PYTHON

Dutchman Guido van Rossum developed python in the 1980s

sturcture of python programming

- input
- algoirthm
- output

- example: addition of two integer numbers (stings is the default data type)

```
n=int(input())
ans=n+n
print(ans)
```

Operators in python
membership operator - “in” checks the presence of 1st operand in 2nd operand
identity operator - “is” checks if the operands are identical

conditional & control statements
conditional statements

1.if

example:

```
if n%2==0:  (indentation)
            print(n)
```

2.if else

example:

```
if n%2==0:
            b=7
            print(n,b)
        else:
            print(b)
```

[ends up in a run time error since b is defined in the if statement which will execute only when the condition is satisfied]

3.elif

example:

```
n=int(input())
if n>0:
    print("positive")
elif n<0:
    print("negative")
else:
    print("0")
```

4.nested if

exmaple:

```
if height>5.8:
if weight<50:
```

```
if gender="female":
    print("selected")
```

- **'dictionaries**' can be used as an alternative for switch

example:

```
#select model
height=float(input())
weight=float(input())
gender=input()
if height>5.8:
    if weight<50:
        if gender=="female":
            print("selected")
        else:
            print("not selected")
    else:
        print("not selected")
else:
    print("not selected")
```

Question 1. find if the year is a leap year

*check if the year is divisible by 4 if not print(not leap year)
*check if it is divisible by 100 if not print(leap year)
*check if it is divisible by 400 if not print(not leap year)

```
#leap year
n=int(input())
if n%4==0:
    if n%100==0:
        if n%400==0:
            print("leap yr")
        else:
```

```
            print("not leap yr")
    else:
        print("leap yr")
else:
    print("not leap yr")
```

control statements
1.for & while loops
2.break,continue,pass statement

for loop

n=input()
for i in range(1[start point],10[end point],2[jump/step]):
print(i)
for i in range(1,10,-2)
[if range doesnt match output is not generated]

- using else with loops
  else is executed only when the execution of the loop is completed , if the the exe is terminated the else does not execute
  ex:

  ```
  for i in range(3):
      print(i,"aisha")
      break
  else:
      print("simra")

  #break ends both for loop and else (without executing) statement
  ```

ex:

```
for i in range(3)
    print(i,"aisha")
```

```
    continue
    print("hi")
else:
    print("simra")
```

#continue & break cannot be used in else, it shows compiler error

Question 2. check if a number is prime or not

```
#prime number without flag
n=int(input())
for i in range(2,n//2):
    if n%i==0:
        print("not prime")
        break
else:
    print("prime")



#prime number w flag
n=int(input())
flag=0
for i in range(2,n//2):
    if n%i==0:
        flag=1
        break
if flag==1:
    print("notprime")
else:
    print("prime")
```

Question 3. greatest common divisor

- find the minimum of two numbers
- set GCD to 1, so the loop starts it's iteration from 1
- for in range(1,m+1) end point
- gcd=i (after each iteration)
- print gcd

```
#greatest common divisor
a=int(input())
b=int(input())
m=min(a,b)
gcd=1
for i in range(1,m+1):
    if a%i==0 and b%i==0:
        gcd=i
print(gcd)


#least common multiple
a=int(input())
b=int(input())
m=max(a,b)
lcm=m
for i in range(m,(a*b)+1):
    if i%a==0 and i%b==0:
        lcm=i
        break
print(lcm)
```

- **Function to check the memory location of a variable**

```
a=1
b=1
```

```
print(id(a),id(b))
```

python does not store repeated values in different memory locations
The value of a & b is stored in the same memory location

- when ‘if’ is followed by a command while it requires a true/false condition it only executes the command & terminates
  ex:

  ```
  if print(”hello”):
      print(”hi”)
  ```

  output: hello
- when ‘if’ is followed by an non zero integer number it returns TRUE in pyhton & zero returns FALSE
  ex:

  ```
  if 9999:
      print(”hi”)
  output: hi
  ex:
  if 0:
    print(”hi”)
  ```

  output: _____
- when ‘while’ is followed by a non zero integer it executes the next statement infinite times
  ex:

  ```
  while 999:
      print(”hi”)
  ```

  output:
  hi
  hi
  hi

hi
(infinite times)

ex:
while 0:
    print(”hi”)

output: _____

Question 4. Find the reverse of a number
~take input num
~divide the num by 10 & store the remainder in rem
~initially rev=0

~rev →rev*10+rem

~divide the num by 10 & update num
~put the steps in a while loop

```
# reverse a number
num=int(input())
rev=0
while num!=0:
    rem=num%10
    rev=rev*10+rem
    num=num//10
print(rev)
```

output:

1234
4321

Sum of the digits

```
# sum of digits
num=int(input())
rev=0
while num!=0:
    rem=num%10
    rev=rev+rem
    num=num//10
print(rev)
```

Product of digits

```
# product of digits
num=int(input())
rev=1
while num!=0:
    rem=num%10
    rev=rev*rem
    num=num//10
print(rev)
```

Find if the number is Palindrome or not

```
#palindrome of a number
num=int(input())
m=num
rev=0
while num>0:
      rem=num%10
      rev=rev*10+rem
      num=num//10
if m==rev:
    print("plaindrome")
else:
    print("not palindrome")
```

Count the number of digits

```
#count the number of digits in a number
n=int(input())
r=0
while n!=0:
    r=r+1
    n=n//10
print(r)
```

Count the number of even & odd digits in a number

```
#count the number of odd & even numbers in a number
n=int(input())
even=0
odd=0
while n!=0:
    rem=n%10
    if rem%2==0:
      even=even+1
    else:
       odd=odd+1
    n=n//10
    print("even numbers=",even)
    print("odd numbers=",odd)

    (partially correct)
```

Question 5. Find fibonacci series of a number

~take inputs

~initialize a=0 & b=1
~print (a,b)

~iterate from the 3rd position to the last using for loop

~initialize c=a+b to print the 3rd number

```
#febonacci series
n=int(input())
a=0
b=1
print(a,b,end=" ")
for i in range(3,n+1):
    c=a+b
    print(c,end=" ")
    a=b
    b=c
```

PATTERNS

```
n=int(input())
for i in range(1,n+1):
    for j in range(i,n+i):
        print(j ,end="  ")
    print()

1  2  3
2  3  4
3  4  5
```

```
n=int(input())
for i in range(0,n):
    for j in range(i,n+i):
        print(chr(65+j) ,end="  ")
    print()
A  B  C
B  C  D
C  D  E
```

```
n=int(input())
for i in range(1,n+1):
    for j in range(0,n):
        print(chr(65+j) ,end="  ")
    print()
A  B  C
A  B  C
A  B  C
```

```
n=int(input())
for i in range(1,n+1):
    for j in range(1,i+1):
        print(j ,end="  ")
    print()
1
1  2
1  2  3
```

```
n=int(input())
for i in range(1,n+1):
    for j in range(i,i+i): or (i,2*i)
        print(j ,end="  ")
    print()
1
2  3
3  4  5
```

```
n=int(input())
for i in range(n,0,-1):
    for j in range(i,0,-1):
        print(j ,end="  ")
    print()
```

```
3  2  1
2  1
1
```

```
n=int(input())
for i in range(n,0,-1):
    for j in range(n,n-i,-1):
        print(j ,end="  ")
    print()
3  2  1
3  2
3
```

```
n=int(input())
for i in range(n,0,-1):
    for j in range(i,i+i):
        print(j ,end="  ")
    print()
3  4  5
2  3
1
```

```
n=int(input())
for i in range(n):
    if i==0 or i==n-1:
        for j in range(n):
            print("*",end=" ")
        print()
    else:
        for j in range(n):
            if j==0 or j==n-1:
                print("*",end=" ")
```

```
            else:
                print(" ",end=" ")
        print()
5
* * * * *
*       *
*       *
*       *
* * * * *
```

```
n=int(input())
for i in range(n):
    if i==0 or i==0+1 or i==n-2 or i==n-1:
        for j in range(n):
            print("*",end=" ")
        print()
    else:
        for j in range(n):
            if j==0 or j==0+1 or j==n-2 or j==n-1:
                print("*",end=" ")
            else:
                print(" ",end=" ")
        print()
5
* * * * *
* *   * *
* *   * *
* *   * *
* * * * *
```

```
n=int(input())
for i in range(n):
    if i==0 or i==n-1:
        for j in range(n):
```

```
            print("*",end=" ")
        print()
    elif i==n//2:
        for j in range(n):
            if j==0 or j==n//2 or j==n-1:
                print("*",end=" ")
            else:
                print(" ",end=" ")
        print()
    else:
        for j in range(n):
            if j==0 or j==n-1:
                print("*",end=" ")
            else:
                print(" ",end=" ")
        print()
```

```
* * * * *
*       *
*   *   *
*       *
* * * * *
```

```
#PYramid
n=int(input())
for i in range(1,n+1):
    for j in range(1,n-i+1):
        print(" ",end="")
    for j in range(0,2*i-1):
        print("*",end="")
    print()
```

```
    *
   ***
  *****
```

```
  *******
 *********
```

n=int(input())
for i in range(1,n+1):
if i==1 or i==n:
for j in range(1,n-i+1):
print(" ",end=" ")
for j in range(0,2
i-1):
print("
",end=" ")
print()

else:
for j in range(1,n-i+1):
print(" ",end=" ")
for j in range(0,2
i-1):
if j==0 or j==2
i-2:
print("*",end=" ")
else:
print(" ",end=" ")
print()

```
# 2 PYramids

    4
    *       *
   ***     ***
  *****   *****
 ***********
```

DATA TYPES IN PYTHON

LIST- collection of different data types

list can hold a mixture diff data types

list is mutable

creation of empty list

a=[]

print(type(a))

EX: a=[″aisha″,5,6,7,8]

a[0]=″simra″

print(a)

TUPLE - tuple is immutable

a=()

print(type(a))

EX:

[″aisha″,5,6,7,8]

print(a)

SET - used to store & access unique elements

a=set()

print(type(a))

Ex:

a={1,1,2,2,3,4,6}

print(a)

DICTIONARIES - store 2 elements : key:value

a={1:″aisha″,2:″simra″}

print(a)

STRING - immutable

a=″aisha″

a[0]=”p”

print(a)

o/p: string does not allow changes

LIST

Reversing a list

initially the index of a list starts from 0

```
#list
l=[1,2,3,4,5]
#print[1::-1]
for i in range(len(1)-1,-1,-1):
    print(l)
```

Finding the maximum/minimum number in a list

```
#min
l=[5,6,7,8,9,10]
m=l[0]
for i in range(len(l)):
    if l[i]<m:
        m=l[i]
print(m)


#for i in l (max)
l=[5,6,7,8,9,10]
m=l[0]
for i in l:
    if i>m:
        m=i
print(m)
```

```
#second largest number
l=[5,6,7,8,9,10]
p=set(l)
print(p)
q=list(p)
print(q[-2])
```

Decimal to Binary

~take input

~initialize string s to zero

~in a while loop while n≠0 followed by the conditions

n%2

~store the remainder in string s=s+str(rem)

~store the remainder in “rem”

~ next condition terminates the loop after checking n//2

~s[::-1] statement reverses the string

```
#decimal to binary
n=int(input())
rev=""
while n>0:
    rem=n%2
    rev=rev+str(rem)
    n=n//2
print(rev[::-1])
```

Perfect square

check whether a number is perfect square or not

~take input

~initialize i to 1

~in a while loop multiply i with 2 & check if it is greater than n

~ if yes increment i

~then if i multiplied w 2 is equal to input n

~print perfect square else print not perfect square

```
#perfect square
n=int(input())
i=1
while i**2<n:
    i+=1
if i**2==n:
    print("perfect square")
else:
    print("not perfect square")
```

```
#power of two
n=int(input())
i=1
while 2**i<n:
    i+=1
if 2**i==n:
    print("power of two")
else:
    print("not power of two")
```

likewise power 3 & any number can be checked

Armstrong number

partial code for armstrong number

```
#armstrong number
n=int(input())
s=str(n)
len(s)
p=int(s)
r=0
while num!=0:
    rem=num%10
    r=r*10+rem
    num=num//10
```

correct code for armstrong number

```
#armstrong number
n=int(input())
a=0
t=n
d=len(str(n))
while t>0:
    rem=t%10
    a=a+(rem**d)
    t=t//10
if a==n:
    print("armstrong")
else:
    print("not armstrong")
```

~convert the input into string to find the length

~d=len(str(n))

~using the code of reversing a number

~in the while loop while t>0

~finding modulo of a t dividing by 10 & store remainder in rem

~to find the if the number is armstrong number a=a+(rem**d)

~to continue iteration divide the number by 10 , terminate the decimal value consider the integer & proceed with the loop

Perfect number & Abundant number

```
#perfect number
n=int(input())
d=0
i=1
for i in range(1,n):
    if n%i==0:
        d=d+i
if d==n:
    print("perfect number")
else:
    print("not pefect number")
```

```
#abundant number
n=int(input())
d=0
i=1
for i in range(1,n):
    if n%i==0:
        d=d+i
if d>n:
    print("abundant number")
else:
    print("not abundant number")
```

STRINGS

slicing

```
#slicing
s="abcdefghijklmnopqrstwxyz"
```

```
print(s[20:-10:-2])
```

slicing into odd & even set

o- [2, 4, 6, 8, 10, 9, 7, 5, 3, 1]

```
s=[1,2,3,4,5,6,7,8,9,10]
a=s[1::2]
b=s[-2::-2]
l=[]
l.extend(a)
l.extend(b)
print(l)
```

or concatenate

ROTATION

```
n=int(input())
s="12345"
if n>len(s):
    n=n%len(s)
print(s[n::]+s[0:n])


 #left rotation
 n=int(input())
s="12345"
if n>len(s):
    n=n%len(s)
print(s[0:n]+s[n::]
```

valid number

```
s=input()
if len(s)!=10:
    if s[0]!='6' or '7' or '8' or '9':
```

```
        if s.isdigit():
            print("valid")
            exit
        else:
            print("invalid")
else:
    print("valid")
```

reverse a string

```
s=input()
if len(s)!=10:
    if s[0]!='6' or '7' or '8' or '9':
        if s.isdigit():
            print("valid")
            exit
        else:
            print("invalid")
else:
    print("valid")
```

reverse the letters of the word in a sentence

```
s="hey how are you"
output="uoy era woh yeh"
l=list(s.split(" "))
s=" "
for i in l:
    s=s+i[::-1]+" "
print(s)
```

Palindrome

```
s=input()
s1=s.upper()
if s1==s1[::-1]:
```

```
    print("plaindrome")
else:
    print("not palindrome")
```

Anagram

```
a=input()
b=input()
s=set(a)
s1=set(b)
if s==s1:
    for i in s:
        if a.count(i)!=b.count(i):
            print("not anagram")
            break
    else:
        print("anagram")
else:
    print("not anagram")
```

Recursion

decrementing the value

```
def fact(a,x):
    if a==x:
        return x
    return a*fact(a+1,x)
n=int(input())
a=fact(1,n)
print(a)
```

```
def fact(x):
    if x==4:
        return x
    return x*fact(x-1)
```

```
n=int(input())
a=fact(n)
print(a)
```

Febonacci

```
def fib(x):
    if x<=1:
        return x
    return fib(x-1)+fib(x-2)
n=int(input())
a=fact(n)
print(a)
```

```
def rev(a,x):
    res=0
    if x<=0:
        return res
    rem=x%10
    res=res*10+rem
    return rev(x//10,res)
n=int(input())
a=rev(n,rev)
print(a)
```

power of 3

```
#power of three
def power(i,x):
    if 3**i==x:
        return True
    elif 3**i>=x:
        return False
    else:
        return power(i+1,x)
```

```
n=int(input())
a=power(1,n)
print(a)
```

traverse & count

```
class Solution(object):
    def mostWordsFound(self, sentences):
        count=[]
        for i in sentences:
            l=i.split(" ")
            count.append(len(l))
        return max(count)
```

s=command.replace(”()”,”o”)

s.replace(”(al)”,”al”)

return s

```
class Solution(object):
    def interpret(self, command):
        a=""
        for i in range(len(command)):
            if command[i]=="G":
                a=a+"G"
            elif command[i]=="(" and command[i]==")":
                a=a+"o"
            elif command[i]=="(" and command[i]=="a":
                a=a+"al"
                return a
goal parse interpretation
```

OOPS IN PYTHON

A class is a user defined data type

defined as attributes/variables & behaviour/functions

```
class person:
    def __init__(self,x,y,z):
        self.nickname=x
        self.roll=y
        self.height=z
    def run(self):
        print("i can run", self.nickname, self.roll)
harsha=person("aisha",79,6)
anjali=person("simra",78,5)
harsha.run()
anjali.run()
```

Concepts of OOPS in Python

abstraction

encapsulation

inheritance

polymorphism

ABSTRACTION: is an idea to be implemented later

abstract method - method that contains an idea but not the body

class containing abstract method is an abstract class

```
class mobile: #abstract class
    def functions(self): #abstract method
        pass
class iphone(mobile):
    def functions(self):
        print("This is iphone")
```

```
class samsung(mobile):
    def functions(self):
        print("This is samsung")
iphone13=iphone()
iphone13.functions()
samsungs3=samsung()
samsungs3.functions()
```

ENCAPSULATION:

wrapping up of data
merging of methods & variables

```
class mobile: #abstract class
    def functions(self): #abstract method
        pass
class iphone(mobile):
    def functions(self):
        print("This is iphone")
class samsung(mobile):
    def functions(self):
        print("This is samsung")
iphone13=iphone()
iphone13.functions()
samsungs3=samsung()
samsungs3.functions()
```

encapsulation in python has many loose ends

INHERITANCE

multiple inheritance

```
class parents:
    def coolness(self):
```

```
        print("parents are cool")
class child:
    def coding(self):
        print("i know coding")
class second(child,parents):
    def coping(self):
        print("i can copy")
aisha=second()
aisha.coolness()
aisha.coding()
aisha.coping()
```

hybrid

```
class parents:
    def coolness(self):
        print("parents are cool")
class child:(parents)
    def coding(self):
        print("i know coding")
class second(parents):
    def coping(self):
        print("i can copy")
class third(second,child):
    def singing(self):
        print("i can sing")
aisha=third()
aisha.coolness()
aisha.coping()
```

over ridding

```
class addition:
    def add(self,x,y):
        print(x+y)
class child(addition):
```

```
    def add(self,x,y,z):
        print(x+y+z)
sim=child()
sim.add(5,6,7)
```

reverse of a string |||

```
class Solution(object):
    def reverseWords(self, s):
        l=list(s.split(" "))
        n=[]
        for i in l:
            n.append(i[::-1])
        return " ".join(n)
```

DSA IN PYTHON

Searching

liner & binary

two pointer approach

```
s=int(input())
l=[1,2,3,4,5,6,7,8,9,10]
l.sort()
print(l)
i=0
j=len(l)-1
while i<j:
    mid=(i+j)//2
    if l[mid]==s:
        print(mid,"found")
        break
    elif l[mid]>s:
        j=mid-1
```

```
    else:
        i=mid+1
else:
    print("not found")
```

complexity

linear search

O(1) - best time complexity

constant time

O(n) - Average time complexity

depends on the fixed number of times

O(n) - Worst time complexity

binary search

O(n) - when i is at mid pos

O(log n) - size decreasing by half with each iteration

SORTING

bubble sort- largest number goes to the end

selection sort - opposite of bubble sort

insertion sort -

bubble sort using sliding window

```
l=[9,7,97,10,5,1,0]
for i in range(0,len(l)-1): #checks for the sorted
    for j in range(0,len(l)-i-1): #checks & compares the consequ
        if l[j]>l[j+1]:
            l[j],l[j+1]=l[j+1],l[j]
print(l)
```

selection sort

```
b=[6,5,4,3,21,19,10]
for i in range(0,len(b)-1):
    m=i
    for j in range(i+1,len(b)):
        if b[m]>=b[j]:
            m=j
    b[i],b[m]=b[m],b[i]
print(b)
```

insertion sort

```
b=[3,4,5,6,19,21,10]
for i in range(1,len(b)):
    j=i-1
    a=b[i]
    while j>=0 and b[j]>a:
        b[j+1]=b[j]
        j-=1
        b[j+1]=a
print(b)
```

Merge sort using Divide & Conquer

~ define a function mergesort(arr,beg,end)

~separate all the elements until we get single separate one

~if beg<end: mid=(beg+end)//2

~after dividing call the functions w parameters -

(left part) mergesort(arr,beg,mid+1)

(right part) mergesort(arr,mid,end)

~conquer by calling merge(arr,beg,mid,end)

```
#merge sort
def merge(arr,beg,mid,end):
    n1=mid-beg+1
```

```
    n2=end-mid
    i=j=0
    left=arr[beg:mid+1]
    right=arr[mid+1:end+1]
    k=beg
    while i<n1 and j<n2:
        if left[i]<right[j]:
            arr[k]=left[i]
            i+=1
        else:
            arr[k]=right[j]
            j+=1
        k+=1
    while i<n1:
        arr[k]=left[i]
        k+=1
        i+=1
    while j<n2:
        arr[k]=right[i]
        k+=1
        j+=1
def mergesort(arr,beg,end):
    if beg<end:
        mid=(beg+end)//2
        mergesort(arr,beg,mid)
        mergesort(arr,mid+1,end)
        merge(arr,beg,mid,end)
a=[8,7,6,5,4,3,2,1]
b=0
e=len(a)-1
mergesort(a,b,e)
print(a)
```

Quick sort

```
#quick sort
def partition(arr,low,high):
    pivot=arr[low]
    start=low+1
    end=high
    while True:
        while start<=end and arr[start]<=pivot:
            start+=1
        while start<=end and arr[end]>pivot:
            end-=1
        if start<end:
            arr[start],arr[end]=arr[end],arr[start]
        else:
            break
    arr[low],arr[end]=arr[end],arr[low]
    return end
def quicksort(arr,beg,end):
    if beg<end:
        p=partition(arr,beg,end)
        quicksort(arr,beg,p-1)
        quicksort(arr,p+1,end)
a=[8,7,6,5,4,3,2,1]
b=0
e=len(a)-1
quicksort(a,b,e)
print(a)
```

STACK

```
class stack:
    def __init__(self):
        self.top=-1
        self.size=5
        self.list=[]
```

```
    def push(self,data):
        if len(self.list)==5:
            print("full")
            return 0
        self.top+=1
        self.list.append(data)
    def pop(self):
        if len(self.list)==0:
            print("empty")
            return 0
        self.top-=1
        self.list.pop()
    def peek(self):
        print(self.list)
        if len(self.list)==0:
            print("empty")
            return 0
        elif self.top>5:
            print("out of index")
        else:
            print(self.list[self.top])
s=stack()
s.push(1)
s.push(2)
s.push(3)
s.push(4)
s.push(5)
s.push(6)
s.pop()
```

QUEUE

```
class queue:
    def __init__(self):
        self.front=-1
        self.rear=-1
```

```
        self.size=5
        self.list=[]
    def enqueue(self,data):
        if len(self.list)==5:
            print("full")
            return 0
        self.rear+=1
        self.list.append(data)
        if self.front==-1:
            self.front+=1
    def dequeue(self):
        if len(self.list)==0:
            print("empty")
            return 0
        self.rear-=1
        self.list.pop(0)
        self.front+=1
    def display(self):
        if len(self.list)==0:
            print("empty")
            return 0
        print(self.list)
s=queue()
s.enqueue(1)
s.enqueue(2)
s.display()
s.dequeue()
s.display()
```

Evaluation of Postfix Expression

```
l="5678+-*"
a=[]
for i in l:
    if i.isdigit():
        a.append(int(i))
```

```
    else:
        f=a.pop()
        s=a.pop()
        if i=="+":
            a.append(f+s)
        elif i=="-":
            a.append(f-s)

        elif i=="*":
            a.append(f*s)
print(a)
```

```
l=[1,2,3,4]
s=[]
def pop():
    for i in range(len(l)):
        s.append(l.pop())
    s.pop()
    for i in range(len(s)):
        l.append(s.pop())
pop()
print(l)
```

```
class Node:
    def __init__(self,value):
        self.data=value
        self.next=None
class linkedlist:
    def __init__(self):
        self.head=None
    def insertbeg(self,value):
        newnode=Node(value)
        if self.head==None:
            self.head=newnode
```

```
        else:
            newnode.next=self.head
            self.head=newnode
    def printlist(self):
        curr=self.head
        while (curr!=None):
            print(curr.data,"->",end=" ")
            curr=curr.next
        print("null")
l=linkedlist()
l.insertbeg(1)
l.insertbeg(2)
l.insertbeg(3)
l.insertbeg(4)
l.printlist()
```

```
class Node:
    def __init__(self,value):
        self.data=value
        self.next=None
class linkedlist:
    def __init__(self):
        self.head=None
    def insertbeg(self,value):
        newnode=Node(value)
        if self.head==None:
            self.head=newnode
        else:
            newnode.next=self.head
            self.head=newnode
    def insertend(self,value):
        newnode=Node(value)
        if self.head==None:
            self.head=newnode
        else:
```

```
            curr=self.head
            while curr.next!=None:
                curr=curr.next
            curr.next=newnode
    def printlist(self):
        curr=self.head
        while (curr!=None):
            print(curr.data,"->",end=" ")
            curr=curr.next
        print("null")
l=linkedlist()
l.insertbeg(1)
l.insertbeg(2)
l.insertbeg(3)
l.insertbeg(4)
l.insertend(5)
l.insertend(6)
l.printlist()
```

All the CRUD funcs in LINKEDLIST

```
class Node:
    def __init__(self,value):
        self.data=value
        self.next=None
class linkedlist:
    def __init__(self):
        self.head=None
    def insertbeg(self,value):
        newnode=Node(value)
        if self.head==None:
            self.head=newnode
        else:
            newnode.next=self.head
            self.head=newnode
    def insertend(self,value):
```

```
        newnode=Node(value)
        if self.head==None:
            self.head=newnode
        else:
            curr=self.head
            while curr.next!=None:
                curr=curr.next
            curr.next=newnode
    def insertany(self,value,key):
        newnode=Node(value)
        if self.head==None:
            self.head=newnode
        else:
            curr=self.head
            while curr!=None:
                if curr.data==key:
                    newnode.next=curr.next
                    curr.next=newnode
                    break
                curr=curr.next
    def count(self):
        count=0
        if self.head==None:
            print("not found")
        else:
            curr=self.head
            while curr!=None:
                count+=1
                curr=curr.next
            print(count)
    def insertmid(self,val):
        newnode=Node(val)
        count=0
        if self.head==None:
            self.head=newnode
        elif self.head.next==None:
```

```
            self.head.next=newnode
        else:
            fast=self.head
            slow=self.head
            while fast.next!=None and fast.next.next!=None:
                fast=fast.next.next
                slow=slow.next
            newnode.next=slow.next
            slow.next=newnode
    def printlist(self):
        curr=self.head
        while curr!=None:
            print(curr.data,"->",end=" ")
            curr=curr.next
        print("null")
    def search(self,target):
        curr=self.head
        while curr!=None:
            if curr.data==target:
                print("found")
                break
            curr=curr.next
        else:
            print("not found")
l=linkedlist()
l.insertbeg(1)
l.insertbeg(2)
l.insertbeg(3)
l.insertbeg(4)
l.insertend(5)
l.insertend(6)
l.insertany(9,3)
l.search(4)
l.insertmid(10)
```

```
l.printlist()
l.count()
```

```
class node:
    def __init__(self,data):
        self.left=None
        self.data=data
        self.right=None
def inorder(root):
    if root:
        inorder(root.left)
        print(root.data)
        inorder(root.right)
def preorder(root):
    if root:
        print(root.data)
        preorder(root.left)
        preorder(root.right)
def postorder(root):
    if root:
        postorder(root.left)
        postorder(root.right)
        print(root.data)
r=node(1)
r.left=node(2)
r.right=node(3)
r.left.left=node(4)
r.left.right=node(5)
inorder(r)
print(" ")
preorder(r)
print(" ")
postorder(r)
```

```
#tree
```

```
#binary search tree
class node:
    def __init__(self,data):
        self.left=None
        self.data=data
        self.right=None
class tree:
    def __init__(self):
        self.root=None
    def insert(self,value):
        newnode=node(value)
        if self.root is None:
            self.root=newnode
        else:
            curr=self.root
            while True:
                if value<=curr.data:
                    if curr.left is None:
                        curr.left=newnode
                        break
                    else:
                        curr=curr.left
                else:
                    if curr.right is None:
                        curr.right=newnode
                        break
                    else:
                        curr=curr.right
def inorder(root):
    if root:
        inorder(root.left)
        print(root.data,end=" ")
```

```
        inorder(root.right)
def preorder(root):
    if root:
        print(root.data,end=" ")
        preorder(root.left)
        preorder(root.right)
def postorder(root):
    if root:
        postorder(root.left)
        postorder(root.right)
        print(root.data,end=" ")
r=tree()
r.insert(5)
r.insert(2)
r.insert(6)
inorder(r.root)
print(" ")
preorder(r.root)
print(" ")
postorder(r.root)
```

GRAPHS

adjacency matrix

```
class graph:
    def __init__(self):
        self.matrix=[[0]*5 for i in range(5)]
    def addvertex(self,a,b):
       self.matrix[a][b]=1
    def print(self):
        for i in self.matrix:
            print(i)
```

```
g=graph()
g.addvertex(1,2)
g.addvertex(4,2)
g.addvertex(1,4)
g.addvertex(2,3)
g.addvertex(4,3)
g.print()
```

adjacency list

```
class graph:
    def __init__(self):
        self.matrix={}
    def addvertex(self,a,b):
        if a not in self.matrix:
            self.matrix[a]=[b]
        else:
            self.matrix[a].append(b)
    def print(self):
            print(self.matrix)
g=graph()
g.addvertex(1,2)
g.addvertex(4,2)
g.addvertex(1,4)
g.addvertex(2,3)
g.addvertex(4,3)
g.print()
```

BFS

```
class graph:
    def __init__(self):
        self.matrix={}
    def addvertex(self,a,b):
         if a not in self.matrix:
           self.matrix[a]=[b]
```

```
          else:
             self.matrix[a].append(b)
    def print(self):
             print(self.matrix)
    def bfs(self,data):
        visited=[]
        queue=[data]
        while queue:
            vertex=queue.pop(0)
            print(vertex)
            if vertex in self.matrix:
             for i in self.matrix[vertex]:
                if  i not in visited:
                    visited.append(i)
                    queue.append(i)
g=graph()
g.addvertex(1,2)
g.addvertex(1,3)
g.addvertex(2,4)
g.addvertex(2,5)
g.print()
```

DFS

```
class graph:
    def __init__(self):
        self.matrix={}
    def addvertex(self,a,b):
         if a not in self.matrix:
           self.matrix[a]=[b]
         else:
            self.matrix[a].append(b)
    def print(self):
             print(self.matrix)
    def dfs(self,data):
        visited=[]
```

```
        stack=[data]
        while stack:
            vertex=stack.pop()
            print(vertex)
            if vertex in self.matrix:
             for i in self.matrix[vertex]:
                if  i not in visited:
                    visited.append(i)
                    stack.append(i)
g=graph()
g.addvertex(1,2)
g.addvertex(1,3)
g.addvertex(2,4)
g.addvertex(2,5)
g.print()
```