

STRUKTUR DATA (PYTHON)

“Instruksi Utama Python”

[@SUARGA] [Pertemuan 03+04]

OutLine





Instruksi Utama Python

- 1. Instruksi Seleksi:** digunakan untuk memilih instruksi yang akan di-eksekusi selanjutnya berdasarkan pada suatu syarat / kondisi. Bilamana syarat dipenuhi maka instruksi di-laksanakan, bila tidak maka hal lain dilakukan
- 2. Instruksi Perulangan:** digunakan untuk mengulang serangkaian instruksi pada perulangan yang terbatas atau tidak terbatas, namun dapat dihentikan berdasarkan suatu kondisi.
- 3. Pembuatan Fungsi :** Pembuatan Fungsi merupakan bagian penting dari pemrograman Python

Instruksi Seleksi if/else

- Instruksi **if** digunakan untuk memilih instruksi yang akan dilaksanakan berdasarkan suatu syarat atau relasi, bentuknya sebagai berikut:

if (relasi) :

true statements #kerjakan bila benar

else:

false statements #kerjakan bila salah

- Contoh apabila anda ingin membuat pilihan antara menampilkan nilai $(x+10)$ bila x bernilai kurang dari 5, dengan nilai $(x*10)$ bila x bernilai lebih dari 5, maka secara algoritmik anda menuliskan:

Algoritma:

Bila $x < 5$:

maka tampilkan $(x + 10)$

selain itu tampilkan $(x * 10)$.

Python:

```
if ( x < 5 ) :
```

```
    print( x + 10)
```

```
else:
```

```
    print( x * 10)
```

pemilihan multi-syarat

- Instruksi seleksi dapat dibuat multi syarat/relasi, dengan memakai *elif* diantaranya, bentuknya sebagai berikut.

```
if (relasi-1):
    statement-1
elif (relasi-2):
    statement-2
elif (relasi-3):
    statement-3
else:
    statement-4
```

Contoh

```
#pilih_Kode.py
#contoh pemakaian if/elif/else
kode = int(input("Masukkan kode agama (1-5) : "))
if (kode==1):
    agama = "Islam"
elif (kode==2):
    agama = "Kristen"
elif (kode==3):
    agama = "Katholik"
elif (kode==4):
    agama = "Hindu"
elif (kode==5):
    agama = "Budha"
else:
    agama = "????? "
    print("Salah kode : harus antara 1 sd 5")

print("Anda memasukkan kode %d untuk agama %s" % (kode,agama))
```

Instruksi match/case (v3.10)

```
kode = int(input("Masukkan angka antara 1 dan 5 : "))
match kode:
    case 1:
        agama = "Islam"
    case 2:
        agama = "Kristen"
    case 3:
        agama = "Katolik"
    case 4:
        agama = "Hindu"
    case 5:
        agama = "Budha"
    case other:
        print("Salah Kode : harus antara 1 sd 5")
        agama='?????'
print("Anda memasukkan kode %d untuk agama %s" % (kode,agama))
```

Contoh Persamaan Kuadrat

```
#akar.py - mencari akar pers ax**2 + bx + c
import math
a = float(input('Masukkan koefisien a : '))
b = float(input('Masukkan koefisien b : '))
c = float(input('Masukkan koefisien c : '))

D = b**2 - 4*a*c
if (D > 0):           #akar riell
    D = math.sqrt(D)
    x1 = (-b + D)/(2*a)
    x2 = (-b - D)/(2*a)
elif (D < 0):          #akar complex
    D = math.sqrt(-D)
    x1 = (-b/2*a) + (D/2*a)*1j
    x2 = (-b/2*a) - (D/2*a)*1j
else:                  #akar kembar
    x1 = -b/(2*a)
    x2 = -b/(2*a)

print('Akar persamaan kuadrat:')
print('x1 = ', x1)
print('x2 = ', x2)
```

Instruksi Perulangan

- instruksi **for** memakai daftar (list)
- instruksi for memakai batas perulangan (range)
- instruksi **while** dengan syarat/kondisi

Contoh for

- Program:

```
# bentuk pertama  
a =['pintu', 'jendela',  
'kusen', 'teralis']  
for x in a:  
    print (x, len(x))
```

```
# bentuk kedua  
for i in range(10):  
    print(i, i*5)
```

- Hasil :

pintu 5
jendela 7
kusen 5
teralis 7

0 0
1 5
2 10
3 15
4 20
5 25
6 30
7 35
8 40
9 45

Contoh for bersusun

```
for n in range(2,21):      # Loop 1 : n bernilai 2 s/d 9
    for x in range(2,n):    # Loop 2 : x bernilai 2 s/d n
        if (n % x) == 0:    # bila n habis dibagi oleh x
            print (n, ' = ', x, '*', n/x)
            break             # Loop 2 x berakhir
    else:                  # keluar loop x, else dari for
        print (n, ' adalah bilangan prima')
```

```
= RESTART: D:\USER\Documents: 11 adalah bilangan prima
2 adalah bilangan prima 12 = 2 * 6.0
3 adalah bilangan prima 13 adalah bilangan prima
4 = 2 * 2.0 14 = 2 * 7.0
5 adalah bilangan prima 15 = 3 * 5.0
6 = 2 * 3.0 16 = 2 * 8.0
7 adalah bilangan prima 17 adalah bilangan prima
8 = 2 * 4.0 18 = 2 * 9.0
9 = 3 * 3.0 19 adalah bilangan prima
10 = 2 * 5.0 20 = 2 * 10.0
```

Perulangan while

- Selain dengan instruksi for, serangkaian instruksi dapat diulang memakai instruksi while, bentuknya:

While (syarat) :

do-something

Instruksi tersebut sepadan dengan kalimat berikut:

**Selama (syarat) ini masih berlaku
maka lakukan: do-something
atau: lakukan do-something hingga syarat tidak
berlaku**

Contoh Program

Disain program untuk menebak angka dengan interface dialog sebagai berikut:

Halo kawan, siapa nama anda?

Eko

Ok, Eko saya memikirkan satu angka antara 1 s/d 20

Coba anda tebak,

10

Angka tersebut lebih kecil dari 10

Tebak lagi,

2

Angka tersebut lebih besar dari 2

Tebak lagi,

4

Bagus, anda telah menebaknya dalam 3 langkah.

```
# program tebakan angka
#tebakan.py
import random          # import pustaka fungsi acak
maxi = 6                # jumlah tebakan maksimum

jumTebakan = 0
print('Hallo kawan, siapa nama anda?')
nama = input()
angka = random.randint(1, 20) # menciptakan angka acak antara 1 s/d 20

print('Ok ' + nama + ' saya memikirkan satu angka antara 1 s/d 20')
while (jumTebakan < maxi):
    print('Coba anda tebak, ')
    tebak = input()
    tebak = int(tebak)
    jumTebakan = jumTebakan + 1
    if (tebak < angka) :
        print('Angka tersebut lebih besar dari ' + str(tebak))
    if (tebak > angka):
        print('Angka tersebut lebih kecil dari ' + str(tebak))
    if (tebak == angka):
        break

if (tebak == angka):
    print ('Bagus, anda telah menebaknya dalam ' + str(jumTebakan) \
          + ' langkah')
if (tebak != angka):
    print ('Maaf, anda tidak berhasil menebak angka ' + str(angka))
```

Contoh Run :

>>>

Hallo kawan, siapa nama anda?

EKO

Ok EKO saya memikirkan satu angka antara 1 s/d 20

Coba anda tebak,

4

Angka tersebut lebih besar dari 4

Coba anda tebak,

20

Angka tersebut lebih kecil dari 20
Coba anda tebak,
12

Angka tersebut lebih kecil dari 12
Coba anda tebak,
8

Angka tersebut lebih kecil dari 8
Coba anda tebak,

6
Angka tersebut lebih kecil dari 6
Coba anda tebak,

5
Bagus, anda telah menebaknya dalam 6 langkah

Definisi Fungsi

- Suatu fungsi dalam python didefinisikan melalui keyword “**def**”, diikuti oleh nama fungsi dan argumen-nya. Sebagai contoh akan dibuat fungsi untuk menghitung:
- $y = x^{**3} + 10*x^{**2}-15*x + 20$
- dimana nilai x dimasukkan lewat keyboard

```
#fungsi_x.py
#definisi fungsi y(x)
def y(x):
    y = x**3 + 10*x**2-15*x + 20
    return y

print("Program menghitung fungs:")
print("y(x) = x**3 + 10*x**2-15*x + 20")
while(True):
    x = int(input("Masukkan nilai x : "))
    yx = y(x) #memanggil y(x) untuk dihitung nilainya
    print("untuk x = %d, y = %d" % (x,yx))
    print()
    jawab = input("Masih mau tanya lagi? (y/n):")
    if (jawab == 'n'):
        print("See you again !")
        break
```

Contoh RUN

```
= RESTART: D:/USER/Documents/myLec
```

```
Program menghitung fungsi:
```

```
y(x) = x**3 + 10*x**2 - 15*x + 20
```

```
Masukkan nilai x : 2
```

```
untuk x = 2, y = 38
```

```
Masih mau tanya lagi? (y/n):y
```

```
Masukkan nilai x : 5
```

```
untuk x = 5, y = 320
```

```
Masih mau tanya lagi? (y/n):n
```

```
See you again !
```

```
def fibo2(n):
    # menampilkan deret fibonacci sampai
    # nilainya <=n.
    # hasil ditampilkan horisontal
    hasil = []
    a, b = 0, 1
    while (b < n):
        hasil.append(b)
        a, b = b, a+b
    return hasil
```

- Run Agar hasilnya bisa ditampilkan, lakukan sebagai berikut.

```
>>> x=fibo2(2000)
>>> x
[1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597]
```

```
>>>
```

Konsep Larik

- Larik atau array adalah struktur data fundamental dalam berbagai bahasa pemrograman, karena struktur data lainnya dapat di-implementasi menggunakan array sebagai struktur dasar.
- Larik berarti suatu deretan tempat/lokasi/sel yang bisa diisi data, setiap lokasi dapat diisi satu datum. Abstraksi dari lokasi memori komputer juga menggunakan larik. Semua bahasa program tingkat tinggi menyediakan struktur larik/array ini.
- Larik 1D disebut vektor, larik 2D disebut matriks

Deklarasi Larik

- Cara mendefinisikan larik/array sangat bergantung pada bahasa program yang digunakan, misalnya: suatu array dengan 6 elemen berupa karakter akan didefinisikan maka pada bahasa:

```
PASCAL :  x : array [0 .. 5] of character; atau,  
          : type larik : array[0 .. 5] of character;  
          var     A : larik;
```

```
C/C++   : char A[6];
```

```
JAVA    : char A[];  A[] = new char[6];
```

- Python pada hakikatnya tidak memerlukan definisi awal (deklarasi) variabel, tetapi langsung ke pemberian nilai variabel, sehingga pernyataan untuk contoh array x yang berisi ‘SAMPLE’ dapat ditulis sebagai:

PYTHON : `x = array('u', ['S','A','M','P','L','E'])`

dimana ‘u’ menyatakan bahwa karakter memakai unicode.

```
primes = array('i', [2,3,5,7,11,13,17,19])
```

mendefinisikan primes sebagai larik bilangan bulat (integer), yaitu angka prima yang < 20.

Pemakaian array ini memiliki keterbatasan, hanya yang memiliki jenis kode tertentu yang dapat dimasukkan ke array.

Code	C Data Type	Typical Number of Bytes
'b'	<code>signed char</code>	1
'B'	<code>unsigned char</code>	1
'u'	<code>Unicode char</code>	2 or 4
'h'	<code>signed short int</code>	2
'H'	<code>unsigned short int</code>	2
'i'	<code>signed int</code>	2 or 4
'I'	<code>unsigned int</code>	2 or 4
'l'	<code>signed long int</code>	4
'L'	<code>unsigned long int</code>	4
'f'	<code>float</code>	4
'd'	<code>float</code>	8

Tabel code tipedata array

selain code diatas tidak dapat dimasukkan dalam array, misalnya nama tipe string tidak dapat dimasukkan.

Cara memakai modul array

```
# 1. import module array
>>> from array import array
# 2. cara definisi array integer
>>> a=array('i',[1,2,3])
# 3 menampilkan array a
>>> a          #atau print(a)
array('i', [1, 2, 3])
>>> a[0]      #akses a[0], index mulai 0
1
>>> print(a) #print isi a
array('i', [1, 2, 3])
```

ADT Modul Array

The constructor is:

`array(typecode [, initializer]) -- create a new array`

| Methods:

- | `append() -- append a new item to the end of the array`
- | `buffer_info() -- return information giving the current memory info`
- | `byteswap() -- byteswap all the items of the array`
- | `count() -- return number of occurrences of an object`
- | `extend() -- extend array by appending multiple elements from an iterable`
- | `fromfile() -- read items from a file object`
- | `fromlist() -- append items from the list`
- | `frombytes() -- append items from the string`
- | `index() -- return index of first occurrence of an object`

ADT Modul Array

```
| insert() -- insert a new item into the array at a provided  
|           position  
| pop() -- remove and return item (default last)  
| remove() -- remove first occurrence of an object  
| reverse() -- reverse the order of the items in the array  
| tofile() -- write all items to a file object  
| tolist() -- return the array converted to an ordinary list  
| tobytes() -- return the array converted to a string
```

```
| Attributes:
```

```
| typecode -- the typecode character used to create the array  
| itemsize -- the length in bytes of one array item
```

Menciptakan Modul array

- Berikut ini disajikan satu model larik (array) biasa, dibangun sebagai suatu program Python, dan memiliki fungsi ADT sebagai berikut:
- **Array(size)** : inisialisasi array berukuran size, A = Array(7), inisialisasi larik dengan 7 lokasi data
- **length()** : panjang atau banyaknya data dalam array, A.length()
- **getitem(index)** : mengambil isi array pada lokasi index, x=A[5]
- **setitem(index, value)** : mengganti isi pada lokasi index menjadi value, misalnya A[4]=10
- **clear(value)** : membersihkan larik dan menetapkan semua nilai menjadi value, A.clear(3), nilai semuanya 3
- **iterator()** : menciptakan objek iterasi untuk larik
- **fromList(L)** : menyalin isi list L kedalam array
- **fromString(Str)** : menyimpan nilai ord dari setiap huruf dalam string Str

inisialisasi class Array

```
class Array :  
    # menciptakan array dengan 'size' elemen.  
    def __init__( self, size ):  
        assert size > 0, "Array size must be > 0"  
        self._size = size  
        # menciptakan array memakai ctypes module.  
        PyArrayType = ctypes.py_object * size  
        self._elements = PyArrayType()  
        # Initialisasi tiap element.  
        self.clear( None )
```

dengan tipe `ctypes.py_object` maka `Array` dapat menerima angka maupun string

```
# Array.py => Implementasi ADT Array ADT memakai struktur ctypes module.
#copyright @Suarga
import ctypes

class Array :
    # menciptakan array dengan 'size' elemen.
    def __init__( self, size ):
        assert size > 0, "Array size must be > 0"
        self._size = size
        # menciptakan array memakai ctypes module.
        PyArrayType = ctypes.py_object * size
        self._elements = PyArrayType()
        # Initialisasi tiap element.
        self.clear( None )

    # memberikan 'size' dari array.
    def __len__( self ):
        return self._size

    # memberikan elemen sesuai index.
    def __getitem__( self, index ):
        assert index >= 0 and index < len(self), "Array subscript out of range"
        return self._elements[ index ]

    # mengganti elemen pada index dengan nilai value.
    def __setitem__( self, index, value ):
        #print('index = ',index)
        assert index >= 0 and index < len(self), "Array subscript out of range"
        self._elements[ index ] = value
```

```
# mengganti semua elemen array menjadi value.
def clear( self, value ):
    for i in range( len(self) ) :
        self._elements[i] = value

# memberikan iterator dari array.
def __iter__( self ):
    return _ArrayIterator( self._elements )

# menyalin isi list L ke array
def fromList(self,L):
    n = self._size
    m = len(L)
    if (m > n):
        self.__init__(m)
    i=0
    for x in L:
        self._elements[i] = x
        i += 1

# menyalin nilai setiap huruf dari string Str ke array
def fromString(self, Str):
    n = self._size
    m = len(Str)
    if (m > n):
        self.__init__(m)

    for i in range(m):
        self._elements[i]=ord(Str[i])
```

```
# An iterator for the Array ADT.
class _ArrayIterator :
    def __init__( self, theArray ) :
        self._arrayRef = theArray
        self._curNdx = 0

    def __iter__( self ) :
        return self

    def __next__( self ) :
        if self._curNdx < len( self._arrayRef ) :
            entry = self._arrayRef[ self._curNdx ]
            self._curNdx += 1
            return entry
        else :
            raise StopIteration
```

Contoh pemakaian ADT array :

```
>>> from Array import Array
>>> larik = Array(7)
>>> larik.clear(5)
>>> for x in larik: print(x)
5
5
5
5
5
5
5
5
>>> larik.__setitem__(3,7) # bisa ditulis sebagai
larik[3]=7
>>> larik.__setitem__(5,3) # bisa ditulis sebagai
larik[5]=3
>>> for x in larik: print(x)
5
5
5
7
5
3
5
```

```
>>> B=Array(5)
>>> L=[1, 4, 6, 3, 2]
>>> B.fromList(L)
>>> for x in B: print(x)
1
4
6
3
2
Lst=["Ali","Badu","Cephi",""
Daud"]
B = Array(4)
B.fromList(Lst)
for i in B: print(i)
Ali
Badu
Cephi
Daud
```

```
>>> D=Array(5)
>>> x='abcdefghijklk'
>>> D.fromStr(x)
>>> for y in D: print(y)
#nilai desimal dari huruf a sd k
97
98
99
100
101
102
103
104
105
106
107
```

Larik memakai python list

```
# Implementasi ADT Larik memakai python list.
# Larik.py == @Suarga
class Larik :
# menciptakan array dengan 'size' elemen.
    def __init__( self, size ):
        # assert size > 0, "Array size must be > 0"
        self._size = size
        # menciptakan array kosong memakai list
        self._elements = []
        for i in range(size):
            self._elements.append(None)

# memberikan 'size' dari array.
    def __len__( self ):
        return self._size

# memberikan elemen sesuai index.
    def __getitem__( self, index ):
        assert index >= 0 and index < len(self), "indeks larik diluar batas"
        return self._elements[ index ]
```

```
# mengganti elemen pada index dengan nilai value.
def __setitem__( self, index, value ):
    #print('index = ',index)
    assert index >= 0 and index < len(self), "indeks tarik diluar batas"
    self._elements[ index ] = value

# mengganti semua elemen array menjadi value.
def clear( self, value ):
    for i in range( len(self) ) :
        self._elements[i] = value

# memberikan iterator dari array.
def __iter__( self ):
    return _ArrayIterator( self._elements )

# menyalin isi list L ke array
def fromList(self,L):
    n = self._size
    m = len(L)
    if (m > n):
        self.__init__(m)
    i=0
    for x in L:
        self._elements[i] = x
        i += 1
```

```
# menyalin nilai setiap huruf dari string Str ke array
def fromString(self, Str):
    n = self._size
    m = len(Str)
    if (m > n):
        self.__init__(m)

    for i in range(m):
        self._elements[i]=ord(Str[i])

# mengisi larik
def isiLarik(self):
    n = self._size
    for i in range(n):
        print('indeks ',i)
        self._elements[i] = int(input(' : '))

# mencari x dalam larik
def cariX(self,x):
    n = self._size
    ketemu = False
    for i in range(n):
        if (x == self._elements[i]):
            print('Ketemu pd index ', i)
            ketemu = True
    if (not ketemu):
        print(x, ' tidak ada dalam larik')
```

```
# menghapus x dari larik
def hapusX(self,x):
    n = self._size
    ketemu = False
    for i in range(n):
        if (x == self._elements[i]):
            print('Ketemu pd index ',i)
            ketemu = True
            print('Sudah dihapus ')
            A[i]=''
    if (not ketemu):
        print(x,' tidak ada dalam array ')

#baca isi larik
def bacaLarik(self):
    r = input('0-Semua-data  1-hanya satu data : ')
    r = int(r)
    if r < 1:
        for x in self._elements:
            print(x)
    else:
        i = int(input('index : '))
        print(self._elements[i])
```

```
# An iterator for the Array ADT.
class _ArrayIterator :
    def __init__( self, theArray ):
        self._arrayRef = theArray
        self._curNdx = 0

    def __iter__( self ):
        return self

    def __next__( self ):
        if self._curNdx < len( self._arrayRef ) :
            entry = self._arrayRef[ self._curNdx ]
            self._curNdx += 1
            return entry
        else :
            raise StopIteration
```

ADT dari Larik.py

Beberapa ADT tambahan adalah:

isiLarik : mengisi data ke larik

cariX : mencari data dalam larik

hapusX : menghapus satu data

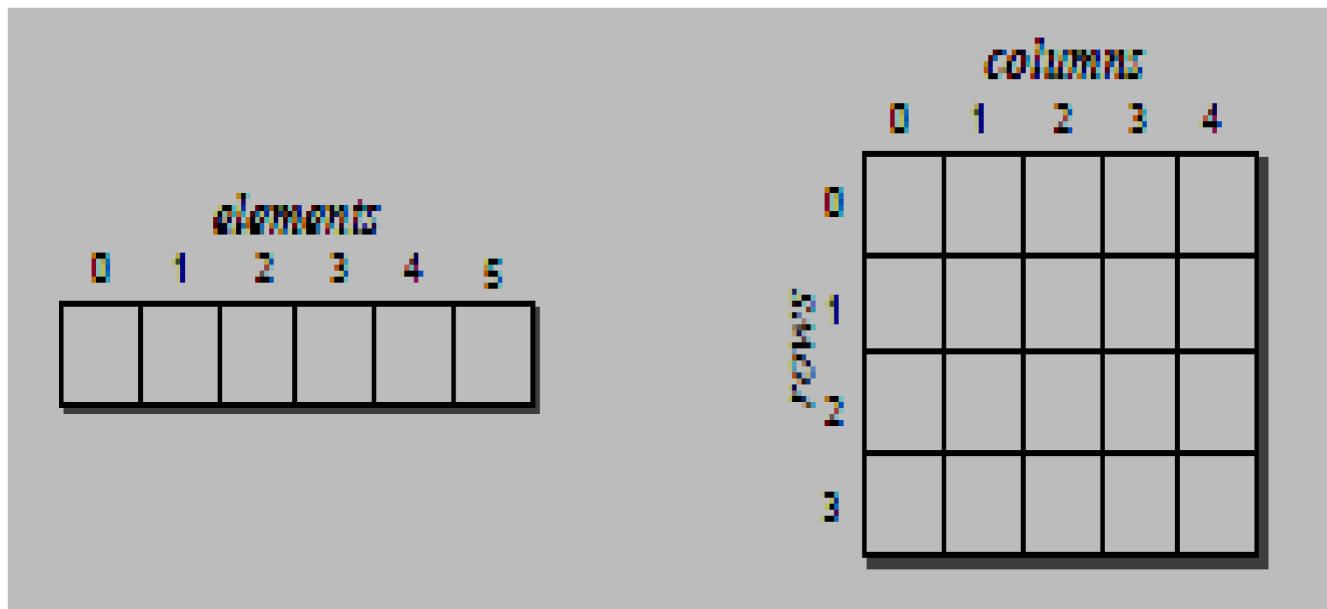
bacaLarik : membaca isi larik

```
>>> A=Larik(5)
>>> A.isiLarik()
indeks 0
: 5
indeks 1
: 3
indeks 2
: 7
indeks 3
: 8
indeks 4
: 2
>>> A.bacaLarik()
0-Semua-data 1-hanya satu data : 0
5
3
7
8
2
```

```
>>> A.bacaLarik()
0-Semua-data 1-hanya satu data : 1
index : 3
8
>>> A.hapusX(1)
1 tidak ada dalam array
>>> A.cariX(4)
4 tidak ada dalam larik
>>> A.cariX(7)
Ketemu pd index 2
>>> A.hapusX(2)
Ketemu pd index 4
Sudah dihapus
>>> A.bacaLarik()
0-Semua-data 1-hanya satu data : 0
5
3
7
8
```

Larik 2D

- Larik 2D adalah larik dengan dua indeks yaitu indeks baris dan indeks kolom.



ADT Larik 2D

- Abstraksi ADT dari larik 2D adalah sebagai berikut.
- **Array2D(nrows, ncols)** : inisialisasi larik 2D dengan nrows baris dan ncols kolom, mula-mula diisi None
- **numRows()** : fungsi untuk memperoleh jumlah baris
- **numCols()** : fungsi untuk memperoleh jumlah kolom
- **clear(value)** : menginisialisasi dengan semua data bernilai value
- **getitem(b, k)** : mengambil data pada baris-b dan kolom-k, boleh ditulis $y = M[b, k]$
- **setitem(b, k, value)** : mengganti nilai data pada baris-b dan kolom-k, boleh ditulis $M[b,k] = value$.
- **dispArr2D()** : menampilkan isi larik 2D

```
>>> B=Larik2D(3,3)          >>> B.dispArr2D()
>>> B.isi2D()
index [0, 0]:
 : 1
index [0, 1]:
 : 2
index [0, 2]:
 : 3
index [1, 0]:
 : 4
index [1, 1]:
 : 5
index [1, 2]:
 : 6
index [2, 0]:
 : 7
index [2, 1]:
 : 8
index [2, 2]:
 : 9
```

```
[1, 2, 3]
[4, 5, 6]
[7, 8, 9]
>>> B.numRows()
3
>>> B.numCols()
3
>>> print(B[1,1])
5
>>> B[1,1]=6
>>> B.dispArr2D()
[1, 2, 3]
[4, 6, 6]
[7, 8, 9]
>>>
```

Inisialisasi Larik2D

```
from Larik import Larik
class Larik2D :
# menciptakan larik 2-D dengan ukuran numRows x numCols.
    def __init__( self, numRows, numCols ):
        # Create a 1-D array to store an array reference
        # for each row.
        self._theRows = Larik( numRows )
        # Create the 1-D arrays for each row of the 2-D
        # array.
        for i in range( numRows ) :
            self._theRows[i] = Larik( numCols )
```

contoh: A = Larik2D(4,5) : 4 baris 5 kolom

Implementasi Larik2D

```
# Implementasi kelas Larik2D, memakai larik dari larik.
# Larik2D.py == @Suarga
from Larik import Larik
class Larik2D :
    # menciptakan larik 2-D dengan ukuran numRows x numCols.
    def __init__( self, numRows, numCols ) :
        # Create a 1-D array to store an array reference for each row.
        self._theRows = Larik( numRows )
        # Create the 1-D arrays for each row of the 2-D array.
        for i in range( numRows ) :
            self._theRows[i] = Larik( numCols )

    # memberikan jumlah baris dari larik 2-D.
    def numRows( self ) :
        return len( self._theRows )

    # memberikan jumlah kolom dari larik 2-D.
    def numCols( self ) :
        return len( self._theRows[0] )
```

```

# memberikan jumlah kolom dari Larik 2-D.
def numCols( self ):
    return len( self._theRows[0] )

# mengisi semua sel Larik 2-D dengan nilai value.
def clear( self, value ):
    for row in range( self.numRows() ):
        self._theRows[row].clear(value)

# menampilkan isi Larik 2-D pada index [i, j]
def __getitem__( self, ndxTuple ):
    assert len(ndxTuple) == 2, "Invalid number of array subscripts."
    row = ndxTuple[0]
    col = ndxTuple[1]
    assert row >= 0 and row < self.numRows() \
        and col >= 0 and col < self.numCols(), \
        "Array subscript out of range."
    the1dArray = self._theRows[row]
    return the1dArray[col]

# mengganti isi sel Larik 2-D pada index [i,j] menjadi value.
def __setitem__( self, ndxTuple, value ):
    assert len(ndxTuple) == 2, "Invalid number of array subscripts."
    row = ndxTuple[0]
    col = ndxTuple[1]
    assert row >= 0 and row < self.numRows() \
        and col >= 0 and col < self.numCols(), \
        "Array subscript out of range."
    the1dArray = self._theRows[row]
    the1dArray[col] = value

```

```
# mengisi Larik 2-D
def isi2D(self):
    for i in range(self.numRows()):
        for j in range(self.numCols()):
            print('index [%d, %d]: ' % (i,j))
            self[i,j] = int(input(' : '))

# menampilkan semua isi Larik 2-D
def dispArr2D( self ):
    for i in range(self.numRows()):
        myRow=[]
        for j in range(self.numCols()):
            myRow.append(self[i,j])
        print(myRow)
```

Matriks

- Matrik adalah larik 2D namun ada beberapa fungsi khusus yang harus ditambahkan khusus untuk ADT dari matrik, yaitu:
- **Matriks(nrows, ncols)** : inisialisasi objek matrik dengan nrows baris dan ncols kolom
- **numRows()** : memberikan jumlah baris dari matrik
- **numCols()** : memberikan jumlah kolom dari matrik
- **getitem(row, col)** : memberikan data pada baris row dan kolom col
- **setitem(row, col)** : mengganti nilai data pada baris row dan kolom col
- **scaleBy(scalar)** : mengalikan setiap data dengan nilai scalar
- **transpose()** : melakukan transpose dari matrik
- **add(rhsMat)** : menjumlahkan matrik ini dengan matrik rhsMat
- **subtract (rhsMat)** : mengurangkan matrik ini dengan matrik rhsMat
- **multiply(rhsMat)**: mengalikan matrik ini dengan matrik rhsMat

fungsi: scaleBy

```
# Scales the matrix by the given scalar.  
def scaleBy( self, scalar ):  
    for r in range( self numRows() ) :  
        for c in range( self numCols() ) :  
            self[ r, c ] *= scalar
```

#mengalikan elemen matrik dengan bil scalar

fungsi: transpose()

```
def transpose( self ):  
    bar = self numRows()  
    kol = self numCols()  
    temp = Matrix(kol, bar)  
    for i in range(bar):  
        for j in range(kol):  
            temp[j,i] = self[i,j]  
    return temp
```

#melakukan transpose baris <==> kolom

fungsi : add()

```
def __add__( self, rhsMatrix ):  
    assert rhsMatrix numRows() == self numRows() and \  
           rhsMatrix numCols() == self numCols(), \  
           "Matrix sizes not compatible for the add \  
           operation."  
    # Create the new matrix.  
    newMatrix = Matrix( self numRows(), self numCols() )  
    # Add the corresponding elements in the two \  
    # matrices.  
    for r in range( self numRows() ) :  
        for c in range( self numCols() ) :  
            newMatrix[ r, c ] = self[ r, c ] + \  
                               rhsMatrix[ r, c ]  
    return newMatrix
```

fungsi subtract: __sub__()

```
def __sub__( self, rhsMatrix ):  
    assert rhsMatrix numRows() == self numRows() and \  
           rhsMatrix numCols() == self numCols(), \  
           "Matrix sizes not compatible for the add \  
           operation."  
    # Create the new matrix.  
    newMatrix = Matrix( self numRows(), self numCols() )  
    # Add the corresponding elements in the two  
    # matrices.  
    for r in range( self numRows() ) :  
        for c in range( self numCols() ) :  
            newMatrix[ r, c ] = self[ r, c ] - \  
                               rhsMatrix[ r, c ]  
    return newMatrix
```

fungsi multiply: __mul__()

```
def __mul__( self, rhsMatrix ):
    assert rhsMatrix numRows() == self numCols(), \
        "Matrix sizes not compatible for the \
        multiplication."
    # create the new matrix
    newMatrix = Matrix( self numRows(), \
                        rhsMatrix numCols() )
    # multiply the matrices
    for r in range( self numRows() ):
        for c in range( rhsMatrix numCols() ):
            newMatrix[r, c] = 0
            for k in range( rhsMatrix numRows() ):
                newMatrix[r, c] += self[r,k] * \
                    rhsMatrix[k, c]
    return newMatrix
```

Implementasi Matriks

```
# Implementasi kelas Matriks memakai Larik2D.
# Matriks.py == @Suarga
from Larik2D import Larik2D

class Matriks :
    # Creates a matrix of size numRows x numCols initialized to 0.
    def __init__( self, numRows, numCols ):
        self._theGrid = Larik2D( numRows, numCols )
        self._theGrid.clear( 0 )

    # Returns the number of rows in the matrix.
    def numRows( self ):
        return self._theGrid.numRows()

    # Returns the number of columns in the matrix.
    def numCols( self ):
        return self._theGrid.numCols()

    # Returns the value of element (i, j): x[i,j]
    def __getitem__( self, ndxTuple ):
        return self._theGrid[ ndxTuple[0], ndxTuple[1] ]
```

```

# Sets the value of element (i,j) to the value s: x[i,j] = s
def __setitem__( self, ndxTuple, scalar ):
    self._theGrid[ ndxTuple[0], ndxTuple[1] ] = scalar

# Scales the matrix by the given scalar.
def scaleBy( self, scalar ):
    for r in range( self.numRows() ) :
        for c in range( self.numCols() ) :
            self[ r, c ] *= scalar

# Creates and returns a new matrix that is the transpose of this matrix.
# added by Suarga
def transpose( self ):
    bar = self.numRows()
    kol = self.numCols()
    temp = Matriks(kol, bar)
    for i in range(bar):
        for j in range(kol):
            temp[j,i] = self[i,j]
    return temp

```

```

# Creates and returns a new matrix that results from matrix addition.
def __add__( self, rhsMatrix ):
    assert rhsMatrix numRows() == self numRows() and \
        rhsMatrix numCols() == self numCols(), \
        "Matrix sizes not compatible for the add operation."
    # Create the new matrix.
    newMatriks = Matriks( self numRows(), self numCols() )
    # Add the corresponding elements in the two matrices.
    for r in range( self numRows() ) :
        for c in range( self numCols() ) :
            newMatriks[ r, c ] = self[ r, c ] + rhsMatrix[ r, c ]
    return newMatriks

# Creates and returns a new matrix that results from matrix subtraction.
# completed by Suarga
def __sub__( self, rhsMatrix ):
    assert rhsMatrix numRows() == self numRows() and \
        rhsMatrix numCols() == self numCols(), \
        "Matrix sizes not compatible for the add operation."
    # Create the new matrix.
    newMatrix = Matriks( self numRows(), self numCols() )
    # Add the corresponding elements in the two matrices.
    for r in range( self numRows() ) :
        for c in range( self numCols() ) :
            newMatrix[ r, c ] = self[ r, c ] - rhsMatrix[ r, c ]
    return newMatrix

```

```

# Creates and returns a new matrix resulting from matrix multiplication.
# added by Suarga
def __mul__( self, rhsMatrix ):
    assert rhsMatrix.numRows() == self.numCols(), \
        "Matrix sizes not compatible for the multiplication."
    # create the new matrix
    newMatrix = Matriks( self.numRows(), rhsMatrix.numCols() )
    # multiply the matrices
    for r in range( self.numRows() ):
        for c in range( rhsMatrix.numCols() ):
            newMatrix[r, c] = 0
            for k in range( rhsMatrix.numRows() ):
                newMatrix[r, c] += self[r,k] * rhsMatrix[k, c]
    return newMatrix

# display matrix
def dispMatriks( self ):
    for i in range(self.numRows()):
        myRow=[]
        for j in range(self.numCols()):
            myRow.append(self[i,j])
        print(myRow)

# isi matriks
def isiMatriks( self ):
    for i in range(self.numRows()):
        for j in range(self.numCols()):
            print('index [%d, %d]: ' % (i,j))
            self[i,j] = int(input(' : '))

```

```
>>> A=Matriks(3,3)          >>> B[0,0]=9  
>>> A[0,0]=1                >>> B[0,1]=8  
>>> A[0,1]=2                >>> B[0,2]=7  
>>> A[0,2]=3                >>> B[1,0]=6  
>>> A[1,0]=4                >>> B[1,1]=5  
>>> A[1,1]=5                >>> B[1,2]=4  
>>> A[1,2]=6                >>> B[2,0]=3  
>>> A[2,0]=7                >>> B[2,1]=2  
>>> A[2,1]=8                >>> B[2,2]=1  
>>> A[2,2]=9                >>> C=A+B  
>>> A.dispMatriks()        >>> C.dispMatriks()  
[1, 2, 3]                  [10, 10, 10]  
[4, 5, 6]                  [10, 10, 10]  
[7, 8, 9]                  [10, 10, 10]  
>>> B=Matriks(3,3)
```

```
>>> D=B-A  
>>> D.dispMatriks()  
[8, 6, 4]  
[2, 0, -2]  
[-4, -6, -8]  
>>> X=A.transpose()  
>>> X.dispMatriks()  
[1, 4, 7]  
[2, 5, 8]  
[3, 6, 9]  
>>> E=A*B  
>>> E.dispMatriks()  
[30, 24, 18]  
[84, 69, 54]  
[138, 114, 90]  
>>>
```

SELESAI