Nama : Annas Fagiat

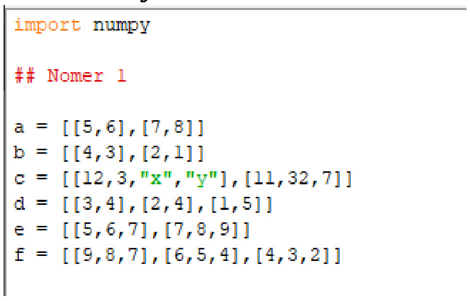
Kelas : D

Nim : L200170163

# LAPORAN PRAKTIKUM ALGORTIMA DAN STRUKTUR DATA
# MODUL KE 3

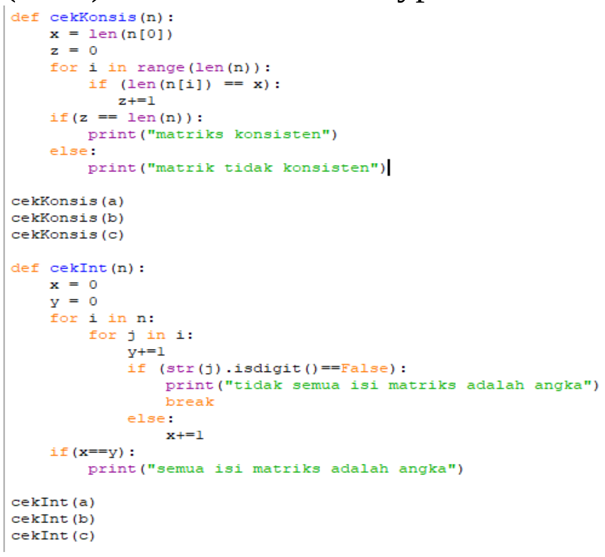
1. Matriks Array 2D

   List Array :



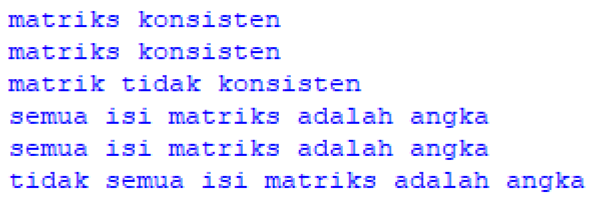
```
import numpy

## Nomer 1

a = [[5,6],[7,8]]
b = [[4,3],[2,1]]
c = [[12,3,"x","y"],[11,32,7]]
d = [[3,4],[2,4],[1,5]]
e = [[5,6,7],[7,8,9]]
f = [[9,8,7],[6,5,4],[4,3,2]]
```

   a. ( Def )Cek konsisten dan type data matriks



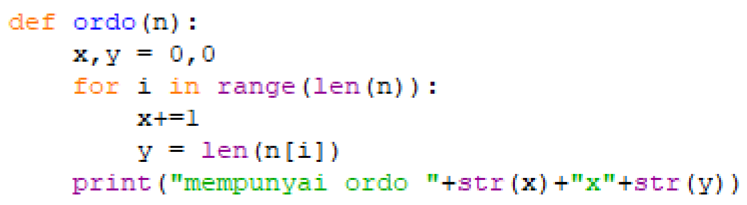
```
def cekKonsis(n):
    x = len(n[0])
    z = 0
    for i in range(len(n)):
        if (len(n[i]) == x):
            z+=1
    if(z == len(n)):
        print("matriks konsisten")
    else:
        print("matrik tidak konsisten")

cekKonsis(a)
cekKonsis(b)
cekKonsis(c)

def cekInt(n):
    x = 0
    y = 0
    for i in n:
        for j in i:
            y+=1
            if (str(j).isdigit()==False):
                print("tidak semua isi matriks adalah angka")
                break
            else:
                x+=1
    if(x==y):
        print("semua isi matriks adalah angka")

cekInt(a)
cekInt(b)
cekInt(c)
```

   Hasil run :

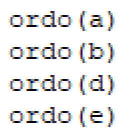


```
matriks konsisten
matriks konsisten
matrik tidak konsisten
semua isi matriks adalah angka
semua isi matriks adalah angka
tidak semua isi matriks adalah angka
```

b. (Def) Mengambil Ukuran Matriks

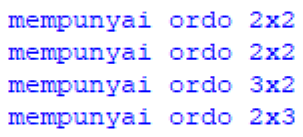


```
def ordo(n):
    x,y = 0,0
    for i in range(len(n)):
        x+=1
        y = len(n[i])
    print("mempunyai ordo "+str(x)+"x"+str(y))
```



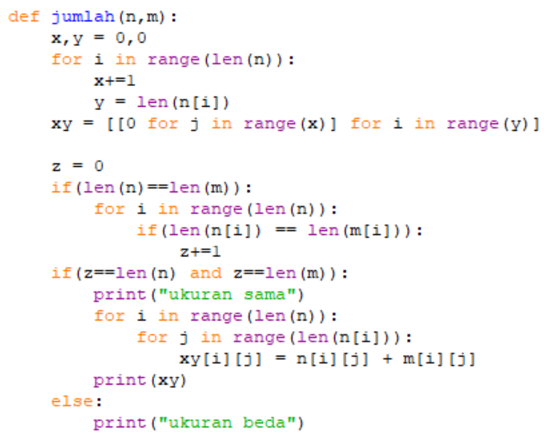
```
ordo(a)
ordo(b)
ordo(d)
ordo(e)
```

Hasil Run :

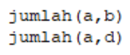


```
mempunyai ordo 2x2
mempunyai ordo 2x2
mempunyai ordo 3x2
mempunyai ordo 2x3
```

c. (Def) Menjumlahkan 2 Matriks

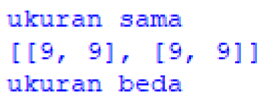


```
def jumlah(n,m):
    x,y = 0,0
    for i in range(len(n)):
        x+=1
        y = len(n[i])
    xy = [[0 for j in range(x)] for i in range(y)]

    z = 0
    if(len(n)==len(m)):
        for i in range(len(n)):
            if(len(n[i]) == len(m[i])):
                z+=1
    if(z==len(n) and z==len(m)):
        print("ukuran sama")
        for i in range(len(n)):
            for j in range(len(n[i])):
                xy[i][j] = n[i][j] + m[i][j]
        print(xy)
    else:
        print("ukuran beda")
```



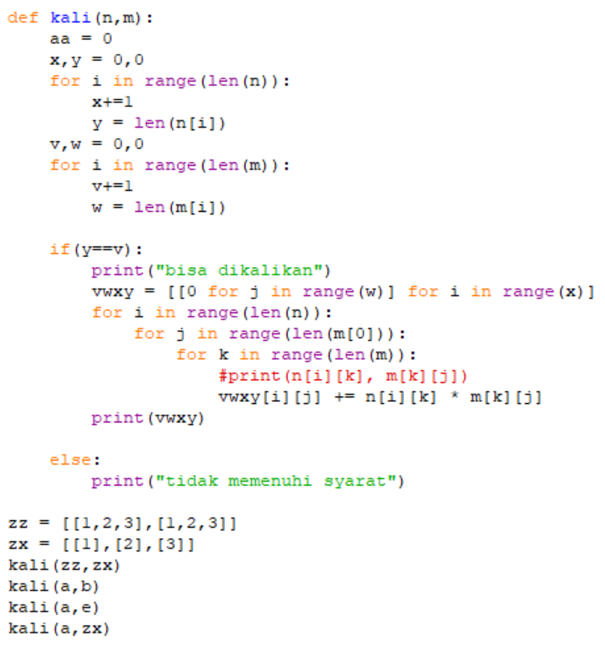
```
jumlah(a,b)
jumlah(a,d)
```

Hasil Run :



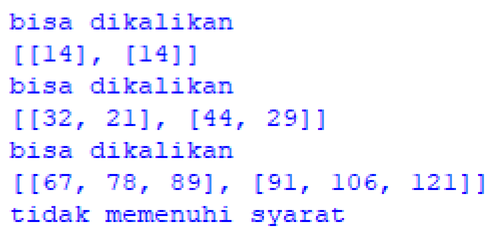
```
ukuran sama
[[9, 9], [9, 9]]
ukuran beda
```

d. (Def) Mengalikan 2 Matriks



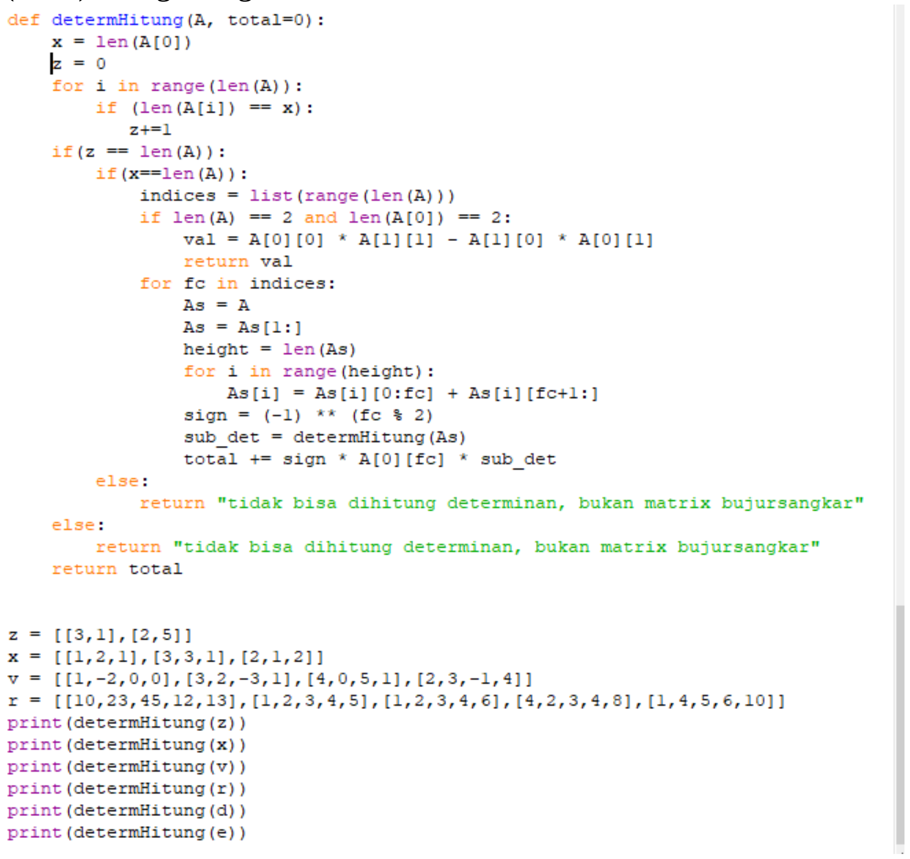
```python
def kali(n,m):
    aa = 0
    x,y = 0,0
    for i in range(len(n)):
        x+=1
        y = len(n[i])
    v,w = 0,0
    for i in range(len(m)):
        v+=1
        w = len(m[i])

    if(y==v):
        print("bisa dikalikan")
        vwxy = [[0 for j in range(w)] for i in range(x)]
        for i in range(len(n)):
            for j in range(len(m[0])):
                for k in range(len(m)):
                    #print(n[i][k], m[k][j])
                    vwxy[i][j] += n[i][k] * m[k][j]
        print(vwxy)

    else:
        print("tidak memenuhi syarat")

zz = [[1,2,3],[1,2,3]]
zx = [[1],[2],[3]]
kali(zz,zx)
kali(a,b)
kali(a,e)
kali(a,zx)
```

Hasil Run :



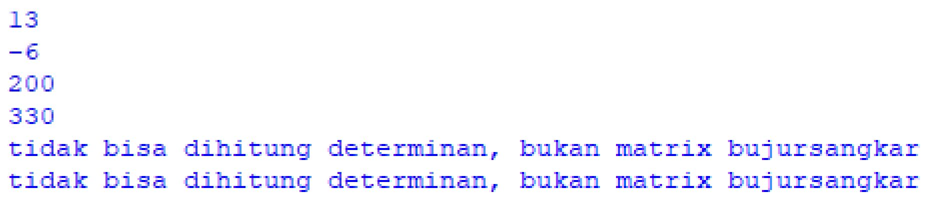
```
bisa dikalikan
[[14], [14]]
bisa dikalikan
[[32, 21], [44, 29]]
bisa dikalikan
[[67, 78, 89], [91, 106, 121]]
tidak memenuhi syarat
```

e. ( Def ) Menghitung Determinan Matriks



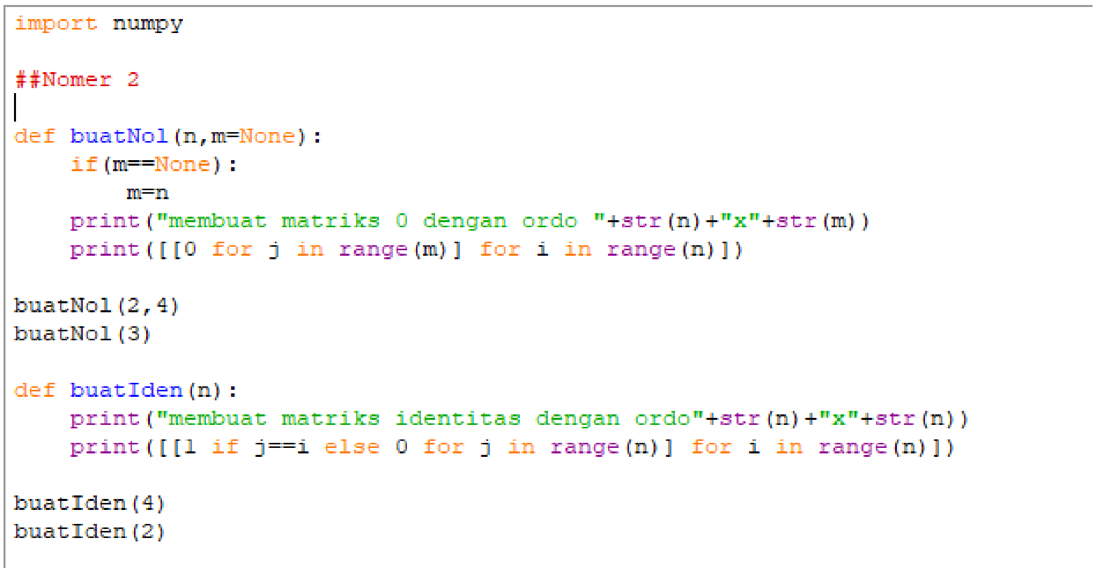
```python
def determHitung(A, total=0):
    x = len(A[0])
    z = 0
    for i in range(len(A)):
        if (len(A[i]) == x):
            z+=1
    if(z == len(A)):
        if(x==len(A)):
            indices = list(range(len(A)))
            if len(A) == 2 and len(A[0]) == 2:
                val = A[0][0] * A[1][1] - A[1][0] * A[0][1]
                return val
            for fc in indices:
                As = A
                As = As[1:]
                height = len(As)
                for i in range(height):
                    As[i] = As[i][0:fc] + As[i][fc+1:]
                sign = (-1) ** (fc % 2)
                sub_det = determHitung(As)
                total += sign * A[0][fc] * sub_det
        else:
            return "tidak bisa dihitung determinan, bukan matrix bujursangkar"
    else:
        return "tidak bisa dihitung determinan, bukan matrix bujursangkar"
    return total


z = [[3,1],[2,5]]
x = [[1,2,1],[3,3,1],[2,1,2]]
v = [[1,-2,0,0],[3,2,-3,1],[4,0,5,1],[2,3,-1,4]]
r = [[10,23,45,12,13],[1,2,3,4,5],[1,2,3,4,6],[4,2,3,4,8],[1,4,5,6,10]]
print(determHitung(z))
print(determHitung(x))
print(determHitung(v))
print(determHitung(r))
print(determHitung(d))
print(determHitung(e))
```

Hasil Run :



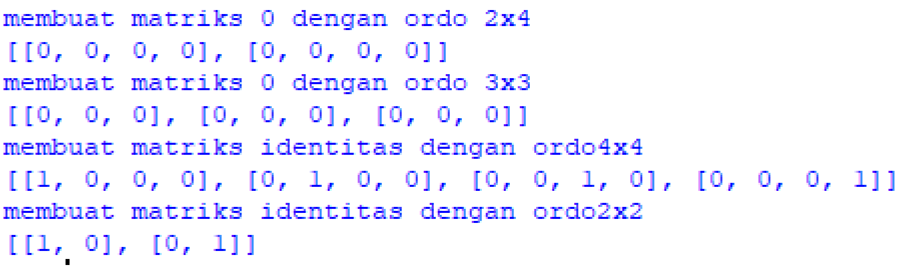
```
13
-6
200
330
tidak bisa dihitung determinan, bukan matrix bujursangkar
tidak bisa dihitung determinan, bukan matrix bujursangkar
```

2. List Comprehension
   - Membuat Matriks 0 dan Matriks Identitas



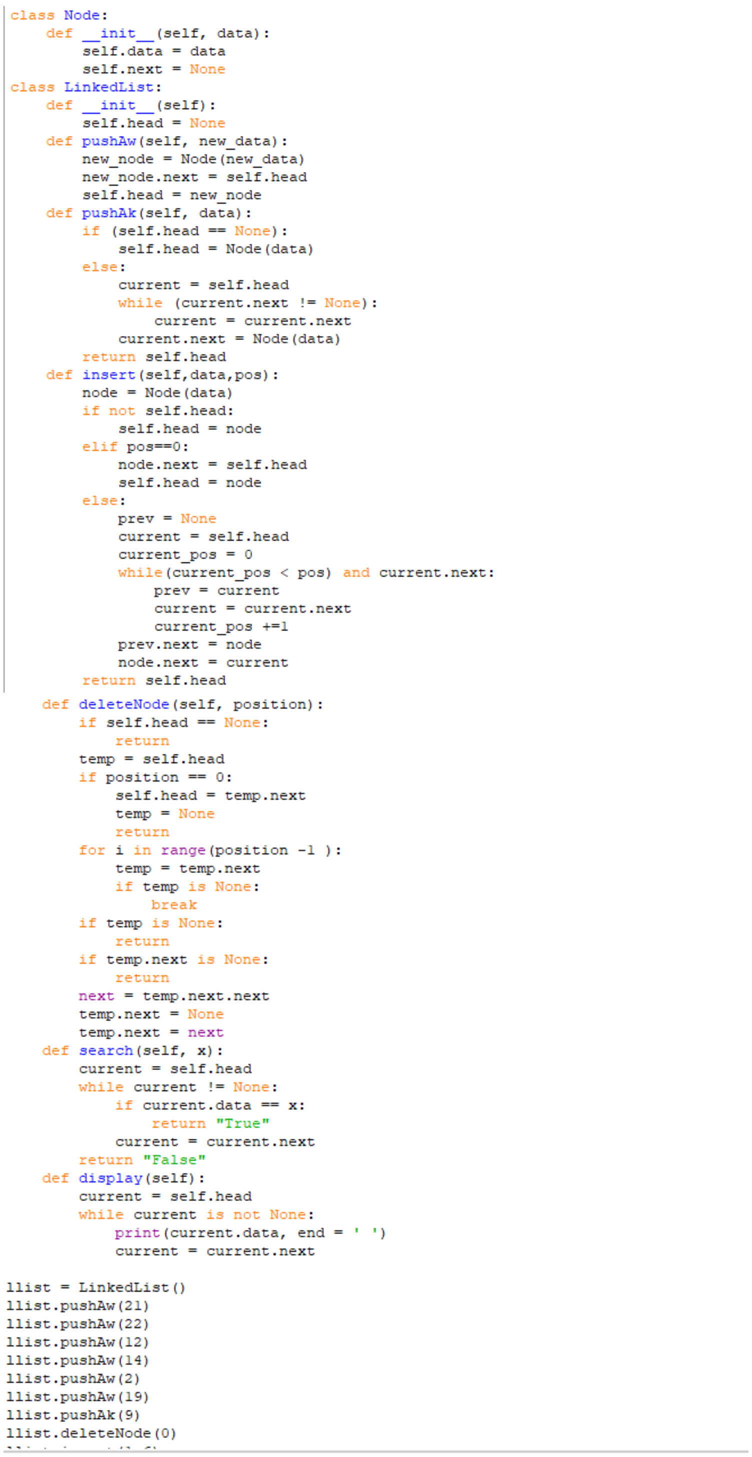
```python
import numpy

##Nomer 2

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

buatNol(2,4)
buatNol(3)

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

buatIden(4)
buatIden(2)
```

Hasil Run :



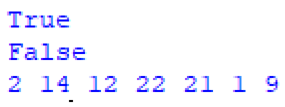
```
membuat matriks 0 dengan ordo 2x4
[[0, 0, 0, 0], [0, 0, 0, 0]]
membuat matriks 0 dengan ordo 3x3
[[0, 0, 0], [0, 0, 0], [0, 0, 0]]
membuat matriks identitas dengan ordo4x4
[[1, 0, 0, 0], [0, 1, 0, 0], [0, 0, 1, 0], [0, 0, 0, 1]]
membuat matriks identitas dengan ordo2x2
[[1, 0], [0, 1]]
```

3. Linked List
   - Mencari data tertentu
   - Menambah simpul di awal dan akhir
   - Menyisipkan simpul di posisi tertentu
   - Menghapus simpul di posisi tertentu



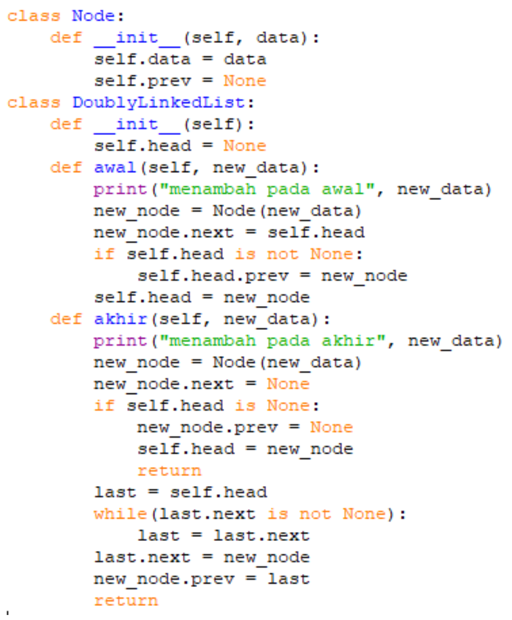
```python
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None
class LinkedList:
    def __init__(self):
        self.head = None
    def pushAw(self, new_data):
        new_node = Node(new_data)
        new_node.next = self.head
        self.head = new_node
    def pushAk(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
    def insert(self,data,pos):
        node = Node(data)
        if not self.head:
            self.head = node
        elif pos==0:
            node.next = self.head
            self.head = node
        else:
            prev = None
            current = self.head
            current_pos = 0
            while(current_pos < pos) and current.next:
                prev = current
                current = current.next
                current_pos +=1
            prev.next = node
            node.next = current
        return self.head
    def deleteNode(self, position):
        if self.head == None:
            return
        temp = self.head
        if position == 0:
            self.head = temp.next
            temp = None
            return
        for i in range(position -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
    def search(self, x):
        current = self.head
        while current != None:
            if current.data == x:
                return "True"
            current = current.next
        return "False"
    def display(self):
        current = self.head
        while current is not None:
            print(current.data, end = ' ')
            current = current.next

llist = LinkedList()
llist.pushAw(21)
llist.pushAw(22)
llist.pushAw(12)
llist.pushAw(14)
llist.pushAw(2)
llist.pushAw(19)
llist.pushAk(9)
llist.deleteNode(0)
```

Hasil Run :

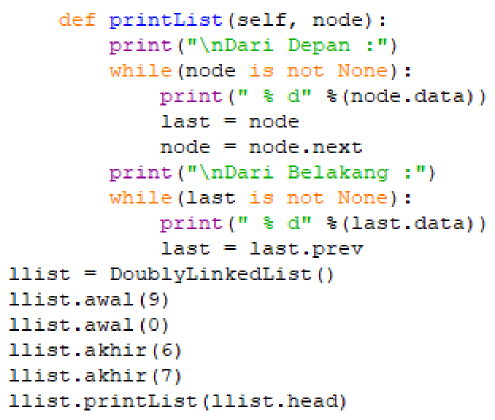


```
True
False
2 14 12 22 21 1 9
```

4. Doubly Linked List
   - Mengunjungi dan mencetak dari depan dan belakang
   - Menambah simpul di awal dan akhir

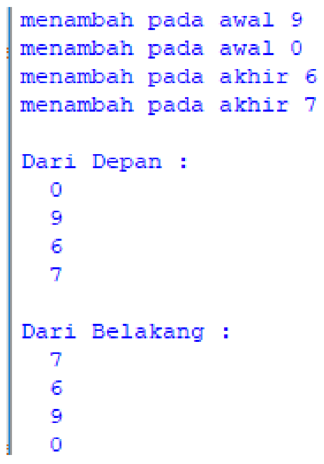


```
class Node:
    def __init__(self, data):
        self.data = data
        self.prev = None
class DoublyLinkedList:
    def __init__(self):
        self.head = None
    def awal(self, new_data):
        print("menambah pada awal", new_data)
        new_node = Node(new_data)
        new_node.next = self.head
        if self.head is not None:
            self.head.prev = new_node
        self.head = new_node
    def akhir(self, new_data):
        print("menambah pada 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
```



```
    def printList(self, node):
        print("\nDari Depan :")
        while(node is not None):
            print(" % d" %(node.data))
            last = node
            node = node.next
        print("\nDari Belakang :")
        while(last is not None):
            print(" % d" %(last.data))
            last = last.prev
llist = DoublyLinkedList()
llist.awal(9)
llist.awal(0)
llist.akhir(6)
llist.akhir(7)
llist.printList(llist.head)
```

Hasil Run :



```
menambah pada awal 9
menambah pada awal 0
menambah pada akhir 6
menambah pada akhir 7

Dari Depan :
  0
  9
  6
  7

Dari Belakang :
  7
  6
  9
  0
```