```
#Python Notes prepared by Ajinkya H.Ban
# Topics covered from String to Inheritance

*String in python*
---------------------------
Q. What is mean string?
-----------------------
Def : String is the collections of characters.
    OR
    The number of characters comes together and create one meaningful word that is called string.
Example:
--------
pen, laptop

Q. how we can declare the string?
---------------------------------------
- In python there is no concept of char data type.
- So, we can declare the string in python by using double quote as well as single quote.

Example:
--------------
str_data = "python"
print(type(str_data))

- String is the immutable type of object means we can not add new element into string, we can not modified the
  existing string and finally we can't remove the element from string.

- String indexing starts from 0.

*Update the string*
--------------------------
- We can not update the existing string in python when we try to do that the python compiler generate the follwoing
error message,
TypeError: 'str' object does not support item assignment

Example:
-----------
str_data = "python"
#suppose i want to update the existing string
str_data[0] = 'P'
print(str_data)

*delete the string*
-------------------------
Suppose i want to delete the specific character from the string using python then python not allowed to delete
specific characrter from string.It will display the following error,

 Example:
 ---------------
str_data = "python"

#suppose i want to delete string.
del str_data[0]
print(str_data)

TypeError: 'str' object doesn't support item deletion

*We can also apply slicing in string*
--------------------------------------------
We can get some portion of the string by using the concept called as slicing. The slice operator can be declared
using square brackets,
Example :

str_data[start:end:step] --> step is the optional. by default the value of step is 1.

Example:
------------------
str_data = "The python is easy to understand"
print(str_data[0:2:1])

*We can also use the membership operator*
---------------------------------------------
In python there are two types of membership operators are present,
a) in
b) not in
Basically these operator can be used for to check a perticular element present in collections or not.

Example:
-------------
str_data = "The python is easy to understand"
print("is" in str_data)

*We can also combine more then two string*
---------------------------------------------
We can use the plus operator to combine more then two string. This is called string concat.

Example:
-----------
str1 = "python"
str2 = " developer"
print(str1 + str2)

*Inbuilt methods of string class*
-------------------------------------
1. capitalize(): this method convert the first character from first word into uppercase letter.
Example:
--------------
str1 = "python"
str2 = "python is easy to use"
print(str1.capitalize())

2. center() :
---------------
This method print the characters from left side as well as right side of string. This method required two parameter
that is
a) width
```

```
b) fillchar --> it is optional ---> you use it as empty

Example:
---------
str1 = "python"
str2 = "python is easy to use"
print(str1.center(10,' '))

OR
str1 = "python"
str2 = "python is easy to use"
print(str1.center(10,'*'))

3.title:
-------------
This method convert first letter of each word into uppercase.
Example:
------------
str1 = "python"
str2 = "python is easy to use"
print(str2.title())

4. upper():
-------------
This method convert all lower case words into uppercase.
Example:
-------------
str1 = "python"
str2 = "python is easy to use"
str3 = "PYTHON IS EASY TO USE"
print(str2.upper())

5. lower():
------------------
This method convert all uppercase letters into lower case letters.
Example:
str1 = "python"
str2 = "python is easy to use"
str3 = "PYTHON IS EASY TO USE"
print(str3.lower())

6. startswith():
----------------------
This method check if the string starts with specific character or not. If the string start with specific character then
it will return True otherwise it will return False.
Example:
--------
str1 = "python"
str2 = " python is easy to use."
str3 = "PYTHON IS EASY TO USE"
print(str2.startswith(' p'))

7. endswith():
-----------------------
This method check if the string ends with specific character or not. If the string ends with specific character then
it will return True otherwise it will return False.
Example:
--------
str1 = "python"
str2 = "python is easy to use"
str3 = "PYTHON IS EASY TO USE"

print(str2.endswith('use'))

8. swapcase():
-------------------------
This method convert uppercase letter into small case letter and small case letter into upper case letter.
Example:
-------------
str4 = "pythON"
print(str4.swapcase())

9. count():
------------------
This method can be used for to count how many time a particular letter or word present in the whole string. If the string
not found then it will return 0. If the string found then it will return count number.
Example:
---------------
str2 = "python is easy to use and is easy to learn"
print(str2.count("is"))


10. isupper():
--------------------
suppose i want to check the string contains upper case letters or not. If all the words are in upper case then it will
return true otherwise it return false.
Example:
--------------
str3 = "PYTHON IS EASY To USE"
print(str3.isupper())

11. islower():
--------------------
suppose i want to check the string contains lower case letters or not. If all the words are in lower case then it will
return true otherwise it return false.

Example:
--------------
str2 = "python is easy to use and is easy to learn"
print(str2.islower())

12. isdigit():
------------------
suppose i want to check if the string contains digit or not. If it contains only digit then it will return true otherwise
it will return false value.
Example:
```

```
-----------
str5 = "1234"
print(str5.isdigit())

13. isdecimal():
------------------
The decimal numbers are in between 0 to 9. Suppose i want to check if the string contains number between 0 to 9 then it will
return true otherwise it will return false.

Example:
--------------
str5 = "89777"
print(str5.isdecimal())

14. isalpha():
----------------
The alphabets that is from a to z, if the string contains the alphabets that are in between a to z then it will return
true otherwise it will return false.

Example:
------------------
str5 = "python"
print(str5.isalpha())

15. replace():
------------------------
Suppose i want to replace the old word with new word then i can use the method replace. The replace method accept two
parameter
1. old char
2. new char

Example:
---------------
str2 = "python is easy to use and is easy to learn"
print(str2.replace("python","java"))

16. istitle():
---------------------
This method can be used for to check the first letter of each word in capital letter. If it is found then it will return
true otherwise it will return false value.

Example:
-----------------
str2 = "Python Is Easy To Use And Is Easy To Learn"
print(str2.istitle())

17. find():
---------------------
The find method returns the index number of word that you find. It will find the first occurence of word then returns the
index number. If the word not found then it will return -1.

Example:
-------------
str2 = "Python Is Easy To Use And Is Easy To Learn"
print(str2.find("Easy"))

18. index:
----------------
The index method returns the index number of word that you find. It will find the first occurence of word then returns the
index number. If the word not found then it will return valueError.

Example:
--------------
str2 = "Python Is Easy To Use And Is Easy To Learn"
print(str2.index("Is"))

19.lstrip(): This method remove the extra white space(blank space) which are present at left side of the string.
Example:
--------------
str5 = "    python"
print(str5.lstrip())

20. rstrip() :This method remove the extra white space(blank space) which are present at right side of the string.

Example:
----------------
str5 = "python     "
print("before calling rstrip function",len(str5))
res=str5.rstrip()
print("After calling rstrip() the length is = ",len(res))

21. strip() : This method remove the blank spaces which are present at both side of the string.

Example:
--------------------
str5 = "      python     "
print(str5.strip())

22. split(): This method divide the string into list with comma separated values.

Example:
------------------
str2 = "Python Is Easy To Use And Is Easy To Learn"
res = str2.split()
res[0] ="python"
print(res)
==============================================================================================================


Python List
-------------------------
Q. What is List?
--------------
Def : List can be used to store the different type of data element into a single variable.

Q. How we can declare the list?
```

```
------------------------------
- In python we can declare the list using the square bracket [].
- In python the list is mutable type of element.
- So, we can add the new element into list, update the existing element from list and delete the elements from the list.
Example:
--------------
lst = [1,2,5,6,3,4]
for i in lst:
     print(i)


*Basic operations on the list*
-------------------------------------
1. concat: We combine two or more list into single list.

Example:
-----------------
lst1 = [1,2,3]
lst2 = [4,5,6]
print(lst1 + lst2)


2. Repeatation operator
----------------------------
How many times the list will printed. This operator indicated by using *.

Example:
---------------
lst1 = [1,2,3]
lst2 = [4,5,6]
print(lst1 * 2)

3. slice operator:
---------------------
If i want to print the data in between the range.Slice operator. The slice operator can be indicted by [start:end:step].

Example:
----------------
lst = [1,2,5,6,3,4]
print(lst[0:4])

or negative indexing
-----
print(lst[-5:-1])

4. membership operator:
---------------------------
1. in
2. not in

Q. how you can say that, list is mutable?
-----------------------------------------------
We can say that the list is mutable because we can easily add new element into list, we can easily update the existing
data from list and we delete the data also.

Example:
-------------
lst1 = [1,2,3]
lst1.append(100)
lst1.append(200)
print(lst1) # this method add the new element into list which has the last location.

# i want to remove the element
del lst1[0]
print(lst1)

#updation
lst1[1] = 300
print(lst1)


Q. how you can take userinput into list?
----------------------------------------------
Example:
--------------
#empty list

lst = []
how_many_element = int(input("Enter how many element you want to add = "))
for i in range(0,how_many_element):
    lst.append(int(input("Enter the elements = ")))


print("The list elements are")
for j in lst:
    print(j)

Q. What are the different methods present in List?
-----------------------------------------------------
1. append(): Used for adding elements to the end of List.

Example:
-----------
lst = [4,7,8,94,6]
lst.append(100)
lst.append(200)
print(lst)

2. copy(): This method copy the element from one list to another list.
Example:
-----------
lst1 = [4,7,8,94,6]
lst2 = [8,2,4,1,9]
lst3 = []
lst3 = lst1.copy()
```

```
3. extend(): Add the elements of a list to the end of the current list.

Example:
----------------
lst1 = [4,7,8,94,6]
lst2 = [8,2,4,1,9]

lst2.extend(lst1)
print(lst2)


4. insert(): Adds the element at the specified location.
Example:
-------------
lst1 = [4,7,8,94,6]
lst2 = [8,2,4,1,9]

lst2.insert(2, 100)
print(lst2)

5. pop(): Remove the element which are present at the end of the list.
 Example:
 --------------
lst1 = [4,7,8,94,6]
lst2 = [8,2,4,1,9]

lst2.pop()
lst2.pop()
print(lst2)

6. pop(index): This method remove element which are present at specified locatiion
Example:
---------------
lst1 = [4,7,8,94,6]
lst2 = [8,2,4,1,9]

lst2.pop(0)
print(lst2)

7. remove():Removes the first item with specified value.

Example
--------------------------
lst1 = [4,7,8,94,6]
lst2 = [8,2,4,1,9]

lst2.remove(2)
print(lst2)

8. reverse(): It print the element in reverse order.
Example:
-------------
lst1 = [4,7,8,94,6]
lst2 = [8,2,4,1,9]

lst1.reverse()
print(lst1)

9. sort(): Sort the element in asc or desc order. By default it will sort in asc order.
Example:
--------
lst1 = [4,7,8,94,6,2,1]
lst1.sort(reverse=True)

print(lst1)

10. clear():
-----------------------------------------------------------------------------------------------------------------------
*Python tuples*
-------------------
- Python tuple is same as List.
- Python tuple is an immutable collection of different data type elements.
- Tuple are used to keep a list of immutable python objects.
- Once we declare the tuple we can not modifiy the element within the tuple.

*Creating the tuple*
-----------------------
- a tuple is formed by enclosing all the items in parentheses (), and each element is seperted by commas.
- A tuple can include elements with different data types.

Example:
--------------
my_tuple = (1,44,78,45,"python")
print(my_tuple)

print("-----------------------------------------------------------------")
for i in my_tuple:
    print(i)

OR
my_tuple = 1,45,88,56
print(my_tuple)
print(type(my_tuple))

*Access the tuple elements*
----------------------------
- Each item in a tuple has its own index value which start from zero.
- Index values are incremented by one.
- We can use the different methods to access the element from tuple.

1. Indexing Method:
-----------------------
- To retrive an item in a tuple we can use the index operator [], where the index start from 0.
- Attempting to access an index that is not inside the tuple index will result in an IndexError.

Example:
```

```
--------------
my_tuple = 1,45,88,56
print(my_tuple[2])

OR
my_tuple = 1,45,88,56
print(my_tuple[5])

o/p:
-------
IndexError: tuple index out of range

2. Negative Indexing
-------------------------
Python support negative indexing also.The value of -1 denotes the last item, the index of -2 the second last element.

Example:
-----------
my_tuple = 1,45,88,56
print(my_tuple[-1]) # 56
print(my_tuple[-2]) # 88

3. Slicing:
---------------
If we wish to obtain a range of items from our tuple, then we use the concept of slicing.

Example:
------------
my_tuple = 1,45,88,56
print(my_tuple[1:3])

OR
my_tuple = 1,45,88,56
print(my_tuple[-1:-3:-1])

4. by using for loop:
--------------------------
my_tuple = 1,45,88,56
for i in my_tuple:
    print(i)

*Changing the element*
--------------------------
Tuples, unlike list, are immutable. This means that once we declare the tuple we can not modify the element from the tuple.
If you attempt to alter a tuple element you will receive a TypeError.

Example:
-------------
my_tuple = 1,45,88,56
my_tuple[1] = 100 #here we can alter the value.
print(my_tuple)

o/p
-------
TypeError: 'tuple' object does not support item assignment

*Deleteing tuple*
-------------------
We cannot delete the elements from tuple because tuple is immutable type of object. Suppose we can try to delete the
element from tuple it will display error message.

Example:
------------
my_tuple = 1,45,88,56
del my_tuple[1]
print(my_tuple)

o/p:
----
TypeError: 'tuple' object doesn't support item deletion

- You can delete the tuple but you cant delete the element which are present in tuple.
Example:
------
my_tuple = 1,45,88,56
del my_tuple
print(my_tuple)

o/p:
------
NameError: name 'my_tuple' is not defined

*Built in Python tuple methods*
-----------------------------------
1. len():  this method returns the total number of element present in tuple.
Example:
------------
my_tuple = 1,45,88,56
length = len(my_tuple)
print(length)

2. max(): This method returns the max number which are present in tuple.
Example:
-------
my_tuple = 1,45,88,56
print(max(my_tuple))

3. min(): This method returns the minimum element which are present in tuple.
Example:
------------
my_tuple = 1,45,88,56
print(min(my_tuple))

4. tuple(): This method can be used for to convert the string into tuple as well as list into tuple.
Example:
-----------
```

```
lst = [1,2,3,4]
print(type(lst))
print("After convert into tuple")
my_tuple = tuple(lst)
print(type(my_tuple))

5. membership operator(in, not in): Using the in and not in keyword we can determine wheather or not an item
   exists in tuple.
Example:
---------
my_tuple = (2,4,5,67,7)
print(5 in my_tuple)
====================================================================================================

*Dictionary in python*
-----------------------------
- Dictionaries are useful data structure for storing the data in python becuase in dictionary we can store the data in the form
  of keys and value pair.
- A python dictionary is a data structure that allows us to easily write very efficient code.
- The data is stored as key value pair using python dictionary.
- The dictionary is mutable type.
- In the dictionary the key must be unique. If the key is duplicated then dictionary will print the latest key value.
- Values can be of type, integer,list and tuple.
- key can be of type string or integer. Key can not be of type float.

*Creating the dictionary*
------------------------------
- The dictionary can be created by using the {} curly bracket.
- The key and the value can be separeted by using colon(:).
Example
-------------
dct = {"name":"Ajay","age":22,"mob":9822334455}
print(dct)

- *If you want to print only keys which are present in dictionary then you can use the following code*
Example:
------------
dct = {"name":"Ajay","age":22,"mob":9822334455,"city":["Pune","Mumbai","Nashik"]}
print(dct.keys())

- *We also pass the list to keys, the following code shows the use of list in dictionary*
Example
-----------------
dct = {"name":"Ajay","age":22,"mob":9822334455,"city":["Pune","Mumbai","Nashik"]}

In above code the city key contains the value in the form of list.
Example:
----------
dct = {"name":"Ajay","age":22,"mob":9822334455,"city":["Pune","Mumbai","Nashik"]}
print(dct['name'])
print(dct['mob'])
print(dct['city'][0])

- In above code we will print the name as Ajay and mob number of ajay by using the keys 'name' and 'mob'.
- We also print the city of ajay.
- print(dct['city'][0]) this code says that we print only one city which is present at the index 0 that is Pune.
- If we want to print two city that is Pune and Mumbai then we can change the statement like,print(dct['city'][0:2]).


*Suppose we want to print only values which are present in dictionary. Then we can use the pre defined function called values()*
Example:
---------
dct = {"name":"Ajay","age":22,"mob":9822334455,"city":["Pune","Mumbai","Nashik"]}
print(dct.values())

*Suppose i want to print both keys and values at a same time, then i can use the predeifined function called items()*
Example:
-------------
dct = {"name":"Ajay","age":22,"mob":9822334455,"city":["Pune","Mumbai","Nashik"]}
for k,v in dct.items():
    print(k,"------>",v)

-------------------------------------------------------------------------------------------------
*Python Set*
---------------
*Q.What is mean by set?*
- A python set is the collection of unordered elements.
- Each element in the set must be unique, immutable.
- Sets are mutable which means we can modify it after its creation.
- We can not directly access any element of the set by the index. We can get the list of elements by looping through the set.

*How we can create the set?*
- The set can be created by enclosing the comma separated immutable elements with the curely bracket.
- Python also provides the set() method which can be used to create the set.

*Example*
-----------------
days = {"monday","tuesday","wend"}
print(days)

#we can also print the element of the set by using loop
for i in days:
    print(i)

#We can also create the set by using inbuilt function set().
days = set(["mon","tue","wensd"])
for i in days:
    print(i)

#set can accept only immutable elements, if we try to insert mutable element into set then it will through error.
numbers = {1, 2, 3, 1, 3, "python", ["javascript",44,66]}
for i in numbers:
    print(i)

*Q. How we can add the element into set?*
----------------------------------------------
```

```
- Python provides the add() method and update() method which can be used to add some particular element to the set.
- The add() method is used to add single element into the set whereas the update() method used to add multiple element
  into the set.

*Example*
------------------
# by using add() method
months = set(["jan","feb","march"])
months.add("apr")
months.add("may")
for i in months:
    print(i)

#by using update() method
months = set(["jan","feb","march"])
months.update(["apr","may","june"])
for i in months:
    print(i)

*How we can remove the element from set?*
-----------------------------------------------
- Python provides the discard() method and remove() method which can be used to to remove the element from the set.
- The difference between these functions is that, using discard() method if the element does not exist in the set then
  the set remain unchanged whereas remove() will through the error.

*Example*
----------------
#by using discard()
months = set(["jan","feb","march"])
months.discard("may")
for i in months:
    print(i)

#by using remove()
months = set(["jan","feb","march"])
months.remove("may")
for i in months:
    print(i)

*Q. What is the difference between discard() and remove() method?*
*Q. What are the different operations perform on set?*
---------------------------------------------------------
Set can be performed mathematical operation such as union, intersection, difference and symmetric difference.

*1. Union of two Sets:*
-----------------------------
- To combine two or more sets into one set in python is called as Union of two sets.
- Python, use the union() function.
- All of the distinctive element from each combined set are present in final set.
- As parameters one or more sets may be passed to the union() function.
- You can use the union in two ways,
    a) union() function
    b) | this is called union operator.

Example-1:
----------------
days1 = {"mon","tue","fri","tue","sun",[]}
days2 = {"fri","sat","sun"}

print(days1.union(days2))

Example-2
---------------
days1 = {"mon","tue","fri","tue","sun"}
days2 = {"fri","sat","sun"}

print(days1 | days2)

Q. How to find common element between three set.
-------------------------------------------------------
s1 = {1,2,3}
s2 = {2,3,4}
s3 = {3,4,5}

common_element = s1.union(s2,s3)
print(common_element)


*2. Intersection of two sets*
-----------------------------------
- To find what is common between two or more sets in python, apply the intersection() function.
- Only the items in all sets being compared are included in final set.
- One or more sets can be also be used as the intersection() function parameter.
- We can use the intersection() function in two ways,
    a) intersection() function
    b) & --> this is called and operator.

Example -1:
-------------
days1 = {"mon","tue","fri","tue","sun"}
days2 = {"fri","sat","sun"}
print(days1.intersection(days2))

Example -2
-------------
days1 = {"mon","tue","fri","tue","sun"}
days2 = {"fri","sat","sun"}
print(days1 & days2)


*3.Intersection_update() method*
--------------------------------------
- The intersection_update() method removes the element from original set that are not present in both the sets.

Example
-----------
```

```
s1 = {1,2,3,4,5}
s2 = {3,4,5,6,7}
s1.intersection_update(s2)
print(s1)

o/p : {3, 4, 5}

*Q.What is the difference between intersection() function and intersection_update() function?*


*4. difference between two sets:*
--------------------------------
- The difference of two sets can be calculated by using the subtraction operator(-) as well as using difference() method.
- suppose there are two sets A and B, and the difference is A-B that denotes the resulting set will be obtained that element of A,
  which is not present set B.

 Example-1
 ------------
s1 = {1,2,3,4,5}
s2 = {3,4,5,6,7}
print(s1-s2)

o/p : {1,2}

Example-2
-----------
s1 = {1,2,3,4,5}
s2 = {3,4,5,6,7}
print(s1.difference(s2))

*5. Symmetric_difference between two sets*
----------------------------------------------
- The symmetric difference between set1 and set2 is the set of elements present in one set or the other set
  but not in bot set.
- We can use the method symmetric_difference() for this operation or we can use ^ cap operator.


Example-1:
------------
s1 = {1,2,3,4,5,6}
s2 = {1,2,9,8,10}
res = s1 ^ s2
print(res)

o/p:
----
{3, 4, 5, 6, 8, 9, 10}

Example -2
-----------------
s1 = {1,2,3,4,5,6}
s2 = {1,2,9,8,10}
res = s1.symmetric_difference(s2)
print(res)


*Set Comparisons*
------------------------
- In python you can compare sets to check if they are equal or of two sets have the common elements or if one set
 is subset another, for this operation set comparison operator are present in python.

- The following are some set comparision operator are present.
1. ==  :- check if two sets have the same element.
Example:
---------------
s1 = {1,2,}
s2 = {True,2}
print(s1 == s2)

2. !=  :- Checks if two sets are not equal.
Example:
----------------
s1 = {1,2,3,5}
s2 = {True,2,3}
print(s1 != s2)


3. <   :- checks if the left set is proper subset of the right set.(all elements in the left set are also
          present in right set, but the right set has additional element)
Example:
-----------------------
s1 = {"Mon","Tue","Wend","Thu"}
s2 = {"Mon","Tue"}
print(s1 < s2)

4. >   :- checks if the left set is proper subset of the right set.(all elements in the right set are also
          present in left set, but the left set has additional element)
Example:
----------------
s1 = {"Mon","Tue","Wend","Thu"}
s2 = {"Mon","Tue"}
print(s1 > s2)
=============================================================================================================================

*Python Array*
--------------------
Def : In python array is a collection of items or elements stored in at contigous memory location. The idea is to store multiple
items of the same data type.
- Arrays in python must have all elements of same type.
- Python supports only single dimensional array.
- If you want use 2D array then we use one library/module called as numpy.

Q. how we can use array in python?
------------------------------------
- If i want to use the array in python first i need to use inbuilt library called as array.
```

```
*Syntax*
-------------
name_variable = array_modulename('typecode',[values])

typecode : typecode is nothing but it is data types like int, float, string. The typecode has the value as below,
a) int ---> i
b) float ---> f
c) string ---> u
d) double --> d

Example -01:
-------------------
import array as arr

myArray = arr.array('i',[10,20,40,50,30])
# by using index we get only one element
print(myArray[0])

# by using slice operator we get multiple elements
print(myArray[0:3])

#by using loop conecept we can also print all elements from array
for i in myArray:
    print(i)


Example -02(change import statement and array declaration)
---------------------------------------------------------------
from array import *
myArray = array('i',[10,20,40,50,30])
# by using index we get only one element
print(myArray[0])

# by using slice operator we get multiple elements
print(myArray[0:3])

#by using loop conecept we can also print all elements from array
for i in myArray:
    print(i)


Example -03(Stored flaot value)
----------------------------------
from array import *  #---> This is called module/library ---> In which multiple classes present
myArray = array('f',[1.12,5.3,6.8,555])

#by using loop conecept we can also print all elements from array
for i in myArray:
    print(round(i,2))

*How we can access the element of array*
---------------------------------------------------
1. print() method
2. by using for loop or while loop.
3. by using index number we can get single element.
4. by using slicing method we get multiple element.

*how to take user input*
------------------------------
from array import *

#create empty array
myarray = array('i',[])

how_many_element = int(input("Enter how many elements = "))
for i in range(0,how_many_element):
    myarray.append(int(input("Enter the elemets = ")))

print("The array elements are below")
for j in myarray:
    print(j)

*What are the different methods present in array*
--------------------------------------------------------
1. append() : It adds the element at the end of array.
2. insert(index, value) : to insert the element at specific location.
3. count : It determine how many times a perticular value present in array.
4. extend: this method is used with array to add element from another array to the end of the array.
5. remove(value) : it remove the value which is present in array.
6. pop() : It remove the last element from array.
7. pop(index) : It removes the specific index value from array.
8. reverse() : reverse the order of the elements.
9. tolist() : convert the array into list.
=========================================================================================================================
*Numpy Introduction*
------------------------
- Numpy stands for Numerical Python, is open source python library that provides support for large, multi dimensional
  arrays.
- It also have a collection of high level mathematical functions to operate on arrays.
- It was created by Travis Oliphant in 2005.

Q. What is NumPy?
-----------------------
- NumPy is a general purpose array processing package.
- It provides a high performance multi dimensional array object and tools for working with these array.
- It is open source software.

Q. How to install numpy?
----------------------------
Numpy can be installed by using following command,
pip install numpy

Q. how we can create single dimensional array using numpy?
--------------------------------------------------------------
Example - 01:
------------------
```

```
import numpy

myarr = numpy.array([10,20,50,60])
print(myarr)

In above we used array() class to create the an array with 4 element.

Example -02
--------------------
from numpy import *

myarr = array([10,20,50,60])
print(myarr)

Example-03
----------------------
import numpy as np
myarr = np.array([10,20,50,60])
print(myarr)

Example-04
------------------------
import numpy as np
myarr = np.array([1.2,3.5,6.1,88],int)
print(myarr)

You can specify the different typecode which is shown as above, int, float, str

Example -05
------------------------
we can also store the string as below,
import numpy as np
myarr = np.array(["python", "java","javascript"],str)
print(myarr)

*We can also create the array by using arange() function:*
-----------------------------------------------------------------
The arange() function in numpy is same as range() function. arange() function is used with the following format.
arange(start, end, step)

Example-01:
--------------
from numpy import *
myArray = arange(10)
print(myArray)


Example-02
--------------------
from numpy import *
myArray = arange(2,11,2)
print(myArray)

Example -03
---------------------
from numpy import *
myArray = arange(2,11,2)
print("The array elements are")
for i in myArray:
    print(i, end=",")

*Mathematical Operations on arrays*
----------------------------------------------
It is possible to perform various mathematical operations like addition, subtraction, division etc... on the elements of an array.

Examples:
-----------------
from numpy import *

myArray = array([10,30,45,67,44],int)
print("orignal array= ",myArray)

#Arithmetic operations
print("The addition is = ",myArray + 5)
print("The subtraction is = ", myArray - 10)
# print("Square root = ",numpy.sqrt(myArray))
print("Sin value = ",numpy.pow(myArray,2))

*numpy multidimensional array*
------------------------------------
Multidimensional arrays are a key feature of the NumPy library in Python, allowing you to store and manipulate data in multiple dimensions efficiently. These a

Key Features of NumPy Multidimensional Arrays:
-------------------------------------------------------
1.Efficient Storage: Memory-efficient storage for large datasets.
2.Multi-Dimensional: Support for 1D, 2D, 3D, or higher-dimensional arrays.
3.Mathematical Operations: Perform element-wise operations, matrix operations, and more.
4.Indexing and Slicing: Access and modify specific elements or subsets easily.

*Creating Multidimensional Arrays*
-------------------------------------
import numpy as np

# 1D array
array_1d = np.array([1, 2, 3])

# 2D array
array_2d = np.array([[1, 2, 3], [4, 5, 6]])

# 3D array
array_3d = np.array([[[1, 2], [3, 4]], [[5, 6], [7, 8]]])

print("1D Array:\n", array_1d)
print("2D Array:\n", array_2d)
print("3D Array:\n", array_3d)
```

```
Important Attributes
--------------------------------
shape: Returns the dimensions of the array.
ndim: Returns the number of dimensions.
size: Returns the total number of elements.
dtype: Returns the data type of the elements.

print("Shape:", array_2d.shape)  # (2, 3)
print("Number of dimensions:", array_2d.ndim)  # 2
print("Size:", array_2d.size)  # 6
print("Data type:", array_2d.dtype)  # int32 (may vary)

Accessing Elements
------------------------------
Access elements using indices or slices:

Example:
----------------
# Accessing elements in 2D array
print(array_2d[0, 1])  # Element at row 0, column 1

# Slicing
print(array_2d[:, 1])  # Second column: [2, 5]
print(array_3d[1, :, 1])  # Second "slice", all rows, second column: [6, 8]

*user input in multidimensional array*
------------------------------------------
import numpy as np

rows = int(input("Enter the number of rows = "))
col = int(input("Enter the number of col = "))

myArr = np.zeros((rows,col), dtype=int) # empty array with 0

print("Enter the element row by row")
for i in range(rows):
    for j in range(col):
        myArr[i,j] = int(input(f"Element at [{i},{j}] = "))

print("\n 2D array is as below")
print(myArr)

*Matrix addition*
------------------------------
#matrix add operation using 2d array
import numpy as np

A = np.array(
    [
        [1,2,3],
        [4,5,6],
        [7,8,9]
    ]
)

B = np.array(
    [
        [9,8,7],
        [6,5,4],
        [3,2,1]
    ]
)

# result = A + B
result = np.add(A,B)
print("Matrix Addition:\n",result)

*Multidimensional Array*
--------------------------------
#this is the example of multi dimensional array

import numpy as np

# 3 X 3 = 9
myArr = np.array(
    [
        [1,2,3],
        [4,5,6],
        [7,8,9]
    ]
)

#print(myArr)
#print("Element at [0,1] =",myArr[0,1])
#print("Row1 = ",myArr[0,:])
#print("Col1 = ",myArr[0:2,0:3])

for i in range(myArr.shape[0]): #rows
    for j in range(myArr.shape[1]): #columns
        print(f"Element at [{i},{j}] is {myArr[i,j]}")

============================================================================================
*Functions in python*
------------------------------
- A function is similar to a programs that consist of group of statements that are used to perform specific task.
- The main purpose of function is to perform a specific task or work.
- There are several in built functions present in python that perform specific task, example print(), sqrt(), id(), range()
- Similar to these functions a programmer can also create his own function which are called as user defined functions.

*What are the advantages of functions*
----------------------------------------------
1. Functions are important in programming language because they are used to process the data, make calculations or perform any
   task which is required in the software development.
2. Once a function is written it can be reused as and when required, so function is also called reusable code. It means it is
   possible to avoid same code again and again.
3. code maintenance will become easy because of functions.
```

```
4. The use of functions in a program will reduce the length of the program.


*How we can define the function*
----------------------------------------------
1. We can define the function using the keyword def followed by function name.

Example:
----------------
def sum(num1,num2):
    result = num1 + num2
    print("Result = ", result)

def sub(num1, num2):
    result = num1 - num2
    print("Sub = ",result)


sub(5,3)
sub(8,7)
sub(9,2)

sum(10,20)
sum(7,8)
sum(8,9)

Example -2
------------------
def sub(num1, num2):
    result = num1 - num2
    return result

# The above function returns the result to the main function.
res = sub(5,3) # here we get result as 2
print(res) # here we print the result.


*Function returns multiple values:*
------------------------------------------------
- A function returns a single value in the programming languages like c, c++ and java.
- But, in python a function can return single value as well as multiple value.

Example:
===============
def DoTask(num1, num2):
    add_result = num1 + num2
    sub_result = num1 - num2
    mul_result = num1 * num2

    return add_result, sub_result, mul_result

num1 = int(input("Enter num1 = "))
num2 = int(input("Enter num2 = "))

add,sub,mul = DoTask(num1,num2)# here we call the function

print("The addition is = ", add)
print("The subtraction is = ", sub)
print("The mul is = ", mul)

*Default arguments:*
==========================================================================
- We can mention some default value for the function parameters in the definition.

Example:
------------------
def grocery(item,price=40.00):
    print("Item = %s" %item)
    print("Price = %.2f" %price)

#function calling
grocery(item="sugar",price=50)
grocery(item="Oil")

- Here, the first argument is item whose default value is not mentioned. But the second argument is price and its default value
  is mentioned to be 40.
- At the time of calling the function,this function if we don't pass any value to price then the default value will be set as 40.
- If the value mention the price value then that mentioned value is utilized.
- So, a default argument is an argument that assumes a default value if the value is not provided in the function call.


*Keyword arguments*
-------------------------------------
- Keyword arguments are arguments that identify the parameteres by their name.
- The following example display grocery item and its price can be written as ,
    def grocery(item,price=40.00):

- At the time of calling the function, we have to pass two values and we can mention which value is for what.

Example:
=========
grocery(item="sugar",price=50)

*Variable lenght of arguments:*
========================================
- Sometimes, the programmer does not know how many values a function may receive. In that case the programmer cannot decide how many
  arguments to be given in the function definition.
- For example, if the programmer is writing a function to add two values or numbers, he can write:
    add(a,b):

- But, the user who is using this function may want to use this function to find sum of three numbers. In that case there is chance of
  user may provide 3 arguments to this function as:
    add(7,8,5)

- Then the add() function will fail and error will be displayed. If the programmer wants to develop a function that can accept n number
  of argument that is also possible in python. For this purpose, a variable length argument is used to in the function definition.
```

```
- A variable length of argument is an argument that can accept any number of values.
- The variable length argument is written with a '*' symbol before it in the function definition.

Example:
-----------------
def add(*args):

- Variable length of argument has two types
1. keyword based arguments / dictionary
2. non keyword based arguments


*keyword based arguments*
=========================================
- Python passes variable length non keyword argument to a function using *args but we can not use this
  to pass keyword arguments.
- For this problem python has got a solution called **kwargs, it allows us to pass the variable length
  of keyword argument to the function.

 Example:
 -----------------------
def person(**data):
    for key, value in data.items():
        # print("{} is {}".format(key, value))
        print(key,"---------->",value)


person(Firstname="Ram", Lastname="Sharma")
person(Firstname="Suresh", Lastname="Sharma", age=45, city="Pune")



*non keyword based arguments*
=========================================
- The special syntax *args in function definition is used to pass a variable number of arguments to a function.
- It is used to pass non keyword based arguments.

Example:
---------------
# function to add 3 numbers
def adder(x,y,z):
    print("Sum =",x+y+z)

# i am calling the function
adder(10,12,13)
# When we run the above function it will print Sum as 35.

# Lets see what happens when we pass more than 3 arguments in the adder function.
adder(3,4,5,6,7,8,9)

# When we run above function,it will generate the below error
# TypeError: adder() takes 3 positional arguments but 7 were given

# So we solve the above problem as below,

def adder(*num):
    sum = 0
    for i in num:
        sum = sum + i
    print("Sum is : ", sum)


adder(2, 4)
adder(3, 4, 5)
adder(9, 7, 9, 5)
adder(8, 4, 6, 7, 2, 9, 3)
=====================================================================================================================

*Exception handling in python*
------------------------------
Def Exception :
-------------
Exception is unwanted error that are occure due to the mistake of developers. It always comes when your program is
running.

- Error in python can be of two types that is Syntax error and Exceptions.
- Error are problems in program due to which the program will stop the execution.
- To handle the exception python uses the keywords like try and except block.

*What are the different types of exceptions in python:*
---------------------------------------------------------
In python, there are servel built in python exceptions that can be raised when error occurs during the execution of
a program.
1) SyntaxError:
-----------------
    - This exception is raised when the interpreter encounters a syntax error in the code.
    - Such as a misspelled keyword, missing colon etc...
2) TypeError:
-----------------
    - Raised when operation or function is applied to an object of an inappropriate type.
    Example:
    ----------------
    print("abc" + 136)

3) ValueError:
-----------------------
    - Raised when a function receives an argument of the correct type but inappropriate value.
    Example:
    ---------
    print(int("abc"))

4) IndexError:
-------------------
    - Raised when trying to access an index tha is out of range.
    Example:
```

```
    -------------
    lst = [1,2,3,4,5]
    print(lst[7])

5) KeyError:
---------------
    - Raised when trying to access a key that does not exist in a dictionary.
    Example:
    ---------
    d = {"city":"Pune"}
    print(d["name"])

6) AttributeError:
-------------------
    - Raised when an object does not have the specified attribute.
    Example:
    ----------
        lst = [1,2,3,4,5]
        lst.appendd(100) # here i am writing wrong method
        print(lst)

7) FileNotFoundError:
------------------------
    - Raised when a file operations because the file does not exist.
    Example:
    ------------
        open(r"D:\test1.txt","r")
    In above statement i am looking for the file test1.txt on the D drive, but this type of file does not exists on that
    drive, so it will generate the exception FileNotFoundError: [Errno 2] No such file or directory: 'D:\\test1.txt'.

8) IndentationError:
-----------------------
    Raised when the code is not properly aligned.
    Example:
    --------
    def printData():
    print("Welcome")

8) IOError(input/output error):
--------------------------------
    - This exception is raised when an I/O operation, such as reading or writing a file fails due to an input/output error.

9) ImportError/ ModuleNotFoundError:
--------------------------------------
    - Raised when an import statement fails.
    Example:
    ------------
    import arrays ---> arrays module not present. The correct module name is array


10) ZeroDivisionError:
-----------------------
    - This exception is raised when an attempt is made to divide a number by zero.

*Q. What is the differenece between Syntax Error and Exception*
-------------------------------------------------------------------
- Syntax Error:
-------------------
1. As the name suggest this error is caused by the wrong syntax in the code. It leads to the termination of the program.
Example:
-----------
amount = 200
if amount > 300
    print("You can purchase the product")
else
    print("Check money")

In above code, there is syntax error in the code. The if statement should be followed by a colon(:).

Exceptions:
----------------
- Exceptions are raised when the program is syntactically correct, but the code results in an error. This error does not
  stop the execution of the program, however it changes the normal flow of the program.
Example:
-------------
marks = 100
res = marks / 0
print(res)

add = marks + 10
print(add)

* How you can avoid the exceptions:*
--------------------------------------------
- Try and except statements are used to catch and handle exceptions in python.
- The try clause contains the code that can raise the exception, while the except clause contains the code lines that handle
  the exception.
- You can use only one try statement but you can use multiple except blocks.
- The try block contains atleast one except block. You dont write try without except.

Example:
------------------
marks = 100
try:
    res = marks / 0
    print(res)
except ZeroDivisionError as err :
    print(err)
finally:
    print("hi welcome")

add = marks + 10
print(add)

---------------------------------------------------------------------------------------
*Try with else block*
```

```
--------------------
- The else clause in Python exception handling is used to specify a block of code that should run only if no exceptions were raised.

Syntax:
----------
try:
    #Code that may raise the exception
except:
    #Code to handle the exception
else:
    #Code that runs if no exception was raised.

Example:
-----------
def divide_numbers(num1,num2):
    try:
        result = num1 / num2
    except ZeroDivisionError:
        print("Error: Division by zero is not allowed")
    else:
        print("Division successful..! The result is = ", result)

divide_numbers(10,5)
divide_numbers(5,0)

-------------------------------------------------------------------------------------------------------------------
Raising Exception:
---------------------
- The raise statement allows the programmer to force a specific exception to occur.
- In Python you can raise an exception using the raise.
- The following is the basic syntax of built-in exception
    raise ExceptionType("Error message")

- You can also define your own exceptions by subclassing the Exception class:
- The following is the syntax of custom_exception
    class CustomeError(Exception):
        def __init__(self, message):
            super().__init__(message)

    #Raise the custom exception
    raise CustomError("This is a custom error")

Example -->01 ---> builtIn
-------------------------------
def divide_numbers(num1, num2):
    if num2 ==0:
        raise ZeroDivisionError("You cant divide by zero")
    return num1 / num2

try:
    result = divide_numbers(10,5)
    print("The result is = ",result)
except ZeroDivisionError as err:
    print("Error:",err)
else:
    print("Finally i am executed")


Example ---> 02 ---> Custom Exception
---------------------------------------
class NegativeNumberError(Exception):
    def __init__(self, message="Negative numbers are not allowed"):
        super().__init__(message)


def square_root(number):
    if number < 0:
        raise NegativeNumberError("Cannot compute the square root of negative number")
    return number ** 0.5
try:
    result = square_root(-25)
    print("The result is = ", result)
except NegativeNumberError as err:
    print("Error:", err)

=======================================================================================
*File handling*
-----------------
- Users can easily handle the files, like read, and write the files in python.
- In other programming language the file handling process is lengthy and complicated.
- But we know python is an easy programming language. So like other programming language the file handling is very easy.
- Sometimes it is not enough to only display the data on the console. The data to be displayed may be very large and only
  limited amount data can be displayed on the console.
- The file handling plays an important role when data needs to be stored permanently into a file.
- In python files are treated in two modes text and other is binary mode.

Q. How we can create file in python?
-----------------------------------------
a) Create a file in the current directory or specified directory.
b) Create a file if not exist.
c) Create a file with date and time as its name.

Syntax:
----------
open(file_path, access_mode)

- Access mode specifies the purpose of opening file.
- Below is the list of access modes for creating a file.

File Mode               Meaning
------------------------------------------------------------------
1. w                    Create a new file for writing. If a file already exist
                        it truncates the file first.
2. r                    This mode can be used for to read the file data.
3. a                    Open a file in the append mode and add new content at the end of the file.
```

```
Q. How we can create empty file?
---------------------------------------
Example:
--------------
fp = open(r'E:\python_aug\test.txt', 'w')
fp.close()

- In above example we create file in the location E:\python_aug folder.
- In above example we write 'r' because if not use r then in file path we write \text.txt so \t
  python considered as tab to avoid this we can use 'r' that indicated it is raw path.

Q. how we can write the data into a file?
--------------------------------------------
- We can write the data into a file we use the function called as write() function.
Example:
-----------
with open('test.txt', 'a') as fp:
    fp.write("javascript is easy.")

Q. How we can check if file exist at given location.
---------------------------------------------------------
- Sometimes it is important not to create a fileif file with same name aleady exists in the given path.
- By default when you open file in write mode, it overwrites if it exist.
- We can create the file only if it is not present using following way,
    Use os.path.exists('filename')

Q. How we can take user input to create a file?
-----------------------------------------------------
import os

#r"E:\python_aug\test1.txt"
file_path = r"E:/python_aug/"
file_name = input("Enter the name of the file = ")
complete_path = file_path+""+file_name
print("The path is = ",complete_path)
if os.path.exists(complete_path):
    print("Already exists")
else:
    with open(complete_path,"a") as fp:
        fp.write("File contents are stored")

Q. how to create a file with datetime as a filename?
----------------------------------------------------------
from datetime import datetime

# first we print current date and time
curr_dt = datetime.now()

# If i want to convert this datetime into a python string then we use strftime() function
file_name = r"E:/python_aug/" + curr_dt.strftime('%Y-%m-%d-%H-%M-%S.txt')
with open(file_name, "a") as fp:
    fp.write("Python is very easy programming language")
    print("Created", file_name)
---------------------------------------------------------------------------------------------------------
Q. how we can read the file contents?
--------------------------------------
Suppose we want to read the contents of the file line by line then we use predefined function called
read() function.

Example:
------------------
try:
    fp = open(r"E:\python_aug\test.txt","r")
    print(fp.read())
    fp.close()
except FileNotFoundError:
    print("Please check the path")


- The readLines() function can be used to convert file content into a list.

Example:
-------------
try:
    fp = open(r"E:\python_aug\test.txt","r")
    print(fp.readlines())
    fp.close()
except FileNotFoundError:
    print("Please check the path")

===============================================================================================
Python with mysql connection
-------------------------------
To connect with python mysql we need one library called as pymysql.
PyMYSQL is a interface between your python code and sql. Simply it is bridge between python and
mysql.

cursor --> it has one important task that it executes the sql query. Cursor can be created from conn
object.
---------------------------------------------------------------------------------------------------

Hotel_App:
------------
How to hide password from users?
------------------------------------
To hide the password from users we can use one file format called as .env file. So if we want to use
.env file then we must installed first.

Main menu
1. Add Category
    a) Add new category
    b) Update Category

2. Add Product
3. Add orders
```

```
cursor.execute(sql_query, (variable_name))
=========================================================================================================================
Class and Objects(OOPS)
---------------------------------
function : It is declared outside the class.
method : It is declared inside the class.

lst = [1,3,5,6]
List is class ---> inbuilt ---> inside of this different methods are present like insert, pop, remove ---> but we cant access
this method directly --> hence we create the object and by this object we can access the methods.
Example:
-------------
lst = [1,3,5,6,7]

Suppose we can insert new element into list then we use the method.
lst.insert(index, value)  ---> here dot represent object.

What is mean by class and object?
-----------------------------------------------
class :
------
The class is user defined data structure that binds the data members and methods into a single unit.
Class is blueprint or code template.
Class contains,
a) data members
b) member functions(methods)
c) constructor


data members : variable declarations
member function(methods) : It is reusable code. We cant directly access the method. So we create the object of the class.

Object:
-------------
An object is instance of class. You can create multiple objects from one class.  It is collection of data member and method.
We can use the object to perform actions. Object has the two characteristics one is state of an object and another is behaviour
of an object. The following information shows the real time and in programming what is state and behaviour.


In Real Life
---------------------
State : colour, height, width, weight
behaviour : writing , code writing, watching movies,

In Programming language:
-------------------------
State : An object has an attribute that represent a state of the object.
Attribute is nothing but extra information about the tag.
Example:
----------
<img src="" height="" width = "" alt="" >

behaviour: methods represents the behaviour.
Suppose i will create one bank project, in that there are two methods one is withdraw method and another is deposite method.

A real life of example of class and object.
Class : Person
    State : name, gender, email, mob, city
    Behaviour : Working, Study, driving ...etc..

Using the above class we can create multiple object.
Object 1:
    State:
         name : Ashok
         gender : male
    Behaviour:
         Working : He is software engineer in ABC company

Object 2:
    State:
         name : Sunil
         gender : male
    Behaviour:
         Working : He is Manager at XYZ company


Q. how we can create the class?
-------------------------------------------------
In python class is defined by using the keyword "class". The syntax of class declaration is as below,
The class name always start with first letter capital.

When we declared any method inside the class the first argument in method is self keyword. If you dont
want use name self then  You can use any name.

The self refers to the current object or variables.

any name

class class_name:
    Data members
    member functions(methods)
    Constructor

 Example:
 ----------------
class Test:
    # member function(methods)
    def show_message(self):
        print("Hello i am from class")

    def show(self):
        print("I am from show")


# if i want to access the show_message() method i need to create the object of class.
```

```
obj_test = Test()  # here i am creating the object
obj_test.show_message()
obj_test.show()

Q. What is mean by constructor in python?
------------------------------------------------
- A constructor in python is special method called when an object is created.
- Its purpose is to assign values to the data members within the class.
- Every python class has a constructor.

Q. What are the rules for constructor?
--------------------------------------------------
1. The constructor method must be named __init__ .
2. The first argument of the constructor must be self. This is reference to the object itself.
3. The constructor method must be defined within the class. It can not be defined outside the class.
4. The constructor automatically called when object of the class is created.
5. You can define both default constructor and parameterized constructor in class. If you define both
   the parameterized constructor will be used when you pass arguments to the object.

Example:
----------------
class Person:
    # here i can define the parameterized constructor
    def __init__(self, name, city, age):
        # here i can define the data members
        self.name = name
        self.city = city
        self.age = age

    def dispalyData(self): # this is the method
        print("The name is = ", self.name)
        print("The city is = ", self.city)
        print("The age is = ", self.age)

# here i can define the object of the class, When we define the object of the class the constructor will
# automatically called.

pname = input("Enter the name of the person = ")
pcity = input("Enter the city name = ")
page = int(input("Enter the age of the person = "))

obj_person  = Person(pname, pcity, page)
obj_person.dispalyData()

Example -2:
-------------------------
Store 5 Persons data
# here i can create the list to store the names of the person
person_lst = []

class Person:
    def __init__(self, name):
        self.name = name

    def show(self):
        print("-------------------------------------------------")
        for per in person_lst:
            print("Hello ", per.name)


# input 5 persons name
for i in range(5):
    name = input(f"Enter the name of Person {i + 1} = ")
    person_lst.append(Person(name))

# create the object of the class and called show() method.
obj_per = Person(person_lst)
obj_per.show()


Q. What are the different types of constructor?
--------------------------------------------------------
In Python there are 2 types of constructors

1. Default constructors / non-parameterized constructors : A default constructors is constructor that
   takes no arguments.
2. Parameterized constructors : A parameterized constructor is constructor that take one or more
   arguments

=========================================================================================================
Inheritance in Python:
------------------------
What is mean by Inheritance In python?
-----------------------------------------------
- The process of inheriting the properties of the parent class into a child class is called as inheritance.
- The existing class is called a base class or parent class.
- The new class is called subclass or child class or derived class.
- In OOP, inheritance is an important aspect. The main purpose of inheritance is the code reusability.
- We can use the existing class to create a new class instead of creating from scratch.

Q. What are the different types of Inheritance?
----------------------------------------------------
1. Single level Inheritance
2. Multiple Inheritance
3. Multilevel Inheritance
4. Hybrid Inheritance
5. Hierarchical Inheritance

1. Single level Inheritance:
------------------------------
In single level inheritance a child class inherits from a single parent class. Here is one child class
and one parent class.

Examples:
--------------
class Vehicle:
    def info(self):
        print("Inside the vehicle class")
#  child class
class Car(Vehicle):
    def car_info(self):
        print("Inside the car class")

# Inheritance always create object of child class
car_obj = Car()
car_obj.info()
car_obj.car_info()

2. Multiple Inheritance
----------------------------------
In multiple inheritance, one child class can inherit from multiple parent classes, is called multiple
inheritance.

Example:
---------------
class Person:
    def person_info(self, name, age):
        print("Inside the person class")
        print("Name = ", name, "Age = ", age)

# Parent class 2
class Company:
    def company_info(self, company_name, company_loc):
        print("Inside the company class")
        print("Company Name = ", company_name, "Company Loc = ", company_loc)

# Child class
class Employee(Person, Company):
    def emp_info(self, salary, skill):
        print("Inside the Employee class")
        print("Salary = ", salary, "Skills = ", skill)

emp_obj = Employee()
emp_obj.person_info("Ajinkya", 34)
emp_obj.company_info("GSoft","Warje-Pune")
emp_obj.emp_info(2000,"Trainer")


3. Multilevel Inheritance:
-----------------------------------------
- In multilevel inheritance a class inherits from a child class or derived class.
- Suppose three classes A,B,C. A is the superclass, B is the child class of A, C is the child class
  of B. We can say that it is a chain of classes that is called multilevel inheritance.

  Example:
  -----------
  # Base class
class Vehicle:
    def vehicle_info(self):
        print('Inside Vehicle class')

# child class
class Car(Vehicle):
    def car_info(self):
        print("Inside from car")

class SportCar(Car):
    def sport_info(self):
        print("Inside the sport car")

# we create the object of sport car
sport_obj = SportCar()

sport_obj.vehicle_info()
sport_obj.car_info()
sport_obj.sport_info()


4. Hierarchical inheritance:
-----------------------------------
Multiple child classes are inherit from one parent.
Examples:
-----------
#one base class
class Vehicle:
    def vehicle_type(self):
        return "This is a vehicle"

#Child class-1
class Car(Vehicle):
    def car_features(self):
        return "Car runs on road"

# Child class -2
class Truck(Vehicle):
    def truck_features(self):
        return "Truck is heavy"

# objects
obj_car = Car()
obj_truck = Truck()

print(obj_car.vehicle_type())
print(obj_car.car_features())
print(obj_truck.vehicle_type())
print(obj_truck.truck_features())

5. Hybrid Inheritance:
--------------------------------
It is the combination of multiple types of inheritance
Example
------------------
```

```python
# take one base class
class A:
    def method_A(self):
        return "Method A"

# class B inherits from A
class B(A):
    def method_B(self):
        return "Method B"

# class C inherits from class B
class C(A):
    def method_C(self):
        return "Method C"

# class D inherits from class B and C
class D(B,C):
    def method_D(self):
        return "Method D"

# create the object
obj_d = D()
print(obj_d.method_A())
print(obj_d.method_B())
print(obj_d.method_C())
print(obj_d.method_D())
```