```
==============================================
              Exception Handling
==============================================
```

Index

===

Exception Handling

===

=>The purpose of Exception Handling in python is that " To develop Robust(Strong)
Application."

=>In real time application development, to develop any project, we need to choose a programming language. By using programming languages, we develop, compile and execute various programs. During this process, we get 3 types of Errors. They are

1. Compile Time Errors
2. Logical Errors
3. Runtime Errors

----

1. Compile Time Errors

=>Compile Time Errors occurs durings Compilation Process(.py----->.pyc)
=>Compile Time Errors occurs due to Syntaxes are not followed
=>Compile Time Errors Solved by Programmers during Development Time.

----

2. Logical Errors

=>Logical Errors occurs durings Execution Time(.pyc----->machine code)
=>Logical Errors occurs due to wrong representation of logic
=>Logical Errors always gives wrong result
=>Logical Errors Solved by Programmers at Development time.

----

3. Runtime Errors

=>Runtime Errors occurs during Execution or Runt Time
=>Runtime Errors occurs due to Invalid or Wrong Input entered by Application Users(End Users)
=>Runtime Errors must be addressed by Programmers in Projects for making the application as Robust

==

Building Points for learning Exception Handling

==

1) When the application User Enters Invalid or Wrong input then we get Runtime Errors.
(Invalid Input----->Run Time Error)

2) By default Runtime Errors generates Technical Error Messages,which are understandable by programmers but not by End Users.

3) Definition of Exception: Runtime Error of Every Program is called Exception.
(Invalid Input----->Run Time Error---->Exception)

4) By default all exceptions generates Technical Error Messages,which are understandable by programmers but not by End Users. Industry is recommended to generates User-Freindly Error Message by using of Exception Handling.

5) Definition of Exception Handling:- The process of Converting Technical Error Messages into
User-Freindly Error Messages is called Exception Handling.

6) When the exception occurs then the following 3 things takes .
a) Program Execution Abnormally Terminated

b) PVM Comes out of program flow
c) By default PVM Generates Technical Error Messages.

7) To do Step-(a), Step-(b), Step-(c), Internally, PVM creates an object of appropriate exception class.

8) In Python Programming, When an exception occurs then PVM creates an object of appropriate exception class

9) Hence Every Exception makes the PVM to create an object appropriate exception class

10) Therefore Every Exception is considered as object.

===

Types of Exceptions

===

=>In python programming we have two types of exceptions. They aare

1. Pre-defined or Built-In Exceptions
2. User OR Programmer or Custom defined exceptions

----

1. Pre-defined or Built-In Exceptions:

=>These exceptions devloped by Language Developers and avilable as a part of Python Software and whose role is to deal with Universal Problems.

=>Some of the universal problems are

1. Divide by zero (ZeroDivisionError)
2. Invalid Indexes (IndexError)
3. Module not existing (ModuleNotFoundError)
4. Invalid Number Conversion (ValueError)
5. Invalid Operations (TypeError)
6. Invalid Variable Name (NameError)....etc

----

2. User OR Programmer or Custom defined exceptions

=>These exceptions devloped by Python Programmers and avilable as a part of Python Project and whose role is to deal with Common Problems.

=>Some of the common problems are

a) Attempting to enter invalid PIN in ATM based applications.
b) Attempting to withdraw More amount than existing balance in an Account.
c) Attempting to enter Invalid user name / password .
d) Attempting to enter invalid patterns
e) wrong OTPs during payment ...etc

==

=>Handling the Exceptions in Python Program is nothing but converting the Technical Error Messages into user-Frendly error messages.
=>For converting the Technical Error Messages into user-Frendly error messages, we have 5 keywords. They are

a) try
b) except
c) else
d) finally
e) raise

=>Syntax for handling the exceptions

```
try:
    Block of statements generates
    Exceptions
except <exception-class-name-1>:
    Block of statements generates
    User-Friendly Error Messages.
except <exception-class-name-2>:
    Block of statements generates
    User-Friendly Error Messages.
-------------------------------------------
except <exception-class-name-n>:
    Block of statements generates
    User-Friendly Error Messages.
else:
    Block of statements recommended to generates
    Results.
finally:
   Block of statements which will execute compulsorily.
```

==
Explanation for the keywords of exception handling
==

=>As a part of exception handling, For converting the Technical Error Messages into user-Frendly error messages, we have 5 keywords. They are

a) try
b) except
c) else
d) finally
e) raise

----
a) try block

=>It is the block, in which we write block of statements which aregenerating exceptions. In Otherwords, what are all the statements generating exceptions, such type of statements must be written within try block and try block is called exception monitering block.
=>When an exception occurs in try block then PVM comes out of try block and executes appropriate
except block.
=>After executing appropriate except block , PVM never goes to try block for executing rest of the
statements in try block.
=>Every try block must be immediately followed by except block.
=>Every try block must contain atleast one except block and recommnded to write multiple except blocks
for generating multiple user-friendly error messages.

----
b) except block

=>It is the block, In which we write block of statements which are generating User-Friendly error messages. In Otherwords, except block supresses the Technical error messages and generates User-Friendly Error Messages and hence except block is called exception processing block.
=>except block will execute when an exception occurs in try block.
=>Even we write multiple except blocks,when an exception occurs in try block then PVM executes appropriate except block.
=>The place of writing except block is that after try block and before else block.

---

c) else block

=>It is the block, In which we write block of statements which are displaying Results.
=>else block will execute where there is no exception in try block.
=>Writing else bock is optional
=>The place of writing else block is that after except block and before finally block(if it present).

---

d) finally block

=>It is the block, In which we write block of statements which are relinquishing(close / release/ give-up/clean-up) the resources (files/Database) which are obtainded in try block
=>finally block will execute compulsorily
=>Writing finally block is optional
=>The place of writing finally block is that after else block.

raise key word

=>raise keyword is used for hitting / raising / generating the exception provided some condition must be satisfied.
=>raise keyword always used inside of Function Definition only.
=>PVM uses raise keyword implicitly for hitting pre-defined Exception where as Programmer makes the PVM to use raise keyword explicitly for Hitting or Generating Programmer-defined Exceptions.

=>Syntax:-
```
if (Test Cond):
        raise   <exception-class-name>
```

=>Syntax:-
```
def     functionname(list of formal parms if any):
        ----------------------------------------------
        ----------------------------------------------
         if (Test Cond):
                raise   <exception-class-name>
        ----------------------------------------------
```

Examples:

```
from kvr import KvrDivisionError
def division(a,b):
        if(b==0):
                raise KvrDivisionError
        else:
                return (a/b)
```

```
                    =====================================================
                                Various Forms of except block
                    =====================================================
```
=>We can use except block in different forms. They are

Form-1: except block with single exception (standard form)

```
       try:
          ---------------
          ---------------
          ---------------
       except  <exception class name-1>:
              ---------------------------------
              ---------------------------------
       except  <exception class name-2>:
              ---------------------------------
              ---------------------------------
       except  <exception class name-n>:
              ---------------------------------
              ---------------------------------
```

Example Program: Div2.py

Form-2: except block with multiple speficific exceptions--known as multi exception handling block.

```
       try:
          ---------------
          ---------------
          ---------------
       except (<exception class name-1>,<exception class name-2>.......):
            -------------------------------------
            Block of statements generating
            multiple User-Freindly Error Messages
            for multiple exceptions
            -----------------------------------------
```
Example Program: Div3.py

Form-3: except block with speficific exception name with alias name

----

```
       try:
          ---------------
          ---------------
          ---------------
       except  <exception class name-1> as alias name:
              ---------------------------------
              ---------------------------------
```
=>In The alias name , we are capturing the message, which is occured due to exception occurence.

Example Program: Div4.py

Form-4: except block without exception name and it is called default except block

---

```
       try:
          ---------------
          ---------------
          ---------------
       except  :  # default except block
              --------------------------------
              --------------------------------
```

=>default except block must be written always at last otherwiose we get SyntaxError

Development of Programmer or User or Custome Defined Exceptions

=>These exception developed by Python Language Programmers and avialable in Python Project and used by all Other python progarmmers for dealing with Common Problems.
=>Some of the Common Problems are

1) Attempting to enter Invalid PIN in ATM applications
2) Attempting to enter wrong User name and password
3) Attempting to withdraw more amount than existing bal in Account.........................etc

=>When the Application user enters Valid Input then we get Valid Output
=>When the application user enters Invalid Input then we get exception. When exception occurs then we must create an object by using appropriate exception class. If the exception class is Pre-defined then it is Pre-defined Exception Class and if it is Programmer-Defined then it is Programmer-defined exception class.

Steps for developing Programmer-Defined Exception class:

---

1. Define a Programmer-Defined Class

2. Programmer-Defined Class must Inherit from super class for all the exception whose name is "Exception" or "BaseException" bcoz "Exception" or "BaseException" provides Exceptrion handling Properties for abnormal Termination when application user enters invalid Input.

3. Save the above development of the code on some file name with an extension .py.

Number of Phases required for developing programmer-defined exception based applicatrions.

Phase-1: Development of Exception classes
Phase-2: Development of Common Functions which will hit the exceptions
Phase-3: Handling the Exceptions

Phase-1: Development of Exception classes---Example

```
       #kvr.py-----File Name and acts as module name
       class  KvrDivisionError(Exception):pass
```

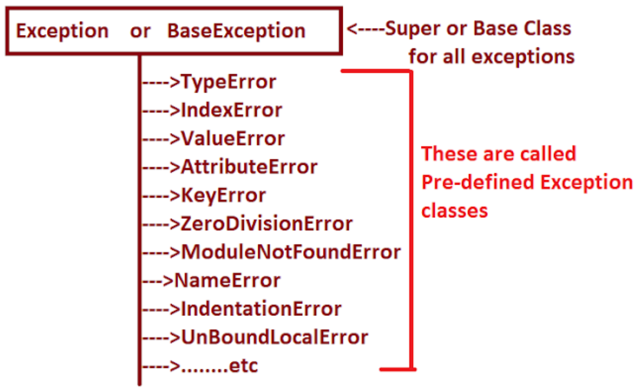
Phase-2: Development of Common Functions which will hit the exceptions

```
              #division.py-----file name and module name
```

```
                from kvr import KvrDivisionError
                def    division(a,b):  # Development of Common Function--
Phase-2
                        if(b==0):
                                raise KvrDivisionError  # Hitting or raising or
generating
Programmer-defined exception
                        else:
                                return(a/b)
```

Phase-3: Handling the Exceptions :

```
                #DivDemo.py
                from division import division
                from kvr import KvrDivisionError
                try:
                        a=int(input("Enter First Value:"))
                        b=int(input("Enter Second Value:"))
                        res=division(a,b)
                except KvrDivisionError:
                        print("\ndon't enter zero for den...")
                except ValueError:
                        print("\nDon't enter strs, symbols and alpha-
                              numerics")
                else:
                        print("Div=",res)
                finally:
                        print("i am from finally block")
```



=======================================**===

==
Files in Python
==

Index

===
Files in Python(OR)Stream Handling in Python
===

=>The purpose of Files concept is that " To store the data Permanently ".
=>The process of storing the data permanently is known as "Persistentcy".

=>In The context of files, we can develop two types of applications. They are

a) Non-persistency Applications
b) Persistent Applications.

=>In Non-persistency Application development, we accept the data from Key board,stores in main memory in the form of objects, process the data and shown the result on the console .
=>The results which are available in the main memory or Temporary.

=>In persistent Application development, we accept the data from Key board,stores in main memory in the form of objects, process the data and whose results are stored permanently.

=>To store the result of any programming language permanently, we have two approaches. they are

a) By using Files
b) By using data base softwares.

=>Note that if we write any python programming by using Files and Database software's then such type of applications are comes under the example Persistent Applications.

==
Data Persistenecy by Files of Python
==

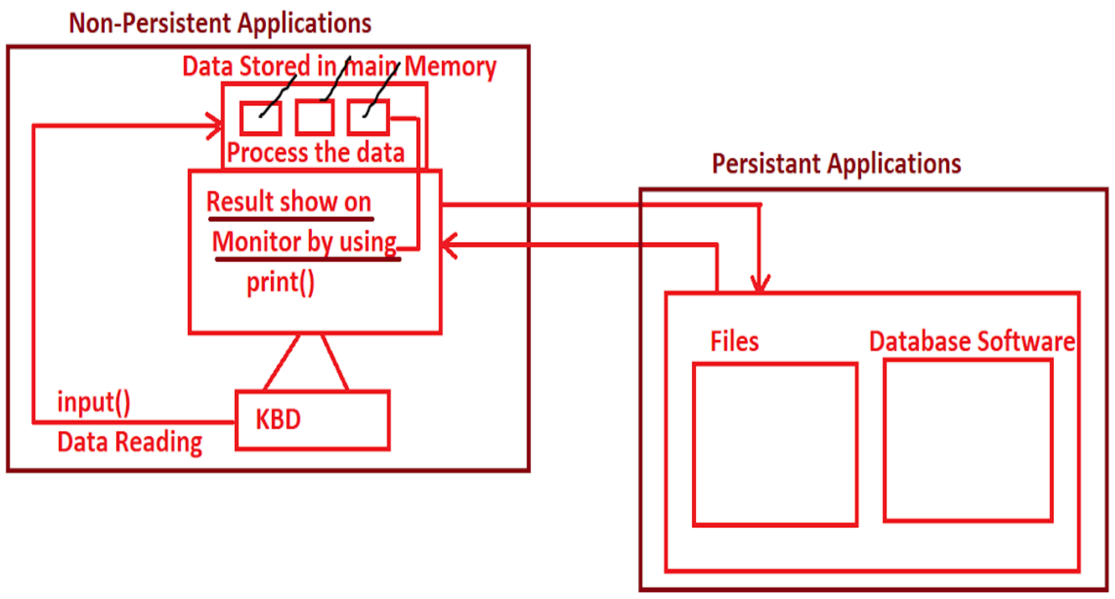
Def. of File:

=>A File is a collection of Records.
=>Files Resides in Secondary Memory.
=>Technically, File Name is a named location in Secondary Memory.

=>All the objects data of main memory becomes records in File of Secondary memory and records of file of secondary memory becomes the objects in main memory.

Def. of Stream:

=>The Flow of Data between object(s) of Main Memory and Files of Seconday memory is called Stream.


Non-Persistent Applications
Data Stored in main Memory
Process the data
Result show on Monitor by using print()
input()
Data Reading
KBD
Persistant Applications
Files
Database Software

```
================================================
                Operations on Files
================================================
```
=>On the files, we can perform Two Types of Operations. They are

1) Write Operation.
2) Read Operation.

1) Write Operation:

=>The purpose of write operation is that " To transfer or save the object data of main memory as record in the file of secondary memory".

=>Steps:

1) Choose the File Name
2) Open the File Name in Write Mode
3) Perform cycle of Write Operations.

=>While we performing write operations, we get the following exceptions.

a) IOError
b) OSError
c) FileExistError

----

2) Read Operation:

=>The purpose of read operation is that " To transfer or read the record from file of secondary memory into the object of main memory".

=>Steps

a) Choose the file name
b) Open the file name in Read Mode
c) Perform cycle of read operations.

=>While we performing read operations, we get the following exceptions.

a) FileNotFoundError
b) EOFError

===
Persistent Application development by using Files in Python
===

=>File Concept is one language Independent where we can store the data permanently.
=>The communication between Python Program and Files is comes under development of Persistant Application.

=>Definition of File:

=> A File is a collection of Records
=>A File Name is one of the Named Location in Secondary Memory.
=>Files of any programming resides in Secondary Memory.
=>In Files Programming, Every Object Data becomes a record in a file of secondary memory and every record of file of sedondary memory will become an object in main memory.

Definition of Steram:

=>The flow of data between main memory and file of secondary memory is called Stream.

================================
Types of Files
================================

=>In Python Programming, we have two types of Files. They are
a) Text Files
b) Binary Files

1) **Text File:**

=>A Text File always contains Alphabets, Digits and Special Symbols.
=>Text Files always denoted by a letter "t"
=>The default file in python is text file.

Examples: .py .java .c .cpp .txt .doc....etc

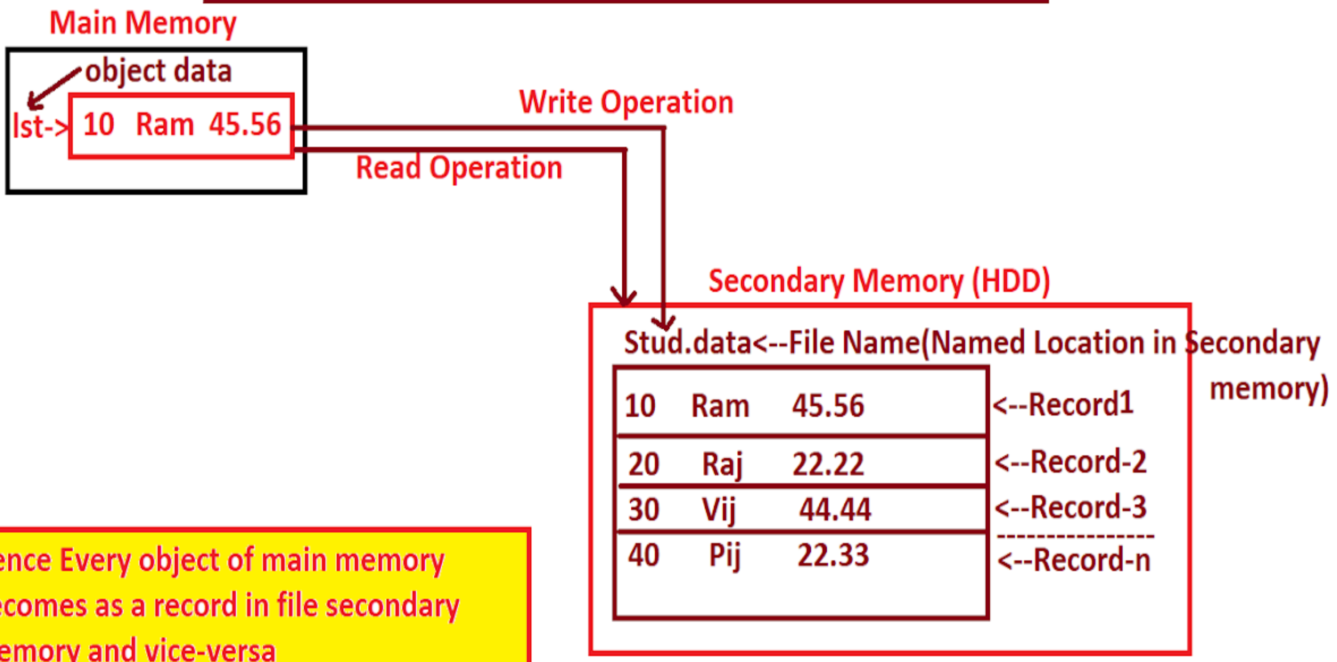
----

2) **Binary File:**

=>A BinaryFile always contains Data in Binary Format (Pixles)
=>Binary Files always denoted by a letter "b"

Examples: images (.jpg, .jpeg, .png, .gif)
audio and video files
PDF documents

View of objects of main memory and Secondary Memory



Hence Every object of main memory becomes as a record in file secondary memory and vice-versa

===
File Opening Modes
===

=>The purpose of File Opening Modes is that " In which mode the files are opened.
=>To perform any type of operation on the files, the files must be opened in appropriate mode.
=>In Python Programming, we have 8 file opening modes. they are

---

1. **r**

=>This mode is used for Opening the file Name in Read Mode
=>It is one of the default file mode
=>If we open the file name in "r" mode and if the file does not exist then we get FileNotFoundError.

---

2. **w**

=>This mode is used creating the file and opening the file in write mode always.
=>When we open NEW FILE in "w" Mode then new file will be opened in write mode and data written form begining of the file always.
=>When we open EXISTING FILE in "w" Mode then existing data of the existing file OVERLAPPED with new Data.

---

3. **a**

=>This mode is used creating the file and opening the file in write mode always.
=>When we open NEW FILE in "a" Mode then new file will be opened in write mode and data written form begining of the file always.
=>When we open EXISTING FILE in "a" Mode then existing data of the existing file APPENDED   with new Data.

---

4. **r+**

=>This mode is also used for Opening the file in Read Mode and Performs Read Operation.
=>After reading the data from file and later we can also Perform Write Operation.
=>When we open the file in "r+" mode and if the file does not exist then we get FileNotFoundError.

---

5. **w+**

=>This mode is used for Opening the file in write mode and Performs Write Operation always and later Additonally we can perform Read Operations also.
=>When we open NEW FILE in "w+" Mode the new file will be opened in write mode and data written form begining of the file always and later we can read the data also.
=>When we open EXISTING FILE in "w+" Mode then existing data of the existing file OVERLAPPED with new Data and later we can read the data also.

---

6. **a+**

=>This mode is used creating the file and opening the file in write mode always and performs write operation First and later additionally we can perform read Operation also.
=>When we open NEW FILE in "a+" Mode the new file will be opened in write mode and data written form the begining of the file always and later we can perform read operation.
=>When we open EXISTING FILE in "a+" Mode then existing data of the existing file APPENDED with new Data and later we can perform read operation.

---

7. **x**

=>This mode is used for creating the new file and opening that file in Write Mode eXclusively only once.
=>If we open exitsing file in "x" mode then PVM generates FileExistError.

---

8. **x+**

=>This mode is used for creating the new file and opening that file in Write Mode eXclusively only once and after performing Write Operation and later we can also perform Read Operations also.
=>If we open exitsing file in "x+" mode then PVM generates FileExistError.

Syntax for Opening the files

=>In Python Programming, we have two syntaxes for opening the files. They are

a) By using open()
b) By using " with open() as "

a) By using open()

=>Syntax:- varname=open("FileName","File Mode")

Explanation:

=>varname represents an object of type <class,"TextIOWrapper"> and it is called File Pointer(is nothing but varname points to the file).
=>open() of the pre-defined function present in "builtins" and it is used for opening the specified File in specified File Mode.
=>"FileName" represents name of the file specified by Programmer
=>"FileMode" represents any file available file opening modes(r,w,a,r+,w+,a+,x)
=>When we open any file with Open() then we must close that file manually by using close().In otherwords Auto-Closable Property not maintained.

---

a) **By using "with open() as "**

Syntax:

```
------------
              with   open( "File Name","File Mode") as  <varname>:
                       ---------------------------------------------------
                       ---------------------------------------------------
                      File related Operation stmts
                      ----------------------------------------------------
                      ----------------------------------------------------

              ------------------------------------------------------------
              Block of statements
              ------------------------------------------------------------
```

Explanation:

=> 'with' and 'as' are the key words
=> open() of the pre-defined function present in "builtins" and it is used for opening the specified File in specified File Mode.

=>varname represents an object of type <class,"TextIOWrapper"> and it is called File Pointer(is nothing but varname points to the file).
=>"FileName" represents name of the file specified by Programmer
=>"FileMode" represents any file available file opening modes(r,w,a,r+,w+,a+,x)
=>The Advantage of "with open() as " approach is that Auto-Closing the File. In otherwords Programmer need not to close the file manually.

Writing the data to the file

=>To write the data to the file, we have two pre-defined functions. They are
1) write()
2) writelines()

---

1) write()

=>This function is used for writing any type of data to the file in the form of str.
=>If we want to write Numerical data to the file then we convert numerical data into str type.

=>Syntax: - filepointer.write(strdata)

Examples

```
#DataWriteEx1.py
with open("addr1.data","a") as fp:
       fp.write("Mc Kinney\n")
       fp.write("FNO 112  North Side\n")
       fp.write("Pandas Software Foundation\n")
       fp.write("Fin Lands-56\n")
       print("Data Written to the file:")
```

----
2) writelines()

=>This function is used for writing any type of Iterable Object data to the file in the form of str.

=>Syntax: - filepointer.writelines(Iterable Object)

Examples:

```
#DataWriteEx2.py
d={10:"Python",20:"Java",30:"Data Sci"}
fp=open("add2.data","a")
fp.writelines(str(d)+"\n")
print("Data Written to the file")
```

==

Reading the Data from the File

==

=>To read the data from the file, we have 4 pre-defined functions. They are

1) read()
2) read(no.of chars)
3) readline()
4) readlines()

----
1) read()

=>This Function is used for reading ENTIRE DATA of the file in the form of str.

=>Syntax:- strobj=filepointer.read()

Examples:-

```
#FileReadEx1.py
filename=input("Enter any File Name for displaying its content:")
try:
       with open(filename,"r") as fp:
              filedata=fp.read()
              print("-"*50)
              print("Content of File:")
              print("-"*50)
              print(filedata)
              print("-"*50)
except FileNotFoundError:
       print("{}: does not exist:".format(filename))
```

----
2) read(no.of chars)

=>This Function is used for reading Specified Number of Chars from the file in the form of str.

=>Syntax:- strobj=filepointer.read(No. of chars)

Examples:

```
#write a python program reading certain chars from given file---tell()
seek()
#FileReadEx2.py
try:
       with open("addr1.data","r") as fp:
              print("Initial Position of fp=",fp.tell())# 0
              filedata=fp.read(6)
              print("No. of chars=",filedata) # Travis
              print("Now Position of fp=",fp.tell())# 6
              filedata=fp.read(8)
              print("No. of chars=",filedata) #
              print("Now Position of fp=",fp.tell())# 14
              filedata=fp.read(50)
              print("No. of chars=",filedata) #
              print("Now Position of fp=",fp.tell())# 67
              filedata=fp.read()
              print("\nNo. of chars=",filedata) #
              print("Now Position of fp=",fp.tell())# 67
              #re-set to the Initial Index
              fp.seek(0)
              print("\nNow Position of fp=",fp.tell())# 0
              filedata=fp.read()
              print("File Data=",filedata)
except FileNotFoundError:
       print("{}: does not exist:".format(filename))
```

----

3) readline()

=>This Function is used for reading One Line at a time from the file in the form of str.

=>Syntax:- strobj=filepointer.readline()

Examples:

```
#write a python program reading one line at a time from File
#FileReadEx3.py
try:
       with open("addr1.data","r") as fp:
              filedata=fp.readline()
              print(filedata)
              filedata=fp.readline()
              print(filedata)
              filedata=fp.readline()
              print(filedata)
except FileNotFoundError:
       print("{}: does not exist:".format(filename))
```

----

4) readlines()

=>This Function is used for reading ALL Lines at a time from the file in the form of list.
=>Syntax:- strobj=filepointer.readline()

==

Pickling and Un-Pickling
(OR)
Object Serialization or Object De-Serialization

==

Pickling (Object Serialization)

=>Let us assume there exist an object which contains multiple values. To save or write object data of main memory into the file of secondary memory by using write() and writelines() , they transfers the values in the form of value by value and it is one of the time consuming process(multiple write operations).

=>To Overcome this time consuming process, we must use the concept of Pickling.
=>The advantage of pickling concept is that with single write operation , we can save or write entire object data of main memory into the file of secondary memory.

=>Definition of Pickling:

=>The Process saving or transfering entire object content of main memory . into the file of secondary memory by performing single write operation is called Pickling
=>Pickling concept participates in Write Operations.

Steps for implementing Pickling Concept:

=>import pickle module, here pickle is one of the pre-defined module
=>Choose the file name and open it into write mode.
=>Create an object with collection of values (Iterable object)
=>use the dump() of pickle module. dump() save the content of any object into the file with single write operation.

Syntax: pickle.dump(object , filepointer)

=>NOTE That pickling concept always takes the file in Binary Format.

Un-Pickling (Object De-Serialization)

=>Let us assume there exists a record with multiple values in a file of secondary memory. To read or trasfer the entire record content from file of secondary memory, if we use read(), read(no.of chars), readline() and readlines() then they read record values in the form of value by value and it is one of the time consuming process(multiple read operations).

=>To overcome this time consuming process, we must use the concept of Un-pickling.
=>The advantange of Un-pickling is that with single read operation, we can read entire record content from the file of secondary memory into the object of main memory.

=>Definition of Un-Pickling:

=>The process of reading or trasefering the enrite record content from file of secondary memory into the object of main memory by performing single read operation is called Un-pickling.

=>Un-Pickling concept participates in Read Operations.

```
---------------------------------------------------------------
Steps for implementing Un-Pickling  Concept:
----------------------------------------------------------------
=>import pickle module
=>Choose the file name and open it into read mode.
=>Use the load() of pickle module. load() is used for transfering or loading
  the entire  record content  from file of secondary memory into object of
  main memory.
           Syntax:     objname=pickle.load(filepointr)

--------------------------------------------------
1) By using csv.writer class object
---------------------------------------------------
=>The csv.writer class object is used to insert data to the CSV file.
=>To create an object of csv.writer class object, we use writer()  and
  present in csv module.
=>csv.writer class  object provides two Functions for writing to CSV file.
=>They are
              1) writerow()
              2) writerows()

1) writerow(): This Function writes a single row at a time.
               Field row / Header can also be written using this Function.
       Syntax:-  csvwriterobj.writerow(fields Row /  Data Row)
2) writerows(): This Function is used to write multiple rows at a time.
                This can be used to write rows list.
       Syntax:   Writing CSV files in Python
                     csvwriterobj.writerows(data rows)
              here data rows can be list, tuple set,frozenset only

---------------------------------------------------------------------------
2) By Using csv.DictWriter class object
-------------------------------------------------------------
=>The csv.DictWriter class object is used to insert dict data to the CSV
  file.
=>To create an object of csv.DictWriter class object, we use DictWriter()
  and present in csv module.
=>csv.DictWriter class  object provides two Functions for writing to CSV.
              1) writeheader()
              2) writerows()
------------------------
1) writeheader():
------------------------
 writeheader() method simply writes the first row of your csv file using the
 pre-specified fieldnames.
 Syntax:  DictWriterObj.writeheader()
---------------------------------------------------------
2) writerows():
------------------------
=>writerows() method simply writes all the values of (Key,Value) from dict
  object in the form of separate rows[ Note: it writes only the values(not
  keys) ]
Syntax:-     DictWriterObj.writerows(dictobject)
---------------------------------------------------------------------------
              Reading the data from  CSV File
---------------------------------------------------------------------------
=>There are various ways to read a CSV file that uses either the CSV module
  or the pandas library.
=>The csv Module provides two classes for reading  information from CSV file
                     1) csv.reader
                     2) csv.DictReader
```

----
1) csv.reader():

=>This Function is used for creating an object of csv.reader class and It helps us to read the data records from csv file.

=>Syntax:- csvreaderobj=csv.reader(filepointer)

----
2) csv.DictReader():

=>This Function is used for creating an object of csv.DictReader class and It helps us to read the data from csv file where it contains dict data(Key,Value).

=>Syntax:- csvdictreaderobj=csv.DictReader(filepointer)

==
OS Module
==

=>os is one of the pre-defined module in python
=>The purpose of 'os' module is that "To perform various OS Based Operations".
=>Some of the OS Based Operations are

```
                    a) obtaining current working folder----getcwd()
                    b) create a folder / Directory--------------mkdir()
                    c) Create Folder Hierarchy----------------makedirs()
                    d) delete folder / Directory----------------rmdir()
                    e) Delete  Folder Hierarchy---------------removedirs()
f) Renaming a Folder---------------rename("Old Folder Name","New Folder
   Name")
g) Finding Number of Files and Folder in Folder------listdir()
```

----
a) **obtaining current working folder**

=>To obtain current working folder, we use getcwd() of os module
=>Syntax:- varname=os.getcwd()

Examples:

```
#Program for obtaining current working folder
#currentworkfolder.py
import os
cw=os.getcwd()
print("Current Working Folder=",cw)
```

----
b) **create a Folder / Directory**

=>To create a Folder / Directory mkdir()
=>**Syntax:** os.mkdir("folder name")
=>If the folder name alerady exist then we get FileExistsError
=>mkdir() can create a folder at a time but not possible create Folders Hierarchy (RootFolder\Sub Folder\Sub-sub folder..etc)

Examples:
```
----------------------
#Program for creating a folder
#FolderCreate1.py
import os
try:
       os.mkdir("D:\KVR")
       print("Folder Created--verify")
except FileExistsError:
       print("Folder already exist:")
except FileNotFoundError:
       print("mkdir() can't create Folders Hierarchy:")
```
----

c) **Create Folders Hierarchy**:

=>To create Folders Hierarchy, we use makedirs()
=>**Syntax:-**   os.makedirs("Folder Hierarchy")
=>Here Folder Hierarchy represents RootFolder\Sub Folder\Sub-sub folder..etc.
=>If the Folder Hierarchy alerady exist then we get FileExistsError

----

Examples

```
-------------------
#Program for creating Folders Hierarchy
#FoldersCreate1.py
import os
try:
       os.makedirs("D:\INDIA\HYD\AMPT\PYTHON")
       print("Folders Hierarchy created--verify")
except FileExistsError:
       print("Folders Hierarchy already exist:")
```
----

d) **Delete a Folder / Directory**

=>To delete a Folder / Directory , we use rmdir()
=>**Syntax**:   os.rmdir("folder name")
=>If the folder name does not exist then we get FileNotFoundError
=>rmdir() can remove one folder at a time but not possible romve Folders Hierarchy (RootFolder\Sub Folder\Sub-sub folder..etc)
=>rmdir() can remove a folder when folder empty otherwise we get OSError.

Examples:
```
------------------
#Program for deleting a folder
#FolderRemove1.py
import os
try:
       os.rmdir("KVR")
       print("Folder Removed--verify")
except FileNotFoundError:
       print("Folder does not exist:")
except OSError:
       print("Folder is not empty:")
```
----

e) **Delete a Folders / Directory Hierarchy**

=>To delete Folders / Directory hierarchy we use removedirs()
=>**Syntax**:   os.removedirs("folders Hiererchy")
=>If the folders Hiererchy"   does not exist then we get FileNotFoundError
=>removedirs() can remove a folder hierarchy when folder empty otherwise we get OSError.

Examples:

```
------------------
#Program for Removing a folders
#FoldersRemove1.py
import os
try:
       os.removedirs("C:\AMEERPET\HYD")
       print("Folder Removed--verify")
except FileNotFoundError:
       print("Folder does not exist:")
except OSError:
       print("Folder is not empty:")
```

---

f) **Renaming a Folder**

=>To rename a folder, we use rename() of os module
=>**Syntax:-**   os.rename("Old Folder Name", "New folder name")
=>if Old Folder Name does not exist then we get FileNotFoundError

Examples:

```
------------------
#Program for re naming a folder
#FolderRename.py
import os
try:
       os.rename("C:\HYD","C:\AMEERPET")
       print("Folder Renamed--verify")
except FileNotFoundError:
       print("Folder name does not exist")
```

---

g) **Finding Number of Files and Folder in Folder-**

=>To list or find number of files in folderm we use listdir()
=>**Syntax:-**   listobj=os.listdir("folder name")
=>If the folder name does not exist then we get FileNotFoundError

Examples:

```
-----------------------
#Program for listing files of a  folders
#FilesInFolder.py
import os
try:
       foldername=input("Enter any folder name:")
       listobj=os.listdir(foldername)
       print("Number of Files :{}".format(len(listobj)))
       for file in listobj:
              print("\t{}".format(file))
except FileNotFoundError:
       print("Folder does not exist:")
```

==================================**====================================

==
Python DataBase Communication (PDBC)
==

=>Even we achieved the Data Persistency with Files concept, we have the following limitations.

1. Files Concept of any language does not contain security bcoz files concept does not contain user names and passwords.
2. To extract or process the data from files is very complex bcoz Files Bada must always processed with Indices.
3. The data of the files does not contain Column Names and Very complex to Process / Query the data.
4. Files are unable to store large Volume of Data.
5. The Architecture of Files Changes from One OS to another OS(OR) Files are Dependent on OS.

=>To Overcome the limitations of Files, we must use the concept of DataBase Softwares Which are purely RDBMS Products(Oracle, MySQL, MongoDB,SQLITE3, DB2,SQL SERVER...........)

=>When we use Data Base Softwares for acheving the data persistency, we get the following advantages.

1. DataBase Softwares are Fully Secured bcoz they provides User Name and password
2. To Process or Extract or Querying the data from DataBase Softwares is very easy bcoz the data present in tables of Database softwares are qualified with Column Names.
3. Data Base Software are able to store large Volume of Data.
4. Data Base Softwares are InDepeendent from OS.

=>If Python Program want to communicate with Any Database Software Product then we must use pre-defined modules and such pre-defined modules are not present in Python Library. So that Python Programmer must get / Install the pre-defined modules of Database Software by using a tool called pip.

=>To Make any Python Program to communicate with any data base software then we must install a third party module which is related Data base software.
=>For Example, To communicate with Oracle Database, we must install cx_Oracle Module, To communicate with MySQL data base , we must install mysql-connector.....etc and they must be installed explicitly by using pip Tool

=>Syntax: pip install   module name

=>here pip tool present in the following folder
"C:\Users\nareshit\AppData\Local\Programs\Python\Python310\Scripts "
=>If we get an error as " pip is not recognized internal command " then set the path as follows

=>C:\Users\nareshit>set
path="C:\Users\nareshit\AppData\Local\Programs\Python\Python310\Scripts "

```
Example:    intsall  cx_Oracle  any where from windows cmd prompt
                                 (not from Python cmd prompt)
```

```
--------------
                   pip   install   cx_Oracle

Example:    install  mysql-connector

                 pip  install   mysql-connector
```

===

Communication between Python Program and Oracle Database

===

=>In order to write python program to communicate with Oracle Database, we must follow 6 steps. They are

1. import cx_Oracle module
2. Python Program must get the connection from Oracle DB
3. Create an object Cursor
4. Python program must Prepare the Query and Execute the Query in Oracle DB
5. Python Program Process the Result of the Query.
6. Python Program closes the connection.

Explanation:

----

1. import cx_Oracle module:

=>If a python Program want to perform any database operations(insert , delete , update record , read records..etc) then we must import a pre-defined third party module "cx_Oracle".
=>We know that a module is a collection of Variables, Function Names and Class Names.

```
  Examples:     import  cx_Oracle
```

----

2. Python Program must get the connection from Oracle DB

=>To do any Data Base Operations, First python program must get the connection from Oracle.
=>To get the connection from any Database, we use connect() which is present in cx_Oracle module.
=>Syntax:-   varame=cx_Oracle.connect("Connection URL")

=>Here connection URL Represents " UserName/Password@DNS/Serviceid "

(OR)

UserName/Password@IPAddress/Serviceid "   "

=>Here Varname is an object of <class, cx_Oracle.Connection>
=>Here "UserName" represents User Name of Oracle Data Base (Ex: scott)
=>here "password" represents Password of Oracle Data Base (Ex: tiger)
=>here DNS(Domain Naming Service) represents name of the machine where Database Software Installed. The default Name of Every Machine is "localhost".
=>Here IPAddress (Internet Protocal Address) represents An address of Physical Machine where Database software Installed. The default IP Address of Every Machine is 127.0.0.1 (Loop back address)

=>Here "serviceid" represents on which name Oracle data base Installed in current working machine. To find Service Id in Oracle Data base, we use the following at SQL Environment

```
              SQL> select * from global_name;
                                      GLOBAL_NAME
                                      --------------------------------------
                                      ORCL <---------------Service id
```

=>When we use / write Invalid Connection URL then we get cx_Oracle.DatabaseError as an exception and must handle.

----

3. Create an object of Cursor

=>The purpose of creating an object of Cursor is that "To carry the Query from Python Program, hand over to Database, and obtains Result from Database and Gives to Python Program".
=>To create an object of Cursor, we use cursor() which is present in Connection Object.
=>Syntax: varname=conobject.cursor()
=>Here Varname reprsents an object of <class, cx_Oracle.Cursor>

----

4. Python program must Prepare the Query and Execute the Query in Oracle DB

=>A Query is a statement or Request or Question to database software for obtaining data base results.
=>To execute the query in any Database software, we use execute() which is present in cursor object.
=>Syntax: cursorobj.execute("Query")
=>Here Query is of type str and In any database software we have different Queries (DDL,DML,DRL)

----

5. Python Program Process the Result of the Query.

=>After Executing DML statements, the result of DML statements is present in cursor object. To extract the result from cursor object, we use "rowcount" attribute of cursor object. "rowcount" attribute gives number of updated / deleted / inserted in the data base.
=>After Executing DRL statements, the result of DRL statements is present in cursor object. To extract the from cursor object, have 3 Functions in cursor object. They are

a) fetchone()
b) fetchmany(no. of records)
c) fetchall()

=>fetchone() is used for obtaining One Record at a Time in the form of tuple. if no records found then we this function returns None.
=>fetchmany(no. of records) is used for obtaining specified number of records.

case-1: if specified number of records==0 then this function obtains all records
case-2: if specified number of records<=Total Number of Records then this function gives specified number of records
case-3: if specified number of records>Total Number of Records then this function obtains all records
case-4: if specified number of records<0 then this function never gives any records.

=>fetchall() is used for obtaining all the records from cursor object.

```
                    =============================================
                     Types of Queries  in Database Softwares
                    =============================================
=>In any database, we have 3 types of Queries. They are

                      1. DDL( Data Definition Language) Queries
                      2. DML (Data Manipulation Language) Queries
                      3. DRL (Data Retrieval Language ) Queries

                    ==============================================
                     1. DDL( Data Definition Language) Queries
                    ==============================================
=>The purpose of DDL Queries is that to deal with Physical Level of Database
  software such as creation of Tables, altering column sizes, adding new
  Columns etc.
=>In any Database software , we have 3 types of DDL Queries. They are

                      1. create
                      2. alter
                      3. drop
-----------------------------------------------------------------------------
1) create:
---------------------
=>This Query is used for creating a table in Oracle Database
=>Syntax:-
  SQL> create table <table-name> ( col name1  database data type, col name2
  database data type, ......col name-n  database data type )

  SQL> create table employee(eno  number(2) primary key ,ename varchar2(10)
  not null , sal number (6,2) not null);
-----------------------------------------------------------------------------
2. alter :
-------------------------------------
=>This Query is used for alter the table structure such as modifying
  (modify) the column sizes and adding (add) new columns.

  Syntax1:-   SQL> alter table <table-name> modify ( existing col name1
  database data type....  existing col name-n  database data type )

  Syntax2:-   SQL> alter table <table-name> add ( new col name1  database
  data type....  new  col name-n  database data type )

  Examples:   SQL> alter table employee add(cname varchar2(10));
                   SQL> alter table employee modify(ename varchar2(20), sal
                   number(8,2));
-----------------------------------------------------------------------------
3) drop :
-----------------------------------------------------------------------------
=>This query is used for removing the table from Database Software
=>Syntax:-    SQL> drop table <table-name>
=>Example:- SQL> drop table employee Inserting the records, deleting the
  records and updating the records.
```

==
2. DML (Data Manipulation Language) Queries
==

=>The purpose of DML operations is that To manipulate the table such Inserting the records, deleting the records and updating the records.
=>In RDBMS database softwares, we have 3 types of DML Operations. They are

1. insert
2. delete
3. update

=>When we execute any DML Operation through python program, we must use commit() for permanent change / update / modification and to roll back we use roolback().
=>commit() and rollback() are present in connection object.

---

1. insert:

=>This query is used for inserting a record in table of database.
=>Syntax:-   SQL> insert into <table-name> values(val1 for column1, val2 for column2.....

val-n f

```
Example:    SQL> insert into student values (20,'DR',33.45,'C');
            SQL>insert into student values (10,'RS',23.45,'Python');
            SQL> commit ;
```

---

2. delete

=>This query is used for deleting a record .
=>Syntax1:   delete from <table name>

=>Syntax2:   delete from <table-name> where cond list;

=>Example: SQL> delete from student where sno=70;

---

3. update

=>This query is used for updating the record values

=>Syntax1: SQL> update   <table-name> set col1=val1,col2=val2.....col-n=val-n;

=>Syntax2:   SQL> update   <table-name> set col1=val1,col2=val2.....col-n=val-n where cond list;

```
Examples:  SQL> update student set marks=marks+marks*0.02;
Examples:  SQL> update student set marks=marks+marks*0.05,crs='Django' where
           sno=90;
```

```
              ==============================================
                 3. DRL (Data Retrieval Language ) Queries
              ==============================================
=>DRL (Data Retrieval Language ) Queries are used for Reading the records
  from table.
=>To read the records from table, we use "select"
=>In Otherwords "select" comes under DRL (Data Retrieval Language ) Query.
=>Syntax1:       SQL>select  col1,col2,.....col-n   from <table-name>
=>Syntax2:       SQL>select  col1,col2,.....col-n   from <table-name> where
                 cond list
=>Syntax3:       SQL>select  *   from <table-name>
=>Syntax4:       SQL>select  *   from <table-name>  where cond list
---------------------------------------------------------------------------
=>Once the select query executed, all records are present in the object of
  cursor in Python.
=>To get the records from cusror object, we have 3 functions. They are
1) fetchone()
2) fetchmany(no. of records)
3) fetchall()
----------------------------------------------------------------------
1) fetchone():
=>This function is used obtaining One Record at a time in the form of tuple
  whereeever cursor object pointing
---------------------------------------------------------------------------
2) fetchmany(no. of records)
---------------------------------------------------------------------------
=>fetchmany(no. of records) is used for obtaining specified number of
  records.
       case-1: if specified number of records==0 then this function obtains
               all records
       case-2: if specified number of records<=Total Number of Records
               then this function gives specified number of records
       case-3: if specified number of records>Total Number of Records then
               this function obtains all records
       case-4: if specified number of records<0 then this function  never
              gives any records.
---------------------------------------------------------------------------
3) fetchall()
---------------------------------------------------------------------------
=>fetchall()  is used for obtaining all the records from cursor object.
```

===
Communication between Python Program and MySQL Database
===

=>In order to write python program to communicate with MySQL Database, we must follow 6 steps. They are

1. import mysql.connector module
2. Python Program must get the connection from Oracle DB
3. Create an object Cursor
4. Python program must Prepare the Query and Execute the Query in Oracle DB
5. Python Program Process the Result of the Query.
6. Python Program closes the connection.

Explanation:

---

1. import mysql.connector module:

=>If a python Program want to perform any database operations(insert , delete , update record , read records..etc) then we must import a pre- defined third party module "mysql-connector ".
=>We know that a module is a collection of Variables, Function Name and Class Names.

```
Examples:     import   mysql-connector
```

---

2. Python Program must get the connection from MySQL DB

=>To do any Data Base Operations, First python program must get the connection from MySQL.
=>To get the connection from any Database, we use connect() which is present in mysql.connector module.
=>Syntax:-   varame=mysql.connector.connect("Connection URL")

=>Here connection URL Represents :   host="DNS" , user="User Name" passwd="password"

```
             con=mysql.connector.connect(host="DNS" ,
                                         user="User Name",
                                         passwd="password" )
Examples:    con=mysql.connector.connect(host="localhost" ,
                                          user="root",
                                          passwd="root" )
```

=>Here Varname is an object of <class, mysql.connector.Connection>
=>Here "UserName" represents User Name of MySQL Data Base (Ex: root)
=>here "passwd" represents Password of MySQL Data Base (Ex: root)
=>here DNS(Domain Naming Service) represents name of the machine where DatabaseSoftware Installed. The default Name of Every Machine is "localhost".
=>Here IPAddress (Internet Protocal Address) represents An address of Physical Machine where Database software Installed. The default IP Address of Every Machine is 127.0.0.1 (Loop back address)
=>When we use / write Invalid Connection URL then we get mysql.connector.DatabaseError as an exception and must handle.

---

3. Create an object of Cursor

=>The purpose of creating an object of Cursor is that "To carry the Query from Python Program, hand over to Database, and obtains Result from Database and Gives to Python Program".
=>To create an object of Cursor, we use cursor() which is present in Connection Object.
=>Syntax: varname=conobject.cursor()

=>Here Varname reprsents an object of <class, mysql.connector.Cursor>

---

4. Python program must Prepare the Query and Execute the Query in MySQL DB

=>A Query is a statement or Request or Question to database software for obtaining data base results.
=>To execute the query in any Database software, we use execute() which is present in cursor object.
=>Syntax: cursorobj.execute("Query")
=>Here Query is of type str and In any database software we have different Queries (DDL,DML,DRL)

---

5. Python Program Process the Result of the Query.

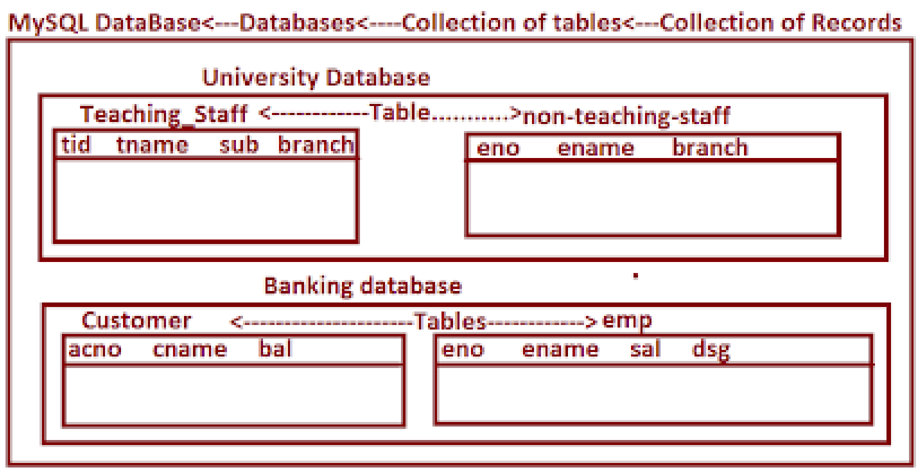
=>After Executing DML statements, the result of DML statements is present in cursor object. To extract the result from cursor object, we use "rowcount" attribute of cursor object. "rowcount" attribute gives number of updated / deleted / inserted in the data base.
=>After Executing DRL statements, the result of DRL statements is present in cursor object. To extract the from cursor object, have 3 Functions in cursor object. They are

a) fetchone()
b) fetchmany(no. of records)
c) fetchall()

=>fetchone() is used for obtaining One Record at a Time in the form of tuple. if no records found then we this function returns None.
=>fetchmany(no. of records) is used for obtaining specified number of records.

case-1: if specified number of records==0 then this function obtains all records
case-2: if specified number of records<=Total Number of Records then this function gives specified number of records
case-3: if specified number of records>Total Number of Records then this function obtains all records
case-4: if specified number of records<0 then this function never gives any records.

=>fetchall() is used for obtaining all the records from cursor object.


MySQL DataBase<---Databases<----Collection of tables<---Collection of Records
University Database
Teaching_Staff <------------Table...........>non-teaching-staff
tid tname sub branch
eno ename branch
Banking database
Customer <---------------------Tables------------> emp
acno cname bal
eno ename sal dsg

```
==================================**=====================================
                ========================================
                          Regular Expressions
                ========================================
=>Regular Expressions is one of the Programming Languages Indepedent
  Concept.
=>The purpose of Regular Expressions in any programming language is that "
  To Perform Data  Validation (Validating End User Inputs) and build Robust
  Application."
--------------------------------------------------------------
Application of Regular Expressions
--------------------------------------------------------------
=>Development of Language Compilers and Interpreters.
=>Development of Operating Systems.
=>Development of pattern matching Applications
=>Development of Electronic Circuit Designing Applications
=>Development of Universal Protocols.
=>......etc
----------------------------------------------------------------------------
Definition of Regular Expression:
----------------------------------------------------------------------------
A Regular Expression is one of the String Pattern which is the combination
of Alphabets,Digits and Sppecial Symbols which is used to search or match or
finding  in Given Data and obtains desired result.

                  ==========================================
                     Pre-defined Functions in re module
                  ==========================================
=>The 're' module contains the follwing essential Functions.
----------------------------------------------------------------------------
1) finditer():
----------------------------------------------------------------------------
Syntax:-      varname=re.finditer("search-pattern","Given data")
=>here varname is an object of type <class,'Callable_Itetaror'>

 =>This function is used for searching the "search pattern" in given data
```

iteratively and it returns table of entries which contains start index , end index and matched value based on the search pattern.

---

2) group():

=>This function is used obtaining matched value by the findIter()
=>This function present in match class of re module
```
       Syntax:-     varname=matchtabobj.group()
```

---

3) start():

=>This function is used obtaining starting index of matched value
=>This function present in match class of re module
```
       Syntax:     varname=matchobj.start()
```

---

4) end():

=>This function is used obtaining end index+1 of matched value
=>This function present in match class of re module
```
       Syntax:     varname=matchobj.end()
```

---

5) search():

```
Syntax:-      varname=re.search("search-pattern","Given data")
```
=>here varname is an object of <class,'re.match'> or <class,'NoneType'>

=>This function is used for searching the search pattern in given data for first occuence / match only but not for other occurences / matches.
=>if the search pattern found in given data then it returns an object of match class which contains matched value and start and end index values and it indicates search is successful.
=>if the search pattern not found in given data then it returns None which is type <class, "NoneType"> and it indicates search is un-successful

---

6) findall():

```
Syntax:-      varname=re.findall("search-pattern","Given data")
```
=>here varname is an object of <class,'list'>
=>This function is used for searching the search pattern in entire given data and find all occurences / matches and it returns all the matched values in the form an object <class,'list'> but not returning Start and End Index Values.

Programmer-Defined Character Classes

=>Programmer-Defined Character Classes are defined by Programmers for designing Search Pattern and it can be used to search or match or find in given data and obtains desired result.
=>Syntax: "[Search Pattern] "
=>The following are the Programmer-Defined Character Classes.

```
1.  [abc]      --------->Searches for Either 'a' or 'b' or 'c' only
2.  [^abc]--------->Searches for all except a' or 'b' or 'c
3.  [a-z]------------>Searches for all Lower case  Alphabets only
4.  [^a-z]------------>Searches for all  except Lower case  Alphabets
5.  [A-Z]------------->Searches for all Upper case  Alphabets only
6.  [^A-Z]------------->Searches for all  except Upper case  Alphabets
7.  [0-9]-------------->Searches for all Digits only
8.  [^0-9]-------------->Searches for all  except Digits .
9.  [A-Za-z]----------->Searches for all Alphabets (Upper and Lower) only
10.[^A-Za-z]----------->Searches for all  except Alphabets (Upper and Lower)
```

```
11. [A-Za-z0-9]------>Searches for Alpha-Numeric Values (Alphabets +Digits)
only
12. [^A-Za-z0-9]------>Searches for Special Symbols  Except Alpha-Numeric
Values (Alphabets .+Digits) only

13. [A-Z0-9]-----Searhes for Upper Case Alpabets and Digits only
14. [^A-Z0-9]-----Searhes for all except Upper Case Akpabets and Digits
15. [a-z0-9]-----Searhes for Lower Case Alpabets and Digits only
16. [^a-z0-9]-----Searhes for all except lower Case Alpabets and Digits
```

===

Quantifiers in Regular Expressions

===

=>Quantifiers in Regular Expressions are used for searching number of occurences of the specified value (alphabets or digits or special symbols) used in search pattern to search in the given data and obtains desired result.

```
1)  "k"------->It search for only one 'k' at a time
2) "k+"------>It search for either one 'k' more 'k' s
3) "k*"------>It search for either zero 'k'  or one 'k'  and  more 'k' s
4) "k?"----->>It search for either zero 'k'  or one 'k'
5)  ". " ---->It searches for all
```

Note

```
\ddd  or   \d{3}----->searches for 3 digit Number
\dd.\dd----searhes for 2 integer values and 2 decimal values  OR \d{2}.\d{2}
\d{2,4}----searches for min 2 digit number and max 4 digit number.
[A-Za-z]+ ---searches one alphabet   or More alphabets.
\w+
```

==

random module

==

=>'random' is one of pre-defined module
=>The purpose of random module is that "To generate random values".
=>random module contains the following pre-defined Functions.

```
                             1) randrange()
                             2) randint()
                             -------------------------
                             3) random()
                             4) uniform()
                             -------------------------
                             5)choice()
                             6) shuffle()
                             7)sample()
                             --------------------------
```

----

1) randrange():

```
=>Syntax1:           varname=randrange(Value)
                                   (OR)
                             randrange(Value)
=>This Syntax generates random Integer Values from 0 to Value-1

=>Syntax2:           varname=randrange(Start,Stop)
                                   (OR)
                             randrange(Start,Stop)
=>This Syntax generates random Integer Values from Start to Stop-1
```

Examples:

```
-------------------
>>> import random as r
>>> print(r.randrange(10))-----------3
>>> print(r.randrange(10))-----------8
>>> print(r.randrange(10))----------3
>>> print(r.randrange(10))----------5
>>> print(r.randrange(10))-------7
>>> print(r.randrange(10))---------2
>>> print(r.randrange(10,20))--------19
>>> print(r.randrange(10,20))----------10
>>> print(r.randrange(10,20))--------12
>>> print(r.randrange(10,20))---------19
>>> print(r.randrange(10,20))--------16
>>> print(r.randrange(10,20))-------11
>>> print(r.randrange(1000,10000))-----7254
>>> print(r.randrange(1000,10000))-----6283
>>> print(r.randrange(1000,10000))------7713
>>> print(r.randrange(1000,10000))----3593
>>> print(r.randrange(1000,10000))----2564
```

ExampleS:

```
#Program form randrange()
#randrangeex.py
import random as r
for i in range(1,8):
       print(r.randrange(1000,2000))
```

----

2) randint()

```
=>Syntax2:          varname=randint(Start,Stop)
                                (OR)
                         randint(Start,Stop)
```

=>This Syntax generates random Integer Values from Start to Stop(Both are Inclusive)

Examples:

```
>>> import random as r
>>> print(r.randint(10,15))--------13
>>> print(r.randint(10,15))----------10
>>> print(r.randint(10,15))--------14
>>> print(r.randint(10,15))--------15
>>> print(r.randint(10,15))--------11
```

Examples:

```
#Program form randint()
#randintex.py
import random as r
for i in range(1,6):
       print(r.randint(100,150))
```

NOTE: here randrange() and randint() are used for generating random Integer values only

----

3) random()

```
Syntax:        random.random()
```

=>This syntax generates random floating values between 0.0 to 1.0 (Exclusive)

Examples:

```
>>> import random as r
>>> print(r.random())----------------0.953414908971505
>>> print(r.random())---------------0.8602010668984944
>>> print(r.random())---------------0.6652750407642519
>>> print(r.random())------------0.8657790572994452
>>> print("%0.2f" %r.random())-----------0.62
>>> print("%0.2f" %r.random())----------0.52
>>> print("%0.2f" %r.random())----------0.48
>>> print("%0.2f" %r.random())-----------0.99
Examples:
--------------------------------
#Program for random()
#randomex.py
import random as r
for i in range(1,6):
       print(r.random())
print("-------------------------------")
for i in range(1,6):
       print("%0.3f"  %r.random())
-----------------------------------------------------------------------------
```

4) uniform()

```
-----------------------------------------------------------------------------
=>Syntax:     random.uniform(Start,Stop)
=>This Syntax generates ra ndom floating point values from Start to Stop-1.
=>Here start and stop values can be either Integer or floating point values
Examples:
-------------------
>>> import random as r
>>> print(r.uniform(2,3))-------2.125910222609465
>>> print(r.uniform(2,3))--------2.5724588835923488
>>> print(r.uniform(2,3))----2.315566668246915
>>> print(r.uniform(2,3))------2.361772917954831
>>> print(r.uniform(2,3))-----2.183749376668188
>>> print(r.uniform(100,300))------255.9741762891575
>>> print(r.uniform(100,300))-------186.53282469558758
>>> print(r.uniform(1.5,2.5))-------2.4826601405944295
>>> print(r.uniform(1.5,2.5))------1.6289136361161278
>>> print(r.uniform(1.5,2.5))-------1.679501275355531
Examples:
------------------
#Program for unitorm()
#uniformex.py
import random as r
for i in range(1,6):
       print(r.uniform(2,3.5))
print("-------------------------------")
for i in range(1,6):
       print("%0.2f"  %r.uniform(100,200))
-----------------------------------------------------------------------------
```

5)choice()

```
-----------------------------------------------------------------------------
=>Syntax:-     random.choice(Iterableobject)

Examples:
---------------
>>> import random as r
>>> s="PYTHON"
>>> r.choice(s)
'T'
>>> r.choice(s)
'O'
>>> r.choice(s)
'O'
```

```
>>> r.choice(s)
'N'
>>> r.choice(s)
'N'
>>> r.choice(s)
'N'
>>> print(r.choice(s),r.choice(s))
N H
>>> print(r.choice(s),r.choice(s))
Y H
>>> print(r.choice(s),r.choice(s))
N H
>>> print(r.choice(s),r.choice(s))
Y N
>>> print(r.choice(s),r.choice(s))
T P
>>> s="ABCD12345efghikl@#$%^&*"
>>> print(r.choice(s),r.choice(s),r.choice(s),r.choice(s))
4 # A B
>>> print(r.choice(s),r.choice(s),r.choice(s),r.choice(s))
B A e
>>> print(r.choice(s),r.choice(s),r.choice(s),r.choice(s))
h @ 3 &
>>> print(r.choice(s),r.choice(s),r.choice(s),r.choice(s))
i 3 e 3
>>> print(r.choice(s),r.choice(s),r.choice(s),r.choice(s))
4 * A &
----------------------------------------------------------------------
Example:
---------------
#Program for choice()
#choiceex2.py
import random as r
nums="0123456789"
alpha1="ABCEDEFGHIJKLMNOPQRSTUVWXYZ"
alpha2="ABCEDEFGHIJKLMNOPQRSTUVWXYZ"
for i in range(1,11):
       print("TS"+r.choice(nums)+r.choice(nums)+r.choice(alpha1)+r.choice(a
lpha2)+r.choice(nums)+r.choice(nums)+r.choice(nums)+r.choice(nums))
print("\n-------------------------------------")
for i in range(1,11):
       print("TS08"+r.choice(alpha1)+r.choice(alpha2)+r.choice(nums)+r.choi
ce(nums)+r.choice(nums)+r.choice(nums))
----------------------------------------------------------------------------
```

6) shuffle()

```
----------------------------------------------------------------------------
=>This function is use for re-organizing the elements of Mutable objects
only (Immutable objects data is not possible to shuffle)
=>Syntax:    random.shuffle(object)
=>Here  object  can be mutable.
Examples:
------------------
>>> l1=[10,20,30,40,50]
>>> r.shuffle(l1)
>>> l1---------------[30, 20, 40, 50, 10]
>>> r.shuffle(l1)
>>> l1--------------[10, 30, 40, 50, 20]
>>> r.shuffle(l1)
>>> l1-------------[10, 20, 40, 50, 30]
-----------------------------
>>> l1=[10,20,30,40,50]
>>> l1
```

```
[10, 20, 30, 40, 50]
>>> r.shuffle(l1)
>>> l1
[20, 40, 30, 10, 50]
>>> print(r.shuffle(l1))----------None
>>> print(l1)
[40, 10, 20, 30, 50]
```

----

7)sample()

=>This Function is used for selecting k number of sample random elements from given Iterable object.
=>Syntax:-   varname= random.sample(object,k)
=>here Varname is an object of list.
=>Here k represents Number of Samples to be selected randomly from Iterable object.
Examples:

```
>>> l1=[10,"KVR",34.56,"Python",4.5,True]
>>> x=r.sample(l1,3)
>>> x
[34.56, 4.5, True]
>>> x=r.sample(l1,3)
>>> x
['Python', 4.5, True]
>>> r.sample(l1,3)
[True, 4.5, 'KVR']
>>> r.sample(l1,3)
[10, 'Python', 'KVR']
>>> r.sample(l1,3)
[10, 4.5, 'Python']
>>> r.sample(l1,3)
['Python', 4.5, True]
>>> r.sample(l1,3)
[4.5, 'KVR', True]
>>> s1="PYTHON"
>>> r.sample(s1,2)
['P', 'N']
>>> r.sample(s1,2)
['T', 'P']
>>> r.sample(s1,2)
['Y', 'O']
>>> r.sample(s1,2)
['O', 'H']
```

Example

```
#Program for sample()
#sampleex.py
import random as r
s="ABCDEFGHI123456@#$%^&*()abcd"
for i in range(1,6):
        x=r.sample(s,6)
        s1=""
        for val in x:
                s1=s1+val
        print(s1)
```

==

Numpy

==

Introduction to Numpy:

=>Numpy stands for Numerical Python.
=>Numpy is one of the pre-defined third party module / Library and numpy module is not a pre-defined module in Python Language.
=>To use numpy as a part of our python program, we must install numpy module explicitly by using a tool called pip and it present in (C:\Users\nareshit\AppData\Local\Programs\Python\Python39\Scripts)
=>Syntax for installing any module:

```
pip    install    module-name
```

=>Example: Install numpy module

```
pip   install   numpy
```

=>To use numpy as part of our program, we must import numpy module.
=>A Numpy module is a collection of Variables, Functions and Classes.

==

History of Numpy:

=>Numpy was developed by studying existing module called "Numeric module is not a pre-defined module in Python Language.
=>The Numeric Library was developed by JIM HUNGUNIAN
=>The Numeric Library was not able to solve complex maths calculations.
=>Numpy module developed by TRAVIS OLIPHANT
=>Numpy Module developed in the year 2005
=>Numpy Module developed in C and PYTHON languages.

==

Advantages of using NumPy

==

Need of NumPy:

=>With the revolution of data science, data analysis libraries like NumPy, SciPy, Scikit, Pandas, etc. have seen a lot of growth. With a much easier syntax than other programming languages, python is the first choice language for the data scientist.
=>NumPy provides a convenient and efficient way to handle the vast amount of data. NumPy is also very convenient with Matrix Operations and data reshaping. NumPy is fast which makes it reasonable to work with a large set of data.

The advantages of Numpy Programming are:

----

1) With Numpy Programming, we can deal with Arrays such 1-D, 2-D and Multi Dimensional Arrays.
2) NumPy maintains minimal memory for large sets of data:
3) Numpy provides Fast in Performing Operations bcoz internally its data is available at same address.
4) NumPy performs array-oriented computing.
5) It efficiently implements the multidimensional arrays.
6) It performs scientific computations.
7) It is capable of performing reshaping the data stored in multidimensional arrays.
8) NumPy provides Many in-built functions for Various Complex Mathematical Operations such as statistical , financial, trigonometric Operations etc.

==
Python Traditional List VS Numpy Module
==

Similarities of python Traditional List VS Numpy Module:

=>An object of list used to store multiple values of same type or different type and both types (unique +duplicates) in single object.
=>In Numpy Programming, the data is organized in the object of "ndarray", which is one of the pre-defined class in numpy module. Hence an object of ndarray can store same type or different type and both types (unique +duplicates) in single object.
=>The objects of ndarray and list are mutable (changes can takes place)

Differences between Python Traditional List and ndarray object of Numpy Module:

=>An object of list contains both homogeneous and hetrogeneous values where as an object of ndarray of numpy can store only similar type of values(even we store different values, internally they are treated as similar type by treating all values of type "object").
=>On the object of list, we can't perform Vector Operations. where as on the object of ndarray, we can perform Vector based operations.
=>In large sampling of data, List based applications takes more memory space where ndarray object takes less memory space.
=>List based applications are not effiecient bcoz list object values takes more time to extract or retrive (they are available at different Address) where as numpy based applications are efficient bcoz of ndarray object values takes less to time to extract or retrive(they are available at Address).
=>List object can't perform complex mathematical operations where as an object of ndarray can perform complex mathematical operations.

==
Number of approaches to create an object of ndarray
==

=>In numpy programming, we have 6 approaches to create an object of ndarray. They are

1. array()
2. arange()
3. zeros()
4. ones()
5. full()
6. identity()

----

1) array():

=>This Function is used for converting Traditional Python Objects into ndrray object.
=>Syntax:- varname=numpy.array(Object,dtype)
Here var name is an object of <class,ndarray>
here array() is pre-defined function of numpy module used for converting Traditional Python Objects into ndrray object.
object represents any Traditional Python Objects
dtype represents any numpy data type such as int8,int16,int32,float16, float 32, float64,....etc

Examples:

```
------------------
>>> import numpy as np
>>> l1=[10,20,30,40,50,60]
>>> print(l1,type(l1))-----------------[10, 20, 30, 40, 50, 60] <class
'list'>
>>> a=np.array(l1)
>>> print(a,type(a))----------------[10 20 30 40 50 60] <class
'numpy.ndarray'>
>>> t=(10,20,30,40,50,60,70)
>>> print(t,type(t))--------------(10, 20, 30, 40, 50, 60, 70) <class
'tuple'>
>>> a=np.array(t)
>>> print(a,type(a))--------------[10 20 30 40 50 60 70] <class
'numpy.ndarray'>
>>> d1={10:1.2,20:4.5,30:6.7}
>>> a=np.array(d1)
>>> a----array({10: 1.2, 20: 4.5, 30: 6.7}, dtype=object)
-----------------------------------------------------------------------------
>>> t=(10,20,30,40,50,60)
>>> a=np.array(t)
>>> a--------------array([10, 20, 30, 40, 50, 60])
>>> a.ndim------------1
>>> a.dtype----------dtype('int32')
>>> a.shape-------------(6,)
>>> b=a.reshape(3,2)
>>> c=a.reshape(2,3)
>>> b--------------
              array([[10, 20],
                     [30, 40],
                     [50, 60]])
>>> c
       array([[10, 20, 30],
              [40, 50, 60]])
>>> print(b,type(b))
              [[10 20]
               [30 40]
               [50 60]] <class 'numpy.ndarray'>
>>> print(c,type(c))
                     [[10 20 30]
                      [40 50 60]] <class 'numpy.ndarray'>
>>> b.ndim-------------2
>>> c.ndim------------2
>>> b.shape---------------(3, 2)
>>> c.shape-------------(2, 3)
>>> d=a.reshape(3,3)-------ValueError: cannot reshape array of size 6 into
shape (3,3)
-----------------------------------------------------------------------------
>>> t1=((10,20),(30,40))
>>> print(t1,type(t1))--------------((10, 20), (30, 40)) <class 'tuple'>
>>> a=np.array(t1)
>>> a
              array([[10, 20],
                     [30, 40]])
>>> a.ndim----------2
>>> a.shape----------(2, 2)
-----------------------------------------------------------------------------
```

```
>>> t1=( ((10,20,15),(30,40,25)),( (50,60,18),(70,80,35) ))
>>> print(t1,type(t1))
(((10, 20, 15), (30, 40, 25)), ((50, 60, 18), (70, 80, 35))) <class 'tuple'>
>>> a=np.array(t1)
>>> a
array([[[10, 20, 15],
        [30, 40, 25]],

       [[50, 60, 18],
        [70, 80, 35]]])
>>> print(a)
[[[10 20 15]
  [30 40 25]]

 [[50 60 18]
  [70 80 35]]]
>>> a.ndim
3
>>> a.shape
(2, 2, 3)
>>> b=a.reshape(4,3)
>>> b
array([[10, 20, 15],
       [30, 40, 25],
       [50, 60, 18],
       [70, 80, 35]])
>>> c=a.reshape(3,4)
>>> c
array([[10, 20, 15, 30],
       [40, 25, 50, 60],
       [18, 70, 80, 35]])
>>> d=a.reshape(3,2,2)
>>> d
array([[[10, 20],
        [15, 30]],

       [[40, 25],
        [50, 60]],

       [[18, 70],
        [80, 35]]])
>>> d[0]
array([[10, 20],
       [15, 30]])
>>> d[1]
array([[40, 25],
       [50, 60]])
>>> d[2]
array([[18, 70],
       [80, 35]])
>>>
```

```
----------------------------------------------------------------------------
2. arange():
----------------------------------------------------------------------------
Syntax1:-  varname=numpy.arange(Value)
Syntax2:-  varname=numpy.arange(Start,Stop)
Syntax3:-  varname=numpy.arange(Start,Stop,Step)
       =>Here var name is an object of <class,ndarray>
```

=>Syntax-1 creates an object of ndarray with the values from 0 to value-1

=>Syntax-2 creates an object of ndarray with the values from Start to Stop-1
=>Syntax-3 creates an object of ndarray with the values from Start to Stop-1 with equal
Interval of Value-----step
=>arange() always create an object of ndarray in 1-D array only but not Possible to create directly 2-D and Multi Dimesional Arrays.
=>To create 2-D and Multi Dimesional Arrays, we must use reshape()

Examples:
```
-----------------
>>> import numpy as np
>>> a=np.arange(10)
>>> a-----------array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
>>> a.ndim--------1
>>> a=np.arange(50,62)
>>> print(a,type(a))---[50 51 52 53 54 55 56 57 58 59 60 61] <class
'numpy.ndarray'>
>>> a.ndim------1
>>> a=np.arange(10,23,2)
>>> a-----array([10, 12, 14, 16, 18, 20, 22])
>>> a=np.arange(10,22,2)
>>> a--------array([10, 12, 14, 16, 18, 20])
>>> b=a.reshape(2,3)
>>> c=a.reshape(3,2)
>>> b-----
              array([[10, 12, 14],
                     [16, 18, 20]])
>>> c
              array([[10, 12],
                     [14, 16],
                     [18, 20]])
>>> b.ndim------ 2
>>> c.ndim------- 2
>>> b.shape-----(2, 3)
>>> c.shape-----(3, 2)
>>> l1=[ [[10,20],[30,40]], [[15,25],[35,45]] ]
>>> l1----------[[[10, 20], [30, 40]], [[15, 25], [35, 45]]]
>>> a=np.arange(l1)----------TypeError: unsupported operand type(s) for -:
'list' and 'int'
==============================================================
```
3. zeros():

=>This Function is used for building ZERO matrix either with 1-D or 2-D or n-D
=>Syntax: varname=numpy.zeros(shape,dtype)

=>Here Shape can be 1-D(number of Zeros) or 2-D(Rows,Cols) or n-D(Number of Matrices,Number of Rows, Number of Columns)

Examples:
```
--------------------
>>> import numpy as np
>>> a=np.zeros(12)
>>> a-----------array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
>>> a=np.zeros(12,dtype=int)
>>> a------------array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
>>> a.reshape(3,4)
                     array([[0, 0, 0, 0],
                            [0, 0, 0, 0],
                            [0, 0, 0, 0]])
```

```
>>> a.reshape(4,3)
                      array([[0, 0, 0],
                             [0, 0, 0],
                             [0, 0, 0],
                             [0, 0, 0]])
>>> a.reshape(6,2)
                      array([[0, 0],
                             [0, 0],
                             [0, 0],
                             [0, 0],
                             [0, 0],
                             [0, 0]])
>>> a.reshape(2,6)
              array([[0, 0, 0, 0, 0, 0],
                     [0, 0, 0, 0, 0, 0]])
>>> a.reshape(2,3,2)
                             array([[[0, 0],
                                     [0, 0],
                                     [0, 0]],

                                    [[0, 0],
                                     [0, 0],
                                     [0, 0]]])

>>> a.reshape(2,2,2,2)------ValueError: cannot reshape array of size 12 into
shape (2,2,2,2)
>>> a.reshape(3,2,2)
                      array([[[0, 0],
                              [0, 0]],

                             [[0, 0],
                              [0, 0]],

                             [[0, 0],
                              [0, 0]]])
>>> a.reshape(2,3,2)
                      array([[[0, 0],
                              [0, 0],
                              [0, 0]],

                             [[0, 0],
                              [0, 0],
                              [0, 0]]])
>>> a.reshape(2,2,3)
                      array([[[0, 0, 0],
                              [0, 0, 0]],

                             [[0, 0, 0],
                              [0, 0, 0]]])
-----------------------------------------------------------------------------
>>> import numpy as np
>>> a=np.zeros((3,3),dtype=int)
>>> a
              array([[0, 0, 0],
                     [0, 0, 0],
                     [0, 0, 0]])
>>> a=np.zeros((2,3))
>>> a
              array([[0., 0., 0.],
                     [0., 0., 0.]])
>>> a=np.zeros((2,3),int)
>>> a
              array([[0, 0, 0],
                     [0, 0, 0]])
>>> a=np.zeros((3,2,3),dtype=int)
>>> a
              array([[[0, 0, 0],
                     [0, 0, 0]],

                     [[0, 0, 0],
                     [0, 0, 0]],

                     [[0, 0, 0],
                     [0, 0, 0]]])
>>> print(a,type(a))
                     [[[0 0 0]
                       [0 0 0]]

                      [[0 0 0]
                       [0 0 0]]

                      [[0 0 0]
                       [0 0 0]]]     <class 'numpy.ndarray'>
```
----

4. ones()

=>This Function is used for building ONEs matrix either with 1-D or 2-D or n-D

=>Syntax: varname=numpy.ones(shape,dtype)

=>Here Shape can be 1-D(number of Zeros) or 2-D(Rows,Cols) or n-D(Number of Matrices,Number of Rows, Number of Columns)

Examples:

```
>>> import numpy as np
>>> a=np.ones(10)
>>> print(a,type(a))----------[1. 1. 1. 1. 1. 1. 1. 1. 1. 1.] <class
'numpy.ndarray'>
>>> a=np.ones(10,dtype=int)
>>> print(a,type(a))-------------[1 1 1 1 1 1 1 1 1 1] <class
'numpy.ndarray'>
>>> a.shape-----------(10,)
>>> a.shape=(5,2)
>>> a
       array([[1, 1],
              [1, 1],
              [1, 1],
              [1, 1],
              [1, 1]])
>>> a.ndim--------------     2
>>> a.shape------------ (5, 2)
>>> a.shape=(2,5)
>>> a
       array([[1, 1, 1, 1, 1],
              [1, 1, 1, 1, 1]])
>>> a.shape----------------------(2, 5)
>>>
>>> a=np.ones((3,4),dtype=int)
>>> a
              array([[1, 1, 1, 1],
                     [1, 1, 1, 1],
                     [1, 1, 1, 1]])
```

```
>>> a=np.ones((4,3),dtype=int)
>>> print(a,type(a))
                      [[1 1 1]
                       [1 1 1]
                       [1 1 1]
                       [1 1 1]] <class 'numpy.ndarray'>
>>> a.shape----------(4, 3)
>>> a.shape=(3,2,2)
>>> a
              array([[[1, 1],
                      [1, 1]],

                      [[1, 1],
                      [1, 1]],

                      [[1, 1],
                      [1, 1]]])
>>> a=np.ones((4,3,3),dtype=int)
>>> a
                      array([[[1, 1, 1],
                             [1, 1, 1],
                             [1, 1, 1]],

                             [[1, 1, 1],
                             [1, 1, 1],
                             [1, 1, 1]],

                             [[1, 1, 1],
                             [1, 1, 1],
                             [1, 1, 1]],

                             [[1, 1, 1],
                             [1, 1, 1],
                             [1, 1, 1]]])

>>> a[0][0][0]-----------1
>>> a[0,0,0]-----------1
>>> a[0][0,0]------------1
================================================================
```

5) full()

=>This is function is used for building a matrix by specifying fill value either 1-D or 2-D or n-D

=>Syntax:-

```
                      varname=numpy.full(shape,fill_value,dtype)
```

=>varname is an obejct of <class, numpy.ndarray>

=>Here Shape can be 1-D(number of Zeros) or 2-D(Rows,Cols) or n-D(Number of Matrices,Number of Rows, Number of Columns)

=>fill_value can be any number of programmer choice

Examples:

```
>>> a=np.full(3,1)
>>> a---------array([1, 1, 1])
>>>print(type(a))--------<class,numpy.ndarray>
>>> a=np.full(3,9)
>>> a------------array([9, 9, 9])
>>> a=np.full(6,8)
>>> a-------------array([8, 8, 8, 8, 8, 8])
>>> a.shape=(3,2)
>>> a
```

```
       array([[8, 8],
              [8, 8],
              [8, 8]])
>>> a=np.full(6,9)
>>> a----------array([9, 9, 9, 9, 9, 9])
>>> a.reshape(2,3)
              array([[9, 9, 9],
                     [9, 9, 9]])
>>> a=np.full((3,3),9)
>>> a
              array([[9, 9, 9],
                     [9, 9, 9],
                     [9, 9, 9]])
>>> a=np.full((2,3),6)
>>> a
       array([[6, 6, 6],
              [6, 6, 6]])
>>> a.reshape(3,2)
       array([[6, 6],
              [6, 6],
              [6, 6]])
>>> a=np.full((3,3,3),7)
>>> a
       array([[[7, 7, 7],
              [7, 7, 7],
              [7, 7, 7]],

              [[7, 7, 7],
              [7, 7, 7],
              [7, 7, 7]],

              [[7, 7, 7],
              [7, 7, 7],
              [7, 7, 7]]])
```
===

6) identity():

=>This function always bulid Identity or unit matrix
=>Syntax:- varname=numpy.identity(N,dtype)
=>Here N represents Either we can take Rows or Columns and PVM takes as NXN Matrix (Square Matrix--Unit or Identity)

Examples:

```
>>> import numpy as np
>>> a=np.identity(3,dtype=int)
>>> print(a,type(a))-------------
              [[1 0 0]
               [0 1 0]
               [0 0 1]] <class 'numpy.ndarray'>
>>> a=np.identity(5,dtype=int)
>>> print(a,type(a))
                     [[1 0 0 0 0]
                      [0 1 0 0 0]
                      [0 0 1 0 0]
                      [0 0 0 1 0]
                      [0 0 0 0 1]] <class 'numpy.ndarray'>
```

===
Numpy---Basic Indexing
===

=>If we want to access Single element of 1D,2D and N-D arrays we must use the concept of Basic Indexing.

=>Accessing Single Element 1D-Array :

=>Syntax:- ndarrayname [Index]

=>Here 'index' can be either either +ve or -ve indexing

Examples:

```
>>> a=np.array([10,20,30,40,50,60])
>>> a
array([10, 20, 30, 40, 50, 60])
>>> a[0]
10
>>> a[3]
40
```

=>Accessing single Element of 2D :

=>Syntax:- ndarrayobj[row index,column index]

Examples:-

```
>>>import numpy as np
>>> a=np.array([10,20,30,40,50,60])
>>> b=a.reshape(2,3)
>>> b
array([[10, 20, 30],
       [40, 50, 60]])
>>> b[0,0]
10
>>> b[0,1]
20
>>> b[1,2]
60
```

==

=>Accessing single Element of 3D :

Syntax:- ndarrayobj[Index of matrix , row index , column index]

Examples:

```
>>> a=np.array([10,20,30,40,50,60,70,80])
>>> b=a.reshape(2,2,2)
>>> b
array([[[10, 20],
        [30, 40]],

       [[50, 60],
        [70, 80]]])

>>> b[0,0,0]-----------10
>>> b[-1,0,0]---------50
>>> b[-2,1,1]---------40
```

```
=================================================================
       Numpy---Indexing and Slicing Operations of 1D,2D and 3D array
=================================================================
------------------------------------
1D Arrays Slicing:
------------------------------------
Syntax:-   1dndrrayobj[begin:end:step]
-----------------------
Examples:
-----------------------
>>> a=np.array([10,20,30,40,50,60,70])
>>> a------------array([10, 20, 30, 40, 50, 60, 70])
>>> a[::-1]-----------array([70, 60, 50, 40, 30, 20, 10])
>>> a[::]-----------array([10, 20, 30, 40, 50, 60, 70])
------------------------------------
2D Arrays Slicing:
------------------------------------
Syntax:- ndrrayobj[i,j]
       here 'i' represents  Row Index
       here 'j' represents Column Index

Syntax:-       2dndrrayobj[Row Index, Column Index]

Syntax:-       2dndrrayobj[begin:end:step, begin:end:step]

--------------------------------------------------------------------
Examples:
--------------------------------------------------------------------
>>> a=np.array([[10,20,30],[40,50,60]])
>>> a
array([[10, 20, 30],
       [40, 50, 60]])
>>> a[0,0]
10
>>> a[0:,0:1]
array([[10],
       [40]])
>>> a[0:,1:2]
array([[20],
       [50]])
>>> a[1:,:]
array([[40, 50, 60]])
===============================================================
3D Arrays Slicing
-----------------------------
Syntax:-       3dndrrayobj[i,j,k]

       here 'i' represents Which 2D matrix ( Matrix Number-->0 1 2 3 4
       5...... )
       here 'j' represents which Rows in that 2D matrix
       here 'k' represents which Columns in that 2D matrix
                            (OR)
Syntax:-       3dndrrayobj[ Matrix Index, Row Index, Column Index ]
                            (OR)
Syntax:-       3dndrrayobj[begin:end:step, begin:end:step, begin:end:step ]
----------------------
Examples:
----------------------
>>> lst=[ [ [1,2,3],[4,5,6],[7,8,9] ],[ [13,14,15],[16,17,18],[19,20,21] ] ]
>>> print(lst)
[[[1, 2, 3], [4, 5, 6], [7, 8, 9]], [[13, 14, 15], [16, 17, 18], [19, 20,
21]]]
```

```
>>> arr2=np.array(lst)
>>> print(arr2)
[[[ 1  2  3]
  [ 4  5  6]
  [ 7  8  9]]

 [[13 14 15]
  [16 17 18]
  [19 20 21]]]
>>> arr2.ndim
3
>>> arr2.shape
(2, 3, 3)
>>> arr2[:,:,0:1]
array([[[ 1],
        [ 4],
        [ 7]],

       [[13],
        [16],
        [19]]])
>>> arr2[:,:,:1]
array([[[ 1],
        [ 4],
        [ 7]],

       [[13],
        [16],
        [19]]])
>>> arr2[: , 0:2, 1:3]
array([[[ 2,  3],
        [ 5,  6]],

       [[14, 15],
        [17, 18]]])
>>> arr2[: , :2, 1:]
array([[[ 2,  3],
        [ 5,  6]],

       [[14, 15],
        [17, 18]]])
```

===
Numpy---Advanced Indexing
===

=>If we want to access multiple elements, which are not in order (arbitrary elements) of 1D,2D and N-D arrays we must use the concept of Advanced Indexing

=>If we want access the elements based on some condition then we can't use basic indexing and Basic Slicing Operations. To fullfill such type of requirements we must use advanced Indexing.

=>Accessing Multiple Arbitrary Elements ---1D :

=>Syntax:- ndarrayname [x]

=>Here 'x' can be either ndarray or list which represents required indexes of arbitrary elements.

Examples:

```
>>> lst=[10,20,30,40,50,60,70,80,90]
>>> a=np.array(lst)
>>> print(a)----------------[10 20 30 40 50 60 70 80 90]
#access 10   30  and 80 elements
# here indexes of 10 30 and 80  are  0 2 7
>>>lst=[0,2,7]  here [0,2,7] are indexes of 10  30 and 80
>>> indexes=np.array(lst) # here lst converted into ndarray object
>>> print(indexes)---------[0 2 7]
>>> print(a[indexes])--------------[10 30 80]
      (OR)
>>> ind=[0,2,7]  # prepare the list of indexes of arbitray
elements(10,30,80) of ndarray and pass to ndarray
>>> print(a[ind]) -----------[10 30 80]
```

Examples:

```
Q1-->Access  20  30 80 10 10 30
>>> lst=[10,20,30,40,50,60,70,80,90]
>>> a=np.array(lst)
>>> print(a)----------------[10 20 30 40 50 60 70 80 90]
>>> ind=[1,2,7,0,0,2] # [1,2,7,0,0,2] are the indexes of 20 30 80 10 10 30
>>> print(a[ind])----------------[20 30 80 10 10 30]
```

=>Accessing Multiple Arbitrary Elements ---2D :

=>Syntax:- ndarrayobj[[row indexes],[column indexes]]

Examples:-

```
>>>import numpy as np
>>>mat=np.array([ [1,2,3,4],[5,6,7,8],[9,10,11,12],[13,14,15,16] ] )
>>> print(mat)
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]
 [13 14 15 16]]

Q1) Access the principle diagnal elements 1  6  11  16

Ans:-      mat[ [0,1,2,3],[0,1,2,3] ]
```

=>When the above statement is executed, The PVM takes internally as

```
          mat[ (0,0), (1,1), (2,2),(3,3) ]--------  1  6   11  16
```

```
>>> mat[ [0,1,2,3],[0,1,2,3] ]-----------array([ 1,  6, 11, 16])

Q2) Access the elements 6 14
Ans:      mat[  [1,3] , [1,1] ]
=>When the above statement is executed, The PVM takes internally as
        mat[ (1,1),(3,1) ]

>>> mat[[1,3],[1,1]]-----------array([ 6, 14])
======================================================================
```

=>Accessing Multiple Arbitrary Elements ---3D :

```
----------------------------------------------------------------------------
Syntax:-   ndarray[ [Indexes of 2Dmatrix],[row indexes],[column indexes]  ]
-------------
Examples:
---------------
>>>import numpy as np
>>>l1=[ [ [1,2,3,4],[5,6,7,8],[9,10,11,12] ],[
[13,14,15,16],[17,18,19,20],[21,22,23,24] ]  ]
>>>mat3d=np.array(l1)
>>>print(mat3d)
>>> print(mat3d)
[[[ 1  2  3  4]
  [ 5  6  7  8]
  [ 9 10 11 12]]

 [[13 14 15 16]
  [17 18 19 20]
  [21 22 23 24]]]
>>> mat3d.ndim
3
>>> mat3d.shape
(2, 3, 4)
----------------------------------------
Q1) Access the elements 1  14  24
Ans:-    mat3d[ [0,1,1], [0,0,2], [0,1,3]  ]

When the above statement is executed, Internally PVM takes as follows.
=>mat3d[ (0,0,0),(1,0,1),(1,2,3) ]-Gives-->1  14  24


Q1) Access the elements   10  16
>>> mat3d[[-2,-1],[-1,-3],[-3,-1]]----------array([10, 16])
==================================================
OR
========
>>> l1=[ [ [1,2,3,4],[5,6,7,8],[9,10,11,12] ],[
[13,14,15,16],[17,18,19,20],[21,22,23,24] ]  ]
>>> a=np.array(l1)
>>> a
array([[[ 1,  2,  3,  4],
        [ 5,  6,  7,  8],
        [ 9, 10, 11, 12]],

       [[13, 14, 15, 16],
        [17, 18, 19, 20],
        [21, 22, 23, 24]]])
>>> #ndarrayobj[ [MatrixIndex],[Row Index],[Col Index] ]---Syntax
>>> #ndarrayobj[ [MatrixIndex],[Row Index],[Col Index] ]
>>> #access 1,8,13,20
>>> matind=(0,0,1,1)
>>> rowind=(0,1,0,1)
>>> colind=(0,3,0,3)
>>> a[matind,rowind,colind]
array([ 1,  8, 13, 20])
>>> a[ [0,0,0,1,1,1],[0,1,2,0,1,2],[0,1,2,0,1,2] ]
array([ 1,  6, 11, 13, 18, 23])
===========================X============================================
A=np.array([10,20,30,40,50,60,70,80,15,25,35,45,55,65,75,85])
print(a)

a.shape=(2,2,2,2)
print(a)
               [[[[10 20]
                  [30 40]]

                 [[50 60]
                  [70 80]]]


                [[[15 25]
                  [35 45]]

                 [[55 65]
                  [75 85]]]]

#access 10 from a---4-D
a[0][0][0][0]------------10
# access 10 and 40 from  a---4D
a[[0,0],[0,0],[0,1],[0,1]]----array([10, 40])
# access 60,55 and 15 from a---4D
a[ [0,1,1],[1,1,0],[0,0,0],[1,0,0] ]----array([60, 55, 15])
```

===

Numpy--selecting the elements based on condition
(OR)
Creating Filter Directly From ndArray

===

=>To select any element from ndarray object, we have two approaches. They are

----
Approach-1:

=>Prepare Boolean Array (It contains True or False. True represents Condition satisfied and False represents Condition not satisfied]

Syntax:- varname=ndarrayobject with condition

varname is called boolean array.

=>Pass the Boolean Array to the ndarray object. so that we can get those elements from ndarray which satisfies with the entry True(or) we can get those elements from ndarray corresponding True entries of Boolean array.

Syntax: ndarray [Boolean Array]

Approach-2:

=>In this approach, we directly pass Boolean array values to the ndarray for getting required elements based on condition.

Syntax: ndarray[ndarrayobject with condition]

Examples:

```
-----------------------------
Q1) Select the Possitive Elements from ndarray
>>> import numpy as np
>>> l=[10,21,-34,23,-45,30,-40]
>>> print(l)--------------[10, 21, -34, 23, -45, 30, -40]
>>> a=np.array(l)
>>> a------------array([ 10,  21, -34,  23, -45,  30, -40])
>>> b=a>0    # Boolean Array
>>> print(b)----[ True  True False  True False  True False]
>>> a[b]-------array([10, 21, 23, 30])
=================OR=========================
>>> a[ a>0 ]-----------array([10, 21, 23, 30])
-------------------------------------------------------------------
Q2) Select the Negative Elements from ndarray
       >>> l=[10,21,-34,23,-45,30,-40]
       >>> a=np.array(l)
       >>> a----------       array([ 10,  21, -34,  23, -45,  30, -40])
       >>> b=a<0  # Boolean Array
       >>> b----     array([False, False,  True, False,  True, False,
True])
       >>> a[b]-------       array([-34, -45, -40])
              =================OR=============
       >>> a[a<0]--------------     array([-34, -45, -40])
----------------------------------------------------------------------------
Q3) Select the Even and Odd  Elements from ndarray
>>> a=np.array([11,20,33,31,41,47,46,12,13])
>>> a------------------array([11, 20, 33, 31, 41, 47, 46, 12, 13])
>>> a[a%2==0]-----------------array([20, 46, 12])
>>> a[a%2!=0]-----------------array([11, 33, 31, 41, 47, 13])
----------------------------------------------------------------------------
>>> a=np.array([10,20,30,40,50,60,70,80,90])
>>> b=a.reshape(3,3)
>>> b
              array([[10, 20, 30],
                     [40, 50, 60],
                     [70, 80, 90]])
#Get Multiples of 3
>>> m3=(b%3==0)
>>> m3  #----------------------Boolean array
                     array([[False, False,  True],
                            [False, False,  True],
                            [False, False,  True]])
>>> b[m3]--------------------------array([30, 60, 90])
=====================OR============================
>>> b[b%3==0]-------------------array([30, 60, 90])
===================================*&*======================================
```

==

Numpy--Arithmetic Operations (OR) Matrix Operations

==

=>On the objects of ndarray, we can apply all types of Arithmetic Operators.
=>To perform Arithmetic Operations on the objects of ndarray in numpy programming, we use the following functions.

```
            a) add()
            b) subtract()
            c) multiply()
            d) dot()  or matmul()
            e) divide()
            f) floor_divide()
            g) mod()
            h) power()
```

=>All the arithmetic Functions can also be perfomed w.r.t Arithmetic Operators.

----

a) add():

Syntax:- varname=numpy.add(ndarrayobj1, ndarrayobj2)
=>This function is used for adding elements of ndarrayobj1, ndarrayobj2 and result can be displayed
Examples:

```
>>> l1=[ [10,20],[30,40] ]
>>> l2=[[1,2],[3,4]]
>>> a=np.array(l1)
>>> b=np.array(l2)
>>> a
       array([[10, 20],
              [30, 40]])
>>> b
       array([[1, 2],
              [3, 4]])
>>> c=np.add(a,b)
>>> c
              array([[11, 22],
                        [33, 44]])
----------------------------------------------------------------------------
 >>> x=np.array([[1,2,3],[4,5,6]])
>>> x
              array([[1, 2, 3],
                     [4, 5, 6]])
>>> y=np.array([4,4,4])
>>> y
              array([4, 4, 4])
>>> z=x+y
>>> z
       array([[ 5,  6,  7],
              [ 8,  9, 10]])
>>> z=np.add(x,y)
>>> z
       array([[ 5,  6,  7],
              [ 8,  9, 10]])
>>> x
       array([[1, 2, 3],
              [4, 5, 6]])
>>> k=np.array([[2,3],[4,5]])
>>> k
       array([[2, 3],
              [4, 5]])
>>> kvr=np.add(x,k)----ValueError: operands could not be broadcast together
```

```
----------------------------------------------------------------------------
>>> l1=[[10,20],[30,40]]
>>> l2=[[1,2],[3,4]]
>>> a=np.array(l1)
>>> b=np.array(l2)
>>> a
              array([[10, 20],
                     [30, 40]])
>>> b
              array([[1, 2],
                     [3, 4]])
>>> c=a+b  # we used operator + instead of add()
>>> c
       array([[11, 22],
              [33, 44]])
=================================
```

b) subtract()

```
------------------------------
```

Syntax:- varname=numpy.subtract(ndarrayobj1, ndarrayobj2)

=>This function is used for subtracting elements of ndarrayobj1, ndarrayobj2 and result can be displayed

Examples:

```
------------------
>>> l1=[[10,20],[30,40]]
>>> l2=[[1,2],[3,4]]
>>> a=np.array(l1)
>>> b=np.array(l2)
>>> a
array([[10, 20],
       [30, 40]])
>>> b
array([[1, 2],
       [3, 4]])
>>> c=np.subtract(a,b)
>>> c
array([[ 9, 18],
       [27, 36]])
-----------------------------------
>>> d=a-b    # we used operator - instead of subtract()
>>> d
array([[ 9, 18],
       [27, 36]])
===================================
```

c) multiply():

```
-----------------------
```

Syntax:- varname=numpy.multiply(ndarrayobj1, ndarrayobj2)

=>This function is used for performing element-wise multiplication of ndarrayobj1, ndarrayobj2 and result can be displayed

Examples:

```
>>> l1=[[1,2],[3,4]]
>>> l2=[[5,6],[4,3]]
>>> a=np.array(l1)
>>> b=np.array(l2)
>>> a
array([[1, 2],
           [3, 4]])
>>> b
array([[5, 6],
          [4, 3]])
>>> c=np.multiply(a,b)
>>> c

array([[ 5, 12],
          [12, 12]])
-----------------------------------------------
>>> e=a*b   # we used operator * instead of multiply()
>>> e
array([[ 5, 12],
       [12, 12]])
------------------------------------------
```

d) dot()

=>To perform Matrix Multiplication, we use dot()

Syntax:-   varname=numpy.dot(ndarrayobj1, ndarrayobj2)

=>This function is used for performing actual matrix multiplication of ndarrayobj1, ndarrayobj2 and result can be displayed

Examples:

```
>>> l1=[[1,2],[3,4]]
>>> l2=[[5,6],[4,3]]
>>> a=np.array(l1)
>>> b=np.array(l2)
>>> a
       array([[1, 2],
              [3, 4]])
>>> b
       array([[5, 6],
              [4, 3]])
>>> d=np.dot(a,b)
>>> d
       array([[13, 12],
              [31, 30]])
```

----

e) divide()

Syntax:-   varname=numpy.divide(ndarray1,ndarry2)

=>This function is used for performing element-wise division of ndarrayobj1, ndarrayobj2 and result can be displayed

```
>>> l1=[[10,20],[30,40]]
>>> l2=[[1,2],[3,4]]
>>> a=np.array(l1)
>>> b=np.array(l2)
>>> a
       array([[10, 20],
              [30, 40]])
>>> b
       array([[1, 2],
              [3, 4]])
>>> c=np.divide(a,b)
>>> c
       array([[10., 10.],
              [10., 10.]])
-------------------------------------------------------------
>>> d=a/b    # we used operator / instead of divide()
>>> d
       array([[10., 10.],
              [10., 10.]])
```

---

f) floor_divide()

```
----------------------------------
```

Syntax:- varname=numpy.floor_divide(ndarray1,ndarry2)

=>This function is used for performing element-wise floor division of ndarrayobj1, ndarrayobj2 and result can be displayed

```
>>> l1=[[10,20],[30,40]]
>>> l2=[[1,2],[3,4]]
>>> a=np.array(l1)
>>> b=np.array(l2)
>>> a
       array([[10, 20],
              [30, 40]])
>>> b
       array([[1, 2],
              [3, 4]])
>>> c=np.floor_divide(a,b)
>>> c
       array([[10, 10],
              [10, 10]])
---------------------------------------------------------------
>>> d=a//b    # we used operator // instead of floor_divide()
>>> d
       array([[10, 10],
              [10, 10]])
```

---

g) mod()

```
-------------------------------
```

Syntax:- varname=numpy.mod(ndarray1,ndarry2)

=>This function is used for performing element-wise modulo division of ndarrayobj1, ndarrayobj2 and result can be displayed

```
--------------------
```

Examples:

```
---------------------
>>> l1=[[10,20],[30,40]]
>>> l2=[[1,2],[3,4]]
>>> a=np.array(l1)
>>> b=np.array(l2)
>>> a
       array([[10, 20],
              [30, 40]])
>>> b
       array([[1, 2],
              [3, 4]])
>>> c=np.mod(a,b)
>>> c
       array([[0., 0.],
              [0., 0.]])
------------------------------------------------------------------------
=>We can also do with operator %
>>> e=a%b
>>> e
       array([[0, 0],
                  [0, 0]],     dtype=int32)
```

---

h) power():

```
---------------------------------------
```

Syntax:- varname=numpy.power(ndarray1,ndarry2)

=>This function is used for performing element-wise exponential of ndarrayobj1, ndarrayobj2 and result can be displayed

```
---------------------------------------
```

```
>>> l1=[[10,20],[30,40]]
>>> l2=[[1,2],[3,4]]
>>> a=np.array(l1)
>>> b=np.array(l2)
>>> a
       array([[10, 20],
              [30, 40]])
>>> b
       array([[1, 2],
              [3, 4]])
>>>c=np.power(a,b)
>>>print(c)
       array([[     10,     400],
              [  27000, 2560000]],
-------------------------------------------
>>> f=a**b   # Instead of using  power() we can use ** operator
>>> f
       array([[     10,     400],
              [  27000, 2560000]],    dtype=int32)
```

===

Numpy--Statistical Operations

===

=>On the object of ndarray, we can the following Statistical Operations .

```
              a) amax()
              b) amin()
              c) mean()
              d) median()
              e) var()
              f)  std()
```

=>These operation we can perform on the entire matrix and we can also peform on columnwise (axis=0) and Rowwise (axis=1)

a) amax():

=>This functions obtains maximum element of the entire matrix.

=>Syntax1:-   varname=numpy.amax(ndarrayobject)

=>Syntax2:-   varname=numpy.amax(ndarrayobject,axis=0)--->obtains max

=>Syntax3:-   varname=numpy.amax(ndarrayobject,axis=1)--->obtains max

Examples:

```
>>> l1=[[1,2,3],[4,2,1],[3,4,2]]
>>> A=np.array(l1)
>>> print(A)
              [[1 2 3]
               [4 2 1]
               [3 4 2]]
>>> max=np.amax(A)
>>> cmax=np.amax(A,axis=0)
>>> rmax=np.amax(A,axis=1)
>>> print("Max element=",max)-----------Max eleemnt= 4
>>> print("Column Max eleemnts=",cmax)---Column Max eleemnts= [4 4 3]
>>> print("Row Max eleemnts=",rmax)---Row Max eleemnts= [3 4 4]
```

----

b) amin():

=>This functions obtains minmum element of the entire matrix.

=>Syntax1:-   varname=numpy.amin(ndarrayobject)

=>Syntax2:-   varname=numpy.amin(ndarrayobject,axis=0)--->obtains min

=>Syntax3:- varname=numpy.amin(ndarrayobject,axis=1)--->obtains min

Examples:

```
-------------------
>>> l1=[[1,2,3],[4,2,1],[3,4,2]]
>>> A=np.array(l1)
>>> print(A)
               [[1 2 3]
                [4 2 1]
                [3 4 2]]
>>> min=np.amin(A)
>>> cmin=np.amin(A,axis=0)
>>> rmin=np.amin(A,axis=1)
>>> print("Min eleemnt=",min)---Min eleemnt= 1
>>> print("Column Min eleemnts=",cmin)---Column Min eleemnts= [1 2 1]
>>> print("Row Min eleemnts=",rmin)---Row Min eleemnts= [1 1 2]
```

---

c) mean():

=>This is used for cal mean of the total matrix elements.

=>The formula for mean=(sum of all elements of matrix) / total number of elements.

Syntax1:- varname=numpy.mean(ndarrayobject)

Syntax2:- varname=numpy.mean(ndarrayobject,axis=0)--->Columnwise Mean

Syntax3:- varname=numpy.mean(ndarrayobject,axis=1)--->Rowwise Mean

Examples:

```
-----------------
>>> l1=[[1,2,3],[4,2,1],[3,4,2]]
>>> A=np.array(l1)
>>> print(A)
               [[1 2 3]
                [4 2 1]
                [3 4 2]]
>>> m=np.mean(A)
>>> cm=np.mean(A,axis=0)
>>> rm=np.mean(A,axis=1)
>>> print("Mean=",m)---------Mean= 2.4444444444444446
>>> print("Column Mean=",cm)-----Column Mean= [2.66666667    2.66666667
2. ]
>>> print("Row Mean=",rm)---Row Mean= [ 2.          2.33333333        3.  ]
```

---

d) median()

=>This is used for calculating / obtaining median of entire matrix elements.

=>Median is nothing but sorting the given data in ascending order and select middle element.

=>If the number sorted elements are odd then center or middle element becomes median.

=>If the number sorted elements are even then select center or middle of two elements, add them and divided by 2 and that result becomes median.

Syntax1:- varname=numpy.median(ndarrayobject)

Syntax2:- varname=numpy.median(ndarrayobject,axis=0)

Syntax3:- varname=numpy.median(ndarrayobject,axis=1)

Examples:

```
--------------------
>>> l1=[[1,2,3],[4,2,1],[3,4,2]]
>>> A=np.array(l1)
>>> print(A)
               [[1 2 3]
                [4 2 1]
                [3 4 2]]
>>> md=np.median(A)
>>> cmd=np.median(A,axis=0)
>>> rmd=np.median(A,axis=1)
>>> print("Median=",md)----Median= 2.0
>>> print("Column Median=",cmd)---Column Median= [3.  2.  2.]
>>> print("Row Median=",rmd)------Row Median= [2.   2.  3.]
>>> l1=[[2,3],[4,1]]
>>> A=np.array(l1)
>>> print(A)
               [[2 3]
                [4 1]]
>>> md=np.median(A)
>>> cmd=np.median(A,axis=0)
>>> rmd=np.median(A,axis=1)
>>> print("Median=",md)---Median= 2.5
>>> print("Column Median=",cmd)---Column Median= [3. 2.]
>>> print("Row Median=",rmd)---Row Median= [2.5 2.5]
```
----

e) var():

Variance= sqr(mean-xi) / total number of elements
here 'xi' represents each element of matrix.

Syntax1:- varname=numpy.var(ndarrayobject)

Syntax2:- varname=numpy.var(ndarrayobject,axis=0)

Syntax3:- varname=numpy.var(ndarrayobject,axis=1)

Examples:

```
>>> l1=[[1,2,3],[4,2,1],[3,4,2]]
>>> A=np.array(l1)
>>> print(A)
       [[1 2 3]
        [4 2 1]
        [3 4 2]]
>>> vr=np.var(A)
>>> cvr=np.var(A,axis=0)
>>> rvr=np.var(A,axis=1)
>>> print("Variance=",vr)----Variance= 1.1358024691358024
>>> print("Column Variance=",cvr)---Column Variance= [1.55555556 0.88888889
>>> print("Row Variance=",rvr)---Row Variance= [0.66666667 1.55555556
0.66666667]
```
----

f) std()

----

standard deviation=sqrt(var)

Syntax1:- varname=numpy.std(ndarrayobject)

Syntax2:- varname=numpy.std(ndarrayobject,axis=0)

Syntax3:- varname=numpy.std(ndarrayobject,axis=1)

Examples:

```
>>> l1=[[1,2,3],[4,2,1],[3,4,2]]
>>> A=np.array(l1)
>>> print(A)
       [[1 2 3]
        [4 2 1]
        [3 4 2]]
>>> vr=np.var(A)
>>> cvr=np.var(A,axis=0)
>>> rvr=np.var(A,axis=1)
>>> print("Variance=",vr)---Variance= 1.1358024691358024
>>> print("Column Variance=",cvr)---Column Variance= [1.55555556 0.88888889
>>> print("Row Variance=",rvr)---Row Variance= [0.66666667 1.55555556
0.66666667]
-----------------------------------------------------------------------------
>>> sd=np.std(A)
>>> csd=np.std(A,axis=0)
>>> rsd=np.std(A,axis=1)
>>> print("std=",sd)---std= 1.0657403385139377
>>> print(" column std=",csd)--- column std= [1.24721913 0.94280904
0.81649658]
>>> print("Row std=",rsd)--Row std= [0.81649658 1.24721913 0.81649658]
```

Adding Elements in Numpy Array

Numpy module in python, provides a function to numpy.append() to add an element in a numpy array.

Syntax: Varname=numpy.append(ndarrayobj, value)

Example.

```
import numpy as np
#Create a Numpy Array of integers
arr = np.array([11, 2, 6, 7, 2])
# Add / Append an element at the end of a numpy array
new_arr = np.append(arr, 10)
print('New Array: ', new_arr)
print('Original Array: ', arr)
```

Insering elements in numpy

Numpy module in python, provides a function numpy.insert() to insert the element in ndarray at particular Index.

Syntax: varname=np.insert((ndarray , index, value)

Example.

```
import numpy as np
a=np.array([10,20,30,40])
a=np.array([10,20,30,40])
b=np.insert(a,1,35)
print(a)# array([10, 35, 20, 30, 40])
```

----

deleting element(s) from Numpy Array

Numpy module in python, provides a function numpy.delete() to delete elements.

Syntax: varname=np.delete(ndaary,index) # Delete element at index

Syntax: varname=np.delete(ndaary,[index1,..index-n]) # Delete element at Indices Example.

```
import numpy as np
arr = np.array([4,5,6,7,8,9,10,11])
arr = np.delete(arr, 2)
print(arr)
arr = np.array([4, 5, 6, 7, 8, 9, 10, 11])
# Delete element at index positions 1,2 and 3
arr = np.delete(arr, [1,2,3])
```

NumPy Sorting Arrays

=>Sorting is nothing arranging the elements in an ordered sequence.
=>Ordered sequence is any sequence that has an order corresponding to elements, like numeric or alphabetical, ascending or descending.
=>The NumPy ndarray object has a function called sort(), that will sort a specified array.

Examples:

```
-------------------
import numpy as np
arr = np.array([3, 2, 0, 1])
print(np.sort(arr))  #   [0 1 2 3]
----------------------------------------------------
import numpy as np
arr = np.array(['banana', 'cherry', 'apple'])
print(np.sort(arr)) # ['apple' 'banana' 'cherry']
-----------------------------------------------------
import numpy as np
arr = np.array([True, False, True])
print(np.sort(arr)) # [False True True]
-------------------------------------------------------
Sorting a 2-D Array
--------------------------------------------------------
If you use the sort() method on a 2-D array, both columns and Rows of nd
array will be sorted.
-------------------
```

Examples:

```
-------------------
              import numpy as np
              arr = np.array([[3, 2, 4], [5, 0, 1]])
              print(np.sort(arr))
#output
                     [[2 3 4]
                      [0 1 5]]
----------------------------------------------------------------------------
a=np.array([110, 20, -30, 40, 50, 160, 7, 8, 90])
print(a)

np.sort(a)-----------array([-30,   7,   8,  20,  40,  50,  90, 110, 160])
np.sort(a)[::-1]-----array([160, 110,  90,  50,  40,  20,   8,   7, -30])
a.shape=(3,3)
a-------------------------------array([[110,  20, -30],
                                    [ 40,  50, 160],
                                    [  7,   8,  90]])

np.sort(a,axis=0)  # ColumnWise
                             array([[  7,   8, -30],
                                    [ 40,  20,  90],
                                    [110,  50, 160]])
-----------------------------------------------------------
print(a)
                     array([[110,  20, -30],
                            [ 40,  50, 160],
                            [  7,   8,  90]])

np.sort(a,axis=1)  # Row Wise
                     array([[-30,  20, 110],
                            [ 40,  50, 160],
                            [  7,   8,  90]])
```

```
        =====================================================
                          NumPy Array Copy vs View
        =====================================================
```

=>The Difference Between Copy and View

=>The main difference between a copy and a view of an array is that the copy is a new array, and the view is just a view of the original array.
=>The copy owns the data and any changes made to the copy will not affect original array, and any changes made to the original array will not affect the copy. modfifications are Independent (Like Shallow Copy)

```
=>Syntax:-      varname=ndrrayobj.copy()
                             (OR)
                     ndarrayobj2=numpy.copy(ndarrayobj1)
```

=>The view does not own the data and any changes made to the view will affect the original array, and any changes made to the original array will affect the view.

```
=>Syntax:-      varname=ndrrayobj.view()

              ERROR :   varname=ndrrayobj.view()
```

COPY:

Example

```
# Make a copy, change the original array, and display both arrays:
import numpy as np
arr = np.array([1, 2, 3, 4, 5])
x = arr.copy()
arr[0] = 42
print(arr)              # [42  2  3  4  5]
print(x)                # [1 2 3 4 5]
```

NOTE: The copy SHOULD NOT be affected by the changes made to the original array.

VIEW:

Example

```
#Make a view, change the original array, and display both arrays:
import numpy as np
arr = np.array([1, 2, 3, 4, 5])
x = arr.view()
arr[0] = 42
print(arr)  # [42  2  3  4  5]
print(x)   # [42  2  3  4  5]
```

NOTE : The view SHOULD be affected by the changes made to the original array.

```
# Make Changes in the VIEW:
------------------
```

Example

```
# Make a view, change the view, and display both arrays:
import numpy as np
arr = np.array([1, 2, 3, 4, 5])
x = arr.view()
x[0] = 31
```

```
print(arr)  # [31  2  3  4  5]
print(x)  # [31  2  3  4  5]
----------------------------------------------------------------------------
```

=>In the case append() , view() does not reflect the changes

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

Pandas

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

Introduction to Pandas:

=>Pandas is an open source Python Library / Module providing high performance and data manipulation and Analysis Tool.
=>The word PANDAs derived from PANel DAta
=>The pandas concept developed by WES MCKinney in the year 2008.
=>The Traditional Python Programming does not contain any Module for Data Analysis and Now Python Programming uses Pandas as an anaysis tool.
=>Python Pandas can be used in wide range of fields like Finance Services, Statistics , retail maketing sectors..etc
=>pandas module developed in C and Python Languages.

Instalation of Pandas:

=>The standard python software / Distribution(CPYTHON) does not contain any module for data analysis and now we are using third party module called PANDAS and whose module name is pandas
=>Programatically to use pandas as part of our python program, we must install pandas module by using pip tool.

Syntax:- pip install module name

Example:- pip install pandas

Key Features of Pandas:-----> Series DataFrame Panel

----

1) Fast and Efficient Data Frame with default costomized indexing
2) Tools for loading the data in in-memory data objects(objects of Series, DataFrame Panel)
3) We can access the data from pandas by using Labeled Based Slicing and indexing.
4) Columns from in-memory data objects(objects of Series, DataFrame Panel) can be deleted and inserted

===

Data Structures used in Pandas

===

=>In Pandas programming, we can store the data in 3 types of Data structures. They are.

a) Series
b) DataFrame
c) Panel

=>The best of way of thinking of these data structires is that The higher dimensional Data Structure is a container of its lower dimensional data structure.

Examples:

=>Series is part of DataFrame.
=>DataFrame is a part of Panel.

```
                                ====================================
                                              Series
                                ====================================
=>It is a One-Dimensional Labelled Array Capable of Storing / Holding
  Homogeneous data of any type (Integer, String, float,.........Python
  objects  etc).
=>The Axis Labels are collectively called Index.
=>Pandas Series is nothing but a column value in excel sheet.
=>Pandas Series Values are Mutable.
=>Pandas Series contains Homogeneous Data ( Internally even we store
  different types values , They are treated as object type)
----------------------------------------------------------------------------
                            Creating a Series
----------------------------------------------------------------------------
=>A Series object can be created by using the folowing Syntax:
Syntax:-
--------------
              varname=pandas.Series(object, index, dtype)
-------------------
Explanation:-
-------------------
=>Here varname is an object of <class, pandas.core.series.Series >
=>pandas is module name
=>Series() is pre-defined Function in pandas module and it is used for
  creating an  object of Series class.
=>'object' can either int, float, complex, bool, str, bytes,
  bytearray,range, list,ndarray,dict .....etc (But not set type bcoz they
  are un-ordered)
=>'index' represents the position of values present Series object. The
  default value of  Index starts from 0 to n-1, Here n represents number of
  values in Series object. Programatically we can give our own Index Values.
=>'dtype' represents data type (Ex:- int32, ,int64, float32, float64...etc)
----------------------------------------------------------------------------
Examples:-    Create a series for 10 20  30 40 50 60

>>> import pandas as pd
>>> import numpy as np
>>> lst=[10,20,30,40,50,60]
>>> s=pd.Series(lst)
>>> print(s,type(s))
                             0    10
                             1    20
                             2    30
                             3    40
                             4    50
                             5    60
dtype: int64                  <class 'pandas.core.series.Series'>
----------------------------
>>> lst=[10,20,30,40,50,60]
>>> s=pd.Series(lst,dtype=float)
>>> print(s,type(s))
              0    10.0
              1    20.0
              2    30.0
              3    40.0
              4    50.0
              5    60.0
dtype: float64 <class 'pandas.core.series.Series'>
------------------------------------------------------------------
```

```
>>> lst=["Rossum","Gosling","Travis","MCKinney"]
>>> a=np.array(lst)
>>> a ------array(['Rossum', 'Gosling', 'Travis', 'MCKinney'], dtype='<U8')
>>> print(a, type(a))--['Rossum' 'Gosling' 'Travis' 'MCKinney'] <class
'numpy.ndarray'>
>>> s=pd.Series(a)
>>> print(s,type(s))
              0      Rossum
              1     Gosling
              2      Travis
              3    MCKinney
dtype: object     <class 'pandas.core.series.Series'>
----------------------------------------------------------------------
>>>lst=[10,"Rossum",34.56,"Author"]
>>> s=pd.Series(lst)
>>> print(s,type(s))
              0        10
              1    Rossum
              2     34.56
              3    Author
dtype: object           <class 'pandas.core.series.Series'>
```

Creating an Series object with Programmer-defined Index

```
>>> lst=[10,"Rossum",34.56,"Author"]
>>> print(lst)--------[10, 'Rossum', 34.56, 'Author']
>>> s=pd.Series(lst,index=["Stno","Name","Marks","Desg"])
>>> print(s)
              Stno         10
              Name     Rossum
              Marks     34.56
              Desg     Author
              dtype: object
>>> print(s["Stno"])-------10
---------------------------------------------------------------------------
>>> lst=["Rossum","Gosling","Travis","MCKinney"]
>>> s=pd.Series(lst,index=[100,200,300,400])
>>> print(s,type(s))
                    100      Rossum
                    200     Gosling
                    300      Travis
                    400    MCKinney
dtype: object <class 'pandas.core.series.Series'>
```

Creating a Series object from dict

=>A dict object can be used for creating a series object
=>If we use dict object in Series() then keys can be taken as Indices (Or Indexes)automatically and corresponding values of dict can be taken as Series data.

Examples:

```
>>> import pandas as pd
>>> d1={"sub1":"Python","sub2":"Java","sub3":"Data Science","sub4":"ML"}
>>> print(d1)--{'sub1': 'Python', 'sub2': 'Java', 'sub3': 'Data Science',
'sub4': 'ML'}
>>> s=pd.Series(d1)
>>> print(s)
              sub1          Python
              sub2            Java
              sub3    Data Science
```

```
               sub4             ML
               dtype: object
>>> d2={"RS":2.3,"JG":1.2,"MCK":4.5,"TOLI":2.4}
>>> print(d2)---{'RS': 2.3, 'JG': 1.2, 'MCK': 4.5, 'TOLI': 2.4}
>>> s=pd.Series(d2)
>>> print(s)
               RS      2.3
               JG      1.2
               MCK     4.5
               TOLI    2.4
               dtype: float64
```

==

DataFrame in Pandas

==

=>A DataFrame is 2-Dimensional Data Structure to organize the data .
=>In Otherwords a DataFrame Organizes the data in the Tabular Format, which is nothing but Collection of Rows and Columns.
=>The Columns of DataFrame can be Different Data Types or Same Type
=>The Size of DataFrame can be mutable.

==

Number of approaches to create DataFrame

==

=>To create an object of DataFrame, we use pre-defined DataFrame() which is present in pandas Module and returns an object of DataFrame class.
=>We have 5 Ways to create an object of DataFrame. They are

a) By using list / tuple
b) By using dict
c) By using Series
d) By using ndarray of numpy
e) By using CSV File (Comma Separated Values)

=>Syntax for creating an object of DataFrame in pandas:

```
       varname=pandas.DataFrame(object,index,columns,dtype)
```

Explanation:

=>'varname' is an object of <class,'pandas.core.dataframe.DataFrame'>
=>'pandas.DataFrame()' is a pre-defined function present in pandas module and it is used to create an object of DataFrame for storing Data sets.
=>'object' represents list (or) tuple (or) dict (or) Series (or) ndarray (or) CSV file
=>'index' represents Row index and whose default indexing starts from 0,1,...n-1 where 'n' represents number of values in DataFrame object.
=>'columns' represents Column index whose default indexing starts from 0,1..n-1 where n number of columns.
=>'dtype' represents data type of values of Column Value.

==

Creating an object DataFrame by Using list / tuple

```
>>>import pandas as pd
>>>lst=[10,20,30,40]
>>>df=pd.DataFrame(lst)
>>>print(df)
                    0
               0  10
               1  20
               2  30
               3  40
```

```
lst=[[10,20,30,40],["RS","JS","MCK","TRV"]]
df=pd.DataFrame(lst)
print(df)
             0   1    2    3
       0  10  20   30   40
       1  RS  JS  MCK  TRV
--------------------------------------------
lst=[[10,'RS'],[20,'JG'],[30,'MCK'],[40,'TRA']]
df=pd.DataFrame(lst)
print(df)
            0     1
       0  10    RS
       1  20    JG
       2  30   MCK
       3  40   TRA
--------------------------------------------------
lst=[[10,'RS'],[20,'JG'],[30,'MCK'],[40,'TRA']]
df=pd.DataFrame(lst, index=[1,2,3,4],columns=['Rno','Name'])
print(df)

           Rno Name
       1    10   RS
       2    20   JG
       3    30  MCK
       4    40  TRA
-------------------------------------------
tpl=( ("Rossum",75), ("Gosling",85), ("Travis",65),
("Ritche",95),("MCKinney",60) )
df=pd.DataFrame(tpl, index=[1,2,3,4,5],columns=['Name','Age'])
print(df)
              Name    Age
       1    Rossum   75
       2   Gosling     85
       3    Travis      65
       4    Ritche      95
       5  MCKinney  60
----------------------------------------------------------------------------
```

Creating an object DataFrame by Using dict object

=>When we create an object of DataFrame by using Dict , all the keys are taken as Column Names and Values of Value are taken as Data.

Examples:

```
>>> import pandas as pd
>>>
dictdata={"Names":["Rossum","Gosling","Ritche","McKinney"],"Subjects":["Pyth
on","Java","C","Pandas"],"Ages":[65,80,85,55]  }
>>> df=pd.DataFrame(dictdata)
>>> print(df)
                    Names   Subjects    Ages
              0    Rossum   Python  65
              1   Gosling     Java         80
              2    Ritche        C         85
              3  McKinney   Pandas  55
>>> df=pd.DataFrame(dictdata,index=[1,2,3,4])
>>> print(df)
                    Names    Subjects      Ages
              1    Rossum   Python          65
              2   Gosling     Java              80
              3    Ritche        C                 85
              4  McKinney   Pandas        55
```

Creating an object DataFrame by Using Series object

```
>>> import pandas as pd
>>> sdata=pd.Series([10,20,30,40])
>>> df=pd.DataFrame(sdata)
>>> print(df)
            0
       0  10
       1  20
       2  30
       3  40
>>> sdata=pd.Series({"IntMarks":[10,20,30,40],"ExtMarks":[80,75,65,50]})
>>> print(sdata)
IntMarks    [10, 20, 30, 40]
ExtMarks    [80, 75, 65, 50]
dtype: object

>>> df=pd.DataFrame(sdata)
>>> print(df)
                              0
       IntMarks  [10, 20, 30, 40]
       ExtMarks  [80, 75, 65, 50]
>>> ddata={"IntMarks":[10,20,30,40],"ExtMarks":[80,75,65,50]}
>>> df=pd.DataFrame(ddata)
>>> print(df)
   IntMarks  ExtMarks
0        10        80
1        20        75
2        30        65
3        40        50
```

Creating an object DataFrame by Using ndarray object

```
>>> import numpy as np
>>> l1=[[10,60],[20,70],[40,50]]
>>> a=np.array(l1)
>>> df=pd.DataFrame(a)
>>> print(df)
                    0   1
              0  10  60
              1  20  70
              2  40  50
>>> df=pd.DataFrame(a,columns=["IntMarks","ExtMarks"])
>>> print(df)
                 IntMarks  ExtMarks
              0        10        60
              1        20        70
              2        40        50
```

----

e) By using CSV File(Comma Separated Values)

```
-----------------------------------------------------------------
import pandas as pd1
df=pd1.read_csv("D:\KVR-JAVA\stud.csv")
print("type of df=",type(df)) #type of df= <class
'pandas.core.frame.DataFrame'>
print(df)
--------------------- OUTPUT--------------------
           stno     name            marks
       0    10    Rossum             45.67
       1    20    Gosling            55.55
       2    30    Ritche            66.66
       3    40    Travis            77.77
       4    50      KVR             11.11
```

```
       =======================================================
                      Accesssing the Data of DataFrame
       =======================================================
1) DataFrameobj.head(no.of rows)
2) DataFrameobj.tail(no.of rows)
3) DataFrameobj.describe()
4) DataFrameobj.shape
5) DataFrameobj [start:stop:step]
6) DataFrameobj["Col Name"]
7) DataFrameobj[ ["Col Name1","Col Name-2"...."Col Name-n"] ]
8) DataFrameobj[ ["Col Name1","Col Name-2"...."Col Name-n"]]
[start:stop:step]
9) DataFrameobj.iterrows()
===================================================
Understabding loc() ----- here start and stop index Included and
                          Col Names can be used(but not column numbers]
----------------------------------------------------------------------------
1) DataFrameobj.loc[row_number]
2) DataFrameobj.loc[row_number,[Col Name,.........] ]
3) DataFrameobj.loc[start:stop:step]
4) DataFrameobj.loc[start:stop:step,["Col Name"] ]
5) DataFrameobj.loc[start:stop:step,["Col Name1", Col Name-2......."] ]
6) DataFrameobj.loc[start:stop:step,"Col Name1" : Col Name-n"]
----------------------------------------------------------------------------
Understabding iloc() ----- here start index included and stop index excluded
and Col Numbers  must be used(but not column names]
----------------------------------------------------------------------------
1) DataFrameobj.iloc[row_number]
2) DataFrameobj.iloc[row_number,Col Number.........]
3) DataFrameobj.iloc[row_number,[Col Number1,Col Number2............] ]
3) DataFrameobj.iloc[row start:row stop, Col Start: Col stop]
4) DataFrameobj.iloc[row start:row stop:step, Col Start: Col stop:step]
5) DataFrameobj.iloc[row start:row stop,Col Number ]
6) DataFrameobj.iloc[ [row number1, row number-2.....] ]
7) DataFrameobj.iloc[ row start: row stop , [Col Number1,Col
   Number2............] ]
8) DataFrameobj.iloc[ : , [Col Number1,Col Number2............] ]
=================================================================
```

Adding Column Name to Data Frame

```
=================================================================
1)  dataframeobj['new col name']=default value
2)  dataframeobj['new col name']=expression
```

```
=======================================================================
               Removing Column Name from Data Frame
=======================================================================
1)dataframe.drop(columns="col name")
2)dataframe.drop(columns="col name",inplace=True)
=======================================================================
               sorting the dataframe data
=======================================================================
1) dataframeobj.sort_values(["colname"])
2) dataframeobj.sort_values(["colname"],ascending=False)
3) dataframeobj.sort_values(["colname1","col name2",...col name-n] )
=======================================================================
               knowing duplicates in dataframe data
=======================================================================
1) dataframeobj.duplicated()---------------gives boolean result
=======================================================================
               Removing duplicates from dataframe data
=======================================================================
1) dataframeobj.drop_duplicates()
2) dataframeobj.drop_duplicates(inplace=True)
=======================================================================
               Data Filtering and Conditional Change  / updations
=======================================================================
1) dataframeobj.loc[ simple condition]

                      Ex:    df.loc[ df["maths"]>75 ]
                              df.loc[df["maths"]>90 ,["name","maths"]]

2) dataframeobj.loc[ compound condition]
                      Ex:   df.loc[ (df["maths"]>60) & (df["maths]<85) ]
                      Ex:  df.loc[ (df["maths"]>95)
                        &(df["maths"]<=99),["name","maths"] ]

3) dataframeobj.loc[ (compund condition), ["Col Name"] ]=Expression

Ex: df.loc[ (df["percent"]>=60)  & (df["percent"]<=80),["grade"] ]="First"
# cond updattion.
```

Multi Threading

=>The Purpose of Multi Threading is that "To Provide Concurrent Execution / Simultaneous Execxition Or Parallel Processing(executing all at once)."
=>In Industry , we have two types of Applications / languages. They are

1. Process Based Applications
2. Thread Based Applications.

---

1. Process Based Applications

=>Process Based Applications contains Single Thread
=>Process Based Applications Provides Sequential Execution
=>Process Based Applications Takes More Exeution Time
=>Process Based Applications are treated as Heavy Weight Components.
Examples: C,CPP.

---

2. Thread Based Applications

=>Thread Based Applications contains by default Single Thread and Programtically we can create Multiple Threads.
=>Therad Based Applications Provides Both Sequential Execution and Concurrent Execution.
=>Thread Based Applications Takes Less Exeution Time
=>Therad Based Applications are treated as Light Weight Components.

Examples: Python, Java, C#.net...etc

Module Name used for developing Thread Based Applications

=>The module name required for developing Thread Based Applications is "threading".
=>In otherwords, "threading" is of the the pre-defined module of python for developing multi threading applications,

---

Module Name : threading

Function Names:

---

1) current_thread(): This Function is used for obtaining Name of thread which is currently executing
Syntax: tname=threading.current_thread().name
Example: Refer FirstThreadEx1.py program

2) active_count(): This Function is used obtaining number of active threads.Active Threads are nothing but which are under execution.
Syntax: avtivethreads=threading.active_count()
Examples: Refer SeconfThreadEx.py program

Class Name: Thread

---

1) Thread(target,args): This Function is used for creating Sub or Child Thread.
Syntax:- varname=threading.Thread(target=functionname,args=(val1,val2..))
=>Here Varname is an object of Thread Class and treated as sub thread.
=>"target=Functionname" represents sub thread is created to execute the specified "FunctionName"
=>"args=(val1,val2...val-n) represents list of values passed as arguments to functionname provided if the function name takes parameters and writing args in syntax is optional.

2) setname(str): This Function is used for setting User-Friendly Name to sub threads.This Function deprecated on the name of "name" attribute

```
        Syntax:      subthreadobj.setName(str)  # Not recommended to use
                                                        (OR)
        subthreadobj.name= str data # recommended to use
        Here str data is one the user-friendly name to therad
        Example:  Refer  FifthThreadEx.py
```

3) getname(): This Function is used for obtaning thread name.
This Function deprecated on the name of "name" attribute

```
Syntax:      tname=subthreadobj.getName()  # Not recommended to use
                                                        (OR)
tianme=subthreadobj.name # recommended to use
Example:  Refer  FifthThreadEx.py
```

4) start(): This Function is used for dispatching or Sending the sub threads to the target function. Without using start(), we can't dispatch the sub thread(s).

```
        Syntax:      subthreadobj.start()
        Example:  Refer  ThirdThreadEx.py
```

5) is_alive(): This Function is used for obtaining whether the thread is active or not.This Function returns True provided when thread is under execution otherwise it return False.

```
                        Syntax:   varname= threading.is_alive()
                        Example:  Refer  ThirdThreadEx.py
```

6) join()   : This Function is used for making the sub threads to join with main thread for collecting all sub threads by main thread and handover to Garbage Collector

```
                     Syntax:     subthreadobj1.join()
                                     subthreadobj2.join()
                                     --------------------------------
                                      subthreadobj-n.join()
                        Examples:  FourthThreadEx.py
```

===

Synchronization in Multi Threading
(OR)
Locking concept in Threading

===

=>When multiple threads are operating / working on the same resource(function / method) then by default we get dead lock result / race condition / wrong result / non-thread safety result.
=>To overcome this dead lock problems, we must apply the concept of Synchronization
=>The advantage of synchronization concept is that to avoid dead lock result and provides Thread Safety Result.
=>In Python Programming, we can obtain synchronization concept by using locking and un-locking concept.

=>Steps for implementing Synchronization Concept:
(OR)
Steps for avoiding dead lock

----

1) obtain / create an object of Lock class, which is present in threading module.

Syntax:-

```
----------------
                        lockobj=threading.Lock()
```

2) To obtain the lock on the sharable resource, we must use acquire()

Syntax:

```
--------------
                        lockobj.acquire()
```

Once current object acquire the lock, other thread objects are made wait until curent thread object releases the lock.

3) To un-lock the sharable resource/current object, we must use release()

Syntax:

```
-------------
                        lockobj.release()
```

Once current object releases the lock, other objects are permitted into shrable resource. This process of aquiring and releasing the lock will be continued until all the objects completed their execution.

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

decorator

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

=>A decorator a function, which will provide additional Processing Capability for Normal Functions.

=>A decorator always takes a Normal Function Name as Parameter.

Syntax:

```
                    def    functionname1(Formal Param):
                         def innerfunctionname():
                              -------------------------
                              -------------------------
                              return  inner function res
                         return Inner Function name
```

Calling Decorator with Normal Function

```
----------------------------------------------------------------------------
                      @functionname1
                         def inormalfunctionname():
                              -------------------------
                              -------------------------
                              return  inner function res
                         return Inner Function name
```

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

JSON file

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

=>JSON (JavaScript Object Notation) is a popular data format used for representing structured data. It's common to transmit and receive data between a server and Client web application development in the form of JSON format.

=>In otherwords,JSON is a lightweight data format for data interchange which can be easily read and written by humans, easily parsed and generated by machines.

=>It is a complete language-independent text format. To work with JSON data, Python has a built-in module called json.

==

Parse JSON (Convert from JSON to Python)

=>json.loads() method can parse a json string and converted into Python dictionary.

Syntax:

```
              dictobj=json.loads(json_string)
```

Examples:

```
# Python program to convert JSON to Python
       import json
       # JSON string
       employee = '  {"id":"09", "name": "Rossum", "department":"IT"}  '
       # Convert JSON string to Python dict
       employee_dict = json.loads(employee)
       print(employee_dict)
```

Python--- read JSON file

=>json.load() method can read Consider a file named employee.json which contains a JSON object.a file which contains a JSON object.

Syntax:

```
       json.load(file_object)
```

Python--- write to JSON file

=>json.dump() method can write dict object data to a file.

Syntax:

```
       json.dump(dict object, file_pointer)
```

=================================
generator in python
=================================

=>generator is one of the function
=>The generator function always contains yield keyword
=>If the function contains return statement then it is called Normal function
=>If the function contains yield keyword then it is called generator
=>Syntax:

```
              def   function_name(start,stop,step):
                     ---------------------------------------
                    ---------------------------------------
                    yield value
                    ---------------
```

=>The 'yield' key word is used for giving the value back to function call from function defintion and continue the function execution until condition becomes false.
=>The advantage of generators over functions concept is that it save lot of memory space in the case large sampling of data. In otherwords Functions gives all the result at once and it take more memory space where as generators gives one value at a time when programmer requested and takes minimized memory space.

====================================
String Handling Part-2
====================================

=>On String Data, we can perform Indexing, Slicing Operations and with these operations, we can also perform different type of operations by using pre-defined functions present in str object.

Pre-defined Functions in str object

----

1) capitalize()

=>This Function is used for capitalizing the first letter First word of a given Sentence only.
=>Syntax: strobj.capitalize()

```
                         (OR)
              strobj=strobj.capitalize()
```

Examples:

```
>>> s="python"
>>> print(s,type(s))-------------------python <class 'str'>
>>> s.capitalize()--------------------'Python'
>>> s="python is an oop lang"
>>> print(s,type(s))------------------------python is an oop lang <class
'str'>
>>> s.capitalize()-----------------------------'Python is an oop lang'
------------------------------------
>>> s="python"
>>> print(s,type(s))--------------------python <class 'str'>
>>> s.capitalize()--------------------'Python'
>>> print(s,type(s))----------------python <class 'str'>
>>> s=s.capitalize()
>>> print(s,type(s))-----------------Python <class 'str'>
```

----
2) title():

=>This is used for obtaining Title Case of a Given Sentence (OR) Making all words First
    Letters are capital.

```
Syntax:          s.title()
                      (OR)
                      s=s.title()
```

Examples:

```
>>> s="python"
>>> print(s,type(s))-------------------python <class 'str'>
>>> s.capitalize()---------------------'Python'
>>> s.title()-----------------------------'Python'
----------------------------------------------------------
>>> s="python is an oop lang"
>>> print(s,type(s))------------------python is an oop lang <class 'str'>
>>> s.capitalize()--------------------'Python is an oop lang'
>>> s.title()----------------------------'Python Is An Oop Lang'
>>> print(s)----------------------------python is an oop lang
>>> s=s.title()
>>> print(s)--------------------------Python Is An Oop Lang
```

----
3) index()

=>This Function obtains Index of the specified Value
=>If the specified value does not exist then we get ValueError
=>Syntax: strobj.index(Value)
=>Syntax: indexvalue=strobj.index(value)

Examples:

```
>>> s="python"
>>> s.index("p")------------------0
>>> s.index("y")--------------------1
>>> s.index("o")-----------------4
>>> s.index("n")----------------5
>>> s.index("K")----------------ValueError: substring not found
```

=>enumerate() is one the general function, which is used for finding Index and Value of anu
       Iterable object.

NOTE:

```
>>> for i,v in  enumerate(s):
...     print("Index:{} and Value:{}".format(i,v))
```

OUTPUT

```
                      Index:0 and Value:p
                      Index:1 and Value:y
                      Index:2 and Value:t
                      Index:3 and Value:h
                      Index:4 and Value:o
                      Index:5 and Value:n
```

```
-----------------------------
>>> lst=[10,"Rossum",23.45,True]
>>> for i,v in  enumerate(lst):
...     print("Index:{} and Value:{}".format(i,v))
---------------
```

OUTPUT

```
---------------
                      Index:0 and Value:10
                      Index:1 and Value:Rossum
                      Index:2 and Value:23.45
                      Index:3 and Value:True
```

----

4) upper()

=>It is used for converting any type of Str Data into Upper Case.

=>Syntax:- strobj.upper()
OR
strobj=strobj.upper()

Examples:

```
-----------------
>>> s="python"
>>> print(s)------------------------------python
>>> s.upper()-----------------------'PYTHON'
>>> s="python is an oop lang"
>>> print(s)---------------------------------python is an oop lang
>>> s.upper()-------------------------------'PYTHON IS AN OOP LANG'
>>> s="Python IS  an OOP lang"
>>> print(s)-------------------------------Python IS  an OOP lang
>>> s.upper()-------------------------'PYTHON IS  AN OOP LANG'
>>> s="AbCdEf"
>>> print(s)------------------------AbCdEf
>>> s.upper()----------------------'ABCDEF'
>>> s="PYTHON"
>>> print(s)--------------------PYTHON
>>> s.upper()-----------------'PYTHON'
>>> s="123"
>>> print(s)------------------123
>>> s.upper()---------------'123'
```

----

5) lower()

=>It is used for converting any type of Str Data into lower Case.

=>Syntax:- strobj.lower()
OR
strobj=strobj.lower()

Examples:

```
-----------------
>>> s="Data Science"
>>> print(s)--------------Data Science
>>> s.lower()------------'data science'
>>> s="python"
>>> print(s)-------------python
>>> s.lower()-----------'python'
>>> s="PYTHON"
>>> print(s)-------------PYTHON
>>> s.lower()------------'python'
>>> s="PYThon"
>>> print(s)----------PYThon
>>> s.lower()---------'python'
```

---

6) isupper()

=>This Function returns True provided the given str object data is purely Upper Case otherwise it returns False.

Syntax: strobj.isupper()

Examples:

```
-----------------
>>> s="PYTHON"
>>> s.isupper()-----------True
>>> s="python"
>>> s.isupper()----------False
>>> s="Python"
>>> s.isupper()----------False
>>> s="PYThon"
>>> s.isupper()----------False
>>> s="123"
>>> s.isupper()------------False
>>> s="%$#^&@"
>>> s.isupper()-----------False
```

---

7)islower()

=>This Function returns True provided the given str object data is purely lower Case otherwise it returns False.

Syntax: strobj.islower()

Examples:

```
-----------------
>>> s="pythopn"
>>> s.islower()------------True
>>> s="pythOn"
>>> s.islower()------------False
>>> s="PYTHON"
>>> s.islower()-----------False
>>> s="123"
>>> s.islower()----------False
```

---

8) isalpha()

=>This Function returns True provided str object contains Purely Alphabets otherwise returns False.

Syntax: strobj.isalpha()

Examples:

```
-------------------
>>> s="Ambition"
>>> s.isalpha()--------------------True
>>> s="Ambition123"
>>> s.isalpha()-------------------False
>>> s="1234"
>>> s.isalpha()------------------False
>>> s="   "
>>> s.isalpha()------------------False
>>> s="#$%^@"
>>> s.isalpha()-----------------False
>>> s="AaBbZz"
>>> s.isalpha()----------------True
```

---

9) isdigit()

=>This Function returns True provided given str object contains purely digits otherwise returns False

Examples:

```
>>> s="python"
>>> s.isdigit()------------------False
>>> s="python123"
>>> s.isdigit()----------------False
>>> s="123"
>>> s.isdigit()-----------------True
>>> s="123 456"
>>> s.isdigit()---------------False
>>> s="1_2_3"
>>> s.isdigit()---------------False
>>> s="123KV"
>>> s.isdigit()-------------False
```

---

10) isalnum()

=>This Function returns True provided str object contains either Alpabets OR Numerics or Alpha-Numerics only otherwise It returns False.

=>Syntax: strobj. isalphanum()

=>Examples:

```
>>> s="python310"
>>> s.isalnum()-----------------True
>>> s="python"
>>> s.isalnum()-----------------True
>>> s="310"
>>> s.isalnum()-----------------True
>>> s="$python310"
>>> s.isalnum()------------------False
>>> s="python 310"
>>> s.isalnum()-----------------False
>>> s="$python3.10"
>>> s.isalnum()-----------------False
>>> s="python3.10"
>>> s.isalnum()-------------False
```

---

11) isspace()

=>This Function returns True provided str obj contains purely space otherwise it returns False.

=>Syntax: strobj.isspace()

Examples:

```
>>> s="  "
>>> s.isspace()-----------True
>>> s=""
>>> s.isspace()--------------False
>>> s="python Prog"
>>> s.isspace()-------------False
>>> s="Prasana Laxmi"
>>> s.isspace()--------------False
>>> s.isalpha()-----------False
>>> s.isalpha() or s.isspace()-----------False
```

---

12) split()

=>This Function is used for splitting the given str object data into different words base specified delimter (- _ # % ^ ^ , ; ....etc)
=>The dafeult deleimter is space
=>The Function returns Splitting data in the form of list object
=>Syntax:

```
strobj.split("Delimter")
        (OR)
strobj.split()
        (OR)
listobj=  strobj.split("Delimter")
        (OR)
listobj=strobj.split()
```

Examples:

```
>>> s="Python is an oop lang"
>>> print(s)----------------Python is an oop lang
>>> s.split()----------------['Python', 'is', 'an', 'oop', 'lang']
>>> len(s.split())-----------5
>>> x=s.split()
>>> print(x,type(x))---------['Python', 'is', 'an', 'oop', 'lang'] <class
'list'>
>>> len(x)---------------5
>>> s="12-09-2022"
>>> print(s)-------------12-09-2022
>>> s.split("-")----------['12', '09', '2022']
>>> s="12-09-2022"
>>> dob=s.split("-")
>>> print(dob,type(dob))------------['12', '09', '2022'] <class 'list'>
>>> print("Day",dob[0])----------Day 12
>>> print("Month ",dob[1])---------Month  09
>>> print("Year ",dob[2])----------Year  2022
------------------------------------------------------
>>> s="Apple#Banana#kiwi/Guava"
>>> words=s.split("#")
>>> print(words)-----------['Apple', 'Banana', 'kiwi/Guava']
>>> words=s.split("/")
>>> print(words)------------------['Apple#Banana#kiwi', 'Guava']
```

---

13) join():

=>This Function is used for combining or joining list of values from any Iterable object
=>Syntax: strobj.join(Iterableobject)
Examples:

```
------------------------------
>>> lst=["HYD","BANG","AP","DELHI"]
>>> print(lst,type(lst))------------------['HYD', 'BANG', 'AP', 'DELHI']
<class 'list'>
>>> s=""
>>> s.join(lst)---------------'HYDBANGAPDELHI'
>>> s=" "
>>> s.join(lst)------------------'HYD BANG AP DELHI'
-----------------------------------------------------------------
>>> t=("Rossum","is", "Father" "of" ,"Python")
>>> print(t,type(t))
('Rossum', 'is', 'Fatherof', 'Python') <class 'tuple'>
>>> k=" "
>>> k.join(t)
'Rossum is Fatherof Python'
```

```
>>> t=("Rossum","is", "Father", "of" ,"Python")
>>> k=" "
>>> k.join(t)
'Rossum is Father of Python'
>>>
```