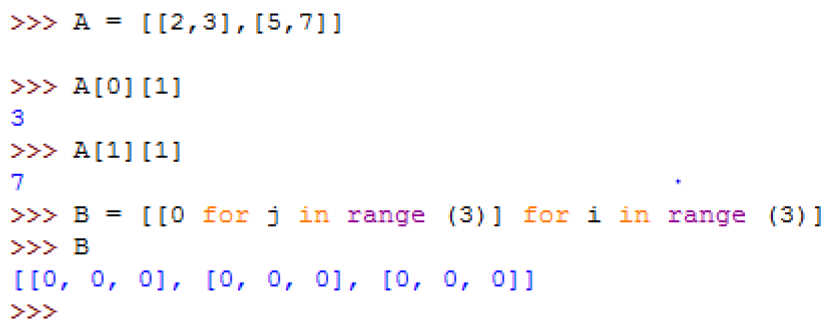
NAMA : DHIYA ULHAQ A

NIM : L200180009

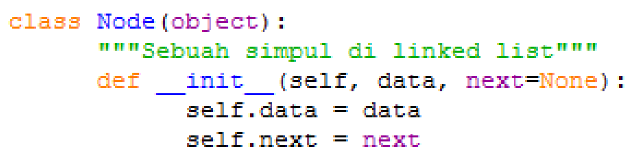
ALGOSTRUK MODUL 3

LATIHAN

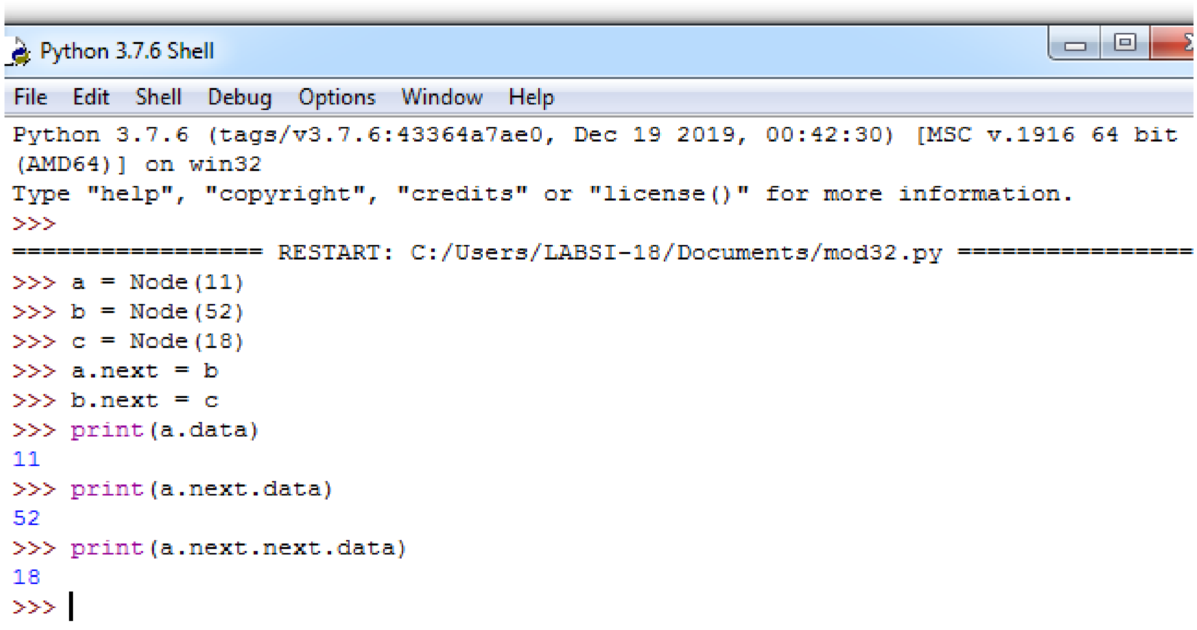
Latihan 3.1 dan 3.2



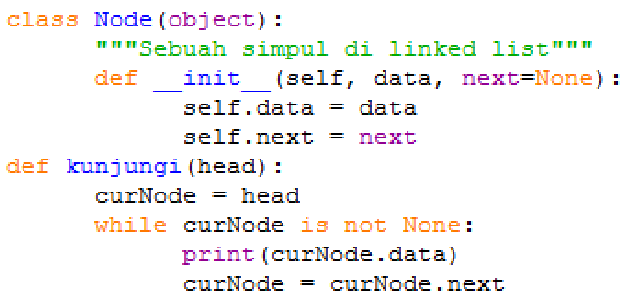
```
>>> A = [[2,3],[5,7]]

>>> A[0][1]
3
>>> A[1][1]
7
>>> B = [[0 for j in range (3)] for i in range (3)]
>>> B
[[0, 0, 0], [0, 0, 0], [0, 0, 0]]
>>>
```



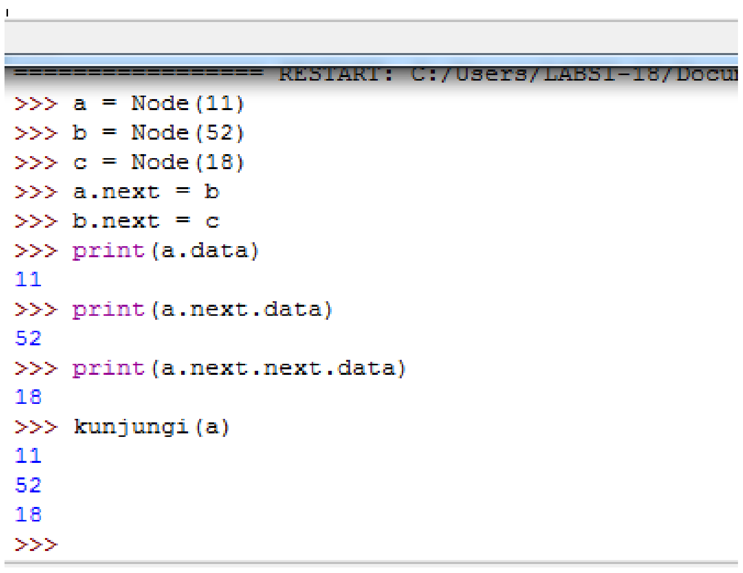
```
class Node(object):
	"""Sebuah simpul di linked list"""
	def __init__(self, data, next=None):
		self.data = data
		self.next = next
```



```
Python 3.7.6 (tags/v3.7.6:43364a7ae0, Dec 19 2019, 00:42:30) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================= RESTART: C:/Users/LABSI-18/Documents/mod32.py ================
>>> a = Node(11)
>>> b = Node(52)
>>> c = Node(18)
>>> a.next = b
>>> b.next = c
>>> print(a.data)
11
>>> print(a.next.data)
52
>>> print(a.next.next.data)
18
>>>
```

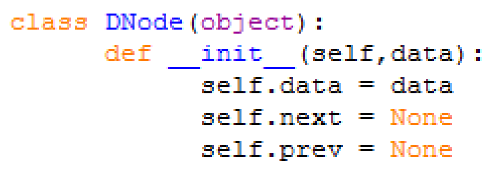


```
class Node(object):
	"""Sebuah simpul di linked list"""
	def __init__(self, data, next=None):
		self.data = data
		self.next = next
def kunjungi(head):
	curNode = head
	while curNode is not None:
		print(curNode.data)
		curNode = curNode.next
```

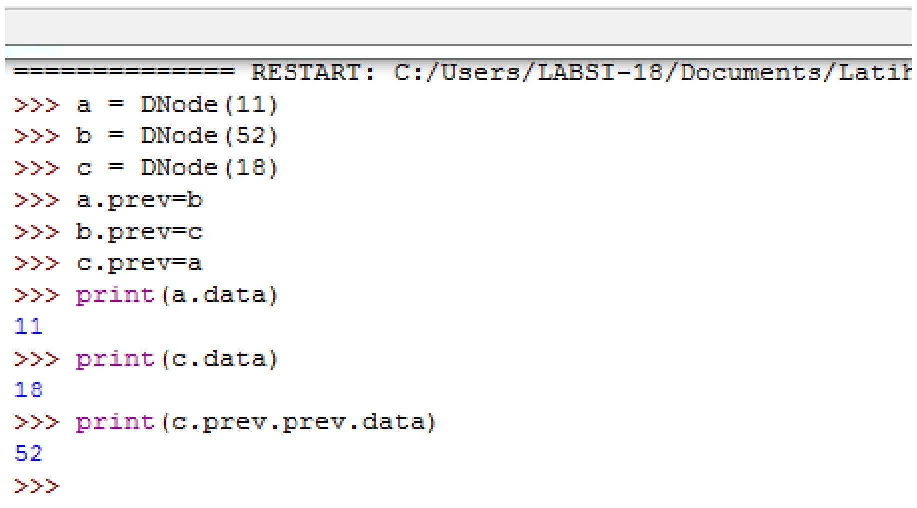


```
================= RESTART: C:/Users/LABSI-18/Docu
>>> a = Node(11)
>>> b = Node(52)
>>> c = Node(18)
>>> a.next = b
>>> b.next = c
>>> print(a.data)
11
>>> print(a.next.data)
52
>>> print(a.next.next.data)
18
>>> kunjungi(a)
11
52
18
>>>
```

Latihan 3.3



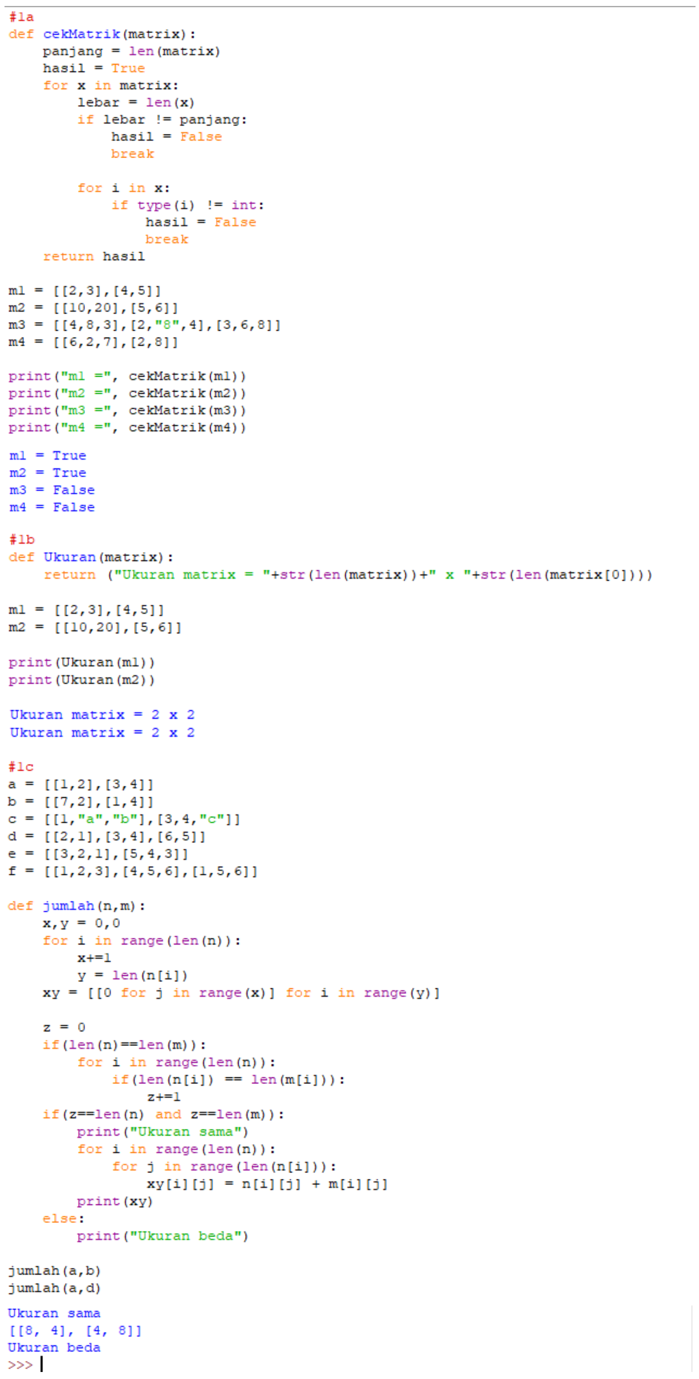
```
class DNode(object):
	def __init__(self,data):
		self.data = data
		self.next = None
		self.prev = None
```



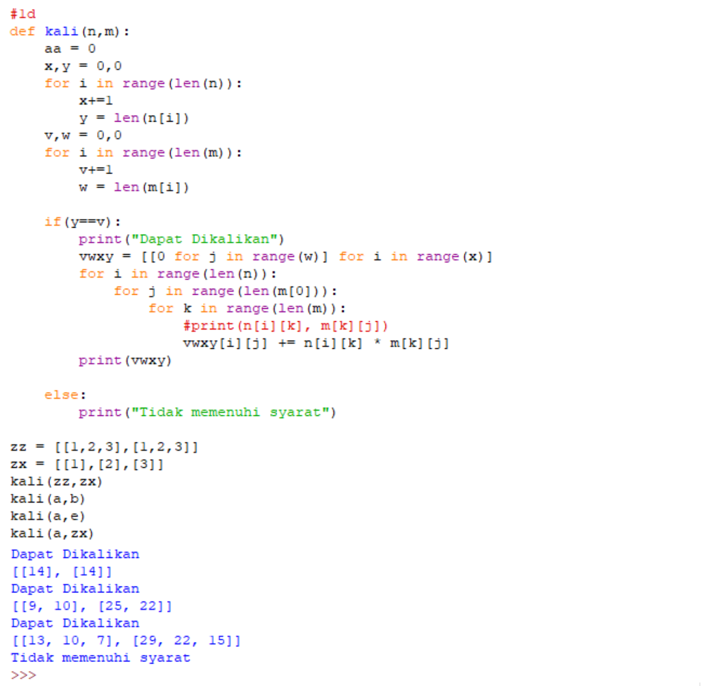
```
============== RESTART: C:/Users/LABSI-18/Documents/Latih
>>> a = DNode(11)
>>> b = DNode(52)
>>> c = DNode(18)
>>> a.prev=b
>>> b.prev=c
>>> c.prev=a
>>> print(a.data)
11
>>> print(c.data)
18
>>> print(c.prev.prev.data)
52
>>>
```

TUGAS

1.

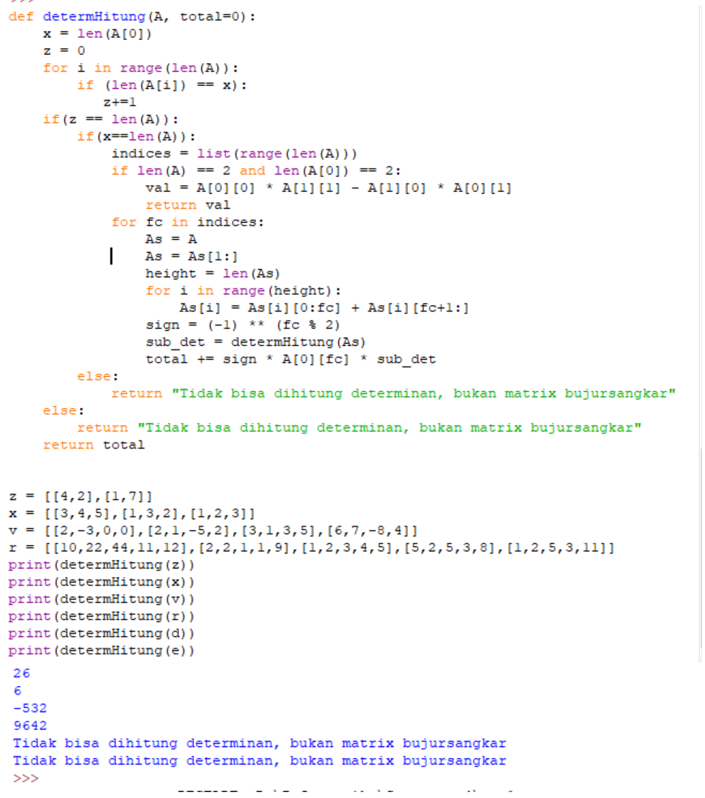


```
#1a
def cekMatrik(matrix):
    panjang = len(matrix)
    hasil = True
    for x in matrix:
        lebar = len(x)
        if lebar != panjang:
            hasil = False
            break

        for i in x:
            if type(i) != int:
                hasil = False
                break
    return hasil

m1 = [[2,3],[4,5]]
m2 = [[10,20],[5,6]]
m3 = [[4,8,3],[2,"8",4],[3,6,8]]
m4 = [[6,2,7],[2,8]]

print("m1 =", cekMatrik(m1))
print("m2 =", cekMatrik(m2))
print("m3 =", cekMatrik(m3))
print("m4 =", cekMatrik(m4))

m1 = True
m2 = True
m3 = False
m4 = False

#1b
def Ukuran(matrix):
    return ("Ukuran matrix = "+str(len(matrix))+" x "+str(len(matrix[0])))

m1 = [[2,3],[4,5]]
m2 = [[10,20],[5,6]]

print(Ukuran(m1))
print(Ukuran(m2))

Ukuran matrix = 2 x 2
Ukuran matrix = 2 x 2

#1c
a = [[1,2],[3,4]]
b = [[7,2],[1,4]]
c = [[1,"a","b"],[3,4,"c"]]
d = [[2,1],[3,4],[6,5]]
e = [[3,2,1],[5,4,3]]
f = [[1,2,3],[4,5,6],[1,5,6]]

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)

Ukuran sama
[[8, 4], [4, 8]]
Ukuran beda
>>> |
```



```
#1d
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("Dapat 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)
```

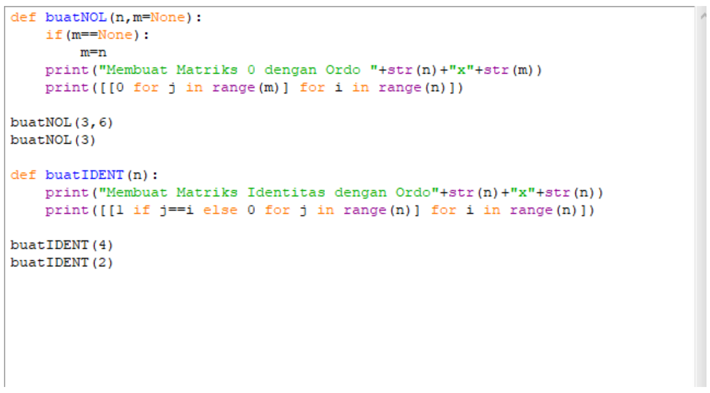
```
Dapat Dikalikan
[[14], [14]]
Dapat Dikalikan
[[9, 10], [25, 22]]
Dapat Dikalikan
[[13, 10, 7], [29, 22, 15]]
Tidak memenuhi syarat
>>>
```



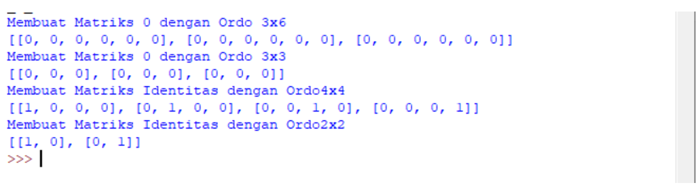
```
def determHitung(A, total=0):
    x = len(A[0])
    z = 0
    for i in range(len(A)):
        if (len(A[i]) == x):
            z+=1
    if(z == len(A)):
        if(x==len(A)):
            indices = list(range(len(A)))
            if len(A) == 2 and len(A[0]) == 2:
                val = A[0][0] * 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 = [[4,2],[1,7]]
x = [[3,4,5],[1,3,2],[1,2,3]]
v = [[2,-3,0,0],[2,1,-5,2],[3,1,3,5],[6,7,-8,4]]
r = [[10,22,44,11,12],[2,2,1,1,9],[1,2,3,4,5],[5,2,5,3,8],[1,2,5,3,11]]
print(determHitung(z))
print(determHitung(x))
print(determHitung(v))
print(determHitung(r))
print(determHitung(d))
print(determHitung(e))
```

```
26
6
-532
9642
Tidak bisa dihitung determinan, bukan matrix bujursangkar
Tidak bisa dihitung determinan, bukan matrix bujursangkar
>>>
```

2.

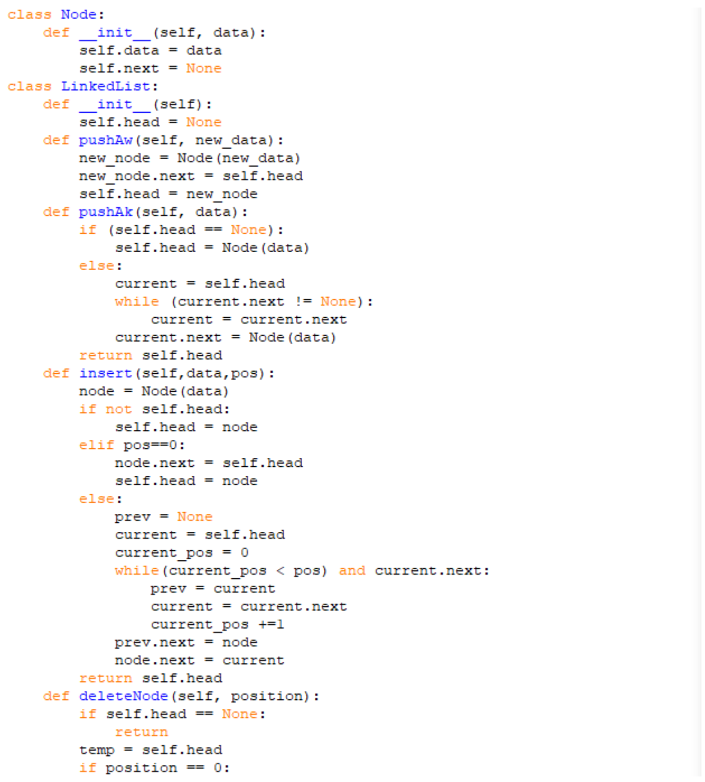


```
def buatNOL(n,m=None):
    if(m==None):
        m=n
    print("Membuat Matriks 0 dengan Ordo "+str(n)+"x"+str(m))
    print([[0 for j in range(m)] for i in range(n)])

buatNOL(3,6)
buatNOL(3)

def buatIDENT(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)])

buatIDENT(4)
buatIDENT(2)
```

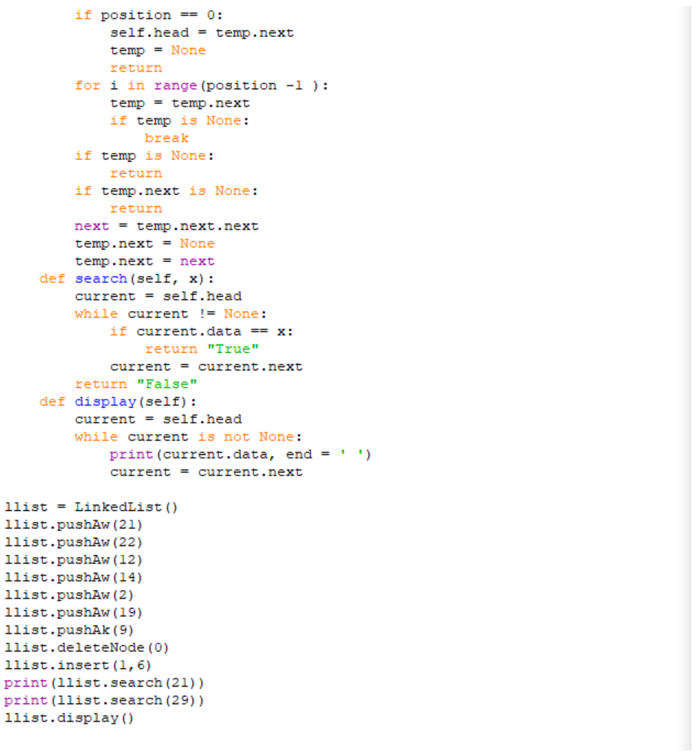


```
Membuat Matriks 0 dengan Ordo 3x6
[[0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0]]
Membuat Matriks 0 dengan Ordo 3x3
[[0, 0, 0], [0, 0, 0], [0, 0, 0]]
Membuat Matriks Identitas dengan Ordo4x4
[[1, 0, 0, 0], [0, 1, 0, 0], [0, 0, 1, 0], [0, 0, 0, 1]]
Membuat Matriks Identitas dengan Ordo2x2
[[1, 0], [0, 1]]
>>> |
```

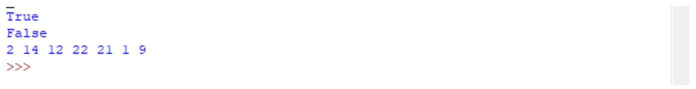
3.



```
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:
```

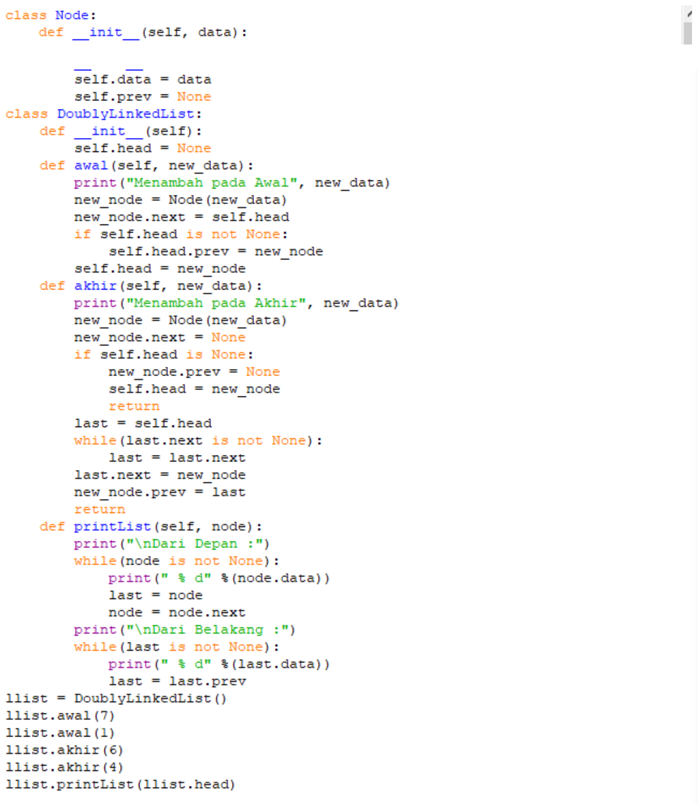


```
        if position == 0:
            self.head = temp.next
            temp = None
            return
        for i in range(position -1 ):
            temp = temp.next
            if temp is None:
                break
        if temp is None:
            return
        if temp.next is None:
            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)
llist.insert(1,6)
print(llist.search(21))
print(llist.search(29))
llist.display()
```

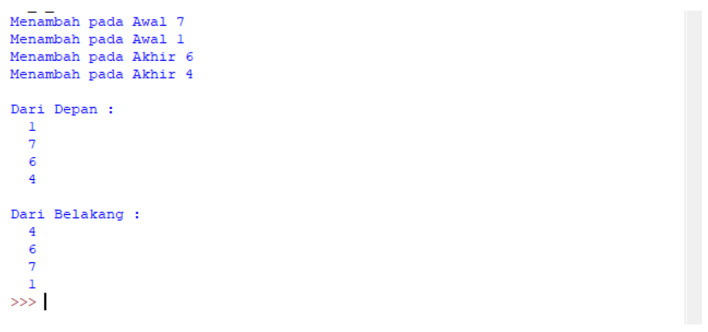


```
True
False
2 14 12 22 21 1 9
>>>
```

4.



```
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(7)
llist.awal(1)
llist.akhir(6)
llist.akhir(4)
llist.printList(llist.head)
```



```
Menambah pada Awal 7
Menambah pada Awal 1
Menambah pada Akhir 6
Menambah pada Akhir 4

Dari Depan :
  1
  7
  6
  4

Dari Belakang :
  4
  6
  7
  1
>>> 
```