What is full stack?
=================
The process of developing both front end and back end applications
Front end refers to client side and back end refers to server side

Client side=>websites refers to the web browser
Server side=>where the data and source code is stored

The difference between the front end and back end is that front refers to how the
Web page look likes and back end refers to how the web page work

As a full stack developer we need to work with both front end back end application

3 phases of full stack

01. Front end -user interface
Basic language of front end developments are:

HTML
CSS
javascript
Bootstarap
reactjs

02. Backend:

Transfer the data from one server to another server, transfer the data from one application to another application
Communication between the two servers
Basics language of backend developments are:

java
python
php

03. Database:

Database stores the some amount of data to that application
my sql
ms sql
mangodb

We can connect front end and database with the help of back end.

What is python?
==============
1. High level programming language -it is used various sectors
-web development
-software development
-machine learning
-AI
-robotics
-data science
-data analytics
-Games etc...

Number of sectors
2. Interpreted programming language
Translator ->convert’s python code ->machine code line by line -running time

Debugging is very easy -handle the errors

3. Interactive programming language
Translator ->convert’s python code ->machine code line by line before running program

Communication between the developer and compiler

4. General purpose programming language
by using python we can develop any kind of webapps/websites/software

-web apps
-android apps
-Ios apps
-Gui
-desktop apps
etc....

=====Advantages====

1. Very easy
2. Syntax is very less
3. Cross platform -it supports multiple os
4. Portable language -if we write code in windows same code we can run in Linux   also
5. Very short time we can develop complex apps
6.100+ libraries
7. Increase the productivity

====Dis Advantages=====
1. Slow speed -code execute line by line
2. Run time errors- identify the errors

Libraries and frame works both are predefined functions

In python we are using Django, tkinter frame works ->web apps/websites

By using python we can develop thousands of applications
1. instagram
2. spotify
3. intel -testing purpose
4. google -web search system purpose
5. youtube -video sharing
etc...

python.org

===python setup====

Python has been installed or not we need to check with the command prompt

Command prompt ->python

Python 3.13 installed successfully

IDLE -code editor

Create new file

file_neme.py
data.py
Bhaskar.py

=====comment lines====

1. It is used to make non executable code, never execute
2. It used to make description about task

2 types

#1.single line comment -#
#2.multiline comment -""" """ or ''' '''

======print statement====

-it is used to print any kind of data
-it is a predefined function

Print ()-already defined

Print (100)
100
Print (20.5)
20.5

======Data types====

1. Primitive data types
In single container we can store only value
-number
-string
-Boolean

1. Number

int -normal numbers -1,2,3,4,5,6
Float -decimal numbers -2.3,5.6,2.6

2. String

-it is a group of alpha numeric characters
-string should be declare "" or ''
-123,'123' we cannot able to do mathematical calculations

3. Boolean

-True =1,Flase=0
-declare without "" or ''
-True or False

```
print(100)
print(29.7)
print("welcome Besant")
print(True)
```

====how to check the data type=====

Type () ->predefined function

```
Print (type(100))
Print (type(4.6))
print (type('welcome'))
print(type(True))
```

=====Variable=====

-Container -blocks/groups
-variable name must be start with a-z or _
-it is used to store the data
  ex:abc or _abc
-variable case sensitive

```
'''
a=100
print(a)
print(type(a))
'''
'''
b=3.4
print(b)
print(type(b))
'''
'''
c='welcome'
print(c)
print(type(c))
'''
d=True
print(d)
print(type(d))
```

delete =del

```
a='hello'
del a
a=100
print(a)
```

We can easily find errors also
The code will be updated every next line

Input (label) ->it is used to get the data from the user

```
a=bool(input("ENTER value: "))
print(a)
print(type(a))
```

Indexing vs slicing
-we want particular value is called indexing
-range of value we can use slicing

position ->0

```
a='welcome'
print(a[3])

#str[start:end:step]

a='welcome'
print(a[0:4:2])

#a[0]+a[1]+a[2]+a[3]
```

'''
Data types

2. **Non primitive data types**

In single container we can store n number of values
4 types
1.List
2.tuple
3.set
4.Dictionary

========List========
-collection of values
-in single container we can store multiple values
-list should be declare -[1,2,2.3,'python','besant',True]
-list is mutable object -we can modify
'''
'''

```
li=[1,2,2.3,'python','besant',True]
print(li)
print(type(li))
'''
#len() ->it is used to check the how many values present in the list
#position ->0
#length->1
'''
li=[1,2,3,4.5,'python','besant',True]
print(li)
print(li[4])
print(len(li))
'''
#list methods ->predefined functions

#01.append () ->it is used add one more value end of the list
 #list.append(value)
'''
li=[1,2,3,4,5]
li.append('python')
li.append('Besant')
print(li)
'''
#02.extend () ->it is used to merge 2 lists -add end of the value
 #list1.extend(list2)
'''
li1=[1,2,3,4,5]
li2=[6,7,8,9]
li1.extend(li2)
print(li1)
'''
#03.insert () ->it is used to add one more value any where
 #list.insert(where[position],value)

li=[1,2,3,4,5,6]
li.insert(2,'python')
li.insert(0,'besant')
li.insert(4,'bhaskar')
print(li)
```

```
#04.pop () ->it is used to delete particular value based on the position
 #list.pop(position)
'''
li=[1,2,3,4,5,6]
li.pop()
print(li)
'''
#05.remove () ->it is used to delete particular value based on the value
 #list.remove(value)
'''
li=[1,2,3,4,5]
li.remove(3)
print(li)
'''
#06.sort () ->it used to make ascending order
 #list.sort()
'''
li=[1,2,6,7,12,34,5,6,78]
li.sort()
print(li)
'''
#07.reverse () ->it is used to make reverse
 #list.reverse()
'''
li=[1,2,3,4,5,6,7,8,9]
li.reverse()
print(li)
'''
#08.index() ->it is used to find the position based on the value
 #list.index(value)
'''
li=[1,2,3,45,6,7,9]
x=li.index(45)
print(x)
'''
#09.max(),min(),sum()
'''
li=[1,2,3,4,5]
print(max(li))
print(min(li))
print(sum(li))
'''
'''
```

======tuple====

```
-collection of values
-in single container we can store multiple values
-tuple should be declare -(1,2,23.4,'python',bhaskar',True)
-tuple is a immutable object -we cant modify
'''
'''
t=(1,2,2.3,'python','Bhaskar',True)
print(t)
print(type(t))
'''
'''
t=(1,2,3,4,5,6)
print(max(t))
print(min(t))
print(sum(t))
'''
'''
t=(1,2,3,47,8,9)
y=t.index(47)
print(y)
'''
#count() ->it used to check the value how many times present in the tuple
'''
t=(1,2,3,4,5,1,2,3,4,6,1,7,8,1)
x=t.count(1)
print(x)
'''
'''
li=[1,2,34,4,5,7,1,5,1]
x=li.count(1)
print(x)
'''
#packing = tuple
#unpacking =every single value assign one indivisual variable
'''
t=(1,2,3,4,5,6,7,8,9,4)
```

```
_,b,*c,d,e=t

print(b)
print(c)
print(d)
print(e)
'''
#rest operator -* -rest of all the values
 #returns as a list -[]

#_ =>null operator =>empty statemet


'''
```
Data types
=======set======
-collection of data
-in single variable we can store multiple values
-set should be declare -{}
-set is a mutable object -we cant modify

Advantages
----
1. Set values defaultly it will comes ascending order
2. Set not allows duplicate values

```
'''
'''
s={1,2,3,4,1,2,3,6,7,8,12,4,9}
print(s)
print(type(s))
'''
#1.add() ->it is used to add more value end of the set
 #set.add(value)
'''
s={1,2,3,4,5,6}
s.add('python')
s.add('Besant')
print(s)
'''
#2.clear() ->it is used to make empty set
```

```
 #set.clear()
'''
s={1,2,3,4,5,8,9}
s.clear()
print(s)
'''
#3.discard() ->it used to delete the particular value based on the value
 #set.discard(value)
'''
s={1,2,34,5,6,78}
s.discard(34)
print(s)
'''
#04.difference() ->it takes 2 sets ->only takes one set difference-unmatched
'''
set1={1,2,3}
set2={1,2,23,6,7,8}
res=set1.difference(set2)
print(res)
'''
#05.symmetric_difference ->it takes 2 sets ->difference of two sets -
>unmatched
'''
set1={1,2,3}
set2={1,23,6,7,8}
res=set1.symmetric_difference(set2)
print(res)
'''
#06.intersection() ->it takes 2  more sets ->common values
'''
set1={1,2,3,4}
set2={1,2,6,7,8}
set3={1}
res=set1.intersection(set2,set3)
print(res)
'''
'''
Dictionary:collection of keys and value pair
"name":'bhaskar'
-Dictionary should be declare -{}
```

```
-dictionary is a mutable object - we can modify

{"key":value,"key":value}
'''
'''
obj={}
print(obj)
print(type(obj))
'''
'''
obj={"name":'bhaskar',
     "email":'bhaskar@gmail',
     "mobile":96785645
     }
print(obj)
print(type(obj))
'''
#get() ->it used to get the particular value based on the key
 #dict.get(key)
'''
obj={
    "name":'bhaskar',
    "email":'bhaskar@gmail',
    "mobile":8786756454
    }
print(obj.get("name"))
print(obj.get("email"))
'''
'''
obj={
    "name":['bhaskar','chinni','madhu'],
    "email":'bhaskar@gmail',
    "mobile":9675645
    }
li=obj.get("name")
print(li[0])
'''
#1.add() ->it is used to add one more key and value
 #dict['key']=value
'''
```

```
obj={
    "name":'bhaskar',
    "email":'bhskar@gmail',
    "mobile":962761251
    }
obj['city']='bangalore'
obj['state']='KA'
print(obj)
'''
#2.pop() ->it is used delete particular valu based on the key
 #dict.pop(key)

obj={
    "name":'bhaskar',
    "email":'bhaskar@gmail',
    "mobile":98675545
    }
obj.pop("name")
obj.pop("email")
print(obj)

#03.update() ->it is used to update partcular value based on the key
 #dict.update({key:value})
'''
obj={
    "name":'chinni',
    "email":"chinni@123",
    "mobile":7867545
    }
obj.update({"name":'bhakskar'})
obj.update({"email":'bahskar@'})
print(obj)
'''
```

'''

Operators

Operators are symboles tod do some mathematical or logical operations
There are 5 types

1. Arithmetic operator
2. comparision operator
3. Logical operator
4. Membership operator
5. Identity operator

======1.Arithmetic operator========

Mathematical calculation
+,-,,/,*,%

```
'''
'''
a=10
b=20
print(a+b)
print(a-b)
print(a*b)
print(b/a)
print(b**a)
print(4 %2)
print(6 %2)
print(7%2)

#even remainder by 2 =0
#odd remainder by 2 =1
'''
'''
```

2. comparision operator

======================

-it used to compare the values
-it will returns as a boolean
<,<=,>,>=,!=,==

```
'''
'''
```

a=100
b=200
print(a<b)
print(a<=b)
print(a>b)
print(a>=b)
print(a!=b)
'''
#it is used to check the value and data type
'''
x=10
y='10'
print(x==y)
'''
'''
3. Logical operator
==================
and -&& ,or -||

and=>if both operators are true - condition becomes true
or =>if any one operator are true ->condition becomes true
'''
'''
x=100
y=200
z=300
print(x==100 and y<=200 and z>300)
print(x==100 or y<=200 or z>300)
'''
'''
4. Membership operator
======================
-it is used to check the sequence
-it will return as a boolean
-==,!=
-in,not in
'''
'''
x='hello'
print('h' in x)

```
print('H' in x)
print('H' not in x)
'''
'''
li=['a','b','c','d']
print('a' in li)
print('A' in li)
'''
'''
```

5. Identity operator

```
===================
-it is used to check the memory location
-it will returns boolen
-is,is not
'''
'''
x=10
y=20
print(x is not y)
'''
'''
x=10
y=20
x=y
print(x is y)
'''
'''
```

=======conditional statements=====

Depends on the condition i just want to execute program,some statement is
Called as conditional statements

```
if,elif,else

if(condition):
  statemet
elif(condition):
  statemt
else:
  statemet
'''
'''
a=100
if(a<100):
   print('i am passed')
elif(a>100):
   print('i am correct')
else:
   print('i am failed')
'''
'''
user=int(input("enter value:"))
if(user %2 ==0):
   print(user,'i am even')
elif(user %2==1):
   print(user,'i am odd')
else:
   print('i am not a number')
'''
#nested if -child element
'''
x=100
y=200
z=300
if(x<=100 and y>=200):
   print('total:',x+y)
   if(z<300):
      print('total:',x+y+z)
elif(x<100 or y>200):
   print('sum:',x+y)
else:
   print('division:',y/x)
'''
'''
user1=int(input('enter value1:'))#10 #30
user2=int(input('enter value2:'))#20 #10
#maximum value
if(user1 > user2):
   print('maximum value is:',user2)
else:
   print('maximum value is:',user1)
''''
x=100
y=200
z=300
if(x<100 and y>200):
   if(z<300):
       print('total:',x+y+z)
elif(x<100 or y>200):
   print('sum:',x+y)
else:
   print('division:',y/x)
'''
'''
```

Loops:
======
It execute repetedly untill the condtion is true/false
2 types
#1.for loop
#2.while loop

#for loop:it is a single line expression /one statement

```
for loop_name in expression:
   satement

for    =>keyword
loop_name=>anything
in      =>keyword
expression =>range(start,end)

range(1,20)=>(1,2,3,4,5,6....19)
'''
'''
for i in range(1,10,1):
   print(i)
'''
'''
for j in range(10,1,-1):
   print(j)
'''
'''
for i in range(1,10):
   if(i %2==0):
      print(i)
'''
'''
1%2=0 =>false
2%2=0 =>true
3%2=0 =>false
'''
'''
for j in range(10,1,-1):
    if(j %2==1):
        print(j)
'''
'''
for i in range(1,20,1):
    if(i%2==0):
        print(i,'=is a even')
    else:
        print(i,'=is a odd')
'''
'''
li=[1,2,3,1,2,23,4,5,65,7,8]
for i in range(0,len(li),1):
    print(i)
'''
'''
li=[1,233,3,4,5,6,7,1]
for i in range(0,len(li),1):
    print(li[i])
'''
'''
li=[1,2,3,4,12,3,45,67,86]
for i in range(0,len(li),1):
    if(li[i]%2==0):
        print(li[i])
'''
'''
li=[1,2,3,4,12,3,45,67,86]
for i in range(0,len(li),1):
    if(li[i]%2==1):
        print(li[i])
'''
'''
```

while loop:

=============

-starting control -condtion started starting only

```
starting_value
while(ending_value):
    statement
    step
'''
'''
i=1
while(i<=20):
    print(i)
    i=i+1
'''
'''
1<=20 =true
2<=20 =true
3<=20 =true
.
.
20<=20=true
'''
'''
i=1
while(i<=20):
    if(i%2==0):
        print(i)
    i=i+1
'''
'''
i=1
while(i<=20):
    if(i%2==1):
        print(i)
    i=i+1
'''
'''
s=0
i=1
while (i<=20):
    if(i%2==0):
        s=s+i
    i=i+1
print(s)
'''
'''
```

```
s=0
i=1
while(i<=20):
   if(i%2==1):
      s=s+i
   i=i+1
print(s)
'''
break:it is used to stop the iterated values/just stop the loops
continue:it skip the iterated values
'''
'''
for i in range(1,20,1):
   if(i==10):
      break
   print(i)
'''
'''
for i in range(1,20,1):
   if(i==10):
      continue
   print(i)
'''
'''
for i in range(1,20,1):
   if(i==10 or i==12 or i==15):
      continue
   print(i)
'''
'''
assert statement:
if condition is true,the program continues to run
if condition is false,the program raises error
'''
'''
a=10
b=0
print(a/b)
'''
'''
```

```
nested loop
inside loop one more loop that is called as nested loop
#j is help us to extract the values
#end="" =>its converts vertical to horizontal
#"\r" =>it help us to create break
'''
'''
for i in range(1,6,1):
   for j in range(1,6,1):
      print(i,end="")
   print("\r")
'''
'''
11111
22222
33333
44444
55555
'''
'''
for i in range(1,6,1):
   for j in range(1,6,1):
      print('*',end="")
   print("\r")
'''
'''
*****
*****
*****
*****
*****
'''
'''
#left angle traingle
for i in range(1,6,1):
   for j in range(1,i,1):
      print(i,end="")
   print("\r")
'''
'''
```

```
2
33
444
5555
'''
'''
for i in range(1,6,1):
    for j in range(1,i,1):
        print('*',end="")
    print("\r")
'''
'''
*
**
***
****
'''
#right angle traingle
'''
for i in range(1,6,1):
    for k in range(1,6-i,1):
        print(end=" ")
    for j in range(1,i,1):
        print(i,end="")
    print("\r")
'''
'''
   2
  33
 444
5555
'''
'''
for i in range(1,6,1):
    for k in range(1,6-i,1):
        print(end=" ")
    for j in range(1,i,1):
        print('*',end="")
    print("\r")
'''
```

```
'''
   *
  **
 ***
****
'''
#traingle
'''
for i in range(1,6,1):
    for k in range(1,6-i,1):
        print(end=" ")
    for j in range(1,i,1):
        print(i,end=" ")
    print("\r")
'''
'''
   2
  3 3
 4 4 4
5 5 5 5
'''
'''
for i in range(1,6,1):
    for k in range(1,6-i,1):
        print(end=" ")
    for j in range(1,i,1):
        print('*',end=" ")
    print("\r")
'''
'''
   *
  * *
 * * *
* * * *
'''
#reverse traingle
'''
for i in range(6,1,-1):
    for k in range(1,6-i,1):
        print(end=" ")
    for j in range(1,i,1):
        print('*',end=" ")
    print("\r")
'''
#diamond shape
'''
for i in range(1,6,1):
    for k in range(1,6-i,1):
        print(end=" ")
    for j in range(1,i,1):
        print('*',end=" ")
    print("\r")
for i in range(6,1,-1):
    for k in range(1,6-i,1):
        print(end=" ")
    for j in range(1,i,1):
        print('*',end=" ")
    print("\r")
'''
'''
for i in range(1,8,1):
    for k in range(1,8-i,1):
        print(end=" ")
    for j in range(1,i,1):
        print('*',end=" ")
    print("\r")
for i in range(8,1,-1):
    for k in range(1,8-i,1):
        print(end=" ")
    for j in range(1,i,1):
        print('*',end=" ")
    print("\r")
''''''
'''
maximum value:>
minimum value:<
'''
'''
li=[1,2,3,12,45,67,34,89]
m=li[0]
```

```
for i in range(0,len(li),1):
    if(li[i]<m):
        m=li[i]
print(m)
'''
#sum of given list
'''
li=[1,2,3,4,5,12,67,4]
s=0
for i in range(0,len(li),1):
    s=s+li[i]
print(s)

#0+1+0+2+0+3
'''
#average
#total/number of values
'''
li=[1,2,3,4,5,6,7,8]
s=0
for i in range(0,len(li),1):
    s=s+li[i]
print(s/li[i])
'''
#duplicate string
'''
x='hello'
#helo
s=''
for i in range(0,len(x),1):
    if(x[i] not in s):
        s=s+x[i]
print(s)
'''
'''
'h' not in ''=true
'e' not in 'h'=true
'l' not in 'he'=true
'l' not in 'hel'=false
'o' not in 'hel'=true
```

```
'''
#duplicate number
'''
li=[1,2,3,1,5,6,7,8,9,3,2,1,4,5]
res=[]
for i in range(0,len(li),1):
    if(li[i] not in res):
        res.append(li[i])
print(res)

'''
'''
1 not in []=true
2 not in [1]=true
3 not in [1,2]=true
1 not in [1,2,3]=false
5 not in [1,2,3]=true
6 not in [1,2,3,5]=true
'''
#reverse string
'''
x='hello'
#olleh
res=""
for i in range(len(x),0,-1):
    res=res+x[i-1]
print(res)
'''
#reverse number
'''
x='12345'
#54321
res=""
for i in range(len(x),0,-1):
    res=res+x[i-1]
    y=int(res)
print(y)
print(type(y))
'''
#given value is palindrome or not
```

```
#the given input reverse output same that is called palindrome
'''
x='malayalam'
res=""
for i in range(len(x),0,-1):
   res=res+x[i-1]
if(x==res):
   print(x,'is a palindrome')
else:
   print(x,'is not a palindrome')
'''
#fibanocci series
#it is used to find the next values
#0 1 default of fibanocci series
#0 1 1 2 3 5 update
'''
n1=0
n2=1
#n3?
for i in range(1,6,1):
   print(n1)
   n3=n1+n2
   n1=n2
   n2=n3
'''
#prime or not
#its divisibile by 1 or itself
'''
n=13
x=100
for i in range(2,n,1):
   if(n%i==0):
      x=200
if(x==100):
   print(n,'is a prime')
else:
   print(n,'is not a prime')
'''
'''
13%2==0 =>false
```

13%3==0 =>false
13%4==0 =>false
.
.
13%12==0 =>false
condition=true=200 =not a prime
condtiion =false=100 =is a prime
‘’’
'''

Function
----
it is nothing but group of statements/block of code is called as function
-it will perform specific task

2 types
1.userdefined function:customized function:based on the user requirement
 user can create own function based on user requirements is called as user
 requirements

2.Built_in functions:predefined function

```
def Function_name(parameters):
   block of code
calling_function(Arguments)
```

```
def              =>key function
Function_name   =>camelcase:myNameBhaskar
                 pascal case:MyNameBhaskar
parameters      =>keys
Arguments       =>values
calling_function=>function_name
'''
'''
def myFunction():
   print('starting loop')
   i=1
   while(i<=5):
       print(i)
       i=i+1
   else:
        print('ending function')
myFunction()
'''
'''
def myData():
   x=100 #local variable
   if(x%2==0):
       print(x,'is a even')
   else:
       print(x,'is a odd')
myData()
'''
'''
x=100 #global variable
def myData():
    if(x%2==0):
        print(x,'is a even')
    else:
        print(x,'is a odd')
#myData()
'''
#global varibale/global scope
#it is outside the function
#global variable access anywhere
#local variable/local scope
#it is inside the function
#it access only in local
'''
'''
a=10
def outer():
    b=20
    print(a+b)
outer()
'''
#global:converts local to global
'''
a=10
def outer():
    global b
    b=20
outer()
print(a+b)
'''
#parameters:keys
#Arguments:values
'''
def addData(a,b):
    print(a+b)
addData(1,2)
addData(12,10)
'''
'''
def employe(id,name,email,*mobile):
    print('ID:',id)
    print('Name:',name)
    print('Email:',email)
    print('Mobile:',mobile)
employe(1,['bhsakar','chinni','madhu'],'bhsakar@',637262837,8736276)
employe(2,'chinni','chinni@',2367268723)

#rest parameter -*
#rest of all the values
#it will return as a tuple -()
#it applicable only last parameters
'''
'''
lambda Function:
================
-there is no function
-it is single line expression
-self invoking method:it will invoke automatically -()
-lambda parameters:statements
'''
'''
def main(a,b):
   print(a+b)
main(1,2)
'''
'''
x=lambda a,b:a+b
print(x(1,2))
print(x(12,10))
'''
#lambda function with map&filter
#lambda map():it is used to map the values
'''
li=[1,2,3,4,5]
```

```
a=list(map(lambda n:n*2,li))
print(a)

b=list(map(lambda x:x+2,li))
print(b)
'''
#lambda filter():it is used to filter the values
'''
li=[1,2,3,4,5,6,7,8,9,10]
a=list(filter(lambda y:y<=5,li))
print(a)

b=list(filter(lambda y:y>=5,li))
print(b)
'''
'''
Exceptional handling:handling the errors-run time errors it will handle
try,except,else,finally -by using blocks only we can handle the errors

try:block of code -we can pass n number of statements
except:exceptional :handle the errors
else:if try is any error except will be excute
     if try no error else will be executed
finally:common operation we are using finally
'''
'''
try:
  block of statements
except:
   handle the errors
else:
   success
finally:
   always
'''
'''
a=10
b=0
print(a/b)
'''
```

```
'''
try:
  a=10
  b=0
  print(a/b)
except:
    print('i am getting error')
else:
    print('executed sucessfully')
finally:
    print('always')
'''
'''
li=[1,2,3,4,5]
print(li[6])
'''
'''
Inbuilt function:predefined functions
#ZeroDivisionError
#ValueError
#IndexError
#NameError
#TypeError
'''
'''
try:
  li=[1,2,3,4,5]
  print(li[6])
except ZeroDivisionError:
    print('it is zerodivision error')
except IndexError:
    print('it is a index error')
except:
    print('i am getting error')
'''
'''
try:
  a=10
  b=10
  print(a/b)
```

```
except IndexError:
    print('it is index error')
except NameError:
    print('it is a name error')
except ZeroDivisionError:
    print('it is zero division error')
else:
    print('executed successfully')
finally:
    print('always')
'''
'''
try:
  def employe(id,name,email,mobile):
    print('ID:',id)
    print('Name:',name)
    print('Email:',email)
    print('Mobile:',mobile)
  employe(1,'bhaskar','bhaskar@',92373723)
except TypeError:
    print('it is a type error')
else:
    print('executed successfully')
finally:
    print('always')
'''
'''
callback function
-calling back function
-one function passed into another function as an argument is called as
  callback functions
'''
'''
def A():
    print('A fun')
A()
def B():
    print('B fun')
B()
'''
```

```
#i can call only one function,remaining i can call call back
#B fun passed into A fun as an argument
'''
def A():
   print('A fun')
A()
def B():
   print('B fun')
A(B)
'''
#x=B
#B is send to A ,receiveing as a parameter,that parameter name is x,x is
 #called as call back function of B fun
'''
def A(x):
   print('A fun')
   x() #call back function of B fun
def B():
   print('B fun')
A(B)
'''
'''
def A(second,third):
   print('A fun')
   second() #callback function of B fun
   third()  #callback function of C fun
def B():
   print('B fun')
def C():
   print('C fun')
A(B,C)
'''
'''
def A(second,third):
   print('A fun')
   second(1,2)
   thrid(1,2,3)
def B(x,y):
   print('B fun:',x+y)
def C(x,y,z):
    print('C fun:',x+y+z)
A(B,C)
'''
'''
```

File handling:used to perform CRUD operations
#Create:create the new file &store the data -write
#Read: Read the data in the existing file   -read
#Update:update the data in the existing file-read
#Delete:Delete the data in the existing file-read

per that perspective here file handling provides some modes
#modes
there are 3 types of modes are
write,read,append
write-w
read-r
append-a

#create

open()->predefined function

```
open('new_file.ext')
open('new_file.ext','mode')

#modes
new_file=open('data.text','w')


new_file.close()
```

#now i just want store the information dynamically

#1.write():takes only one argument
#2.writelines():collection of values

```
'''
'''
new_file=open('data.text','w')
new_file.write('python\n')
new_file.write('java\n')
new_file.write('bhaskar\n')
new_file.write('css\n')
new_file.write('Html\n')


new_file.close()
'''
'''
new_file=open('data.text','w')
li=['python\n','java\n','chinni\n','bhaskar\n','madhu\n']
new_file.writelines(li)
new_file.close()
'''
'''
new_file=open('data.text','w')
li=['python','bahskar','madhu','chinni','css','javascript','aws']
for i in li:
   new_file.writelines(i)
   new_file.writelines('\n')

new_file.close()
'''

#Read:per that perspective we can use some modes
#read():returns as a same existing structure
#readlines():return as a list
#readline():returns only one row
'''
file=open('data.text','r')
res=file.read()
print(res)
'''
'''
file=open('data.text','r')
res=file.readlines()
print(res)
'''
'''
file=open('data.text','r')
res=file.readline()
```

```
print(res)
'''
#list -we can add,update,change
'''
file=open('data.text','r')
res=file.readlines()
res.pop()
print(res)
'''
'''
file=open('data.text','r')
res=file.readlines()
res.insert(2,'reactjs')
print(res)
'''
'''
file=open('data.text','r')
res=file.readlines()
res.sort()
print(res)
'''
#append():used to add one more value end of the list
'''
new_file=open('data.text','w')
new_file.write('python\n')
new_file.write('java\n')
new_file.write('bhaskar\n')
new_file.write('css\n')
new_file.write('Html\n')


new_file.close()
'''
'''
new_file=open('data.text','a')
li=['python','bahskar','madhu','chinni','css','javascript','aws']
for i in li:
   new_file.writelines(i)
   new_file.writelines('\n')
new_file.close()
'''
'''
import os
os.remove('data.text')
'''
'''
Generators:Functions
-it is used to generate/iterate the values
-when ever user requirement that purpose we are using generators

li=[1,2,3,4,5,6,7,8,9,10,3,2,3,4,5,6,7,8]
for i in li:
    print('token:',i)

2 types
#1.infinite generator:n number of tokens we can generate
#2.custom generator:maximum 40,maximum 100,maximum 1000

as per today date we generate 10 tokens,we sale 5 tokens,remaing 5 tokens
we cant use tomorrow...now if use generators means we can over come that

now a days bus conductor using machine,before bundle of tickets,now
we can use genartor genarte a tickets when ever customer requirements that
is called as generators
'''
#return:it takes only one value
#      :one function-one return
'''
def gen():
  return 'hello'
  return 'hi'
  return 'bhaskar'
print(gen())
'''
#yield=return
#it takes collection of values
#multiple yield statement
'''
def gen():
    yield 1
    yield 2
    yield 3
print(gen())
'''
#i just want iterate 1 by 1 when ever user requirement
#iter(gen_fun):it is used to iterate the exitsing generator object
#next(iter_value):it is used to generate the values
'''
def gen():
    yield 1
    yield 2
    yield 3
x=gen()
n=iter(x)
print(next(n))
print(next(n))
print(next(n))
'''
#geeartor wont return at a time,generator return 1 by 1 when ever user
 #requirement
'''
def custom():
    for i in range(1,10,1):
        return i
print(custom())
'''
'''
def custom():
    for i in range(1,10,1):
        yield i
print(custom())
'''
'''
def custom():
    for i in range(1,10,1):
        yield i
y=custom()
n=iter(y)
print(next(n))
print(next(n))
print(next(n))
print(next(n))
print(next(n))
print(next(n))
print(next(n))
print(next(n))
print(next(n))
print(next(n))
print(next(n))
print(next(n))
'''
'''
def infinite():
    i=1
    while(i<=10):
        print(i)
        i=i+1
infinite()
'''
'''
def infinite():
    i=1
    while(True):
        print(i)
        i=i+1
infinite()
'''
'''
def infinite():
    i=1
    while(True):
        yield i
        i=i+1
x=infinite()
n=iter(x)
print(next(n))
print(next(n))
print(next(n))
print(next(n))
print(next(n))
print(next(n))
print(next(n))
print(next(n))
print(next(n))
print(next(n))
print(next(n))
print(next(n))
'''
'''
```

Threadings:Functions
-it is used to increase the application performance
-it is used to execute the program paralelly
-interpreted programming -execute step by step
-if you have a 10 lines of code it will take 2 sec time
-if you have a 1000 lines of complete applicaton code it will take more
 time
-it is used to make faster program

2 types
#1.multi level threading:_thread   =>pacakage only we can achieve
#2.multiple threading   :threading =>package only we can achieve

fun A +fun B +fun C

```
'''
#1.multi level threading:
'''
def A():
   print('A fun')
A()
def B():
    print('B fun')
B()
def C():
    print('C fun')
C()
'''
#_thread.start_new_thread(function,(arguments))
'''
import _thread

def A(msg):
    print(msg)
def B(msg):
    print(msg)
def C(msg):
    print(msg)
_thread.start_new_thread(A,('A fun',))
_thread.start_new_thread(B,('B fun',))
_thread.start_new_thread(C,('C fun',))
'''
'''
def A(msg):
    i=1
    while(i<=5):
        print(msg)
        i=i+1
def B(msg):
    i=1
    while(i<=5):
        print(msg)
        i=i+1
def C(msg):
    i=1
    while(i<=5):
        print(msg)
        i=i+1
A('Thread:1')
B('Thread:2')
C('Thread:3')
'''
```

```
'''
import _thread
def A(msg):
   i=1
   while(i<=5):
      print(msg)
      i=i+1
def B(msg):
   i=1
   while(i<=5):
      print(msg)
      i=i+1
def C(msg):
   i=1
   while(i<=5):
      print(msg)
      i=i+1
_thread.start_new_thread(A,('Thread:1',))
_thread.start_new_thread(B,('Thread:2',))
_thread.start_new_thread(C,('Thread:3',))
'''
'''
import _thread
import time

def A(msg):
   i=1
   while(i<=5):
      time.sleep(2)
      print(msg)
      i=i+1
def B(msg):
   i=1
   while(i<=5):
      time.sleep(1)
      print(msg)
      i=i+1
def C(msg):
   i=1
   while(i<=5):
        time.sleep(3)
        print(msg)
        i=i+1
_thread.start_new_thread(A,('Thread:1',))
_thread.start_new_thread(B,('Thread:2',))
_thread.start_new_thread(C,('Thread:3',))
'''
'''
#2.multiple threading
-fun main=we can pass multiple arguments
-x=threading.Thread(target=function,args=(Arguments))
 x.start()
'''
'''
import time
def main(r,s,msg):
    for i in range(1,r):
        time.sleep(s)
        print(msg)
main(5,10,'Thread-1')
main(5,1,'Thread-2')
main(5,3,'Thread-3')
'''
'''
import threading
import time
def main(r,s,msg):
    for i in range(1,r):
        time.sleep(s)
        print(msg)
x=threading.Thread(target=main,args=(5,2,'Thread-1',))
x.start()
y=threading.Thread(target=main,args=(5,1,'Thread-2',))
y.start()
z=threading.Thread(target=main,args=(5,3,'Thread-3',))
z.start()
'''
```

Collections

-Collections are store multile elements like lists,sets,tuples,dictioniories

-python provides more data structures that can used as an alternative built in data types the module name is called as collections

4 types
#1.Counter
#2.namedtuple
#3.deque
#4.chainmap

#1.counter:
============
-it takes input as string,list,tuple
-it will count each and every value
-it makes desending order
'''
'''

```
from collections import Counter
li=[1,2,3,4,5,1,2,35,7,8,9,4,5,7,1,1,1,1]
res=Counter(li)
print(res)

x=(1,2,3,4,1,2,3,3,4,5,1,1,1,1,'hello','good')
res1=Counter(x)
print(res1)

y='hello good morning'
res2=Counter(y)
print(res2)
'''
'''
```

#2.namedtuple:
==============
-namedtuple takes unstructure tuples and dictionaries
-which makes better structure for the data
-it is used to accessing the data
'''
'''

```
from collections import namedtuple
employe={
        "name":'bhaskar',
        "email":'bhaksar@',
        "city":'bangalore',
        "mobile":927676,
        }
#from collections import namedtuple
x=namedtuple('employe',['name','email','city','mobile'])
y=x('bhskar','bahskar@','bangalore',9126172162)

#keys,index
print(y.name)
print(y[0])
'''
'''
#3.deque:
=========
-deque is choosen over the list when we need faster pop and append
-it is used to make faster program at a time
'''
'''
from collections import deque
li=[1,2,3,4,5,6,7]
x=deque(li)
x.pop()
x.popleft()
print(x)

from collections import deque
li=[1,2,3,4,5,6,7]
x=deque(li)
x.append('hello')
x.appendleft('hi')
print(x)
'''
'''
#4.chainmap:
-python has a container called chainmap.it combines many dictionaries
 into a single unit
'''
'''
from collections import ChainMap
d1={'a':'bhaksar','b':'chinni'}
```

```
d2={'c':'bangalore','d':'vizag'}
d3={'e':826515165,'f':2893623767}
res=ChainMap(d1,d2,d3)
print(res)
print(res.maps)
print(list(res.keys()))
print(list(res.values()))
'''''
'''
string methods:
===============
-it is a group of alphanumeric characters
-string should be declare '' or ""
-we cant able to do mathematical calculations

#indexing():we want particular values is called indexing
#positive indexing:consider left to right:starts from 0
#negative indexing:consider right to left:starts from -1
'''
'''
x='Besanttechonology'
print(x[4])

x='Besanttechnology'
print(x[-4])
'''
#slicing():range of value we can use slicing
#str(start:end:step)
'''
x='Besanttechnology'
print(x[0:5])

#x[o]+x[1]+x[2]+x[4]
'''
#split():it is used to convert str to list
'''
x='hello good morning'
y=x.split()
print(y)
'''
```

```
#join():it is used to convert list to str
'''
x='hello good morning'
y=x.split()
y.pop()
z=str.join(', ',y)
print(z)
'''
#upper():it is used to convert lower to upper
'''
x='hello good morning'
y=x.upper()
print(y)
'''
#lower():it is used to convert upper to lower
'''
x='HELLO GOOD MORNING'
y=x.lower()
print(y)
'''
#title():every word first letter upper
'''
x='hello good morning'
y=x.title()
print(y)
'''
#capitalize():first word ,first lettet upper
'''
x='hi good morning'
y=x.capitalize()
print(y)
'''
#swapcase():it is used to convert lower to upper and upper to lower
'''
x='Hello Good Morning'
y=x.swapcase()
print(y)
'''
#format()
#format_map()
```

```
#it is used to make default template with different values
#{}
'''
x="my name is {},i am from {}"
y=x.format('bhaksar','vizag')
y=x.format('chinni','vizag')
print(y)
'''
#format_map():it is used to make existing dict values
'''
x="my name is {name[0]},i am from {city[0]},state{state[2]}"
d={
   "name":['bhaskar','chinni','madhu'],
   "city":['vizag','chennai','bangalore'],
   "state":['AP','TN','KA'],
   }
y=x.format_map(d)
print(y)
'''
#index():consider left to right
#rindex():consider right to left
#it is used to find the position based on the value
'''
x='hello good morning'
y=x.index('o')
print(y)
'''
'''
x='good'
y=x.index('o')
print(y)
'''
#count():it is used to count the values how many times present in the string
'''
x='hello good morning'
y=x.count('o')
print(y)
'''
#center():it is used to make center of the string,fill reamaing index values some
elements
```

```
'''
x='hello good mornig'
y=x.center(8,'*')
print(y)
'''
#replace:it is used to replace string value
'''
x='hello good morning'
y=x.replace('hello','hi')
print(y)
'''
'''
#class
-group of functions/group of object is called as class
-inside class describe n number of functions

class class_name:
   group of functions

calling_object()

class        =>keyword
class_name   =>anything
calling_obj()=>class name
'''
'''
class Main:
   print('this is first class')
Main()
'''
'''
class Main:
   print('this is first class')
   def show(self):
      print('this is show fun')
obj=Main()
obj.show()
'''
'''
class Main:
    print('this is first class')
    def show(self):
        print('this is show fun')
    def display(self):
        print('this is display fun')
obj=Main()
obj.show()
obj.display()
'''
'''
class Main:
    print('this is first class')
    def show(self,a,b):
        print('this is show fun:',a+b)
    def display(self,x,y,z):
        print('this is display fun:',x+y+z)
obj=Main()
obj.show(1,2)
obj.display(1,2,3)
'''
'''
class Main:
    print('this is first class')
    def show(self,a,b):
        print('this is show fun:',a+b)
    def display(self,x,y,z):
        print('this is display fun:',x+y+z)
    def display(self,a,b):
        print('total:',a+b)
obj=Main()
obj.show(1,2)
obj.display(1,2,3)
'''
#class provide basically constructor function
#it is used to define instance of objcet/collection of objcets
#it will call automatically when we call main class
#_init_ =>constructor function
#it will always overloaded last one
#one class =only one constructor fun
'''
```

```
class Main:
    print('this is first class')
    def _init_(self):
        print('this is constructor fun')
    def show(self,a,b):
        print('this is show fun:',a+b)
    def display(self,x,y,z):
        print('this is display fun:',x+y+z)
obj=Main()
obj.show(1,2)
obj.display(1,2,3)
'''
'''
class Main:
    print('this is first class')
    def _init_(self):
        print('this is constructor fun')
    def _init_(self):
        print('this is constructor fun:1')
    def show(self,a,b):
        print('this is show fun:',a+b)
    def display(self,x,y,z):
        print('this is display fun:',x+y+z)
obj=Main()
obj.show(1,2)
obj.display(1,2,3)
'''
#Main class only we can pass arguments
#Main class autoamatically,this constuctor fun will be triggered
'''
class Main:
    print('this is first class')
    def _init_(self,name,email):
        print('this is constructor fun')
        print(name,email)
    def show(self,a,b):
        print('this is show fun:',a+b)
    def display(self,x,y,z):
        print('this is display fun:',x+y+z)
obj=Main('bahskar','bahskar@')
```

```
obj.show(1,2)
obj.display(1,2,3)
'''
'''
#In the present of constructor fun values name,email those information
 i just want to push into the show fun,display fun
#istance of object describe self only
#self.key=value
#self.a=name
#self.b=email
'''
'''
class Main:
   print('this is first class')
   def _init_(self,name,email):
      self.a=name
      self.b=email
   def show(self,a,b):
      print('this is show fun:',a+b)
      print('Name:',self.a)
      print('Email:',self.b)
   def display(self,x,y,z):
      print('this is display fun:',x+y+z)
obj=Main('bahskar','bahskar@')
obj.show(1,2)
obj.display(1,2,3)
'''
'''
class person:
   def _init_(self,name,age,gender):
      self.x=name
      self.y=age
      self.z=gender
   def data(self):
      print('Name:',self.x)
   def vote(self):
      if self.y<18:
         print('your not eligible for vote')
         print('Gender:',self.z)
      else:
            print('your eligible for vote')
            print('Gender:',self.z)
a=person('bhaskar',10,'Male')
a.data()
a.vote()
'''
'''
class employe:
    def _init_(self,id,name,city,mobile):
        self.a=id
        self.b=name
        self.c=city
        self.d=mobile
    def display(self,state):
        print('Id:',self.a)
        print('Name:',self.b)
        print('City:',self.c)
        print('Mobile:',self.d)
        print('State:',state)
x=employe(1,'chinni','vizag',98162161,9783373673)
x.display('Ap')
'''
'''
```

Inheritance:
-used to one class derived into another class
callback:one function passed into another function as an argument
callback:only functions
inheritance:only class

drawbacks:
-parent class derived inside child class
-child class not access parent class

4 types
#1.single inheritance:derived inside child class
class A ->class B(A) only one time

#2.multilevel inheritance:
class A ->class B(A) ->class C(B) ->class D(C)...
chain

```
#3.multiple inheritance:
class A ->class B ->class C ->class D(A,B,C)
multiple parents and one child
#4.herachical inheritance:
class A =>class B(A) +class C(A)+class D(A)
one parent and multiple inheritance
'''
#single inheritance:
'''
class A:
   def show(self):
      print('this is class A')
class B(A): #with the help of B access A
   def show1(self):
      print('this is class B')
obj=B()
obj.show()
obj.show1()
'''
#multilevel inheritance:
'''
class A:
   def show(self):
      print('this is class A')
class B(A): #with the help of B access A
   def show1(self):
      print('this is class B')
class C(B): #with the help of C access A,B
   def show2(self):
      print('this is class C')
class D(C): #with the help of D access A,B,C
   def show3(self):
      print('this is class D')
obj=D()
obj.show()
obj.show1()
obj.show2()
obj.show3()
'''
```

```
#multiple inheritance:
'''
class A:
    def show(self):
        print('this is class A')
class B:
    def show1(self):
        print('this is class B')
class C:
    def show2(self):
        print('this is class C')
class D(A,B,C): #with the help of D access A,B,C
    def show3(self):
        print('this is class D')
obj=D()
obj.show()
obj.show1()
obj.show2()
obj.show3()
'''
#herachical inheritance:
'''
class A:
    def show(self):
        print('this is calss A')
class B(A): #B is a objcet access A
    def show1(self):
        print('this is class B')
class C(A): #with the help of C access A
    def show2(self):
        print('this is calss C')
class D(A): #with the help of D access A
    def show3(self):
        print('this is class D')
obj=D()
obj.show()
obj.show3()
'''
#super() ->it is used to call the parameters
'''
```

```
class student:
    def _init_(self,name,marks):
        self.a=name
        self.b=marks
    def display(self):
        print('Name:',self.a)
        print('Marks:',self.b)
#obj=student('bhsakar',50)
#obj.display()
class child(student):
    def _init_(self,name,marks,mobile):
        super()._init_(name,marks)
        self.c=mobile
    def contact(self):
        print('Mobile:',self.c)
obj=child('bhaskar',50,8217261726)
obj.display()
obj.contact()
'''
'''
polymorphism:
polomorphism contains two "poly" and "morphs" .poly means many and
morphs means shape.
-it is used one task performs multiple types
'''
#method overriding:
#different class +same method +different parameters
'''
class parent:
    def show(self,a,b):
        print('total:',a+b)
class child(parent):
    def show(self,x,y,z):
        super().show(x,y) #x=a,y=b
        print('total:',x+y+z)
obj=child()
obj.show(1,2,3)
'''
#method overload:
#same class +same method+different parameters
```

```
'''
class parent:
   def show(self,a,b):
      print('total:',a+b)
   def show(self,x,y,z):
      print('total:',x+y+z)
obj=parent()
obj.show(1,2,3)
'''
#we have same methods it always overloaded last one
#that we can over come by using overriding mething
'''
Encapsulation:
-accesaible from anywhere,both the inside and outside the class
encapuslation is the process of hiding the internal data of an object
'''
'''
class public:
   def _init_(self):
      self.x="Bhaskar"
   def display(self):
      print(self.x)
obj=public()
obj.display()
print(obj.x)
'''
'''
Decorators:functions,used to add some more functionalities without
           distrubing existing functions
whatsapp -without distrubing existing structure there we need to push
          some more features is called as decorators
Decorator -@

#decorators
#nested functions
#callback
#parameters & arguments
'''
'''
def main():
    return 'hello good morning'
print(main())
'''
#upper():used to convert lower to upper
#split():it is used to convert str to list
'''
def dec(x): #call back
   def inner(): #access the parameters
      return x().upper() #callback for main function
   return inner
@dec
def main():
   return 'hello good morning'
print(main())
'''
'''
def dec(x):
   def inner():
      return x().upper()
   return inner
def dec1(y):
   def inner():
      return y().split()
   return inner
@dec1
@dec
def main():
   return 'hello good morning'
print(main())
'''
'''
Data Abstraction:
-internal data is hiding,show only functionalities
-if any one application back end we cant able to see that is called as
 data abstraction
-ATM:able to see screen,get cash and all
    we cant able to see data,process,code how it will work and all
'''
'''
class person:
   def _init_(self,name,age,gender):
      self.x=name
      self.y=age
      self.z=gender
   def data(self):
      print('Name:',self.x)
   def vote(self):
      if self.y<18:
          print('your not eligible for vote...')
          print('Gender:',self.z)
      else:
          print('your eligible for vote....')
          print('Gender:',self.z)
'''
'''
from chinni import person
obj=person('bhaskar',10,'Male')
obj.data()
obj.vote()
'''
'''
```

Regular expression =Regex =re
-It is used to make pattern matching

operators:

```
#Quantity operator:+,*,?
    +:it will take min 1 -max infinite
      ex:aaaaaaaaaaaaaaaa
    *:it will take min 0 -max infinite
      ex:aaaaaaaaaaaaaaaaa
    ?:it will take min 0 -max 1
     ex:a
#Group operators:[]
            :[0-9a-zA-Z]
            :[0-9a-zA-Z]+
            :defaulty quantity ?
#digits:\d
      :\d+
      :[0-9]+
      :it consider only numbers
```

```
#words:\w
     :[0-9a-zA-Z]+
#rangeoperator:{start,end}
            :a{5,10}
            :min 5,max 10
#escape operator:\
              :it consider special characters
              :@,#,$,%,&

by using these operators we can create one strong pattern

import re
re.compile('pattern')

import re
pattern=re.compile('[a-z]+')
user='12345'

i want to check pattern and user matching or not
#search():it takes only one value
#findall():it search all the values
         returns as a list
'''
'''
import re
pattern=re.compile('[a-z]+')
user='12345'
res=pattern.search(user)
print(res)
'''
'''
import re
pattern=re.compile('[a-z]+')
user='Hello all good morning'
res=pattern.search(user)
print(res)
'''
'''
import re
pattern=re.compile('[a-z]+')
```

```
user='Hello all good morning'
res=pattern.findall(user)
print(res)
'''
'''
import re
pattern=re.compile('[a-zA-Z]+')
user='Hello all good morning'
res=pattern.findall(user)
print(res)
'''
'''
import re
pattern=re.compile('\d+')
user='Hello 27867 good 83626 morning 3262'
res=pattern.findall(user)
print(res)
'''
'''
import re
pattern=re.compile('\w+')
user='Hello 25425 good 377653 morning 263 all'
res=pattern.findall(user)
print(res)
'''
'''
import re
pattern=re.compile('\&+[A-Za-z]+\@+[0-9]+\#+[0-9]+')
user='&Bhaskar@7582#632832'
res=pattern.findall(user)
print(res)
'''
#user='#Bhaskar%6578&@86162Chinni$5765765'

'''
```

Tkinter
-Tkinter also called as library,framework/modules
-just import it and we can use it
-Widgets:we can create user interface
* -we can pass n number of functions

```
1.import tkinter module
2.create root window
3.add widgets
4.start main root
'''
'''
from tkinter import *
from PIL import Image, ImageTk
#pip install pillow

root=Tk() #tool kit
root.title('Whatsapp App')
ico = Image.open('whatsapp.jpg')
photo = ImageTk.PhotoImage(ico)
root.wm_iconphoto(False, photo)


def red_btn():
   print('i am red...')
def yellow_btn():
   print('i am yellow...')
def blue_btn():
   print('i am blue...')
def lightgreen_btn():
   print('i am lightgreen...')


Button(root,text='submit',command=red_btn,activebackground='blue',activefo
reground='red',bg='red',fg='white',font=('times',20,'bold'),width=15,height=4).
grid(row=0,column=0)
Button(root,text='Register',command=yellow_btn,activebackground='blue',acti
veforeground='red',bg='yellow',fg='white',font=('times',20,'bold'),width=15,hei
ght=4).grid(row=0,column=1)
Button(root,text='Login',command=blue_btn,activebackground='blue',activefo
reground='red',bg='blue',fg='white',font=('times',20,'bold'),width=15,height=4)
.grid(row=1,column=0)
Button(root,text='Clear',command=lightgreen_btn,activebackground='blue',act
iveforeground='red',bg='lightgreen',fg='white',font=('times',20,'bold'),width=1
5,height=4).grid(row=1,column=1)
```

```
Label(root,text='welcome besant
technologies',font=('times',20,'bold')).grid(row=2,column=0)



root.mainloop()

'''
#Checkbutton:

from tkinter import *

root=Tk()
Checkbutton(root,text='Gender').grid(row=0)
Checkbutton(root,text='Male').grid(row=1)
Checkbutton(root,text='Female').grid(row=2)

root.mainloop()
'''
#Radiobutton:
'''
from tkinter import *

root=Tk()
Radiobutton(root,text='Gender').grid(row=0)
Radiobutton(root,text='Male').grid(row=1)
Radiobutton(root,text='Female').grid(row=2)

root.mainloop()
'''
#Listbox:it offers list to the user which user can accept any number of options
'''
from tkinter import *

root=Tk()
Lb=Listbox(root)
Lb.insert(1,'Python')
Lb.insert(2,'Java')
Lb.insert(3,'Html')
Lb.insert(4,'Css')
Lb.insert(5,'Javascript')
Lb.pack()

root.mainloop()
'''
#Scale:it is used to provide graphical slides,it allows to select values from the
  #scale
'''
from tkinter import *

root=Tk()
w=Scale(root,from_=0,to=46)
w.grid()
w=Scale(root,from_=0,to=200,orient=HORIZONTAL)
w.grid()

root.mainloop()
'''
#spinbox:it is an enrty of 'ENTRY' widget.values can allow selecting fixed
  #values for that numbers
'''
from tkinter import *

root=Tk()
w=Spinbox(root,from_=-10,to=10)
w.grid()

root.mainloop()
'''
#Messagebox:it is used to create one msgbox
'''
import tkinter
from tkinter import messagebox
root=tkinter.Tk()
root.geometry('150x150')
messagebox.showinfo('information','information for user')

root.mainloop()
'''
```

```
#Checkbutton:

from tkinter import *

root=Tk()
Checkbutton(root,text='Gender').grid(row=0)
Checkbutton(root,text='Male').grid(row=1)
Checkbutton(root,text='Female').grid(row=2)

root.mainloop()
'''
#Radiobutton:
'''
from tkinter import *

root=Tk()
Radiobutton(root,text='Gender').grid(row=0)
Radiobutton(root,text='Male').grid(row=1)
Radiobutton(root,text='Female').grid(row=2)

root.mainloop()
'''
#Listbox:it offers list to the user which user can accept any number of options
'''
from tkinter import *

root=Tk()
Lb=Listbox(root)
Lb.insert(1,'Python')
Lb.insert(2,'Java')
Lb.insert(3,'Html')
Lb.insert(4,'Css')
Lb.insert(5,'Javascript')
Lb.pack()

root.mainloop()
'''
#Scale:it is used to provide graphical slides,it allows to select values from the
  #scale
'''
```

```
from tkinter import *

root=Tk()
w=Scale(root,from_=0,to=46)
w.grid()
w=Scale(root,from_=0,to=200,orient=HORIZONTAL)
w.grid()

root.mainloop()
'''
#spinbox:it is an enrty of 'ENTRY' widget.values can allow selecting fixed
  #values for that numbers
'''
from tkinter import *

root=Tk()
w=Spinbox(root,from_=-10,to=10)
w.grid()

root.mainloop()
'''
#Messagebox:it is used to create one msgbox
'''
import tkinter
from tkinter import messagebox
root=tkinter.Tk()
root.geometry('150x150')
messagebox.showinfo('information','information for user')

root.mainloop()
'''#calculator app
from tkinter import *


root=Tk()
root.title('Calculator App')


#def btn_click(value):
#    print(value)
```

```
def btn_click(value):
   global data
   data=data+str(value)
   input_text.set(data)

def btn_equal():
   global data
   result=str(eval(data))
   input_text.set(result)

def btn_clear():
   global data
   data=" "
   input_text.set(' ')


data=" "
input_text=StringVar()


input_frame=Frame(root,height=20,width=300,highlightbackground='black',hi
ghlightthickness=1)
input_frame.pack(side=TOP)

input_field=Entry(input_frame,width=22,textvariable=input_text,bg='#eee',jus
tify=RIGHT,font=('times',20,'bold'))
input_field.grid(row=0,column=0)
input_field.pack(ipady=10)

btn_frame=Frame(root,height=220,width=300,bg='lightgreen')
btn_frame.pack()

#first row

clear=Button(btn_frame,text='C',command=lambda:btn_clear(),bd=0,width=32
,height=3).grid(row=0,column=0,columnspan=3,padx=1,pady=1)
devide=Button(btn_frame,text='/',command=lambda:btn_click('/'),bd=0,width
=10,height=3).grid(row=0,column=3,padx=1,pady=1)
```

```
#second row

seven=Button(btn_frame,text=7,command=lambda:btn_click(7),bd=0,width=1
0,height=3).grid(row=1,column=0,padx=1,pady=1)
eight=Button(btn_frame,text=8,command=lambda:btn_click(8),bd=0,width=10
,height=3).grid(row=1,column=1,padx=1,pady=1)
nine=Button(btn_frame,text=9,command=lambda:btn_click(9),bd=0,width=10,
height=3).grid(row=1,column=2,padx=1,pady=1)
multiply=Button(btn_frame,command=lambda:btn_click(''),text='',width=10,he
ight=3).grid(row=1,column=3,padx=1,pady=1)

#third row

six=Button(btn_frame,command=lambda:btn_click(6),text=6,bd=0,width=10,h
eight=3).grid(row=2,column=0,padx=1,pady=1)
five=Button(btn_frame,command=lambda:btn_click(5),text=5,bd=0,width=10,
height=3).grid(row=2,column=1,padx=1,pady=1)
four=Button(btn_frame,command=lambda:btn_click(4),text=4,bd=0,width=10,
height=3).grid(row=2,column=2,padx=1,pady=1)
plus=Button(btn_frame,command=lambda:btn_click('+'),text='+',bd=0,width=1
0,height=3).grid(row=2,column=3,padx=1,pady=1)

#fourth row

three=Button(btn_frame,command=lambda:btn_click(3),text=3,bd=0,width=1
0,height=3).grid(row=3,column=0,padx=1,pady=1)
two=Button(btn_frame,command=lambda:btn_click(2),text=2,bd=0,width=10,
height=3).grid(row=3,column=1,padx=1,pady=1)
one=Button(btn_frame,command=lambda:btn_click(1),text=1,bd=0,width=10,
height=3).grid(row=3,column=2,padx=1,pady=1)
minus=Button(btn_frame,command=lambda:btn_click('-'),text='-
',bd=0,width=10,height=3).grid(row=3,column=3,padx=1,pady=1)

#fifth row

zero=Button(btn_frame,command=lambda:btn_click(0),text=0,bd=0,width=22,
height=3).grid(row=4,columnspan=2,padx=1,pady=1)
dot=Button(btn_frame,text='.',command=lambda:btn_click('.'),bd=0,width=10,
height=3).grid(row=4,column=2,padx=1,pady=1)
```

```
equal=Button(btn_frame,text='=',command=lambda:btn_equal(),bd=0,width=1
0,height=3).grid(row=4,column=3,padx=1,pady=1)



root.mainloop()'''

'''
API handling:
-application programming interface
-api can devide static and dynamic
-if you want to update,modify the content server only we can change,not in
  web site.same changes reflecting web page aslo that is called as dynamic
-they dont have any server,api normal data,normal content that is called as
 static
-dynamic only we can use api
-it is used to connect two servers

implementation:
requests:fetch the data from the existing server
        :it is used to connecting api data
        :pip install requests
        :import requests
        :requests.get(API_URL)

JSON():javascript object notation
      :it is used to interchange the data
      :converts json string to json object
      :in application default format is json

{"name":'bhaskar',"email":'bhaskar@',"Mobile":783676376} ->json string
{name:'bhaskar',email:'bhaskar@',mobile:2763782638726} ->json object

json string -we cant able to access
json object -we can able to access easily

json place holder

jsonplaceholder.typicode.com
```

```
#i just want to get fetch over the data in python

API_URL="https://jsonplaceholder.typicode.com/posts"
'''
'''
import requests
import json
def get_data():
    API_URL="https://jsonplaceholder.typicode.com/posts"
    resp=requests.get(API_URL)
    #return resp
    if resp.status_code == 200:
        data=resp.json()
        return data
    else:
        print('error',resp.status_code)


post=get_data()
if post:
 for i in range(0,100):
    print("user ID:",post[i]['userId'])
    print("ID:",post[i]['id'])
    print("Title:",post[i]['title'])
    print("Body:",post[i]['body'])
    print("====================")
else:
    print('failed to fetch the data')
'''


'''
weather application project
-i just want to get current temperature,current pressure,current
humidity,current wind
'''
from tkinter import *
import requests
import json
```

```
root=Tk()
root.title('Weather app')

def get_data(root):
    city=user.get()
    print(city)

api="https://api.openweathermap.org/data/2.5/weather?q="+city+"&appid=0
03da33b59d01689a7e171acf4d7976c"
    resp=requests.get(api)
     #print(resp)
    if resp.status_code == 200:
        data=resp.json()
        temp=int(data['main']['temp']-273.15)
        #print(temp)
        pressure=int(data['main']['pressure'])
        hum=int(data['main']['humidity'])
        condition=data['weather'][0]['main']
        wind=int(data['wind']['speed'])
         #print(temp,pressure,hum,wind,condition)

output="temperature:"+str(temp)+"°C"+"\n"+"pressure:"+str(pressure)+"hpa"
+"\n"+"humidity:"+str(hum)+"%"+"\n"+"wind:"+str(wind)+"m/s"+"\n"+"conditi
on:"+condition
        print(output)
        label.configure(text=output)


user=Entry(root,width=30,justify='center',font=('poppins',20,'bold'))
user.grid(row=0,column=0)
user.bind("<Return>",get_data)

label=Label(root,font=('poppins',20,'bold'))
label.grid(row=1,column=0)


root.mainloop()
```