###happy number

def happy(n):

r=s=0

while(n>=0):

for i in range(0,len(str(n))+1):

r=n%10

s=s+r\*\*2

n=n//10

return s

n= int(input("enter the number:"))

res=n

while (res!=1 and res!=4):

res=happy(res)

if res==1:

print("happy number:")

else:

print("not a happy number:")

**Encapsulation**

class encap:

\_a=10

c=20

def encapfunction(self):

\_b=200

print("encap function accessing protected")

print(self.\_a+10)

obj=encap()

print(obj.\_a)

obj.encapfunction()

print(obj.c)

#private

class encap:

\_\_a=10

print(\_\_a)

def encapfunction(self):

print("encap function:")

print(self.\_\_a)

obj=encap()

obj.encapfunction()

print(obj.\_\_a) #will throw error bcs a is private cant be accessed

POLYMORPHISM

>method overloading

class methodoverload:

def display(self,a=None,b=None):

print(a,b)

obj=methodoverload()

print("without arguament")

obj.display()

print("with all atguament")

obj.display(20,30)

print("with one arguament")

obj.display(10)

>>overidding

class vijayawada():

def placename(self):

print ("vijayawada place name is KLU")

def student(self):

print("yes -vijayawada")

def which\_year(selfs):

print("3rd year")

class hyd():

def placename(self):

print ("hyd place name is hyd-KLU")

def student(self):

print("yes -hyd")

def which\_year(self):

print("3rd year")

obj\_vij=vijayawada()

obj\_hyd=hyd()

for details in (obj\_vij,obj\_hyd):

details.placename()

details.student()

details.which\_year()

PHASE II : DATA STRUCTURES USING PYTHON

$$ LINKED LIST

#CREATING NODE DECLARATION AND DEFINITION

class node :

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

self.data=data

self.next=None

class singlelinkedlist:

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

self.head=None

def display(self):

if self.head is None:

print("the linked list is empty")

else:

temp=self.head #temp=first node

while temp:

print(temp.data,"-->",end=" ")

temp=temp.next

obj=singlelinkedlist()

#node creation - initialising

n=node(10)

obj.head=n

n1=node(20)#assigning fst node as head

obj.head.next=n1

n2=node(30)

n1.next=n2

obj.display()

class node :

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

self.data=data

self.next=None

class singlelinkedlist:

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

self.head=None

def display(self):

if self.head is None:

print("the linked list is empty")

else:

temp=self.head #temp=first node

while temp:

print(temp.data,"-->",end=" ")

temp=temp.next

obj=singlelinkedlist()

#node creation - initialising

n=node("W")

obj.head=n

n1=node("T")#assigning fst node as head

obj.head.next=n1

n2=node("N")

n1.next=n2

n3=node("Z")

n2.next=n3

n4=node("E")

n3.next=n4

n5=node("R")

n4.next=n5

obj.display()

OPERATIONS IN LINKED LIST

🡪AT THE BEGINNING

class node:

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

self.data=data

self.next=None

class singlelinkedlist:

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

self.head=None

def insert\_end(self,data):

ne=node(data)

temp=self.head

while temp.next:

temp=temp.next

temp.next=ne

def display(self):

if self.head is None:

print("linked list is empty")

else:

temp=self.head

while temp.next!=None:

print(temp.data,"-->",end=" ")

temp=temp.next

print(temp.data)

obj=singlelinkedlist()

n=node(10)

obj.head=n

n1=node(20)#assigning fst node as head

n.head.next=n1

n2=node(30)

n1.next=n2

n3.node(40)

n2.next=n3

n4=node(50)

n3.next=n4

n5=node(60)

n4.next=n5

print("before inserting 100:")

obj.display()

print(" ")

print("after inserting 100:")

obj.insert\_beginning(100)

obj.display()

print("after inserting 555")

obj.insert\_beginning(555)

obj.display()

-🡪AT THE ENDING

class node:

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

self.data=data

self.next=None

class singlelinkedlist:

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

self.head=None

def insert\_end(self,data):

ne=node(data)

temp=self.head

while temp.next:

temp=temp.next

temp.next=ne

def display(self):

if self.head is None:

print("linked list is empty")

else:

temp=self.head

while temp.next!=None:

print(temp.data,"-->",end=" ")

temp=temp.next

print(temp.data)

obj=singlelinkedlist()

n=node(10)

obj.head=n

n1=node(20)

n.next=n1#assigning fst node as head

n2=node(30)

n1.next=n2

n3=node(40)

n2.next=n3

n4=node(50)

n3.next=n4

n5=node(60)

n4.next=n5

obj.display()

print("before inserting 100:")

obj.display()

print(" ")

print("after inserting 100:")

obj.insert\_end(100)

obj.display()

print("after inserting 555")

obj.insert\_end(555)

obj.display()

$$AT ANY POSITION

class node:

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

self.data=data

self.next=None

class singlelinkedlist:

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

self.head=None

def insert\_position(self,pos,data):

np=node(data)

temp=self.head

for i in range(pos-1):

temp=temp.next

np.next=temp.next

temp.next=ne

def display(self):

if self.head is None:

print("linked list is empty")

else:

temp=self.head

while temp.next!=None:

print(temp.data,"-->",end=" ")

temp=temp.next

print(temp.data)

obj=singlelinkedlist()

n=node(10)

obj.head=n

n1=node(20)#assigning fst node as head

obj.head.next=n1

n2=node(30)

n1.next=n2

n3=node(40)

n2.next=n3

n4=node(50)

n3.next=n4

n5=node(60)

n4.next=n5

obj.display()

print()

obj.display()

print("before inserting 100:")

obj.display()

print(" ")

print("after inserting 100:")

obj.insert\_end(100)

obj.display()

print("after inserting 555")

obj.insert\_end(555)

obj.display()