LAPORAN PRAKTIKUM ALGORITMA STRUKTUR DATA MODUL 3 "COLLECTIONS, ARRAYS, AND LINKED STRUCTURES"

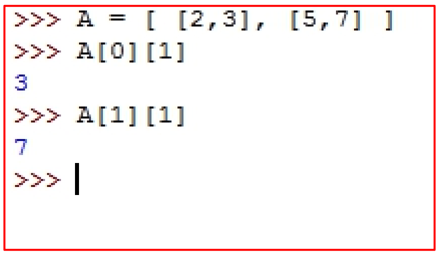
Oleh:

NAMA	**: Daffa Putra Alwansyah**
NIM	**: L200190031**
KELAS	**: B**
PRODI	**: INFORMATIKA**

Fakultas Komunikasi dan Informatika Universitas Muhammadiyah Surakarta

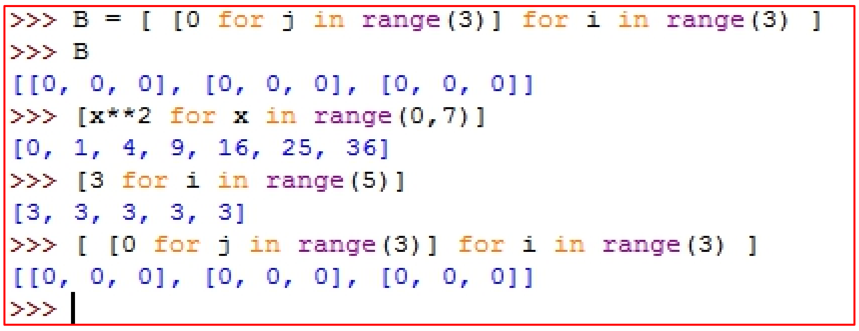
Latihan

Latihan3.1



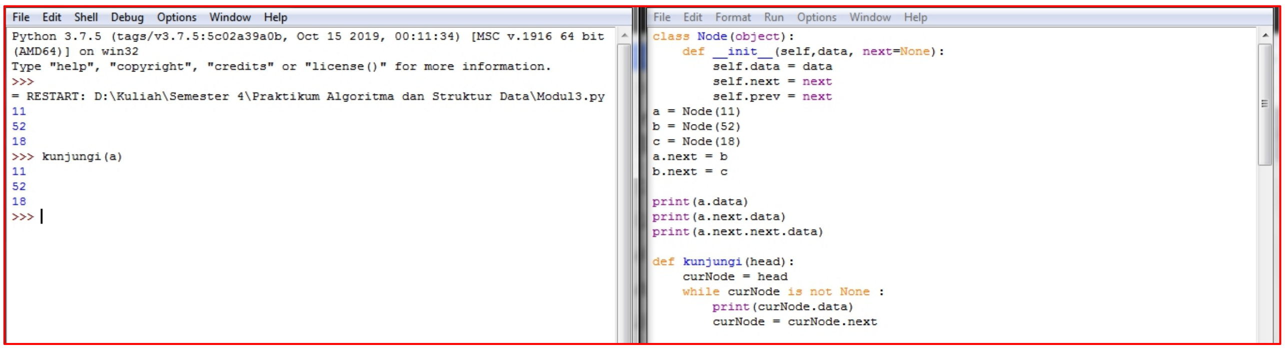
```
>>> A = [ [2,3], [5,7] ]
>>> A[0][1]
3
>>> A[1][1]
7
>>> 
```

Latihan3.2



```
>>> B = [ [0 for j in range(3)] for i in range(3) ]
>>> B
[[0, 0, 0], [0, 0, 0], [0, 0, 0]]
>>> [x**2 for x in range(0,7)]
[0, 1, 4, 9, 16, 25, 36]
>>> [3 for i in range(5)]
[3, 3, 3, 3, 3]
>>> [ [0 for j in range(3)] for i in range(3) ]
[[0, 0, 0], [0, 0, 0], [0, 0, 0]]
>>> 
```

Latihan3.3


File Edit Shell Debug Options Window Help
File Edit Format Run Options Window Help

```
Python 3.7.5 (tags/v3.7.5:5c02a39a0b, Oct 15 2019, 00:11:34) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 
= RESTART: D:\Kuliah\Semester 4\Praktikum Algoritma dan Struktur Data\Modul3.py
11
52
18
>>> kunjungi(a)
11
52
18
>>> 
```

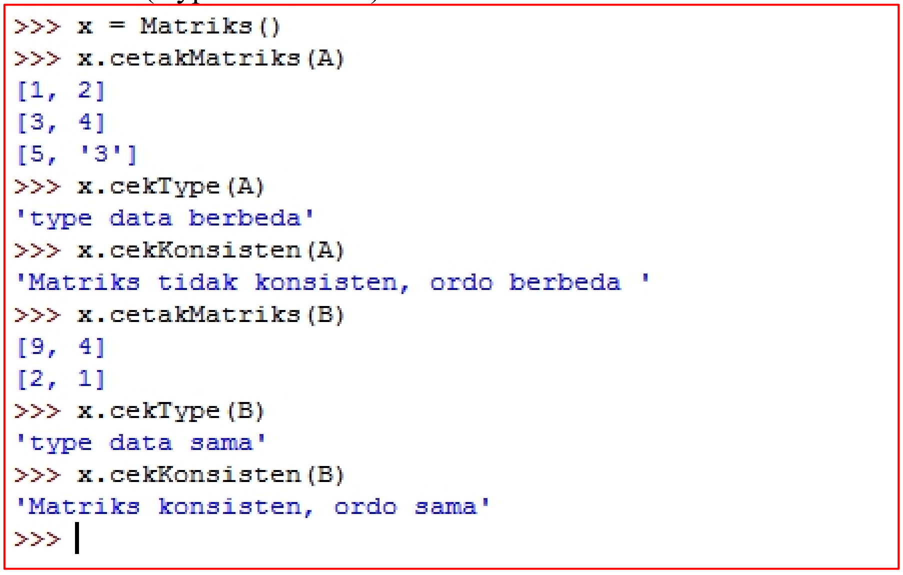
```python
class Node(object):
    def __init__(self,data, next=None):
        self.data = data
        self.next = next
        self.prev = next
a = Node(11)
b = Node(52)
c = Node(18)
a.next = b
b.next = c

print(a.data)
print(a.next.data)
print(a.next.next.data)

def kunjungi(head):
    curNode = head
    while curNode is not None :
        print(curNode.data)
        curNode = curNode.next
```

3.4 Soal-soal untuk Mahasiswa

```
A = [[1,2],[3,4],[5,'3']]
B = [[9,4],[2,1]]
C = [[8,5],[1,3]]
```

#Nomor 1A

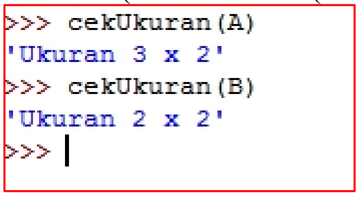
```
class Matriks (object):
    def cetakMatriks(self, matriks):
        for i in matriks:
            print(i)
    def cekKonsisten(self, matriks):
        if len(matriks[0]) == len(matriks) :
            return ("Matriks konsisten, ordo sama")
        else:
            return ("Matriks tidak konsisten, ordo berbeda ")
    def cekType(self, matriks):
        for i in matriks:
            for x in i:
                if type(x) != int:
                    return("type data berbeda")
        return("type data sama")
```



```
>>> x = Matriks()
>>> x.cetakMatriks(A)
[1, 2]
[3, 4]
[5, '3']
>>> x.cekType(A)
'type data berbeda'
>>> x.cekKonsisten(A)
'Matriks tidak konsisten, ordo berbeda '
>>> x.cetakMatriks(B)
[9, 4]
[2, 1]
>>> x.cekType(B)
'type data sama'
>>> x.cekKonsisten(B)
'Matriks konsisten, ordo sama'
>>> 
```

#Nomor 1B

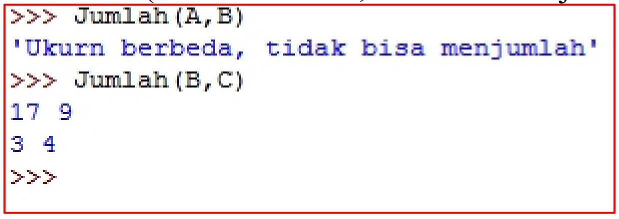
```
def cekUkuran(matriks):
    return ("Ukuran "+str(len(matriks))+" x "+str(len(matriks[0])))
```



```
>>> cekUkuran(A)
'Ukuran 3 x 2'
>>> cekUkuran(B)
'Ukuran 2 x 2'
>>> 
```

#Nomor 1C

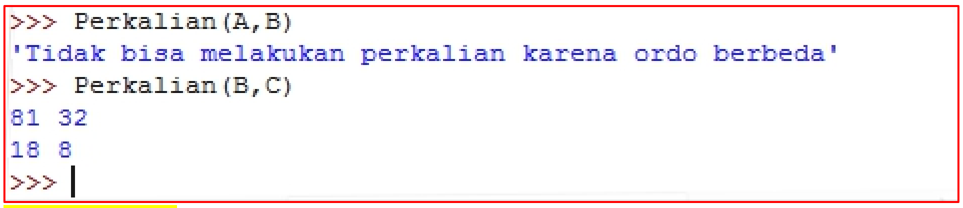
```
def Jumlah(m1, m2):
    if cekUkuran(m1) == cekUkuran(m2):
        for x in range(0, len(m1)):
            for y in range(0, len(m1[0])):
                print (m1[x][y] + m2[x][y],end=' '),
            print()
    else:
        return("Ukurn berbeda, tidak bisa menjumlah")
```



```
>>> Jumlah(A,B)
'Ukurn berbeda, tidak bisa menjumlah'
>>> Jumlah(B,C)
17 9
3 4
>>>
```

##Nomor 1D

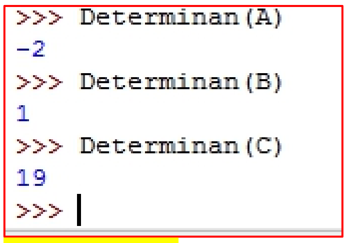
```
i =[]
def Perkalian(m1,m2):
    if cekUkuran(m1) == cekUkuran(m2):
        for  x in range(0, len(m1)):
            row = []
            for y in range (0, len(m1[0])):
                total = 0
                for z in range (0, len(m1)):
                    total = total + (m1[x][y]*m2[z][y])
                row.append(total)
            i.append(row)

        for x in range (0, len(i)):
            for y in range(0, len(i[0])):
                print (i[x][y], end=' '),
            print()
    else:
        return("Tidak bisa melakukan perkalian karena ordo berbeda")
```



```
>>> Perkalian(A,B)
'Tidak bisa melakukan perkalian karena ordo berbeda'
>>> Perkalian(B,C)
81 32
18 8
>>> 
```

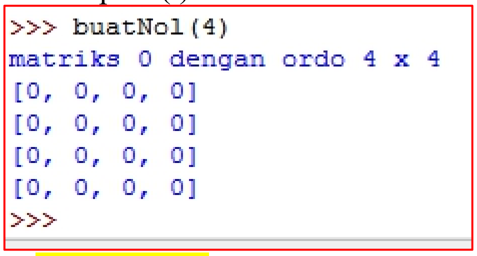
##Nomor 1E

```
def Determinan(x):
    for i in range(2):
        if i == 0:
            ad = x[i][i]*x[i+1][i+1]
        elif i == 1:
            bc = x[i-1][i]*x[i][i-1]
    return ad-bc
```


```
>>> Determinan(A)
-2
>>> Determinan(B)
1
>>> Determinan(C)
19
>>> 
```

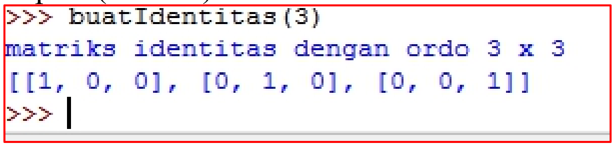
##Nomor 2A

```
def buatNol(n, m=None):

    if (m == None):
        m = n
    print ("matriks 0 dengan ordo "+str(n)+" x "+str(m))
    x = ([[0 for j in range(m)] for i in range(n)])
    for i in x:
        print(i)
```


```
>>> buatNol(4)
matriks 0 dengan ordo 4 x 4
[0, 0, 0, 0]
[0, 0, 0, 0]
[0, 0, 0, 0]
[0, 0, 0, 0]
>>> 
```

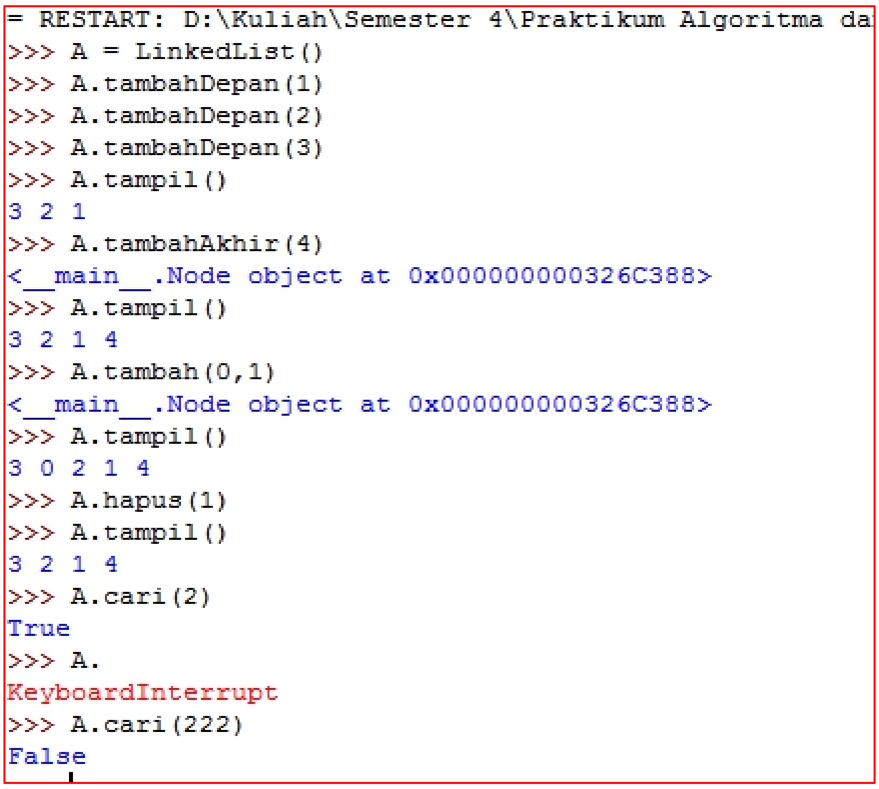
##Nomor 2B

```
def buatIdentitas(m):
    print("matriks identitas dengan ordo "+str(m)+" x "+str(m))
    matriks = [[1 if j == i else 0 for j in range(m)] for i in range(m)]
    print(matriks)
```


```
>>> buatIdentitas(3)
matriks identitas dengan ordo 3 x 3
[[1, 0, 0], [0, 1, 0], [0, 0, 1]]
>>> 
```

###Nomor 3

```
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None
class LinkedList:
    def __init__(self):
        self.head = None
    #menammbah suatu simpul di awal
    def tambahDepan(self, new_data):
        new_node = Node(new_data)
        new_node.next = self.head
        self.head = new_node
    #menambah suatu simpul di akhir
    def tambahAkhir(self, data):
        if (self.head == None):
            self.head = Node(data)
    else:
        current = self.head
        while (current.next != None):
            current = current.next
        current.next = Node(data)
    return self.head
#menyisipkan suatu simpul di mana saja
def tambah(self,data,posisi):
    node = Node(data)
    if not self.head:
        self.head = node
    elif posisi == 0:
        node.next = self.head
        self.head = node
    else:
        prev = None
        current = self.head
        current_posisi = 0
        while (current_posisi < posisi) and current.next:
            prev = current
            current = current.next
            current_posisi += 1
        prev.next = node
        node.next = current
    return self.head
#menghapus suatu simpul di awal, di akhir, atau di mana saja
def hapus(self,posisi):
    if self.head == None:
        return
    temp = self.head
    if posisi == 0:
        self.head = temp.next
        temp = None
        return
    for i in range(posisi - 1):
        temp = temp.next
        if temp is None:
            break
    if temp is None:
        return
    if temp.next is None:
        return
    next = temp.next.next
    temp.next = None
    temp.next = next
##mencari data yang isinya tertentu
def cari(self,x):
    current = self.head
    while current != None:
        if current.data == x:
                return True
            current = current.next
        return False
    def tampil(self):
        current = self.head
        while current is not None:
            print(current.data, end = ' ')
            current = current.next
```



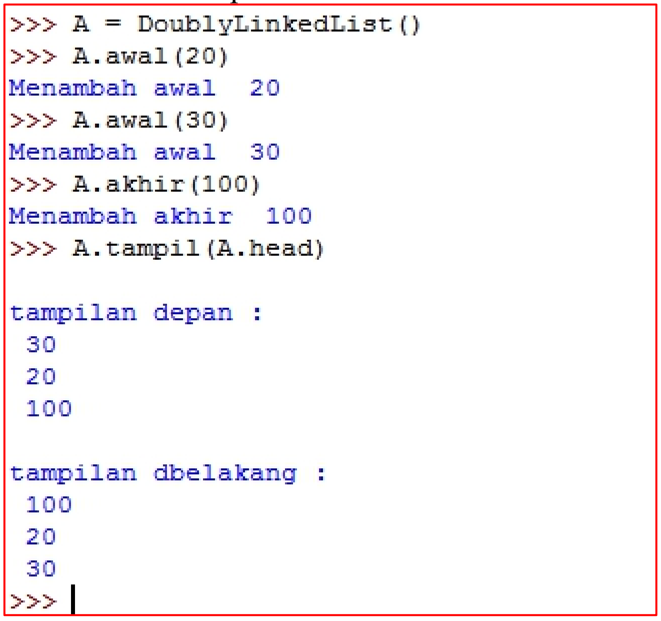
```
= RESTART: D:\Kuliah\Semester 4\Praktikum Algoritma da
>>> A = LinkedList()
>>> A.tambahDepan(1)
>>> A.tambahDepan(2)
>>> A.tambahDepan(3)
>>> A.tampil()
3 2 1
>>> A.tambahAkhir(4)
<__main__.Node object at 0x000000000326C388>
>>> A.tampil()
3 2 1 4
>>> A.tambah(0,1)
<__main__.Node object at 0x000000000326C388>
>>> A.tampil()
3 0 2 1 4
>>> A.hapus(1)
>>> A.tampil()
3 2 1 4
>>> A.cari(2)
True
>>> A.
KeyboardInterrupt
>>> A.cari(222)
False
```

#Nomor 4

```
class Node:
    def __init__(self, data):
        self.data = data
        self.prev = None
class DoublyLinkedList:
    def __init__(self):
        self.head = None
    #menambah suatu simpul di awal
    def awal(self, new_data):
        print("Menambah awal ",new_data)
        new_node = Node(new_data)
        new_node.next = self.head
        if self.head is not None:
            self.head.prev = new_node
        self.head = new_node
    #menambah suatu simpul di akhir
    def akhir(self,new_data):
        print("Menambah akhir ",new_data)
        new_node = Node(new_data)
        new_node.next = None
        if self.head is None:
            new_node.prev = None
            self.head = new_node
            return
        last = self.head
        while(last.next is not None):
            last = last.next
        last.next = new_node
        new_node.prev = last
        return
```

#mengunjungi dan mencetak data tiap simpul dari depan dan dari belakang

```
    def tampil(self,node):
        print("\ntampilan depan :")
        while (node is not None):
            print (" %d "%(node.data))
            last = node
            node = node.next
        print ("\ntampilan dbelakang :")
        while (last is not None):
            print (" %d "%(last.data))
            last = last.prev
```



```
>>> A = DoublyLinkedList()
>>> A.awal(20)
Menambah awal  20
>>> A.awal(30)
Menambah awal  30
>>> A.akhir(100)
Menambah akhir  100
>>> A.tampil(A.head)

tampilan depan :
 30
 20
 100

tampilan dbelakang :
 100
 20
 30
>>> 
```