Praktikum_TT_M2_ElmoAllistair_12118220

March 18, 2021

0.1 Python Matriks dan NumPy Array

Matriks adalah struktur data dua dimensi di mana angka-angka disusun menjadi baris dan kolom. Contohnya :

Matriks ini adalah matriks 3x4 (dibaca "tiga kali empat") karena memiliki 3 baris dan 4 kolom.

0.1.1 Python Matriks

Python tidak memiliki tipe built-in untuk matriks. Namun, kita dapat memperlakukan list di dalam list sebagai matriks. Sebagai contoh:

[1]:
```
A = [[1, 4, 5],
     [-5, 8, 9]]
```

Kita dapat memperlakukan list ini sebagai matriks yang memiliki 2 baris dan 3 kolom.

Contoh Program

[2]:
```
A = [[1, 4, 5, 12],
     [-5, 8, 9, 0],
     [-6, 7, 11, 19]]
```

Untuk mencetak atau mendapatkan seluruh nilai yang ada pada matriks di atas dapa digunakan perintah :

[3]:
```
print("A =", A)
```

```
A = [[1, 4, 5, 12], [-5, 8, 9, 0], [-6, 7, 11, 19]]
```

Selain itu, kita juga dapat mendapatkan elemen tertentu dari matriks tersebut dengan cara :

[4]:
```
print("A[1] =", A[1])       # Baris ke-2

print("A[1][2] =", A[1][2])    # Elemen ke-3 dari baris ke-2

print("A[0][-1] =", A[0][-1])    # Elemen terakhir dari Baris ke-1
```

```
A[1] = [-5, 8, 9, 0]
A[1][2] = 9
A[0][-1] = 12
```

0.1.2 NumPy Array

NumPy adalah sebuah library untuk komputasi ilmiah yang memiliki dukungan untuk objek array berdimensi-N. NumPy menyediakan deretan angka multidimensi (yang sebenarnya adalah sebuah objek). Contohnya:

[5]:
```
import numpy as np

a = np.array([1, 2, 3])

print(a)

print(type(a))
```

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

Seperti yang Anda lihat, kelas array NumPy disebut ndarray.

0.1.3 Bagaimana cara membuat array NumPy?

Ada beberapa cara untuk membuat array NumPy.

1. Array bilangan bulat, float dan Bilangan kompleks

[6]:
```
A = np.array([[1, 2, 3], [3, 4, 5]]) # Bilangan bulat
print(A)
```

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

[7]:
```
A = np.array([[1.1, 2, 3], [3, 4, 5]]) # float
print(A)
```

```
[[1.1 2.  3. ]
 [3.  4.  5. ]]
```

[8]:
```
A = np.array([[1, 2, 3], [3, 4, 5]], dtype = complex) # bilangan kompleks
print(A)
```

```
[[1.+0.j 2.+0.j 3.+0.j]
 [3.+0.j 4.+0.j 5.+0.j]]
```

---

2. Array angka nol dan satu

[9]:
```
zeroes_array = np.zeros( (2, 3) )
print(zeroes_array)
```

```
[[0. 0. 0.]
 [0. 0. 0.]]
```

Selain itu, kita juga dapat menetapkan `dtype` menjadi 32 bit (4 byte). Karenanya, array ini dapat mengambil nilai dari -2-31 hingga 2-31-1.

[10]:
```
ones_array = np.ones( (1, 5), dtype=np.int32 ) # menentukan dtype
print(ones_array)
```

```
[[1 1 1 1 1]]
```

0.1.4 Operasi Matriks

Pada contoh program selanjutnya akan dibuat 3 jenis program yaitu : penjumlahan dua buah matriks, perkalian dua buah matriks dan transpos sebuah matriks. Jika menggunakan program python biasa, kita memerlukan `nested lists` untuk membuat program tersebut. Namun ada cara yang lebih baik yaitu menggunakan library NumPy.

Penambahan Dua Matriks Kita dapat menggunakan operator + untuk menambahkan elemen yang sesuai dari dua matriks NumPy.

[11]:
```
A = np.array(
    [[3, 4],
     [2, 1]])

B = np.array(
    [[1, 5],
     [3, 7]])

C = A + B

print(C)
```

```
[[4 9]
 [5 8]]
```

Perkalian Dua Matriks Untuk mengalikan dua matriks, kita menggunakan metode `dot()`.

Catatan: * digunakan untuk perkalian array (perkalian elemen yang sesuai dari dua array) bukan perkalian matriks.

[12]:
```
A = np.array(
    [[3, 4],
     [2, 1]])

B = np.array(
    [[1, 5],
     [3, 7]])

C = A.dot(B)

print(C)
```

```
[[15 43]
 [ 5 17]]
```

Transpos Matriks Kita dapat menggunakan `numpy.transpose` untuk menghitung transpos matriks.

[13]:
```
A = np.array(
    [[3, 4],
     [2, 1]])

print(A.transpose())
```

```
[[3 2]
 [4 1]]
```

0.2 Tugas

Pada contoh program di atas, sudah di buat 3 program dengan memanfaatkan library NumPy yaitu : 1. Penambahan dua matriks 2. Perkalian dua matriks 3. Transpos matriks

Tugas anda adalah membuat 3 program tersebut tanpa menggunakan library NumPy.

---

[22]:
```
def penambahan(A, B):
    return [[*map(sum, zip(*row))] for row in zip(A, B)]

def perkalian(A, B):
```

```
    C = [[0 for row in range(len(A))] for col in range(len(B[0]))]
    for i in range(len(A)):
        for j in range(len(B[0])):
            for k in range(len(B)):
                C[i][j] += A[i][k]*B[k][j]
    return C

def transpos(A):
    return [*map(list, zip(*A))]
```

[15]:
```
penambahan(A,B)
```

[15]:
```
[[4, 9], [5, 8]]
```

[16]:
```
perkalian(A,B)
```

[16]:
```
[[15, 43], [5, 17]]
```

[23]:
```
transpos(A)
```

[23]:
```
[[3, 2], [4, 1]]
```