



PYTHON

15-04-2024

By:H SRINIVAS

Under the guidance of Bijen Singha sir.

Creator of Python - Dutchman Guido van Rossum

STRUCTURE

- input algorithm
- output
- INPUT

they are of 2 types

②② **Run time/dynamic**(if the error occurs during run time it is said to be run time error)

②② **Compile time/static**(if the error occurs due to syntactical errors or spelling mistakes then it is called as compile time error)

- In python if `input()` - is a default function for string `n=input()// n`
stores data in string format `n==int(input())//n` stores data in integer format

•

Python doesn't use braces, it uses the concept of indentation •Colon is used in conditional statements

| *Some important things to remember:*

OPERATORS

②② MEMBERSHIP OPERATOR `⇒in` only works

with strings checks if first thing is present in
second

```

#ex 1
if 'hi ' in 'hi,hello':
    print("hi")
else:
    print("bye")
print("abc")

#prints bye abc because 'hi ' as has space in it

#ex 2
if 1 in 123:
    print(1)
elif 1 in '123':
    print(2)
else:
    print(3)

#membership operator only works on strings and not on integers

```

[IDENTITY OPERATOR \$\Rightarrow\$ is](#)

checks if both are identical are not

```

if not true is true:
    print(true)
elif not true is false:
    print(false)
else:
    print("error")

#o/p: false

```

[TERNARY OPERATOR](#)

SYNTAX  <expression>if<if condition>else<else condition>

```

x=y=10
z=1+(x if x>y else y)+2
print(z)

#ans=13

#ex:1
age =20
x 'can vote' if age>18 else 'cannot vote'
print(x)

#o/p:can vote

#ex 2:
x=y==10
z= 1+(x if x>y else y)+2

#o/p:13

```

~Derived Datatypes

List	Tuple	Dictionary	Set
a collection of different datatypes	tuple doesn't allow the data to be changed it is immutable.	in python dictionary contains key and values	set only shows unique values
empty list - a=[]	empty tuple - a=()	empty dictionary - a={ }	empty set - a=set()
a[] print(type(a)) o/p: <class 'list'>	a() print(type(a)) o/p: <class 'tuple'>	a{} print(type(a)) o/p: <class 'dictionary'>	a=set() print(type(a)) o/p: <class 'set'>
list rep: a=["hey","h",3.09,9] o/p: ["hey","h",3.09,9]	tuple rep: a=("hey","h",3.09,9) o/p: ("hey","h",3.09,9)		a="bijen" a[0]=p print(a)
always represented by index value			#this only prints bijen as string is not mutable

list is mutable			
-----------------	--	--	--

- just extra

```
#ex 1
if 3 and 0 or 2:
    print("true")
else:
    print("false")

#prints true

#ex 2
if 3>7:
    if -3>-6:
        print("nested if")
    else:
        print("nested else")
else:
    print("else")
#o/p: else

#ex 3
if 11>15 or 12<15:
    if true and false:
        if not false:
            print(1)
            if 101 and 543:
                print(2)
            else:
                print(3)
        else:
            print(4)
```

- all non zero integers will return true in python and only zero will return false

```
if 999:
    print("hi")

#this prints hi as 999 is a non zero integer
```

- condition is a statement which will return in true or false

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

#this prints only hello as - if print("hello") is a statement and not
```

```
while print("hi"):
    print("hello")

#prints hi
```

```
while 999:
    print("hello")

#prints hello n number of times
#if while 0: then it prints no output
```

- Python stores variables with same value in the same location
- checking location of a variable : use id()

```
a=1
b=1
print(id(a),id(b))

#prints the location of the variable
```

- else can be used inside loopsonly (only when there's no break inside loop i.e. for/while will run completely)

```
for i in range(3):
    print(i,"i am sorry")
    #break(if used prints "0 i am sorry"
else:
    print("not sorry")
```

o/p:

0 i am sorry

1 i am sorry2 i am sorry not sorry [continue](#) will skip the lines below it and go to next statement or loop [compile time error](#) : syntax or mistake in program [run time error](#) : error after running [default datatype - string](#)

- basically concatenates ex : n=2 then n+n=22
-
-
-

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

o/p:

4
8

- python doesn't have concept of braces it uses [indentation use colon\(:\)](#)
- -to represent end of condition

-after else:

•[flag](#)-uses binary digits, used when there are only two conditions

2 MAIN CONCEPTS OF PROGRAMMING

CONDITIONAL

-[if](#): (exactly one condition)

```
#ex1
n=int(input())
if n%2==0:
    b=7
    print(n, b)  #program doesn't execute because of indentation
```

```
#ex2
```

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

-**if else:** (exactly 2 conditions)

```
n=int(input())  
if n%2==0:  
    b=7  
    print(n, b)  
else:  
    print(b)          #doesn't get printed as b is defined in if block not  
  
#o/p: name b is not defined//run time error
```

-**elif**(no limit to no of conditions) example:

```
n=int(input())  
if n>0:  
    print("positive")      #(indentation is important in the next statement  
elif n<0:  
    print("negative")  
else:  
    print("0")
```

-**nested if** (all conditions need to be satisfied)

```
if height<5.5  
    if weight==65  
        if gender==female  
            print("All conditions satisfied")
```

-**dictionary**(switch can be implemented in python using dictionary(switcher.get() is the function used to get the value from switcher)

CONTROL

-for -

while loop -while loop doesn't work if it equal to zero or if it starts from zero -no do while loop in python

break

continue -

pass : has no effect on the program

*programs done using for loop or while loop can also be done using recursion

FOR LOOP

*reduces one by default. if 10 is given it'll run up to 9 SYNTAX for i in range():

inside range we have ~initialization

~number of times to iterate

~step(basically skipping)

example :

```
n=input()
for i in range(1,10,2):
    print(n)
```

*try 1,10,2 1,10,2 10,1,2 so on...

*in order to get the values of i or know how many times the loop is getting executed we can write the program like this

```
n=input()
for i in range(10):
    print(i,n)
```

o/p:

```
0 i am sorry
1 i am sorry
2 i am sorry3 i am sorry 4 i am sorry 5 i am sorry 6 i am sorry
7 i am sorry
8 i am sorry9 i
am sorry
```

Extra examples:

```
n=input()
for i in range(1,10,2):
    print(i, n)
```

//range is 1 to 10 and it'll jump by 2 so
output will be

1 sorry
3 sorry 5
sorry
7 sorry 9
sorry

```
n=input()
for i in range(1,10,-2):
    print(i, n)
```

no o/p is generated as range is 1 to 10 but step is -2 which means move behind which can't be done

```
n=input()
for i in range(-10,1,-2):

    print(i, n)
```

no o/p as -10-2 is -12 which means to move behind it can't be done

```
n=input()
for i in range(10,-1,-2):
    print(i)
```

o/p
8 6
4
2
0

In python loops can be executed along with conditional statements which cant be done in c example:

```

#ex1
n=input()
for i in range(3):
    print(i, n)
else:
    print("not sorry")

#ex2
n=input()
for i in range(3):
    print(i ,n)
    break
else:
    print("not sorry")

#o/p: 0 i am sorry
#since break is given else part is not executed

#ex3
n=input()

for i in range(3):
    print(i)
    continue
    print("hi")
else:
    print("not sorry")

```

ex3 o/p:

```

0
1
2
not sorry

```

#after continue the lines which will be written those will be skipped where as break will break the entire loop

Imp:

Continue break or pass must be used only inside the loops

If there is conditional part along with the loop we can't use continue break or pass inside the conditional part

QUESTIONs

check for leap year year

if divisible by 4 check if divisible by
100 check if divisible by 400 if satisfies all
conditions then LEAP YEAR

```
year=int(input())
if year%4==0:
    if year%100==0:
        if year%400==0:
            print("leap year")
        else:
            print("not leap")
    else:
        print(" leap")
else:
    print("not leap")
```

prime number

• prime number is a number which is divisible by itself and 1

ex: 17

17 must not be divisible by any number b/w 1 to 17

-

```
n=int(input())
for i in range(2,n):          #(2,n//2)
    if n%i==0:
        print("not prime")
    else:
        print("prime")
```

range starts from 2 because 1 is divisible by all numbers

*using flag:

```
n=int(input())
flag=0
for i in range(2,n):
    if n%i==0:
        flag=1
        break
    if flag==1:
        print("not prime")
    else:
        print("prime")
```

Q2 Greatest common divisor(GCD) min()- determines the

- smallest number b/w a and b
- g- will be the least gcd i.e. 1 because 1 is a divisor for all numbers and will iterate to i range lies b/w 1
- and smallest number in all cases

ex: 12 1 2 3 4 6 12 36

1 2 3 6 9 12 18 36

ans=12

```
a=int(input())
b=int(input())
m=min(a,b)
g=1
for i in range(1,m+1):
    if a%i==0 and b%i==0:
        g=i
print(g)
```

Least Common Multiple(LCM)

*max()
*g not equal to 1 but bigger number
*limit starts from max no because bigger no can be multiple of small but not vice versa ex: 6 is multiple of 3 3 is not multiple of 6

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

16/04/2024

Find reverse of a number ex:

n=1234
→rev=0 stores the result

→divide 1234 by 10 to get remainder as 4 and store it in rem $\text{rem}=\text{n}\%10$
→ $\text{rev}=\text{rev}*10+\text{rem}$
→then update the n from 1234 to 123 then 12 then 1 so on $\text{n}=\text{n}/10$ this will give us
123.4 but as we used // we ignore the decimal part

```
n=int(input())
rev=0
while n>0:                      #while n!=0
    rem=n%10
    rev=rev*10+rem
    n=n//10
print(rev)
```

② Find multiplication of all the digits in a number

```
n=int(input())
rev=1
while n>0:
    rem=n%10
    rev=rev*rem
    n=n//10
print(rev)

#ans will be multiplication of all numbers
```

③ Find the addition of all the digits in a number

```
n=int(input())
rev=0
while n>0:
    rem=n%10
    rev=rev+rem
    n=n//10
print(rev)

#ans will be addition of all numbers
```

22 Find if a number is palindrome or not

Palindrome is a number read same forward and backward

→store the number in different variable as n will be updated and end value will be 0

→so to compare it with the reverse number we need the original value so store n in m

```
n=int(input())
m=n
rev=0
while n>0:
    rem=n%10
    rev=rev*10+rem
    n=n//10

if m==rev:
    print("palindrome")
else:
    print("not palindrome")
```

22 Find the number of digits in a number

```
n=int(input())
r=0
while n>0:
    r=r+1
    n=n//10
print(r)
```

222 Find number of even and odd numbers

```
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)
print(odd)
```

222 Find factorial of a number

```
n=int(input())
fact=1
for i in range(1,n+1):
    fact=fact*i
print(fact)
```

222 Find fibonacci series

0 1 1 2 3 5 8 13 21

→assume 2 numbers 0 and 1

```
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
```

QUESTION PATTERNS 20

1.

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

o/p:

1 2 3

1 2 3

1 2 3

ANSWER

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

o/p:

1 3 5

1 3 5

1 3 5

ANSWER

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

o/p: 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()
```

o/p:

A B C

B C D

C D E

¶¶

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

        print(j,end=" ")
    print()
```

o/p:

1

1 2

1 2 3 ¶¶

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

o/p:

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

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

o/p:

```
1
3 5
5 7 9
```

??

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

o/p: 3

```
2 1
3 2
3
```

Q3

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

O/P: * * * *

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

Q3Q3

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

o/p: * * *

* * * *

* *

████

```
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()
```

o/p:

5

* * * *

*

* *

*

* * * *

17/04/2024

प्र० १

```
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()
```

5

*

* * *

* * * * *

* * * * * *

* * * * * * *

प्र० २

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

5

* * * * * * * *

* * * * * *

* * * * *

* * *

*

प्र० ३

```

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()

```



प्र०

```

n=int(input())
for i in range(n,0,-1):
    if i==0 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()

```



प्र०

```

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

```

```

for j in range(0,2*i-1):
    print("*",end=" ")
print()

```



प्र०००

```

n=int(input())
for i in range(1,n+1):
    if i==1:
        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()
for i in range(n-1,0,-1):
    if i==1:
        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()

```



प्र० १

```

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=" ")
    for j in range(1,(n-i+1)*2):

```

```

    print(" ",end=" ")
for j in range(0,2*i-1):
    print("*",end=" ")
print()

```

5

```

      *
      * * *
      * * * * *
      * * * * * *
      * * * * * * *
      * * * * * * * *
      * * * * * * * * *
      * * * * * * * * *
      * * * * * * * * *
      * * * * * * * * *
      * * * * * * * *
      * * * * * *
      * * *
      *

```

प्र०प्र०प्र०

```

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

```

5

```

      *
      * * *
      * * * * *
      * * * * * *
      * * * * * * *
      * * * * * * * *
      * * * * * * * * *
      * * * * * * * * *
      * * * * * * * * *
      * * * * * * * * *
      * * * * * * * *
      * * * * * *
      * * *
      *

```

प्र०प्र०प्र० program to print heart

```

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

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

```

a. Program to print K

```

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

```

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

b. Program to print A

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

c. program to print s

```

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

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

```

d. program to print T

```

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

```

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

FUNCTION

set of instructions that perform particular task 3

aspects of function are:

②② definition

②② call

②② no declaration SYNTAX② def

function_name(): example:

```
def addition():
    a=3
    b=8
    c=a+b
    print(c)
```

4 TYPES②

1. with parameters and with return values:

```
def addition(a,b):
    c=a+b
```

```
    return c
```

```
addition(5, 6)
```

2. with parameters and with no return values:

```
def addition(a,b):  
    c=a+b  
    print(c)  
addition(5, 6)
```

3. without parameters and with return values:

```
def addition():  
    a=4  
    b=9  
    c=a+b  
    return c  
print(addition(c))
```

4. without parameters and without return values:

```
def addition():  
    a=4  
    b=9  
    c=a+b  
    print(c)  
addition()
```

LIST

- reversing a list: 2 ways

i)

```
l=[1,2,3,4,5] print(::-1) { reversing a list
```

by slicing} ii)

```
l=[1,2,3,4,5] for i in range( len
```

```
(l)-1,-1,-1 print(l[i])
```

i) maximum:

```
l=[10,3,9,1,0,5,60]
m=l[0]
for i in l:
    if i>m:
        m=i
print(m)
```

or

```
l=[10,3,9,1,0,5,60]
m=l[0]
for i in range(len(l)):
    if l[i]>m:
        m=l[i]
print(m)
```

ii) minimum:

```
l=[10,3,9,1,0,5,60]
m=l[0]
for i in range(len(l)):
    if l[i]<m:
        m=l[i]
print(m)
```

or

```
l=[10,3,9,1,0,5,60]
```

```
m=l[0]
```

```
for i in l:
```

```
    if i<m:
```

```
        m=i
```

```
print(m)
```

-ve indexing					
6	5	4	3	2	1
10	3	0	8	11	1
0	1	2	3	4	5
+ve indexing					

```
#finding the second largest from the list
```

```
l=[10,3,9,1,0,5,60] p=set(l) #set will sort the elements in the list  
q=list(p) #as negative indexing can't be done in set we will convert it  
in print(q[-2]) #using negative indexing we will find the second largest  
numb
```

```
#finding the second smallest from the list
```

```
l=[10,3,9,1,0,5,60]  
p=set(l) #set will sort the elements in the list  
q=list(p)  
print(q[1]) #using positive indexing we will find the second largest numbe
```

```
#finding the index value of the 2 numbers which is equal to the target by adding the contents of the list:
```

```

nums=[1,4,2,6,3,5]
target=6
def addtwo(nums,target):
    a=[]
    for i in range(len(nums)):
        for j in range(i+1,len(nums)):

            if nums[i]+nums[j]==target:
                a.append(i)
                a.append(j)
    return a

print(addtwo(nums,target))

```

o/p:

¶0,5¶

#finding the index value of the 3 numbers which is equal to the target by adding the contents of the list:

```

nums=[1,4,2,6,3,5]
target=12
def addthree(nums,target):
    a=[]
    for i in range(len(nums)):
        for j in range(i+1,len(nums)):
            for k in range(j+1,len(nums)):
                if nums[i]+nums[j]+nums[k]==target:
                    a.append(i)
                    a.append(j)
                    a.append(k)
    return a

print(addthree(nums,target))

```

o/p:

¶0,3,5¶

#remove duplicates from the list(without using set)

```
l=[1,1,1,2,2,3,3,4]
m=[]
for i in l:
    if i not in m:
        m.append(i)
print(m)
```

o/p:

[1,2,3,4]

OR

```
l=[1,1,1,2,2,3,3,4]
m=[]
for i in l:
    if i in m:
        pass
    else:
        m.append(i)
print(m)
```

#remove duplicates from string

```
s="aaaaabbbcd"
s1=[]
for i in s:
    if i in s1:
        pass
    else:
        s1.append(i)
print(s1)
```

o/p:

`['a','b','c','d']`

#counting how many times each letter is repeated in a string

```
s="aaaabbbcd"
s1=set(s)
for i in s1:
    count=0
    for j in s:
        if i==j:
            count+=1
    print(i,"-",count)
```

o/p:

`d-1 c-`

`1 b-3`

`a-4`

#counting how many times each number is repeated in a string

```
l=[1,3,2,4,1,3,2,2,1,4]
l1=set(l)
for i in l1:
    c=0
    for j in l:
        if i==j:
            c+=1
    print(i,"-",c)
```

o/p:

`1@3`

`2@3`

`3@2`

`4@2`

#sum of all even and odd numbers in list

```
l=[1,2,3,4,5,6,7,8]
even=0
odd=0
for i in l:
    if i%2==0:
        even+=i
    else:
        odd+=i

print("even sum=",even)
print("odd sum=",odd)
```

even sum=20 odd

sum=16

#finding common elements from the list

```
l1=[1,3,2,4,5]
l2=[4,6,7,8,9]
l=[]
for i in l1:
    for j in l2:
        if i==j:
            l.append(i)
print(l)
```

OR

```
l1=[1,3,2,4,5]
l2=[4,6,7,8,9]
l=[]
for i in l1:
    if i in l2:
        l.append(i)
print(l)
```

```
#finding uncommon elements in the list
```

```
l1=[1,3,2,4,5]
l2=[4,6,7,8,9]
l=[]
for i in l1:
    if i not in l2:
        l.append(i)
for i in l2:
    if i not in l1:
        l.append(i)
print(l)
```

o/p:

```
[1,3,2,5,6,7,8,9]
```

```
#finding prime numbers from the list
```

```
def isprime(x):
    for i in range(2,x//2+1):
        if x%i==0:
            return 0
    else:
        return 1
l=int(input())
u=int(input())
for i in range(l,u+1):
    a=isprime(i)
    if a==1:
        print(i)
```

```
#converting decimal to binary
```

```
n=int(input())
b=""
while n>0:
    rem=n%2
    b=b+str(rem)
    n=n//2
print(b[::-1])
```

o/p:

10

1010

```
#check if a number is a perfect square
```

```
n=int(input())
i=1
while i*i<n:
    i+=1
if i*i==n:
```

```
    print("perfect")
else:
    print("not perfect square")
```

o/p:

6

not perfect square

#check if a number is a power of two

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

o/p:

5

not power of 2

8

3

#check if a number is Armstrong number

Armstrong number is a number when each digit of the number powered to the length of the number and added together gives the same number ex: 371

$3^3 + 7^3 + 1^3 = 371$

*first find the length of the number i.e. c

*a is to store the result

*the original number is stored in m as it'll be reduced in $n=n//10$

*later m is compared with a if it is equal then prints Armstrong or else not

```
n=int(input())
c=len(str(n))
```

```

a=0
m=n
while n>0:
    a=a+((n%10)**c)
    n=n//10
if a==m:
    print("armstrong")
else:
    print("not armstrong")

```

#check whether a number is a perfect number

- ▼ If sum of the divisors of the given number is less than that of the given number, then that number is called as perfect number
- ▼ If sum of the divisors of the given number is less than that of the given number, then that number is called as perfect number

```

n=int(input())
m=n
sum=0
i=1
while i<n: #for i in range(1,n):
    if n%i==0:
        sum=sum+i
    i+=1
if sum==m:
    print("perfect number")
else:
    print("not perfect number")

```

o/p:

```

6 perfect
number
8
not a perfect number

```

#check whether the number is abundant number

- ▼ If the sum of all the divisors of the given number is greater than the number itself. Then that number is called as abundant number

```

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

```

o/p: 12 abundant

number 6 not abundant

number

STRINGS:

converting a list into string:

→`split(" ")` is used to convert a string into list

→it separates the words in a string with spaces into each element in a list.

→`join()` is used to convert all the elements of the list into string.

→`var="".join(name of var to be converted)` ex1 ↴

```

s="hey you are idiot"
l=list(s.split(" "))
l=(l[::-1])
s=" ".join(l)
print(s)

```

ex2 ↴

```

s="hey you are idiot"
l=list(s.split(" "))
s=""
for i in l:
    s=s+i[::-1]+" "
print(s)
'''o/p:
yeh ouy era toidi'''

```

#checking if a word is palindrome or not

```

s="Madam"
s1=s.upper()
if s1==s1[::-1]:
    print("palindrome")
else:
    print("not palindrome")

'''o/p:
palindrome'''

```

#check whether the given word is anagram or not

```

a=input()
b=input()
s1=set(a) #to get the unique elements of a
s2=set(b)
if s1==s2:           #checks whether the unique elements are same, if they're
    for i in s1:   #to run the unique elements''
        if a.count(i)!=b.count(i):
            print("not anagram")
            break
    else:
        print("anagram")
'''o/p:
dog
god
anagram'''

```

RECURSION:

a function which calls itself repeatedly until certain condition is satisfied is called as recursive function.

This process is called as ***recursion*** difference

between recursion and loop is:

→ a loop will run infinitely

→ but by default recursion runs for 0-999 times a function

recurs based on the limit given for recurring. ex:

```
def recur(x):
```

```
    print(x) x+=1
```

```
    recur(x)
```

recur() rules for writing recursive

func:(format)

②② define function

②② base condition must be given (in 4 to 1, 1 is base condition:1 to 4, 4 is base condition ②② call the function again

```
#factorial of a number
```

```
def fact(n):
    if n==1:
        return n
    else:
        return n*fact(n-1)
x=int(input())
print(fact(x))
```

```
#reverse of a number using recursion
```

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

#Fibonacci series using recursion

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

OOPS!

Class:

```
a=[] print(type(a))
```

o/p:

```
<class 'list'>
```

- In python, everything will be viewed in the form of class and objects.
- List is stored in different locations even if the contents of list are same, because list is mutable and list is viewed as object
- And if we check the id of the contents of 2 lists then it'll be stored in the same location as the contents of list can't be changed.
- CLASS is a user-defined datatype.
- There are few in-built classes such as list, tuple and dictionary. Whenever
- we refer class, we refer collectively maybe animals ,students, places etc.
- When we refer an object, we refer a particular thing maybe a
- person's name, animals name etc.
-

Attributes and Behavior:

attributes: variables. Behavior:

functions.

collectively writing attributes and behavior forms class.

how to define class: class

class_name:

ex:

```

def __init__(self,x,y,z): { self is object, create a constructor and initialize 3 values
which will be initialized below} self.nickname=x self.rollno=y self.height=z def
run(self):
    print("i can run", self.height, self.roll) { self is object that we are passing}
harsha=person("chintu",78,6) {person() is a constructor}
anjali=person("mary",89,5.6) harsha.run() anjali.run()

```

o/p:

i can run chintu 78 i

can run mary 89

#if we don't create a constructor the contents will be common for all the objects.

constructor will initialize objects that are created.

difference between function and method:

method is also a function which is written inside a class method is accessed using objects where function is directly accessed

Abstraction:

- it's an idea which is not implemented. it is used to hide the unnecessary data and
- show only necessary data.
- in python logically abstraction, polymorphism and encapsulation does not work.

abstract method: it is a method which doesn't

contain any body.

class person:

```

def mobile():
    pass

```

```

'''abstraction'''

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

```

#polymorphism

```

class mobile: #abstract class
    def functions(self): #abstract method    pass

    def functions(self,camera,display,battery):
        self.camera=camera
        self.display=display    self.battery=battery
        print(self.camera) print(self.display)
        print(self.battery)

iphone=mobile() iphone.functions("12mp","4k","60mh")
samsung=mobile()

samsung.functions("24mp","6k","80mh") abstract class:
it is a class that contain abstract method.

```

#inheritance:

```
class mobile: #abstract class
    def functions(self): #abstract method
        pass
class iphone(mobile): '''passing the idea'''
    def functions(self):
        print("This is iphone")
class samsung(mmobile):
    def functions(self):
        print("This is samsung")
iphone13=iphone()
iphone13.functions()
```

```
samsungs3=samsung()
samsungs3.functions()
```

Encapsulation:

if anything is private we cant access them directly instead we access them using methods."

"_" =private variables. "__"=protected variables. In Python we can change the private variables using a public variable(loophole)

```
class car:
    _engine="v8"
    __wires="blue"
bmw=car()
bmw._engine="v9"
```

encapsulation can be implemented using getter and setter methods

methods are always public variables can be either private, protected or public

```

class car:
    _engine="v8"
    _wires="blue"
    def getter(self):
        print(self._engine)
        print(self._wires)
    def setter(self,engine,wires):
        self._engine=engine
        self._wires=wires
bmw=car()

```

```

bmw.setter("v9","red")
bmw.getter()

```

```

class person:
    def __init__(self,x,y,z):          #self is object, create a cons
        self.nickname=x
        self.rollno=y
        self.height=z
    def run(self):
        print("i can run",self.height, self.roll)      #{ self is o
harsha=person("chintu",78,6)           #{person()
anjali=person("mary",89,5.6)
harsha.run()
anjali.run()

```

Inheritance:

?? single

?? multiple

¶¶ multilevel

¶¶ hierarchical

¶¶ hybrid

1. single

inheriting from one single class

```
class parents:  
    def coolness(self):  
        print("parents are cool")  
  
class child(parents):  
  
    def coding(self):  
        print("i know coding")  
yashu=child()  
yashu.coolness()  
yashu.coding()
```

¶¶ multilevel

a class(child2) is inherited from another class(child) but that class(child) is inherited from its parent class(parents)

```

class parents:
    def coolness(self):
        print("parents are cool")
class child(parents):
    def coding(self):
        print("i know coding")
class child2(child):
    def singing(self):
        print("i can sing")
yashu=child2()
yashu.coolness()
yashu.coding()
yashu.singing()

```

3.multiple:

a child will inherit from 2 parents

```

class dad:
    def coolness(self):
        print("parents are cool")
class mom:
    def coding(self):
        print("i know coding")
class child(dad,mom):
    def singing(self):
        print("i can sing")
yashu=child()
yashu.coolness()
yashu.coding()
yashu.singing()

```

¶¶ hierarchical:

```

class grandfather:
    def coolness(self):
        print("parents are cool")
class father(grandfather):
    def coding(self):
        print("i know coding")
class daughter(father):
    def singing(self):
        print("i can sing")
yashu=daughter()
yashu.coolness()
yashu.coding()
yashu.singing()

```

□□ hybrid:

```

class grandfather:
    def coolness(self):
        print("i'm cool")
class father(grandfather):
    def coding(self):
        print("i know coding")
class mother(grandfather):
    def cooking(self):
        print("i can cook")

```

```

class daughter(father,mother):
    def singing(self):
        print("i can sing")
yashu=daughter()
yashu.coolness()
yashu.coding()
yashu.cooking()
yashu.singing()

```

Polymorphism:

poly: many morphism: forms **overloading**:

same name and different parameters.

```

class add:
    def sum(self,x,y):
        print(x+y)
    def sum(self,x,y,z):
        print(x+y+z)
i=add()
i.sum(10,8)
i.sum(10,8,1)

```

o/p: error

```

class add:
    def sum(self,x,y):
        print(x+y)
class child(add):
    def sum(self,x,y,z):
        print(x+y+z)
i=add()
i.sum(10,8)

```

o/p: 18

overriding:



DATA STRUCTRES!

Use:

It reduces time and space complexity.

LINEAR SEARCH

searching the value one by one. it requires more time.

```
l=[1,2,3,4,5,6,7,8,9,10]
for i in l:
    if i==7:
        print("found")
        break
else:
    print("not found")
```

BINARY SEARCH

→use 2 pointers to search

→one pointing to first number(left) - increment it i=0

#left=0

→another to the last one(right) - decrement it

*for loop : when we know how many times to run it

*while loop : we do not know how many times but know the condition

*when we use elif : only one condition will run

*when we use multiple if : all conditions will run

```

l=[1,17,3,4,10,8,12,22]
l.sort()
print(l)
s=8
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")

```

TIME COMPLEXITY

Best $O(1)$ searching in the first and getting the first value.

Average $[O(n)]$ it depends the value of n

Worst $[O(n)]$ only one loop is used

SPACE COMPLEXITY $O(1)$ memory

space will be fixed.

$O(\log(n))$:

is used when the size is decreasing by half. n is number of elements.

SORTING

the 5 sorting techniques are:

• bubble

• merge

• quick

• selection

• insert

BUBBLE SORT

best case scenario for sorting is when the list of elements are already sorted.

•

```
l=[9,7,78,10,5,1,0]
for i in range(0,len(l)-1):
    for j in range(0,len(l)-i-1):
        if l[j]>l[j+1]:
            l[j],l[j+1]=l[j+1],l[j] #swap {a,b=b,a}
print(l)
```

the greater number will move to the last

QUESTION SELECTION SORT

```
b=[2, 6, 8, 4, 19, 5, 44]
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)
```

QUESTION INSERTION SORT

→ always starts with second number while sorting:

→ the sorted array will be on left(backward)

→ unsorted array will be on right(forward)

⇒ i will be all the forward elements

⇒ j will be all the backward elements

```
b=[2, 6, 8, 4, 19, 5, 44]
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¶

- works on divide and conquer rule.
- break the given array by dividing it until we get a singular element this is done by using the condition beginning<ending (base condition)

```
#merge sort
def merge(arr,mid,beg,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[j]
        k+=1
        j+=1
```

¶¶ QUICK SORT¶

- it is the most effective sorting technique.
- follows 2 pointer approach (start and end pointers)

```

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=[8,7,6,1,4,5,2,3]
b=0
e=len(a)-1
quicksort(a,b,e)
print(a)

```

STACKS AND QUEUES

stacks and queues can be implemented using list. stacks and queues are implemented using oops concept.

for implementing stack we should create a class and inside a class constructor is used. terminologies for implementing stack:

push, pop, and peek(topmost element of stack) code for
implementing stack:

```

class stack:
    def __init__(self):
        self.top=-1 #top value can also be given using negative
        self.size=5 #just like how we give MAX in C
        self.list=[]
    def push(self,a):
        if len(self.list)>=5:
            print("list is full")
            return 0
        self.top+=1
        self.list.append(a)
    def pop(self):
        if len(self.list)==0:
            print("list is empty")
            return 0
        self.top-=1
    def peek(self):
        print(self.list)
        if len(self.list)==0:
            print("list is 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.peek()
s.pop()
s.peek()

```

```
s.pop()  
s.peek()
```

QUEUES?

```
class queue:  
    def __init__(self):  
        self.front=-1 #top value can also be given using negative  
        self.rear=-1  
        self.size=5 #just like how we give MAX in C.  
        self.list=[]  
    def enqueue(self,a):  
        if len(self.list)>=5:  
            print("list is full")  
            return 0  
        self.rear+=1  
        self.list.append(a)  
        if self.front== -1:  
            self.front=0  
    def dequeue(self):  
        if len(self.list)==0:  
            print("list is empty")  
            return 0  
        elif self.front>self.rear:  
            print("list is empty")  
        else:  
            self.list.pop(0)  
            self.front+=1  
            print(self.list)  
    def display(self):  
        print(self.list)  
        if len(self.list)==0:  
            print("list is empty")  
            return 0
```

```
s=queue()
s.enqueue(1)
s.enqueue(2)
s.enqueue(3)
s.enqueue(4)
s.enqueue(5)
s.display()
s.dequeue()
s.dequeue()
s.dequeue()
s.dequeue()
s.dequeue()
```

#evaluation of postfix expression:

```
s="5678+-*"
s1=[]
for i in s:
    if i.isdigit():
        s1.append(int(i))
    else:
        op2=s1.pop()
        op1=s1.pop()
        if i=="+":
            s1.append(op1+op2)
        elif i=="-":
            s1.append(op2-op1)
        elif i=="*":
            s1.append(op1*op2)
        elif i=="/":
            s1.append(op1/op2)
print(s1)
```

#Implement queue using stack:

in queue we delete elements from 0th index but in stack we delete from the top.

```

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()
pop()
print(l)

```

OR

```

class MyQueue:
    def __init__(self):
        self.s1=[]
        self.s2=[]
    def push(self, x: int) -> None:
        self.s1.append(x)
    def pop(self) -> int:
        for i in range(len(self.s1)):
            self.s2.append(self.s1.pop())
        a=self.s2.pop()
        for i in range(len(self.s2)):
            self.s1.append(self.s2.pop())
        return a
    def peek(self) -> int:
        return self.s1[0]
    def empty(self) -> bool:
        return len(self.s1)==0

```

LINKED LIST:

Linked lists: why linked list when there is set, list etc.?

② time complexity

② space complexity linked list is represented by nodes node contains 2 sections:

② data section

② address section(address of next node)

#insert beginning and delete beginning

```
class Node:  
    def __init__(self,v):  
        self.data=v  
        self.next=None #NULL  
        #make sure u understand each and every line sowmyaa!!!!  
class linkedlist:  
    def __init__(self):  
        self.head=None  
    def insertbeg(self,v):  
        nn=Node(v)  
        if self.head==None:  
            self.head=nn  
        else:  
            nn.next=self.head  
            self.head=nn  
    def delbeg(self):  
        if self.head==None:  
            print("list empty")
```

```

        else:
            temp=self.head
            self.head=temp.next
    def display(self):
        curr=self.head
        while curr!=None:
            print(curr.data)
            curr=curr.next
        print("null")
l=linkedlist()
l.insertbeg(10)
l.insertbeg(2)
l.insertbeg(8)
l.display()
l.delbeg()
l.display()

```

#insert end and delete end

```

class Node:
    def __init__(self,v):
        self.data=v
        self.next=None #NULL
        #make sure u understand each and every line sowmyaa!!!!
class linkedlist:
    def __init__(self):
        self.head=None
    def insertend(self,v):
        nn=Node(v)
        if self.head==None:
            self.head=nn
        else:
            curr=self.head
            while curr.next!=None:
                curr=curr.next
            curr.next=nn

```

```

def delend(self):
    if self.head==None:
        print("list empty")
    else:
        curr=prev=self.head
        while curr.next!=None:
            prev=curr
            curr=curr.next
        prev.next=None
def display(self):
    curr=self.head
    while curr!=None:
        print(curr.data,"->",end=" ")
        curr=curr.next
    print("null")
l=linkedlist()
l.insertend(10)
l.insertend(2)
l.insertend(8)
l.insertend(11)
l.display()
l.delend()
l.display()

```

#insert anywhere and delete anywhere

```

class Node:
    def __init__(self,v):
        self.data=v
        self.next=None #NULL
        #make sure u understand each and every line sowmyaa!!!!
class linkedlist:
    def __init__(self):
        self.head=None
    def insertend(self,v):
        nn=Node(v)

```



```

        if self.head==None:
self.head=nn           else:
                curr=self.head
while curr.next!=None:
                curr=curr.next
curr.next=nn           def
insertany(self,v,k):
    nn=Node(v)
if self.head==None:
    print("list is empty")
else:
    curr=self.head
while curr!=None:           if
curr.data==k:
    nn.next=curr.next
    curr.next=nn
    break           curr=curr.next
def search(self,k):
    curr=self.head           while
    curr!=None:           if
    curr.data==k:
        print("found")           break
    curr=curr.next           else:
        print("not found")
def delany(self,k):           if
self.head==None:
    print("list empty")
else:
    curr=prev=self.head
while curr!=None:
    if curr.data==k:
        prev.next=curr.next

```

```

        break
    prev=curr
    curr=curr.next
else:
    print("key not found")
def display(self):
    curr=self.head
    while curr!=None:
        print(curr.data,"->",end=" ")
        curr=curr.next
    print("null")
l=linkedlist()
l.insertend(10)
l.insertend(2)
l.insertend(8)
l.insertend(11)
l.insertany(17,2)
l.display()
l.search(8)
l.delany(8)
l.display()

```

#insert at middle of the list using count

```

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

```

```

        newnode.next=self.head
        self.head=newnode

    def insertmid(self,value):
        newnode=Node (value)
        c=0
        if self.head==None:
            self.head=newnode
        elif self.head.next==None:
            self.head.next=newnode
        else:
            curr=self.head
            while curr!=None:
                c+=1
                curr=curr.next
            curr=self.head
            for i in range(c//2):
                curr=curr.next
            newnode.next=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.insertatbeg(1)
l.insertatbeg(2)
l.insertatbeg(3)
l.insertatbeg(8)
l.insertmid(4)
l.printlist()

```

```
#insert at middle of the list without using count
```

```
def insertmid(self, val):  
    newnode=node(val)  
    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
```

```
#reverse of a linked list
```

Trees:

```
#creating a node in a normal tree

class node:
    def __init__(self,v):
        self.left=None
        self.right=None
        self.data=v

def inorder(root):
    if root:
        inorder(root.left)
        print(root.data)
        inorder(root.right)

r=node(1)
r.left=node(2)
r.right=node(3)
r.left.left=node(4)
r.left.right=node(5)
inorder(r)
```

we create a function to print the contents of the tree instead of method because:

since there is no root inside node class, for traversing we need to pass root as parameter
hence we create a function for traversing not a method.

Binary search tree:

```

class node:
    def __init__(self,v):
        self.left=None
        self.right=None
        self.data=v

class trees: #in trees root is the only variable we can have
    def __init__(self):
        self.root=None
    def insert(self,value):
        nn=node(value)
        if self.root is None:
            self.root=nn
        else:
            curr=self.root
            while True:
                if value<=curr.data:
                    if curr.left==None:
                        curr.left=nn
                        break
                    else:
                        curr=curr.left
                else:
                    curr=curr.right

```

```

        if curr.right==None:
            curr.right=nn
            break
        else:
            curr=curr.right

    def preorder(self,root):
        if root:
            print(root.data)
            self.preorder(root.left)
            self.preorder(root.right)

    def inorder(root):
        if root:
            inorder(root.left)
            print(root.data)
            inorder(root.right)

    def postorder(root):
        if root:
            postorder(root.left)
            postorder(root.right)
            print(root.data)

r=trees()
r.insert(2)
r.insert(8)
r.insert(1)
r.preorder(r.root)
print(" ")
inorder(r.root)
print(" ")
postorder(r.root)

```

Graphs:

It is a non primitive no linear data structure which contains edges and vertices.

Applications of graphs:

maps

?? social media apps

#adjacency matrix

```
class Graph:  
    def __init__(self):  
        self.matrix=[[0]*5 for i in range(5)]  
        print(self.matrix)  
    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()
```

O/P?

#adjacency list using dictionary

```
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()

```

O/P:

1 2, 4 2, 4 2, 3 2, 2 3 3

#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):
        v=[]
        q=[data]
        while q:
            vertex=q.pop(0)
            print(vertex)
            if vertex in self.matrix:

```

```

        for i in self.matrix[vertex]:
            if i not in v:
                v.append(i)
                q.append(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()
g.bfs(1)

```

#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):
        v=[]
        q=[data]
        while q:
            vertex=q.pop()
            print(vertex)
            if vertex in self.matrix:
                for i in self.matrix[vertex]:
                    if i not in v:
                        v.append(i)
                        q.append(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()
g.dfs(1)
```

LEETCODE PROBLEMS:

810. Faulty Keyboard

```
class Solution:
    def finalString(self, s: str) -> str:
        l=""
        for i in s:
            if i!="i":
                l+=i
            else:
                l=l[::-1]
        return l
```

832. Check the given string is PANGRAM or not

```
class Solution:
    def checkIfPangram(self, sentence: str) -> bool:
        a="abcdefghijklmnopqrstuvwxyz"
        for i in a:
            if i not in sentence:
                return False
        else:
            return True
```

557. Reverse words in a string III

```
class Solution:
    def reverseWords(self, s: str) -> str:
        a=""
```

```

l=s.split(" ")
for i in range(len(l)):
    if i==len(l)-1:
        a=a+l[i][::-1]
    else:
        a=a+l[i][::-1]+" "
return a

```

828. Check if a String is an Acronym or not

```

class Solution:

    def isAcronym(self, words: List[str], s: str) -> bool:
        if len(words)!=len(s):
            return False
        else:
            for i in range(len(s)):
                if s[i]!=words[i][0]:
                    return False
            return True

```

804. Unique Morse Code Words

```

class Solution:

    def uniqueMorseRepresentations(self, words: List[str]) -> int:
        n=[".-","-...","-.-.","-..",".","..-.","--.","....",".."
a="abcdefghijklmnopqrstuvwxyz"
p=[]
ans=[]
for q in words:
    b=""
    for i in q:
        b+=n[a.index(i)]
    p.append(b)

```

```
for q in p:  
    if q not in ans:  
        ans.append(q)  
return len(ans)
```

315. Count Asterisks

```
class Solution:  
    def countAsterisks(self, s: str) -> int:  
        n=""  
        c=0  
        t=0  
        for i in s:  
            if i=="|":  
                t=t+1  
            elif t%2==0:  
                c+= i=="*"  
        return c
```

744. Maximum number of String pairs

```
class Solution:  
    def maximumNumberOfStringPairs(self, words: List[str]) -> int:  
        c=0  
        for i in range(len(words)):  
            for j in range(i+1, len(words)):  
                if words[i][0]==words[j][1] and words[i][1]==words[j][0]:  
                    c+=1  
        return c
```

967. Number of strings that appear as substrings

```

class Solution:
    def maximumNumberOfStringPairs(self, words: List[str]) -> int:
        c=0
        for i in range(len(words)):
            for j in range(i+1, len(words)):
                if words[i][0]==words[j][1] and words[i][1]==words[j][0]:
                    c+=1
        return c

```

019. Count the key Changes

```

class Solution:
    def countKeyChanges(self, s: str) -> int:
        n=s.lower()
        c=0
        for i in range(len(n)-1):
            if n[i]!=n[i+1]:
                c+=1
        return c

```

344. Reverse String

```

class Solution:
    def reverseString(self, s: List[str]) -> None:
        s[::-1]=s[::-1]

```

110. Score of a String

```

class Solution:
    def scoreOfString(self, s: str) -> int:
        a=0
        for i in range(1,len(s)):
            a+=abs(ord(s[i])-ord(s[i-1]))
        return a

```

418. Sort the People

```

class Solution:
    def sortPeople(self, names: List[str], heights: List[int]) -
        a=[]
        for i in range(len(names)):
            a.append([heights[i],names[i]])
        a.sort(reverse=True)
        b=[]
        for i in range(len(names)):
            b.append(a[i][1])
        return b

```

1. Two Sum

```

class Solution:
    def twoSum(self, nums, target):
        a=[]
        for i in range(len(nums)):
            for j in range(i+1,len(nums)):
                if nums[i]+nums[j]==target:
                    a.append(i)
                    a.append(j)

```

```
        return a
    return(twosum(nums,target))
```

58. Length of Last Word

```
class Solution:
    def lengthOfLastWord(self, s: str) -> int:
        l=s.split()
        a=len(l[-1])
        return a
```

859. Sorting the sentence

```
class Solution:
    def sortSentence(self, s: str) -> str:
        l=s.split(" ")
        d={}
        for i in l:
            d[i[-1]]=i[:-1]
        s=""
        for i in range(1,len(l)+1):
            if i==len(l):
                s=s+d[str(i)]
            else:
                s=s+d[str(i)]+" "
        return s
```

20. Valid Parenthesis

```
class Solution:
    def isValid(self, s: str) -> bool:
```

```

a=[]
for i in s:
    if i=="(" or i=="{" or i=="[":
        a.append(i)
    elif (i==")" or i=="}" or i=="]") and len(a)==0:
        return False
    else:
        if i==")" and a[-1]=="(":
            a.pop()
        elif i=="}" and a[-1]=="{":
            a.pop()
        elif i=="]" and a[-1]=="[":
            a.pop()
        else:
            a.append(i)
if len(a)==0:
    return True
else:
    return False

```

929. Concatenation of Array

```

class Solution:
    def getConcatenation(self, nums: List[int]) -> List[int]:
        return 2*nums

```

470. Shuffle the Array

```

class Solution:
    def shuffle(self, nums: List[int], n: int) -> List[int]:
        n=[]
        h=len(nums)//2

```

```
for i in range(len(nums)//2):
    n.append(nums[i])
    n.append(nums[i + h])
return n
```

672. Richest Customer Wealth

```
class Solution:
    def maximumWealth(self, accounts: List[List[int]]) -> int:
        maxw=0
        for i in range(len(accounts)):
            t=sum(accounts[i])
            maxw=max(maxw, t)
        return maxw
```

798. Number of employees who met the target

```
class Solution:
    def numberOfEmployeesWhoMetTarget(self, hours: List[int], ta
        c=0
        for i in hours:
            if i>=target:
                c+=1
        return c
```

431. Kids with Greatest number of candies

```
class Solution:
    def kidsWithCandies(self, candies: List[int], extraCandies:
        maximum=max(candies)
        r=[ ]
```

```

for i in range(len(candies)):
    t=candies[i]+extraCandies
    r.append(t>=maximum)
return r

```

824. Count pairs whose sum is less than target

```

class Solution:
    def countPairs(self, nums: List[int], target: int) -> int:
        c=0
        for i in range(len(nums)):
            for j in range(i+1,len(nums)):
                if nums[i]+nums[j]<target:
                    c+=1
        return c

```

480. Running Sum of 1D Array

```

class Solution:
    def runningSum(self, nums: List[int]) -> List[int]:
        for i in range(1,len(nums)):
            nums[i]+=nums[i-1]
        return nums

```

974. Minimum number game

```

class Solution:
    def numberGame(self, nums: List[int]) -> List[int]:
        nums=sorted(nums)
        for i in range(0,len(nums),2):

```

```
        nums[i], nums[i+1]=nums[i+1], nums[i]
    return nums
```

588. Sum of Odd Length Subarrays

```
class Solution:
    def sumOddLengthSubarrays(self, arr: List[int]) -> int:
        r=0
        for i in range(len(arr)):
            for j in range(i,len(arr),2):
                r+=sum(arr[i:j+1])
        return r
```

232. Implement Queue using Stack

```
class MyQueue:

    def __init__(self):
        self.s1=[]
        self.s2=[]

    def push(self, x: int) -> None:
        self.s1.append(x)

    def pop(self) -> int:
        for i in range(len(self.s1)):
            self.s2.append(self.s1.pop())
        a= self.s2.pop()
        for i in range(len(self.s2)):
            self.s1.append(self.s2.pop())
        return a
```

```

def peek(self) -> int:
    return self.s1[0]

def empty(self) -> bool:
    if len(self.s1)==0:
        return True
    else:
        return False

```

225. Implement of Stack using Queues

```

class MyStack:

    def __init__(self):
        self.q1=[]
        self.q2=[]

    def push(self, x: int) -> None:
        self.q1.append(x)

    def pop(self) -> int:
        for i in range(len(self.q1)):
            self.q2.append(self.q1.pop())
        a=self.q2.pop(0)
        for i in range(len(self.q2)):
            self.q1.append(self.q2.pop())
        return a

    def top(self):
        return self.q1[-1]

    def empty(self):
        if len(self.q1)==0:

```

```
        return True
else:
    return False
```

26. Remove Duplicates from the Sorted Array

```
class Solution:
    def removeDuplicates(self, nums: List[int]) -> int:
        s=set(nums)
        l=list(s)
        l.sort()
        for i in range(len(l)):
            nums[i]=l[i]
        return len(l)
```

58. Length of Last Word

```
class Solution:
    def lengthOfLastWord(self, s: str) -> int:
        c=0
        i=len(s)-1
        while i>=0 and s[i]==' ':
            i-=1
        while i>=0 and s[i]!=' ':
            c+=1
            i-=1
        return c
```

242. Valid Anagram

```
class Solution:
    def isAnagram(self, s: str, t: str) -> bool:
        sorted_s=sorted(s)
        sorted_t=sorted(t)
        return sorted_s==sorted_t
```

258. Add Digits

```
class Solution:
    def addDigits(self, num: int) -> int:
        if num>=10:
            output=num//10+num%10
            if output<10:
                return output
            else:
                return self.addDigits(output)
        else:
            return num
```

876. Middle of the Linked List

```
# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, val=0, next=None):
#         self.val = val
#         self.next = next
class Solution:
    def middleNode(self, head: Optional[ListNode]) -> Optional[L
        temp=head
        c=0
        while temp!=None:
            c+=1
```

```

        temp=temp.next
    temp=head
    for i in range(c//2):
        temp=temp.next
    return temp

```

206. Reverse Linked List

```

# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, val=0, next=None):
#         self.val = val
#         self.next = next
class Solution:

    def reverseList(self, head: Optional[ListNode]) -> Optional[ListNode]:
        prev=None
        curr=head
        while curr!=None:
            next=curr.next
            curr.next=prev
            prev=curr
            curr=next
        head=prev
        return head

```

21. Merge Two Sorted Lists

```

# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, val=0, next=None):
#         self.val = val
#         self.next = next

```



```

class Solution:
    def mergeTwoLists(self, list1: Optional[ListNode], list2: Optional[ListNode]) -> Optional[ListNode]:
        if list1 == None:
            return list2
        if list2 == None:
            return list1
        if list1.val <= list2.val:
            list1.next = self.mergeTwoLists(list1.next, list2)
            return list1
        else:
            list2.next = self.mergeTwoLists(list1, list2.next)
            return list2

```

```

while curr2!=None:
    newnode=ListNode(curr2.val)
    temp=curr3
    if temp==None:
        curr3=newnode
    else:
        while temp.next!=None:
            temp=temp.next
        temp.next=newnode
    curr2=curr2.next
return curr3

```

160. Intersection Of Two Linked Lists

```

# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, x):
#         self.val = x
#         self.next = None

class Solution:
    def getIntersectionNode(self, headA: ListNode, headB: ListNode):
        a=headA
        b=headB
        c=0
        while a!=b:
            a=a.next
            b=b.next
            if a==None:
                a=headB
                c+=1
            if b==None:
                b=headA

```

```

        c+=1
    if c>=3:
        return None
    return b

```

234. Palindrome Linked List

```

# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, val=0, next=None):
#         self.val = val
#         self.next = next
class Solution:
    def isPalindrome(self, head: Optional[ListNode]) -> bool:
        curr=head
        newnode=ListNode(curr.val)
        a=newnode
        curr=curr.next
        while curr!=None:
            new=ListNode(curr.val)
            a.next=new
            curr=curr.next
            a=a.next
        prev=None
        next=None
        curr=head
        while curr!=None:
            next=curr.next
            curr.next=prev
            prev=curr
            curr=next
        head=prev

        curr=head

```

```

a=newnode
while curr!=None:
    if curr.val!=a.val:
        return False
    curr=curr.next
    a=a.next
else:
    return True

```

83. Remove Duplicates From the List

```

# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, val=0, next=None):
#         self.val = val
#         self.next = next
class Solution:

    def deleteDuplicates(self, head: Optional[ListNode]) -> Optional[ListNode]:
        curr=head
        next=None
        if head==None:
            return None
        while curr.next!=None:
            next=curr.next
            if curr.val!=next.val:
                curr.next=next.next
            else:
                curr.next=next.next
        return head

```

141. Linked List Cycle

```

# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, x):
#         self.val = x
#         self.next = None

class Solution:
    def hasCycle(self, head: Optional[ListNode]) -> bool:
        if head==None:
            return False
        fast=head
        slow=head
        while fast.next!=None and fast.next.next!=None:
            fast=fast.next.next
            slow=slow.next
            if fast==slow:
                return True
        return False

```

203. Remove Linked List Elements

```

# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, val=0, next=None):
#         self.val = val
#         self.next = next

class Solution:
    def removeElements(self, head: Optional[ListNode], val: int):
        temp=ListNode
        curr=temp
        temp.next=head
        while curr.next!=None:
            if curr.next.val==val:

```

```

        curr.next=curr.next.next
    else:
        curr=curr.next
    return temp.next

```

237. Delete Node in a Linked List

```

# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, x):
#         self.val = x
#         self.next = None

class Solution:
    def deleteNode(self, node):
        """
        :type node: ListNode
        :rtype: void Do not return anything, modify node in-place
        """
        node.val=node.next.val
        node.next=node.next.next

```

2. Add Two Numbers

```

# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, val=0, next=None):
#         self.val = val
#         self.next = next
class Solution:
    def addTwoNumbers(self, l1: Optional[ListNode], l2: Optional

```

```

s=""
s2=""

while l1!=None:
    s=s+str(l1.val)
    l1=l1.next

while l2!=None:
    s2=s2+str(l2.val)
    l2=l2.next

s=s[::-1]
s2=s2[::-1]
a=int(s)+int(s2)
a=str(a)
a=a[::-1]
q=str(a)
z=ListNode()
curr=z

for i in q:
    new=ListNode(int(i))
    curr.next=new
    curr=curr.next

return z.next

```

145. Binary Tree Postorder Traversal

```

# Definition for a binary tree node.
# class TreeNode:
#     def __init__(self, val=0, left=None, right=None):
#         self.val = val
#         self.left = left
#         self.right = right
class Solution:

    def postorderTraversal(self, root: Optional[TreeNode]) -> List[int]:
        s=[]
        def order(root,s):

```

```

        if root:
            order(root.left,s)
            order(root.right,s)
            s.append(root.val)
    order(root,s)
    return s

```

94. Binary Tree Preorder Traversal

```

# Definition for a binary tree node.
# class TreeNode:
#     def __init__(self, val=0, left=None, right=None):
#         self.val = val
#         self.left = left
#         self.right = right
class Solution:
    def preorderTraversal(self, root: Optional[TreeNode]) -> List[int]:
        s=[]
        def order(root,s):
            if root:
                s.append(root.val)
                order(root.left,s)
                order(root.right,s)
        order(root,s)
        return s

```

144. Binary Tree Inorder Traversal

```

# Definition for a binary tree node.
# class TreeNode:
#     def __init__(self, val=0, left=None, right=None):
#         self.val = val
#         self.left = left

```

```

#           self.right = right
class Solution:
    def inorderTraversal(self, root: Optional[TreeNode]) -> List[int]:
        s = []
        def order(root, s):
            if root:
                order(root.left, s)
                s.append(root.val)
                order(root.right, s)

        order(root, s)
        return s

```

104. Maximum Depth of Binary Tree

```

class Solution:
    def maxDepth(self, root: Optional[TreeNode]) -> int:
        def height(root):
            if root:
                leftnode=height(root.left)
                rightnode=height(root.right)
                return max(leftnode,rightnode)+1
            else:
                return 0
        a=height(root)
        return a

```

111. Minimum Depth of the Binary Tree

```

# Definition for a binary tree node.
# class TreeNode:
#     def __init__(self, val=0, left=None, right=None):
#         self.val = val

```

```

#           self.left = left
#           self.right = right
class Solution:
    def minDepth(self, root: Optional[TreeNode]) -> int:
        def height(root):
            if root:
                if root.left is None:
                    return height(root.right)+1
                if root.right is None:
                    return height(root.left)+1
                leftnode=height(root.left)
                rightnode=height(root.right)
                return min(leftnode,rightnode)+1
            else:
                return 0
        a=height(root)
        return a

```

100. Same Tree

```

# Definition for a binary tree node.
# class TreeNode:
#     def __init__(self, val=0, left=None, right=None):
#         self.val = val
#         self.left = left
#         self.right = right
class Solution:
    def isSameTree(self, p: Optional[TreeNode], q: Optional[TreeNode]):
        def same(p,q):
            if p is None and q is None:
                return True
            if p is None or q is None:
                return False

```

```

        if p.val!=q.val:
            return False
        return same(p.left,q.left) and same(p.right,q.right)
    return same(p,q)

```

101. Symmetric Tree

```

# Definition for a binary tree node.
# class TreeNode:
#     def __init__(self, val=0, left=None, right=None):
#         self.val = val
#         self.left = left #             self.right = right
class Solution:     def isSymmetric(self, root:
Optional[TreeNode]) -> bool:           def same(p,q):
if p is None and q is None:
    return True
if p is None or q is None:
    return False
if p.val!=q.val:
    return False
return (p.val==q.val) and same(p.left,q.right) a         return
same(root.left,root.right)

```

108. Convert Sorted Array to Binary Search Tree

222. Count Complete Tree Nodes

```

# Definition for a binary tree node.
# class TreeNode:
#     def __init__(self, val=0, left=None, right=None):
#         self.val = val

```

```

#           self.left = left
#           self.right = right
class Solution:
    def countNodes(self, root: Optional[TreeNode]) -> int:
        s=[]
        def order(root,s):
            if root:
                order(root.left,s)
                s.append(root.val)
                order(root.right,s)
        order(root,s)
        return len(s)

```

27. Remove Elements

```

class Solution:
    def removeElement(self, nums: List[int], val: int) -> int:
        n=0
        for i in range(len(nums)):
            if nums[i]!=val:
                nums[n]=nums[i]
                n+=1
        return n

```

28. Find the Index of the First Occurrence in a String

```

class Solution:
    def strStr(self, haystack: str, needle: str) -> int:
        for i in range(len(haystack)-len(needle)+1):
            if haystack[i:i+len(needle)]==needle:
                return i
        return -1

```

283. Move Zeroes

```
class Solution:
    def moveZeroes(self, nums: List[int]) -> None:
        left=0
        for right in range(len(nums)):
            if nums[right]!=0:
                nums[right],nums[left]=nums[left],nums[right]
                left+=1
        return nums
```

268. Missing Number

```
class Solution:
    def missingNumber(self, nums: List[int]) -> int:
        n=len(nums)
        expected_s=n*(n+1)//2
        actual_s=sum(nums)
        missing_num=expected_s-actual_s
        return missing_num
```

748. Sum of Unique Elements

```
class Solution:
    def sumOfUnique(self, nums: List[int]) -> int:
        c=[]
        for i in nums:
            if nums.count(i)>1:
                continue
            else:
```

```
        c.append(i)
    return sum(c)
```

351. Count Negative Numbers in a Sorted Matrix

```
class Solution:
    def countNegatives(self, grid: List[List[int]]) -> int:
        c=0
        for i in grid:
            for j in i:
                if j<0:
                    c+=1
        return c
```

716. Minimum String Length

```
class Solution:
    def minimizedStringLength(self, s: str) -> int:
        a=""
        for i in s:
            if(i not in a):
                a+=i
        return len(a)
```