syllabus:

01. introduction to DSA
02. python basics (20 programs)
03. python inbuilt data structures (str, list, tuple, set and dict) (30 programs)
04. sample algorithms and implementation (30 programs)
05. array data structure
06. programs on array data structure
07. string data structures and programs
08. recursion and its application
09. backtracking
10. sorting
11. searching
12. divide and conquer algorithms (mergesort and quick sort)
13. list data structure (SLL, DLL, CSLL, CDLL)
14. stack data structures
15. queue data structures
16. hashtable data structure
17. tree data structures
18. priority queues or heaps
19. graph data structure
20. dynamic programming
21. greedy methods
22. complexities (time and space)
23. bitmanipulations

Algorithm:

step by step process for solving any problem is called as an algorithm.

Ex: addition of three numbers

```
Alg:
      step1: read 'a' value from the user
      step2: read 'b' value from the user
      step3: read 'c' value from the user
      step4: calculate sum = a+b+c
      step5: print/return the result sum
```

Flowchart:

diagrametic representation or pictorial representation of an alg is called as flow chart.

Implementation:

```
a=int(input("Enter a value: "))
b=int(input("Enter b value: "))
c=int(input("Enter c value: "))
sum=a+b+c
print(f"sum = {sum}")
```

```
C:\8pm>py test.py
Enter a value: 10
Enter b value: 20
Enter c value: 30
sum = 60
```

advantages of algorithm/flowchart

1) problem will be simplified.
2) easy to understand problem statement.

3) easy to implement
4) we will get a format/template/pattern to solve the problem.

properties of algorithm:

1) zero or more inputs.
2) one or more outpus (atleast one output should be there).
3) deterministic (same output for same input again again).
4) correct
5) terminate at finate steps (base condition)
6) efficient (logic should be clear)

Complexity:

complexity of an algorithm is the amount of time or space required by the algorithm or program to the process the inputs and produce output.

1) time complexity
2) space complexity

time complexity

The amount of time taken by the algorithm to process the inputs is called as time complexity, which is measured by using T(n).

space complexity

The amount of space taken by the algorithm to process the inputs is called as space complexity, which is measured by using S(n).

Asymptotic notations:

Big-Oh notation: O(n)
Omega notation : W(n)
Theta notation : 0(n)

All these programs or algorithms are classified into three types

1) worst case complexity ****
2) average case complexity
3) best case complexity

O(1)	constant time
O(n)	linear time
O(logn)	logarithmic time
O(nlogn)	logarithmic time
O(n^2)	quadratic time
O(2^n)	exponential time
O(n!)	factorial time etc

Ex1:

```
def fun(n):
      c=0
      i=0
      while i<n:
            c=c+1
            i=i+1
      return c

print("N=100, number of instructions in O(n): ",fun(100))
```

```
C:\8pm>py test.py
N=100, number of instructions in O(n):  100
```

```
complexity: O(n)

Ex2:
----
def fun(n):
      c=0
      i=0
      while i<n:
            j=0
            while j<n:
                  c=c+1
                  j=j+1
            i=i+1
      return c

print("N=100, number of instructions in O(n^2): ",fun(100))

complexity: O(n^2)

Ex3:
----
#half iterations
def fun(n):
      c=0
      i=n
      while i>0:
            c=c+1
            i=i//2
      return c

print("N=100, number of instructions in O(n^2): ",fun(100)) #7


complexity: O(logn)

Ex1: WPP to read a string and convert all even indexed values into upper case.
------------------------------------------------------------------------------
abc ---> AbC
abcd --> AbCd
abcde--> AbCdE

abc
s[0] = a
s[1] = b
s[2] = c

def myfun(s):
      l = list(s.lower())
      for i in range(len(l)):
            if i%2==0:
                  l[i] = l[i].upper()
      return ''.join(l)

s = "prakash BaBu"
print(s) #prakash BaBu
print(myfun(s)) #PrAkAsH BaBu

Ex2: WPP to read a string and convert all odd indexed values into upper case.
------------------------------------------------------------------------------
abc ---> aBc
abcd --> aBcD
abcde--> aBcDe
```

```
abc
s[0] = a
s[1] = b
s[2] = c

def myfun(s):
      l = list(s.lower())
      for i in range(len(l)):
            if i%2!=0:
                  l[i] = l[i].upper()
      return ''.join(l)

s = "prakash BaBu"
print(s) #prakash BaBu
print(myfun(s)) #pRaKaSh bAbU

Ex3: WPP to find sum of all elements present in a list
------------------------------------------------------
import functools

def fun_version1(L):
      s=0
      for i in L:
            s=s+i
      return s

def fun_version2(L):
      return sum(L)

def fun_version3(L):
      return functools.reduce(lambda i,j:i+j,L)

L = [11,22,33,44,55]
print(fun_version1(L)) #11+22+33+44+55=165
print(fun_version2(L)) #11+22+33+44+55=165
print(fun_version3(L)) #11+22+33+44+55=165

Ex4: WPP to find max of two numbers
-----------------------------------
def maxfun_version1(a,b):
      return max(a,b)

def maxfun_version2(a,b):
      return a if a>b else b

def maxfun_version3(a,b):
      if a>b:
            return a
      else:
            return b
def maxfun_version4(a,b):
      call = lambda a,b: a if a>b else b
      return call(a,b)

def maxfun_version5(a,b):
      L=[]
      L.append(a)
      L.append(b)
      return max(L)

a = int(input())
b = int(input())
print("max value by using version1:",maxfun_version1(a,b))
print("max value by using version2:",maxfun_version2(a,b))
print("max value by using version3:",maxfun_version3(a,b))
print("max value by using version4:",maxfun_version4(a,b))
print("max value by using version5:",maxfun_version5(a,b))
```

```
C:\dsapb>py test.py
1
2
max value by using version1: 2
max value by using version2: 2
max value by using version3: 2
max value by using version4: 2
max value by using version5: 2
```

```
C:\dsapb>py test.py
1
-2
max value by using version1: 1
max value by using version2: 1
max value by using version3: 1
max value by using version4: 1
max value by using version5: 1
```

case1: insert a new node at the begining of single linked list

```
def add_first(self,value):
      newnode = self.node(value,None)
      if self.head==None:
            self.head = newnode
            return
      newnode.next = self.head
      self.head = newnode
```

case2: insert a new node at the end of single linked list

```
def add_first(self,value):
      newnode = self.node(value,None)
      if self.head==None:
            self.head = newnode
            return
      temp = self.head
      while temp.next != None:
            temp = temp.next
      temp.next = newnode
```

case3: traverse or display single linked list

```
def display():
      temp = self.head
      if temp==None:
            print("SLL is empty")
            return
      while temp!=None:
            print(temp.data)
            temp = temp.next
```

Ex5: WPP to find min of two numbers

```
def minfun_version1(a,b):
      return min(a,b)

def minfun_version2(a,b):
      return a if a<b else b

def minfun_version3(a,b):
      if a<b:
            return a
      else:
            return b
def minfun_version4(a,b):
      call = lambda a,b: a if a<b else b
      return call(a,b)

def minfun_version5(a,b):
      L=[]
      L.append(a)
      L.append(b)
      return min(L)

a = int(input())
b = int(input())
print("min value by using version1:",minfun_version1(a,b))
print("min value by using version2:",minfun_version2(a,b))
print("min value by using version3:",minfun_version3(a,b))
print("min value by using version4:",minfun_version4(a,b))
print("min value by using version5:",minfun_version5(a,b))
```

```
C:\8pm>py test.py
10
20
min value by using version1: 10
min value by using version2: 10
min value by using version3: 10
min value by using version4: 10
min value by using version5: 10
```

```
C:\8pm>py test.py
10
-20
min value by using version1: -20
min value by using version2: -20
min value by using version3: -20
min value by using version4: -20
min value by using version5: -20
```

Ex6: WPP to find max of three numbers

```
#Ex6: WPP to find max of three numbers

def maxfun_version1(a,b,c):
      return max(a,b,c)

def maxfun_version2(a,b,c):
      return a if a>b and a>c else b if b>c else c

def maxfun_version3(a,b,c):
      if a>b and a>c:
            return a
      elif b>c:
            return b
      else:
            return c
def maxfun_version4(a,b,c):
      call = lambda a,b,c: a if a>b and a>c else b if b>c else c
      return call(a,b,c)

def maxfun_version5(a,b,c):
      L=[]
      L.append(a)
      L.append(b)
      L.append(c)
      return max(L)

a = int(input())
b = int(input())
c = int(input())
print("max value by using version1:",maxfun_version1(a,b,c))
print("max value by using version2:",maxfun_version2(a,b,c))
print("max value by using version3:",maxfun_version3(a,b,c))
print("max value by using version4:",maxfun_version4(a,b,c))
print("max value by using version5:",maxfun_version5(a,b,c))

C:\8pm>py test.py
1
2
3
max value by using version1: 3
max value by using version2: 3
max value by using version3: 3
max value by using version4: 3
max value by using version5: 3

C:\8pm>py test.py
1
2
-3
max value by using version1: 2
max value by using version2: 2
max value by using version3: 2
max value by using version4: 2
max value by using version5: 2

C:\8pm>py test.py
1
-2
-3
max value by using version1: 1
max value by using version2: 1
max value by using version3: 1
max value by using version4: 1
max value by using version5: 1

Ex7: WPP to find min of three numbers
Ex8: WPP to find max of four numbers
------------------------------------
#Ex8: WPP to find max of four numbers

def maxfun_version1(a,b,c,d):
      return max(a,b,c,d)

def maxfun_version2(a,b,c,d):
      return a if a>b and a>c and a>d else b if b>c and b>d else c if c>d else d

def maxfun_version3(a,b,c,d):
      if a>b and a>c and a>d:
            return a
      elif b>c and b>d:
            return b
      elif c>d:
            return c
      else:
            return d
def maxfun_version4(a,b,c,d):
      call = lambda a,b,c,d: a if a>b and a>c and a>d else b if b>c and b>d else
c if c>d else d
      return call(a,b,c,d)

def maxfun_version5(a,b,c,d):
      L=[]
      L.append(a)
      L.append(b)
      L.append(c)
      L.append(d)
      return max(L)

a = int(input())
b = int(input())
c = int(input())
d = int(input())
print("max value by using version1:",maxfun_version1(a,b,c,d))
print("max value by using version2:",maxfun_version2(a,b,c,d))
print("max value by using version3:",maxfun_version3(a,b,c,d))
print("max value by using version4:",maxfun_version4(a,b,c,d))
print("max value by using version5:",maxfun_version5(a,b,c,d))
```

```
C:\8pm>py test.py
1
2
3
4
max value by using version1: 4
max value by using version2: 4
max value by using version3: 4
max value by using version4: 4
max value by using version5: 4

C:\8pm>py test.py
1
2
3
-4
max value by using version1: 3
max value by using version2: 3
max value by using version3: 3
max value by using version4: 3
max value by using version5: 3

C:\8pm>py test.py
1
2
-3
-4
max value by using version1: 2
max value by using version2: 2
max value by using version3: 2
max value by using version4: 2
max value by using version5: 2

C:\8pm>py test.py
1
-2
-3
-4
max value by using version1: 1
max value by using version2: 1
max value by using version3: 1
max value by using version4: 1
max value by using version5: 1

Ex9: WPP to find min of four numbers
Ex10: WPP to find max of five numbers
Ex11: WPP to find min of five numbers
Ex12: WPP to find difference between max and min of five numbers

Ex13: WPP to find factorial of the given number
-----------------------------------------------
Ex:
      5 -----> 5x4x3x2x1 = 120
      3 -----> 3x2x1 = 6

Algorithm:
----------
1) read n value from the user
2) apply business logic
logic1: by using while loop
logic2: by using recursion
logic3: by using predefined functions
3) print the result

logic1: by using while loop
---------------------------
fact = 1
i = 1
while i<=n:
      fact=fact*i
      i=i+1
print fact

logic2: by using recursion
---------------------------
def fun(n):
      if n==0:
            return 1
      else:
            return n*fun(n-1)

logic3: by using predefined functions
-------------------------------------
print math.factorial(n)

Implementation:
---------------
#Ex13: WPP to find factorial of the given number
import math
def factorial_logic1(n):
      f=1
      i=1
      while i<=n:
            f=f*i
            i=i+1
      return f

def factorial_logic2(n):
      if n==0:
            return 1
      else:
            return n*factorial_logic2(n-1)

def factorial_logic3(n):
      return math.factorial(n)
```

```
#main code
for i in range(10+1):

print(i,factorial_logic1(i),factorial_logic2(i),factorial_logic3(i),sep='\t\t')
```

```
C:\8pm>py test.py
0               1               1               1
1               1               1               1
2               2               2               2
3               6               6               6
4               24              24              24
5               120             120             120
6               720             720             720
7               5040            5040            5040
8               40320           40320           40320
9               362880          362880          362880
10              3628800         3628800         3628800
```

Ex14: WPP to check whether the given number is prime or not.

Ex:

```
      2     True
      3     True
      4     False
      5     True
      6     False
      7     True
```

Algorithm:
----

1) read 'n' value from the user.
2) apply business logic

logic1: by using loops
logic2: by using recursion

3) print the result

Ex:

```
def isprime1(n):
      factors=0
      for i in range(1,n+1):
            if n%i==0:
                  factors=factors+1
      return factors==2

def isprime2(n,i):
      if i==1:
            return True
      elif n%i==0:
            return False
      else:
            i=i-1
            return isprime2(n,i)

for i in range(2,11):
      print(f"i={i}\t{isprime1(i)}\t{isprime2(i,i//2)}")
```

```
C:\8pm>py test.py
i=2     True    True
i=3     True    True
```

```
i=4     False   False
i=5     True    True
i=6     False   False
i=7     True    True
i=8     False   False
i=9     False   False
i=10    False   False
```

Ex15: WPP to extract digits present in the given number.

Ex:

```
      123

      1 ----> 100th place
      2 ----> 10th place
      3 ----> 1unit place

      3x1     =  3
      2x10    = 20
      1x100   =100
      -------------
             123
      -------------
```

```
n = int(input("Enter any number: "))
while n!=0:
      d=n%10
      print(d)
      n=n//10
```

```
C:\8pm>py test.py
Enter any number: 123
3
2
1

C:\8pm>py test.py
Enter any number: 3409
9
0
4
3
```

Ex16: sum of digits present in the given number

Ex:

```
      123 -----> 1+2+3 = 6
      102 -----> 1+0+2 = 3
```

Algorithm:

```
      1. read a number from the user
      2. apply business logic
         logic1: by using digits extraction
         logic2: by using list
      3. print the result
```

logic1: by using digits extraction

```
sum=0
while n!=0:
    d=n%10
    sum=sum+d
    n=n//10
print sum
```

logic2: by using list

```
ptint sum([int(i) for i in n])
```

Ex:

```
def sumofdigits_v1(n):
      s=0
      while n!=0:
            d=n%10
            s=s+d
            n=n//10
      return s

def sumofdigits_v2(n):
      return sum([int(i) for i in str(n)])

n = int(input("Enter any number: "))
print(f'Sum of digits present in {n} is {sumofdigits_v1(n)}')
print(f'Sum of digits present in {n} is {sumofdigits_v2(n)}')
```

```
C:\8pm>py test.py
Enter any number: 123
Sum of digits present in 123 is 6
Sum of digits present in 123 is 6

C:\8pm>py test.py
Enter any number: 1209
Sum of digits present in 1209 is 12
Sum of digits present in 1209 is 12
```

Ex17: reverse of the given number

Ex:

```
      123 -----> 321
      121 -----> 121
      789 -----> 987
```

algorithm:

```
      1) read 'n' value from the user
      2) apply business logic
            logic1: extracting digits
            logic2: by using strings
      3) print the result
```

logic1: extracting digits

```
r = 0
while n!=0:
    d=n%10
    r=r*10+d
    n=n//10
print r
```

logic2: by using strings

```
str(n)[::-1]

def reverse_v1(n):
      r=0
      while n!=0:
            d=n%10
            r=r*10+d
            n=n//10
      return r

def reverse_v2(n):
      return str(n)[::-1]

n = int(input("Enter any number: "))
print(f'Reverse of {n} is {reverse_v1(n)}')
print(f'Reverse of {n} is {reverse_v2(n)}')
```

```
C:\8pm>py test.py
Enter any number: 123
Reverse of 123 is 321
Reverse of 123 is 321

C:\8pm>py test.py
Enter any number: 121
Reverse of 121 is 121
Reverse of 121 is 121

C:\8pm>py test.py
Enter any number: 120
Reverse of 120 is 21
Reverse of 120 is 021
```

Ex18: The given number is paliandrome number or not

```
Ex:
      123 ------> 321 -----> No
      121 ------> 121 -----> Yes
```

algorithm:
----

1) read a number from the user.
2) apply business logic.

logic1: by using digits extraction
logic2: by using strings

3) print the result.

logic1: by using digits extraction

```
temp=n
r = 0
while(n!=0)
    d = n % 10
    r = r * 10 + d
    n = n//10
if temp==r then 'Yes' else 'No'
```

logic2: by using strings

```
s = str(n)

if s==s[::-1] then 'Yes' else 'No'
```

```
def ispali_v1(n):
      temp = n
      r = 0
      while n!=0:
            d=n%10
            r=r*10+d
            n=n//10
      return r==temp

def ispali_v2(n):
      s=str(n)
      return s==s[::-1]

n = int(input("Enter any number: "))
print(ispali_v1(n))
print(ispali_v2(n))

C:\8pm>py test.py
Enter any number: 123
False
False

C:\8pm>py test.py
Enter any number: 121
True
True

Ex19: check whether the given digit is there in the number or not
-----------------------------------------------------------------
Ex:
      123,2 -------> True
      123,5 -------> False

algorithm:
----------
1) read a number from the user.
2) apply business logic
logic1: by using digits
logic2: by using strings
3) print the result

logic1: by using digits
-----------------------
n and digit

flag = False
while n!=0:
    d = n%10
    if digit == d:
       flag = True
       break
print flag

logic2: by using strings
------------------------
n and digit into string

print "digit in n"


def fun1(n,key):
      flag = False
      while n!=0:
            d = n%10
            if d==key:
                  flag = True
                  break
            n=n//10
      return flag
```

```
def fun2(n,key):
      return str(key) in str(n)

n = int(input("Enter any number: "))
key = int(input("Enter digit to check: "))
print(fun1(n,key))
print(fun2(n,key))
```

```
C:\8pm>py test.py
Enter any number: 1234
Enter digit to check: 4
True
True

C:\8pm>py test.py
Enter any number: 1234
Enter digit to check: 5
False
False
```

next class is on monday.....

8pm to 9pm.....

Ex1: WPP to read a string and convert all even indexed values into upper case.
Ex2: WPP to read a string and convert all odd indexed values into upper case.
Ex3: WPP to find sum of all elements present in a list
Ex4: WPP to find max of two numbers
Ex5: WPP to find min of two numbers
Ex6: WPP to find max of three numbers
Ex7: WPP to find min of three numbers
Ex8: WPP to find max of four numbers
Ex9: WPP to find min of four numbers
Ex10: WPP to find max of five numbers
Ex11: WPP to find min of five numbers
Ex12: WPP to find difference between max and min of five numbers
Ex13: WPP to find factorial of the given number
Ex14: WPP to check whether the given number is prime or not.

K Prakash Babu
7386237319

Name:
Email:
Batch: DSA with Python B1

Chapter:01 --> Algorithms and Analysis of Algorithms

Algorithm:

```
=> a step by step process to solve a problem is called as an algorithm.
=> finate set of steps.
=> unambiguos step.
=> advantage: we will get a pattern or template or format to solve the problem.

Properties of an algorithm:
---------------------------
1) zero or more inputs.
2) one or more outputs.
3) algorithm should be deterministic (same ouput if we run any times).
4) instructions should clear and correct.
5) terminate at finate steps.
6) efficient in solving problems.

complexity of algorithm:
~~~~~~~~~~~~~~~~~~~~~~~~
complexity of an algorithm is the amount of time and space required to complete
its execution.

Time Complexity T(n) => Amount of time taken by an algorithm
Space Complexity S(n)=> Amount of space taken by an algorithm

Note: sec, msec, nsec, bytes, bits, kb, mb, etc

Asymptotic Analysis or Asymptotic Notations
-------------------------------------------
calculating running time and space of any algorithm in mathmatical units of
computation is know as asymptotic analysis.

1) Big-O notation
2) Omega-w notation
3) Theta-0 notation

Big-O notation:
---------------
f(n) = -----
g(n) = -----

f(n) = O(g(n))

f(n)<=g(n)

Omega-w notation
----------------
f(n) = -----
g(n) = -----

f(n) = w(g(n)

f(n) >= g(n)

Theta notation
--------------
f(n) = ------
g(n) = ------

f(n) = 0(g(n))

c1g(n)<=f(n)<=c2g(n)

where c1 and c2 are some constant values...

Complexity analysis of algorithms:
----------------------------------
```

worst case complexity -----> max steps required by an algorithm -----> O

best case complexity ------> min steps required by an algorithm -----> w

average case complexity ---> average steps required by an algorithm -> 0

Note: by default we will calculate time and space complexity for worst case.

Growth of functions:

1) constant time O(1)

algorithm will return a constant time.

Ex:

- access nth element in a list
- push and pop operations stack
- add and remove from queue
- accessing element from hash table etc

2) Linear time O(n)

linear time i.e. execution time is directly proportional to input size.

Ex:

- search
- min element in the list
- max element in the list
- traveral operation (visiting each node/data/field) etc

3) Logarithmic time O(logn)

algorithm is said to run in logarithmic time. if the execution time of an alg is proportional to logarithm of input size.

Ex:

- binary search

4) O(nlogn)

algorithm will run in n*logn time, if the execution time of an alg is proportional to the product of input size and logarithmic of input size.

Ex:

- merge sort
- quick sort
- heap sort etc

5) Quadratic time O(n^2)

An algorithm is said to run in quadratic time of an alg is proportional to square of the input size.

Ex:

- bubble sort
- selection sort
- insertion sort etc

6) Exponential Time O(2^n)

In these algorithms, all possible subsets of elements of input data are generated.

Ex:

- power set
- sub sets etc

7) Factorial Time O(n!)

All possible permutations of all the elements of input data are generated.

Ex:
      finding permutations of string etc

Ex1: loops

```
def fun(n):
      c = 0
      i = 0
      while i < n:
            c = c + 1
            i = i + 1
      return c

print("N=100, number of instructions is O(n): ",fun(100))
```

```
C:\8pm>py test.py
N=100, number of instructions is O(n):  100
```

Ex2: nested loops-1

```
def fun(n):
      c = 0
      i = 0
      while i < n:
            j = 0
            while j < n:
                  c = c + 1
                  j = j + 1
            i = i + 1
      return c

print("N=100, number of instructions is O(n^2): ",fun(100)) #100x100=10000
```

```
C:\8pm>py test.py
N=100, number of instructions is O(n^2):  10000
```

Ex3: nested loops-2

```
def fun(n):
      c = 0
      i = 0
      while i < n:
            j = 0
            while j < n:
                  k = 0
                  while k < n:
                        c = c + 1
                        k = k + 1
                  j = j + 1
            i = i + 1
      return c

print("N=100, number of instructions is O(n^3): ",fun(100)) #100x100x100=1000000
```

```
C:\8pm>py test.py
N=100, number of instructions is O(n^3):  1000000
```

Ex4: Arithmetic Series

```
def fun(n):
      c = 0
      i = 0
      while i < n:
            j = 0
            while j < i:
                  c = c + 1
                  j = j + 1
            i = i + 1
      return c

print("N=100, number of instructions is O(n^2)",fun(100))
```

```
C:\8pm>py test.py
N=100, number of instructions is O(n^2) 4950
```

Ex5: Double The Iteration Variable

```
def fun(n):
      c = 0
      i = 1
      while i < n:
            c = c + 1
            i = i * 2
      return c

print("N=100, number of instructions is O(logn)",fun(100))
```

```
C:\8pm>py test.py
N=100, number of instructions is O(logn) 7
```

Ex6: Half the iteration variable

```
def fun(n):
      c = 0
      i = n
      while i > 0:
            c = c + 1
            i = i // 2
      return c

print("N=100, number of instructions is O(logn)",fun(100))
```

```
C:\8pm>py test.py
N=100, number of instructions is O(logn) 7
```

Ex7: Consecutive statements

```
def fun(n):
      c = 0
      i = 0
      while i < n: #O(n2)
            j = 0
            while j < n:
                  c = c + 1
                  j = j + 1
            i = i + 1
      i = 0
      while i < n: #O(n2)
            k = 0
            while k < n:
                  c = c + 1
                  k = k + 1
            i = i + 1
      return c

print("N=100, number of instructions is O(n2)",fun(100))

C:\8pm>py test.py
N=100, number of instructions is O(n2) 20000

Ex8:
----
def fun(n):
      c = 0
      i = n
      while i > 0:
            j = 0
            while j < i:
                  c = c + 1
                  j = j + 1
            i = i // 2
      return c


print("N=100, number of instructions is O(logn)",fun(100))

C:\8pm>py test.py
N=100, number of instructions is O(logn) 197


Ex9:
----
def fun(n):
      c = 0
      i = 1
      while i < n:
            j = 0
            while j < i:
                  c = c + 1
                  j = j + 1
            i = i * 2
      return c


print("N=100, number of instructions is O(logn)",fun(100))

C:\8pm>py test.py
N=100, number of instructions is O(logn) 127

Ex10: Multiple loops in O(n) time
----------------------------------
def fun(n):
      c = 0
      i = 0
      j = 0
      while i < n:
            while j < n:
                  c = c + 1
                  j = j + 1
            i = i + 1
      return c


print("N=100, number of instructions is O(n)",fun(100))

C:\8pm>py test.py
```

N=100, number of instructions is O(n) 100

Chapter:02 --> Approach to solve Problems

=> Theoretical knowledge is essential but it is insufficient.
=> The following are the main approaches to solve any problem in real world.

1) constraints
2) idea generation
3) complexities analysis
4) coding
5) testing

1) constraints

Given problem constrains are very very imp, first we have to identify all the constraints related to the given problem.

Ex: sorting application

- -> asc order or desc order
- -> number of elements
- -> type of elements

2) idea generation

* more if you practice, you will get idea.
* by practicing you will get a pattern of problem.
* easily we can solve unseen problems

  1) try to simplify task at hand
  2) few examples (apply)
  3) think about sutable data structure
  4) think about similiar problems you solved

3) complexities analysis

=> finding solution for a problem is not sufficient.
=> find a solution which is fast and take less memory.
=> try to find time and space complexities and find the best algorithm.

4) coding

=> if you have all data, then we can write the code.
=> select programming language (python)
=> select proper IDE
=> and try to write MODULAR code (reusability)

5) testing

=> after completion of program, validate code.
=> apply varies test cases and solve

   1) normal test cases ---> basic +ve cases
   2) edge test cases -----> corner test case

Chapter:03 --> Sample Algorithms and Implementation

01. Even or Odd

```
def evenorodd_v1(n):
      return n%2==0

def evenorodd_v2(n):
      return (n&1)==0
```

```
n = int(input("Enter any number: "))
print(evenorodd_v1(n))
print(evenorodd_v2(n))
```

```
C:\8pm>py test.py
Enter any number: 5
False
False

C:\8pm>py test.py
Enter any number: 6
True
True
```

02. Max of two numbers

```
def max_v1(a,b):
      if a>b:
            return a
      else:
            return b

def max_v2(a,b):
      return a if a>b else b

def max_v3(a,b):
      res = lambda a,b: a if a>b else b
      return res(a,b)

def max_v4(a,b):
      return max(a,b)

a = int(input("Enter a value: "))
b = int(input("Enter b value: "))
print(max_v1(a,b))
print(max_v2(a,b))
print(max_v3(a,b))
print(max_v4(a,b))
```

```
C:\8pm>py test.py
Enter a value: 10
Enter b value: 20
20
20
20
20

C:\8pm>py test.py
Enter a value: 10
Enter b value: -20
10
10
10
10
```

03. Min of two numbers

```
def min_v1(a,b):
      if a<b:
            return a
      else:
            return b

def min_v2(a,b):
      return a if a<b else b

def min_v3(a,b):
      res = lambda a,b: a if a<b else b
      return res(a,b)

def min_v4(a,b):
      return min(a,b)

a = int(input("Enter a value: "))
b = int(input("Enter b value: "))
print(min_v1(a,b))
print(min_v2(a,b))
print(min_v3(a,b))
print(min_v4(a,b))

C:\8pm>py test.py
Enter a value: 10
Enter b value: 20
10
10
10
10

C:\8pm>py test.py
Enter a value: 10
Enter b value: -20
-20
-20
-20
-20
```

04. Max of three numbers

```
def max_v1(a,b,c):
      if a>b and a>c:
            return a
      elif b>c:
            return b
      else:
            return c

def max_v2(a,b,c):
      return a if a>b and a>c else b if b>c else c

def max_v3(a,b,c):
      res = lambda a,b,c: a if a>b and a>c else b if b>c else c
      return res(a,b,c)

def max_v4(a,b,c):
      return max(a,b,c)

a = int(input("Enter a value: "))
b = int(input("Enter b value: "))
c = int(input("Enter c value: "))
print(max_v1(a,b,c))
print(max_v2(a,b,c))
print(max_v3(a,b,c))
print(max_v4(a,b,c))

C:\8pm>py test.py
Enter a value: 1
Enter b value: 2
Enter c value: 3
```

```
3
3
3
3

C:\8pm>py test.py
Enter a value: 1
Enter b value: 2
Enter c value: -3
2
2
2
2

C:\8pm>py test.py
Enter a value: 1
Enter b value: -2
Enter c value: -3
1
1
1
1
```

```
05. Min of three numbers
------------------------
def min_v1(a,b,c):
      if a<b and a<c:
            return a
      elif b<c:
            return b
      else:
            return c

def min_v2(a,b,c):
      return a if a<b and a<c else b if b<c else c

def min_v3(a,b,c):
      res = lambda a,b,c: a if a<b and a<c else b if b<c else c
      return res(a,b,c)

def min_v4(a,b,c):
      return min(a,b,c)

a = int(input("Enter a value: "))
b = int(input("Enter b value: "))
c = int(input("Enter c value: "))
print(min_v1(a,b,c))
print(min_v2(a,b,c))
print(min_v3(a,b,c))
print(min_v4(a,b,c))
```

```
06. Max of four numbers
-----------------------
def max_v1(a,b,c,d):
      if a>b and a>c and a>d:
            return a
      elif b>c and b>d:
            return b
      elif c>d:
            return c
      else:
            return d

def max_v2(a,b,c,d):
      return a if a>b and a>c and a>d else b if b>c and b>d else c if c>d else d

def max_v3(a,b,c,d):
      res = lambda a,b,c,d: a if a>b and a>c and a>d else b if b>c and b>d else
c if c>d else d
      return res(a,b,c,d)

def max_v4(a,b,c,d):
      return max(a,b,c,d)

a = int(input("Enter a value: "))
b = int(input("Enter b value: "))
c = int(input("Enter c value: "))
d = int(input("Enter d value: "))
print(max_v1(a,b,c,d))
print(max_v2(a,b,c,d))
print(max_v3(a,b,c,d))
print(max_v4(a,b,c,d))
```

```
C:\8pm>py test.py
Enter a value: 1
Enter b value: 2
Enter c value: 3
Enter d value: 4
4
4
4
4

C:\8pm>py test.py
Enter a value: 1
Enter b value: 2
Enter c value: 3
Enter d value: -4
3
3
3
3

C:\8pm>py test.py
Enter a value: 1
Enter b value: 2
Enter c value: -3
Enter d value: -4
2
2
2
2

C:\8pm>py test.py
Enter a value: 1
Enter b value: -2
Enter c value: -3
Enter d value: -4
1
1
1
1
```

07. Min of four numbers

```
def min_v1(a,b,c,d):
      if a<b and a<c and a<d:
            return a
```

```
        elif b<c and b<d:
                return b
        elif c<d:
                return c
        else:
                return d

def min_v2(a,b,c,d):
        return a if a<b and a<c and a<d else b if b<c and b<d else c if c<d else d

def min_v3(a,b,c,d):
        res = lambda a,b,c,d: a if a<b and a<c and a<d else b if b<c and b<d else
c if c<d else d
        return res(a,b,c,d)

def min_v4(a,b,c,d):
        return min(a,b,c,d)

a = int(input("Enter a value: "))
b = int(input("Enter b value: "))
c = int(input("Enter c value: "))
d = int(input("Enter d value: "))
print(min_v1(a,b,c,d))
print(min_v2(a,b,c,d))
print(min_v3(a,b,c,d))
print(min_v4(a,b,c,d))
```

```
C:\8pm>py test.py
Enter a value: 1
Enter b value: 2
Enter c value: 3
Enter d value: 4
1
1
1
1

C:\8pm>py test.py
Enter a value: 1
Enter b value: 2
Enter c value: 3
Enter d value: -4
-4
-4
-4
-4

C:\8pm>py test.py
Enter a value: 1
Enter b value: 2
Enter c value: -3
Enter d value: 4
-3
-3
-3
-3

C:\8pm>py test.py
Enter a value: 1
Enter b value: -2
Enter c value: 3
Enter d value: 4
-2
-2
```

```
-2
-2
```

08. Max of five numbers

```
def max_v1(a,b,c,d,e):
      if a>b and a>c and a>d and a>e:
            return a
      elif b>c and b>d and b>e:
            return b
      elif c>d and c>e:
            return c
      elif d>e:
            return d
      else:
            return e

def max_v2(a,b,c,d,e):
      return a if a>b and a>c and a>d and a>e else b if b>c and b>d and b>e else c if c>d and c>e else d if d>e else e

def max_v3(a,b,c,d,e):
      res = lambda a,b,c,d,e: a if a>b and a>c and a>d and a>e else b if b>c and b>d and b>e else c if c>d and c>e else d if d>e else e
      return res(a,b,c,d,e)

def max_v4(a,b,c,d,e):
      return max(a,b,c,d,e)

a = int(input("Enter a value: "))
b = int(input("Enter b value: "))
c = int(input("Enter c value: "))
d = int(input("Enter d value: "))
e = int(input("Enter e value: "))
print(max_v1(a,b,c,d,e))
print(max_v2(a,b,c,d,e))
print(max_v3(a,b,c,d,e))
print(max_v4(a,b,c,d,e))
```

```
C:\8pm>py test.py
Enter a value: 1
Enter b value: 2
Enter c value: 3
Enter d value: 4
Enter e value: 5
5
5
5
5

C:\8pm>py test.py
Enter a value: 1
Enter b value: 2
Enter c value: 3
Enter d value: 4
Enter e value: -5
4
4
4
4

C:\8pm>py test.py
Enter a value: 1
Enter b value: 2
Enter c value: 3
Enter d value: -4
Enter e value: -5
3
3
3
3

C:\8pm>py test.py
Enter a value: 1
Enter b value: 2
Enter c value: -3
Enter d value: -4
Enter e value: -5
2
2
2
2

C:\8pm>py test.py
Enter a value: 1
Enter b value: -2
Enter c value: -3
Enter d value: -4
Enter e value: -5
1
1
1
1
```

```
09. Min of five numbers
-----------------------
def min_v1(a,b,c,d,e):
      if a<b and a<c and a<d and a<e:
            return a
      elif b<c and b<d and b<e:
            return b
      elif c<d and c<e:
            return c
      elif d<e:
            return d
      else:
            return e

def min_v2(a,b,c,d,e):
      return a if a<b and a<c and a<d and a<e else b if b<c and b<d and b<e else c if c<d and c<e else d if d<e else e

def min_v3(a,b,c,d,e):
      res = lambda a,b,c,d,e: a if a<b and a<c and a<d and a<e else b if b<c and b<d and b<e else c if c<d and c<e else d if d<e else e
      return res(a,b,c,d,e)

def min_v4(a,b,c,d,e):
      return min(a,b,c,d,e)

a = int(input("Enter a value: "))
b = int(input("Enter b value: "))
c = int(input("Enter c value: "))
d = int(input("Enter d value: "))
e = int(input("Enter e value: "))
print(min_v1(a,b,c,d,e))
print(min_v2(a,b,c,d,e))
print(min_v3(a,b,c,d,e))
print(min_v4(a,b,c,d,e))

C:\8pm>py test.py
Enter a value: 1
Enter b value: 2
Enter c value: 3
Enter d value: 4
Enter e value: 5
1
1
1
1

C:\8pm>py test.py
Enter a value: 1
Enter b value: 2
Enter c value: 3
Enter d value: 4
Enter e value: -5
-5
-5
-5
-5

C:\8pm>py test.py
Enter a value: 1
Enter b value: 2
Enter c value: 3
Enter d value: -4
Enter e value: 5
-4
-4
-4
-4

C:\8pm>py test.py
Enter a value: 1
Enter b value: 2
Enter c value: 3
Enter d value: -4
Enter e value: -5
-5
-5
-5
-5

C:\8pm>py test.py
Enter a value: 1
Enter b value: -2
Enter c value: 3
Enter d value: 4
Enter e value: 5
-2
-2
-2
-2
```

10. Swaping of two integer values

```
def swap_v1(a,b):
      print(f"before swaping: a={a} and b={b}")
      #logic
      a,b = b,a
      print(f"after  swaping: a={a} and b={b}")
```

```
def swap_v2(a,b):
      print(f"before swaping: a={a} and b={b}")
      #logic
      t = a
      a = b
      b = t
      print(f"after  swaping: a={a} and b={b}")

def swap_v3(a,b):
      print(f"before swaping: a={a} and b={b}")
      #logic
      a = a + b
      b = a - b
      a = a - b
      print(f"after  swaping: a={a} and b={b}")

def swap_v4(a,b):
      print(f"before swaping: a={a} and b={b}")
      #logic
      a = a * b
      b = a // b
      a = a // b
      print(f"after  swaping: a={a} and b={b}")

def swap_v5(a,b):
      print(f"before swaping: a={a} and b={b}")
      #logic
      a = a ^ b
      b = a ^ b
      a = a ^ b
      print(f"after  swaping: a={a} and b={b}")

def swap_v6(a,b):
      print(f"before swaping: a={a} and b={b}")
      #logic
      a = a+b-(b:=a)
      print(f"after  swaping: a={a} and b={b}")


a = int(input("Enter a value: "))
b = int(input("Enter b value: "))
swap_v1(a,b)
swap_v2(a,b)
swap_v3(a,b)
swap_v4(a,b)
swap_v5(a,b)
swap_v6(a,b)
```

```
C:\8pm>py test.py
Enter a value: 10
Enter b value: 20
before swaping: a=10 and b=20
after  swaping: a=20 and b=10
before swaping: a=10 and b=20
after  swaping: a=20 and b=10
before swaping: a=10 and b=20
after  swaping: a=20 and b=10
before swaping: a=10 and b=20
after  swaping: a=20 and b=10
before swaping: a=10 and b=20
after  swaping: a=20 and b=10
before swaping: a=10 and b=20
after  swaping: a=20 and b=10
```

11. Absolute value

```
Ex:
      5 ---> 5
      -5 --> 5
```

```
def abs_v1(n):
      if n<0:
            return -n
      else:
            return n

def abs_v2(n):
      return abs(n)

n = int(input("Enter n value: "))
print(f"Original Value= {n} and Absolute Value= {abs_v1(n)}")
print(f"Original Value= {n} and Absolute Value= {abs_v2(n)}")
```

```
C:\test>py test.py
Enter n value: 10
Original Value= 10 and Absolute Value= 10
Original Value= 10 and Absolute Value= 10

C:\test>py test.py
Enter n value: -111
Original Value= -111 and Absolute Value= 111
Original Value= -111 and Absolute Value= 111
```

12. Sum of n natural numbers

```
1+2+3+4+...+n

n=4 ---> 1+2+3+4 = 10
n=5 ---> 1+2+3+4+5 = 15
```

```
def sum_v1(n):
      s=0
      for i in range(1,n+1):
            s=s+i
      return s

def sum_v2(n):
      return n*(n+1)//2

def sum_v3(n):
      if n==0:
            return 0
      else:
            return n+sum_v3(n-1)

n = int(input("Enter n value: "))
print(f"number= {n} and sum= {sum_v1(n)}")
print(f"number= {n} and sum= {sum_v2(n)}")
print(f"number= {n} and sum= {sum_v3(n)}")
```

```
C:\test>py test.py
Enter n value: 0
number= 0 and sum= 0
number= 0 and sum= 0
number= 0 and sum= 0
```

```
C:\test>py test.py
Enter n value: 1
number= 1 and sum= 1
number= 1 and sum= 1
number= 1 and sum= 1

C:\test>py test.py
Enter n value: 2
number= 2 and sum= 3
number= 2 and sum= 3
number= 2 and sum= 3
```

13. Factorial of given number

n=3 ---> 3x2x1 = 6
n=4 ---> 4x3x2x1 = 24
n=5 ---> 5x4x3x2x1 = 120

```
import math

def fact_v1(n):
      f=1
      for i in range(1,n+1):
            f=f*i
      return f

def fact_v2(n):
      return math.factorial(n)

def fact_v3(n):
      if n==0:
            return 1
      else:
            return n*fact_v3(n-1)

n = int(input("Enter n value: "))
print(f"number= {n} and factorial= {fact_v1(n)}")
print(f"number= {n} and factorial= {fact_v2(n)}")
print(f"number= {n} and factorial= {fact_v3(n)}")
```

```
C:\test>py test.py
Enter n value: 5
number= 5 and factorial= 120
number= 5 and factorial= 120
number= 5 and factorial= 120

C:\test>py test.py
Enter n value: 0
number= 0 and factorial= 1
number= 0 and factorial= 1
number= 0 and factorial= 1

C:\test>py test.py
Enter n value: 3
number= 3 and factorial= 6
number= 3 and factorial= 6
number= 3 and factorial= 6
```

14. Digits Extraction

extract digits from the given numbers

```
def digits_v1(n):
      while n!=0:
            print(n%10)
            n=n//10

def digits_v2(n):
      s = str(n)
      for i in s[::-1]:
            print(i)

n = int(input("Enter n value: "))
digits_v1(n)
print("-----")
digits_v2(n)
```

```
C:\test>py test.py
Enter n value: 123
3
2
1
-----
3
2
1
```

15. Count digits

```
def count_v1(n):
      return len(str(n))

def count_v2(n):
      c=0
      while n!=0:
            c=c+1
            n=n//10
      return c

def count_v3(n):
      if n==0:
            return 0
      else:
            return 1+count_v3(n//10)

n = int(input("Enter n value: "))
print(f"number= {n} and count of digits= {count_v1(n)}")
print(f"number= {n} and count of digits= {count_v2(n)}")
print(f"number= {n} and count of digits= {count_v3(n)}")
```

```
C:\test>py test.py
Enter n value: 1234
number= 1234 and count of digits= 4
number= 1234 and count of digits= 4
number= 1234 and count of digits= 4
```

16. Reverse a number

```
def rev_v1(n):
      r = 0
      while n!=0:
            d = n % 10
            r = r * 10 + d
            n = n // 10
      return r

def rev_v2(n):
      return str(n)[::-1]
```

```
n = int(input("Enter any number: "))
print(f"original number= {n} and reverse= {rev_v1(n)}")
print(f"original number= {n} and reverse= {rev_v2(n)}")
```

```
C:\test>py test.py
Enter any number: 123
original number= 123 and reverse= 321
original number= 123 and reverse= 321

C:\test>py test.py
Enter any number: 101
original number= 101 and reverse= 101
original number= 101 and reverse= 101
```

17. Paliandrome number

```
def rev_v1(n):
      t = n
      r = 0
      while n!=0:
            d = n % 10
            r = r * 10 + d
            n = n // 10
      return t==r

def rev_v2(n):
      return str(n)==str(n)[::-1]


n = int(input("Enter any number: "))
print(f"original number= {n} and ispali= {rev_v1(n)}")
print(f"original number= {n} and ispali= {rev_v2(n)}")
```

```
C:\test>py test.py
Enter any number: 123
original number= 123 and ispali= False
original number= 123 and ispali= False

C:\test>py test.py
Enter any number: 101
original number= 101 and ispali= True
original number= 101 and ispali= True
```

18. Trailing Zeros of factorial

```
import math

def fun(n):
      f = math.factorial(n)
      c = 0
      while f!=0:
            if f%10!=0:
                  break
            c=c+1
            f=f//10
      return c

n = int(input("Enter any number: "))
print(f"number= {n}, fact= {math.factorial(n)} and training 0s= {fun(n)}")
```

```
C:\test>py test.py
Enter any number: 7
```

```
number= 7, fact= 5040 and training 0s= 1

C:\test>py test.py
Enter any number: 10
number= 10, fact= 3628800 and training 0s= 2
```

```
Note:
-----
import math

def fun1(n):
      f = str(math.factorial(n))[::-1]
      c = 0
      for i in f:
            if i=='0':
                  c=c+1
      return c

def fun(n):
      f = math.factorial(n)
      c = 0
      while f!=0:
            if f%10!=0:
                  break
            c=c+1
            f=f//10
      return c

n = int(input("Enter any number: "))
print(f"number= {n}, fact= {math.factorial(n)} and training 0s= {fun1(n)}")
```

19. x to the power y

```
-------------------
x^y -----> 2^3 ----> 8
```

```
import math

def power_v1(x,y):
      res = 1
      for i in range(1,y+1): #(0,y)
            res = res * x
      return res

def power_v2(x,y):
      return x**y

def power_v3(x,y):
      return int(math.pow(x,y))

x = int(input("Enter x value: "))
y = int(input("Enter y value: "))
print(f"x= {x}, y= {y} and {x} to the power {y} is : {power_v1(x,y)}")
print(f"x= {x}, y= {y} and {x} to the power {y} is : {power_v2(x,y)}")
print(f"x= {x}, y= {y} and {x} to the power {y} is : {power_v3(x,y)}")
```

```
C:\test>py test.py
Enter x value: 2
Enter y value: 3
x= 2, y= 3 and 2 to the power 3 is : 8
x= 2, y= 3 and 2 to the power 3 is : 8
x= 2, y= 3 and 2 to the power 3 is : 8
```

```
C:\test>py test.py
Enter x value: 10
Enter y value: 4
x= 10, y= 4 and 10 to the power 4 is : 10000
x= 10, y= 4 and 10 to the power 4 is : 10000
x= 10, y= 4 and 10 to the power 4 is : 10000
```

20. Sum of digits

```
def sum_v1(n):
      s=0
      while n!=0:
            d = n % 10
            s = s + d
            n = n // 10
      return s

def sum_v2(n):
      return sum([int(i) for i in str(n)])


n = int(input("Enter n value: "))
print(f"n= {n} and sum of digits is : {sum_v1(n)}")
print(f"n= {n} and sum of digits is : {sum_v2(n)}")
```

```
C:\test>py test.py
Enter n value: 123
n= 123 and sum of digits is : 6
n= 123 and sum of digits is : 6

C:\test>py test.py
Enter n value: 10982
n= 10982 and sum of digits is : 20
n= 10982 and sum of digits is : 20
```

21. Sum of even digits

```
def sum_v1(n):
      s=0
      while n!=0:
            d = n % 10
            if d%2==0:
                  s = s + d
            n = n // 10
      return s

def sum_v2(n):
      return sum([int(i) for i in str(n) if int(i)%2==0])


n = int(input("Enter n value: "))
print(f"n= {n} and sum of even digits is : {sum_v1(n)}")
print(f"n= {n} and sum of even digits is : {sum_v2(n)}")
```

```
C:\test>py test.py
Enter n value: 1234
n= 1234 and sum of even digits is : 6
n= 1234 and sum of even digits is : 6
```

22. Sum of odd digits

```
def sum_v1(n):
      s=0
      while n!=0:
            d = n % 10
            if d%2!=0:
                  s = s + d
            n = n // 10
      return s

def sum_v2(n):
      return sum([int(i) for i in str(n) if int(i)%2!=0])


n = int(input("Enter n value: "))
print(f"n= {n} and sum of odd digits is : {sum_v1(n)}")
print(f"n= {n} and sum of odd digits is : {sum_v2(n)}")
```

```
C:\test>py test.py
Enter n value: 1234
n= 1234 and sum of odd digits is : 4
n= 1234 and sum of odd digits is : 4
```

23. Sum of prime digits

```
def sum_v1(n):
      s=0
      while n!=0:
            d = n % 10
            if d==2 or d==3 or d==5 or d==7:
                  s = s + d
            n = n // 10
      return s

def sum_v2(n):
      return sum([int(i) for i in str(n) if i in "2357"])


n = int(input("Enter n value: "))
print(f"n= {n} and sum of prime digits is : {sum_v1(n)}")
print(f"n= {n} and sum of prime digits is : {sum_v2(n)}")
```

```
C:\test>py test.py
Enter n value: 12345
n= 12345 and sum of prime digits is : 10
n= 12345 and sum of prime digits is : 10
```

24. Prime Number or Not

```
def prime_v1(n):
      f = 0
      for i in range(1,n+1):
            if n%i==0:
                  f=f+1
      return f==2

def prime_v2(n,i):
      if i==1:
            return True
      elif n%i==0:
            return False
      else:
            return prime_v2(n,i-1)

n = int(input("Enter n value: "))
print(f"n= {n} and it is prime : {prime_v1(n)}")
print(f"n= {n} and it is prime : {prime_v2(n,n//2)}")
```

```
C:\test>py test.py
Enter n value: 7
n= 7 and it is prime : True
n= 7 and it is prime : True

C:\test>py test.py
Enter n value: 8
n= 8 and it is prime : False
n= 8 and it is prime : False
```

25. All divisors of N

```
---------------------
def divisors(n):
    L = []
    for i in range(1,n+1):
        if n%i==0:
            L.append(i)
    return L

n = int(input("Enter n value: "))
print(f"n= {n} and divisors : {divisors(n)}")

C:\test>py test.py
Enter n value: 10
n= 10 and divisors : [1, 2, 5, 10]

C:\test>py test.py
Enter n value: 20
n= 20 and divisors : [1, 2, 4, 5, 10, 20]

C:\test>py test.py
Enter n value: 30
n= 30 and divisors : [1, 2, 3, 5, 6, 10, 15, 30]
```

26. Perfect Number

```
------------------
n ----> sum of its factors excluding that number should be equal to n

6 ----> 1, 2, 3, 6 ----> 1+2+3 = 6 -----> True
10 ---> 1, 2, 5, 10 ---> 1+2+5 = 8 -----> False

def divisors(n):
    L = []
    for i in range(1,n):
        if n%i==0:
            L.append(i)
    return L

def isperfect(n):
    L = divisors(n)
    return n==sum(L)


n = int(input("Enter n value: "))
print(f"n= {n} and it is perfect number : {isperfect(n)}")

C:\test>py test.py
Enter n value: 6
n= 6 and it is perfect number : True

C:\test>py test.py
Enter n value: 7
n= 7 and it is perfect number : False
```

```
C:\test>py test.py
Enter n value: 8
n= 8 and it is perfect number : False

C:\test>py test.py
Enter n value: 12
n= 12 and it is perfect number : False

C:\test>py test.py
Enter n value: 496
n= 496 and it is perfect number : True

C:\test>py test.py
Enter n value: 28
n= 28 and it is perfect number : True
```

27. armstrong number

```
--------------------
123 ----> 1^3 + 2^3 + 3^3 = 1 + 8 + 27 = 36 ----> False
153 ----> 1^3 + 5^3 + 3^3 = 1 + 125 + 27 = 153 -> True
```

```
def armstrong(n):
      s=0
      t=n
      while n!=0:
            d=n%10
            s=s+d**3
            n=n//10
      return s==t

for i in range(1,1000+1):
      if armstrong(i):
            print(i)
```

```
C:\test>py test.py

C:\test>py test.py
1
153
370
371
407
```

28. strong number

```
-----------------
123 ----> 1! + 2! + 3! = 1 + 2 + 6 = 9 ------> False
145 ----> 1! + 4! + 5! = 1 + 24 + 120 = 145 -> True
```

```
import math
def strong(n):
      s=0
      t=n
      while n!=0:
            d=n%10
            s=s+math.factorial(d)
            n=n//10
      return s==t

for i in range(1,100000+1):
      if strong(i):
            print(i)
```

```
C:\test>py test.py
1
2
145
40585
```

29. Fib sequence

0 1 1 2 3 5 8 13 ......

```
def fib(n):
      L = []
      a = 0
      b = 1
      L.append(a)
      L.append(b)
      for i in range(n-2):
            c = a + b
            L.append(c)
            a = b
            b = c
      return L


print(fib(5)) #[0, 1, 1, 2, 3]
```

30. Trib Sequence

0 1 2 3 6 11 20 37 ...

```
def trib(n):
      L = []
      a = 0
      b = 1
      c = 2
      L.append(a)
      L.append(b)
      L.append(c)
      for i in range(n-3):
            d = a + b + c
            L.append(d)
            a = b
            b = c
            c = d
      return L


print(trib(6)) #[0, 1, 2, 3, 6, 11]
```

```
C:\test>py test.py
[0, 1, 2, 3, 6, 11]
```

Chapter:04 -----> data structures in python

01. mutable and immutable objects
02. string data structure
03. list data structure
04. tuple data structure
05. set data structure
06. dict data structure
07. sample programs

01. mutable and immutable objects

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

mutable objects:

once if an object is created, if we are trying to perform modifications on the existing object, those modifications will be reflected on the same object, then such type of objects are called as mutable objects.

Ex: list, set and dict

Ex:

```
L = [10, 20, 30]
print(L) #[10, 20, 30]
L[0] = 999
print(L) #[999, 20, 30]

C:\test>py test.py
[10, 20, 30]
[999, 20, 30]
```

immutable objects:

once if an object is created, if we are trying to perform modifications on the existing object, with those modifications a new object will be created modifications wn't be reflected on the same object, then such type of objects are called as immutable objects.

Ex: tuple, string and fundamental data types

Ex:

```
S = "WELKOME"
print(S) #WELKOME
S[3] = 'C'

#TypeError: 'str' object does not support item assignment
```

02. string data structure

introduction:

==> collection or sequence or group of characters is called as string.
==> <class 'str'>
==> we can represent string objects in the following ways

    1. single quotes
    2. double quotes
    3. triple single quotes
    4. triple dounle quotes

Ex:

```
s1 = "Hai"
s2 = 'Hi'
s3 = """Bye"""
s4 = '''Bi'''

print(s1,type(s1)) #Hai <class 'str'>
print(s2,type(s2)) #Hi <class 'str'>
print(s3,type(s3)) #Bye <class 'str'>
print(s4,type(s4)) #Bi <class 'str'>

C:\test>py test.py
Hai <class 'str'>
```

```
Hi <class 'str'>
Bye <class 'str'>
Bi <class 'str'>
```

index concept:

we can use index and subscript combination to extract individual characters from a string.

syntax:

```
      s[index_value]
```

=> index value is always integer value.
=> +ve and -ve values.
=> +ve ---> left to right
=> -ve ---> right to left

Ex:

```
s = "hai"
#    012
#   -321
print(s[0]) #h
print(s[1]) #a
print(s[2]) #i
print(s[-1]) #i
print(s[-2]) #a
print(s[-3]) #h
```

Ex:

```
s = "hai"
#    012
#   -321
print(s[3]) #IndexError: string index out of range
```

Ex:

```
s = "hai"
#    012
#   -321
print(s[-5]) #IndexError: string index out of range
```

accessing string objects:

The following are the various method to access string objects

1) directly we can print
2) index concept
3) slice operator
4) while loop
5) for each loop

Ex:

```
s = "prakash"

#1) directly we can print
print(s) #prakash

#2) index concept
print(s[0]) #p
print(s[1]) #r
print(s[2]) #a
print(s[3]) #k
print(s[4]) #a
print(s[5]) #s
print(s[6]) #h

#4) while loop

index = 0
while index<len(s):
      print(s[index])
      index = index + 1

#5) for each loop
for i in s:
      print(i)
```

```
Ex:
---
s = "hai"
#    012
#   -321

i1 = 0
i2 = -len(s)

while i1 < len(s):
      print(f"+ve index {i1} value= {s[i1]} and -ve index {i2} value= {s[i2]}")
      i1 = i1 + 1
      i2 = i2 + 1
```

```
C:\test>py test.py
+ve index 0 value= h and -ve index -3 value= h
+ve index 1 value= a and -ve index -2 value= a
+ve index 2 value= i and -ve index -1 value= i
```

slice operator:
----

we can use slice operator to extract(sub-string) or slice the given string.

syntax:

```
      s[s:s:s]

      s ----> start value
      s ----> stop value
      s ----> step value
```

like range(a), range(a,b) and range(a,b,c)

```
Ex:
---
s = "abcdefg"
#    0123456

print(s) #abcdefg
print(s[0:7:1]) #abcdefg
print(s[0:7:2]) #aceg
print(s[0:7:3]) #adg
print(s[0:6:1]) #abcdef
```

Note: when we are moving from left to right +ve indexes following are def value

```
      1. start ------> 0
      2. stop -------> len(s)
      3. step -------> 1
```

Ex:

```
s = "abcdefg"
#    0123456

print(s) #abcdefg
print(s[0:7:1]) #abcdefg
print(s[:7:1]) #abcdefg
print(s[0::1]) #abcdefg
print(s[0:7:]) #abcdefg
print(s[::]) #abcdefg
```

Note: when we are moving from right to left -ve indexes following are def value

1. start ------> -1
2. stop -------> -(len(s)+1)

Ex:

```
s = "abcdefg"
#    0123456
#   -7654321

print(s) #abcdefg
print(s[-1:-8:-1]) #gfedcba
print(s[-3:-6:-1]) #edc
print(s[:-8:-1]) #gfedcba
print(s[-1::-1]) #gfedcba
print(s[-1:-8:]) #no output
print(s[::]) #abcdefg
print(s[::-1]) #gfedcba
```

```
C:\test>py test.py
abcdefg
gfedcba
edc
gfedcba
gfedcba

abcdefg
gfedcba
```

Note: slice operator never generates error.

Ex:

```
s = "abcdefg"
#    0123456
#   -7654321

print(s) #abcdefg
print(s[-1:-888:-1]) #gfedcba
```

operators on string objects

+	string concatenation
*	string repeatation
in	membership checking
not in	membership checking
<	comparing
<=	comparing
>	comparing
>=	comparing
==	comparing
!=	comparing

```
Ex1:
----
s1 = "abc"
s2 = "def"
print(s1+s2) #abcdef

Ex2:
----
s = "abc"
i = 10
print(s+i) #TypeError: can only concatenate str (not "int") to str

Ex3:
----
s = "abc"
i = 10
print(s+str(i)) #abc10

Ex4:
----
s = "ab"
i1 = 2
i2 = 3
print(s*i1) #abab
print(i2*s) #ababab

Ex5:
----
s = "ab"
i = "3"
print(s*i) #TypeError: can't multiply sequence by non-int of type 'str'

Ex6:
----
s = "ab"
i = "3"
print(s*int(i)) #ababab

Ex7:
----
s = "prakash"
print('a' in s) #True
print('b' in s) #False

Ex8:
----
s = "prakash"
print('k' not in s) #False
print('l' not in s) #True

Ex9:
----
print("abc" < "mno") #True
print("mno" < "mno") #False

Ex10:
-----
print("abc" < "mno") #True
print("mno" < "mno") #False
print("mno" <= "mno") #True

prakash < prasanth
```

```
abc < mno

raj < raju ---> T

common functions
----------------
len(s)                  length of string
max(s)                  max char based on ascii value
min(s)                  min char based on ascii value
sorted(s)         sorted list will all char in asc order
sorted(s,reverse=True)  sorted list will all char in desc order
ord(ch)                 returns ascii value of char
chr(ascii)        returns char value for given ascii

Ex:
---
s = "prakash"

print(s) #prakash
print(len(s)) #7
print(max(s)) #s
print(min(s)) #a
print(sorted(s)) #['a', 'a', 'h', 'k', 'p', 'r', 's']
print(sorted(s,reverse=True)) #['s', 'r', 'p', 'k', 'h', 'a', 'a']

Ex:
---
print(ord('a')) #97
print(ord('A')) #65
print(ord('0')) #48

print(chr(97)) #a
print(chr(65)) #A
print(chr(48)) #0

string specific methods:
------------------------
upper():
--------
it converts the given str into upper case

lower():
--------
it converts the given str into lower case

swapcase():
-----------
it converts lower case into upper case and upper case into lower case

title():
--------
each word's first char will be converted into upper case

capitalize():
-------------
sentence first char will be converted into upper case

Ex:
---
s = "welcome TO pYtHoN PROGRamming"

print(s) #welcome TO pYtHoN PROGRamming
print(s.lower()) #welcome to python programming
print(s.upper()) #WELCOME TO PYTHON PROGRAMMING
print(s.swapcase()) #WELCOME to PyThOn progrAMMING
print(s.title()) #Welcome To Python Programming
print(s.capitalize()) #Welcome to python programming
print(s) #welcome TO pYtHoN PROGRamming
```

```
C:\test>py test.py
welcome TO pYtHoN PROGRamming
welcome to python programming
WELCOME TO PYTHON PROGRAMMING
WELCOME to PyThOn progrAMMING
Welcome To Python Programming
Welcome to python programming
welcome TO pYtHoN PROGRamming
```

count(substr):

it returns number of occurrences of given substring

Ex:

```
s = "abcdabcaba"
print(s.count("a")) #4
print(s.count("b")) #3
print(s.count("c")) #2
print(s.count("d")) #1
print(s.count("e")) #0
```

```
C:\test>py test.py
4
3
2
1
0

C:\test>
```

replace(old,new)

it replaces the occurence of old char with new char

Ex:

```
s = "abcdabcaba"
print(s.replace('a','b')) #bbcdbbcbbb
print(s.replace("ab","x")) #xcdxcxa
```

```
C:\test>py test.py
bbcdbbcbbb
xcdxcxa
```

startswith():

it returns true if the given string starts with another string else false

endswith():

it returns true if the given string ends with another string else false

Ex:

```
s = "python is very easy"
print(s.startswith("java")) #False
print(s.startswith("python")) #True
print(s.endswith("difficult")) #False
print(s.endswith("easy")) #True

index(substring):
-----------------
it returns index of sub-string in the main string, else it raises error

find(substring):
----------------
it returns index of sub-string in the main string, else it returns -1

Ex:
---
s = "python is very easy"
print(s.index("is")) #7
print(s.index("was")) #ValueError: substring not found

Ex:
---
s = "python is very easy"
print(s.find("is")) #7
print(s.find("was")) #-1

split(delimiter)
----------------
it splits the given string based on delimiter.

Ex:
---
s = "python is very easy"
print(s.split(" ")) #['python', 'is', 'very', 'easy']

Ex:
---
s = "01/05/2023"
print(s.split("/")) #['01', '05', '2023']

Ex:
---
s = "08:29:45"
print(s.split(":")) #['08', '29', '45']

seperator.join(list)
--------------------
it takes seperator and join each element in list with given seperator.

Ex:
---
L = ['python', 'is', 'very', 'easy']
print(' '.join(L)) #python is very easy
print(':'.join(L)) #python:is:very:easy

isalnum():
----------
returns True if the given string contains only alpha numeric values else False

Ex:
---
print("1".isalnum()) #True
print("a".isalnum()) #True
print("#".isalnum()) #False

isalpha():
----------
returns True if the given string contains only alphabets else False
```

Ex:

```
print("1".isalpha()) #False
print("a".isalpha()) #True
print("#".isalpha()) #False
```

isdigit():

returns True if the given string contains only digits else False

Ex:

```
print("1".isdigit()) #True
print("a".isdigit()) #False
print("#".isdigit()) #False
```

islower():

returns True if the given string/char is in lower case else False

Ex:

```
print("a".islower()) #True
print("A".islower()) #False
```

isupper():

returns True if the given string/char is in upper case else False

Ex:

```
print("a".isupper()) #False
print("A".isupper()) #True
```

isspace():

returns True if the given string/char contains only space else False

Ex:

```
print(" ".isspace()) #True
print("".isspace()) #False
```

list data structure:

introduction:

==> it is a group of objects of different types.
==> it is represented by using []
==> it is growable (add/remove/update)
==> mutable object
==> it is index based data structure
==> slicing is allowed
==> insertion order is preserved
==> duplicates are allowed.

Ex:

```
L = [10, 10.34, "abc", True, 1+2j]
print(L) #[10, 10.34, "abc", True, 1+2j]
print(type(L)) #<class 'list'>
```

```
C:\test>py test.py
[10, 10.34, 'abc', True, (1+2j)]
<class 'list'>
```

creation of list objects:
----
1) []
2) [obj1, obj2, obj3, obj4,....]
3) list()
4) split()
5) input() with list()
6) eval()

accessing list objects:
----
1) directly
2) index concept
3) slice operator
4) while loop
5) for each loop

Ex:

```
L = [10, 20, 30, 40]

print(L[0:len(L)]) #[10, 20, 30, 40]
print(L[::]) #[10, 20, 30, 40]

#while loop
print("while loop")
index = 0
while index<len(L):
      print(L[index])
      index = index + 1

#for each loop
print("for each loop")
for item in L:
      print(item)
```

nested list objects:
----
a list object within another list object is called as nested list.

Ex:

```
L = [100, 200, [111, 222, 333], 300, 400, 500]
#    0    1     2               3    4     5

print(L[0]) #100
print(L[1]) #200
print(L[2]) #[111, 222, 333]
print(L[3]) #300
print(L[4]) #400
print(L[5]) #500
```

```
C:\test>py test.py
100
200
[111, 222, 333]
300
400
500
```

Ex:

```
L = [100, 200, [111, 222, 333], 300, 400, 500]
#    0    1     2               3    4     5

print(L[0]) #100
print(L[1]) #200
print(L[2]) #[111, 222, 333]
print(L[2][0]) #111
print(L[2][1]) #222
print(L[2][2]) #333
print(L[3]) #300
print(L[4]) #400
print(L[5]) #500
```

```
C:\test>py test.py
100
200
[111, 222, 333]
111
222
333
300
400
500
```

list aliasing:
----

assigning a new reference variable for an existing list object is called as list aliasing.

Ex:

```
L1 = [1, 2, 3, 4, 5]
L2 = L1

print(L1 is L2) #True
```

Ex:

```
L1 = [1, 2, 3, 4, 5]
L2 = L1

print(L1) #[1, 2, 3, 4, 5]
print(L2) #[1, 2, 3, 4, 5]

L1[2] = 999

print(L1) #[1, 2, 999, 4, 5]
print(L2) #[1, 2, 999, 4, 5]
```

list cloning:
----

it is used to create a duplicate copy of existing list object. it is possible by using copy() method.

Ex:

```
L1 = [1, 2, 3, 4, 5]
L2 = L1.copy()

print(L1) #[1, 2, 3, 4, 5]
print(L2) #[1, 2, 3, 4, 5]
print(L1 is L2) #False
```

Ex:

```
L1 = [1, 2, 3, 4, 5]
L2 = L1.copy()

print(L1) #[1, 2, 3, 4, 5]
print(L2) #[1, 2, 3, 4, 5]

L1[2] = 888

print(L1) #[1, 2, 888, 4, 5]
print(L2) #[1, 2, 3, 4, 5]
```

special case in cloning w.r.t nested list

case1:

```
L1 = [11, 22, 33, [111, 222, 333], 44, 55]
L2 = L1.copy()

print(L1) #[11, 22, 33, [111, 222, 333], 44, 55]
print(L2) #[11, 22, 33, [111, 222, 333], 44, 55]

L1[1] = 888

print(L1) #[11, 888, 33, [111, 222, 333], 44, 55]
print(L2) #[11, 22, 33, [111, 222, 333], 44, 55]
```

case2:

```
L1 = [11, 22, 33, [111, 222, 333], 44, 55]
L2 = L1.copy()

print(L1) #[11, 22, 33, [111, 222, 333], 44, 55]
print(L2) #[11, 22, 33, [111, 222, 333], 44, 55]

L1[3][1] = 999

print(L1) #[11, 22, 33, [111, 999, 333], 44, 55]
print(L2) #[11, 22, 33, [111, 999, 333], 44, 55]
```

shallow copy and deep copy

once if copy of list object got created, if it contains any nested list objects, then if we perform any modifications on nested list object, those modifications will be reflected for both list pbjects i.e. nested list objects are shared. this type of copy operation is called as shallow copy or normal copy.

Ex:

```
L1 = [11, 22, 33, [111, 222, 333], 44, 55]
L2 = L1.copy() #shallow copy

print(L1) #[11, 22, 33, [111, 222, 333], 44, 55]
print(L2) #[11, 22, 33, [111, 222, 333], 44, 55]

L1[3][1] = 999

print(L1) #[11, 22, 33, [111, 999, 333], 44, 55]
print(L2) #[11, 22, 33, [111, 999, 333], 44, 55]
```

Ex:

```
import copy
L1 = [11, 22, 33, [111, 222, 333], 44, 55]
L2 = copy.copy(L1) #shallow copy

print(L1) #[11, 22, 33, [111, 222, 333], 44, 55]
print(L2) #[11, 22, 33, [111, 222, 333], 44, 55]

L1[3][1] = 777

print(L1) #[11, 22, 33, [111, 777, 333], 44, 55]
print(L2) #[11, 22, 33, [111, 777, 333], 44, 55]
```

once if copy of list object got created, if it contains any nested list objects, then if we perform any modifications on nested list object, those modifications will be reflected for only one list objects i.e. nested list objects are not shared. this type of copy operation is called as deep copy.

Ex:

```
import copy
L1 = [11, 22, 33, [111, 222, 333], 44, 55]
L2 = copy.deepcopy(L1) #deep copy

print(L1) #[11, 22, 33, [111, 222, 333], 44, 55]
print(L2) #[11, 22, 33, [111, 222, 333], 44, 55]

L1[3][1] = 666

print(L1) #[11, 22, 33, [111, 666, 333], 44, 55]
print(L2) #[11, 22, 33, [111, 222, 333], 44, 55]
```

list comprehension:
----

easiest way to create list objects.

```
syntax1: [expr for i in sequence]
syntax2: [expr for i in sequence if condition]
```

Ex1: increment each element present in a list

```
OL = [1, 2, 3, 4, 5]
NL = [i+1 for i in OL]
print(OL) #[1, 2, 3, 4, 5]
print(NL) #[2, 3, 4, 5, 6]
```

Ex2: find factorial of each element present in a list

```
import math
OL = [1, 2, 3, 4, 5]
NL = [math.factorial(i) for i in OL]
print(OL) #[1, 2, 3, 4, 5]
print(NL) #[1, 2, 6, 24, 120]
```

Ex3: convert every name present in list into upper case

```
OL = ["prakash","raju","ram","somu"]
NL = [i.upper() for i in OL]
print(OL) #[prakash, raju, ram, somu]
print(NL) #[PRAKASH, RAJU, RAM, SOMU]
```

Ex4: extract even numbers from a list

```
OL = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
NL = [i for i in OL if i%2==0]
print(OL) #[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
print(NL) #[2, 4, 6, 8, 10]

common functions on list
------------------------
```

len(L)	length of list
max(L)	max element in list
min(L)	min element in list
sorted(L)	sorted list with all elements in asc order
sorted(L,reverse=True)	sorted list with all elements in desc order
sum(L)	sum of all the elements in list

```
list specific methods:
----------------------
append(object)
--------------
it is used to add an object into list at the ending.

Ex:
---
L = [10, 20, 30, 40]
print(L) #[10, 20, 30, 40]
L.append(50)
print(L) #[10, 20, 30, 40, 50]

insert(index,object)
--------------------
it is used to add an object into list at the given index value

Ex:
---
L = [10, 20, 30, 40]
print(L) #[10, 20, 30, 40]
L.insert(0,999)
print(L) #[999, 10, 20, 30, 40]

remove(object)
--------------
it will remove the given object from the list

Ex:
---
L = [10, 20, 30, 40]
print(L) #[10, 20, 30, 40]
L.remove(20)
print(L) #[10, 30, 40]

pop()
-----
it will remove the element located at last location

Ex:
---
L = [10, 20, 30, 40]
print(L) #[10, 20, 30, 40]
L.pop()
print(L) #[10, 20, 30]

pop(index)
----------
it will remove the element located at given location
```

```
Ex:
---
L = [10, 20, 30, 40]
print(L) #[10, 20, 30, 40]
L.pop(0)
print(L) #[20, 30, 40]

clear()
-------
it will remove all the elements from a list

Ex:
---
L = [10, 20, 30, 40]
print(L) #[10, 20, 30, 40]
L.clear()
print(L) #[]

Sir, remove will perform first occurrence right or all 20 related elements from
list? confised

index(object)
-------------
it returns location of the given object

Ex:
---
L = [10, 20, 30, 40, 10, 20, 20, 10, 20]
print(L)
print(L.index(30)) #2
print(L.index(70)) #Error

count(object)
-------------
it returns number of occurrences of the given object

Ex:
---
L = [10, 20, 30, 40, 10, 20, 20, 10, 20]
print(L)
print(L.count(10)) #3
print(L.count(20)) #4

reverse()
---------
it reverse the given list object

Ex:
--
L = [10, 30, 20, 50, 40]
print(L) #[10, 30, 20, 50, 40]
L.reverse()
print(L) #[40, 50, 20, 30, 10]

sort()
------
it sorts the given list in asc order

Ex:
---
L = [10, 30, 20, 50, 40]
print(L) #[10, 30, 20, 50, 40]
L.sort()
print(L) #[10, 20, 30, 40, 50]
```

sort(reverse=True)

it sorts the given list in desc order

Ex:

```
L = [10, 30, 20, 50, 40]
print(L) #[10, 30, 20, 50, 40]
L.sort(reverse=True)
print(L) #[50, 40, 30, 20, 10]
```

04. tuple data structure

introduction:

==> it is a group of objects of different types.
==> it is represented by using ()
==> it is not growable
==> immutable object
==> it is index based data structure
==> slicing is allowed
==> insertion order is preserved
==> duplicates are allowed.

creation of tuple objects:

1) ()
2) (obj1, obj2, obj3, obj4,....)
3) (obj,)
4) tuple()
5) input() with tuple()
6) eval()

accessing tuple objects:

1) directly
2) index concept
3) slice operator
4) while loop
5) for each loop

nested tuple objects:

a tuple object within another tuple object is called as nested tuple.

tuple aliasing:

assigning a new reference variable for an existing tuple object is called as tuple aliasing.

tuple comprehension:

easiest way to create list objects.

```
syntax1: tuple(expr for i in sequence)
syntax2: tuple(expr for i in sequence if condition)
```

Ex

```
T1 = (1, 2, 3, 4)
T2 = tuple(i+1 for i in T1)
print(T1)
print(T2)
```

common functions on tuple
----

len(L)	length of tuple
max(L)	max element in tuple
min(L)	min element in tuple
sorted(L)	sorted list with all elements in asc order
sorted(L,reverse=True)	sorted list with all elements in desc order
sum(L)	sum of all the elements in tuple

tuple specific methods:
----

index(object)
----

it returns location of the given object

count(object)
----

it returns number of occurrences of the given object

sir internally all inbuilt function we can use whatever using for list, if tuple the convert into list then we can use n sir

tuple packing and tuple unpacking
----

converting individual objects into tuple is called as tuple packing
converting tuple into individual objects is called as tuple unpacking

Ex:

```
a=111
b=222
c=333
t = a,b,c
print(a,type(a)) #111 <class 'int'>
print(b,type(b)) #222 <class 'int'>
print(c,type(c)) #333 <class 'int'>
print(t,type(t)) #(111,222,333) <class 'tuple'>
```

```
C:\test>py test.py
111 <class 'int'>
222 <class 'int'>
333 <class 'int'>
(111, 222, 333) <class 'tuple'>
```

Ex:

```
t = (999,888,777,666)
w,x,y,z = t #tuple unpacking
print(w,type(w)) #999 <class 'int'>
print(x,type(x)) #888 <class 'int'>
print(y,type(y)) #777 <class 'int'>
print(z,type(z)) #666 <class 'int'>
print(t,type(t)) #(999,888,777,666) <class 'tuple'>
```

```
C:\test>py test.py
999 <class 'int'>
888 <class 'int'>
777 <class 'int'>
666 <class 'int'>
(999, 888, 777, 666) <class 'tuple'>
```

05. set data structure
----

introduction:

==> it is a group of objects of different types. (immutable)
==> it is represented by using {}
==> it is growable (add/remove)
==> mutable object
==> it is not index based data structure
==> slicing is not allowed
==> insertion order is not preserved
==> duplicates are not allowed.

creation of set objects:
----
1) set()
2) {obj1, obj2, obj3, obj4,....}
3) input() with set()
4) eval()

accessing set objects:
----
1) directly
2) for each loop

Ex:

```
s = {1, 2, 3, 4}
print(s)
for i in s:
      print(i)
```

common functions on set
----

len(L)	length of set
max(L)	max element in set
min(L)	min element in set
sorted(L)	sorted list with all elements in asc order
sorted(L,reverse=True)	sorted list with all elements in desc order
sum(L)	sum of all the elements in set

```
s = {1,2,{3,4}}
print(s)
TypeError: unhashable type: 'set'
```

set specific methods:
----
add(object)
----
it adds the given object into set

Ex:

```
s = {10, 20, 30}
print(s)
s.add(40)
s.add(50)
print(s)
```

```
C:\test>py test.py
{10, 20, 30}
{40, 10, 50, 20, 30}
```

Ex:

```
s = {10, 20, 30}
print(s)
```

```
s.remove(20)
print(s)

C:\test>py test.py
{10, 20, 30}
{10, 30}

Ex:
---
s = {10, 20, 30}
print(s)
s.remove(40)
print(s)

C:\test>py test.py
{10, 20, 30}
Traceback (most recent call last):
  File "C:\test\test.py", line 3, in <module>
    s.remove(40)
KeyError: 40

Ex:
---
s = {10, 20, 30}
print(s)
s.discard(40)
print(s)

C:\test>py test.py
{10, 20, 30}
{10, 20, 30}

Ex:
---
heros = {"chiranjeevi", "balakrishna", "pawankalyan", "venkatesh", "prabhas"}
politician = {"cbn", "kcr", "jagan", "balakrishna", "pawankalyan"}

print(heros)
#{'prabhas', 'balakrishna', 'venkatesh', 'chiranjeevi', 'pawankalyan'}
print(politician)
#{'balakrishna', 'jagan', 'cbn', 'pawankalyan', 'kcr'}

print(heros.union(politician))
#{'prabhas', 'cbn', 'pawankalyan', 'balakrishna', 'venkatesh', 'jagan',
'chiranjeevi', 'kcr'}
print(heros.intersection(politician))
#{'balakrishna', 'pawankalyan'}

print(heros.difference(politician))
#{'venkatesh', 'chiranjeevi', 'prabhas'}

print(politician.difference(heros))
#{'kcr', 'cbn', 'jagan'}

print(heros.symmetric_difference(politician))
#{'prabhas', 'venkatesh', 'jagan', 'cbn', 'chiranjeevi', 'kcr'}

dictionary data structure or dict data structure:
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
==> collection of individual objects ---> str, list, tuple, set
==> collection of key and value pairs are called as dictionary or dict
==> it is represented by using {}
==> <class 'dict'>
```

```
Ex:
---
d = {1:"One", 2:"Two", 3:"Three", 4:"Four", 5:"Five"}
print(d)
print(type(d))#<class 'dict'>

C:\test>py test.py
{1: 'One', 2: 'Two', 3: 'Three', 4: 'Four', 5: 'Five'}
<class 'dict'>

==> index concept is not allowed but keys are acting as index.
==> duplicate keys are not allowed but values can be duplicated.

Ex:
---
d = {1:"One", 2:"Two", 3:"Three", 4:"Dhoni", 5:"Five", 6:"Dhoni"}
print(d)

C:\test>py test.py
{1: 'One', 2: 'Two', 3: 'Three', 4: 'Dhoni', 5: 'Five', 6: 'Dhoni'}

Ex:
---
d = {1:"One", 2:"Two", 3:"Three", 4:"Dhoni", 5:"Five", 5:"Dhoni"}
print(d)

C:\test>py test.py
{1: 'One', 2: 'Two', 3: 'Three', 4: 'Dhoni', 5: 'Dhoni'}

==> both keys and values must be objects.
==> modifications are allowed.

Ex:
---
d = {1:"One", 2:"Two", 3:"Three", 4:"Dhoni", 5:"Five"}
print(d)
d[2] = "AAA"
print(d)

C:\test>py test.py
{1: 'One', 2: 'Two', 3: 'Three', 4: 'Dhoni', 5: 'Five'}
{1: 'One', 2: 'AAA', 3: 'Three', 4: 'Dhoni', 5: 'Five'}

key ---> object
value -> object

dict methods:
-------------
d[key] = value

it adds key and value pair into the existing dict.

Ex:
---
d = {1:"One", 2:"Two", 3:"Three", 4:"Dhoni", 5:"Five"}
print(d)
d[6] = "Six"
print(d)

del d[key]

it deletes key and value pair from the existing dict

Ex:
```

```
---
d = {1:"One", 2:"Two", 3:"Three", 4:"Dhoni", 5:"Five"}
print(d)
del d[2]
print(d)

C:\test>py test.py
{1: 'One', 2: 'Two', 3: 'Three', 4: 'Dhoni', 5: 'Five'}
{1: 'One', 3: 'Three', 4: 'Dhoni', 5: 'Five'}

clear()
-------
it clear all the key value pairs from dict

Ex:
---
d = {1:"One", 2:"Two", 3:"Three", 4:"Dhoni", 5:"Five"}
print(d)
d.clear()
print(d)

C:\test>py test.py
{1: 'One', 2: 'Two', 3: 'Three', 4: 'Dhoni', 5: 'Five'}
{}

keys()
------
it returns a list of all keys existed in dict

Ex:
---
d = {1:"One", 2:"Two", 3:"Three", 4:"Dhoni", 5:"Five"}
print(d)
print(d.keys())

C:\test>py test.py
{1: 'One', 2: 'Two', 3: 'Three', 4: 'Dhoni', 5: 'Five'}
dict_keys([1, 2, 3, 4, 5])

values()
--------
it returns a list of all values existed in dict

Ex:
---
d = {1:"One", 2:"Two", 3:"Three", 4:"Dhoni", 5:"Five"}
print(d)
print(d.keys())
print(d.values())

C:\test>py test.py
{1: 'One', 2: 'Two', 3: 'Three', 4: 'Dhoni', 5: 'Five'}
dict_keys([1, 2, 3, 4, 5])
dict_values(['One', 'Two', 'Three', 'Dhoni', 'Five'])

items()
-------
it returns a list of all key and value pairs in the form of tuple

Ex:
---
d = {1:"One", 2:"Two", 3:"Three", 4:"Dhoni", 5:"Five"}
print(d)
print(d.keys())
print(d.values())
print(d.items())
```

```
C:\test>py test.py
{1: 'One', 2: 'Two', 3: 'Three', 4: 'Dhoni', 5: 'Five'}
dict_keys([1, 2, 3, 4, 5])
dict_values(['One', 'Two', 'Three', 'Dhoni', 'Five'])
dict_items([(1, 'One'), (2, 'Two'), (3, 'Three'), (4, 'Dhoni'), (5, 'Five')])
```

get(key)

it returns a value associated with given key

Ex:

```
d = {1:"One", 2:"Two", 3:"Three", 4:"Dhoni", 5:"Five"}
print(d)
print(d.get(2)) #Two
print(d.get(6)) #None
print(d.get(5)) #Five
```

```
C:\test>py test.py
{1: 'One', 2: 'Two', 3: 'Three', 4: 'Dhoni', 5: 'Five'}
Two
None
Five
```

get(key,default)

it returns a value associated with given key if not, it returns default value

Ex:

```
d = {1:"One", 2:"Two", 3:"Three", 4:"Dhoni", 5:"Five"}
print(d)
print(d.get(2,"NA")) #Two
print(d.get(6,"NA")) #NA
print(d.get(5,"NA")) #Five
```

```
C:\test>py test.py
{1: 'One', 2: 'Two', 3: 'Three', 4: 'Dhoni', 5: 'Five'}
Two
NA
Five
```

dictionary comprehension:

easiest way to create dict is nothing dict comprehension.

```
{key_expr:val_expr for i in seq}
{key_expr:val_expr for i in seq if cond}
```

Ex:

```
d = {i:i*i for i in range(1,11)}
print(d)
```

```
C:\test>py test.py
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81, 10: 100}
```

Ex:
--

```
import math
```

```
def isprime1(n):
	factors=0
	for i in range(1,n+1):
		if n%i==0:
			factors=factors+1
	return factors==2

d = {i:math.factorial(i) for i in range(1,11) if not isprime1(i)}
print(d)
```

```
C:\test>py test.py
{1: 1, 4: 24, 6: 720, 8: 40320, 9: 362880, 10: 3628800}
```

Chapter 05 ----> List(array) Programs

01. sum of elements present in the given list

version1:

```
def fun(L):
	s = 0
	for i in L:
		s = s + i
	return s

L = [1, 2, 3, 4, 5]
print(f"List={L} and Sum={fun(L)}")
L = [1, 1, 1]
print(f"List={L} and Sum={fun(L)}")
L = []
print(f"List={L} and Sum={fun(L)}")
```

```
C:\test>py test.py
List=[1, 2, 3, 4, 5] and Sum=15
List=[1, 1, 1] and Sum=3
List=[] and Sum=0

C:\test>
```

version2:

```
L = [1, 2, 3, 4, 5]
print(f"List={L} and Sum={sum(L)}")
L = [1, 1, 1]
print(f"List={L} and Sum={sum(L)}")
L = []
print(f"List={L} and Sum={sum(L)}")
```

```
C:\test>py test.py
List=[1, 2, 3, 4, 5] and Sum=15
List=[1, 1, 1] and Sum=3
List=[] and Sum=0
```

02. average of elements present in the list

sum of elements / number of elements

```
def fun(L):
	if len(L)==0:
		return 0.0
	else:
		s = 0
```

```
            for i in L:
                  s=i+s
            return s/len(L)

L = [1, 2, 3, 4, 5]
print(f"List={L} and avg={fun(L)}")
L = [1, 1, 2]
print(f"List={L} and avg={fun(L)}")
L = []
print(f"List={L} and avg={fun(L)}")
```

```
C:\test>py test.py
List=[1, 2, 3, 4, 5] and avg=3.0
List=[1, 1, 2] and avg=1.3333333333333333
List=[] and avg=0.0
```

03. seperate even and odd elements from a list
----

version1:

```
def fun(L):
      L1=[]
      L2=[]
      for i in L:
            if i%2==0:
                  L1.append(i)
            else:
                  L2.append(i)
      print(L)
      print(L1)
      print(L2)

L = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
fun(L)
```

```
C:\test>py test.py
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[2, 4, 6, 8, 10]
[1, 3, 5, 7, 9]
```

version2:

```
def fun(L):
      L1=[x for x in L if x%2==0]
      L2=[x for x in L if x%2!=0]
      print(L)
      print(L1)
      print(L2)

L = [20, 3, 14, 12, 11, 5, 10]
fun(L)
```

```
C:\test>py test.py
[20, 3, 14, 12, 11, 5, 10]
[20, 14, 12, 10]
[3, 11, 5]
```

```
ans = even_odd([1,2,3,4,5,6,7,8,9,10])
for i in ans:
    print(i)

even,odd = fun(L)
print(L)
print(even)
print(odd)

04. generate a lists with even and odd elements in the range of 0 to 10
inclusive
--------------------------------------------------------------------------------
-
def fun():
      L1=[x for x in range(0,10+1) if x%2==0]
      L2=[x for x in range(0,10+1) if x%2!=0]
      return L1,L2


even,odd = fun()
print(even)
print(odd)

C:\test>py test.py
[0, 2, 4, 6, 8, 10]
[1, 3, 5, 7, 9]

05. get smaller elements from a list lesser then the given element x
--------------------------------------------------------------------
L = [9, 11, 15, 12, 3, 7, 14, 10]
x = 10

output ---> [9, 3, 7]

version1:
---------
def fun(l,x):
      ll=[]
      for i in l:
            if i<x:
                  ll.append(i)
      return ll

l = [9, 11, 15, 12, 3, 7, 14, 10]
x = 10
print(fun(l,x))

C:\test>py test.py
[9, 3, 7]

version2:
---------
def fun(l,x):
      return [i for i in l if i<x]

l = [9, 11, 15, 12, 3, 7, 14, 10]
x = 8
print(fun(l,x)) #[3, 7]

C:\test>py test.py
[3, 7]

C:\test>

06. Generate 10 random numbers from 1 to 20 inclusive and append them to the
list
--------------------------------------------------------------------------------
--
import random
def fun():
      L = []
      for i in range(10):
            L.append(random.randint(1,20))
      return L

print(fun())
```

```
C:\test>py test.py
[11, 16, 7, 16, 2, 7, 12, 10, 2, 8]

C:\test>py test.py
[8, 19, 12, 19, 17, 13, 5, 3, 14, 10]

C:\test>py test.py
[20, 3, 1, 3, 11, 19, 18, 4, 8, 16]

C:\test>py test.py
[13, 6, 15, 10, 10, 5, 7, 4, 15, 13]
```

07. swap first and last elements in the list

L = [1, 2, 3, 4, 5]

output: [5, 2, 3, 4, 1]

```
def fun(L):
      L[0],L[-1] = L[-1],L[0]

L = [1, 2, 3, 4, 5]
print(L)
fun(L)
print(L)
```

```
C:\test>py test.py
[1, 2, 3, 4, 5]
[5, 2, 3, 4, 1]

C:\test>
```

8) count number of elements greater than x

L = [1, 2, 3, 4, 5, 6]
x = 2

output: 4

```
def fun(l,x):
      c=0
      for i in l:
            if i>x:
                  c=c+1
      return c

l = [1, 2, 3, 4, 5]
x = 2
print(fun(l,x))
```

```
C:\test>py test.py
3
```

version

```
def fun(l,x):
      return len([i for i in l if i>x])
```

```
l = [1, 2, 3, 4, 5]
x = 3
print(fun(l,x))

C:\test>py test.py
2
```

9) reverse the given list
~~~~~~~~~~~~~~~~~~~~~~~~~

```
l = [1, 2, 3, 4, 5]

output: [5, 4, 3, 2, 1]
```

version1:

```
def fun(l,s,e):
      while s<=e:
            l[s],l[e]=l[e],l[s]
            s=s+1
            e=e-1

l = [1, 2, 3, 4, 5]
print(l)
fun(l,0,len(l)-1)
print(l)

C:\test>py test.py
[1, 2, 3, 4, 5]
[5, 4, 3, 2, 1]
```

version2:

```
def fun(l):
      l.reverse()

l = [1, 2, 3, 4, 5, 6]
print(l)
fun(l)
print(l)

C:\test>py test.py
[1, 2, 3, 4, 5, 6]
[6, 5, 4, 3, 2, 1]
```

version3:

```
def fun(l):
      return l[::-1]

l = [1, 2, 3, 4, 5, 6, 7]
print(l)
print(fun(l))
```

10) Get the index of the given element

```
Ex1:
      L = [11, 22, 33, 44, 55]
      x = 33
      output: 2
      x = 55
      output: 4
      x = 66
      output: None
```
~~~~~~~~~~~~~~~~~~~~~~~~~

```
version1:
--------
def fun(l,x):
      for i in range(len(l)):
            if x==l[i]:
                  return i
      return None


l = [11, 22, 33, 44, 55]
#    0   1   2   3   4
x = 22
print(fun(l,x))
x = 55
print(fun(l,x))
x = 66
print(fun(l,x))

C:\test>py test.py
1
4
None

version2:
---------
def fun(l,x):
      if x in l:
            return l.index(x)
      else:
            return None


l = [11, 22, 33, 44, 55]
#    0   1   2   3   4
x = 22
print(fun(l,x))
x = 55
print(fun(l,x))
x = 66
print(fun(l,x))

C:\test>py test.py
1
4
None

11) Find the largest/max element in a list
------------------------------------------
version1:
---------
def fun(l):
      max = l[0]
      for i in range(1,len(l)):
            if max<l[i]:
                  max=l[i]
      return max


l = [1, 5, 12, 14, 3]
print(fun(l)) #14

version2:
---------
```

```
def fun(l):
      return max(l)


l = [1, 5, 12, 14, 3]
print(fun(l)) #14

12) find second max/largest element
-----------------------------------
version1:
---------
def fun1(l):
      max = l[0]
      for i in range(1,len(l)):
            if max<l[i]:
                  max=l[i]
      return max

def fun2(l):
      max = fun1(l)
      smax = None
      for i in l:
            if i!=max:
                  if smax==None:
                        smax = i
                  else:
                        smax = smax if smax>i else i
      return smax

l = [1, 5, 12, 14, 3]
print(fun2(l)) #12

version2:
---------
def fun(l):
      l.sort()
      return l[-2]

l = [1, 5, 12, 14, 13]
print(fun(l)) #13

13) check if a list is sorted list or not
-----------------------------------------
version1:
---------
def fun(l):
      i = 1
      while i<len(l):
            if l[i]<l[i-1]:
                  return False
            i=i+1
      return True

l = [1, 2, 3, 4, 5]
print(fun(l)) #True
l = [1, 2, 5, 4, 3]
print(fun(l)) #False

version2:
---------
def fun(l):
      ll = sorted(l)
      return l==ll
```

```
l = [1, 2, 3, 4, 5]
print(fun(l)) #True
l = [1, 2, 5, 4, 3]
print(fun(l)) #False
```

14) remove duplicate elements from a list

```
-----------------------------------------
def fun(l):
      ll=[]
      for i in l:
            if i not in ll:
                  ll.append(i)
      return ll

l = [1, 5, 2, 1, 3, 4, 4, 2, 5]
print(l) #[1, 5, 2, 1, 3, 4, 4, 2, 5]
l = fun(l)
print(l) #[1, 5, 2, 3, 4]
```

15) left rotate a list by one element

```
-------------------------------------
[1, 2, 3, 4, 5] -----> [2, 3, 4, 5, 1]

version1:
---------
l = [1, 2, 3, 4, 5]
print(l) #[1, 2, 3, 4, 5]
l = l[1:] + l[0:1]
print(l) #[2, 3, 4, 5, 1]

version2:
---------
l = [1, 2, 3, 4, 5, 6]
print(l) #[1, 2, 3, 4, 5, 6]
l.append(l.pop(0))
print(l) #[2, 3, 4, 5, 6, 1]

version3:
---------
def leftRotateByOne(l):
      n=len(l)
      x = l[0]
      for i in range(1,n):
            l[i-1] = l[i]
      l[n-1] = x

l = [1, 2, 3, 4, 5]
print(l) #[1, 2, 3, 4, 5]
leftRotateByOne(l)
print(l) #[2, 3, 4, 5, 1]

[1,2,3,4,5]
[5,1,2,3,4]
[4,5,1,2,3]
[3,4,5,1,2]
```

this is original list if doing left rotate at 3
then output should be like this sir
[3,4,5,1,2] ,
its correct sir, as i understood sir

16) Right rotate a list by one unit

```
-----------------------------------
version1:
---------
```

```
l = [11, 55, 22, 66, 44]
print(l) #[11, 55, 22, 66, 44]
l= l[-1:-2:-1] + l[0:len(l)-1]
print(l) #[44, 11, 55, 22, 66]

version2:
---------
l = [11, 55, 22, 66, 44]
print(l) #[11, 55, 22, 66, 44]
l.insert(0,l.pop(-1))
print(l) #[44, 11, 55, 22, 66]

version3:
---------
def rightRotateByOne(l):
      n=len(l)
      x=l[n-1]
      for i in range(n-1,0,-1):
            l[i] = l[i-1]
      l[0] = x

l = [1, 2, 3, 4, 5, 6]
print(l)
rightRotateByOne(l)
print(l)

17) left rotate a list by 'd' places
------------------------------------
version1:
---------
L = [1, 2, 3, 4, 5]
print(L)
d = int(input("Enter number of rotations: "))
L = L[d:] + L[:d]
print(L)

C:\test>py test.py
[1, 2, 3, 4, 5]
Enter number of rotations: 1
[2, 3, 4, 5, 1]

C:\test>py test.py
[1, 2, 3, 4, 5]
Enter number of rotations: 2
[3, 4, 5, 1, 2]

version2:
---------
from collections import deque
L = [1, 2, 3, 4, 5, 6]
print(L)
d = int(input("Enter number of rotations: "))
dq = deque(L)
dq.rotate(-d) #-d indicates left rotation
L=list(dq)
print(L)

C:\test>py test.py
[1, 2, 3, 4, 5, 6]
Enter number of rotations: 1
[2, 3, 4, 5, 6, 1]

C:\test>py test.py
[1, 2, 3, 4, 5, 6]
```

```
Enter number of rotations: 2
[3, 4, 5, 6, 1, 2]

C:\test>py test.py
[1, 2, 3, 4, 5, 6]
Enter number of rotations: 3
[4, 5, 6, 1, 2, 3]
```

sir if for version 1 if we take d=5 or more then?

```
n=5 ---> [1,2,3,4,5]
d=1 ---> [2,3,4,5,1]
d=2 ---> [3,4,5,1,2]
d=3 ---> [4,5,1,2,3]
d=4 ---> [5,1,2,3,4]
d=5 ---> [1,2,3,4,5] --> d=d%n=5%5=0
d=6 ---> [2,3,4,5,1] --> d=d%n=6%5=1
```

version3:

```
def leftRotateMethod1(L,d):
      for i in range(0,d):
            L.append(L.pop(0))

L = [1, 2, 3, 4, 5, 6]
print(L)
d = int(input("Enter number of rotations: "))
leftRotateMethod1(L,d)
print(L)
```

```
C:\test>py test.py
[1, 2, 3, 4, 5, 6]
Enter number of rotations: 1
[2, 3, 4, 5, 6, 1]

C:\test>py test.py
[1, 2, 3, 4, 5, 6]
Enter number of rotations: 2
[3, 4, 5, 6, 1, 2]

C:\test>py test.py
[1, 2, 3, 4, 5, 6]
Enter number of rotations: 3
[4, 5, 6, 1, 2, 3]
```

version4:

```
def reverse(L,b,e):
      while b<e:
            L[b],L[e]=L[e],L[b]
            b=b+1
            e=e-1

def leftRotateMethod2(L,d):
      n=len(L)
      reverse(L,0,d-1)
      reverse(L,d,n-1)
      reverse(L,0,n-1)

L = [1, 2, 3, 4, 5, 6]
print(L)
d = int(input("Enter number of rotations: "))
leftRotateMethod2(L,d)
print(L)
```

```
C:\test>py test.py
[1, 2, 3, 4, 5, 6]
Enter number of rotations: 1
[2, 3, 4, 5, 6, 1]
```

18) right rotate a list by 'd' places

version1:

```
L = [1,2,3,4,5]
print(L)
d=int(input("Enter num of rotations: "))
L = L[len(L)-d:] + L[:len(L)-d]
print(L)
```

```
C:\test>py test.py
[1, 2, 3, 4, 5]
Enter num of rotations: 3
[3, 4, 5, 1, 2]
```

version2:

```
from collections import deque
L = [1, 2, 3, 4, 5, 6]
print(L)
d = int(input("Enter number of rotations: "))
dq = deque(L)
dq.rotate(d) #d indicates right rotation
L=list(dq)
print(L)
```

```
C:\test>py test.py
[1, 2, 3, 4, 5, 6]
Enter number of rotations: 2
[5, 6, 1, 2, 3, 4]

C:\test>py test.py
[1, 2, 3, 4, 5, 6]
Enter number of rotations: 3
[4, 5, 6, 1, 2, 3]

C:\test>py test.py
[1, 2, 3, 4, 5, 6]
Enter number of rotations: 4
[3, 4, 5, 6, 1, 2]
```

version3:

```
def rightRotateMethod1(L,d):
      for i in range(0,d):
            L.insert(0,L.pop(-1))

L = [1, 2, 3, 4, 5, 6]
print(L)
d = int(input("Enter number of rotations: "))
rightRotateMethod1(L,d)
print(L)
```

```
C:\test>py test.py
[1, 2, 3, 4, 5, 6]
Enter number of rotations: 1
[6, 1, 2, 3, 4, 5]
```

```
C:\test>py test.py
[1, 2, 3, 4, 5, 6]
Enter number of rotations: 2
[5, 6, 1, 2, 3, 4]

C:\test>py test.py
[1, 2, 3, 4, 5, 6]
Enter number of rotations: 3
[4, 5, 6, 1, 2, 3]
```

```
version4:
---------
def reverse(L,b,e):
      while b<e:
            L[b],L[e]=L[e],L[b]
            b=b+1
            e=e-1

def rightRotateMethod2(L,d):
      n=len(L)
      reverse(L,0,n-1)
      reverse(L,0,d-1)
      reverse(L,d,n-1)

L = [1, 2, 3, 4, 5, 6]
print(L)
d = int(input("Enter number of rotations: "))
rightRotateMethod2(L,d)
print(L)
```

```
C:\test>py test.py
[1, 2, 3, 4, 5, 6]
Enter number of rotations: 1
[6, 1, 2, 3, 4, 5]

C:\test>py test.py
[1, 2, 3, 4, 5, 6]
Enter number of rotations: 2
[5, 6, 1, 2, 3, 4]
```

```
19) merge two lists into third list
-----------------------------------
def fun(l1,l2):
      l=[]
      for i in l1:
            l.append(i)
      for i in l2:
            l.append(i)
      return l


l1 = [1, 3, 5, 2, 4]
l2 = [6, 8, 7, 9, 10]
l3 = fun(l1,l2)
print(l1)
print(l2)
print(l3)
```

```
C:\test>py test.py
[1, 3, 5, 2, 4]
[6, 8, 7, 9, 10]
[1, 3, 5, 2, 4, 6, 8, 7, 9, 10]
```

20) merge two lists into thrid list and sort

```
def fun(l1,l2):
      l=l1+l2
      l.sort()
      return l


l1 = [1, 3, 5, 2, 4]
l2 = [6, 8, 7, 9, 10]
l3 = fun(l1,l2)
print(l1)
print(l2)
print(l3)
```

```
C:\test>py test.py
[1, 3, 5, 2, 4]
[6, 8, 7, 9, 10]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

21) Who has the majority?

Given a list of size n and return an element's position which appears greater than n/2 times.

```
L = [8, 3, 4, 8, 8] ---> n=5/2=2 output: 0 or 3 or 4 --> 8
L = [8, 7, 6, 8, 6, 6, 6, 6] --> n=8/2=4 output: 2 or 4 or 5 or 6 or 7 --> 6
L = [3, 7, 4, 7, 7, 5] ----> n=6/2=3 output: -1
```

```
def fun(L):
      n=len(L)
      for i in range(0,n):
            count=1
            for j in range(i+1,n):
                  if L[i]==L[j]:
                        count=count+1
            if count>n/2:
                  return L[i]
      return -1

print(fun([8, 3, 4, 8, 8])) #8
print(fun([8, 7, 6, 8, 6, 6, 6, 6])) #6
print(fun([3, 7, 4, 7, 7, 5])) #-1
```

22) find union of two list objects

```
def fun(L1,L2):
      L=[]
      for i in L1:
            if i not in L:
                  L.append(i)
      for i in L2:
            if i not in L:
                  L.append(i)
      return L

L1 = [1,2,3,4,5]
L2 = [4,5,6,7,8]
L3 = fun(L1,L2)
print(L1) #[1,2,3,4,5]
print(L2) #[4,5,6,7,8]
print(L3) #[1,2,3,4,5,6,7,8]
```

```
C:\test>py test.py
[1, 2, 3, 4, 5]
[4, 5, 6, 7, 8]
[1, 2, 3, 4, 5, 6, 7, 8]

C:\test>
```

23) find intersection of two list objects

```
def fun(L1,L2):
      L=[]
      for i in L1:
            if i in L2:
                  L.append(i)
      return L

L1 = [1,2,3,4,5]
L2 = [4,5,6,7,8]
L3 = fun(L1,L2)
print(L1) #[1,2,3,4,5]
print(L2) #[4,5,6,7,8]
print(L3) #[4,5]
```

```
C:\test>py test.py
[1, 2, 3, 4, 5]
[4, 5, 6, 7, 8]
[4, 5]

C:\test>
```

Ch06: Matrices Programs
----

1) read and write matrix elements
2) sum of each element in matrix
3) addition of two matrices
4) subtraction of two matrices
5) multiplication of two matrices
6) row wise sum calculation
7) col wise sum calculation
8) sum of diagonal elements
9) sum of opposite diagonal elements
10) transpose of given matrix
11) identity matrix or not
12) swaping of two rows
13) swaping of two cols
14) swpaing of diagonal elements
15) rotate matrix by 90 degrees

1) read and write matrix elements

```
def read(L,m,n):
      for i in range(m):
            TempL = [int(i) for i in input().split()]
            L.append(TempL)

def printm(L,m,n):
      print()
      for i in range(m):
            for j in range(n):
                  print(L[i][j],end=" ")
            print()

m = int(input("Enter number of rows: "))
n = int(input("Enter number of cols: "))
```

```
L = []
read(L,m,n)
printm(L,m,n)
```

```
C:\test>py test.py
Enter number of rows: 3
Enter number of cols: 3
1 2 3
4 5 6
7 8 9

1 2 3
4 5 6
7 8 9
```

2) sum of each element in matrix

```
def sumofelements(L,m,n):
      s=0
      for i in range(m):
            s=s+sum(L[i])
      return s

def read(L,m,n):
      for i in range(m):
            TempL = [int(i) for i in input().split()]
            L.append(TempL)

def printm(L,m,n):
      print(L)
      for i in range(m):
            for j in range(n):
                  print(L[i][j],end=" ")
            print()

m = int(input("Enter number of rows: "))
n = int(input("Enter number of cols: "))
L = []
read(L,m,n)
printm(L,m,n)
print(sumofelements(L,m,n))
```

```
C:\test>py test.py
Enter number of rows: 3
Enter number of cols: 3
1 2 3
4 5 6
7 8 9
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
1 2 3
4 5 6
7 8 9
45
```

3) addition of two matrices

```
def addition(L1,m,n,L2,L3):
      for i in range(m):
            for j in range(n):
                  L3[i][j] = L1[i][j] + L2[i][j]

def init(L,m,n):
      for i in range(m):
            TempL=[0 for i in range(n)]
            L.append(TempL)

def read(L,m,n):
      for i in range(m):
            TempL = [int(i) for i in input().split()]
            L.append(TempL)

def printm(L,m,n):
      #print(L)
      for i in range(m):
            for j in range(n):
                  print(L[i][j],end=" ")
            print()

m = int(input("Enter number of rows: "))
n = int(input("Enter number of cols: "))

L1 = []
read(L1,m,n)

L2 = []
read(L2,m,n)

L3=[]
init(L3,m,n)

addition(L1,m,n,L2,L3)
print("Matrix:A")
printm(L1,m,n)
print("Matrix:B")
printm(L2,m,n)
print("Matrix:C")
printm(L3,m,n)


C:\test>py test.py
Enter number of rows: 3
Enter number of cols: 3
1 2 3
4 5 6
7 8 9
1 1 1
1 1 1
1 1 1
Matrix:A
1 2 3
4 5 6
7 8 9
Matrix:B
1 1 1
1 1 1
1 1 1
Matrix:C
2 3 4
5 6 7
8 9 10
```

4) subtraction of two matrices

```
def subtraction(L1,m,n,L2,L3):
      for i in range(m):
            for j in range(n):
                  L3[i][j] = L1[i][j] - L2[i][j]
```

```
def init(L,m,n):
      for i in range(m):
            TempL=[0 for i in range(n)]
            L.append(TempL)

def read(L,m,n):
      for i in range(m):
            TempL = [int(i) for i in input().split()]
            L.append(TempL)

def printm(L,m,n):
      #print(L)
      for i in range(m):
            for j in range(n):
                  print(L[i][j],end=" ")
            print()

m = int(input("Enter number of rows: "))
n = int(input("Enter number of cols: "))

L1 = []
read(L1,m,n)

L2 = []
read(L2,m,n)

L3=[]
init(L3,m,n)

subtraction(L1,m,n,L2,L3)
print("Matrix:A")
printm(L1,m,n)
print("Matrix:B")
printm(L2,m,n)
print("Matrix:C")
printm(L3,m,n)
```

```
C:\test>py test.py
Enter number of rows: 3
Enter number of cols: 3
1 2 3
4 5 6
7 8 9
1 1 1
1 1 1
1 1 1
Matrix:A
1 2 3
4 5 6
7 8 9
Matrix:B
1 1 1
1 1 1
1 1 1
Matrix:C
0 1 2
3 4 5
6 7 8
```

5) multiplication of two matrices

```
---------------------------------
def multiplication(L1,m,n,L2,L3):
      for i in range(m):
            for j in range(n):
                  for k in range(n):
                        L3[i][j] = L3[i][j] + L1[i][k] * L2[k][j]

def init(L,m,n):
      for i in range(m):
            TempL=[0 for i in range(n)]
            L.append(TempL)

def read(L,m,n):
      for i in range(m):
            TempL = [int(i) for i in input().split()]
            L.append(TempL)

def printm(L,m,n):
      #print(L)
      for i in range(m):
            for j in range(n):
                  print(L[i][j],end=" ")
            print()

m = int(input("Enter number of rows: "))
n = int(input("Enter number of cols: "))

L1 = []
read(L1,m,n)

L2 = []
read(L2,m,n)

L3=[]
init(L3,m,n)

multiplication(L1,m,n,L2,L3)
print("Matrix:A")
printm(L1,m,n)
print("Matrix:B")
printm(L2,m,n)
print("Matrix:C")
printm(L3,m,n)
```

```
C:\test>py test.py
Enter number of rows: 2
Enter number of cols: 2
1 2
3 4
5 6
7 8
Matrix:A
1 2
3 4
Matrix:B
5 6
7 8
Matrix:C
19 22
43 50
```

6) row wise sum calculation

```
----------------------------
def rowwisesum(L1,m,n):
      for i in range(m):
            s=0
            for j in range(n):
                  s=s+L1[i][j]
            print(f"{i} row sum:{s}")

def init(L,m,n):
      for i in range(m):
            TempL=[0 for i in range(n)]
            L.append(TempL)

def read(L,m,n):
      for i in range(m):
            TempL = [int(i) for i in input().split()]
            L.append(TempL)

def printm(L,m,n):
      #print(L)
      for i in range(m):
            for j in range(n):
                  print(L[i][j],end=" ")
            print()

m = int(input("Enter number of rows: "))
n = int(input("Enter number of cols: "))

L1 = []
read(L1,m,n)


print("Matrix:A")
printm(L1,m,n)
rowwisesum(L1,m,n)
```

```
C:\test>py test.py
Enter number of rows: 3
Enter number of cols: 3
1 2 3
4 5 6
7 8 9
Matrix:A
1 2 3
4 5 6
7 8 9
0 row sum:6
1 row sum:15
2 row sum:24
```

7) col wise sum calculation

```
def colwisesum(L1,m,n):
      for i in range(m):
            s=0
            for j in range(n):
                  s=s+L1[j][i]
            print(f"{i} col sum:{s}")

def init(L,m,n):
      for i in range(m):
            TempL=[0 for i in range(n)]
            L.append(TempL)

def read(L,m,n):
      for i in range(m):
            TempL = [int(i) for i in input().split()]
            L.append(TempL)
```

```
def printm(L,m,n):
      #print(L)
      for i in range(m):
            for j in range(n):
                  print(L[i][j],end=" ")
            print()

m = int(input("Enter number of rows: "))
n = int(input("Enter number of cols: "))

L1 = []
read(L1,m,n)


print("Matrix:A")
printm(L1,m,n)
colwisesum(L1,m,n)
```

```
C:\test>py test.py
Enter number of rows: 3
Enter number of cols: 3
1 2 3
4 5 6
7 8 9
Matrix:A
1 2 3
4 5 6
7 8 9
0 col sum:12
1 col sum:15
2 col sum:18
```

8) sum of diagonal elements

```
def diagonalsum(L1,m,n):
      s=0
      for i in range(m):
            for j in range(n):
                  if i==j:
                        s=s+L1[j][i]
      return s

def init(L,m,n):
      for i in range(m):
            TempL=[0 for i in range(n)]
            L.append(TempL)

def read(L,m,n):
      for i in range(m):
            TempL = [int(i) for i in input().split()]
            L.append(TempL)

def printm(L,m,n):
      #print(L)
      for i in range(m):
            for j in range(n):
                  print(L[i][j],end=" ")
            print()

m = int(input("Enter number of rows: "))
n = int(input("Enter number of cols: "))

L1 = []
read(L1,m,n)
```

```
print("Matrix:A")
printm(L1,m,n)
print(diagonalsum(L1,m,n))
```

```
C:\test>py test.py
Enter number of rows: 3
Enter number of cols: 3
1 2 3
4 5 6
7 8 9
Matrix:A
1 2 3
4 5 6
7 8 9
15
```

9) sum of opposite diagonal elements

```
def diagonalsum(L1,m,n):
      s=0
      for i in range(m):
            s=s+L1[i][i]
      return s

def opdiagonalsum(L1,m,n):
      s=0
      for i in range(m):
            s=s+L1[i][m-i-1] #where m is row
      return s

def init(L,m,n):
      for i in range(m):
            TempL=[0 for i in range(n)]
            L.append(TempL)

def read(L,m,n):
      for i in range(m):
            TempL = [int(i) for i in input().split()]
            L.append(TempL)

def printm(L,m,n):
      #print(L)
      for i in range(m):
            for j in range(n):
                  print(L[i][j],end=" ")
            print()

m = int(input("Enter number of rows: "))
n = int(input("Enter number of cols: "))

L1 = []
read(L1,m,n)


print("Matrix:A")
printm(L1,m,n)
print(diagonalsum(L1,m,n))
print(opdiagonalsum(L1,m,n))
```

```
C:\test>py test.py
Enter number of rows: 3
Enter number of cols: 3
```

```
1 2 3
4 5 6
7 8 9
Matrix:A
1 2 3
4 5 6
7 8 9
15
15

C:\test>py test.py
Enter number of rows: 3
Enter number of cols: 3
1 0 0
0 1 0
0 0 1
Matrix:A
1 0 0
0 1 0
0 0 1
3
1
```

```
rishikesh query
---------------
def diagonalsum(L1,m,n):
      s=0
      for i in range(m):
            s=s+L1[i][i]
      return s

def opdiagonalsum(L1,m,n):
      s=0
      for i in range(m):
            s=s+L1[i][m-i-1] #where m is row
      return s

def spdiagonalsum(L1,m,n):
      s=0
      for i in range(m):
            s=s+L1[i][i]+L1[i][m-i-1]
      return s-L1[m//2][n//2]

def init(L,m,n):
      for i in range(m):
            TempL=[0 for i in range(n)]
            L.append(TempL)

def read(L,m,n):
      for i in range(m):
            TempL = [int(i) for i in input().split()]
            L.append(TempL)

def printm(L,m,n):
      #print(L)
      for i in range(m):
            for j in range(n):
                  print(L[i][j],end=" ")
            print()

m = int(input("Enter number of rows: "))
n = int(input("Enter number of cols: "))

L1 = []
```

```
read(L1,m,n)


print("Matrix:A")
printm(L1,m,n)
print(diagonalsum(L1,m,n))
print(opdiagonalsum(L1,m,n))
print(spdiagonalsum(L1,m,n))
```

```
C:\test>py test.py
Enter number of rows: 3
Enter number of cols: 3
1 2 3
4 5 6
7 8 9
Matrix:A
1 2 3
4 5 6
7 8 9
15
15
25
```

10) transpose of given matrix

```
def printmm(L,m,n):
      #print(L)
      for i in range(m):
            for j in range(n):
                  print(L[j][i],end=" ")
            print()

def read(L,m,n):
      for i in range(m):
            TempL = [int(i) for i in input().split()]
            L.append(TempL)

def printm(L,m,n):
      #print(L)
      for i in range(m):
            for j in range(n):
                  print(L[i][j],end=" ")
            print()



m = int(input("Enter number of rows: "))
n = int(input("Enter number of cols: "))

L1 = []
read(L1,m,n)


print("Matrix:A")
printm(L1,m,n)
printmm(L1,m,n)
```

```
C:\test>py test.py
Enter number of rows: 3
Enter number of cols: 3
1 2 3
4 5 6
7 8 9
Matrix:A
```

```
1 2 3
4 5 6
7 8 9
1 4 7
2 5 8
3 6 9
```

11) identity matrix or not

```
#identity matrix or not
def fun(L,m,n):
      for i in range(m):
            for j in range(n):
                  if i==j and L[i][j]!=1:
                        return False
                  if i!=j and L[i][j]!=0:
                        return False
      return True

def read(L,m,n):
      for i in range(m):
            TempL = [int(i) for i in input().split()]
            L.append(TempL)

def printm(L,m,n):
      #print(L)
      for i in range(m):
            for j in range(n):
                  print(L[i][j],end=" ")
            print()

m = int(input("Enter number of rows: "))
n = int(input("Enter number of cols: "))

L1 = []
read(L1,m,n)

print("Matrix:A")
printm(L1,m,n)
print(fun(L1,m,n))
```

```
C:\test>py test.py
Enter number of rows: 3
Enter number of cols: 3
1 2 3
4 5 6
7 8 9
Matrix:A
1 2 3
4 5 6
7 8 9
False

C:\test>py test.py
Enter number of rows: 3
Enter number of cols: 3
1 0 0
0 1 0
0 0 1
Matrix:A
1 0 0
0 1 0
0 0 1
True
```

```
12) swaping of two rows
-----------------------
#swap two rows
def fun(L,m,n,x,y):
      L[x-1],L[y-1]=L[y-1],L[x-1]

def read(L,m,n):
      for i in range(m):
            TempL = [int(i) for i in input().split()]
            L.append(TempL)

def printm(L,m,n):
      #print(L)
      for i in range(m):
            for j in range(n):
                  print(L[i][j],end=" ")
            print()

m = int(input("Enter number of rows: "))
n = int(input("Enter number of cols: "))

L1 = []
read(L1,m,n)

print("Matrix:A")
printm(L1,m,n)
fun(L1,m,n,1,3) #1st and 3rd row
print("Updated Matrix:A")
printm(L1,m,n)

C:\test>py test.py
Enter number of rows: 4
Enter number of cols: 4
1 1 1 1
2 2 2 2
3 3 3 3
9 9 9 9
Matrix:A
1 1 1 1
2 2 2 2
3 3 3 3
9 9 9 9
Updated Matrix:A
3 3 3 3
2 2 2 2
1 1 1 1
9 9 9 9

13) swaping of two cols
-----------------------
#swap two cols
def fun(L,m,n,x,y):
      for i in range(m):
            L[i][x-1],L[i][y-1] = L[i][y-1],L[i][x-1]

def read(L,m,n):
      for i in range(m):
            TempL = [int(i) for i in input().split()]
            L.append(TempL)

def printm(L,m,n):
      #print(L)
      for i in range(m):
            for j in range(n):
                  print(L[i][j],end=" ")
            print()

m = int(input("Enter number of rows: "))
n = int(input("Enter number of cols: "))

L1 = []
read(L1,m,n)

print("Matrix:A")
printm(L1,m,n)
fun(L1,m,n,1,4) #1col and 4th col
print("Updated Matrix:A")
printm(L1,m,n)
```

```
C:\test>py test.py
Enter number of rows: 4
Enter number of cols: 4
1 1 1 1
2 2 2 2
3 4 5 6
1 2 3 4
Matrix:A
1 1 1 1
2 2 2 2
3 4 5 6
1 2 3 4
Updated Matrix:A
1 1 1 1
2 2 2 2
6 4 5 3
4 2 3 1
```

```
14) swpaing of diagonal elements
--------------------------------
#swpaing of diagonal elements
def fun(L,m,n):
      for i in range(m):
            L[i][i],L[i][m-i-1] = L[i][m-i-1],L[i][i]

def read(L,m,n):
      for i in range(m):
            TempL = [int(i) for i in input().split()]
            L.append(TempL)

def printm(L,m,n):
      #print(L)
      for i in range(m):
            for j in range(n):
                  print(L[i][j],end=" ")
            print()

m = int(input("Enter number of rows: "))
n = int(input("Enter number of cols: "))

L1 = []
read(L1,m,n)

print("Matrix:A")
printm(L1,m,n)
fun(L1,m,n)
print("Updated Matrix:A")
printm(L1,m,n)
```

```
C:\test>py test.py
Enter number of rows: 4
Enter number of cols: 4
1 0 0 0
0 1 0 0
0 0 1 0
0 0 0 1
Matrix:A
1 0 0 0
0 1 0 0
0 0 1 0
0 0 0 1
Updated Matrix:A
0 0 0 1
0 0 1 0
0 1 0 0
1 0 0 0

15) rotate matrix by 90 degrees
-------------------------------
1 2 3
4 5 6
7 8 9

def fun(L,m,n):
      TL = [[0,0,0],[0,0,0],[0,0,0]]
      for i in range(m):
            for j in range(n):
                  TL[i][j] = L[j][i]
      printmm(TL,m,n)

def read(L,m,n):
      for i in range(m):
            TempL = [int(i) for i in input().split()]
            L.append(TempL)

def printm(L,m,n):
      #print(L)
      for i in range(m):
            for j in range(n):
                  print(L[i][j],end=" ")
            print()

def printmm(L,m,n):
      #print(L)
      for i in range(m-1,-1,-1):#i=3,i>=0,i--
            for j in range(n):
                  print(L[i][j],end=" ")
            print()

m = int(input("Enter number of rows: "))
n = int(input("Enter number of cols: "))

L1 = []
read(L1,m,n)

print("Matrix:A")
printm(L1,m,n)
fun(L1,m,n)
#print("Updated Matrix:A")
#printm(L1,m,n)

C:\test>py test.py
```

```
Enter number of rows: 3
Enter number of cols: 3
1 2 3
4 5 6
7 8 9
Matrix:A
1 2 3
4 5 6
7 8 9
3 6 9
2 5 8
1 4 7
```

Ch07: Recursion

introduction:

the process in which a function calls itself is called as recursion, the function which is invoked in this process is called as recursive function.

direct recursion ----> a function which calls same function

indirect recursion --> a function calles fun1() and fun1 calls fun2() fun2 calls fun1

finate recursion ---> terminate at a point
infinate recursion--> never terminates (RecursionError --> Runtime Error)

recursion
recursive function
direct
indirect
finate
infinate

Ex:
- binary search
- quick sort
- merge sort
- factorial
- prime number
- linked list
- trees
- graphs
- etc

Example

```
def fun():
      print("Prakash")
      fun()

fun()

#RecursionError: maximum recursion depth exceeded while calling a Python object
```

Example

```
def fun(n):
      if n==0:
            return
      print("Prakash")
      fun(n-1)
fun(3)

C:\test>py test.py
Prakash
Prakash
Prakash

C:\test>

Ex:
---
def fun(n):
      if n<=0:
            return
      print("Prakash")
      fun(n-1)

fun(-3)

output:
-------
nothing

structure/syntax of recursion:
------------------------------
def fun():
    base condition
    -------------
    -------------
    recursive call (call to fun()) with atleast one change to parameter
    once if we change the parameter value then it satifies base condition
    -------------
    -------------

Need for recursion:
-------------------
Recursion is the amazing technique with the help of which we can reduce length
of our code and make it easier to read and write.

properties of recursion:
------------------------
* perform some operation multiple times with diff input
* we will try with small inputs to make problem smaller
* base condition is required to return the solution for smallest problem

Applications of recursions:
---------------------------
linked lists
trees
graphs
chess board game
sudoku game
candy crush game
math applications (prime, factorial, ncr, npr, lcm, gcd etc)
sorting
searching etc

mathematical interpretation of recursion
----------------------------------------
math
----
f(n) = 1 + 2 + 3 + 4 + 5 + .... + n
```

recursion

```
f(n) = 1          if n==1
f(n) = n + f(n-1) if n>1

f(1) = 1
f(2) = 2 + 1
f(3) = 3 + 2 + 1
f(4) = 4 + 3 + 2 + 1
f(5) = 5 + 4 + 3 + 2 + 1
f(100) = 100 + f(99)
```

memory management in recursion

```
=> It uses more memory.
=> It uses stack data structure.
=> LIFO last - in - first - out
=> stack contains complete information about function calls with parameters
```

base condition:

one critical requirement of recursive functions is the termination point or base condition. Every recursive function must have a base condition to make sure that the recursion process will terminate. missing base case results in unexpected behavior. (RecursionError)

Example: Factorial of the given number

```
n = 4 -----> 24
n = 3 -----> 6
n = 0 -----> 1

def fact(n):
      if n==0:
            return 1
      return n*fact(n-1)

print(fact(4)) #24
print(fact(3)) #6
print(fact(2)) #2
print(fact(1)) #1
print(fact(0)) #1
```

```
C:\test>py test.py
24
6
2
1
1
```

Example: Fibanocci Number

```
0 1 1 2 3 5 8 13 21 34 .....

n = 4 ------> 3
n = 3 ------> 2
n = 1 ------> 1
n = 0 ------> 0

def fib(n):
      if n==0:
            return 0
      if n==1:
            return 1
      return fib(n-1) + fib(n-2)

print(fib(6)) #8

Tail Recursion:
~~~~~~~~~~~~~~~
a recursive function is called tail recursive if the function does not do any
thing after the last recursive call.

Ex:
---
def fun(n):
      if n<=0:
            return
      print(n,end="")
      fun(n-1)

fun(3)

C:\test>py test.py
321

Example for non-tail recursion
------------------------------
def fun(n):
      if n<=0:
            return
      fun(n-1)
      print(n,end="")

fun(3)

C:\test>py test.py
123

Ex1: print numbers from 1 to n
------------------------------
def fun(n):
      if n<=0:
            return
      fun(n-1)
      print(n)

fun(3)

C:\test>py test.py
1
2
3

Ex2: print numbers from n to 1
------------------------------
def fun(n):
      if n<=0:
            return
      print(n)
      fun(n-1)

fun(3)

C:\test>py test.py
3
```

```
2
1
```

Ex3: sum of digits

```
def fun(n):
      if n<10:
            return n
      return n%10+fun(n//10)

print(fun(253)) #10
```

```
C:\test>py test.py
10
```

Ex4: sum of n natural numbers

```
def fun(n):
      if n==0:
            return 0
      return n+fun(n-1)

print(fun(10)) #55
```

```
C:\test>py test.py
55
```

Ex5: string paliandrome

```
def pali(s,start,end):
      if start>=end:
            return True
      return s[start]==s[end] and pali(s,start+1,end-1)

print(pali("sir",0,2)) #False
print(pali("madam",0,4)) #True
```

```
C:\test>py test.py
False
True
```

Ex6: factorial of the given number

```
def fact(n):
      if n==0:
            return 1
      return n*fact(n-1)

print(fact(5))
print(fact(4))
print(fact(0))
```

```
C:\test>py test.py
120
24
1
```

Ex7: calculate a power b

```
def power(a,b):
      if b==0:
            return 1
      return a*power(a,b-1)
```

```
print(power(2,0))
print(power(2,1))
print(power(2,2))
print(power(2,3))
print(power(2,4))
print(power(2,5))

C:\test>py test.py
1
2
4
8
16
32

Ex8: prime number application
-----------------------------
def prime(n,i):
      if i==1:
            return True
      elif n%i==0:
            return False
      return prime(n,i-1)


print(prime(4,4//2))
print(prime(5,5//2))
print(prime(6,6//2))

C:\test>py test.py
False
True
False

Ex9: fibonacci seq
-------------------
def fib(n):
      if n==0 or n==1:
            return n
      return fib(n-1) + fib(n-2)

for i in range(11):
      print(fib(i),end=" ")

C:\test>py test.py
0 1 1 2 3 5 8 13 21 34 55

Ex10: reverse of string
-----------------------
def fun(s):
      if len(s)==0 or len(s)==1:
            return s
      return fun(s[1:])+s[0]

print(fun("prakash"))

C:\test>py test.py
hsakarp

Ex: permutations of a given string
----------------------------------
"a" ----> "a"
"ab" ---> "ab", "ba"
"abc" --> "abc", "acb", "bac", "bca", "cab", "cba"
```

```
n ------> n!

def permutations(s,ans):
      if len(s)==0:
            print(ans)
            return
      for i in range(0,len(s)):
            permutations(s[:i]+s[i+1:], ans+s[i])

permutations("abc","")

C:\test>py test.py
abc
acb
bac
bca
cab
cba

Ex:
---
def permutations(s,ans):
      if len(s)==0:
            global c
            c=c+1
            print(ans)
            return
      for i in range(0,len(s)):
            permutations(s[:i]+s[i+1:], ans+s[i])

c=0
permutations("abc","")
print(c)

C:\test>py test.py
abc
acb
bac
bca
cab
cba
6

Ex: subsets of the given string
-------------------------------
"a" ----> "", "a"
"ab" ---> "", "a", "b", "ab"
"abc" --> "", "a", "b", "c", "ab", "ac", "bc", "abc"

n ------> 2^n

def subsets(s,ans,index):
      if index==len(s):
            if len(ans)==0:
                  print("null")
            else:
                  print(ans)
            return
      subsets(s,ans+s[index],index+1)
      subsets(s,ans,index+1)

subsets("abc","",0)

C:\test>py test.py
```

```
abc
ab
ac
a
bc
b
c
null
```

Ex: Towers of Hanoi

Rules

1) Only one disc moves at a time.
2) No larger disc above smaller disc.
3) Only top disc of the tower can be moved

```
def towers(n,src,temp,dest):
      if n==1:
            print(f"move disc 1 from {src} to {dest}")
      else:
            towers(n-1,src,dest,temp)
            print(f"move disc {n} from {src} to {dest}")
            towers(n-1,temp,src,dest)

n=int(input("Enter number of discs: "))
towers(n,"Src","Temp","Dest")
```

```
C:\test>py test.py
Enter number of discs: 1
move disc 1 from Src to Dest

C:\test>py test.py
Enter number of discs: 2
move disc 1 from Src to Temp
move disc 2 from Src to Dest
move disc 1 from Temp to Dest

C:\test>py test.py
Enter number of discs: 3
move disc 1 from Src to Dest
move disc 2 from Src to Temp
move disc 1 from Dest to Temp
move disc 3 from Src to Dest
move disc 1 from Temp to Src
move disc 2 from Temp to Dest
move disc 1 from Src to Dest

C:\test>py test.py
Enter number of discs: 4
move disc 1 from Src to Temp
move disc 2 from Src to Dest
move disc 1 from Temp to Dest
move disc 3 from Src to Temp
move disc 1 from Dest to Src
move disc 2 from Dest to Temp
move disc 1 from Src to Temp
move disc 4 from Src to Dest
move disc 1 from Temp to Dest
move disc 2 from Temp to Src
move disc 1 from Dest to Src
move disc 3 from Temp to Dest
move disc 1 from Src to Temp
move disc 2 from Src to Dest
move disc 1 from Temp to Dest

C:\test>py test.py
Enter number of discs: 5
move disc 1 from Src to Dest
move disc 2 from Src to Temp
move disc 1 from Dest to Temp
move disc 3 from Src to Dest
move disc 1 from Temp to Src
move disc 2 from Temp to Dest
move disc 1 from Src to Dest
move disc 4 from Src to Temp
move disc 1 from Dest to Temp
move disc 2 from Dest to Src
move disc 1 from Temp to Src
move disc 3 from Dest to Temp
move disc 1 from Src to Dest
move disc 2 from Src to Temp
move disc 1 from Dest to Temp
move disc 5 from Src to Dest
move disc 1 from Temp to Src
move disc 2 from Temp to Dest
move disc 1 from Src to Dest
move disc 3 from Temp to Src
move disc 1 from Dest to Temp
move disc 2 from Dest to Src
move disc 1 from Temp to Src
move disc 4 from Temp to Dest
move disc 1 from Src to Dest
move disc 2 from Src to Temp
move disc 1 from Dest to Temp
move disc 3 from Src to Dest
move disc 1 from Temp to Src
move disc 2 from Temp to Dest
move disc 1 from Src to Dest

C:\test>py test.py
Enter number of discs: 6
move disc 1 from Src to Temp
move disc 2 from Src to Dest
move disc 1 from Temp to Dest
move disc 3 from Src to Temp
move disc 1 from Dest to Src
move disc 2 from Dest to Temp
move disc 1 from Src to Temp
move disc 4 from Src to Dest
move disc 1 from Temp to Dest
move disc 2 from Temp to Src
move disc 1 from Dest to Src
move disc 3 from Temp to Dest
move disc 1 from Src to Temp
move disc 2 from Src to Dest
move disc 1 from Temp to Dest
move disc 5 from Src to Temp
move disc 1 from Dest to Src
move disc 2 from Dest to Temp
move disc 1 from Src to Temp
move disc 3 from Dest to Src
move disc 1 from Temp to Dest
move disc 2 from Temp to Src
move disc 1 from Dest to Src
move disc 4 from Dest to Temp
move disc 1 from Src to Temp
move disc 2 from Src to Dest
move disc 1 from Temp to Dest
move disc 3 from Src to Temp
move disc 1 from Dest to Src
move disc 2 from Dest to Temp
move disc 1 from Src to Temp
move disc 6 from Src to Dest
move disc 1 from Temp to Dest
move disc 2 from Temp to Src
move disc 1 from Dest to Src
move disc 3 from Temp to Dest
move disc 1 from Src to Temp
move disc 2 from Src to Dest
move disc 1 from Temp to Dest
move disc 4 from Temp to Src
move disc 1 from Dest to Src
move disc 2 from Dest to Temp
move disc 1 from Src to Temp
move disc 3 from Dest to Src
move disc 1 from Temp to Dest
move disc 2 from Temp to Src
move disc 1 from Dest to Src
move disc 5 from Temp to Dest
move disc 1 from Src to Temp
move disc 2 from Src to Dest
move disc 1 from Temp to Dest
move disc 3 from Src to Temp
move disc 1 from Dest to Src
move disc 2 from Dest to Temp
move disc 1 from Src to Temp
move disc 4 from Src to Dest
move disc 1 from Temp to Dest
move disc 2 from Temp to Src
move disc 1 from Dest to Src
move disc 3 from Temp to Dest
move disc 1 from Src to Temp
move disc 2 from Src to Dest
move disc 1 from Temp to Dest
```

Chapter:08 --> Backtracking

introduction:

Number locks
path finding
sudoku

it is a method by which a solution found by moving/searching through large volume of data with some boundary conditions.

Ex:

- Number Locks
- Rat in maze
- N-Queens
- Grid Based Problems
- Sudoku etc

Types of backtracking

1) decision based problems -----------> yes/no
2) optimized solution based problems -> only one solution which is best
3) enumeration based problems --------> all possible solutions

Backtracking with list

```
def change(L,index,value):
      if index==len(L):
            print(L) #[1,2,3,4,5]
            return
      L[index] = value
      change(L,index+1,value+1) #recursion
      L[index]=L[index]-2 #backtracking

L = [0, 0, 0, 0, 0]
print(L) #[0, 0, 0, 0, 0]
change(L,0,1)
print(L) #[-1, 0, 1, 2, 3]
'''
C:\test>py test.py
[0, 0, 0, 0, 0]
[1, 2, 3, 4, 5]
[-1, 0, 1, 2, 3]
'''

N-Queens
~~~~~~~~
Rules:
1. we have place each queen in each row.
2. no two queens should attack each other.

def issafe(board,row,col):
      #vertical up
      i=row-1
      while i>=0:
            if board[i][col]=='Q':
                  return False
            i=i-1
      #diagonal left up
      i=row-1
      j=col-1
      while i>=0 and j>=0:
            if board[i][j]=='Q':
                  return False
            i=i-1
            j=j-1
      #diagonal right up
      i=row-1
      j=col+1
      while i>=0 and j<len(board):
            if board[i][j]=='Q':
                  return False
            i=i-1
            j=j+1
      return True

def nqueens(board,row):
      if row==len(board):
            global c
            c=c+1
            printboard(board)
            return
      for j in range(len(board)):
            if issafe(board,row,j):
                  board[row][j] = 'Q'
                  nqueens(board,row+1) #recursion
                  board[row][j] = 'X' #backtracking

def printboard(board):
      print("-----chessboard-----")
```

```
    for i in range(len(board)):
        for j in range(len(board)):
            print(board[i][j],end=" ")
        print()
    print("--------------------")

c=0
n=5
board = [['X' for i in range(n)] for i in range(n)]
nqueens(board,0)
print("num of solutions:",c)
'''
C:\test>py test.py
-----chessboard-----
Q X X X X
X X Q X X
X X X X Q
X Q X X X
X X X Q X
--------------------
-----chessboard-----
Q X X X X
X X X Q X
X Q X X X
X X X X Q
X X Q X X
--------------------
-----chessboard-----
X Q X X X
X X X Q X
Q X X X X
X X Q X X
X X X X Q
--------------------
-----chessboard-----
X Q X X X
X X X X Q
X X Q X X
Q X X X X
X X X Q X
--------------------
-----chessboard-----
X X Q X X
Q X X X X
X X X Q X
X Q X X X
X X X X Q
--------------------
-----chessboard-----
X X Q X X
X X X X Q
X Q X X X
X X X Q X
Q X X X X
--------------------
-----chessboard-----
X X X Q X
Q X X X X
X X Q X X
X X X X Q
X Q X X X
--------------------
-----chessboard-----
X X X Q X
```

```
X Q X X X
X X X X Q
X X Q X X
Q X X X X
--------------------
-----chessboard-----
X X X X Q
X Q X X X
X X X Q X
Q X X X X
X X Q X X
--------------------
-----chessboard-----
X X X X Q
X X Q X X
Q X X X X
X X X Q X
X Q X X X
--------------------
num of solutions: 10
'''
```

Grid Ways
----

find number of ways to reach from (0,0) to (m-1,n-1) in a mxn grid(matrix).
we can move only in right direction and down direction.

```
def gridways(i,j,m,n):
    if i==m-1 and j==n-1:
        return 1
    if i==m or j==n:
        return 0
    val1 = gridways(i+1,j,m,n)
    val2 = gridways(i,j+1,m,n)
    return val1+val2

m = 3
n = 3
print(gridways(0,0,m,n))
```

```
C:\test>py test.py
6
```

Suduko Problem

```
def issafe(sudoku,row,col,d):
    #case1: row
    for i in range(0,9):
        if sudoku[i][col]==d:
            return False
    #case2: col
    for j in range(0,9):
        if sudoku[row][j]==d:
            return False
    #case3: grid
    sr = (row//3)*3
    sc = (col//3)*3
    for i in range(sr,sr+3):
        for j in range(sc,sc+3):
            if sudoku[i][j]==d:
                return False
    return True

def sudokusolver(sudoku,row,col):
      if row==9 and col==0:
            return True
      nextrow = row
      nextcol = col+1
      if col+1==9:
            nextrow = row+1
            nextcol = 0
      if sudoku[row][col]!=0:
            return sudokusolver(sudoku,nextrow,nextcol)
      for d in range(1,10):
            if issafe(sudoku,row,col,d):
                  sudoku[row][col]=d
                  if sudokusolver(sudoku,nextrow,nextcol): #recursion
                        return True
                  sudoku[row][col]=0 #backtracking
      return False

def printsudoku(sudoku):
      for i in range(len(sudoku)):
            for j in range(len(sudoku)):
                  print(sudoku[i][j],end=" ")
            print()

sudoku = [
                  [0, 0, 0, 0, 6, 8, 0, 0, 3],
                  [6, 0, 0, 1, 9, 7, 2, 5, 0],
                  [9, 0, 5, 3, 0, 0, 6, 8, 7],
                  [0, 5, 0, 0, 0, 3, 8, 9, 0],
                  [3, 0, 0, 8, 0, 0, 0, 0, 0],
                  [0, 0, 0, 9, 0, 6, 3, 4, 0],
                  [0, 3, 0, 6, 8, 0, 0, 0, 9],
                  [5, 0, 0, 7, 3, 0, 1, 0, 2],
                  [7, 0, 4, 2, 1, 0, 0, 0, 0]
            ]

printsudoku(sudoku)

if sudokusolver(sudoku,0,0):
      print("solution existed")
      printsudoku(sudoku)
else:
      print("solution not existed")

'''
C:\test>py test.py
0 0 0 0 6 8 0 0 3
6 0 0 1 9 7 2 5 0
9 0 5 3 0 0 6 8 7
0 5 0 0 0 3 8 9 0
3 0 0 8 0 0 0 0 0
0 0 0 9 0 6 3 4 0
0 3 0 6 8 0 0 0 9
5 0 0 7 3 0 1 0 2
7 0 4 2 1 0 0 0 0
solution existed
4 2 7 5 6 8 9 1 3
6 8 3 1 9 7 2 5 4
9 1 5 3 4 2 6 8 7
2 5 6 4 7 3 8 9 1
3 4 9 8 5 1 7 2 6
8 7 1 9 2 6 3 4 5
1 3 2 6 8 5 4 7 9
5 9 8 7 3 4 1 6 2
7 6 4 2 1 9 5 3 8
```

```
'''

Chapter:09 --> Dynamic Programming
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
==> optimization in recursion is dynamic programming
==> overlapping calculations
==> we are storing the results in a array/list
==> reuse the already calculated solutions(sub-problems)

1) memoization or top-down approach ---> recursion
2) tabulation or bottom-up approach ---> loops

Application1 ====> Fibonacci Numbers
------------------------------------
Intially we have two numbers are 0 and 1, next number is calculated by adding
previous two numbers.

Implementation of fib seq with recursion
----------------------------------------
def fib(n):
      global c
      c=c+1
      if n==0 or n==1:
            return n
      return fib(n-1) + fib(n-2)

c=0
print(fib(6)) #8
print(f"number of calculations:{c}")
'''
C:\test>py test.py
8
number of calculations:25
'''

Implementation of fib seq with memoization
------------------------------------------
L = [None]*100

def fib(n):
      global c
      c=c+1
      if L[n]!=None:
            return L[n]
      if n==0 or n==1:
            L[n] = n
      else:
            L[n] = fib(n-1) + fib(n-2)
      return L[n]

c=0
print(fib(6)) #8
print(f"number of calculations:{c}")
'''
C:\test>py test.py
8
number of calculations:11
'''
Implementation of fib seq with tabulation
-----------------------------------------
def fib(n):
      dp = [None]*(n+1)
      dp[0] = 0
      dp[1] = 1
```

```
      for i in range(2,n+1):
            dp[i] = dp[i-2] + dp[i-1]
      return dp[n]

print(fib(6)) #8
'''
C:\test>py test.py
8
'''
```

Application2: Climbing Stairs
----
count number of ways to reach nth stair, the person can climb only either 1 or 2 steps at a time.

```
n=0 ----> 1
n=1 ----> 1
n=2 ----> 2
n=3 ----> 3
n=4 ----> 5
n=5 ----> 8
```

Ex:

```
def countways(n):
      if n==0:
            return 1
      if n<0:
            return 0
      return countways(n-1) + countways(n-2)

for i in range(10):
      print(f"num of steps:{i} and num of ways to reach top:{countways(i)}")

C:\test>py test.py
num of steps:0 and num of ways to reach top:1
num of steps:1 and num of ways to reach top:1
num of steps:2 and num of ways to reach top:2
num of steps:3 and num of ways to reach top:3
num of steps:4 and num of ways to reach top:5
num of steps:5 and num of ways to reach top:8
num of steps:6 and num of ways to reach top:13
num of steps:7 and num of ways to reach top:21
num of steps:8 and num of ways to reach top:34
num of steps:9 and num of ways to reach top:55
```

Ex:

```
def countways(n):
      dp = [0 for i in range(n+1)]
      dp[0] = 1
      for i in range(1,n+1):
            if i==1:
                  dp[i] = dp[i-1]
            else:
                  dp[i] = dp[i-1] + dp[i-2]
      return dp[n]

for i in range(10):
      print(f"{i} ====> {countways(i)}")

C:\test>py test.py
0 ====> 1
1 ====> 1
2 ====> 2
3 ====> 3
```

```
4 ====> 5
5 ====> 8
6 ====> 13
7 ====> 21
8 ====> 34
9 ====> 55
```

```
application: knapsack problem
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Example1: Gold Coins
Example2: Online Examination

items values ----> [10, 40, 30, 50]
items weights ---> [5, 4, 6, 3]
total weight ----> 10

max profit ------> 90
```

```
def knap(w,wts,vals,n):
      if n==0 or w==0:
            return 0
      if wts[n-1]>w:
            return knap(w,wts,vals,n-1)
      return max(vals[n-1]+knap(w-wts[n-1],wts,vals,n-1),knap(w,wts,vals,n-1))


w = 10
wts = [5, 4, 6, 3]
vals = [10, 40, 30, 50]
n = 4
print(knap(w,wts,vals,n))
```

```
items values ----> [10, 20, 30]
items weights ---> [60, 100, 120]
total weight ----> 50

max profit ------> 220
```

```
dynamic programming solution for knapsack problems
--------------------------------------------------
def knapsack(w,wts,vals,n):
      dp = [[0 for i in range(w+1)] for i in range(n+1)]
      for i in range(1,n+1):
            for j in range(1,w+1):
                  if wts[i-1]<=j:
                        dp[i][j] = max(dp[i-1][j],dp[i-1][j-wts[i-1]]+vals[i-1])
                  else:
                        dp[i][j] = dp[i-1][j]
      return dp[n][w]

w = 10
wts = [5, 4, 6, 3]
vals = [10, 40, 30, 50]
n = 4
print(knapsack(w,wts,vals,n)) #90
```

```
C:\test>py test.py
90
```

PYTHON @7AM -------->
CORE JAVA @9PM ----->

Chapter:10 --> String

Ex1) read and write a string

```
s = input("enter string value: ")
print(s)

C:\test>py test.py
enter string value: welcome
welcome
```

Ex2) read string and print characters present at +ve and -ve indexes

```
s = input("enter string value: ")
print(s)

index1=0
index2=-1
while index1<len(s) and index2>-(len(s)+1):
      print(f"index={index1} value:{s[index1]} index={index2} value:{s[index2]}")
      index1+=1
      index2-=1

C:\test>py test.py
enter string value: welcome
welcome
index=0 value:w index=-1 value:e
index=1 value:e index=-2 value:m
index=2 value:l index=-3 value:o
index=3 value:c index=-4 value:c
index=4 value:o index=-5 value:l
index=5 value:m index=-6 value:e
index=6 value:e index=-7 value:w
```

Ex3) print characters present at even index values

```
s = input("enter string value: ")
print(s)

index=0
while index<len(s):
      if index%2==0:
            print(f"index={index} and char={s[index]}")
      index+=1

C:\test>py test.py
enter string value: prakash
prakash
index=0 and char=p
index=2 and char=a
index=4 and char=a
index=6 and char=h
```

Ex4) print characters present at odd index

```
s = input("enter string value: ")
print(s)

index=0
while index<len(s):
      if index%2!=0:
            print(f"index={index} and char={s[index]}")
      index+=1
```

```
C:\test>py test.py
enter string value: prakash
prakash
index=1 and char=r
index=3 and char=k
index=5 and char=s

Ex5) count number of vowels present in a string
-----------------------------------------------
import re

def fun(s):
      return len(re.findall("[aeiou]",s))

s = input("enter string value: ")
print(fun(s))

C:\test>py test.py
enter string value: python programming
4

Ex6) count number of consonants present in a string
---------------------------------------------------
import re

def fun(s):
      if s.isalpha():
            return len(re.findall("[^aeiou]",s))

s = input("enter string value: ")
print(fun(s))

C:\test>py test.py
enter string value: python
5

C:\test>py test.py
enter string value: python123
None

Ex7) count number of alphabets
------------------------------
import re

def fun(s):
      return len(re.findall("[a-zA-Z]",s))

s = input("enter string value: ")
print(fun(s))

C:\test>py test.py
enter string value: admin@Python1234
11

Ex8) count number of digits
---------------------------
import re

def fun(s):
      return len(re.findall("[0-9]",s))

s = input("enter string value: ")
print(fun(s))
```

```
C:\test>py test.py
enter string value: admin@Python1234
4
```

```
Ex9) count spaces
-----------------
import re

def fun(s):
      return len(re.findall("[ ]",s))

s = input("enter string value: ")
print(fun(s))

C:\test>py test.py
enter string value: admin@Python1234
0

C:\test>py test.py
enter string value: admin Python1234
1

C:\test>py test.py
enter string value: admin Python1234 Java
2
```

```
Ex10) count special characters
------------------------------
import re

def fun(s):
      return len(re.findall("[^a-zA-Z0-9 ]",s))

s = input("enter string value: ")
print(fun(s))

C:\test>py test.py
enter string value: python#java$c*c++
5
```

```
Ex11) reverse the given string
------------------------------
def fun(s):
      return s[::-1]

def fun1(s):
      res = ""
      for i in s:
            res = i + res
      return res

s = input("enter string value: ")
print(fun(s))
print(fun1(s))

C:\test>py test.py
enter string value: prakash
hsakarp
hsakarp
```

```
Ex12) convert given string into upper case
------------------------------------------
def fun1(s):
      return s.upper()

def fun2(s):
      res = ""
      for i in s:
            res = res + chr(ord(i)-32)
      return res

s=input("Enter any string: ")
print(fun1(s))
print(fun2(s))
```

```
C:\test>py test.py
Enter any string: prakash
PRAKASH
PRAKASH
```

Ex13) convert given string into lower case

```
def fun1(s):
      return s.lower()

def fun2(s):
      res = ""
      for i in s:
            res = res + chr(ord(i)+32)
      return res

s=input("Enter any string: ")
print(fun1(s))
print(fun2(s))
```

```
C:\test>py test.py
Enter any string: PYTHON
python
python
```

Ex14) toggle case or swap case

```
def fun1(s):
      return s.swapcase()

def fun2(s):
      res = ""
      for i in s:
            if i.isupper():
                  res = res + chr(ord(i)+32)
            if i.islower():
                  res = res + chr(ord(i)-32)
      return res

s=input("Enter any string: ")
print(fun1(s))
print(fun2(s))
```

```
C:\test>py test.py
Enter any string: WeLcOmE
wElCoMe
wElCoMe
```

Ex15) convert even index characters into upper case odd indexed char into lower case
----

Ex:

```
welcome ---> WeLcOmE
python java ---> PyThOn jAvA
```

```
def fun(s):
      res=""
      for i in range(len(s)):
            if i%2==0:
                  res=res+s[i].upper()
            else:
                  res=res+s[i].lower()
      return res

s=input("Enter any string: ")
print(fun(s))
```

```
C:\test>py test.py
Enter any string: prakash
PrAkAsH
```

Ex16) reverse the sentence

```
def fun(s):
      return s[::-1]

s=input()
print(fun(s))
```

```
C:\test>py test.py
a ab abc abcd abcde abcdef abcdefg x xy xyz wxyz
zyxw zyx yx x gfedcba fedcba edcba dcba cba ba a
```

Ex17) reverse the even indexed words in sentence

```
def fun(s):
      L = []
      index=0
      for i in s.split():
            if index%2==0:
                  L.append(i[::-1])
            else:
                  L.append(i)
            index+=1
      return ' '.join(L)

s=input()
print(fun(s))
```

```
C:\test>py test.py
a ab abc abcd abcde abcdef abcdefg x xy xyz wxyz
a ab cba abcd edcba abcdef gfedcba x yx xyz zyxw
```

Ex18) reverse the odd indexed words in sentence

```
def fun(s):
      L = []
      index=0
      for i in s.split():
            if index%2!=0:
                  L.append(i[::-1])
            else:
                  L.append(i)
            index+=1
      return ' '.join(L)
```

```
s=input()
print(fun(s))

C:\test>py test.py
a ab abc abcd abcde abcdef abcdefg x xy xyz wxyz
a ba abc dcba abcde fedcba abcdefg x xy zyx wxyz
```

Ex19) reverse individual words in sentence

```
def fun(s):
      L = []
      for i in s.split():
            L.append(i[::-1])
      return ' '.join(L)

s=input()
print(fun(s))

C:\test>py test.py
a ab abc abcd abcde abcdef abcdefg x xy xyz wxyz
a ba cba dcba edcba fedcba gfedcba x yx zyx zyxw
```

Ex20) convert each word first char into upper case

```
def fun(s):
      L = []
      for i in s.split():
            L.append(i[0].upper()+i[1:].lower())
      return ' '.join(L)

s=input()
print(fun(s))

C:\test>py test.py
a ab abc abcd abcde abcdef abcdefg x xy xyz wxyz
A Ab Abc Abcd Abcde Abcdef Abcdefg X Xy Xyz Wxyz
```

Ex21) convert each word first and last char into upper case

```
def fun(s):
      L = []
      for i in s.split():
            if len(i)!=1:
                  L.append(i[0].upper()+i[1:len(i)-1].lower()+i[-1].upper())
            else:
                  L.append(i[0].upper())
      return ' '.join(L)

s=input()
print(fun(s))
```

Ex22) convert every word into upper case except first and last

```
def fun(s):
      L = []
      for i in s.split():
            if len(i)!=1:
                  L.append(i[0].lower()+i[1:len(i)-1].upper()+i[-1].lower())
            else:
                  L.append(i[0].lower())
      return ' '.join(L)

s=input()
```

```
print(fun(s))
```

```
C:\test>py test.py
a ab abc abcd abcde abcdef abcdefg x xy xyz wxyz
a ab aBc aBCd aBCDe aBCDEf aBCDEFg x xy xYz wXYz
```

Ex23) convert every even length words into upper case remaining lower case

```
def fun(s):
      L = []
      for i in s.split():
            if len(i)%2==0:
                  L.append(i.upper())
            else:
                  L.append(i.lower())
      return ' '.join(L)

s=input()
print(fun(s))
```

```
C:\test>py test.py
a ab abc abcd abcde abcdef abcdefg x xy xyz wxyz
a AB abc ABCD abcde ABCDEF abcdefg x XY xyz WXYZ
```

Ex24) return middle char(s) from a string

```
abc ----> b
abcd ---> bc
```

```
def fun(s):
      if len(s)%2!=0:
            return s[len(s)//2]
      else:
            return s[len(s)//2-1:len(s)//2+1]

print(fun("abc")) #b
print(fun("abcd")) #bc
print(fun("prakash")) #k
```

Ex25) sort the string i.e. characters present in the string

```
badc ---> abcd
```

```
def fun(s):
      L = list(s)
      L.sort()
      return ''.join(L)

print(fun("bacd")) #abcd
print(fun("yzxw")) #wxyz
print(fun("prakash")) #aahkprs
```

```
C:\test>py test.py
abcd
wxyz
aahkprs
```

Ex26: sort the names

```
def fun(L):
      L.sort()

L = ["ijk","xyz","pqrs","abc","mno"]
print(L)
```

```
fun(L)
print(L)

C:\test>py test.py
['ijk', 'xyz', 'pqrs', 'abc', 'mno']
['abc', 'ijk', 'mno', 'pqrs', 'xyz']
```

Ex27: remove duplicate characters

```
def fun(s):
      L = []
      for i in s:
            if i not in L:
                  L.append(i)
      return ''.join(L)

s = input("Enter any string: ")
print(fun(s))

C:\test>py test.py
Enter any string: prakash
praksh

C:\test>py test.py
Enter any string: welcome
welcom

C:\test>py test.py
Enter any string: programmer
progame
```

Ex28: remove special characters

```
import re

def fun(s):
      return re.sub("[^a-zA-Z0-9 ]","",s)

s = input("Enter any string: ")
print(fun(s))

C:\test>py test.py
Enter any string: admin#abc*456^p
adminabc456p
```

Ex29: remove vowels from a string

```
import re

def fun(s):
      return re.sub("[aeiou]","",s)

s = input("Enter any string: ")
print(fun(s))

C:\test>py test.py
Enter any string: welcome
wlcm
```

Ex30: check a string is paliandrome or not

```
def fun(s):
      low = 0
      high = len(s)-1
      while low<high:
            if s[low]!=s[high]:
                  return False
            low = low + 1
            high = high - 1
      return True

print(fun("abba")) #True
print(fun("abc")) #False
```

Ex31: return longest (w.r.t len) pali in a string

```
def fun1(s):
      low = 0
      high = len(s)-1
      while low<high:
            if s[low]!=s[high]:
                  return False
            low = low + 1
            high = high - 1
      return True

def fun2(s):
      return len(s)

def fun(s):
      L = []
      for i in s.split():
            if fun1(i):
                  L.append(i)
      L.sort(key=fun2,reverse=True)
      #print(L)
      return L[0]

print(fun("a ab abc aba madam sir malayalam liril prakash")) #malayalam
print(fun("a ab abc aba madam sir aaaaa liril prakash")) #madam/liril
```

```
C:\test>py test.py
malayalam
madam
```

Ex32: check if a string is rotated

```
def fun(s1,s2):
      if len(s1)!=len(s2):
            return False
      temp=s1+s1
      return temp.find(s2)!=-1


print(fun("ABCD","CDAB")) #True
print(fun("ABCD","CDBA")) #False
print(fun("ABAAA","BAAAA")) #True
print(fun("ABAB","ABBA")) #False
```

Ex33: Check two strings are anagrams or not

```
def fun(s1,s2):
      if len(s1)!=len(s2):
            return False
      s1=sorted(s1)
      s2=sorted(s2)
      return s1==s2
```

```
print(fun("listen","silent")) #True
print(fun("abcd","cadb")) #True
print(fun("abc","abd")) #False
```

Ex34: pangram

```
the quick brown fox jumps over the lazy dog ---> all alphabets --> True
the quick brown fox jumps over the lay dog  ---> all alphabets --> False
```

```
def fun(s):
    for i in range(97,123):
        if chr(i) not in s:
            return False
    return True

print(fun("the quick brown fox jumps over the lazy dog")) #True
print(fun("the quick brown fox jumps over the lay dog")) #Flase
```

Ex35: string validation

Given a string representing a password. you need to check if the string is valid or not. a valid string has the following properties.

1) length greater than or equal to 10
2) contain at least 1 numeric character
3) contain at least 1 upper case character
4) contain at least 1 lower case character
5) contain at least 1 special character @#$-*

```
#a = "abcdcda"
#b = "cd"

a = "aaaaa"
b = "a"

i = 0
j = 0
c = 0
while i<len(a):
    if a[i]==b[j]:
        j=j+1
        if j==len(b):
            c=c+1
            j=0
    i=i+1

print(c)
```

Chapter:11 --> Sorting

arranging elements/objects in ascending or descending order. we can do this by using predefined methods and we can implement our own methods. we have some predefined functions are existed in python library and we can implement some data structure sorting algorithms (bubble, selection, insertion etc)

1) list.sort()
2) sorted(sequence)

sort() method:

1) it is applicable only for list objects.
2) if we want perform in asc order ---> list.sort()
3) if we want perform in desc order --> list.sort(reverse=True)
4) if we want to sort based our req --> list.sort(key=fun)

5) predefined sort method is following stable sort.

```
Example1: sort the elements in a list in ascending order
--------------------------------------------------------
L = [10, 14, 12, 15, 13, 11]
print(L)
L.sort()
print(L)

C:\test>py test.py
[10, 14, 12, 15, 13, 11]
[10, 11, 12, 13, 14, 15]
```

```
Example2: sort the elements in a list in descending order
---------------------------------------------------------
L = [10, 14, 12, 15, 13, 11]
print(L)
L.sort(reverse=True)
print(L)

C:\test>py test.py
[10, 14, 12, 15, 13, 11]
[15, 14, 13, 12, 11, 10]
```

```
Example3: sort the list of names in alphabetical order (asc)
------------------------------------------------------------
L = ["HHH","BBB","PPP","CCC","KKK","AAA"]
print(L)
L.sort()
print(L)

C:\test>py test.py
['HHH', 'BBB', 'PPP', 'CCC', 'KKK', 'AAA']
['AAA', 'BBB', 'CCC', 'HHH', 'KKK', 'PPP']
```

```
Example4: sort the names in ascending order based on length of name
-------------------------------------------------------------------
def fun(s):
      return len(s)

L = ["Ram", "Prabas", "Charan", "Tarak", "Arjun"]
print(L)
L.sort(key=fun)
print(L)

C:\test>py test.py
['Ram', 'Prabas', 'Charan', 'Tarak', 'Arjun']
['Ram', 'Tarak', 'Arjun', 'Prabas', 'Charan']
```

```
Example5: sort the student objects based on name
------------------------------------------------
class student:
      def __init__(self,name,htno,marks):
            self.name = name
            self.htno = htno
            self.marks = marks
      def __str__(self):
            return f"({self.name},{self.htno},{self.marks})"

def fun(s):
      return s.name

s1 = student("Prakash",333,90)
s2 = student("Prasanth",666,50)
```

```
s3 = student("Prabas",222,80)
s4 = student("Prakesh",444,70)
s5 = student("Prabhu",111,60)
L = [s1, s2, s3, s4]
print("before sorting")
for i in L:
      print(i)

L.sort(key=fun)

print("after sorting")
for i in L:
      print(i)

'''
C:\test>py test.py
before sorting
(Prakash,333,90)
(Prasanth,666,50)
(Prabas,222,80)
(Prakesh,444,70)
after sorting
(Prabas,222,80)
(Prakash,333,90)
(Prakesh,444,70)
(Prasanth,666,50)
'''
```

Example6: sort the student objects based on htno

```
------------------------------------------------
class student:
      def __init__(self,name,htno,marks):
            self.name = name
            self.htno = htno
            self.marks = marks
      def __str__(self):
            return f"({self.name},{self.htno},{self.marks})"

def fun(s):
      return s.htno

s1 = student("Prakash",333,90)
s2 = student("Prasanth",666,50)
s3 = student("Prabas",222,80)
s4 = student("Prakesh",444,70)
s5 = student("Prabhu",111,60)
L = [s1, s2, s3, s4]
print("before sorting")
for i in L:
      print(i)

L.sort(key=fun)

print("after sorting")
for i in L:
      print(i)

'''
C:\test>py test.py
before sorting
(Prakash,333,90)
(Prasanth,666,50)
(Prabas,222,80)
(Prakesh,444,70)
```

```
after sorting
(Prabas,222,80)
(Prakash,333,90)
(Prakesh,444,70)
(Prasanth,666,50)
'''

Example7: sort the student objects based on marks
------------------------------------------------
class student:
      def __init__(self,name,htno,marks):
            self.name = name
            self.htno = htno
            self.marks = marks
      def __str__(self):
            return f"({self.name},{self.htno},{self.marks})"

def fun(s):
      return s.marks

s1 = student("Prakash",333,90)
s2 = student("Prasanth",666,50)
s3 = student("Prabas",222,80)
s4 = student("Prakesh",444,70)
s5 = student("Prabhu",111,60)
L = [s1, s2, s3, s4]
print("before sorting")
for i in L:
      print(i)

L.sort(key=fun)

print("after sorting")
for i in L:
      print(i)

'''
C:\test>py test.py
before sorting
(Prakash,333,90)
(Prasanth,666,50)
(Prabas,222,80)
(Prakesh,444,70)
after sorting
(Prasanth,666,50)
(Prakesh,444,70)
(Prabas,222,80)
(Prakash,333,90)
'''

Example8: sort the elements in a list in asc order using sorted method
----------------------------------------------------------------------
L = [10, 16, 13, 11, 15]
NL = sorted(L)
print(L)
print(NL)

C:\test>py test.py
[10, 16, 13, 11, 15]
[10, 11, 13, 15, 16]

Example9: sort the elements in a list in asc order using abs method
-------------------------------------------------------------------
L = [10, -16, 13, -11, 15]
```

```
NL = sorted(L,key=abs)
print(L)
print(NL)

C:\test>py test.py
[10, -16, 13, -11, 15]
[10, -11, 13, 15, -16]
```

bubble sort algorithm:

==> It is used to sort the data in asc/desc.

==> It is a stable sort

==> First element and compare with second element, if first element is greater than second, then swap is required else no swap is required.

==> second element and compare with third element this process will continue

Example10: implement bubble sort algorithm in asc order

```
import random

def bubblesort(l):
    for i in range(len(l)-1):
        for j in range(len(l)-i-1):
            if l[j] > l[j+1]:
                l[j],l[j+1] = l[j+1],l[j]

l = []
for i in range(10):
    l.append(random.randint(10,99))
print(l)
bubblesort(l)
print(l)

'''
C:\test>py test.py
[73, 88, 67, 22, 58, 40, 13, 37, 10, 26]
[10, 13, 22, 26, 37, 40, 58, 67, 73, 88]
'''
```

Example11: implement bubble sort algorithm in desc order

```
import random

def bubblesort(l):
    for i in range(len(l)-1):
        for j in range(len(l)-i-1):
            if l[j] < l[j+1]:
                l[j],l[j+1] = l[j+1],l[j]

l = []
for i in range(10):
    l.append(random.randint(10,99))
print(l)
bubblesort(l)
print(l)

'''
C:\test>py test.py
[10, 57, 80, 28, 29, 64, 25, 80, 81, 45]
[81, 80, 80, 64, 57, 45, 29, 28, 25, 10]
'''
```

Example12: modified bubble sort if the data is already in sorted order

```
----------------------------------------------------------------------
import random

def bubblesort(l):
      for i in range(len(l)-1):
            swap=False
            for j in range(len(l)-i-1):
                  if l[j] > l[j+1]:
                        l[j],l[j+1] = l[j+1],l[j]
                        swap = True
            if swap==False:
                  return

l = [10, 20, 30, 40, 50]
print(l)
bubblesort(l)
print(l)

'''
C:\test>py test.py
[10, 20, 30, 40, 50]
[10, 20, 30, 40, 50]
'''
```

```
selection sort:
---------------
==> select an element and fix that element in its position.
==> first min element into first location
==> second min element into second location and so on
```

```
Example13: implementation of selection sort in asc order
--------------------------------------------------------
import random

def selectionsort(l):
      n = len(l)
      for i in range(n-1):
            min_i = i
            for j in range(i+1,n):
                  if l[j] < l[min_i]:
                        min_i = j
            l[i],l[min_i] = l[min_i],l[i]


l = []
for i in range(10):
      l.append(random.randint(10,99))
print(l)
selectionsort(l)
print(l)

'''
C:\test>py test.py
[94, 78, 11, 51, 15, 98, 63, 86, 39, 96]
[11, 15, 39, 51, 63, 78, 86, 94, 96, 98]
'''
```

```
Example14: implementation of selection sort in desc order
--------------------------------------------------------
import random

def selectionsort(l):
      n = len(l)
      for i in range(n-1):
        max_i = i
        for j in range(i+1,n):
            if l[j] > l[max_i]:
                max_i = j
        l[i],l[max_i] = l[max_i],l[i]


l = []
for i in range(10):
    l.append(random.randint(10,99))
print(l)
selectionsort(l)
print(l)

'''
C:\test>py test.py
[50, 76, 24, 99, 68, 59, 21, 42, 42, 91]
[99, 91, 76, 68, 59, 50, 42, 42, 24, 21]
'''
```

insertion sort

Example15: implementation of insertion sort in asc order

```
import random

def insertionsort(l):
    n = len(l)
    for i in range(1,n):
        x = l[i]
        j = i-1
        while j>=0 and x < l[j]:
            l[j+1] = l[j]
            j=j-1
        l[j+1] = x


l = []
for i in range(10):
    l.append(random.randint(10,99))
print(l)
insertionsort(l)
print(l)

'''
C:\test>py test.py
[99, 62, 89, 77, 34, 75, 59, 20, 72, 26]
[20, 26, 34, 59, 62, 72, 75, 77, 89, 99]
'''
```

Example16: implementation of insertion sort in desc order

```
import random

def insertionsort(l):
    n = len(l)
    for i in range(1,n):
        x = l[i]
        j = i-1
        while j>=0 and x > l[j]:
            l[j+1] = l[j]
            j=j-1
        l[j+1] = x
```

```
l = []
for i in range(10):
      l.append(random.randint(10,99))
print(l)
insertionsort(l)
print(l)

'''
C:\test>py test.py
[46, 14, 19, 86, 69, 13, 83, 39, 57, 10]
[86, 83, 69, 57, 46, 39, 19, 14, 13, 10]
'''
```

```
counting sort:
--------------
Ex1:
----
L = [1, 4, 4, 1, 0, 1]
K = 5

output: [0, 1, 1, 4, 4]

Ex2:
----
L = [2, 1, 8, 9, 4]
K = 10

output: [1, 2, 4, 8, 9]
```

```
def countingsort(L,K):
      output = [0] * len(L)
      count = [0] * K
      for i in L:
            count[i] = count[i]+1
      for i in range(1,K):
            count[i] = count[i] + count[i-1]
      for i in reversed(L):
            output[count[i]-1] = i
            count[i] = count[i] - 1
      for i in range(len(L)):
            L[i] = output[i]

L = [1, 4, 4, 1, 0, 1]
K = 5
print(L)
countingsort(L,K)
print(L)

C:\test>py test.py
[1, 4, 4, 1, 0, 1]
[0, 1, 1, 1, 4, 4]
```

```
radix sort
----------
def countingsort(l,pos):
      output = [0] * len(l)
      count = [0] * 10
      for i in range(0,len(l)):
            index = (l[i]//pos)%10
            count[index] = count[index]+1
      for i in range(1,10):
            count[i] = count[i] + count[i-1]
      i = len(l)-1
      while i>=0:
            index = (l[i]//pos)%10
            output[count[index]-1] = l[i]
            count[index] = count[index]-1
            i=i-1
      for i in range(0,len(l)):
            l[i] = output[i]

def radixsort(l):
      mx = max(l)
      pos = 1
      while mx//pos>0:
            countingsort(l,pos)
            pos = pos * 10
l = [319, 212, 6, 8, 100, 50]
print(l)
radixsort(l)
print(l)
```

```
C:\test>py test.py
[319, 212, 6, 8, 100, 50]
[6, 8, 50, 100, 212, 319]
```

bubble sort
selection sort
insertion sort
counting sort
radix sort

Chapter:13 --> Divide and Conquer Algorithms

divide and conquer is a method, which divides a big problem into individual small sub-problems, later we can solve these sub-problems to get solution for the main problem.

Ex:
- merge sort
- quick sort
- binary search algorithm
- etc

merge sort

Ex1: Given two sorted list objects, merge two list objects and sort

```
a = [10, 15, 20]
b = [5, 6, 6, 30]

output ---> [5, 6, 6, 10, 15, 20, 30]
```

```
def merge(a,b):
      c = a + b
      c.sort()
      return c

a = [10, 15, 20]
b = [5, 6, 6, 30]
print(a)
print(b)
print(merge(a,b))
```

```
C:\test>py test.py
[10, 15, 20]
[5, 6, 6, 30]
[5, 6, 6, 10, 15, 20, 30]
```

Ex2: Given two sorted list objects, merge two list objects and sort
----

```
def merge(a,b):
      c = []
      m,n = len(a),len(b)
      i,j=0,0
      while i<m and j<n:
            if a[i] < b[j]:
                  c.append(a[i])
                  i=i+1
            else:
                  c.append(b[j])
                  j=j+1
      while i<m:
            c.append(a[i])
            i=i+1
      while j<n:
            c.append(b[j])
            j=j+1
      return c

a = [1, 2, 8, 9]
b = [3, 5, 6, 7]
print(a)
print(b)
print(merge(a,b))
```

```
C:\test>py test.py
[1, 2, 8, 9]
[3, 5, 6, 7]
[1, 2, 3, 5, 6, 7, 8, 9]
```

Ex3: merge sub arrays

```
L = [10, 15, 20, 11, 13]

low = 0
high = 4
mid = 2

output: [10, 11, 13, 15, 20]


def merge(a,low,mid,high):
      l = a[low:mid+1]
      r = a[mid+1:high+1]
      i=j=0
      k=low
      while i<len(l) and j<len(r):
            if l[i] < r[j]:
                  a[k] = l[i]
                  k=k+1
                  i=i+1
            else:
                  a[k] = r[j]
                  k=k+1
                  j=j+1
      while i<len(l):
            a[k] = l[i]
            k=k+1
            i=i+1
      while j<len(r):
            a[k] = r[i]
            k=k+1
            j=j+1
L = [10, 15, 20, 11, 13]
print(L)
merge(L,0,len(L)//2,len(L)-1)
print(L)
```

Ex4: merge sort application

```
def mergesort(l,lindex,rindex):
      if rindex > lindex:
            mid = (lindex+rindex)//2
            mergesort(l,lindex,mid)
            mergesort(l,mid+1,rindex)
            merge(l,lindex,mid,rindex)

def merge(a,low,mid,high):
      l = a[low:mid+1]
      r = a[mid+1:high+1]
      i=j=0
      k=low
      while i<len(l) and j<len(r):
            if l[i] < r[j]:
                  a[k] = l[i]
                  k=k+1
                  i=i+1
            else:
                  a[k] = r[j]
                  k=k+1
                  j=j+1
      while i<len(l):
            a[k] = l[i]
            k=k+1
            i=i+1
      while j<len(r):
            a[k] = r[j]
            k=k+1
            j=j+1
L = [4, 6, 1, 9, 2, 7, 3, 8, 5]
print(L)
mergesort(L,0,len(L)-1)
print(L)
```

```
C:\test>py test.py
[4, 6, 1, 9, 2, 7, 3, 8, 5]

[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

quick sort:

```
def quicksort(l,low,high):
      if high<=low:
            return
      pivot = l[low]
      start = low
      end = high
      while low < high:
            while l[low] <= pivot and low < high:
                  low = low + 1
            while l[high] > pivot and low <= high:
                  high = high - 1
            if low < high:
                  l[high],l[low] = l[low],l[high]
      l[high],l[start] = l[start],l[high]
      quicksort(l,start,high-1)
      quicksort(l,high+1,end)


L = [3, 5, 4, 2, 1, 6]
print(L)
quicksort(L,0,len(L)-1)
print(L)
```

```
C:\test>py test.py
[3, 5, 4, 2, 1, 6]
[1, 2, 3, 4, 5, 6]
```

Chapter:12 --> Searching

searching is the process of finding an item / object / element in a collection of items The item may be keyword in file, book in library, student record in db, an obj in list etc.

The following are the two methods existed to perform search operation.

1) linear search
2) binary search

Ex1: Implement linear search algorithm

```
def linearsearch(L,key):
      for i in range(len(L)):
            if key == L[i]:
                  return i
      return -1

L = [10, 30, 90, 20, 50, 80, 70, 60, 40, 100]
print(L)
print(f"10  =>     {linearsearch(L,40)}") #8
print(f"120 =>   {linearsearch(L,120)}") #-1
print(f"50  =>   {linearsearch(L,50)}") #4
print(f"55  =>   {linearsearch(L,55)}") #-1
```

```
C:\test>py test.py
[10, 30, 90, 20, 50, 80, 70, 60, 40, 100]
10  =>       8
120 =>   -1
50  =>   4
55  =>   -1
```

Note: If we want to apply linear search algorithm, list can be in any order

Ex2: Implement linear search algorithm to return first and second occurrence

```
def lsearchfirstandsecoccurrece(L,key):
      TL = []
      for i in range(len(L)):
            if key == L[i]:
                  TL.append(i)
      return TL[:2]

L = [10, 30, 10, 20, 10, 80, 10, 60, 10, 90]
```

```
print(L)
print(f"10: {lsearchfirstandsecoccurrece(L,10)}")
print(f"80: {lsearchfirstandsecoccurrece(L,80)}")
print(f"50: {lsearchfirstandsecoccurrece(L,50)}")
```

```
C:\test>py test.py
[10, 30, 10, 20, 10, 80, 10, 60, 10, 90]
10: [0, 2]
80: [5]
50: []
```

Ex3: Implement linear search algorithm to return all occurrences
----

```
def lsearchalloccurreces(L,key):
      TL = []
      for i in range(len(L)):
            if key == L[i]:
                  TL.append(i)
      return TL

L = [10, 30, 10, 20, 10, 80, 10, 60, 10, 80, 90]
print(L)
print(f"10: {lsearchalloccurreces(L,10)}")
print(f"80: {lsearchalloccurreces(L,80)}")
print(f"50: {lsearchalloccurreces(L,50)}")
```

```
C:\test>py test.py
[10, 30, 10, 20, 10, 80, 10, 60, 10, 80, 90]
10: [0, 2, 4, 6, 8]
80: [5, 9]
50: []
```

Ex4: Implement linear search algorithm to return last occurrence
----

```
def lsearchlastcoccurrece(L,key):
      TL = []
      for i in range(len(L)):
            if key == L[i]:
                  TL.append(i)
      return TL[-1:-2:-1]

L = [10, 30, 10, 20, 10, 80, 10, 60, 10, 80, 90]
print(L)
print(f"10: {lsearchlastcoccurrece(L,10)}")
print(f"80: {lsearchlastcoccurrece(L,80)}")
print(f"50: {lsearchlastcoccurrece(L,50)}")
```

```
C:\test>py test.py
[10, 30, 10, 20, 10, 80, 10, 60, 10, 80, 90]
10: [8]
80: [9]
50: []
```

```
binary search algorithm:
~~~~~~~~~~~~~~~~~~~~~~~~
```

Note: If we want to apply binary search algorithm compulsory list must be in sorted order.

Ex5: Implement binary search algorithm to search for an element using iteration (loops)

```
def binarysearch1(L,key):
      low = 0
      high = len(L)-1
      while low <= high:
            mid = (low+high)//2
            if key == L[mid]:
                  return mid
            elif key < L[mid]:
                  high = mid-1
            else:
                  low = mid+1
      return -1

L = [10, 30, 60, 20, 50, 80, 70, 90, 40]
print(L)
L.sort()
print(L)
key=int(input("Enter key value: "))
print(f"{key} ==> {binarysearch1(L,key)}")
#[10, 20, 30, 40, 50, 60, 70, 80, 90]
# 0   1   2   3   4   5   6   7   8
```

```
C:\test>py test.py
[10, 30, 60, 20, 50, 80, 70, 90, 40]
[10, 20, 30, 40, 50, 60, 70, 80, 90]
Enter key value: 50
50 ==> 4

C:\test>py test.py
[10, 30, 60, 20, 50, 80, 70, 90, 40]
[10, 20, 30, 40, 50, 60, 70, 80, 90]
Enter key value: 80
80 ==> 7

C:\test>py test.py
[10, 30, 60, 20, 50, 80, 70, 90, 40]
[10, 20, 30, 40, 50, 60, 70, 80, 90]
Enter key value: 100
100 ==> -1
```

Ex6: Implement binary search algorithm to search for an element using recursion

```
def binarysearch2(L,key,low,high):
      if low>high:
            return -1
      mid = (low+high)//2
      if key==L[mid]:
            return mid
      elif key<L[mid]:
            return binarysearch2(L,key,low,mid-1)
      else:
            return binarysearch2(L,key,mid+1,high)

L = [10, 30, 60, 20, 50, 80, 70, 90, 40]
print(L)
L.sort()
print(L)
key=int(input("Enter key value: "))
print(f"{key} ==> {binarysearch2(L,key,0,len(L)-1)}")
#[10, 20, 30, 40, 50, 60, 70, 80, 90]
# 0   1   2   3   4   5   6   7   8
```

```
C:\test>py test.py
```

```
[10, 30, 60, 20, 50, 80, 70, 90, 40]
[10, 20, 30, 40, 50, 60, 70, 80, 90]
Enter key value: 30
30 ==> 2

C:\test>py test.py
[10, 30, 60, 20, 50, 80, 70, 90, 40]
[10, 20, 30, 40, 50, 60, 70, 80, 90]
Enter key value: 90
90 ==> 8

C:\test>py test.py
[10, 30, 60, 20, 50, 80, 70, 90, 40]
[10, 20, 30, 40, 50, 60, 70, 80, 90]
Enter key value: 88
88 ==> -1
```

Ex7: Implement binary search algorithm to search for an element in the first half

```
---------------------------------------------------------------------------------
def binarysearch2(L,key,low,high):
      if low>high:
            return -1
      mid = (low+high)//2
      if key==L[mid]:
            return mid
      elif key<L[mid]:
            return binarysearch2(L,key,low,mid-1)
      else:
            return binarysearch2(L,key,mid+1,high)

L = [10, 30, 60, 20, 50, 80, 70, 90, 40]
print(L)
L.sort()
print(L)
key=int(input("Enter key value: "))
print(f"{key} ==> {binarysearch2(L,key,0,len(L)//2)}")
#[10, 20, 30, 40, 50, 60, 70, 80, 90]
# 0   1   2   3   4   5   6   7   8

C:\test>py test.py
[10, 30, 60, 20, 50, 80, 70, 90, 40]
[10, 20, 30, 40, 50, 60, 70, 80, 90]
Enter key value: 10
10 ==> 0

C:\test>py test.py
[10, 30, 60, 20, 50, 80, 70, 90, 40]
[10, 20, 30, 40, 50, 60, 70, 80, 90]
Enter key value: 20
20 ==> 1

C:\test>py test.py
[10, 30, 60, 20, 50, 80, 70, 90, 40]
[10, 20, 30, 40, 50, 60, 70, 80, 90]
Enter key value: 30
30 ==> 2

C:\test>py test.py
[10, 30, 60, 20, 50, 80, 70, 90, 40]
[10, 20, 30, 40, 50, 60, 70, 80, 90]
Enter key value: 40
40 ==> 3
```

```
C:\test>py test.py
[10, 30, 60, 20, 50, 80, 70, 90, 40]
[10, 20, 30, 40, 50, 60, 70, 80, 90]
Enter key value: 50
50 ==> 4

C:\test>py test.py
[10, 30, 60, 20, 50, 80, 70, 90, 40]
[10, 20, 30, 40, 50, 60, 70, 80, 90]
Enter key value: 60
60 ==> -1
```

Ex8: Implement binary search algorithm to search for an element in the second half
----

```
def binarysearch2(L,key,low,high):
      if low>high:
            return -1
      mid = (low+high)//2
      if key==L[mid]:
            return mid
      elif key<L[mid]:
            return binarysearch2(L,key,low,mid-1)
      else:
            return binarysearch2(L,key,mid+1,high)

L = [10, 30, 60, 20, 50, 80, 70, 90, 40]
print(L)
L.sort()
print(L)
key=int(input("Enter key value: "))
print(f"{key} ==> {binarysearch2(L,key,len(L)//2+1,len(L)-1)}")
#[10, 20, 30, 40, 50, 60, 70, 80, 90]
# 0   1   2   3   4   5   6   7   8

C:\test>py test.py
[10, 30, 60, 20, 50, 80, 70, 90, 40]
[10, 20, 30, 40, 50, 60, 70, 80, 90]
Enter key value: 90
90 ==> 8

C:\test>py test.py
[10, 30, 60, 20, 50, 80, 70, 90, 40]
[10, 20, 30, 40, 50, 60, 70, 80, 90]
Enter key value: 80
80 ==> 7

C:\test>py test.py
[10, 30, 60, 20, 50, 80, 70, 90, 40]
[10, 20, 30, 40, 50, 60, 70, 80, 90]
Enter key value: 70
70 ==> 6

C:\test>py test.py
[10, 30, 60, 20, 50, 80, 70, 90, 40]
[10, 20, 30, 40, 50, 60, 70, 80, 90]
Enter key value: 60
60 ==> 5

C:\test>py test.py
[10, 30, 60, 20, 50, 80, 70, 90, 40]
[10, 20, 30, 40, 50, 60, 70, 80, 90]
```

```
Enter key value: 50
50 ==> -1
```

Ex9: Implement binary search algorithm to search for an element in the given range

```
--------------------------------------------------------------------------------
--
def binarysearch(L,key,low,high):
      if low>high:
            return -1
      mid = (low+high)//2
      if key==L[mid]:
            return mid
      elif key<L[mid]:
            return binarysearch(L,key,low,mid-1)
      else:
            return binarysearch(L,key,mid+1,high)

L = [10, 30, 60, 20, 50, 80, 70, 90, 40]
print(L)
L.sort()
print(L)
key=int(input("Enter key value: "))
start = int(input("Enter start value of search: "))
end = int(input("Enter end value of search: "))
print(f"{key} ==> {binarysearch(L,key,start,end)}")
#[10, 20, 30, 40, 50, 60, 70, 80, 90]
# 0   1   2   3   4   5   6   7   8

C:\test>py test.py
[10, 30, 60, 20, 50, 80, 70, 90, 40]
[10, 20, 30, 40, 50, 60, 70, 80, 90]
Enter key value: 60
Enter start value of search: 0
Enter end value of search: 3
60 ==> -1

C:\test>py test.py
[10, 30, 60, 20, 50, 80, 70, 90, 40]
[10, 20, 30, 40, 50, 60, 70, 80, 90]
Enter key value: 20
Enter start value of search: 0
Enter end value of search: 3
20 ==> 1

C:\test>py test.py
[10, 30, 60, 20, 50, 80, 70, 90, 40]
[10, 20, 30, 40, 50, 60, 70, 80, 90]
Enter key value: 70
Enter start value of search: 3
Enter end value of search: 8
70 ==> 6
```

Ex10: number of occurrences of the given object in sorted list

```
--------------------------------------------------------------
L = [10, 20, 20, 20, 30, 30] , 20 -----> 3
L = [10, 10, 10, 10], 10 --------------> 4
L = [5, 8, 10], 7 --------------------> 0
```

Method1:

```
--------
def count(L,key):
      c=0
      for i in L:
            if key==i:
                  c=c+1
      return c

print(f"[10, 20, 20, 20, 30, 30], 20 => {count([10, 20, 20, 20, 30, 30], 20)}")
print(f"[10, 10, 10, 10], 10 => {count([10, 10, 10, 10], 10)}")
print(f"[5, 8, 10], 7 => {count([5, 8, 10], 7)}")
```

```
C:\test>py test.py
[10, 20, 20, 20, 30, 30], 20 => 3
[10, 10, 10, 10], 10 => 4
[5, 8, 10], 7 => 0
```

Method2:

```
def firstoccurrence(L,key):
      for i in range(len(L)):
            if key == L[i]:
                  return i
      return -1

def lastoccurrence(L,key):
      TL = []
      for i in range(len(L)):
            if key == L[i]:
                  TL.append(i)
      for x in TL[-1:-2:-1]:
            return x

def count(L,key):
      fo = firstoccurrence(L,key)
      if fo==-1:
            return 0
      return lastoccurrence(L,key) - fo + 1

print(f"[10, 20, 20, 20, 30, 30], 20 => {count([10, 20, 20, 20, 30, 30], 20)}")
print(f"[10, 10, 10, 10], 10 => {count([10, 10, 10, 10], 10)}")
print(f"[5, 8, 10], 7 => {count([5, 8, 10], 7)}")
```

```
C:\test>py test.py
[10, 20, 20, 20, 30, 30], 20 => 3
[10, 10, 10, 10], 10 => 4
[5, 8, 10], 7 => 0
```

Chapter:14 --> Linked List

collection of nodes. there are two types of nodes are existed.

1) single node ----> data, next
2) double node ----> prev, data, next

The following are the different types of linked lists

==> Single Linked List
==> Double Linked List
==> Circular Single Linked List
==> Circular Double Linked List

The following are the most commonly performed operations on linked list

01. creation of LL

02. insertion at first location
03. insertion at last location
04. insertion at given location
05. sorted insertion in ASC order
06. sorted insertion in DESC order
07. display or traversing loops
08. display or traversing using recursion
09. size of linked list
10. middle element in linked list
11. delete at first
12. delete at last
13. delete at given location
14. delete an element
15. delete elements
16. search operation - I
17. search operation - II
18. search operation - III
19. search operation - IV
20. remove duplicated in sorted linked list
21. copy the list
22. reverse the linked list
23. compare two linked lists
24. nth node from begining / ending
25. paliandrom or not

JNoteBook ----> Final copy pdf

double linked list
----
01. creation of dll class
02. creation of node class
03. size of dll
05. is empty
06. print list / display iteration
07. print list / display recursion
08. overriding string representation
09. insert at first
10. insert at last
11. insert at location
12. delete at first
13. delete at last
14. delete at location
15. delete element
16. search iteration
17. search recursion
18. reverse operation
19. copy list

```
class dll:

      #node class => DLL
      class node:

            #constructor for node
            def __init__(self,data,next=None,prev=None):
                  self.data = data
                  self.next = next
                  self.prev = prev

      #constructor for dll
      def __init__(self):
            self.head = None
            self.tail = None
            self.count = 0

    #size of dll
    def size(self):
            return self.count

    #is empty
    def isempty(self):
            return self.count==0

    #print dll by using iteration --> loop
    def printlistiterative(self):
            if self.head == None:
                    print("list is empty")
                    return
            currNode = self.head
            while currNode!=None:
                    print(currNode.data,end=" => ")
                    currNode = currNode.next
            print("None")
            return

    #__str__ override
    def __str__(self):
            s = ""
            currNode = self.head
            while currNode!=None:
                    s=s+f"{currNode.data} => "
                    currNode = currNode.next
            s = s + "None"
            return s

    #print dll by using recursion --> recursion
    def printlistrecursion(self,temp):
            if temp==None:
                    print("None")
                    return
            print(temp.data,end=" => ")
            self.printlistrecursion(temp.next)

    #insert at first location
    def insertatfirst(self,data):
            newnode = self.node(data,None,None)
            self.count = self.count + 1
            if self.head == None:
                    self.head = newnode
                    self.tail = newnode
                    return
            self.head.prev = newnode
            newnode.next = self.head
            self.head = newnode
            return

    #insert at last location
    def insertatlast(self,data):
            newnode = self.node(data,None,None)
            self.count = self.count + 1
            if self.head==None:
                    self.head = newnode
                    self.tail = newnode
                    return
            newnode.prev = self.tail
            self.tail.next = newnode
            self.tail = newnode
```

```
        return
    #insert at given location
    def insertatlocation(self,index,data):
        if index<0 or index>self.size():
            print("out of range")
            return
        newnode = self.node(data,None,None)
        self.count = self.count + 1
        if index==0:
            self.head.prev = newnode
            newnode.next = self.head
            self.head = newnode
            return
        if index==self.count-1:
            newnode.prev = self.tail
            self.tail.next = newnode
            self.tail = newnode
            return
        temp1 = self.head
        temp2 = None
        i = 0
        while temp1!=None and i<index:
            temp2 = temp1
            temp1 = temp1.next
            i=i+1
        temp2.next = newnode
        newnode.prev = temp2
        newnode.next = temp1
        temp1.prev = newnode
    #delete at first
    def deleteatfirst(self):
        if self.head==None:
            print("list is empty")
            return
        self.count = self.count - 1
        self.head = self.head.next
        if self.head!=None:
            self.head.prev = None

    #delete at last location
    def deleteatlast(self):
        if self.head==None:
            print("list is empty")
            return
        self.count = self.count -1
        temp = self.tail.prev
        temp.next = None
        self.tail.prev = None
        self.tail = temp
        return

    #delete at given location
    def deleteatlocation(self,index):
        if self.head==None:
            print("list is empty")
            return
        if index<0 or index>=self.count:
            print("out of range")
            return
        if index==0:
            self.count = self.count - 1
            self.head = self.head.next
            if self.head!=None:
                self.head.prev = None
```

```
                  return
            if index==self.count-1:
                  self.count = self.count -1
                  temp = self.tail.prev
                  temp.next = None
                  self.tail.prev = None
                  self.tail = temp
                  return
            i=1
            temp1 = self.head
            while temp1.next!=None and i<=index:
                  if i==index:
                        temp1.next = temp1.next.next
                        temp2 = temp1.next
                        if temp2!=None:
                              temp2.prev = temp1
                        self.count = self.count - 1
                        return
                  temp2 = temp1
                  temp1 = temp1.next
                  i=i+1

      #deleteing element
      def deleteelement(self,data):
            if self.head==None:
                  print("list is empty")
                  return
            if self.head.data==data:
                  self.head = self.head.next
                  if self.head!=None:
                        self.head.prev = None
                  self.count = self.count - 1
                  return
            temp1 = self.head
            temp2 = None
            while temp1.next != None:
                  if temp1.next.data == data:
                        temp1.next = temp1.next.next
                        temp2 = temp1.next
                        if temp2!=None:
                              temp2.prev = temp1
                        else:
                              self.tail = temp1
                        self.count = self.count - 1
                        return
                  temp1 = temp1.next

      #search for element
      def search1(self,data):
            currNode = self.head
            while currNode!=None:
                  if currNode.data == data:
                        return True
                  currNode = currNode.next
            return False

      #search for element
      def search2(self,data):
            currNode = self.head
            i=0
            while currNode!=None:
                  if currNode.data == data:
                        return i
                  i=i+1
                  currNode = currNode.next
            return -1

      #copy list
      def copylist(self):
            headNode = None
            tailNode = None
            tempNode = None
            currNode = self.head
            if currNode == None:
                  return None
            headNode = self.node(currNode.data,None,None)
            tailNode = headNode
            currNode = currNode.next
            while currNode!=None:
                  tempNode = self.node(currNode.data,None,None)
                  tailNode.next = tempNode
                  tempNode.prev = tailNode
                  tailNode = tempNode
                  currNode = currNode.next
            newlist = dll()
            newlist.head = headNode
            return newlist

      #reverse
      def reverse(self):
            temp = None
            currNode = self.head
            while currNode!=None:
                  temp = currNode.prev
                  currNode.prev = currNode.next
                  currNode.next = temp
                  currNode = currNode.prev
            if temp!=None:
                  self.head = temp.prev

list = dll()
list.insertatlast(111)
list.insertatlast(222)
list.insertatlast(333)
list.insertatlast(444)
list.insertatlast(555)
list.insertatlast(666)
list.insertatlast(777)
print(list)
list.reverse()
print(list)

circular single linked list
---------------------------
class csll:

      class node:

            def __init__(self,data,next=None):
                  self.data = data
                  self.next = None

      def __init__(self):
            self.tail = None
            self.count = 0

      def size(self):
            return self.count
```

```
#display
def printlist(self):
      if self.tail==None:
            print("list is empty")
            return
      currNode = self.tail.next
      while currNode!=self.tail:
            print(currNode.data,end=" ")
            currNode=currNode.next
      print(currNode.data)

#insert at first
def insertatfirst(self,data):
      newnode = self.node(data,None)
      self.count = self.count + 1
      if self.tail == None:
            self.tail = newnode
            newnode.next = newnode
            return
      newnode.next = self.tail.next
      self.tail.next = newnode
      return

#insert at last
def insertatlast(self,data):
      newnode = self.node(data,None)
      self.count = self.count + 1
      if self.tail == None:
            self.tail = newnode
            newnode.next = newnode
            return
      newnode.next = self.tail.next
      self.tail.next = newnode
      self.tail = newnode
      return

#deletion from begining
def deleteatfirst(self):
      if self.count == 0:
            print("list is empty")
            return
      self.count = self.count - 1
      if self.tail == self.tail.next:
            self.tail = None
            return
      self.tail.next = self.tail.next.next

#deletion from ending
def deleteatlast(self):
      if self.count==0:
            print("list is empty")
            return
      self.count = self.count -1
      if self.tail == self.tail.next:
            self.tail = None
            return
      currNode = self.tail.next
      while currNode.next != self.tail:
            currNode = currNode.next
      currNode.next = self.tail.next
      self.tail = currNode
```

```
obj = csll()
obj.insertatlast(333)
obj.insertatlast(444)
obj.insertatlast(555)
obj.insertatfirst(222)
obj.insertatfirst(111)
obj.printlist()
obj.deleteatlast()
obj.printlist()

circular double linked list
---------------------------
class cdll:

      class node:

            def __init__(self,data,next=None,prev=None):
                  self.data = data
                  self.next = next
                  self.prev = prev

      def __init__(self):
            self.head = None
            self.tail = None
            self.count = 0

      def size(self):
            return self.count

      #display
      def printlist(self):
            if self.tail==None:
                  print("list is empty")
                  return
            currNode = self.tail.next
            while currNode!=self.tail:
                  print(currNode.data,end=" ")
                  currNode = currNode.next
            print(currNode.data)

      #insert at first
      def insertatfirst(self,data):
            newnode = self.node(data,None,None)
            self.count = self.count + 1
            if self.tail == None:
                  self.tail = newnode
                  self.head = newnode
                  newnode.next = newnode
                  newnode.prev = newnode
                  return
            newnode.next = self.head
            newnode.prev = self.head.prev
            self.head.prev = newnode
            newnode.prev.next = newnode
            self.head = newnode
            return

      #insert at last
      def insertatlast(self,data):
            newnode = self.node(data,None,None)
            self.count = self.count + 1
            if self.tail == None:
                  self.tail = newnode
                  self.head = newnode
                  newnode.next = newnode
                  newnode.prev = newnode
                  return
            newnode.next = self.tail.next
            newnode.prev = self.tail
            self.tail.next = newnode
            newnode.next.prev = newnode
            self.tail = newnode

      #delete at first
      def deleteatfirst(self):
            if self.count == 0:
                  print("list is empty")
                  return
            self.count = self.count -1
            if self.count == 0:
                  self.head = None
                  self.tail = None
                  return
            temp = self.head.next
            temp.prev = self.tail
            self.tail.next = temp
            self.head = temp

      #delete at last
      def deleteatlast(self):
            if self.count == 0:
                  print("list is empty")
                  return
            self.count = self.count -1
            if self.count == 0:
                  self.tail = None
                  self.head = None
                  return
            temp = self.tail.prev
            temp.next = self.head
            self.head.prev = temp
            self.tail = temp


obj = cdll()
obj.insertatfirst(333)
obj.insertatfirst(222)
obj.insertatfirst(111)
obj.insertatlast(444)
obj.insertatlast(555)
obj.insertatlast(666)
obj.printlist()
obj.deleteatlast()
obj.printlist()
```

```
C:\test>py test.py
111 222 333 444 555 666
111 222 333 444 555
```