', 'model', 'year'])
```

Metode values() akan mengembalikan daftar semua nilai dalam dictionary.

```python
thisdict = {
"brand": "Ford",
"model": "Mustang",
"year": 1964
}
x = thisdict.values()
print(x)
```

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
dict_values(['Ford', 'Mustang', 1964])
```

Untuk mengembalikan pasangan kunci dan nilai untuk setiap item dari dictionary dapat menggunakan metode items(). Fungsi ini akan mengembalikan setiap item dalam dictionary, sebagai tupel dalam dictionary.

```python
thisdict = {
"brand": "Ford",
"model": "Mustang",
"year": 1964
}
x = thisdict.items()
print(x)
```

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
dict_items([('brand', 'Ford'), ('model', 'Mustang'), ('year', 1964)])
```

Untuk menentukan apakah kunci tertentu ada dalam dictionary, gunakan kata kunci in:

```python
thisdict = {
"brand": "Ford",
"model": "Mustang",
"year": 1964
}

if "model" in thisdict:
    print("Yes, 'model' is one of the keys in the thisdict dictionary")
```

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
Yes, 'model' is one of the keys in the thisdict dictionary
```

MENGUBAH NILAI DICTIONARY

Untuk mengubah nilai suatu kata kunci pada dictionary, kita tinggal panggil saja kata kuncinya dengan kurung siku.

Perhatikan contoh berikut:

```python
thisdict = {
"brand": "Ford",
"model": "Mustang",
"year": 1964
}
thisdict["year"] = 2018
print(thisdict)
```

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
{'brand': 'Ford', 'model': 'Mustang', 'year': 2018}
```

Di contoh diatas kita mengubah kata kunci “year” yang awalnya 1964 menjadi 2018.

Untuk mengubah nilai dictionary, selain menggunakan kurung siku, kita juga bisa menggunakan fungsi update()

```python
thisdict = {
"brand": "Ford",
"model": "Mustang",
"year": 1964
}
thisdict.update({"year": 2023, "model": "SUV"})
print(thisdict)
```

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
{'brand': 'Ford', 'model': 'SUV', 'year': 2023}
```

MENAMBAHKAN ITEM

Menambahkan item ke dictionary dapat dilakukan dengan menggunakan kunci dan indeks baru dan menetapkan nilai untuk itu. Perhatikan contoh berikut.

```
thisdict = {
"brand": "Ford",
"model": "Mustang",
"year": 1964
}
thisdict["color"] = "red"
print(thisdict)
```

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964, 'color': 'red'}
```

Pada contoh diatas, perhatikan kode thisdict[“color”]=“red”. Oleh karena kunci “color” belum ada di dictionary, maka python akan mengindikasikan ini sebagai penambahan item baru yaitu dengan kunci “color” dan nilai “red”. Pada kasus yang lain, apabila kata kunci yang dipanngil sudah ada di dictionary maka nilainya akan diupdate.

MENGHAPUS ITEM

Ada beberapa metode untuk menghapus item dari dictionary. Yang pertama adalah menggunakan fungsi pop().

```
thisdict = {
"brand": "Ford",
"model": "Mustang",
"year": 1964
}
thisdict.pop("model")
print(thisdict)
```

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964, 'color': 'red'}
```

Pada contoh diatas, kode thisdict.pop("model"), artinya adalah kita akan menghapus item dengan kata kunci "model". Cara yang lain untuk menghapus item di dictionary adalah dengan fungsi del. Perhatikan contoh berikut, kita akan coba menghapus item dengan kata kunci "model" namun dengan cara **del.**

```
thisdict = {
"brand": "Ford",
"model": "Mustang",
"year": 1964
}
del thisdict["model"]
print(thisdict)
```

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
{'brand': 'Ford', 'year': 1964}
```

Sedangkan untuk menghapus total keseulurhan isi dari dictionary kita dapat menggunakan fungsi clear().

PERULANGAN FOR LOOP PADA DICTIONARY

Sama seperti list, set, dan tuple, kita juga bisa melakukan pengulangan perintah untuk setiap item di dictionary menggunakan for loop in. Perhatikan contoh berikut.

```
thisdict = {
"brand": "Ford",
"model": "Mustang",
"year": 1964
}
for x in thisdict:
    print(x)
```

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
brand
model
year
```

Perlu ingat kembali seperti yang sudah kita bahas di bab sebelumnya, jangan lupa perintah yang ingin kita jalankan di dalam for loop, harus menjorok ke depan (indent).

Di contoh diatas, kita dapat lihat bahwa ketika kita gunakan kode x in thisdict, maka x akan menunjukkan kata kunci. Untuk menampilkan nilainya, alih-alih kata kuncinya, kita bisa menggunakan contoh berikut

```
thisdict = {
"brand": "Ford",
"model": "Mustang",
"year": 1964
}
for x in thisdict:
    print(thisdict[x])
```

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
Ford
Mustang
1964
```

Atau jika kita ingin menampilkan pasangan kunci dan nilai, dengan sedikit strategi, kita bisa menggunakan fungsi items() yang sudah kita bahas sebelumnya. Perhatikan contoh berikut:

```
thisdict = {
"brand": "Ford",
"model": "Mustang",
"year": 1964
}
for x, y in thisdict.items():
    print(x, y)
```

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
brand Ford
model Mustang
year 1964
```

MENYALIN DICTIONARY

Kita tidak dapat menyalin dictionary hanya dengan mengetik dict2 = dict1, karena: dict2 hanya akan menjadi referensi ke dict1, dan perubahan yang dibuat dict1 secara otomatis juga akan dibuat di dict2.
Ada cara untuk membuat salinan, salah satu caranya adalah dengan menggunakan metode **copy()**. Perhatikan contoh berikut.

```python
thisdict = {
"brand": "Ford",
"model": "Mustang",
"year": 1964
}
mydict = thisdict.copy()

print(mydict)
```

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
```

Cara lain untuk membuat salinan adalah dengan menggunakan fungsi bawaan dict(). Perhatikan contoh berikut.

```python
thisdict = {
"brand": "Ford",
"model": "Mustang",
"year": 1964
}
mydict = dict(thisdict)
print(mydict)
```

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
```

NESTED DICTIONARY

Di dalam dictionary, juga dapat berisi dictionary, kita sebut ini nested dictionary. Perhatikan contoh berikut.

```python
myfamily = {
"child1" : {
"name" : "Emil",
"year" : 2004
},
"child2" : {
"name" : "Tobias",
"year" : 2007
},
"child3" : {
"name" : "Linus",
"year" : 2011
}
}
print(myfamily)
```

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
{'child1': {'name': 'Emil', 'year': 2004}, 'child2': {'name': 'Tobias', 'year': 2007}, 'child3': {'name':
 'Linus', 'year': 2011}}
```

Pada contoh diatas kita membuat dictionary myfamily yang isinya adalah 3 item dengan kunci child1, child2, dan child3

Atau, jika kita ingin menambahkan tiga kamus ke dalam kamus baru juga bisa, perhatikan contoh berikut.

```python
child1 = {
  "name" : "Emil",
  "year" : 2004
}
child2 = {
  "name" : "Tobias",
  "year" : 2007
}
child3 = {
  "name" : "Linus",
  "year" : 2011
}

myfamily = {
  "child1" : child1,
  "child2" : child2,
  "child3" : child3
}

print(myfamily)
```

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
{'child1': {'name': 'Emil', 'year': 2004}, 'child2': {'name': 'Tobias', 'year': 2007}, 'child3': {'name':
 'Linus', 'year': 2011}}
```

Pada contoh diatas hasilnya sama juga dengan contoh sebelumya.

Latihan 3.3

1. Buatlah sebuah dictionary yang berisi biodata masing-masing praktikan
   - Nama
   - Nim
   - Kelas
   - Jurusan

   Kemudian tambahkan alamat pada dictionry tersebut

Praktikum 4

Judul : preprocessing data

Deskripsi :

Preprocessing data adalah tahap awal dalam proses data mining yang sangat penting. Tujuannya adalah untuk membersihkan, mengubah, dan mempersiapkan data mentah menjadi format yang lebih mudah dipahami dan diproses oleh algoritma data mining.

Python adalah bahasa pemrograman yang sangat populer untuk melakukan preprocessing data dalam data mining. Ada beberapa teknik preprocessing data yang dapat dilakukan menggunakan Python, di antaranya:

1. Data Cleaning: Teknik untuk membersihkan data mentah dari noise dan missing values. Beberapa library Python seperti pandas dan numpy menyediakan fungsi untuk mengatasi masalah ini.
2. Data Transformation: Teknik untuk mengubah format data menjadi format yang lebih mudah diproses oleh algoritma data mining. Beberapa contoh transformasi data adalah normalisasi data, encoding kategori data, dan pengurangan dimensi data.
3. Data Integration: Teknik untuk menggabungkan data dari berbagai sumber menjadi satu. Library pandas menyediakan fitur untuk menggabungkan data dari berbagai sumber.
4. Data Reduction: Teknik untuk mengurangi ukuran data agar lebih mudah diproses oleh algoritma data mining. Contoh teknik ini adalah feature selection dan feature extraction.
5. Data Discretization: Teknik untuk mengubah data kontinu menjadi data diskrit. Contoh teknik ini adalah binning, clustering, dan decision tree.

Setelah melakukan preprocessing data, data siap untuk diproses menggunakan algoritma data mining yang sesuai. Dalam Python, banyak library seperti Scikit-learn dan TensorFlow yang menyediakan berbagai algoritma data mining yang dapat digunakan.

1. Modul dalam python
   Modul adalah file yang berisi sekumpulan fungsi dan kode yang ingin Anda simpan dalam format .py dan ingin anda sertakan dalam aplikasi Anda.

MEMBUAT MODUL

Untuk membuat modul cukup simpan kode yang Anda inginkan dalam file dengan ekstensi file .py: Contoh, simpan kode dibawah ini sebagai **mymodule.py**.

```
mymodule.py > greeting
1   def greeting(name):
2       print("Hello, " + name)
3       person1 = {
4       "name": "John",
5       "age": 36,
6       "country": "Norway"
7   }
```

Modul mymodule.py ini kita buat memiliki satu buah fungsi yaitu greeting yang memiliki satu argumen. Dan juga satu buah variabel global yakni sebuah dictiionary yang bernama person1 yang memiliki 3 item dengan kunci secara berurutan name, age, dan country.

MENGIMPORT DAN MENGGUNAKAN MODUL

Sekarang kita dapat menggunakan modul yang baru saja kita buat (mymodule.py), dengan menggunakan pernyataan import. Perhatikan contoh berikut.

```
1   import mymodule
2
3   mymodule.greeting("Jonathan")
```

Output

```
PS D:\praktikum data mining\python> python modul.py
Hello, Jonathan
```

Saat menggunakan fungsi dari modul, gunakan sintaks: module_name.function_name.

Pada contoh diatas kode *import mymodule*, artinya adalah kita mengimport modul mymodule.py. Ingat, kita tidak perlu menuliskan ".py"-nya. Lalu kode *mymodule.greeting("jonathan")*, artinya adalah kita mencoba mengeksekusi fungsi greeting yang ada di dalam modul mymodule. Hasilnya kita bisa lihat di console.

Modul dapat berisi fungsi, seperti yang telah dijelaskan, tetapi juga variabel dari semua jenis (array, kamus, objek, dll). Untuk mengaksesnya adalah sama persis seperti mengakses fungsi. Berikut contoh untuk mengakses varibel person1 dari module mymodule.

```
PS D:\praktikum data mining\python> python modul.py
36
```

Anda juga dapat menamai ulang module yang diimport dengan kata kunci as. Biasanya digunakan nama yang lebih singkat.

```
1   import mymodule as mx
2
3   a = mx.person1["age"]
4   print(a)
```

Ada beberapa modul bawaan dalam Python, yang dapat Anda impor kapan pun Anda mau seperti platform, dan lain-lain.
Untuk melihat list dari semua nama fungsi dan variabel dari suatu module kita dapat menggunakan fungsi dir().

From nama_module import nama_fungsi

Perhatikan contoh berikut. Kita mau mencoba mengimport variable person1 dari module mymodule.

```
1   from mymodule import person1
2
3   print(person1["age"])
```

Perlu dicatat, saat mengimport hanya satu fungsi atau variabel dari suatu modul dengan syntaks from, untuk memanggil fungsi atau variabel tersebut langsung panggil saja tanpa perlu merujuk ke nama modulenya.

Latihan 3.4

a. Buatlah module yang berisi fungsi untuk menampilkan nama, nim dan kelas masing-masing
b.

2. PIP

ython. PIP adalah sebuah Python, dan memudahkan i paket Python.

telah dikembangkan oleh luas fungsionalitas Python. tidak perlu men-download

etahui oleh praktikan Python:

1. Instalasi library menggunakan PIP
   Untuk menginstal sebuah library, dapat menggunakan perintah "pip install nama_library". Contoh: untuk menginstal library Numpy, gunakan perintah "pip install numpy".
2. Menghapus library menggunakan PIP
   Untuk menghapus sebuah library yang sudah terinstal, dapat menggunakan perintah "pip uninstall nama_library". Contoh: untuk menghapus library Numpy, gunakan perintah "pip uninstall numpy".
3. Melihat daftar library yang sudah terinstal menggunakan PIP
   Untuk melihat daftar library yang sudah terinstal, dapat menggunakan perintah "pip list".
4. Mencari informasi tentang sebuah library menggunakan PIP

Untuk mencari informasi tentang sebuah library, dapat menggunakan perintah "pip search nama_library". Contoh: untuk mencari informasi tentang library Numpy, gunakan perintah "pip search numpy".

5. Memperbarui library menggunakan PIP

   Untuk memperbarui sebuah library yang sudah terinstal, dapat menggunakan perintah "pip install --upgrade nama_library". Contoh: untuk memperbarui library Numpy, gunakan perintah "pip install --upgrade numpy".

PIP adalah alat yang sangat berguna bagi praktikan Python untuk menginstal dan mengelola library Python yang diperlukan untuk proyek atau tugas mereka. Sebaiknya selalu menginstal library melalui PIP untuk memastikan library yang terinstal up-to- date dan terintegrasi dengan baik dalam lingkungan Python.

3. File Handling

File handling adalah teknik mengoperasikan file baik itu membaca, menulis, menghapus, dan mengedit. Ini sangat penting dan wajib untuk dipelajari.

MEMBACA FILE

Asumsikan kita memiliki file berikut, terletak di folder yang sama dengan file Python yang mau anda run.

Nama file : *demofile.txt*
Isi : "python itu mudah"

Untuk membuka file tersebut (mengimport), di python kita bisa menggunakan fungsi open() dan lalu dengan fungsi read(). Ini adalah fungsi bawaannya python jadi kita tidak perlu mengimport modul apapun.

```
1   f = open("demofile.txt", "r")
2
3   print(f.read())
4   f.close()
```

```
PS D:\praktikum data mining\python> python modul.py
python itu mudah
```

Jika file berada di folder atau directory yang berbeda maka, kita harus tuliskan full path nya.

Secara default fungsi read() mengembalikan seluruh teks, tetapi Anda juga dapat menentukan berapa banyak karakter yang ingin Anda kembalikan.

Sebagai catatan, setelah mengoperasikan file, kita juga harus menutupnya dengan fungsi close()

```
f = open("demofile.txt", "r")
print(f.read(5))
f.close()
```

```
PS D:\praktikum data mining\python> python modul.py
pytho
```

MEMBACA FILE PER BARIS

Kita dapat mengembalikan satu baris dengan menggunakan fungsi**readline()**.

Buat file Pancasila.txt denga nisi adalah 5 sila Pancasila

pancasila.txt

```
1. Ketuhanan yang Maha Esa
2. Kemanusiaan yang adil dan beradab
3. Persatuan Indonesia
4. Kerakyatan yang dipimpin oleh hikmat kebijaksanaan dalam permusyawaratan/perwakilan, serta
5. Keadilan sosial bagi seluruh rakyat Indonesia
```

```
f = open("pancasila.txt", "r")
print(f.readline())
f.close()
```

```
PS D:\praktikum data mining\python> python modul.py
1. Ketuhanan yang Maha Esa
```

Dengan memanggil readline() dua kali, kita dapat membaca dua baris pertama dari file yang didefinisikan.

```
f = open("pancasila.txt", "r")
print(f.readline())
print(f.readline())
f.close()
```

```
PS D:\praktikum data mining\python> python modul.py
1. Ketuhanan yang Maha Esa

2. Kemanusiaan yang adil dan beradab
```

Selain menggunakan fungsi readline(), untu membaca file baris per baris, kita juga bisa menggunakan looping for in. Perhatikan contoh berikut.

```
1   f = open("pancasila.txt", "r")
2   for x in f:
3       print(x)
4   f.close()
```

```
PS D:\praktikum data mining\python> python modul.py
1. Ketuhanan yang Maha Esa

2. Kemanusiaan yang adil dan beradab

3. Persatuan Indonesia

4. Kerakyatan yang dipimpin oleh hikmat kebijaksanaan dalam permusyawaratan/perwakilan, serta

5. Keadilan sosial bagi seluruh rakyat Indonesia
```

MENAMBAHKAN KONTEN KE FILE YANG ADA

Untuk menulis ke file yang ada, maka sintaks

```
f=open(“namafile.txt”,                                “a”)
f.write("kontan yang baru")
```

Perhatikan pada contoh sebeumnya, di fungsi open, kita menggunakan parameter “r”, yang artinya adalah read-only.

Latihan 3.5

1. Buatlah sebuah file yang berisi biodata masing-masing yang berisi
   - Nama
   - Nim
   - Kelas
   - Jurusan
   - Alamat
   - Hoby

   Kemudian tampilkan dengan python

MODUL 3
(MINGGU KETIGA)

Praktikum 5

Numpy

NumPy adalah salah satu paket paling penting di Python untuk pengolahan data numerik. Paket ini menyediakan struktur data array multidimensi, fungsi matematika yang kuat, dan algoritma untuk memanipulasi array. Untuk menginstall numpy, kita bisa dengan mudah menggunakan PIP .

pip install `numpy`

Lalu untuk menggunakan numpy kita bisa dengan mudah mengimportnya di file koding kita.

```
import numpy as np
```

Pada contoh diatas kita mengimport numpy dan lalu membuat alias sebagai np. Jadi di kodingan kita nantinya untuk memanggil numpy cukup dengan np saja

PERBEDAAN NUMPY DENGAN LIST

NumPy memberikan berbagai cara cepat dan efisien untuk membuat array dan memanipulasi data numerik di dalamnya. Sementara list Python dapat berisi tipe data yang berbeda dalam satu daftar, semua elemen dalam array NumPy harus homogen. Operasi matematika yang dimaksudkan untuk dilakukan pada array akan sangat tidak efisien jika array tidak homogen. Juga di list, tidak banyak fungsi yang bisa digunakan untuk operasi array matematika.

Array NumPy lebih cepat dan lebih ringkas daripada List bawaan Python. Sebuah array dari numpy mengkonsumsi lebih sedikit memori dan nyaman untuk digunakan. NumPy menggunakan lebih sedikit memori untuk menyimpan data dan menyediakan mekanisme untuk menentukan tipe data. Ini memungkinkan kode untuk dioptimalkan lebih jauh.

MEMBUAT ARRAY NUMPY.

Array adalah struktur data dari library NumPy. Array dapat diindeks oleh bilangan bulat non-negatif, dengan boolean, oleh array lain, atau dengan bilangan bulat. Jumlah dimensi dari suatu array biasa juga kita sebut rank. Sebagaimana dalam matematika, kita sudah mengenal array atau matrix 1D, 2D, 3D, dan seterusnya. Namun yang umum digunakan

adalah yang 2D. Bentuk array adalah list bilangan bulat yang memberikan ukuran array sepanjang setiap dimensi.

Salah satu cara kita dapat menginisialisasi array NumPy adalah dari list Python, menggunakan list bersarang (nested list) untuk data dua dimensi atau lebih tinggi. Perhatikan contoh berikut.

```
tes.py > ...
1   import numpy as np
2   arr = np.array([1, 2, 3, 4, 5])
3   print(arr)
```

Pada contoh diatas kita membuat array 1D dengan numpy. Perhatikan pada kode np.array([1,2,3,4,5]), ini adalah sintaks untuk membuat array 1D dengan data yang sudah ditentukan.

Lalu untuk membuat array 2 dimensi, kita dapat menggunakan cara seperti pada contoh berikut:

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

Output

```
PS C:\praktikum> python .\tes.py
[[1 2 3]
 [4 5 6]]
```

Pada contoh diatas kita membuat array 2D, yang jika digambarkan lebih tervisual, seperti pada gambar diawah ini.

1	2	3
4	5	6

Untuk memriksa jumlah dimensi (rank) dari suatu array kita dapat mengguakan fungsi ndims(). Perhatikan contoh berikut.

```
tes.py > ...
1   import numpy as np
2   a = np.array(42)
3   b = np.array([1, 2, 3, 4, 5])
4   c = np.array([[1, 2, 3], [4, 5, 6]])
5   d = np.array([[[1, 2, 3], [4, 5, 6]], [[1,2, 3], [4, 5, 6]]])
6   print(a.ndim)
7   print(b.ndim)
8   print(c.ndim)
9   print(d.ndim)
```

Output

```
PS C:\praktikum> python .\tes.py
0
1
2
3
```

Selain mengetahui jumlah dimensinya, kita juga bisa mengetahui bentuk dari array tersebut dengan fungsi shape. Dengan fungsi shape, kita bisa mengetahui jumlah elemen dari array

```
tes.py > ...
1   import numpy as np
2   arr = np.array([[1, 2, 3, 4], [5, 6, 7, 8]])
3   print(arr.shape)
```

Output

```
PS C:\praktikum> python .\tes.py
(2, 4)
```

MENGAKSES ARRAY

Kita dapat mengakses elemen array dengan mengacu pada nomor indeksnya. Indeks dalam array NumPy dimulai dengan 0, artinya elemen pertama memiliki indeks 0, dan yang kedua memiliki indeks 1 dll. Perhatikan contoh berikut unttuk mengakses indeks ke 0 (elemen ke 1).

```
tes.py > ...
1   import numpy as np
2   arr = np.array([1, 2, 3, 4])
3   print(arr[0])
```

Output

```
PS C:\praktikum> python .\tes.py
1
```

Contoh yang lain, kita coba jumlahkan array indeks ke 2 dan ke 3 (elemen ke 3 dan 4).

```
tes.py > ...
1   import numpy as np
2   arr = np.array([1, 2, 3, 4])
3   print(arr[2] + arr[3])
```

Output

```
PS C:\praktikum> python .\tes.py
7
```

Untuk mengakses elemen dari array 2-D, kita dapat menggunakan bilangan bulat yang dipisahkan koma yang mewakili dimensi dan indeks elemen. Pikirkan array 2-D seperti tabel dengan baris dan kolom, di mana baris mewakili dimensi dan indeks mewakili kolom.

```
tes.py > ...
1   import numpy as np
2   arr = np.array([[1,2,3,4,5],
3   [6,7,8,9,10]])
4
5   print('Pada baris 1 dan kolom 2: ', arr[0, 1])
```

Output

```
PS C:\praktikum> python .\tes.py
Pada baris 1 dan kolom 2:  2
```

Pada contoh diatas, kode arr[0,1] artinya adalah kita sedang mengakses array arr pada elem baris ke 1 dan kolom ke 2.

Kita juga bisa menggunakan pengindeksan negatif untuk mengakses array dari akhir. Coba perhatikan contoh berikut.

```
tes.py > ...
1   import numpy as np
2   arr = np.array([[1,2,3,4,5], [6,7,8,9,10]])
3
4   print('pada baris terakhir, yang paling brlakang adalah ', arr[1, -1])
```

Output

```
PS C:\praktikum> python .\tes.py
pada baris terakhir, yang paling brlakang adalah  10
```

Pada contoh diatas, kode arr[1,-1] mempunyai arti yaitu kita mencoba mengakses array arr pada baris ke 2 dan kolom ke 1 dari akhir. Sehingga ini sama saja dengan kode arr[1,4].

MENGIRIS ARRAY

Kita juga dapat mengambil beberapa bagian dari array dengan range dan interval tertentu.

```
tes.py > ...
1   import numpy as np
2   arr = np.array([1, 2, 3, 4, 5, 6, 7])
3   print(arr[1:5])
```

Perhatikan contoh berikut.

Output

```
PS C:\praktikum> python .\tes.py
[2 3 4 5]
```

Pada contoh diatas perhatikan pada kode arr[1:5], ini artinya adalah kita mengambil array arr dari indeks ke 1 – 4. Ingat, indeks ke-5 (elemen ke 6) tidak termasuk ya. Kalian bisa buka kembali pada bab list untuk mengingat kembali terkait range ini.

Perhatikan lagi contoh dibawah ini. Di contoh ini, kita ingin mengambil array dari indeks ke 4 sampai akhir.

```
tes.py > ...
1   import numpy as np
2   arr = np.array([1, 2, 3, 4, 5, 6, 7])
3   print(arr[4:])
```

Output

```
PS C:\praktikum> python .\tes.py
[5 6 7]
```

Perhatikan lagi contoh dibawah ini, kita akan mencoba mengambil dari indeks pertama sampai indeks ke 3.

```
tes.py > ...
1   import numpy as np
2   arr = np.array([1, 2, 3, 4, 5, 6, 7])
3   print(arr[:4])
```

Output

```
PS C:\praktikum> python .\tes.py
[1 2 3 4]
```

Kita juga bisa menggunakan interval nilai untuk menentukan langkah pemotongan. Secara default nilai interval adalah 1, artinya adalah setiap langkah sampel adalah naik 1 angka.

```
tes.py > ...
1   import numpy as np
2   arr = np.array([1, 2, 3, 4, 5, 6, 7])
3   print(arr[1:5:2])
```

Output

```
PS C:\praktikum> python .\tes.py
[2 4]
```

Pada contoh diatas, pada kode arr[1:5:2] artinya adalah kita akan mengambil dari indeks ke 1 sampai indeks ke 4 dengan interval 2 sehinga hasilnya adalah [2 4].

Perhatikan lagi contoh ini.

```
tes.py > ...
1   import numpy as np
2   arr = np.array([1, 2, 3, 4, 5, 6, 7])
3   print(arr[::2])
```

Output

```
PS C:\praktikum> python .\tes.py
[1 3 5 7]
```

Pada contoh diatas kode arr[::2] artinya adalah kita akan mengambil dari indeks pertama sampai akhir dengan interval 2.

Cara mengiris dengan range dan interval ini juga bisa digunaka pada array 2D ataupun lainnya. Berikut contoh penggunaannya untuk array 2D, perhatikan conoth dibawah ini.

```
tes.py > ...
1   import numpy as np
2   arr = np.array([[1, 2, 3, 4, 5], [6, 7, 8, 9, 10]])
3   print(arr[1, 1:4])
```

Output

```
PS C:\praktikum> python .\tes.py
[7 8 9]
```

Pada contoh diatas kode arr[1,1:4], ini artinya adalah kita akan mengambil bagian dari array arr yaitu untuk row indeks 1, dan kolom indeks 1 sampai 3. Oleh karena hanya 1 row yang diambil maka hasilnya adalah 1 dimensi.

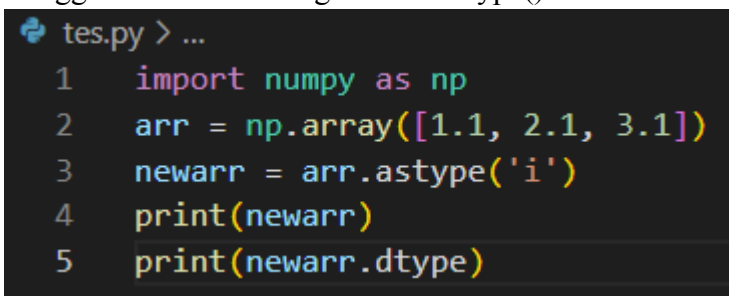
TIPE DATA PADA ARRAY NUMPY

Di bawah ini adalah daftar semua tipe data di NumPy dan karakter yang digunakan untuk mewakilinya.

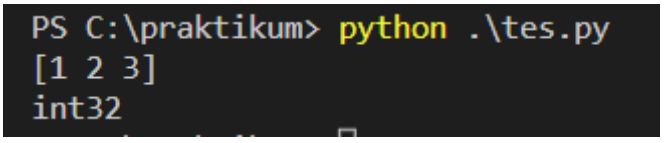
- i : bilangan Bulat
- b : Boolean
- u : bilangan bulat tak bertanda
- f : mengambang
- c : pelampung kompleks
- m : delta waktu
- M : tanggal Waktu
- O : objek
- S : rangkaian
- U : string unicode
- V : potongan memori tetap untuk tipe lain (void)

Dalam satu array hanya boleh satu tipe data. Tidak boleh ada tipe data lain, jika ada tipe data lain, maka python akan error.

Untuk mengubah tipe data, misalkan dari float ke integer, suatu array, maka kita dapat menggunakan fungsi astype(). Perhatikan contoh berikut.



Output



Perhatikan pada contoh diatas. Kode arr.astype('i') artinya adalah kita mencoba mengubah tipe data array arr yang awalnya adalah float (lihat pada kode ke dua, disitu sangat jelas bahwa bilangan desimal). Lalu kita bisa cek apakah tipe datanya benar sudah berubah atau belum dengan kode newarr.dtype. Dengan kode ini kita bisa tau tipe dataya.

MENYALIN ARRAY

Untuk membuat salinan dari suatu array kita dapat menggunakan fungsi **copy()**. Hasil dari fungsi ini adalah array baru, dan tidak ada kaitannya dengan array sebelumnya. Artinya

jika ada perubahan pada array baru tersebut, array yang disalin tidak akan terpengaruh.

```
tes.py > ...
1   import numpy as np
2   arr = np.array([1, 2, 3, 4, 5])
3   x = arr.copy()
4   arr[0] = 42
5   print(arr)
6   print(x)
```

Output

```
PS C:\praktikum> python .\tes.py
[42  2  3  4  5]
[1 2 3 4 5]
```

RESHAPE

Reshaping berarti mengubah bentuk array. Bentuk array adalah jumlah elemen dalam setiap dimensi. Dengan membentuk kembali kita dapat menambah atau menghapus dimensi atau mengubah jumlah elemen di setiap dimensi.
Contoh kita ingin mengubah array 1D menjadi 2D, perhatikan contoh berikut:

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

Output

```
PS C:\praktikum> python .\tes.py
[[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [10 11 12]]
```

Perhatikan pada contoh diatas kode arr.reshape(4,3) artinya adalah kita ingin mengubah array arr menjadi array dengan ukuran 4,3 yaitu jumlah baris 4 dan kolom 3. Sehingga hasilnya adalah sebagaimana contoh diatas. Sedangkan di awal array arr adalah array 1D dengan jumlah elemen 12.

Yang perlu diperhatikan saat reshape adalah jumlah elemen harus sama. Misalkan pada contoh sebelumnya, awalnya array adalah 1D dengan jumlah elemen 12, lalu diubah menjadi array 2D dengan dimensi 4x3 yang artinya adalah total elemennya adalah 12 juga. Apabila jumlah elemennya ini berbeda maka akan error. Lihat contoh berikut.

```
tes.py > ...
1   import numpy as np
2   arr = np.array([1, 2, 3, 4, 5, 6, 7, 8])
3   newarr = arr.reshape(3, 3)
4   print(newarr)
```

Output

```
PS C:\praktikum> python .\tes.py
Traceback (most recent call last):
  File ".\tes.py", line 3, in <module>
    newarr = arr.reshape(3, 3)
ValueError: cannot reshape array of size 8 into shape (3,3)
```

Pada contoh diatas kita mau mencoba mengubah array 1D dengan jumlah elemen 8, menjadi array 2D dengan dimensi 3x3 yang artinya mempunyai jumlah elemen 9. Oleh karena berbeda jumlah elemennya maka python error.

Terkadang kita ingin mengubah suatu array ke dalam bentuk array lain namun hanya dimensi tertentu doang yang anda ingin anda benar-benar wajib pastikan sedangkan dimensi yang lain anda bebaskan. Misalkan kita punya array 1D. Lalu kita ingin mengubahnya ke dalam bentuk 2D, dengan dimensi 1 nya 2 baris, dengan jumlah kolomya bebas. Maka ini bisa kita lakukan dengan menggunakan angka **-1**. Perhatikan contoh berikut

```
tes.py > ...
1   import numpy as np
2   arr = np.array([1, 2, 3, 4, 5, 6, 7, 8])
3   newarr = arr.reshape(2, -1)
4   print(newarr)
```

Output

```
PS C:\praktikum> python .\tes.py
[[1 2 3 4]
 [5 6 7 8]]
```

Pada contoh diatas kita, kode arr.reshape(2,-1) artinya adalah kita akan mengubah bentuk arr menjadi array 2D dengan dimensi pertama jumlah barisnya 2 sedangkan untuk dimensi ke 2 yaitu kolomnya bebas. Kita serahkan ke python untuk menghitungnya sendiri.

Untuk mengubah array dari dimensi berapapun ke dimensi satu, kita bisa menggunakan sintaks

nama_arrynya.reshape(-1).

Perhatikan contoh berikut.

```
tes.py > ...
1   import numpy as np
2   arr = np.array([[1, 2, 3], [4, 5, 6]])
3   newarr = arr.reshape(-1)
4   print(newarr)
```

Output

```
PS C:\praktikum> python .\tes.py
[1 2 3 4 5 6]
```

ARRAY LOOP

Untuk mengakses satu per satu item array, kita bisa menggunakan for loop biasa. Perhatikan contoh berikut.

```
tes.py > ...
1   import numpy as np
2   arr = np.array([1, 2, 3])
3   for x in arr:
4       print(x)
```

Output

```
PS C:\praktikum> python .\tes.py
1
2
3
```

Pada array 2D, maka kita membutuhkan 2 buah for, karena jika hanya 1, maka kita hanya akan mengakses array baris per baris. Perhatikan contoh berikut.

```
tes.py > ...
1   import numpy as np
2   arr = np.array([[1, 2, 3], [4, 5, 6]])
3
4   for x in arr:
5       print(x)
```

Output

```
PS C:\praktikum> python .\tes.py
[1 2 3]
[4 5 6]
```

Untuk bisa mengakses satu per satu item, untuk array 2D, maka kita butuh 2 for, for yang pertama adalah untuk baris (dimensi ke 1), dan for yang kedua untuk kolom (dimensi ke 2). Perhatikan contoh berikut.

```
tes.py > ...
1   import numpy as np
2   arr = np.array([[1, 2, 3], [4, 5, 6]])
3   for x in arr:
4       for y in x:
5           print(y)
```

Output

```
PS C:\praktikum> python .\tes.py
1
2
3
4
5
6
```

MENGGABUNGKAN ARRAY

Bergabung berarti menempatkan isi dari dua atau lebih array dalam satu array. Kita dapat melakukannya dengan fungsi concatenate().

```
tes.py > ...
1   import numpy as np
2   arr1 = np.array([1, 2, 3])
3   arr2 = np.array([4, 5, 6])
4   arr = np.concatenate((arr1, arr2))
5   print(arr)
```

Output

```
PS C:\praktikum> python .\tes.py
[1 2 3 4 5 6]
```

Jika menggabungkan array 1D cukup mudah. Namun untuk menggabungkan array 2D, kita perlu definisikan juga, cara menggabungkannya secara vertikal (axis dimensi 1) atau secara horizontal (axis dimensi 2).
Sintaknya pun berbeda yaitu

np.concatenate((
arr1,arr2),axis=0
))
np.concatenate((
arr1,arr2),axis=1
))

Axis=1 artinya adalah penggabungan pada dimensi 1, sedangkan axis=1 artinya adalah penggabungan pada dimensi 2. Perhatikan contoh berikut.

```
tes.py > ...
1   import numpy as np
2   arr1 = np.array([[1, 2], [3, 4]])
3   arr2 = np.array([[5, 6], [7, 8]])
4   arr = np.concatenate((arr1, arr2), axis=1)
5   print(arr)
```

Output

```
PS C:\praktikum> python .\tes.py
[[1 2 5 6]
 [3 4 7 8]]
```

Pada contoh diatas kita menggabungkan dua array 2D pada arah dimensi 2 (secara horizontal). Perhatikan juga contoh dibawah ini, contoh menggabungkan array 2D pada arah dimensi 1.

Output

```
tes.py > ...
1   import numpy as np
2   arr1 = np.array([[1, 2], [3, 4]])
3   arr2 = np.array([[5, 6], [7, 8]])
4   arr = np.concatenate((arr1, arr2), axis=0)
5   print(arr)
```

```
PS C:\praktikum> python .\tes.py
[[1 2]
 [3 4]
 [5 6]
 [7 8]]
```

PENCARIAN

Kita dapat mencari array untuk nilai tertentu, dan mengembalikan indeks yang cocok. Untuk mencari array, kita bisa menggunakan fungsi where(). Perhatikan contoh berikut.

```python
import numpy as np
arr = np.array([1, 2, 3, 4, 5, 4, 4])
x = np.where(arr == 4)
print(x)
```

Output

```
PS C:\praktikum> python .\tes.py
(array([3, 5, 6], dtype=int64),)
```

Lihat pada contoh diatas, kode np.where(arr==4) artinya adalah kita mencari indeks-indeks pada array yang memiliki nilai sama dengan 4. Hasilnya adalah sebagiamana output contoh diatas.

SORTIR ARRAY

Objek array NumPy memiliki fungsi yang disebut sort(), yang akan mengurutkan array yang ditentukan. Perhatikan contoh berikut;

```python
import numpy as np

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

Output

```
PS D:\praktikum data mining\python> python .\array.py
[0 1 2 3]
```

OPERASI ARITMATIKA ARRAY

Untuk menjumlahkan konten dari dua array, dan mengembalikan hasilnya dalam array baru kita dapat menggunakan fungsi add().

```python
import numpy as np

arr1 = np.array([10, 11, 12, 13, 14, 15])
arr2 = np.array([20, 21, 22, 23, 24, 25])

newarr = np.add(arr1, arr2)
print(newarr)
```

Output

```
PS D:\praktikum data mining\python> python .\array_1.py
[30 32 34 36 38 40]
```

Contoh di atas menghasilkan [30 32 34 36 38 40] yang merupakan jumlah dari 10+20, 11+21, 12+22 dst. Untuk mengurangi nilai dari satu array dengan nilai dari array lain, dan mengembalikan hasilnya dalam array baru dapat menggunakan fungsi **subtract()**.

Untuk mengalikan nilai dari satu array dengan nilai dari array lain, dan mengembalikan hasilnya dalam array baru kita dapat menggunakan fungsi **multiply()**.

Untuk membagi nilai dari satu array dengan nilai dari array lain, dan mengembalikan hasilnya dalam array baru kita dapat menggunakan fungsi **divide()**.

Untuk mempangkatkan setiap elemen yang ada di suatu array dengan elemen pada posisi yang sama di array yang, kita dapat menggunakan fungsi **power()**.

Untuk mendapatkan nilai absolut dari suatu array kita dapat menggunakan fungsi **absolute()** atau **abs()**.

Untuk melakukan pembulatan untuk setiap elemen pada suatu array, kita dapat menggunakan fungsi **around()**.

Untuk mendapatkan nilai total penjumlah keseluruhan elemen yang ada di dalam suatu array kita bisa menggunakan fungsi **sum()**.

Untuk menemukan elemen-elemen yang unique, elemen yang ada kembaran maka hanya akan diambil satu, kita dapat menggunakan fungsi **unique()**.

Latihan 3.1

1. Buatlah sebuah array dengan isi adalah nim masing-masing praktikan menggunakan lbrary NumPy,
   Kemudian tampilkan
   a) Total jumlah nim
   b) Nilai rata-rata nim
   c) Angka paling besar
   d) Angka paling kecil
   e) Angka unik saja
   f) Urutkan dari yang besar ke kecil

PANDAS

Pandas adalah salah satu library paling populer di Python yang digunakan untuk melakukan manipulasi, analisis, dan visualisasi data. Pandas menyediakan struktur data yang fleksibel dan efisien, yaitu Dataframe dan Series, yang memungkinkan pengguna untuk bekerja dengan data tabular (seperti spreadsheet) dengan mudah dan cepat.

Berikut adalah beberapa fitur dan fungsi yang disediakan oleh library pandas:

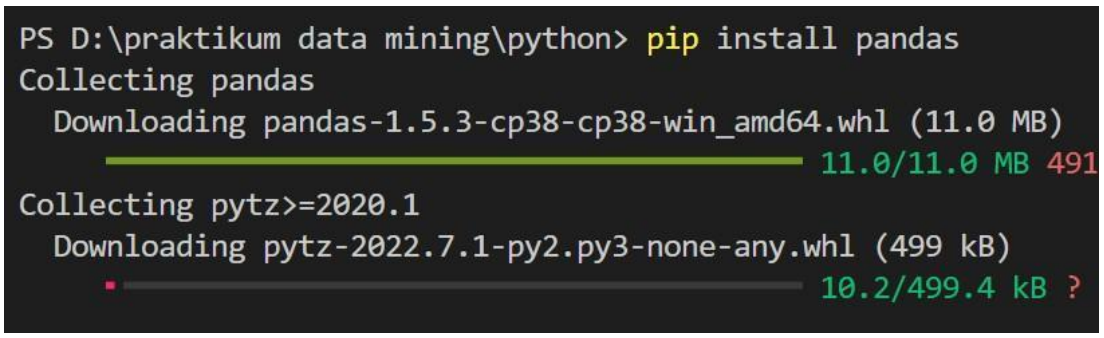
1. Dataframe: Dataframe adalah struktur data 2 dimensi (baris dan kolom) yang dapat menyimpan berbagai tipe data. Dataframe memungkinkan pengguna untuk membaca data dari berbagai format file, seperti CSV, Excel, SQL, dan lainnya.
2. Series: Series adalah struktur data 1 dimensi (hanya memiliki baris) yang merupakan bagian dari Dataframe. Series dapat digunakan untuk melakukan operasi pada satu kolom dari Dataframe.
3. Data Cleaning: Pandas menyediakan banyak fungsi untuk membersihkan data yang tidak lengkap atau tidak valid, seperti menghapus duplikat, mengisi nilai yang hilang, dan mengubah format data.
4. Data Manipulation: Pandas memungkinkan pengguna untuk melakukan manipulasi data dengan mudah, seperti menggabungkan data dari beberapa sumber, memfilter data, dan mengubah tipe data.
5. Data Analysis: Pandas menyediakan banyak fungsi untuk melakukan analisis data, seperti menghitung nilai rata-rata, maksimum, minimum, median, dan lainnya.
6. Data Visualization: Pandas dapat digunakan untuk membuat visualisasi data yang menarik dengan menggunakan library seperti Matplotlib dan Seaborn.

Pandas juga memiliki dokumentasi yang lengkap dan tutorial yang dapat membantu pengguna untuk memulai dengan library ini. Pandas merupakan library yang sangat penting bagi pengguna Python yang ingin melakukan analisis data dan memanipulasi data secara efisien dan cepat.

Pandas adalah library open source. Kita bisa mengecek repository kodingnya di https://github.com/pandas- dev/pandas.

INSTALASI

Untuk menginstall pandas, kita dapat melakukannya dengan sangat mudah menggunakan PIP



Setelah pandas kita install, untuk menggunakannya kita dapat mengimportnya di kodingan kita dengan perintah import pandas.

import pandas as pd

Pada contoh diatas, kita gunakan alias as pd. Ini untuk mempermudah kita membuat kodingan saja. Jadi saat mau mengguakan pandas, kita cukup menggunakan aliasnya saja yaitu pd.

SERIES

Series, di pandas adalah array 1D yang nantinya akan kita oleh. Perhatikan contoh berikut. Pada contoh ini, kita akan mengubah List menjadi series pandas.

Praktikum 6

Pandas

Pandas adalah library pada python yang memungkinkan kita untuk menganalisis data besar dan membuat kesimpulan berdasarkan teori statistik. Pandas dapat membersihkan kumpulan data yang berantakan, dan membuatnya dapat dibaca dan relevan. Data yang relevan sangat penting dalam ilmu data. Jika kamu punya tujuan untuk menjadi data engineer, mempelajari modul ini adalah tepat untuk kamu. Pandas wajib kamu pelajari.

Pandas adalah library open source. Kita bisa mengecek repository kodingnya di https://github.com/pandas-dev/pandas.

INSTALASI

Untuk menginstall pandas, kita dapat melakukannya dengan sangat mudah menggunakan PIP.

```
D:\MPP>pip install pandas
```

Setelah pandas kita install, untuk menggunakannya kita dapat mengimportnya di kodingan kita dengan perintah import pandas.

Contoh

```
import pandas as pd
```

Pada contoh diatas, kita gunakan alias as pd. Ini untuk mempermudah kita membuat kodingan saja. Jadi saat mau mengguakan pandas, kita cukup menggunakan aliasnya saja yaitu pd.

SERIES

Series, di pandas adalah array 1D yang nantinya akan kita oleh. Perhatikan contoh berikut. Pada contoh ini, kita akan mengubah List menjadi series pandas.

Contoh: https://s.id/mpp_ex253

```
import pandas as pd a =

 [1, 7, 2]

myvar = pd.Series(a)

print(myvar)
```

Output

```
0    1
1    7
2    2
dtype:
```

Pada contoh diatas, pada kolom pertama ada angka 0 1 2, itu adalah label indeks penomeran baris saja.

Kita dapat mengakses series dengan memanggil indeksnya dalam kurung siku. Perhatikan contoh berikut. Pada contoh ini kita akan memanggil indeks 0 pada series myvar.

Contoh: https://s.id/mpp_ex254

```
s as pd a =

[1, 7, 2]

myvar = pd.Series(a)


print(myvar[0])
```

Output

```
1
```

Sebenarnya kita bisa menentukan sendiri label untuk indeks pada series. Berikut contohnya.

Contoh: https://s.id/mpp_ex255

```
ndas as pd


a = [1, 7, 2]
```

```
myvar = pd.Series(a, index = ["x", "y", "z"])

print(myvar)
```

Output

```
x    1
y    7
z    2
dtype:
```

Pada contoh diatas, lihat pada kode pd.Series(a,index=[“x”, “y”, “z”]), disitu kita menentukan penamaan label indeks kita sendiri. Namun yang perlu dicatat adalah jumlah indeks yang kita tentukan harus tepat sama dengan jumlah komponen pada list yang akan kita jadikan series.

Dengan label indeks yang kita buat sendiri, untuk mengakses series tersebut, kita panggil menggunakan label tersebut juga. Perhatikan contoh berikut.

Contoh: https://s.id/mpp_ex256

```
ndas as pd

a = [1, 7, 2]

myvar = pd.Series(a, index = ["x", "y", "z"])

print(myvar["y"])
```

Output

```
7
```

Pada contoh diatas kita memanggil item di series yang memiliki indeks y.

Jika memang tujuan kita adalah kita ingin membuat label indeks series susuai yang kita mau, kita sebenarnya bisa menggunakan dictionary. Perhatikan contoh berikut.

Contoh: https://s.id/mpp_ex257

```
import pandas as pd

calories = {"day1": 420, "day2": 380, "day3": 390}

myvar = pd.Series(calories)

print(myvar)
```

Output

```
day1    420
day2    380
day3    390
dtype: int64
```

Pada contoh diatas, kita bisa melihat dictionary calories. Saat diubah menjadi series, maka secara otomatis kunci pada calories akan menjadi label indeks.

DATAFRAME

Kumpulan data di Pandas biasanya berupa tabel multidimensi, yang disebut DataFrames. Jika series adalah tabel 1D, maka untuk tabel 2D adalah dataframe. Perhatikan contoh berikut.

Contoh: https://s.id/mpp_ex258

```
import pandas as pd

data = {
"calories": [420, 380, 390],
"duration": [50, 40, 45]
}

myvar = pd.DataFrame(data)

print(myvar)
```

Output

```
   calories
duration 0   420  50
1   380  40
```

Pada contoh diatas kita membuat dataframe pandas dengan menconvert directory yang masing-masing itemnya bernilai list atau array 1D.

Untuk mengakses data dataframe untuk baris tertentu, kita bisa menggunakan fungsi loc[indeksnya]. Perhatikan contoh berikut.

Contoh: https://s.id/mpp_ex259

```
import pandas as pd

data = {
"calories": [420, 380, 390],
"duration": [50, 40, 45]
}

#load data into a DataFrame object: df =
 pd.DataFrame(data)

print(df.loc[0])
```

Output

Pada contoh diatas, kode df.loc[0] artinya adalah kita mencoba mengakses dataframe df baris ke 1 (karena baris ke 1 adalah indeks ke 0). Kita bisa lihat hasilnya adalah calories 420, dan duration 50.

Dengan fungsi loc ini, kita juga bisa mengambil beberapa baris sekaligus. Perhatikan contoh berikut.

Contoh: https://s.id/mpp_ex260

```
import pandas as pd


data = {
"calories": [420, 380, 390],
"duration": [50, 40, 45]
}


#load data into a DataFrame object: df =
 pd.DataFrame(data)


print(df.loc[[0, 1]])
```

Output

```
   calories
duration 0   420  50
1   380  40
```

Pada contoh diatas, kode df.loc[[0,1]] berarti adalah kita mengambil dataframe df pada baris ke 1 dan 2.

BERINTERAKSI DENGAN FILE CSV

Salah satu keunggulan pandas, adalah kemudahannya berinteraksi dengan file csv. Kita dapat membuka file csv dan menjadikannya sebagai dataframe menggunakan fungsi pd.read_csv('nama_file.csv'). Perhatikan contoh berikut.

Contoh

```
import pandas as pd
df = pd.read_csv('data.csv') print(df)
```

Output

```
     Duration  Pulse  Maxpulse  Calories
0          60    110       130     409.1
1          60    117       145     479.0
2          60    103       135     340.0
3          45    109       175     282.4
4          45    117       148     406.0
..        ...    ...       ...       ...
164        60    105       140     290.8
165        60    110       145     300.4
166        60    115       145     310.2
167        75    120       150     320.4
168        75    125       150     330.4

[169 rows x 4 columns]
```

Jika Anda memiliki DataFrame besar dengan banyak baris, Pandas hanya akan memprintout 5 baris pertama, dan 5 baris terakhir saja di command line. Namun kita bisa mengubah jumlah maksimum ini dengan fungsi

pd.options.display.max_rows = 9999

Perhatikan contoh berikut

Contoh

```
import pandas as pd

 pd.options.display.max_rows = 9999 df =

 pd.read_csv('data.csv')

print(df)
```

Pada contoh ini, outputnya tidak kita tampilkan di buku ini karena jumlah barisnya ada 169 sehingga cukup panjang.

MEMBACA DATA JSON

Kita juga bisa mengubah data JSON menjadi dataframe. Data JSON ini sangat sering sekali digunakan untuk berinteraksi antar sistem, biasaya digenerate dari aplikasi web. File ini sebenarnya adalah string yang bentuknya seperti dictionary. Perhatikan contoh berikut.

Contoh

```
import pandas as pd


 data = { "Duration":{
"0":60,

"1":60,
"2":60,
"3":45,
```

```
"4":45,

"5":60
},
"Pulse":{
"0":110,
"1":117,
"2":103,
"3":109,
"4":117,
"5":102
},
"Maxpulse":{
"0":130,
"1":145,
"2":135,
"3":175,
"4":148,
"5":127
},
"Calories":{
"0":409.1,
"1":479.0,
"2":340.0,
"3":282.4,
"4":406.0,
"5":300.5
}
```

```
}
df = pd.DataFrame(data) print(df)
```

Output

	Duration	Pulse	Maxpulse	Calories
0	60	110	130	409.1
1	60	117	145	479.0
2	60	103	135	340.0
3	45	109	175	282.4
4	45	117	148	406.0
5	60	102	127	300.5

FUNGSI HEAD DAN TAIL

Metode head() adalah fungsi yang mengembalikan header dan sejumlah baris tertentu, mulai dari atas dari sebuah dataframe. Perhatikan contoh berikut.

Contoh

```
import pandas as pd
df = pd.read_csv('data.csv') print(df.head(10))
```

Output

	Duration	Pulse	Maxpulse	Calories
0	60	110	130	409.1
1	60	117	145	479.0
2	60	103	135	340.0

3	45	109	175	282.4
4	45	117	148	406.0
5	60	102	127	300.5
6	60	110	136	374.0
7	45	104	134	253.3
8	30	109	133	195.1
9	60	98	124	269.0

Pada contoh diatas, kode df.head(10) artinya adalah kita akan menampilkan header dataframe dan data 10 baris pertama. Ubah angka 10 menjadi angka yang kalian inginkan lalu perhatikan hasilnya. Jika angka 10 ini dikosongkan (tidak diisi angka), maka python akan menetapkan secara default yaitu 5.

Jika fungsi head adalah menampilkan data paling atas, maka fungsi tail(), adalah sebaliknya, ia akan menampilkan data paling terbawah.

Contoh

```
import pandas as pd
df = pd.read_csv('data.csv') print(df.tail(10))
```

Output

	Duration	Pulse	Maxpulse	Calories
159	30	80	120	240.9
160	30	85	120	250.4
161	45	90	130	260.4
162	45	95	130	270.0
163	45	100	140	280.9
164	60	105	140	290.8

165	60	110	145	300.4
166	60	115	145	310.2
167	75	120	150	320.4
168	75	125	150	330.4

MENGHILANGKAN BARIS TIDAK LENGKAP

Terkadang, di suatu dataset ada beberapa baris yang kolomya tidak lengkap. Kita dapat menghapus baris-baris yang tidak lengkap ini dengan fungsi dropna().

Sebelum kita bereksperimen dengan beberapa fungsi, kita tetapkan dataset kita yang akan kita pakai adalah data.csv yang memiliki data sebagaimana berikut.

data.csv

Duration	Date	Pulse	Maxpulse	Calories
0	60 '2020/12/01'	110	130	409.1
1	60 '2020/12/02'	117	145	479.0
2	60 '2020/12/03'	103	135	340.0
3	45 '2020/12/04'	109	175	282.4
4	45 '2020/12/05'	117	148	406.0
5	60 '2020/12/06'	102	127	300.0
6	60 '2020/12/07'	110	136	374.0
7	450 '2020/12/08'	104	134	253.3
8	30 '2020/12/09'	109	133	195.1
9	60 '2020/12/10'	98	124	269.0
10	60 '2020/12/11'	103	147	329.3
11	60 '2020/12/12'	100	120	250.7
12	60 '2020/12/12'	100	120	250.7
13	60 '2020/12/13'	106	128	345.3
14	60 '2020/12/14'	104	132	379.3

15	60	'2020/12/15'	98	123	275.0
16	60	'2020/12/16'	98	120	215.2
17	60	'2020/12/17'	100	120	300.0
18	45	'2020/12/18'	90	112	NaN
19	60	'2020/12/19'	103	123	323.0
20	45	'2020/12/20'	97	125	243.0
21	60	'2020/12/21'	108	131	364.2
22	45	NaN	100	119	282.0
23	60	'2020/12/23'	130	101	300.0
24	45	'2020/12/24'	105	132	246.0
25	60	'2020/12/25'	102	126	334.5
26	60	2020/12/26	100	120	250.0
27	60	'2020/12/27'	92	118	241.0
28	60	'2020/12/28'	103	132	NaN
29	60	'2020/12/29'	100	132	280.0
30	60	'2020/12/30'	102	129	380.3
31	60	'2020/12/31'	92	115	243.0

Sekarang coba kita perhatikan pada baris ke 18 22 dan 28, terdapat kolom yang kosong. Kita dapat menghapusnya secara otomatis menggunakan fungsi dropna(). Perhatikan contoh berikut.

Contoh

```
import pandas as pd


df = pd.read_csv('data.csv') new_df =

 df.dropna()

print(new_df.to_string())
```

Output

	Duration	Date	Pulse	Maxpulse	Calories
0	60	'2020/12/01'	110	130	409.1
1	60	'2020/12/02'	117	145	479.0
2	60	'2020/12/03'	103	135	340.0
3	45	'2020/12/04'	109	175	282.4
4	45	'2020/12/05'	117	148	406.0
5	60	'2020/12/06'	102	127	300.0
6	60	'2020/12/07'	110	136	374.0
7	450	'2020/12/08'	104	134	253.3
8	30	'2020/12/09'	109	133	195.1
9	60	'2020/12/10'	98	124	269.0
10	60	'2020/12/11'	103	147	329.3
11	60	'2020/12/12'	100	120	250.7
12	60	'2020/12/12'	100	120	250.7
13	60	'2020/12/13'	106	128	345.3
14	60	'2020/12/14'	104	132	379.3
15	60	'2020/12/15'	98	123	275.0
16	60	'2020/12/16'	98	120	215.2
17	60	'2020/12/17'	100	120	300.0
19	60	'2020/12/19'	103	123	323.0
20	45	'2020/12/20'	97	125	243.0
21	60	'2020/12/21'	108	131	364.2
23	60	'2020/12/23'	130	101	300.0
24	45	'2020/12/24'	105	132	246.0
25	60	'2020/12/25'	102	126	334.5
26	60	2020/12/26	100	120	250.0
27	60	'2020/12/27'	92	118	241.0
29	60	'2020/12/29'	100	132	280.0
30	60	'2020/12/30'	102	129	380.3
31	60	'2020/12/31'	92	115	243.0

Pada contoh diatas, kita bisa lihat indeks 18, 22, dan 28 sudah hilang secara otomatis. Perhatikan pada kode newdf.to_string(), kenapa diberi fungsi to_string()?. Fungsi to_string() ini memberikan isyarat ke python agar menampilkan semua data dataframe ke command line tanpa batasan.

MEREPLACE DATA PADA BARIS TIDAK LENGKAP

Ada kasus dimana kadang kita tidak ingin menghapus baris yang memiliki kolom tidak lengkap. Misalkan ada kolom kosong (NaN), maka kita ingin mengisinya secara otomatis dengan angka 0. Kita dapat menggunakan fungsi fillna().

Contoh

```
import pandas as pd


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

df.fillna(130, inplace = True)

print(df.to_string())
```

Output

	Duration	Date	Pulse	Maxpulse	Calories
0	60	'2020/12/01'	110	130	409.1
1	60	'2020/12/02'	117	145	479.0
2	60	'2020/12/03'	103	135	340.0
3	45	'2020/12/04'	109	175	282.4
4	45	'2020/12/05'	117	148	406.0
5	60	'2020/12/06'	102	127	300.0
6	60	'2020/12/07'	110	136	374.0
7	450	'2020/12/08'	104	134	253.3
8	30	'2020/12/09'	109	133	195.1
9	60	'2020/12/10'	98	124	269.0
10	60	'2020/12/11'	103	147	329.3
11	60	'2020/12/12'	100	120	250.7
12	60	'2020/12/12'	100	120	250.7
13	60	'2020/12/13'	106	128	345.3
14	60	'2020/12/14'	104	132	379.3
15	60	'2020/12/15'	98	123	275.0
16	60	'2020/12/16'	98	120	215.2
17	60	'2020/12/17'	100	120	300.0
18	45	'2020/12/18'	90	112	130.0
19	60	'2020/12/19'	103	123	323.0
20	45	'2020/12/20'	97	125	243.0

21	60	'2020/12/21'	108	131	364.2
22	45	130	100	119	282.0
23	60	'2020/12/23'	130	101	300.0
24	45	'2020/12/24'	105	132	246.0
25	60	'2020/12/25'	102	126	334.5
26	60	2020/12/26	100	120	250.0
27	60	'2020/12/27'	92	118	241.0
28	60	'2020/12/28'	103	132	130.0
29	60	'2020/12/29'	100	132	280.0
30	60	'2020/12/30'	102	129	380.3
31	60	'2020/12/31'	92	115	243.0

Pada contoh diatas kode df.fillna(130, inplace = True), ini artinya adalah pada dataframe df, untuk kolom yang NaN (kosong) akan kita isi dengan angka 130. Anda bisa ganti 130 dengan apapun yang kamu suka.

Sebenarnya kita juga bisa definisikan fungsi fillna() pada kolom tertentu saja, misalkan pada kolom calories. Perhatikan contoh berikut.

Contoh

```
import pandas as pd

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

df["Calories"].fillna(130, inplace = True)

print(df.to_string())
```

Output

	Duration	Date	Pulse	Maxpulse	Calories
0	60	'2020/12/01'	110	130	409.1
1	60	'2020/12/02'	117	145	479.0
2	60	'2020/12/03'	103	135	340.0

3	45	'2020/12/04'	109	175	282.4
4	45	'2020/12/05'	117	148	406.0
5	60	'2020/12/06'	102	127	300.0
6	60	'2020/12/07'	110	136	374.0
7	450	'2020/12/08'	104	134	253.3
8	30	'2020/12/09'	109	133	195.1
9	60	'2020/12/10'	98	124	269.0
10	60	'2020/12/11'	103	147	329.3
11	60	'2020/12/12'	100	120	250.7
12	60	'2020/12/12'	100	120	250.7
13	60	'2020/12/13'	106	128	345.3
14	60	'2020/12/14'	104	132	379.3
15	60	'2020/12/15'	98	123	275.0
16	60	'2020/12/16'	98	120	215.2
17	60	'2020/12/17'	100	120	300.0
18	45	'2020/12/18'	90	112	130.0
19	60	'2020/12/19'	103	123	323.0
20	45	'2020/12/20'	97	125	243.0
21	60	'2020/12/21'	108	131	364.2
22	45	NaN	100	119	282.0
23	60	'2020/12/23'	130	101	300.0
24	45	'2020/12/24'	105	132	246.0
25	60	'2020/12/25'	102	126	334.5
26	60	2020/12/26	100	120	250.0
27	60	'2020/12/27'	92	118	241.0
28	60	'2020/12/28'	103	132	130.0
29	60	'2020/12/29'	100	132	280.0
30	60	'2020/12/30'	102	129	380.3
31	60	'2020/12/31'	92	115	243.0

Pada contoh diatas, kode df["Calories"].fillna(130, inplace = True), artinya kita hanya mengeksekusi fungsi fillna hanya pada kolom Calories.

FUNGSI MEAN MEDIAN

Fungsi mean() adalah fungsi untuk menghitung nilai rata-rata dari dataframe. Perhatikan contoh berikut.

Contoh

```
import pandas as pd

df = pd.read_csv('data.csv') x =
 df["Calories"].mean() print(x)
```

Output

```
304.68
```

Pada contoh diatas, kode df[“Calories”].meand() artinya adalah kita menghitung nilai rata-rata untuk kolom Calories. Anda bisa juga lakukan ini untuk kolom yang lain.

Selain fungsi mean(), terdapat fungsi median(), untuk menghitung nilai tengah. Cara menggunakannya sama seperti mean.

Matplotlib

Matplotlib adalah package atau modul yang sangat familiar digunakan di python untuk visualisasi grafik. Saat belajar python, ini adalah termasuk modul yang wajib dipelajari.

Pada bab ini, kita tidak sediakan QR code dan link untuk online compiler karena matplotlib belum kompatibel dengan online compiler onecompiler yang kita gunakan. Kita berharap pembaca bisa mempraktekkannya langsung dari komputer atau laptopnya masing-masing.

INSTALASI

Untuk menginstall matplotlib kita dapat menggunakan PIP dengan sangat mudah. Kamu bisa buka kembali bab PIP jika lupa-lupa ingat apa itu PIP.

```
D:\MPP>pip install matplotlib
```

Setelah berhasil diinstall, maka untuk menggunakan matlplotlib kita tinggal import saja, dengan sintaks import matplotib.
Di dalam matplotlib terdapat sub modul yaitu pyplot. Kita akan menggunakan sub modul ini, karena kebanyakan fungsi yang

banyak digunakan ada di sub modul ini. Untuk menggunakannya, perhatikan contoh berikut:

Contoh

```
import matplotlib.pyplot as plt
```

Pada contoh diatas kita mengimport sub modul pyplot dari modul matplotlib lalu kita singkat namanya menjadi plt untuk digunakan di kodingan kita.

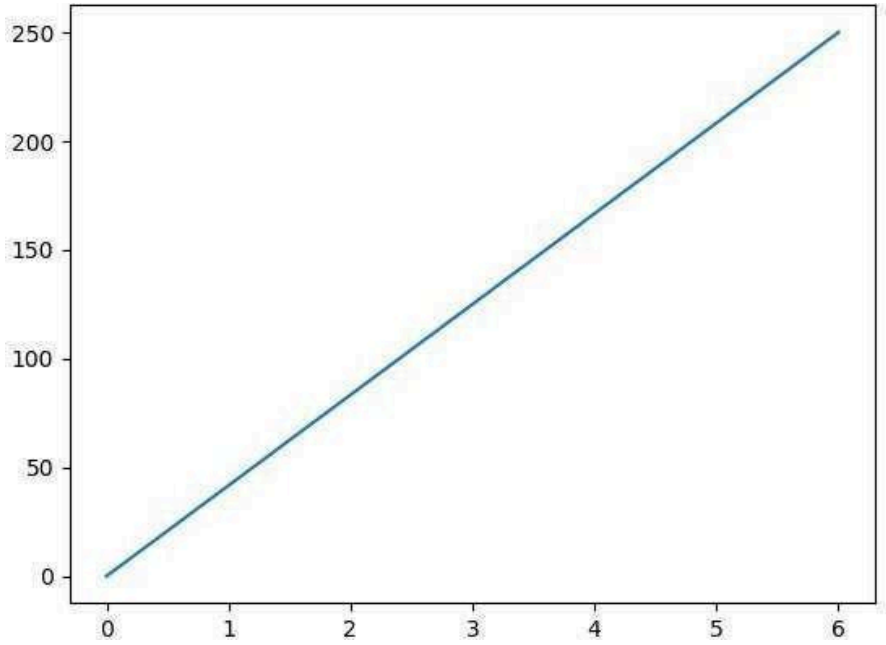
MENGEPLOT TITIK X DAN Y

Secara default, fungsi plot(parameter1, parameter2) menarik garis dari titik ke titik. Fungsi mengambil parameter untuk menentukan titik dalam diagram. Parameter 1 adalah array yang berisi titik-titik pada sumbu x . Parameter 2 adalah array yang berisi titik-titik pada sumbu y . Jika kita perlu memplot garis dari (0, 0) ke (6, 250), kita harus melewatkan dua array [0, 6] dan [0, 250] ke fungsi plot.

Contoh: https://s.id/mpp_ex261

```
import matplotlib.pyplot as plt import
 numpy as np

xpoints = np.array([0, 6]) ypoints =
 np.array([0, 250])

plt.plot(xpoints, ypoints)
 plt.show()
```

Output



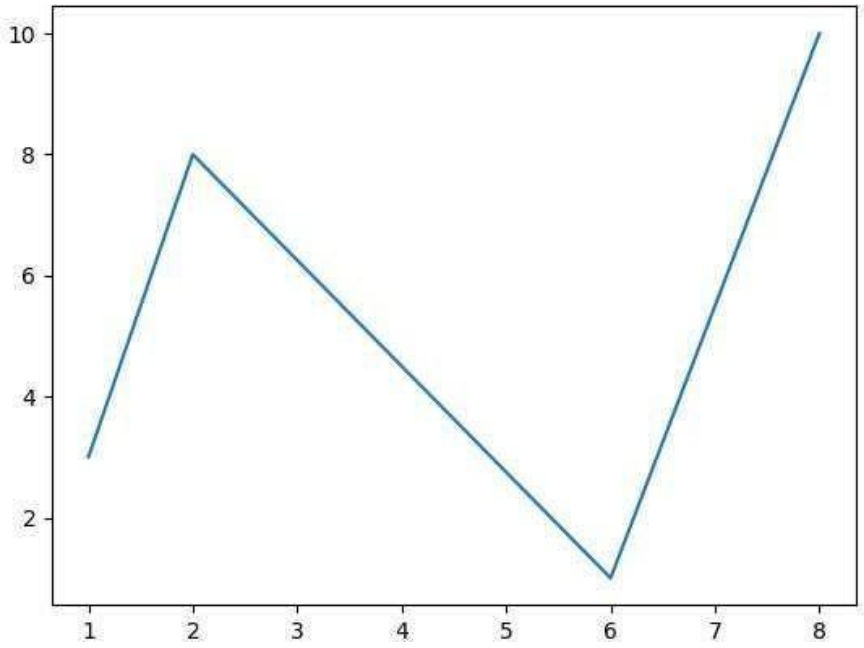
Pada contoh diatas, itu adalah contoh yang paling sederhana dari penggunaan pyplot matplotlob. Ada beberapa hal yang harus diperhatikan. Pertama kode *plt.plot(xpoints, ypoints)*, kode ini artinya adalah kita membuat plot dengan data sumbu-x-nya adalah data pada variabel xpoints dan data sumbu-y-nya adalah data pada variable ypoints. Sudah kita ketahui xpoints dan ypoints adalah variabel numpy array 1D. Namun dengan fungsi plt.plot(), hasil plot nya belum ditampilkan ke jendela. Untuk menampilkan hasisl plot ke layar, kita gunakan kode plt.show(). Contoh yang sederhana ini harap untuk diingat-ingat dan menjadi rujukan. Karena setelah ini akan banyak kita bahas lagi dan sebenarnya dasarnya adalah contoh ini.

Berikut contoh lain untuk titik yang lebih banyak.

Contoh

```
import matplotlib.pyplot as plt import
 numpy as np

xpoints = np.array([1, 2, 6, 8])
ypoints = np.array([3, 8, 1, 10])

plt.plot(xpoints, ypoints)
 plt.show()
```

Output

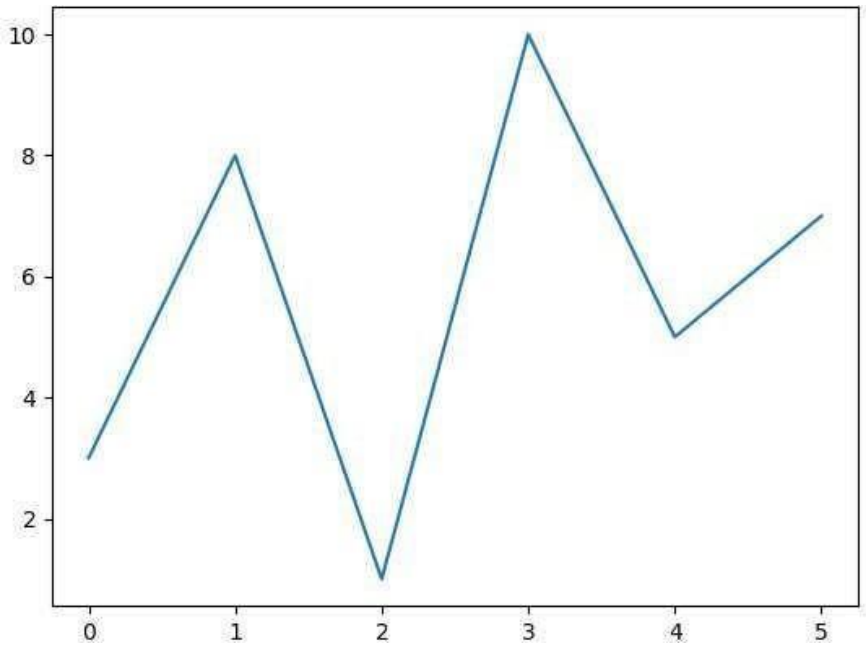


Sebenarnya, matplotlib sudah memiliki nilai sumbu bawaan yaitu dimulai dari 0,1,2, dst. Jadi jika kita menggunakan fungsi plt.plot() dengan hanya data sumbu y saja, itu tetap bisa dijalankan. Perhatikan contoh berikut.

Contoh

```
import matplotlib.pyplot as plt import
 numpy as np
ypoints = np.array([3, 8, 1, 10, 5, 7])


 plt.plot(ypoints)


plt.show()
```

Output



Perhatikan pada contoh diatas, terutama pada gambar sumbu x. Sumbu x dimulai dari angka 0 sampai 5 mengikuti jumlah datanya.

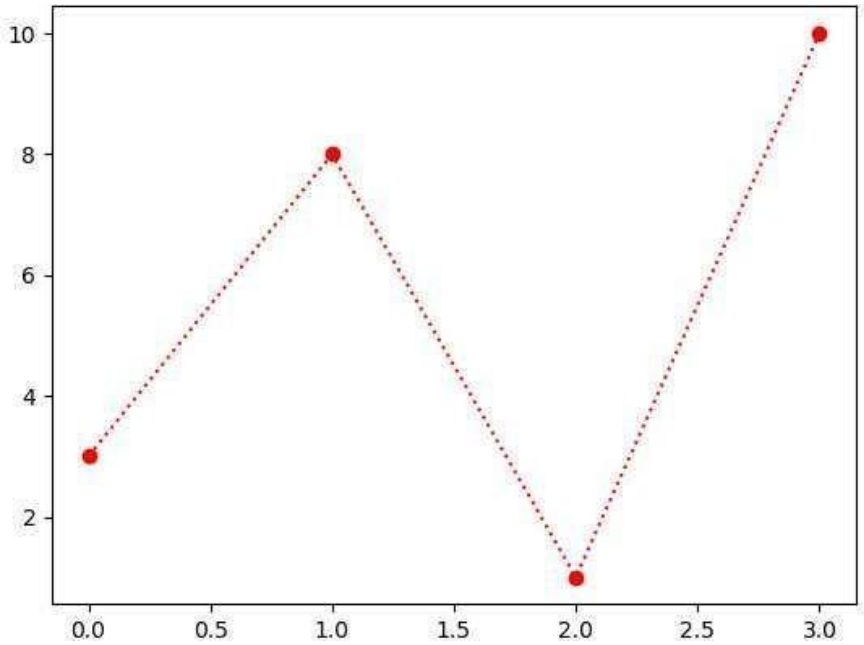
MARKER

Marker adalah penanda di setiap titik kordinat. Perhatikan contoh berikut untuk lebih jelasnya, terutama pada kode ang dicetak tebal.

Contoh

```
import matplotlib.pyplot as plt import
 numpy as np

ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, 'o:r')
 plt.show()
```

Output



Pada contoh diatas, pada kode yang dicetak tebal, ’o:r’. Kode tersebut terdapat 3 karakter. Karakter pertama mendefinisikan tipe marker, karakter kedua mendefinisikan tipe garis, dan

karakter ketiga adalah mendefinisikan tipe warna. Sehingga ‘o:r’ artinya adalah kita mendefinisikan tipe marker o yairu sebuah simbol lingkaran. Sedangkan tipe garis adalah ‘:’ yaitu titik-titik putus-putus. Sedangkan tipe warnanya adalah ‘r’ yakni warna merah. Bisa kita lihat pada gambar hasil outputnya. Untuk simbol-simbol ini bisa kita lihat pada tiga tabel dibawah ini.

Marker	Description
'o'	Circle
'*'	Star
'.'	Point
','	Pixel
'x'	X
'X'	X (filled)
'+'	Plus
'P'	Plus (filled)
's'	Square
'D'	Diamond
'd'	Diamond (thin)
'p'	Pentagon
'H'	Hexagon
'h'	Hexagon
'v'	Triangle Down
'^'	Triangle Up
'<'	Triangle Left
'>'	Triangle Right
'1'	Tri Down
'2'	Tri Up
'3'	Tri Left
'4'	Tri Right
'|'	Vline
'_'	Hline

Line Syntax	Description
'-'	Solid line
':'	Dotted line
'--'	Dashed line
'-.'	Dashed/dotted line

Color Syntax	Description
'r'	Red
'g'	Green
'b'	Blue
'c'	Cyan
'm'	Magenta
'y'	Yellow
'k'	Black
'w'	White

GARIS LEBIH DARI SATU

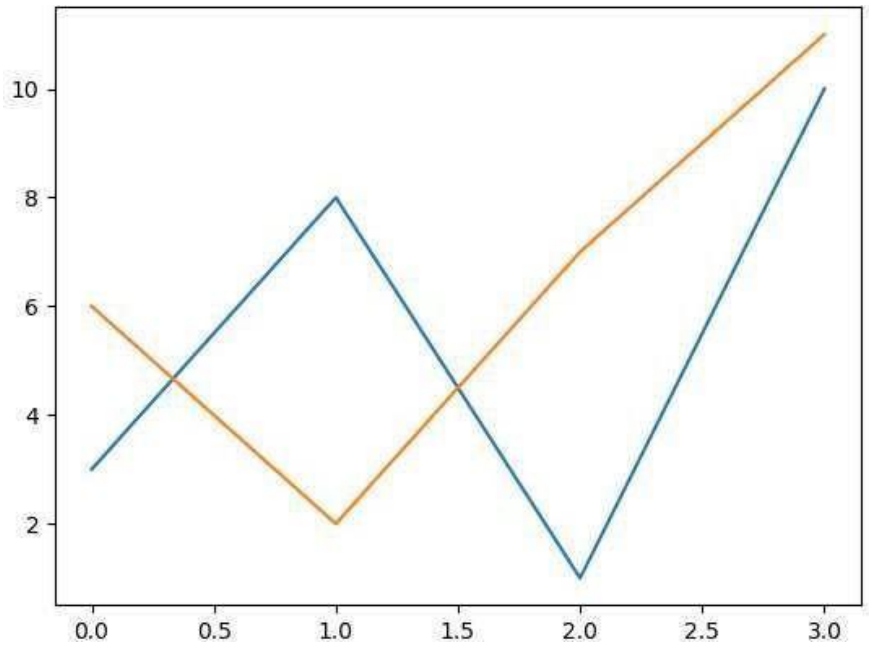
Kita dapat memplot baris sebanyak yang Anda suka hanya dengan menambahkan lebih banyak plt.plot(). Perhatikan contoh berikut.

Contoh

```
import matplotlib.pyplot as plt import
 numpy as np

y1 = np.array([3, 8, 1, 10])
y2 = np.array([6, 2, 7, 11])

plt.plot(y1) plt.plot(y2)
```

```
plt.show()
```

Output



Pada contoh diatas coba perhatikan, sebelum kode plt.show(), kita menuliskan dua kali plt.plot() yaitu yang pertama plt.plot(y1) lalu plt.plot(y2), ini menjadikan 2 grafik dalam satu kanvas.

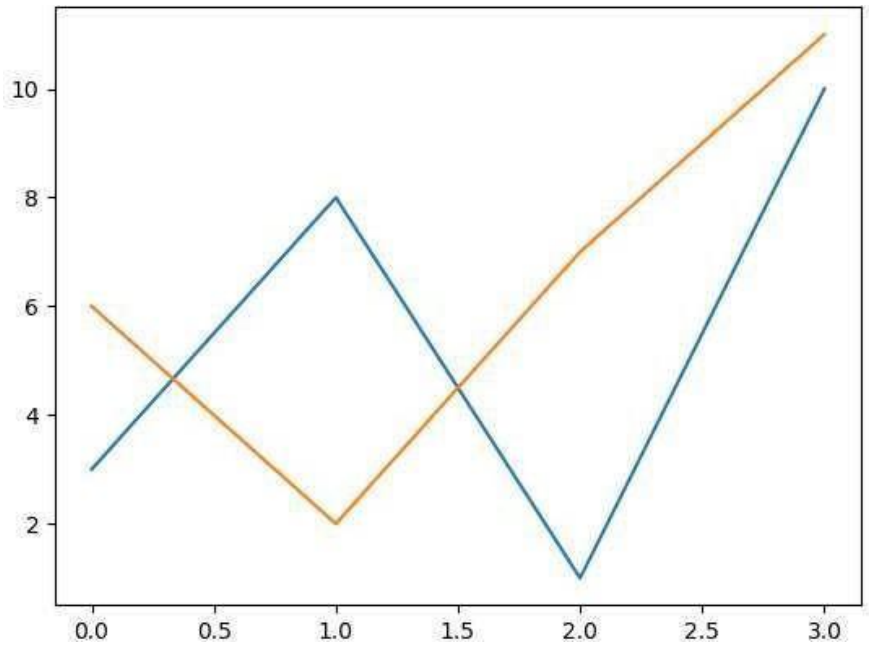
Jika contoh diatas adalah dua grafik dengan data sumbu x-nya dikosoongkan, lihat juga contoh berikut.

Contoh

```
import matplotlib.pyplot as plt import
 numpy as np


x1 = np.array([0, 1, 2, 3])
```

```
y1 = np.array([3,  8,  1,  10])
x2 = np.array([0,  1,  2,  3])
y2 = np.array([6,  2,  7,  11])

plt.plot(x1,  y1,  x2,  y2)

plt.show()
```

Output



Pada contoh diatas, perhatikan kode *plt.plot(x1,y1,x2,y2)*. Kode plt.plot() apabila memiliki parameter data array yang genap (berpasangan) maka akan dikenali sebagai pasangan data x dan y secara berurutan.

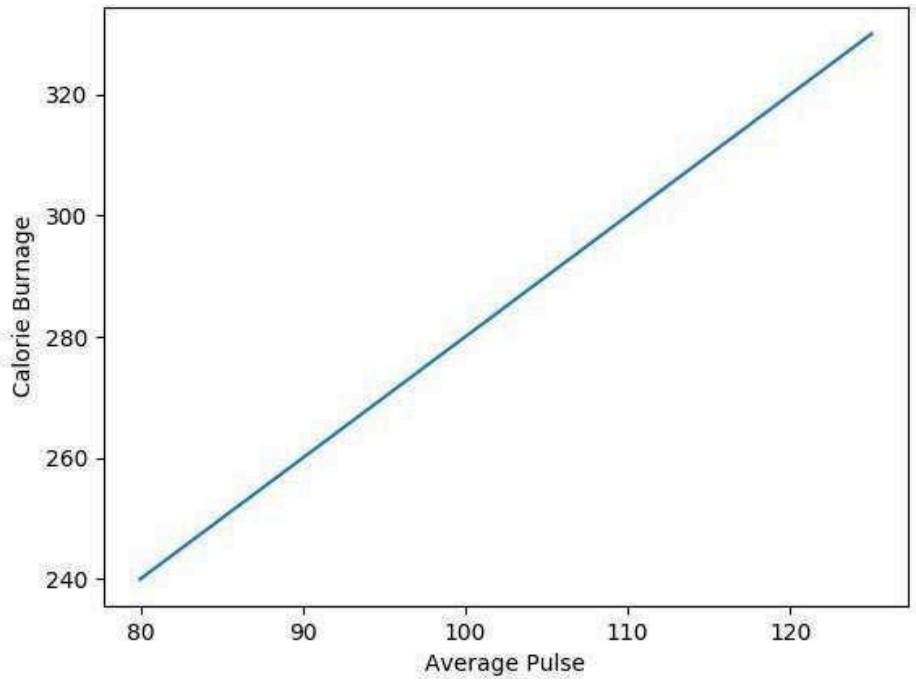
MENAMBAHKAN LABEL

Untuk menambahkan label pada sumbu x maupun sumbu y, kita dapat menggunakan fungsi xlabel() dan ylabel(). Perhatikan contoh berikut, terutama pada kodeyang dicetak tebal.

Contoh

```
import numpy as np
import matplotlib.pyplot as plt

x = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])
y = np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])

plt.plot(x, y)

plt.xlabel("Average Pulse")
plt.ylabel("Calorie Burnage")

plt.show()
```

Output



Perhatikan pada kode yang tercetak tebal yaitu plt.xlabel("Average Pulse") dan plt.ylabel("Calorie Burnage").

Kode ini mengisyaratkan matplotlib untuk mencetak label pada
sumbu x “Average Pulse” dan “Calorie Burnage” pada sumbu y.

Penempatan kode plt.xlabel() maupun plt.ylabel() harus sebelum plot.show(), namun tidak harus setelah plt.plot().

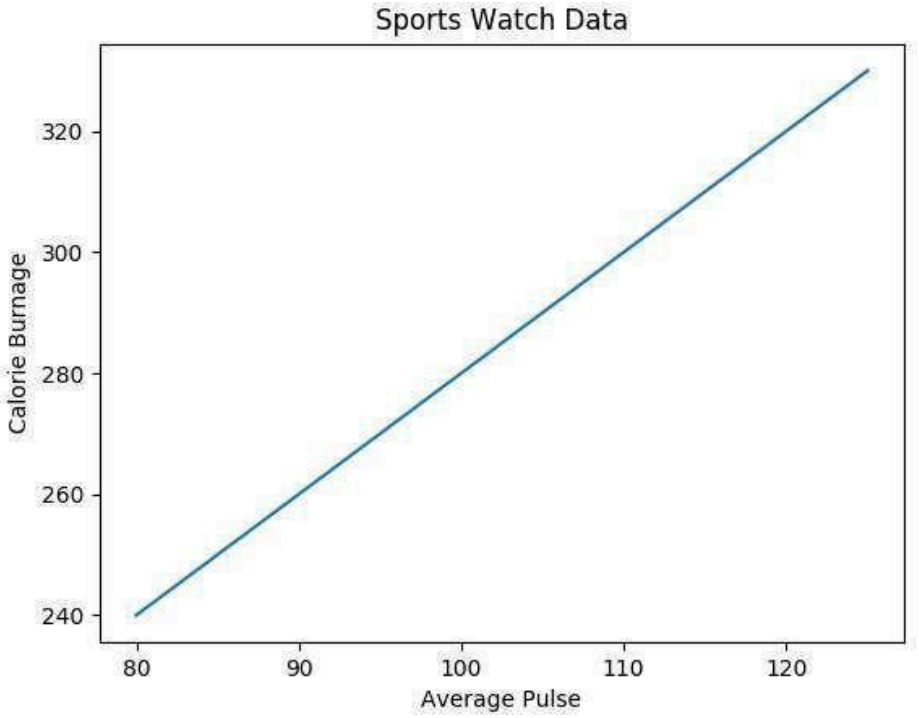
MENAMBAHKAN JUDUL

Untuk menambahkan judul pada plot kita dapat menggunakan fungsi title().
Perhatikan contoh berikut.

Contoh

```
import numpy as np
import matplotlib.pyplot as plt

x = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])
y = np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])

plt.plot(x, y)

plt.title("Sports Watch Data")
plt.xlabel("Average Pulse")
plt.ylabel("Calorie Burnage")

plt.show()
```

Output



GRID LINE

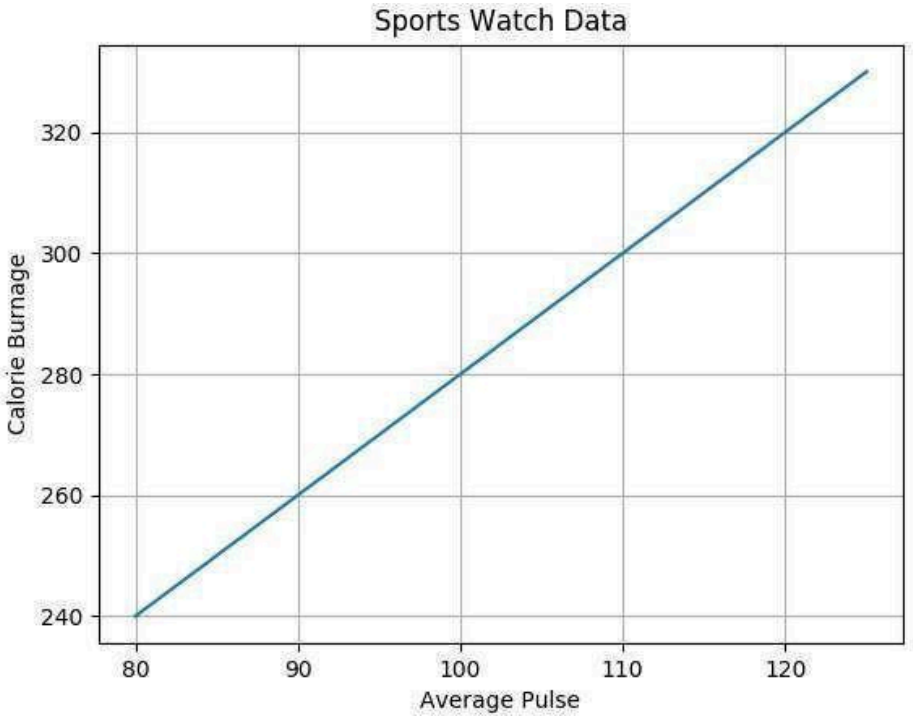
Jika kita lebih suka di kanvas plot kita terdapat garis grid line (kisi), maka kita bisa gunakan kode plt.grid() sebelum plot.show(). Perhatikan contoh berikut.

Contoh

```
import numpy as np
import matplotlib.pyplot as plt

x = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])
y = np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])

plt.title("Sports Watch Data")
plt.xlabel("Average Pulse")
plt.ylabel("Calorie Burnage")

plt.plot(x, y)

plt.grid()
```

```
plt.show()
```

Output



Jika kita hanya ingin menampilkan hanya garis grid pada sumbu x saja, maka kode plt.grid() kita ubah menjadi plt.grid(axis = 'x'). Sedangkan jika kita ingin menampilkan hanya sumbu y, kita harus ubah kode plt.grid() menjadi plt.grid(axis = 'y'). Anda bisa mencobanya langsung dan melihat perubahannya.

SUBPLOT

Dengan subplot, kita dapat menunjukkan beberapa kanvas dalam satu jendela. Fungsi ini memiliki sintaks dengan 3 parameter:

plt.subplot(jumlah_baris, jumlah_kolom, nomor_cell)

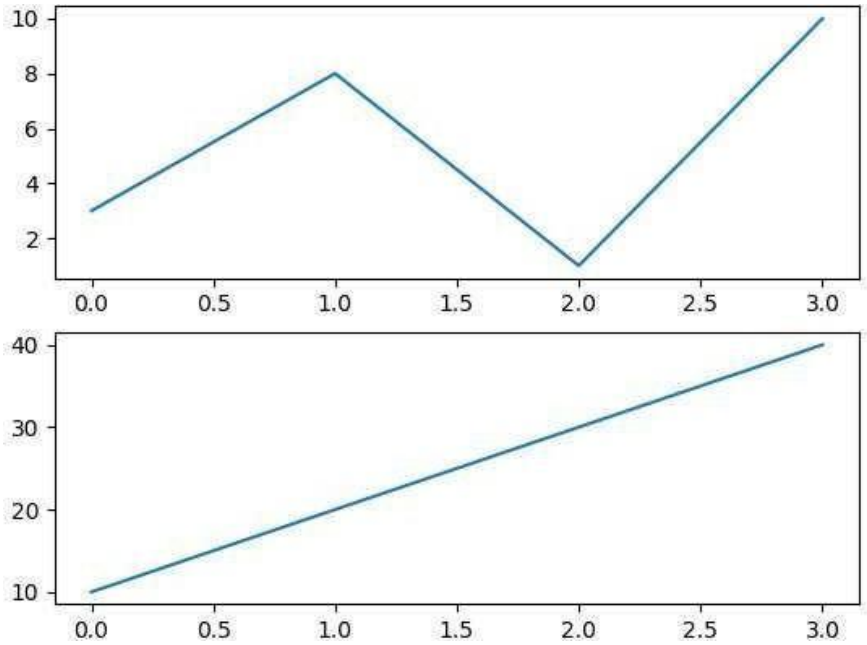
Pada parameter pertama adalah menentukan jumlah barisnya, parameter kedua adalah jumlah kolom, dan parameter ketiga adalah nomor cell yang saat ini aktif. Nomor cell aktif ini artinya adalah jika kita mengeksekusi kode plt.plot(), maka pada cell tersebut kanvas kita ditampilkan. Ingat, perhitungan nomor cell adalah dimulai dari baris 1 kolom 1, lalu beris 1 kolom 2, baris
1 kolom 3, …. , baris 1 kolom terakhir, lanjut baris 2 kolom 1,
baris 2 kolom 2, dan seterusnya. Perhatikan contoh berikut.

Contoh

```
import matplotlib.pyplot as
plt
import numpy as np
#plot 1:
x = np.array([0, 1, 2,                    3])
y = np.array([3, 8, 1,                    10])

plt.subplot(2, 1, 1)
plt.plot(x,y)

#plot 2:
x = np.array([0, 1, 2,                    3])
y = np.array([10, 20, 30,
40])
plt.subplot(2, 1,
2)
plt.plot(x,y)
```

```
plt.show()
```

Output



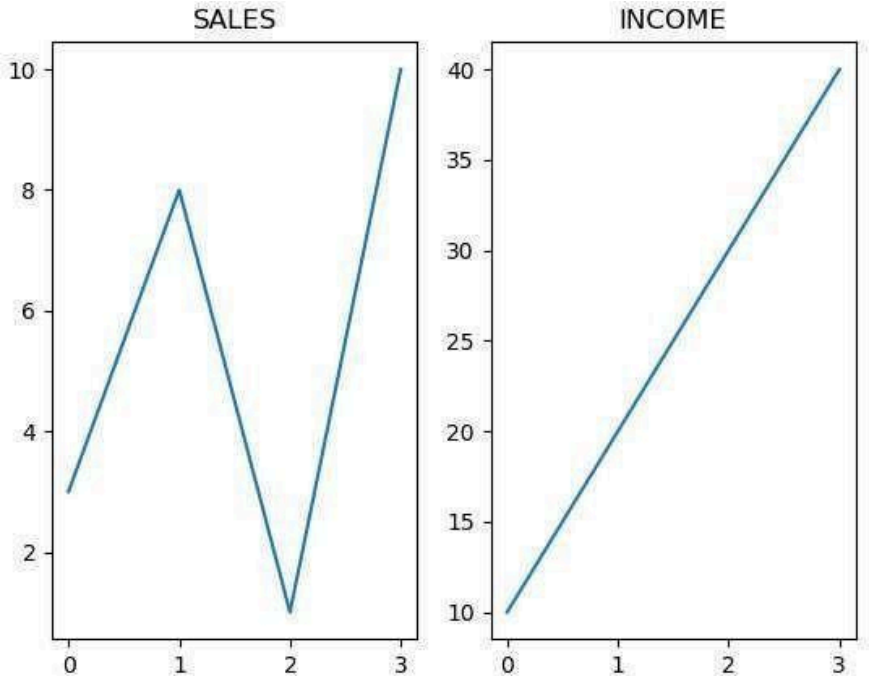
Pada contoh diatas, perhatikan pada kode tercetak tebal pertama plt.subplot(2,1,1). Kode ini artinya adalah kita mendefinisikan kita membagi kanvas menjadi beberapa bagian yaitu 2 baris dan 1 kolom. Lalu kita menetapkan cell yang aktif sekarag adalah cell nomor 1. Sehingga saat kode plt.plot(x,y) dibawahnya tepat, grafik tersebut akan ditampilkan di cell 1 yaitu baris 1 kolom 1. Lalu perhatikan pada kode tercetak tebal di bagian bawah yaitu plt.subplot(2,1,2). Kode ini mengisyaratkan cell aktifnya pindah ke cell nomor 2. Yang perlu kita perhatikan adalah nilai parameter jumlah baris dan kolom harus sama dengan yang kode plt.subplot yang lain.

Setiap kanvas pada setiap cell di dalam subplot, kita dapat mengatur title-nya masing-masing dengan fungsi title(). Perhatikan contoh berikut.

Contoh

```
import matplotlib.pyplot as plt
import numpy as np

#plot 1:
x = np.array([0, 1, 2,                    3])
y = np.array([3, 8, 1,                    10])

plt.subplot(1, 2, 1) plt.plot(x,y)

plt.title("SALES")

#plot 2:
x = np.array([0, 1, 2,                    3])
y = np.array([10, 20, 30, 40])
plt.subplot(1, 2, 2)
plt.plot(x,y)
plt.title("INCOME"
plt.show
)
()
```

Output



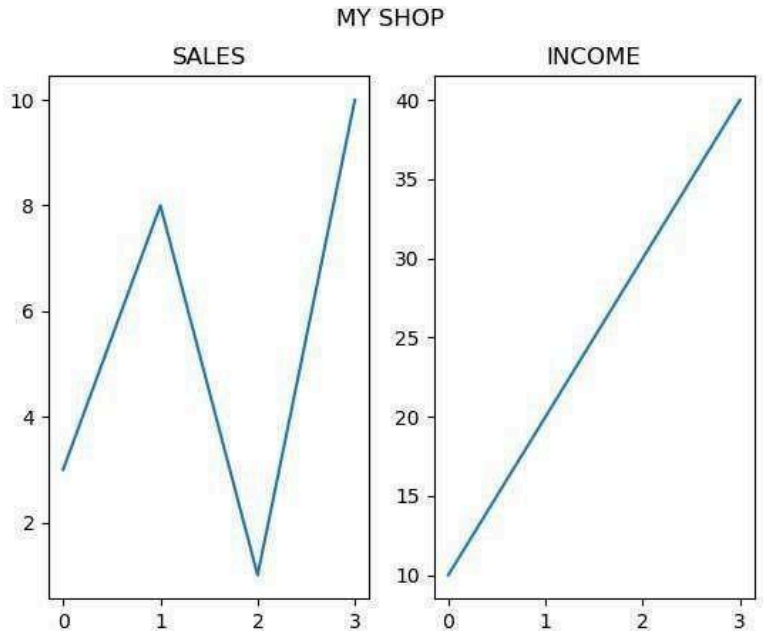
Kita juga bisa membuat judul pada kanvas utama dengan fungsi suptitle(). Perhatikan contoh berikut.

Contoh

```
import matplotlib.pyplot as plt import
 numpy as np

#plot 1:
x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])

plt.subplot(1, 2, 1)
 plt.plot(x,y)
 plt.title("SALES")
```

```
#plot 2:

x = np.array([0, 1, 2, 3])
y = np.array([10, 20, 30, 40])


plt.subplot(1, 2, 2)
 plt.plot(x,y)
 plt.title("INCOME")


plt.suptitle("MY SHOP")
plt.show()
```

Output



SCATTER PLOT

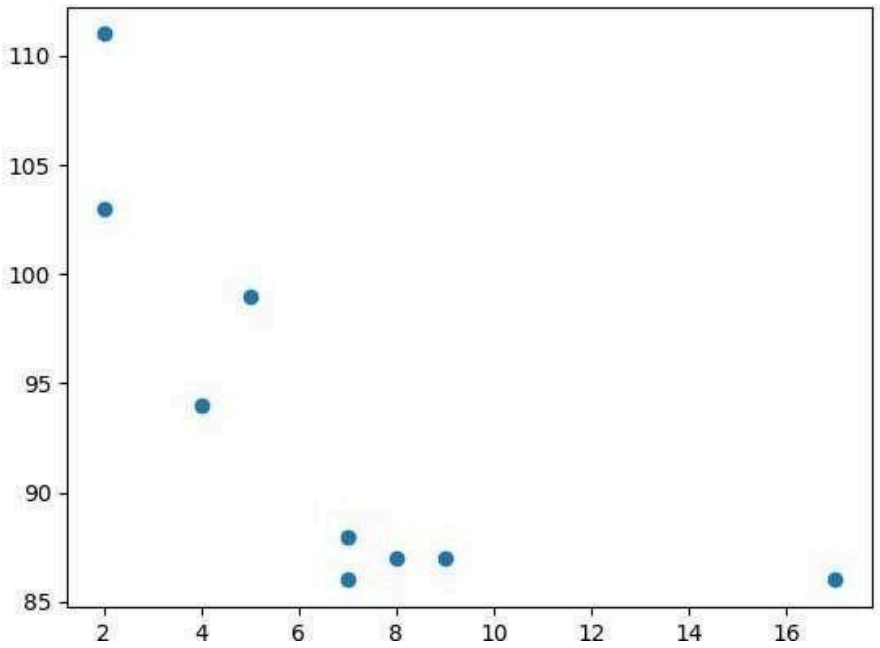
Sebenarnya, matplotlib menyediakan banyak cara mengeplot grafik. Fungsi plt.plot() adalah salah satunya. Salah satu lainnya

adalah plt.scatter(). Fungsi ini biasanya digunakan untuk melihat penyebaran data. Perhatikan contoh berikut.

Contoh

```
import matplotlib.pyplot as plt import numpy as np

x = np.array([5,7,8,7,2,17,2,9,4])
y =
np.array([99,86,87,88,111,86,103,87,94])

plt.scatter(x, y)
plt.show()
```

Output

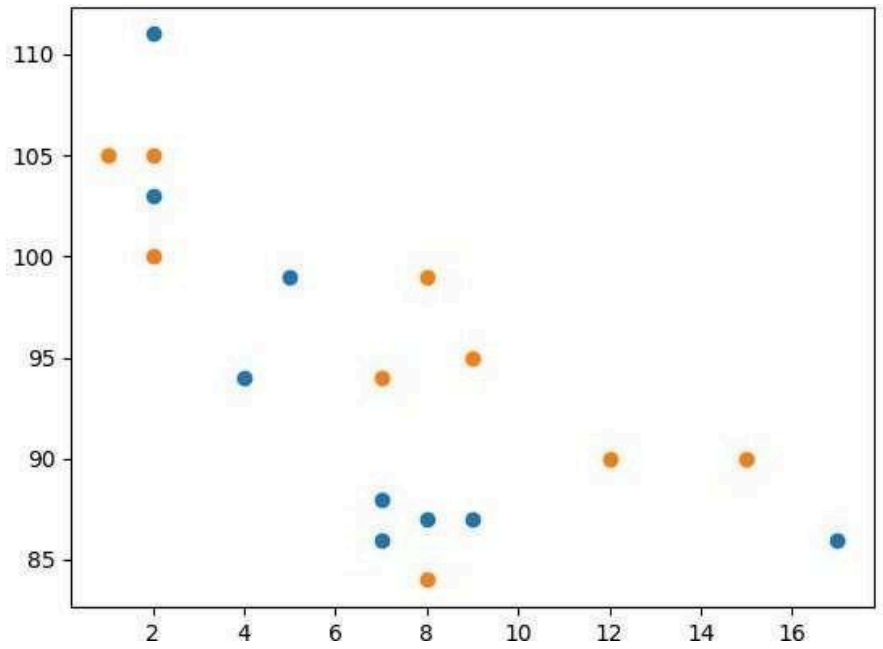


Kita juga bisa menampilkan 2 grafik dalam satu kanvas dengan scatter ini.

Contoh

```
import matplotlib.pyplot as plt import numpy as np

#day one, the age and speed of 13 cars: x = np.array([5,7,8,7,2,17,2,9,4])
y                                                                    =
np.array([99,86,87,88,111,86,103,87,94])
plt.scatter(x, y)

#day two, the age and speed of 15 cars: x =
np.array([2,2,8,1,15,8,12,9,7])
y                                                                    =
np.array([100,105,84,105,90,99,90,95,94])
plt.scatter(x, y)

plt.show()
```

Output



Kita juga bisa mendefinisikan warna untuk setiap scatter plot kita dengan sintaks plt.scatter(x,y,color="Blues"). Kode Blues adalah kode warna yang sudah ada di python. Selain dengan kode warna kita juga bisa menggunakan kode warna hexadecimal misal plt.scatter(x,y,color="#88c999"). Kode warna yang ada di python cukup banyak, kita bisa cek di https://matplotlib.org/2.0.1/examples/color/named_colors.html atau lihat pada gambar dibawah ini.


black
gray
silver
whitesmoke
rosybrown
firebrick
red
darksalmon
sienna
sandybrown
bisque
tan
moccasin
floralwhite
gold
darkkhaki
lightgoldenrodyellow
olivedrab
chartreuse
palegreen
darkgreen
seagreen
mediumspringgreen
lightseagreen
paleturquoise
darkcyan
darkturquoise
deepskyblue
aliceblue
slategray
royalblue
navy
blue
mediumpurple
darkorchid
plum
m
mediumvioletred
palevioletred
k
grey
lightgray
w
lightcoral
maroon
mistyrose
coral
seashell
peachpuff
darkorange
navajowhite
orange
darkgoldenrod
lemonchiffon
ivory
olive
yellowgreen
lawngreen
lightgreen
g
mediumseagreen
mediumaquamarine
mediumturquoise
darkslategray
c
cadetblue
skyblue
dodgerblue
slategrey
ghostwhite
darkblue
slateblue
rebeccapurple
darkviolet
violet
fuchsia
deeppink
crimson

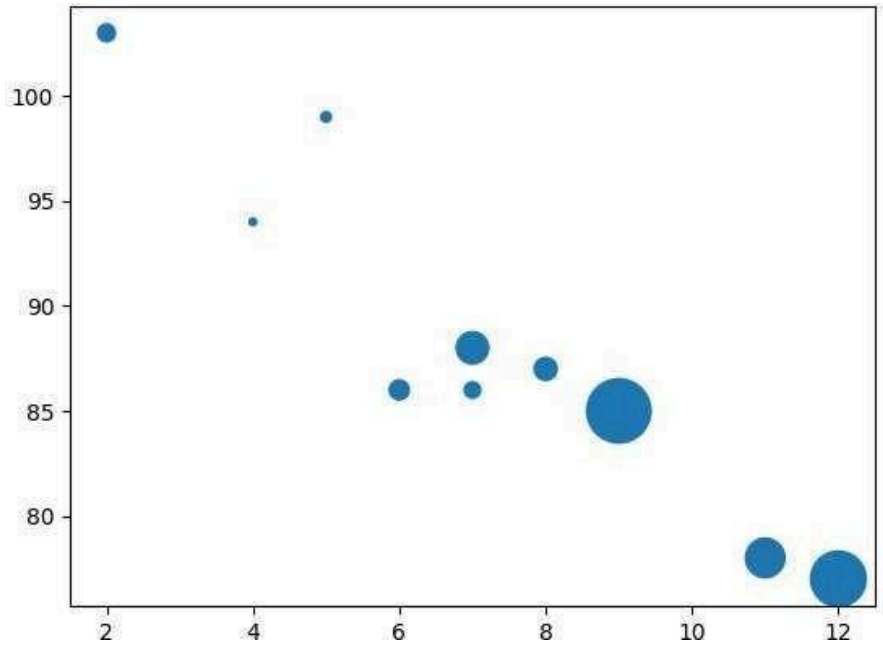
Dengan fungsi plt.scatter() kita juga dapat menentukan besar dari setiap titik yang akan ditampilkan di canvas. Perhatikan contoh berikut.

Contoh

```
import matplotlib.pyplot as plt
import numpy as np

x = np.array([5,7,8,7,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,103,86,94,78,77,85,86])
sizes = np.array([20,50,100,200,60,90,10,300,600,800,75])

plt.scatter(x, y, s=sizes)
```

```
plt.show()
```

Output



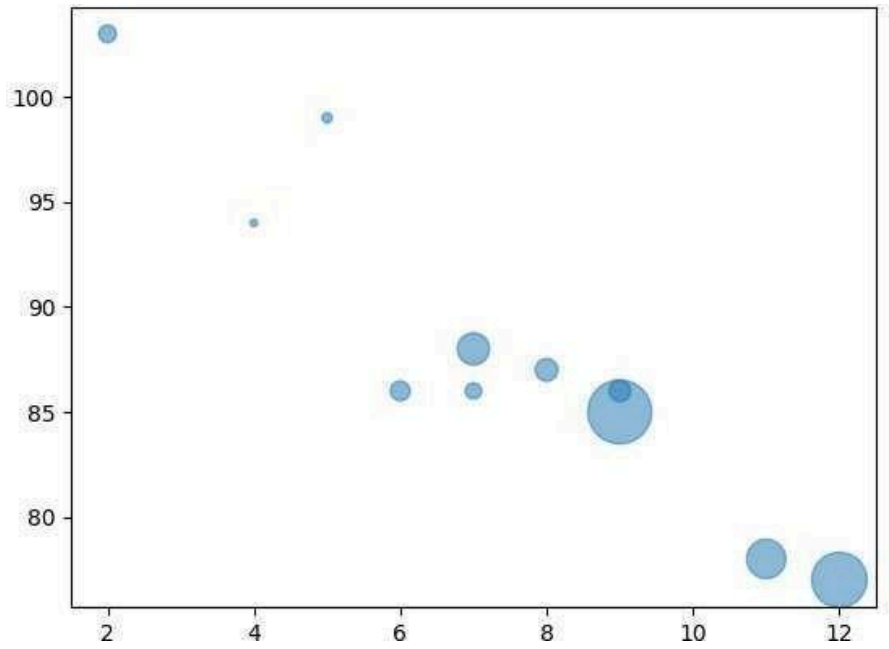
Saat kita bermain dengan size yang besar untuk data di scatter, terkadang antar data tumpang tindih. Kita bisa menggunakan transparansi dengan menambahkan parameter alpha pada sintaks plt.scatter(). Perhatikan contoh berikut.

Contoh

```
import matplotlib.pyplot as plt import
numpy as np

x = np.array([5,7,8,7,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,103,86,94,78,77,85,86])
sizes = np.array([20,50,100,200,60,90,10,300,600,800,75])

plt.scatter(x, y, s=sizes, alpha=0.5)
```

plt.show()

Output



FUNGSI LAIN PYPLOT

Pyplot memiliki fungsi yang cukup banyak. Sampai disini, kita yakin, jika anda mengikuti dari awal, anda sudah punya cukup skill untuk bisa membaca dokumentasi langsung dari pyplot dan mempraktekkannya sendiri. Untuk melihat fungsi apa saja yang disediakan oleh pyplot, anda bisa cek disini. Dibawah ini kita tampilkan dalam tabel berikut fungsi-fungsi yang disediakan oleh pyplot untuk menunjang plot grafik.

Fungsi dan sintaksnya	Deskripsi
acorr(x, *[, data])	Plot the autocorrelation of *x*.

angle_spectrum(x[, Fs, Fc, window, pad_to, ...])	Plot the angle spectrum.
annotate(text, xy, *args, **kwargs)	Annotate the point *xy* with text *text*.
arrow(x, y, dx, dy, **kwargs)	Add an arrow to the Axes.
autoscale([enable, axis, tight])	Autoscale the axis view to the data (toggle).
autumn()	Set the colormap to 'autumn'.
axes([arg])	Add an axes to the current figure and make it the current axes.
axhline([y, xmin, xmax])	Add a horizontal line across the axis.
axhspan(ymin, ymax[, xmin, xmax])	Add a horizontal span (rectangle) across the Axes.
axis(*args[, emit])	Convenience method to get or set some axis properties.
axline(xy1[, xy2, slope])	Add an infinitely long straight line.
axvline([x, ymin, ymax])	Add a vertical line across the Axes.
axvspan(xmin, xmax[, ymin, ymax])	Add a vertical span (rectangle) across the Axes.
bar(x, height[, width, bottom, align, data])	Make a bar plot.
bar_label(container[, labels, fmt, ...])	Label a bar plot.
barbs(*args[, data])	Plot a 2D field of barbs.
barh(y, width[, height, left, align])	Make a horizontal bar plot.
bone()	Set the colormap to 'bone'.
box([on])	Turn the axes box on or off on the current axes.
boxplot(x[, notch, sym, vert, whis, ...])	Draw a box and whisker plot.

broken_barh(xranges, yrange, *[, data])	Plot a horizontal sequence of rectangles.
cla()	Clear the current axes.
clabel(CS[, levels])	Label a contour plot.
clf()	Clear the current figure.
clim([vmin, vmax])	Set the color limits of the current image.
close([fig])	Close a figure window.
cohere(x, y[, NFFT, Fs, Fc, detrend, ...])	Plot the coherence between *x* and *y*.
colorbar([mappable, cax, ax])	Add a colorbar to a plot.
connect(s, func)	Bind function *func* to event *s*.
contour(*args[, data])	Plot contour lines.
contourf(*args[, data])	Plot filled contours.
cool()	Set the colormap to 'cool'.
copper()	Set the colormap to 'copper'.
csd(x, y[, NFFT, Fs, Fc, detrend, window, ...])	Plot the cross-spectral density.
delaxes([ax])	Remove an Axes (defaulting to the current axes) from its figure.
disconnect(cid)	Disconnect the callback with id *cid*.
draw()	Redraw the current figure.
draw_if_interactive()	Redraw the current figure if in interactive mode.
errorbar(x, y[, yerr, xerr, fmt, ecolor, ...])	Plot y versus x as lines and/or markers with attached errorbars.
eventplot(positions[, orientation, ...])	Plot identical parallel lines at the given positions.
figimage(X[, xo, yo, alpha, norm, cmap, ...])	Add a non-resampled image to the figure.

figlegend(*args, **kwargs)	Place a legend on the figure.
fignum_exists(num)	Return whether the figure with the given id exists.
figtext(x, y, s[, fontdict])	Add text to figure.
figure([num, figsize, dpi, facecolor, ...])	Create a new figure, or activate an existing figure.
fill(*args[, data])	Plot filled polygons.
fill_between(x, y1[, y2, where, ...])	Fill the area between two horizontal curves.
fill_betweenx(y, x1[, x2, where, step, ...])	Fill the area between two vertical curves.
findobj([o, match, include_self])	Find artist objects.
flag()	Set the colormap to 'flag'.
gca(**kwargs)	Get the current Axes.
gcf()	Get the current figure.
gci()	Get the current colorable artist.
get(obj, *args, **kwargs)	Return the value of an Artist's *property*, or print all of them.
get_current_fig_manager()	Return the figure manager of the current figure.
get_figlabels()	Return a list of existing figure labels.
get_fignums()	Return a list of existing figure numbers.
get_plot_commands()	Get a sorted list of all of the plotting commands.
getp(obj, *args, **kwargs)	Return the value of an Artist's *property*, or print all of them.
ginput([n, timeout, show_clicks, mouse_add, ...])	Blocking call to interact with a figure.
gray()	Set the colormap to 'gray'.

grid([visible, which, axis])	Configure the grid lines.
hexbin(x, y[, C, gridsize, bins, xscale, ...])	Make a 2D hexagonal binning plot of points *x*, *y*.
hist(x[, bins, range, density, weights, ...])	Plot a histogram.
hist2d(x, y[, bins, range, density, ...])	Make a 2D histogram plot.
hlines(y, xmin, xmax[, colors, linestyles, ...])	Plot horizontal lines at each *y* from *xmin* to *xma x*.
hot()	Set the colormap to 'hot'.
hsv()	Set the colormap to 'hsv'.
imread(fname[, format])	Read an image from a file into an array.
imsave(fname, arr, **kwargs)	Save an array as an image file.
imshow(X[, cmap, norm, aspect, ...])	Display data as an image, i.e., on a 2D regular raster.
inferno()	Set the colormap to 'inferno'.
install_repl_displayhook()	Install a repl display hook so that any stale figure are automatically redrawn when control is returned to the repl.
ioff()	Disable interactive mode.
ion()	Enable interactive mode.
isinteractive()	Return whether plots are updated after every plotting command.
jet()	Set the colormap to 'jet'.
legend(*args, **kwargs)	Place a legend on the Axes.
locator_params([axis, tight])	Control behavior of major tick locators.
loglog(*args, **kwargs)	Make a plot with log scaling on both the x and y axis.

magma()	Set the colormap to 'magma'.
magnitude_spectrum(x[, Fs, Fc, window, ...])	Plot the magnitude spectrum.
margins(*margins[, x, y, tight])	Set or retrieve autoscaling margins.
matshow(A[, fignum])	Display an array as a matrix in a new figure window.
minorticks_off()	Remove minor ticks from the Axes.
minorticks_on()	Display minor ticks on the Axes.
new_figure_manager(num, *args, **kwargs)	Create a new figure manager instance.
nipy_spectral()	Set the colormap to 'nipy_spectral'.
pause(interval)	Run the GUI event loop for *interval* seconds.
pcolor(*args[, shading, alpha, norm, cmap, ...])	Create a pseudocolor plot with a non-regular rectangular grid.
pcolormesh(*args[, alpha, norm, cmap, vmin, ...])	Create a pseudocolor plot with a non-regular rectangular grid.
phase_spectrum(x[, Fs, Fc, window, pad_to, ...])	Plot the phase spectrum.
pie(x[, explode, labels, colors, autopct, ...])	Plot a pie chart.
pink()	Set the colormap to 'pink'.
plasma()	Set the colormap to 'plasma'.
plot(*args[, scalex, scaley, data])	Plot y versus x as lines and/or markers.
plot_date(x, y[, fmt, tz, xdate, ydate, data])	Plot coercing the axis to treat floats as dates.
polar(*args, **kwargs)	Make a polar plot.
prism()	Set the colormap to 'prism'.

psd(x[, NFFT, Fs, Fc, detrend, window, ...])	Plot the power spectral density.
quiver(*args[, data])	Plot a 2D field of arrows.
quiverkey(Q, X, Y, U, label, **kwargs)	Add a key to a quiver plot.
rc(group, **kwargs)	Set the current rcParams. *grou p* is the grouping for the rc, e.g., for lines.linewidth the group is lines, for axes.facecolor, the group is axes, and so on. Group may also be a list or tuple of group names, e.g., (*xtick*, *ytick*). *kwargs* is a dictionary attribute name/value pairs, e.g.,::.
rc_context([rc, fname])	Return a context manager for temporarily changing rcParams.
rcdefaults()	Restore the rcParams from Matplotlib's internal default style.
rgrids([radii, labels, angle, fmt])	Get or set the radial gridlines on the current polar plot.
savefig(*args, **kwargs)	Save the current figure.
sca(ax)	Set the current Axes to *ax* and the current Figure to the parent of *ax*.
scatter(x, y[, s, c, marker, cmap, norm, ...])	A scatter plot of *y* vs.
sci(im)	Set the current image.
semilogx(*args, **kwargs)	Make a plot with log scaling on the x axis.
semilogy(*args, **kwargs)	Make a plot with log scaling on the y axis.

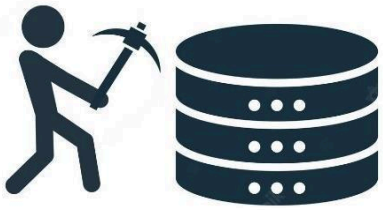
set_cmap(cmap)	Set the default colormap, and applies it to the current image if any.
set_loglevel(*args, **kwargs)	Set Matplotlib's root logger and root logger handler level, creating the handler if it does not exist yet.
setp(obj, *args, **kwargs)	Set one or more properties on an Artist, or list allowed values.
show(*[, block])	Display all open figures.
specgram(x[, NFFT, Fs, Fc, detrend, window , ...])	Plot a spectrogram.
spring()	Set the colormap to 'spring'.
spy(Z[, precision, marker, markersize, ...])	Plot the sparsity pattern of a 2D array.
stackplot(x, *args[, labels, colors, ...])	Draw a stacked area plot.
stairs(values[, edges, orientation, ...])	A stepwise constant function as a line with bounding edges or a filled plot.
stem(*args[, linefmt, markerfmt, basefmt, ...])	Create a stem plot.
step(x, y, *args[, where, data])	Make a step plot.
streamplot(x, y, u, v[, density, linewidth, ...])	Draw streamlines of a vector flow.
subplot(*args, **kwargs)	Add an Axes to the current figure or retrieve an existing Axes.
subplot2grid(shape, loc[, rowspan, colspan, f ig])	Create a subplot at a specific location inside a regular grid.
subplot_mosaic(mosaic, *[, sharex, sharey, .. .])	Build a layout of Axes based on ASCII art or nested lists.
subplot_tool([targetfig])	Launch a subplot tool window for a figure.

subplots([nrows, ncols, sharex, sharey, ...])	Create a figure and a set of subplots.
subplots_adjust([left, bottom, right, top, ...])	Adjust the subplot layout parameters.
summer()	Set the colormap to 'summer'.
suptitle(t, **kwargs)	Add a centered suptitle to the figure.
switch_backend(newbackend)	Close all open figures and set the Matplotlib backend.
table([cellText, cellColours, cellLoc, ...])	Add a table to an Axes.
text(x, y, s[, fontdict])	Add text to the Axes.
thetagrids([angles, labels, fmt])	Get or set the theta gridlines on the current polar plot.
tick_params([axis])	Change the appearance of ticks, tick labels, and gridlines.
ticklabel_format(*[, axis, style, ...])	Configure the ScalarFormatter us ed by default for linear axes.
tight_layout(*[, pad, h_pad, w_pad, rect])	Adjust the padding between and around subplots.
title(label[, fontdict, loc, pad, y])	Set a title for the Axes.
tricontour(*args, **kwargs)	Draw contour lines on an unstructured triangular grid.
tricontourf(*args, **kwargs)	Draw contour regions on an unstructured triangular grid.
tripcolor(*args[, alpha, norm, cmap, vmin, ...])	Create a pseudocolor plot of an unstructured triangular grid.
triplot(*args, **kwargs)	Draw a unstructured triangular grid as lines and/or markers.

twinx([ax])	Make and return a second axes that shares the *x*-axis.
twiny([ax])	Make and return a second axes that shares the *y*-axis.
uninstall_repl_displayhook()	Uninstall the Matplotlib display hook.
violinplot(dataset[, positions, vert, ...])	Make a violin plot.
viridis()	Set the colormap to 'viridis'.
vlines(x, ymin, ymax[, colors, linestyles, ...])	Plot vertical lines at each *x* from *ymin* to *yma* *x*.
waitforbuttonpress([timeout])	Blocking call to interact with the figure.
winter()	Set the colormap to 'winter'.
xcorr(x, y[, normed, detrend, usevlines, ...])	Plot the cross correlation between *x* and *y*.
xkcd([scale, length, randomness])	Turn on xkcd sketch-style drawing mode.
xlabel(xlabel[, fontdict, labelpad, loc])	Set the label for the x-axis.
xlim(*args, **kwargs)	Get or set the x limits of the current axes.
xscale(value, **kwargs)	Set the x-axis scale.
xticks([ticks, labels])	Get or set the current tick locations and labels of the x-axis.
ylabel(ylabel[, fontdict, labelpad, loc])	Set the label for the y-axis.
ylim(*args, **kwargs)	Get or set the y-limits of the current axes.
yscale(value, **kwargs)	Set the y-axis scale.
yticks([ticks, labels])	Get or set the current tick locations and labels of the y-axis.

MODUL 4
(MINGGU KEEMPAT)

Praktikum 7



DATA MINING

Data mining adalah proses penggalian atau ekstraksi informasi yang berguna dari data mentah yang besar dan kompleks. Tujuan utama dari data mining adalah untuk menemukan pola tersembunyi dan wawasan yang dapat membantu pengambilan keputusan yang lebih baik.

Proses data mining melibatkan beberapa tahap, termasuk pengumpulan data, pemrosesan data, transformasi data, pemodelan, dan evaluasi. Pada tahap pengumpulan data, data yang relevan dikumpulkan dari berbagai sumber. Kemudian, data diolah dan dibersihkan untuk menghilangkan kesalahan dan outlier. Selanjutnya, data diubah menjadi format yang dapat diproses oleh algoritma data mining.

Selanjutnya, model data mining dibangun dengan menggunakan teknik seperti clustering, klasifikasi, regresi, asosiasi, atau segmentasi. Setelah model dibangun, model dievaluasi dan diuji dengan menggunakan data yang tidak digunakan selama proses pembangunan model. Hasil evaluasi ini digunakan untuk mengukur keefektifan model dalam memprediksi atau mengidentifikasi pola.Data mining digunakan di banyak bidang, seperti bisnis, keuangan, kesehatan, dan ilmu pengetahuan. Contohnya, di bidang bisnis, data mining dapat membantu dalam identifikasi tren pasar, segmentasi pelanggan, dan analisis risiko keuangan. Sedangkan di bidang kesehatan, data mining dapat membantu dalam analisis data medis untuk mendukung diagnosis dan perawatan pasien.

Dalam ringkasannya, Data mining adalah teknik penggalian informasi yang menggunakan algoritma komputer untuk menemukan pola dan wawasan dari data yang besar dan kompleks. Hal ini dapat membantu dalam mengidentifikasi tren, pola, dan hubungan dalam data yang dapat digunakan untuk pengambilan keputusan yang lebih baik.

Berikut adalah beberapa metode data mining yang umum digunakan:

1. Klasifikasi: metode ini digunakan untuk memprediksi kelas atau label dari suatu objek berdasarkan kumpulan data yang sudah diketahui. Contoh penggunaan klasifikasi adalah untuk memprediksi apakah seorang pelanggan akan membeli produk tertentu atau tidak.

2. Regresi: metode ini digunakan untuk memprediksi nilai kontinu dari suatu variabel berdasarkan variabel lainnya. Contohnya adalah untuk memprediksi harga rumah berdasarkan ukuran, lokasi, dan fitur lainnya.
3. Asosiasi: metode ini digunakan untuk menemukan hubungan antara item dalam dataset. Contoh penggunaan asosiasi adalah untuk menemukan produk yang sering dibeli bersamaan oleh pelanggan.
4. Segmentasi: metode ini digunakan untuk membagi dataset menjadi kelompok yang lebih kecil berdasarkan karakteristik atau atribut yang sama. Contoh penggunaan segmentasi adalah untuk membagi pelanggan ke dalam kelompok berdasarkan preferensi dan perilaku mereka.
5. Klastering: metode ini digunakan untuk membagi dataset menjadi kelompok yang lebih kecil berdasarkan kesamaan karakteristik atau atribut. Contohnya adalah untuk membagi produk ke dalam kelompok berdasarkan fitur atau kriteria tertentu.
6. Pengelompokan: metode ini digunakan untuk mengelompokkan data menjadi kelas atau kelompok tertentu berdasarkan atribut yang telah ditentukan sebelumnya. Contohnya adalah untuk mengelompokkan pelanggan ke dalam kategori seperti pelanggan loyal, pelanggan berpotensi, dan sebagainya.
7. Analisis deret waktu: metode ini digunakan untuk menganalisis data yang terjadi seiring waktu, seperti tren penjualan atau perilaku pelanggan.
8. Analisis teks: metode ini digunakan untuk mengekstraksi informasi dari data teks, seperti ulasan produk atau dokumen bisnis.

Setiap metode data mining memiliki kegunaan yang berbeda dan dapat membantu mengidentifikasi pola dan wawasan yang berbeda dalam data.

Judul : metode regresi linier sederhana

Deksripsi : prediksi harga rumah dengan regresi linier Dataset : https://tifupb.id/hargarumah

Libray linier regesri : sklearn

Instalasi : pip install -U scikit-learn scipy matplotlib

Analisis Regresi Linear adalah metode statistika yang digunakan untuk membentuk model hubungan antara variabel terikat/dependen (Y) dengan satu atau lebih variabel bebas/independen (X). Berdasarkan banyaknya variabel bebas yang ada dalam model, Regresi Linear dibagi menjadi 2 jenis yaitu : Regresi Linear sederhana dan Regresi Linear berganda. Apabila banyaknya variabel bebas (X) hanya ada satu, maka disebut sebagai Regresi Linear sederhana. Sedangkan apabila terdapat lebih dari 1 variabel bebas (X) maka disebut sebagai Regresi Linear berganda.

1. Import libarary yang dibutuhkan

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import linear_model
```

2. Ambil dataset rumah dengan pandas kemudian tampilkan datanya

```
df = pd.read_csv("datarumah.csv")
print(df)
```

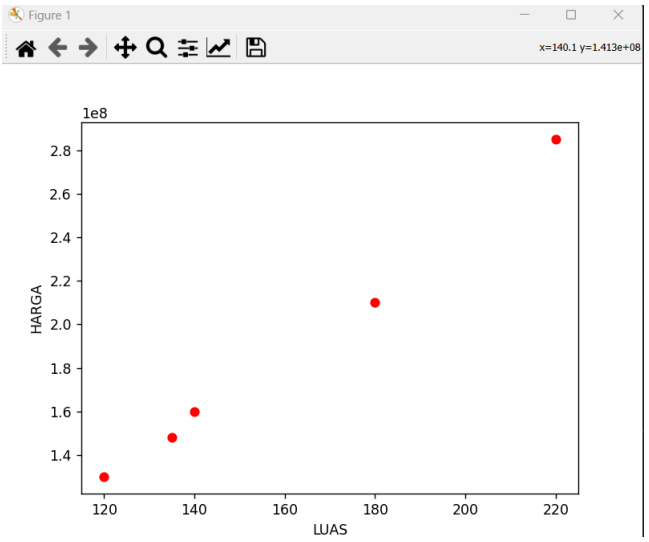
Output :

```
PS D:\praktikum data mining\python> python modul4.py
   luas      harga
0   120  130000000
1   135  148000000
2   140  160000000
3   180  210000000
4   220  285000000
```

3. Visualisasi data menggunakan scater plot

```
plt.xlabel("LUAS");
plt.ylabel("HARGA")
plt.scatter(df.luas, df.harga, color='red')
plt.show()
```

Output



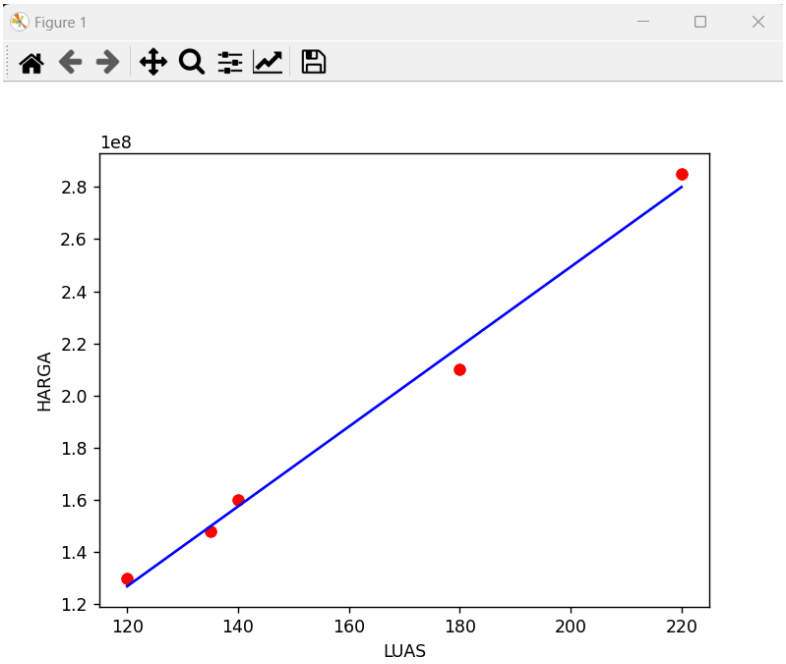
4. Membuat model agar mempelajari data dengan sendirinya

```
model_regresi_linier  = linear_model.LinearRegression()
model_regresi_linier.fit(df[['luas']], df.harga)
```

5. Visualisasi data sebenarnya dengan garis prediksi

```
plt.xlabel("LUAS")
plt.ylabel("HARGA")
plt.scatter(df.luas, df.harga, color='red')
plt.plot(df.luas, model_regresi_linier.predict(df[['luas']]), color='blue')
plt.show()
```

Output



6. Prediksi harga untuk rumah seluas 140 M2

```
hasil = int(model_regresi_linier.predict([[140]]))
print(hasil)
```

Praktikum 8

Streamlit

Streamlit adalah sebuah library open-source Python yang digunakan untuk membuat aplikasi web interaktif untuk data science dan machine learning dengan mudah dan cepat. Streamlit memungkinkan pengguna untuk membuat aplikasi web interaktif dengan kode Python sederhana, sehingga dapat dengan mudah memvisualisasikan data, membuat model machine learning, dan mengeksplorasi dataset secara interaktif.

Beberapa fitur dan keuntungan dari Streamlit antara lain:

1. Mudah digunakan: Streamlit dirancang untuk mempermudah pembuatan aplikasi web interaktif. Dengan Streamlit, pengguna dapat membuat aplikasi web dengan kode Python yang sederhana dan mudah dipahami.
2. Terintegrasi dengan mudah: Streamlit dapat diintegrasikan dengan berbagai library Python yang populer untuk data science dan machine learning seperti NumPy, Pandas, Matplotlib, dan Scikit-Learn.
3. Menampilkan data secara interaktif: Streamlit menyediakan berbagai jenis widget interaktif yang dapat digunakan untuk menampilkan data dengan cara yang lebih mudah dipahami seperti plot, tabel, dan grafik.
4. Deploy dengan mudah: Streamlit menyediakan fitur untuk deploy aplikasi web secara mudah dengan layanan hosting seperti Heroku, Google Cloud Platform, dan Amazon Web Services.

Dalam pengembangan aplikasi web interaktif dengan Streamlit, pengguna dapat memanfaatkan fitur-fitur seperti:

- Menggunakan widget seperti slider, checkbox, dan radio button untuk memudahkan interaksi pengguna dengan aplikasi.
- Menggunakan library visualisasi data seperti Matplotlib dan Seaborn untuk memvisualisasikan data dengan berbagai jenis plot.
- Menggunakan library machine learning seperti Scikit-Learn untuk membangun model machine learning dan memprediksi hasil berdasarkan input yang diberikan oleh pengguna.
- Menambahkan layout dan komponen HTML dan CSS untuk membuat tampilan yang lebih menarik dan profesional.

Streamlit dapat digunakan oleh siapa saja yang ingin membuat aplikasi web interaktif dengan mudah dan cepat, baik bagi pengguna yang belum memiliki pengalaman dalam pengembangan web maupun bagi pengguna yang sudah berpengalaman. Dalam penggunaannya, pengguna dapat memanfaatkan dokumentasi resmi Streamlit dan komunitas pengguna yang aktif untuk mendapatkan bantuan dan dukungan.

Dokumentasi slengkapnya bisa dilihat disini :

https://docs.streamlit.io/library/get-started Instalasi

Pip Install streamlit

Cara menjalankan
streamlit run namafile.py

contoh : membuat aplikasi menhitung segitiga

1. Panggil library yang dibutuhkan

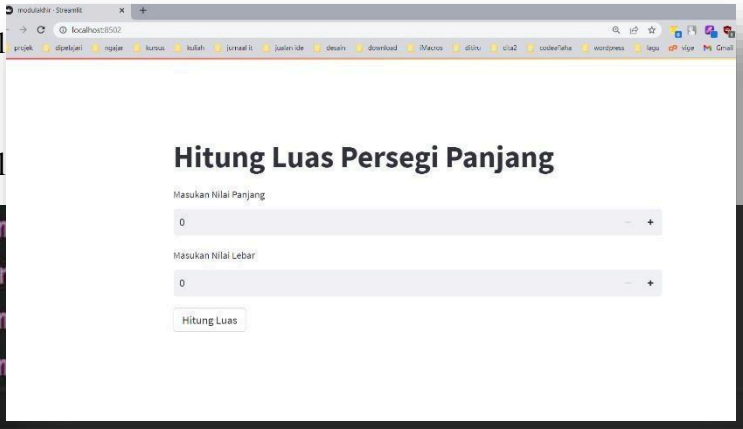
```
import streamlit as st
st.title('Hitung Luas Persegi Panjang')
panjang = st.number_input ("Masukan Nilai Panjang", 0)
lebar = st.number_input ("Masukan Nilai Lebar", 0)
hitung = st.button ("Hitung Luas")

if hitung :
    luas = panjang * lebar
    st.write ("Luas Persegi Panjang Adalah = ",luas)
    st.success("Luas Persegi Panjang Adalah = {luas}")
```

2. Jalankan dengan perintah berikut

```
PS D:\praktikum data mining\python> streamlit run .\modulakhir.py

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8502
  Network URL: http://10.10.82.213:8502
```

3. Output bisa dibuka di url diatas

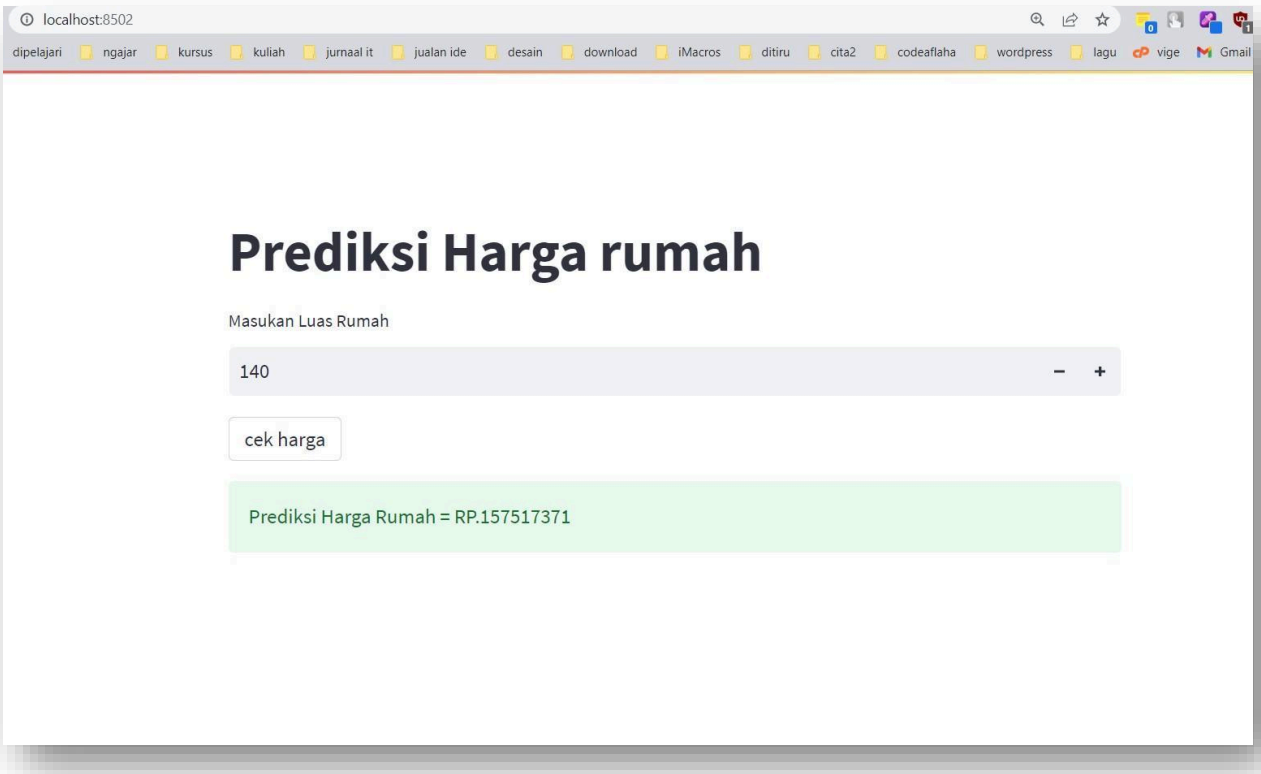
Depl



Tul

```
df = pd.read_csv("datarumah.csv")

model_regresi_linier  = linear_model.LinearRegression()
model_regresi_linier.fit(df[['luas']], df.harga)

st.title('Prediksi Harga rumah')
luas = st.number_input ("Masukan Luas Rumah", 0)

if st.button ("cek harga") :
    hasil =  int(model_regresi_linier.predict([[luas]]))
    st.success(f"Prediksi Harga Rumah = RP.{hasil}")
```

Output


localhost:8502
dipelajari
ngajar
kursus
kuliah
jurnaal it
jualan ide
desain
download
iMacros
ditiru
cita2
codeaflaha
wordpress
lagu
vige
Gmail
Prediksi Harga rumah
Masukan Luas Rumah
140
cek harga
Prediksi Harga Rumah = RP.157517371