Materi Hari Ke-2

1. **Percabangan dengan IF Statement**

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.Dalam kondisi if-else, python dapat dijalankan dengan format:

```
if (kondisi pertama) :
     print("kata-kata")
elif (kondisi kedua) :
     print("kata-kata kedua")
else (kondisi ketiga) :
     print("kata-kata ketiga")
```

Perlu diingat bahwa titik dua (:) setelah inisiasi kondisi wajib diberikan. Jika tidak mengikuti format tersebut, maka akan error. Contoh codingnya :

```
In [4]: a=40
        b=50

In [5]: if(b > a):
            print("b is greater than A")
        elif (a == b):
            print("a and b are equal")
        else:
            print("a is greater than b")

        b is greater than A
```

Dari contoh diatas, diketahui bahwa if merupakan keyword untuk menguji kedua variabel yang telah diinisiasikan (a dan b). Dari format if diatas diartikan jika variabel b lebih besar dari variabel a, maka nanti akan ada kalimat “b is greater than a” dicetak ke layar. Selanjutnya untuk elif sendiri merupakan pengujian untuk kondisi

kedua. Dalam contoh diatas dimisalkan jika a dan b memiliki nilai yang sama, maka nanti akan ada kalimat “a is greater than b” dicetak ke layar. Elif dapat digunakan berulang kali jika ada lebih dari 2 kondisi. Selanjutnya yang terakhir yaitu else, else menangkap apa pun yang tidak tertangkap oleh kondisi sebelumnya. Contoh :

```
: a=60
```

```
: b=30
```

```
: if(b > a):
      print("b is greater than A")
  else:
      print("a is greater than b or a and b are equal")
```

```
a is greater than b or a and b are equal
```

Jika kita hanya memiliki dua kondisi saja, maka kita dapat menggunakan else.

2. **Perulangan / Looping**

Perualangan dalam bahasa pemrograman berfungsi menyuruh komputer melakukan sesuatu secara berulang-ulang. Terdapat dua jenis perualangan dalam bahasa pemrograman python, yaitu perulangan dengan for dan while.

Perulangan for disebut *counted loop* (perulangan yang terhitung), sementara perulangan while disebut *uncounted loop* (perulangan yang tak terhitung). Perbedaannya adalah perulangan for biasanya digunakan untuk mengulangi kode yang sudah diketahui banyak perulangannya. Sementara while untuk perulangan yang memiliki syarat dan tidak tentu berapa banyak perulangannya.

Berikut beberapa contoh dari **for** looping :

```
item = ['kopi','nasi','teh','jeruk']

for isi in item:
    print(isi)
```

```
kopi
nasi
teh
jeruk
```

Dari contoh diatas diketahui bahwa pertama-tama kita membuat variabel “item” berupa array yang berisi banyak benda seperti kopi,nasi,teh,dan jeruk. Selanjutnya untuk dapat mencetak isi dari variabel item tersebut, kita ketik dengan format :

```
for VariabelPengganti in VariabelUtama :
    print(VariabelPengganti)
```

Dimana VariabelPengganti disini merupakan variabel yang memisalkan variabel utama untuk dapat diakses dan digunakan. Contoh lainnya :

```
for i in range(0, 10):
    print (i)
```

```
0
1
2
3
4
5
6
7
8
9
```

Dari contoh kedua tersebut, variabel i memisalkan range angka dari 0-10 lalu nantinya i akan dicetak ke layar. Selanjutnya kita masuk ke perulangan **while**. Contoh untuk perulangan menggunakan while :

```
i = 0
```

```
while i < 5:
    print(i)
    i += 1
```

```
0
1
2
3
4
```

Dari contoh diatas, diketahui bahwa variabel i diinisiasikan sama dengan 0. Untuk proses loopingnya, while i < 5 artinya selama i < 5, nantinya i akan dicetak ke layar. Pada line ketiga yaitu i += 1 artinya setelah dicetak ke layar, nilai yang ada di variabel i akan bertambah 1. Maka hasil akhirnya akan tercetak angka 0 sampai dengan angka 4.

Bahasa pemrograman Python memungkinkan penggunaan satu lingkaran di dalam loop lain atau yang biasa disebut **nested looping.** Berikut contoh codingnya :

```
i = 2
```

```
while(i < 100):
    j = 2
    while(j <= (i/j)):
        if not(i%j): break
        j = j + 1
    if (j > i/j) : print(i, " is prime")
    i = i + 1

print("Good bye!")
```

```
2  is prime
3  is prime
5  is prime
7  is prime
11  is prime
13  is prime
17  is prime
19  is prime
23  is prime
29  is prime
31  is prime
37  is prime
41  is prime
43  is prime
47  is prime
53  is prime
59  is prime
61  is prime
67  is prime
71  is prime
73  is prime
79  is prime
83  is prime
89  is prime
97  is prime
Good bye!
```

Dari contoh diatas terlihat ada while didalam while, nah ini yang disebut nested loop. Menaruh for dalam while ataupun while dalam for sangatlah bisa dan biasanya dalam nested loop paling tidak memiliki 2 variabel yang dapat digunakan.

3. VARIABEL BERINDEX ATAU ARRAY

Array adalah sebuah kumpulan data yang memiliki tipe taya yang sama. Pada setiap nilai yang berada pada array yaitu dapat disebut dengan *element*. dengan tipe data apapun yang digunakan oleh array. Pada array memiliki nomor data yang dapat menunjuk pada setiap element array yang dapat disebut dengan subscript atau offset.

3.1. Numpy Array

Penggunaan array pada python juga dapat ditambahkan dengan beberapa library salah satunya library numpy. Numpy library merupakan salah satu library dari python yang dapat digunakan untuk dapat bekerja dengan array. Pada numpy library memiliki fungsi untuk bekerja dalam konsep domain aljaar linier, transformasi fourier dan matriks.

```
In [4]: from IPython.core.interactiveshell import InteractiveShell
        InteractiveShell.ast_node_interactivity = "all"

        # import numpy library
        import numpy as np
        from scipy import stats
```

Gambar 3.1. Import numpy library.

Langkah pertama pada penerapan numpy library untuk matrix yaitu dengan melakukan import library pada jupyter notebook yang bertujuan agar variable yang terdapat pada numpy library dapat digunakan untuk melakukan perhitungan matrix dan vector.

```
In [11]: arr = np.array([5, 15, 20])
         print (arr)
         print ("shape of the array:", arr.shape)
         print ("type of the array:", arr.dtype)

         [ 5 15 20]
         shape of the array: (3,)
         type of the array: int32
```

Gambar 3.2. Array satu dimensi.

Setelah import numpy library dapat dilakukan insialisasi library dengan menggunakan kata np yang bertujuan untuk memepermudah pemanggilan numpy library jika ingin digunakan.

Pada tahap awal menggunakan library numpy dapat dilakukan pembuatan penerapan array dalam 1 dimensi terlebih dahuluseperti contoh di atas. Pada array 1 dimensi tersebut berisikan 3 buah data dengan tipe data *integer.*

```
In [7]: arr = np.array([[5., 3.], [9., 2.]])
        print (arr)
        print ("shape of the array:", arr.shape)
        print ("type of the array:", arr.dtype)

        [[5. 3.]
         [9. 2.]]
        shape of the array: (2, 2)
        type of the array: float64
```

Gambar 3.3. Array dua dimensi.

Selain penerapan numpy library pada array dengan dimensi satu, numpy library dapat digunakan dengan penerapan array dua dimensi. Pada array dua dimensi dapat digambarkan dengan sebuah larik didalam larik atau penerapan pada matematis yaitu pada penerapan suatu matriks. Pada contoh di atas pemanggilan array dua dimensi dengan didahului dengan pemanggilan *numpy library* terlebih dahulu dan menghasilkan sebuah matriks 2 x 2.

```
# array of zeros
zeros = np.zeros((2, 2))
print (zeros)
print ("shape of the array:", zeros.shape)
print ("type of the array:", zeros.dtype)
```

```
[[0. 0.]
 [0. 0.]]
shape of the array: (2, 2)
type of the array: float64
```

Gambar 3.4. fungsi zeros

```
# we can also specify the data type
zeros = np.zeros((2, 2), dtype=int)
print (zeros)
print ("shape of the array:", zeros.shape)
print ("type of the array:", zeros.dtype)
```

```
[[0 0]
 [0 0]]
shape of the array: (2, 2)
type of the array: int32
```

Gambar 3.5. fungsi zeros

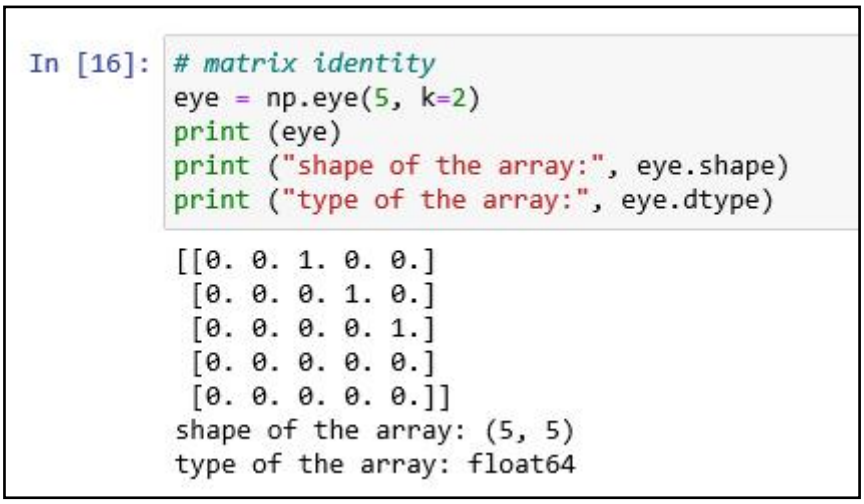
Penerapan *numpy library* pada fungsi zeros. Zeros function dapat digunakan untuk membuat sebuah matrix baru dengan nilai 0 dan menggunakan tipe data tertentu. Penggunaan dtype pada gambar 5.6 dapat digunakan jika ingin menentukan tipe data yang akan digunakan dalam membuat sebuah matrix dengan fungsi zeros, hal tersebut tipe data yang merupakan nilai default pada fungsi zeros merupakan tipe data float.

```
In [12]: # array of ones
         ones = np.ones((3, 4))
         print (ones)
         print ("shape of the array:", ones.shape)
         print ("type of the array:", ones.dtype)
```

```
[[1. 1. 1. 1.]
 [1. 1. 1. 1.]
 [1. 1. 1. 1.]]
shape of the array: (3, 4)
type of the array: float64
```

Gambar 3.6. Fungsi Ones.

Berbeda dengan fungsi sebelumnya yaitu fungsi zeros. Pada fungsi numpy library ini yaitu fungsi ones merupakan sebuah fungsi yang dapat membentuk suatu matriks dengan memiliki nilai pada setiap elemennya berupa nilai satu. Fungsi ini sangat mirip dengan fungsi numpy zeros yaitu memiliki tipe data dengan default float.



Gambar 3.7. Fungsi eye().

Pada fungsi eye() dari *numpy library* merupakan salah satu metode dari *numpy library* yang dapat mengembalikan suatu nilai dengan nilai satu. Pada fungsi eye() mirip dengan fungsi ones() namun, yang membedakan dari nilai satu yang membentuk suatu diagonal dari sebuah matriks.

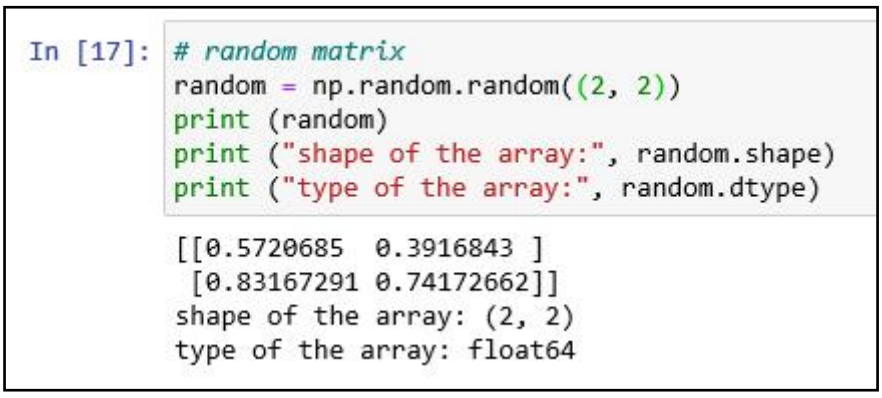
```
numpy.eye(N, M=None, k=0, dtype=<class 'float'>)
```

Pada fungsi eye() memiliki beberapa parameter yang dapat digunakan untuk membuat sebuah matriks, yaitu :

N : menentukan jumlah baris.

M : menentukan jumlah kolom.

K : dapat menentukan posisi diagonal akan dibuat pada sebuah matriks.



Gambar 3.8. Fungsi Random.

Fungsi random() pada *numpy library* memiliki fungsi untuk mengacak suatu nilai dari sebuah data. Jika ingin membuat sebuah matriks 2 x 2 maka dapat dilakukan pemanggilan np.random.random((2,2)). Tipe data yang digunakan dalam matrix tersebut akan secara default berupa float.

3.2. Slicing

Slicing merupakan salah satu fitur yang dapat digunakan dalam penerapan *numpy library,* pada fitur ini dapat mengakses berbagai bagian dari *sequences* seperti :string, tuples dan list. Selain itu pada fitur Slicing dapat digunakan untuk melakukan pengubahan atau penghapusan dari item dari sebuah list yang ada.

```
In [36]: random[:, 1]
Out[36]: array([0.70735219, 0.1828092 , 0.44560817, 0.37722057, 0.45333538])
```

Gambar 3.9. Slicing fungsi Random ().

Salah satu penerapan dari *slicing* yaitu dengan membuat sebuat list dalam bentuk array 1 dimensi yang memiliki default data sebanyak 5 data. Penerapan *slicing* digunakan untuk melakukan konversi dari variable random kedalam sebuah array baik array satu dimensi maupun array dua dimensi.

```
In [45]: # sub matrix
         random[:2, :2]
         random[[2, 4, 1], [1]]
         random[[2, 4, 1], 2:]

Out[45]: array([[0.29559425, 0.70735219],
                [0.36371469, 0.1828092 ]])

Out[45]: array([0.44560817, 0.45333538, 0.1828092 ])

Out[45]: array([[0.22150379, 0.05609165, 0.78988864],
                [0.38139132, 0.60017814, 0.61324742],
                [0.79988296, 0.88007635, 0.61770214]])
```

Gambar 3.10. Slicing fungsi random().

Selain itu penerapan dari *slicing* dapat digunakan untuk membentuk sebuah matrix dengan memanfaatkan hasil yang didapatkan dari pengambilan nilai random dari fungsi ramdom untuk menghasilkan sebuah matrix dalam bentuk array. Untuk

membentuk sebuah matrix 2x2 dapat dilakukan dengan menggunakan random[:2,:2], penentuan penggunaan slicing tersebut berdasarkan pada [start:end].

3.3. Boolean array indexing

Boolean array indexing dapat dikatakan berjalan ketika object array bertipe bolean, seperti nilai yang dapat dikembalikan dari operator perbandingan. Pada Boolean array indexing dapat dilulis seperti dibawah ini x [obj.nonzero ()].

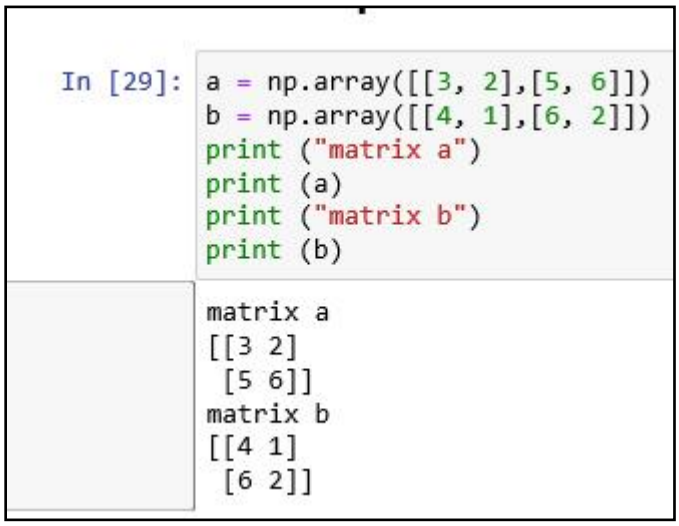
```
In [26]: # boolean indexing
         random[random < 0.3]
         random

Out[26]: array([0.26008441, 0.0890574 , 0.01285795, 0.11000257, 0.05850891])

Out[26]: array([[0.26008441, 0.64151813, 0.35072454, 0.0890574 , 0.58393984],
                [0.33819663, 0.64368152, 0.80713434, 0.01285795, 0.87754789],
                [0.8579803 , 0.6593731 , 0.11000257, 0.45805826, 0.87193956],
                [0.33841485, 0.44895781, 0.63316046, 0.62403213, 0.73997819],
                [0.63931743, 0.05850891, 0.94942485, 0.7344217 , 0.43362076]])
```

Gambar 3.11. Slicing fungsi random().

Penerapan Boolean array indexing dapat diterapkan dengan fungsi random pada numpy library seperti pada contoh di atas yaitu dengan menghasilkan dua matriks yang berbeda satu nanya menerapkan fungsi random pada numpy library dan lainnya menerapkan Boolean array indexing.

3.4. Operasi Matriks

Seperti yang telah dijabarkan pada penjelasan mengenai array dua dimensi, array dua dimensi dapat membentuk suatu matrix.



Gambar 3.12. Matrix

Pada kedua matrix tersebut dapat di lakukan operasi matematis untuk menentukan hasil dari operasi yang dilakukan pada matrix seperti penjumlahan, pengurangan, perkalian, dan pembagian.

Table 3.1. Operasi Matrix

In [33]: `# sum of two or more arrays` `a+b` `a+3` Out[33]: array([[7, 3], [11, 8]]) Out[33]: array([[6, 5], [8, 9]])	Penjumlahan Matrix
In [31]: `# difference` `a - b` Out[31]: array([[-1, 1], [-1, 4]])	Pengurangan Matrix
In [23]: `# element-wise division` `a / b` `a / b.astype(float)` Out[23]: array([[0.2 , 0.33333333], [0.42857143, 0.5]]) Out[23]: array([[0.2 , 0.33333333], [0.42857143, 0.5]])	Pembagian Matrix

In [34]: ```# element-wise multiplication a * b a * 2``` Out[34]: ```array([[12, 2], [30, 12]])``` Out[34]: ```array([[6, 4], [10, 12]])```	Perkalian Matrix
In [22]: ```# square np.square(a)``` Out[22]: ```array([[1, 4], [9, 16]])```	Matrix Square
In [23]: ```# power a ** 3``` Out[23]: ```array([[1, 8], [27, 64]])```	Power Square
In [24]: ```# square root np.sqrt(a)``` Out[24]: ```array([[1. , 1.41421356], [1.73205081, 2.]])```	Matrix Square Root

3.5. Numpy dot Product

Numpy dot merupakan salah satu fungsi yang dapat digunakan untuk melakukan perkalian dari titik a dan b.

In [25]:
```
# dot product of two 1-d arrays returns a single value
print (a[1, :], "dot", b[:, 1])
np.dot(a[1, :], b[:, 1])
```

```
[5 6] dot [1 2]
```

Out[25]:
```
17
```

Gambar 3.13. Fungsi dot().

Pada fungsi numpy dot juga dapat menangani array 2D dan melakukan perkalian matrix. Pada kasus seperti gambar di atas menggambarkan penerapan numpy dot product pada array 1D dengan dilakukan perkalian matrix 1D yaitu 5x1+6x2 = 17 dan hasil yang didapatkan yaitu 17.

```
In [36]: # dot product of two matrices
         print ("a dot b")
         np.dot(a, b)

         a dot b

Out[36]: array([[24,  7],
                [56, 17]])
```

Gambar 3.14. Fungsi dot().

Pada gambar di atas merupakan salah satu penerapan dari numpy dot product digunakan untuk melakukan perkalian matrix dengan memanfaatkan rumus perkalian matrix. Matrix asal dapat ditentukan sebelumnya sebelum menggunakan numpy dot product.

3.6. Fungsi SUM

Pada numpy library memiliki sebuah fungsi yang dapat digunakan untuk menjumlahkan seluruh element atau bagian dari elemen disuatu array yaitu fungsi sum

```
In [38]: # sum of all elements
         np.sum(a)

Out[38]: 16
```

Gambar 3.15. Fungsi sum().

Gambar 3.15 merupakan salah satu penerapan fungsi dari sum yang digunakan untuk menjumlahkan seluruh elemen yang terdapat dari sebuah array.

```
In [39]: # sum of elements along given axis
         np.sum(a, axis=0)
         np.sum(a, axis=1)

Out[39]: array([8, 8])

Out[39]: array([ 5, 11])
```

Gambar 3.16. Fungsi sum().

Gambar 3.16 menghasilkan hasil penjumlahan yang berdasarkan penjumlahan sepanjang sumbuh pada array. jika suatu array terdiri dari bentuk (X,Y) maka penjumlahan ini dilakukan pada sepanjang sumbu 0 dengan membentuk (1,Y) atau penjumlahan dilakukan dengan sumbu 1 maka akan membentuk (1,X).

3.7. Transpose

Sebuah konsep transpose pada suatu matrix dimana konsep tersebut dilakukan pertukaran antara baris dan kolom pada suatu matrix.

```
In [41]: a
         a.T

Out[41]: array([[3, 2],
                [5, 6]])

Out[41]: array([[3, 5],
                [2, 6]])
```

Gambar 3.16. Transpose Matrix.

Transpose matrix dapat dilakukan jika suatu matrix berawal dari $\begin{matrix} a & b \\ c & d \end{matrix}$ menjadi $\begin{matrix} a & c \\ b & d \end{matrix}$ atau dapat dilihat seperti contoh pada gambar di atas.

3.8. Reshape

Fungsi reshape() dapat digunakan untuk membentuk suatu bentuk baru dari sebuah array atau dalam penerapan sebuah matrix.

```
In [42]: np.reshape(a, (1, 4))

Out[42]: array([[3, 2, 5, 6]])
```

Gambar 3.17. Fungsi reshape().

Pada penerapan sebuah matrix fungsi reshape() akan mengubah dari bentuk $\begin{matrix} a & b \\ c & d \end{matrix}$ menjadi bentuk $[a \quad c \quad b \quad d]$ atau seperti contoh di atas yang berawal dari matrix a.T $\begin{matrix} 3 & 5 \\ 2 & 6 \end{matrix}$ menjadi bentuk $[3 \quad 2 \quad 5 \quad 6]$.