MODUL 3

(MINGGU KETIGA)

Praktikum 5

Judul : mengenal 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.

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

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

```python
import numpy as np
a = np.array(42)
b = np.array([1, 2, 3, 4, 5])
c = np.array([[1, 2, 3], [4, 5, 6]])
d = np.array([[[1, 2, 3], [4, 5, 6]], [[1,2, 3], [4, 5, 6]]])
print(a.ndim)
print(b.ndim)
print(c.ndim)
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.

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

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.

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

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. Perhatikan contoh berikut.

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

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.

```python
import numpy as np
arr = np.array([1, 2, 3, 4, 5, 6, 7])
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.

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

```python
import numpy as np
arr = np.array([1.1, 2.1, 3.1])
newarr = arr.astype('i')
print(newarr)
print(newarr.dtype)
```

Output

```
PS C:\praktikum> python .\tes.py
[1 2 3]
int32
```

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.

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

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

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

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

Perhatikan contoh berikut.

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

```python
import numpy as np
arr = np.array([1, 2, 3])
for x in arr:
    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().

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

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

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

Output

```
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.

tes.py > ...

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

array_1.py > ...

```
1  import numpy as np
2
3  arr = np.array([3, 2, 0, 1])
4  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:

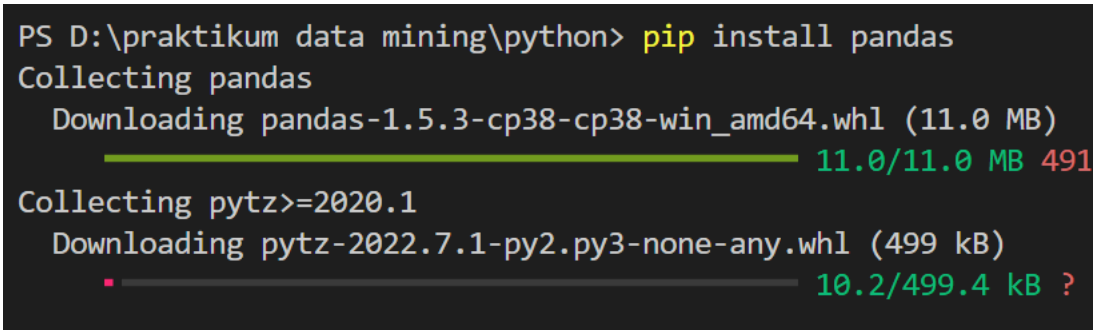
1. Dataframe: Dataframe adalah struktur data 2 dimensi (baris dan kolom) yang dapat menyimpan berbagai tipe data. Dataframe memungkinkan pengguna untuk membaca data dari berbagai format file, seperti CSV, Excel, SQL, dan lainnya.
2. Series: Series adalah struktur data 1 dimensi (hanya memiliki baris) 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.