Library Numpy

NumPy (Numerical Python) adalah library Python yang fokus pada scientific computing.

Numpy menyediakan fungsi yang siap pakai untuk memudahkan kita melakukan perhitungan saintifik seperti matriks, aljabar, statistik, dan sebagainya. List dan math yang sudah terdapat di Python memang bisa dipakai untuk perhitungan saintifik. Namun masih kurang lengkap, karena beberapa operasi perhitungan harus dibuat secara manual.

Contoh: Jika ingin menghitung hasil penjumlahan tiap elemen di list.

```
# kita punya list a dan b
a = [1, 2, 3]
b = [4, 5, 6]
# lalu kita jumlahkan
hasil = a + b
print(hasil) # [1, 2, 3, 4, 5, 6]
```

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

Hasilnya tidak sesui dengan yang diharapkan. Seharusnya tiap elemen dijumlahkan, tapi malah list-nya digabungkan.

Jika ingin menjumlahkan, maka harus membuat rumus atau fungsi secara manual seperti ini:

```
def add(list_a, list_b):
  result = []
  for first, second in zip(list_a, list_b):
    result.append(first + second)
  return result


hasil = add(a,b)
print(hasil) #[5,7,9]
```

```
[5, 7, 9]
```

Karena itu, supaya tidak secara manual dari nol, sebaiknya pakai yang sudah ada dari Numpy.

Selain itu, performa Numpy juga lebih cepat dibandingkan list karena library Numpy ditulis dengan bahasa C dan sebagaian lagi Python.

```
import numpy as np

#membuat array dengan numpy
mylist = [1,2,3]
myarray = np.array(mylist)

print(type(mylist))
print(type(myarray))
print(myarray.shape)
print(myarray)
```

```
<class 'list'>
<class 'numpy.ndarray'>
(3,)
[1 2 3]
```

```
#access data pada numpy array
mylist_1 = [[1,2,3], [4,5,6]]
myarray_1 = np.array(mylist_1)
print(myarray_1)
print(myarray_1.shape)

print("Baris pertama:", myarray_1[0])
print("Baris kedua:", myarray_1[1])
print("Baris pertama kolom ketiga:", myarray_1[0][2])
```

```
[[1 2 3]
 [4 5 6]]
(2, 3)
Baris pertama: [1 2 3]
Baris kedua: [4 5 6]
Baris pertama kolom ketiga: 3
```

Bagaimana jika anda ingin mengakses data kolom ke-2 pada myarray_1?

```
#operasi aritmatika pada numpy array

myarray1 = np.array([1,2,3])
myarray2 = np.array([4,5,6])

print("Penjumlahan array: ", myarray1 + myarray2)
```

```
print("Perkalian array: ", myarray1 * myarray2)
```

```
Penjumlahan array:  [5 7 9]
Perkalian array:  [ 4 10 18]
```

Perbedaan Array dan List

Perbedaannya adalah Array harus dideklarasikan terlebih dahulu apakah dia di bawah modul array atau package numpy, sedangkan list tidak perlu. Itulah mengapa list lebih sering digunakan jika dibandingkan array. Perbedaan lainnya adalah array lebih handal dan efisien dalam menyimpan data yang begitu compact. Selain itu, array juga lebih unggul untuk operasi numerik. Untuk membuktikannya, kita akan mencoba membagi antara array dan list.

```
array = np.array([3, 6, 9, 12])
division = array/3
print(division)
print (type(division))
```

```
[1. 2. 3. 4.]
<class 'numpy.ndarray'>
```

```
#bagaimana dengan list?
list = [3, 6, 9, 12]
division = list/3
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-18-01e33d141aff> in <cell line: 0>()
      1 #bagaimana dengan list?
      2 list = [3, 6, 9, 12]
----> 3 division = list/3

TypeError: unsupported operand type(s) for /: 'list' and 'int'
```

Langkah berikutnya: Jelaskan error

Meskipun hasil pembagian ini error, namun kita tetap bisa menggunakan list untuk operasi matematika lainnya, kok. Namun kurang efisien jika dibandingkan array.

Library Matplotlib

Matplotlib adalah library python paling populer untuk melakukan visualisasi data yang lebih menarik dan mudah dipahami sehingga matplotlib akan terasa lebih alami untuk dipelajari. Matplotlib disusun oleh John Hunter di tahun 2002, dan didesain agar dapat digunakan selayaknya menggunakan MATLAB. Matplotlib dapat digunakan untuk memvisualisasikan data secara 2D maupun 3D dan menghasilkan gambar berkualitas yang bahkan dapat kamu simpan dalam berbagai format gambar, seperti JPEG dan PNG. Jika, kamu menggunakan python script maka kamu perlu menginstall matplotlib terlebih dahulu. Tapi, jika kebetulan kamu menggunakan python melalui anaconda meliputi spyder dan jupyter notebook maka, kamu tidak perlu menginstalnya lagi karena sudah menjadi built-in library.

Selain karena mudah dipelajari, matplotlib memiliki banyak contoh grafik yang bisa digunakan untuk memvisualisasikan data agar menjadi lebih menarik. Antara lain seperti, bar plot yang merupakan jenis grafik yang paling sering digunakan untuk merepresentasikan data numerik dan kategorik dalam bentuk bar. Lalu ada histogram yang digunakan untuk merepresentasikan distribusi frekuensi dan data numerik dengan batang. Selain itu, line plot yaitu jenis grafik untuk menampilkan informasi dengan menggunakan banyak titik yang saling terhubung dan membentuk garis lurus atau lengkung. Ada pula box plot untuk membuat bentuk visualisasi data secara statistik melalui lima dimensi utama yaitu nilai minimum, kuartil 1, kuartil 2, kuartil 3 dan nilai maksimum, box plot sering digunakan untuk memeriksa keberadaan outlier, dan masih banyak ada banyak lagi grafik yang dapat kamu gunakan untuk memvisualisasikan data.

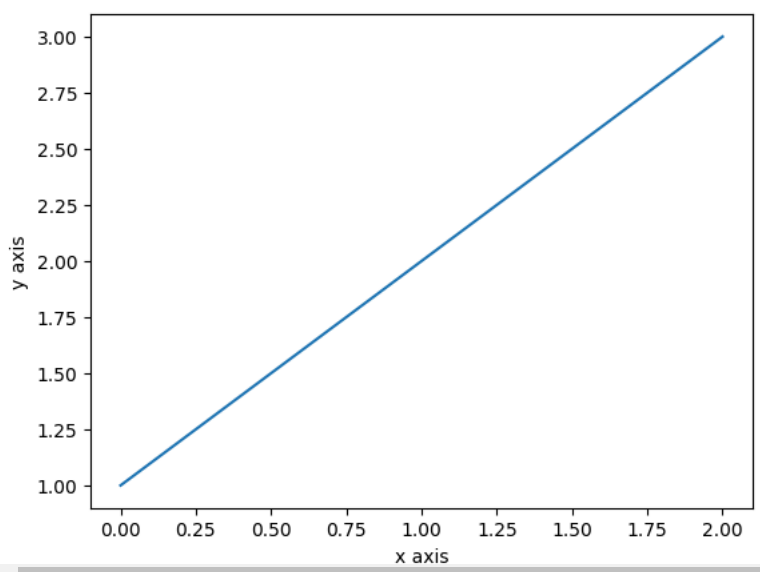
Jika, sudah memilih grafik yang ingin diterapkan untuk memvisualisasikan data. Selanjutnya, setiap elemen dalam gambar yang diwakili oleh Matplotlib dan masing-masing memiliki daftar properti yang luas untuk mengkonfigurasi penampilannya. Angka mengandung persegi panjang persis ukuran gambar, yang dapat digunakan untuk mengatur warna latar belakang dan transparansi angka-angka.

Matplotlib memiliki sejumlah colormaps bawaan yang dapat diakses melalui matplotlib.cm.get.cmap. Ada di luar library seperti installable yang memiliki banyak colormaps ekstra. Untuk memilih colormaps dan mengaksesnya yang pertama, mendapatkan named colormap, yang sebagian besar tercantum dalam memilih colormap di matplotlib, dapat dilakukan menggunakan matplotlib.cm.get_cmap, yang mengembalikan objek colormap. Membuat colormap ada dasarnya adalah operasi menyediakan list atau array spesifikasi warna ke listed colormap untuk membuat colormap baru.

Tutorial Matplotlib dapat diakses pada https://matplotlib.org/stable/tutorials/index.html#

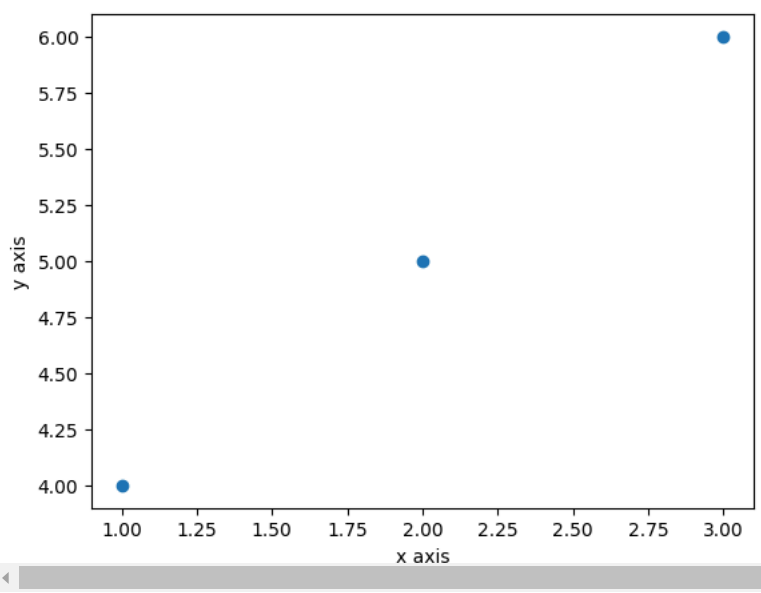
```
#kita membuat basic line plot
import matplotlib.pyplot as plt
import numpy as np
```

```
myarray = np.array([1,2,3])
plt.plot(myarray)
plt.xlabel('x axis')
plt.ylabel('y axis')
plt.show()
```




```
#membuat scatter plot
x = np.array([1,2,3])
y = np.array([4,5,6])

plt.scatter(x,y)
plt.xlabel('x axis')
plt.ylabel('y axis')
plt.show()
```



Library Pandas

Pandas adalah sebuah library di Python yang berlisensi BSD dan open source yang menyediakan struktur data dan analisis data yang mudah digunakan. Pandas biasa digunakan untuk membuat tabel, mengubah dimensi data, mengecek data, dan lain sebagainya. Struktur data dasar pada Pandas dinamakan DataFrame, yang memudahkan kita untuk membaca sebuah file dengan banyak jenis format seperti file .txt, .csv, dan .tsv. Fitur ini akan menjadikannya table dan juga dapat mengolah suatu data dengan menggunakan operasi seperti join, distinct, group by, agregasi, dan teknik lainnya yang terdapat pada SQL.

Library Pandas memiliki dua tipe struktur data untuk versi terbaru yaitu Series dan Data Frame serta satu deprecated struktur data yaitu Panel (deprecated). Series diibaratkan sebagai array satu dimensi sama halnya dengan numpy array, hanya bedanya mempunyai index dan kita dapat mengontrol index dari setiap elemen tersebut. Sedangkan data frame merupakan array dua dimensi dengan baris dan kolom. Struktur data ini merupakan cara paling standar untuk menyimpan data dalam bentuk tabel/data tabular. Dapat disimpulkan, bahwa Pandas merupakan library analisis data yang diperlukan untuk membersihkan data mentah ke dalam sebuah bentuk yang bisa untuk diolah.

Mencoba Series

Series merupakan struktur data dasar dalam Pandas. Series diibaratkan sebagai array satu dimensi sama halnya dengan numpy array, hanya bedanya mempunyai index dan index tersebut dapat kita kontrol dari setiap elemen tersebut. Perintah dasar untuk membuat sebuah series dengan Pandas adalah

```
pandas.Series(data=None, index=None, dtype=None, name=None, copy=False, fastpath=False)
```

Parameter data, diisi dengan data yang akan dibuat series. Struktur data yang bisa ditampung berupa integer, float, dan juga string. parameter index, diisi dengan index dari series. Jumlah index harus sama dengan jumlah data. Jika kita tidak mengisi parameter index, maka series akan memiliki index integer seperti halnya array biasa. Parameter dtype, diisi dengan tipe data dari series, dan parameter copy untuk copy data, secara default akan bernilai false. Berdasarkan sintaks di atas, kita akan membuat contoh series menggunakan Python list ataupun numpy sebagai contoh data untuk membuat series.

```
import pandas as pd
import numpy as np

data = pd.Series(['a','b','c','d'])
np1 = np.array(['a','b','c','d'])
data_np = pd.Series(np1)

print(data)
print(data_np)
```

```
0    a
1    b
2    c
3    d
dtype: object
0    a
1    b
2    c
3    d
dtype: object
```

Mengatur Indeks

Untuk mengatur indeks dari series, kita bisa menggunakan parameter index. Contohnya disini akan dilakukan pengaturan indeks pada data numpy array yang sudah dibuat sebelumnya dengan indeks [12,13,14,15].

```
data_np = pd.Series(np1, index=[12,13,14,15])

print(data_np)
```

```
12    a
13    b
14    c
15    d
dtype: object
```

Terlihat bahwa indeks dari series berubah menjadi [12,13,14,15]. Perlu diingat bahwa pengaturan indeks harus sesuai dengan panjang dari data yang ada, bila tidak sesuai akan menimbulkan error panjang dari indeks tidak sesuai.

Mencoba Data Frame

Data frame merupakan tabel/data tabular dengan array dua dimensi yaitu baris dan kolom. Struktur data ini merupakan cara paling standar untuk menyimpan data. Setiap kolom pada data frame merupakan objek dari Series, dan baris terdiri dari elemen yang ada pada Series.

Untuk membuat data frame, digunakan sintaks berikut:

```
pandas.DataFrame(data, index, columns, dtype, copy)
```

Dengan keterangan:

- index merupakan label untuk baris
- columns merupakan label untuk kolom
- dtype merupakan tipe data per kolom
- copy digunakan untuk menyalin data, defaultnya False

Berikut contoh untuk menunjukkan pembuatan Data Frame yang sederhana:

```
import pandas as pd
import numpy as np

dict = {"Negara": ["Indonesia", "Jepang", "India", "China", "Amerika Serikat"],
        "Ibu Kota": ["Jakarta", "Tokyo", "New Delhi", "Beijing", "Washington D.C."],
        "Luas" : [1905, 377972, 3287, 9597, 9834],
        "Populasi" : [264, 143, 1252, 1357, 5298]}

df = pd.DataFrame(dict)

print(df)
```

```
            Negara         Ibu Kota    Luas  Populasi
0        Indonesia          Jakarta    1905       264
1           Jepang            Tokyo  377972       143
2            India        New Delhi    3287      1252
3            China          Beijing    9597      1357
4  Amerika Serikat  Washington D.C.    9834      5298
```

LATIHAN

1. Buat array 1 dimensi dengan elemen dari 1 hingga 10. Hitung:

- Rata-rata dari elemen array.
- Elemen maksimum dan minimum.
- Jumlah semua elemen dalam array.

2. Buat array 2 dimensi dengan ukuran 3x3, isi dengan angka acak dari 0 hingga 20.

- Tambahkan 5 ke semua elemen array.
- Cari elemen dengan nilai maksimum di setiap kolom.

3. Buat array 1 dimensi dengan elemen genap dari 2 hingga 20. Tampilkan:

- Elemen pertama hingga ke-5.
- Semua elemen dengan indeks ganjil.

4. Buat grafik garis sederhana untuk y=x^2, dengan x dari -10 hingga 10.

- Tambahkan judul, label sumbu, dan grid pada grafik.

5. Buat DataFrame dengan kolom berikut:

```
Name: ['Anna', 'Ben', 'Charlie', 'Diana', 'Edward']
Age: [23, 21, 25, 22, 24]
Score: [85, 90, 88, 79, 95]
```

Lakukan operasi berikut:

- Tampilkan data mahasiswa dengan skor di atas 85.
- Tambahkan kolom baru bernama Passed (True jika skor >= 80, False jika tidak).