```
#----------------ATM1
name=['shruthi','sonu','reena','riya']
pas=[1,2,3,4]
balance=[1000,1500,1200,1000]
def withdraw(current):
    amt=int(input('Enter a amount:'))
    if amt<=balance[current]:
        balance[current]-=amt
        print(balance[current])
    else:print('insufficient')
def deposit(current):
    amt=amt=int(input('Enter a amount:'))
    balance[current]+= amt
    print(balance[current])
def c_balance(current):
    print('Balance is:',balance[current])
def default(current):
    print("enter crt option")


#-----------------ATM2
balance= 50000
srushti_balance=20000
def lavanya():
    print("hi lavanya ")
    pas=input("enter the password: ")
    if pas == "lav18":
        print("access granted")
    while pas!= "lav18":
        new_pas=input("enter correct password")
        if new_pas == "lav18" :
            print("acsess granted")
```

```
            break

    work=input("enter the work ")
    if work == "withdraw":
        amount=int(input("enter the amount"))

        if amount <=balance:
            print("amount withdraw done",amount)
            new_bal= lavanya_balance=balance-amount
            print(new_bal)
        else:
            print("withdraw amount is exceeded than balance")
def srushti():
    print("hi lavanya ")
    pas=input("enter the password: ")
    if pas == "lav18":
        print("access granted")
    while pas!= "lav18":
        new_pas=input("enter correct password")
        if new_pas == "lav18" :
            print("acsess granted")
            break

    work=input("enter the work ")
    if work == "withdraw":
        amount=int(input("enter the amound"))
        lavanya_balance = 20000
        if amount <=srushti_balance :
            print("amount withdraw done",amount)
            new_bal= lavanya_balance=lavanya_balance-amount
            print(new_bal)
        else:
            ("withdraw amount is exceeded than balance")
```

```
def information ():
    person=(input("enter the person name: "))
    switch_data={"lavanya":lavanya,"srushti":srushti}
    result=switch_data.get(person,0)
    result()
information()



#------------------sort
l=[2,5,6,3,8,9,11]
n=len(l)
for i in range(n):
    for j in range(n-i-1):
        if l[j] > l[j+1]:
            l[j],l[j+1]=l[j+1],l[j]
print("sorted list",l)



#--------------------searching
list=[1,2,3,4,5,6,7]
key=5
l=0
r=len(list)-1

while l<=r:
    mid=(l+r)//2
    if list[mid]==key:
        print(f"the vallue found at index of {mid}")
        break
    elif list[mid]>key:
```

```
        r=mid-1
    else:
        l=mid+1
if key  not in list:
    print("value not found")



#------------ linear search
l=list(map(int,input("enter the values of list: ").split(' ')))
key=int(input("enter the key value:"))
for i in range(len(l)):
    if l[i] == key:
        print(f"value found in {i}th index")
        break
if key not in l:
    print("value not found")

if i==len(l)-1 and l[i]!=key:
    print("not found")

#----------------module
import calendar
yy=2004
mm=11
print(calendar.month(yy,mm))
#1
class a:
    def __init__(self):
        print("bingo")
class b(a):
    def __init__(self):
```

```
        print("lays")
class c(b):
      def __init__(self):
          super().__init__()

c()
#2
class dog:
    def sound(self):print("wow")
class cat:
    def sound(self):print("meow")
def make_sound(noise):
    noise.sound()
c=cat()
d=dog()
make_sound(c)

#---------------linkedlist
class node:
    def __init__(self,data):
        self.data=data
        self.next=None

class linkedlist:
    def __init__(self):
        self.head=None

    def _insert_at_end(self,data):
        newnode = node(data)
        if self.head is None:
            self.head= newnode
        else:
            current=self.head
```

```
            while current.next is not None:
                current=current.next
            current.next=newnode


    def _insert_at_beggining_(self,data):
        newnode = node(data)
        if self.head is None:
            self.head= newnode
        else:
            newnode.next = self.head
            self.head = newnode

    def display(self):
        current=self.head
        while current:
            print(current.data,end="-->")
            current=current.next
    def _search_(self,data):
        current=self.head
        count=0
        while current:
            count += 1
            if current.data==data:
                print("data is  found",count)
                count=-1
                break
            current= current.next
        if count!=-1:
                print("data not found")

obj=linkedlist()
obj._insert_at_beggining_(7)
```

```
obj._insert_at_beggining_(6)
obj._insert_at_end(8)
obj.display()
obj._search_(2)


#------------------doublelinkedlist
class node:
    def __init__(self,data):
        self.data=data
        self.next=None
        self.prev=None
class linklist:
    def __init__(self):
        self.head=None
        self.tail=None
    def insert_end(self,data):
        newnode=node(data)
        self.tail=newnode
        if self.head is None:
            self.head=newnode
        else:
            current=self.head
            while current.next:
                current=current.next
            current.next=newnode
            newnode.prev=current
    def insert_before(self,data):
        newnode=node(data)
        self.head=newnode
        if self.head is None:
            self.head=newnode
        else:
```

```
            current=self.head
            while current:
                current=current.next
            current=newnode
            newnode.next=current

    def display_fw(self):
        current=self.head
        while current:
            print(current.data,end="--> ")
            current=current.next
    def display_bw(self):
        current=self.tail
        while current:
            print(current.data,end="<--")
            current=current.prev

obj=linklist()
obj.insert_before(5)
obj.insert_end(6)
obj.insert_before(3)
obj.display_bw()
```