1.

n=str(input())

m=str(input())

list1=[n]

list2=[m]

list3=list1+list2

list=' '.join(list3)

print(list)

2.

import numpy as np

m=int(input())

n=int(input())

def transpose(value):

return [[i[j] for i in value] for j in range(0, len(value[0]))]

value=[[0]\*m for i in range(n)]

for i in range(n):

for j in range(m):

a=int(input())

value[i][j]=a

A = np.array(value)

print(A)

trans=transpose(value)

B = np.array(trans)

print(B)

3.

import numpy as np

m1=int(input())

n1=int(input())

m2=int(input())

n2=int(input())

value1=[[0]\*m1 for i in range(n1)]

value2=[[0]\*m2 for i in range(n2)]

for i in range(n1):

for j in range(m1):

a1=int(input())

value1[i][j]=a1

for i in range(n2):

for j in range(m2):

a2=int(input())

value2[i][j]=a2

A = np.array(value1)

B = np.array(value2)

print(A)

print(B)

print(A+B)

print(A.dot(B))

4.

class Polynomial():

def \_\_init\_\_(self,A,B):

self.A = A

self.B = B

def add (self):

b=[]

for i in range(len(self.A)):

b.append(int(self.A[i])+int(self.B[i]))

return b[1:]

def sub(self):

b=[]

for i in range(len(self.A)):

b.append(int(self.A[i])-int(self.B[i]))

return b[1:]

def mul(self):

b = [0 for i in range(self.A[0]+self.B[0]+1)]

self.A.reverse()

self.B.reverse()

for i in range(len(self.A)-1):

for j in range(len(self.B)-1):

X = self.A[i]\*self.B[j]

Y = i+j

b[Y] = b[Y]+X

b.append(self.A[-1]+self.B[-1])

b.reverse()

return(b)

A = [4,3,7,6,0,2]

B = [4,1,5,2,0,9]

C = Polynomial(A,B)

print(C.add())

print(C.sub())

print(C.mul())

5.

class CreateBankAccount():

def \_\_init\_\_(self,ID,name,count):

self.id = ID

self.name = name

self.balance = count

def deposit(self,amount):

if amount<=0:

print("失敗")

else:

self.balance += amount

print("帳戶",self.name,"餘額=",self.balance)

def withdraw(self,amount):

if amount<=self.balance:

self.balance -=amount

print("帳戶",self.name,"餘額=",self.balance)

else:

print("失敗")

def give(self,name,givemoney):

self.withdraw(givemoney)

name.deposit(givemoney)

id1=int(input("請輸入帳戶1之密碼= "))

name1=str(input("請輸入帳戶1之name= "))

id2=int(input("請輸入帳戶2之密碼= "))

name2=str(input("請輸入帳戶2之name= "))

count1 = 50000

count2 = 100000

account1 = CreateBankAccount(id1,name1,count1)

account2 = CreateBankAccount(id1,name2,count2)

a = int(input("請輸入帳戶1之存款金額"))

account1.deposit(a)

b=int(input("請輸入帳戶1之提款金額="))

account1.withdraw(b)

c=int(input("請輸入帳戶2之存款金額="))

account2.deposit(c)

d=int(input("請輸入帳戶2之提款金額="))

account2.withdraw(d)

givename = str(input("請輸入匯入帳戶(account1 or account2)= "))

givemoney = int(input("請輸入匯款金額= "))

if givename == 'account2':

print('已匯款給帳戶2')

account1.give(account2,givemoney)

else:

print('已匯款給帳戶1')

account2.give(account1,givemoney)

6.

class Node():

def \_\_init\_\_(self, data=None):

self.data=data

self.next=None

class Linked\_list():

def \_\_init\_\_(self):

self.head=None

def print\_list(self):

ptr=self.head

while ptr:

print(ptr.data)

ptr=ptr.next

def length(self):

curr = self.head

counter = 0

while curr is not None:

counter += 1

curr = curr.next

print(counter)

link=Linked\_list()

link.head=Node(5)

n2=Node(15)

n3=Node(25)

link.head.next=n2

n2.next=n3

link.print\_list()

link.length()

7.

class Node():

def \_\_init\_\_(self, data=None):

self.data=data

self.next=None

class Linked\_list():

def \_\_init\_\_(self):

self.head=None

def print\_list(self):

ptr=self.head

while ptr:

print(ptr.data)

ptr=ptr.next

def search5(self):

current = self.head

count = 0

while(current is not None):

if current.data == 5 :

count += 1

current = current.next

if count == 0 :

print('5不存在')

if count != 0 :

print('5存在')

def search15(self):

current = self.head

count = 0

while(current is not None):

if current.data == 15 :

count += 1

current = current.next

if count == 0 :

print('15不存在')

if count != 0 :

print('15存在')

def search20(self):

current = self.head

count = 0

while(current is not None):

if current.data == 20 :

count += 1

current = current.next

if count == 0 :

print('20不存在')

if count != 0 :

print('20存在')

def count(self, search\_for):

current = self.head

count = 0

while(current is not None):

if current.data == search\_for:

count += 1

current = current.next

return count

link=Linked\_list()

link.head=Node(5)

n2=Node(15)

n3=Node(15)

link.head.next=n2

n2.next=n3

link.search5()

link.search15()

link.search20()

print('5的出現次數為'+str(link.count(5)))

print('15的出現次數為'+str(link.count(15)))

print('20的出現次數為'+str(link.count(20)))

8.

class Node():

def \_\_init\_\_(self, data=None):

self.data = data

self.next = None

self.prev = None

class Double\_linked\_list():

def \_\_init\_\_(self):

self.head = None

self.tail = None

def print\_list1(self):

ptr = self.head

while ptr:

print(ptr.data)

ptr = ptr.next

def print\_list2(self):

ptr = self.tail

while ptr:

print(ptr.data)

ptr = ptr.prev

link = Double\_linked\_list()

link.head = Node('Sun')

n2 = Node('Mon')

n3 = Node('Tues')

n4 = Node('Wed')

n5 = Node('Thur')

n6 = Node('Fri')

link.tail = Node('Sat')

link.head.next = n2

n2.next = n3

n3.next = n4

n4.next = n5

n5.next = n6

n6.next = link.tail

link.print\_list1()

print('')

link.tail.prev = n6

n6.prev = n5

n5.prev = n4

n4.prev = n3

n3.prev = n2

n2.prev = link.head

link.print\_list2()