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

Files in Python

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

Index

-----------

=>Purpose of Files

=>Types of Applications

a) Non-Persistant Applications

b) Persistant Applications

=>Definition of File,Stream

=>Operations on Files

a) Write Operation

b) Read Operations

=>File Opening Modes

1) r 2) w 3) a

4) r+ 5) w+ 6) a+

7) x 8) x+

=>Number of Approaches to Open the File

a) By Using open ()

b) By using " with open () as "

=>Writing the Data to the file

1) write ()

2) writelines()

=>Reading the data from the files

1) read ()

2) readlines()

=>Programming Examples

---------------------------------------------------------------------------------------------------------

=>Pickling (Object Serrialization) and Un-pickling (Object De-Serrialization)

=>pickle module

=>Implementation of pickling and un-pickling

=>Programming Examples

-----------------------------------------------------------------------------------------------------------

=>Workling CSV Files (Excel Sheet of MS)

=>csv module

=>Operations on CSV Files

a) csv.reader

b) csv.writer

c) csv.DictReader

d) csv.DictWriter

=>Programming Examples

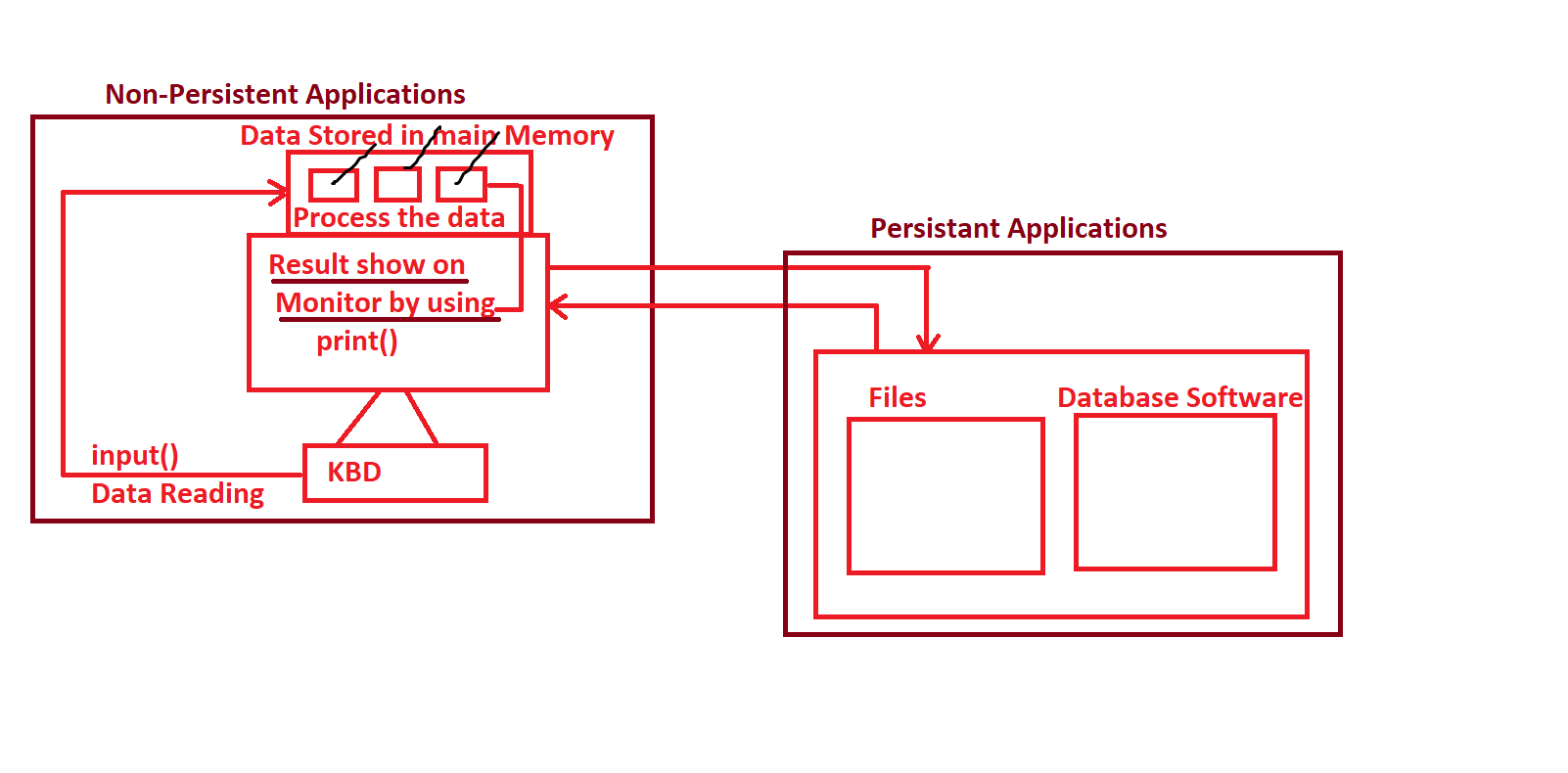
===========================================================

=>OS Modules

=>OS Operations

=>Programming Examples

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

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

Types of Application in Files

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

=>The purpose of Files in any programming language is that " To maintain Data Persistency".

=>The Process of storing the data permanently is called Data Persistency.

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

1) Non-Persistant Applications.

2) Persistant Applications.

=>In Non-Persistant Applications development, we read the data from Keyboard, stored in main memory (RAM) in the form objects, processed and whose results displayed on Moniter.

Examples: ALL our previous examples comes under Non-Persistant Applications.

=>We know that Data stored in Main Memory is temporary.

=>In Persistant Applications development, we read the data from Keyboard, stored in main memory(RAM) in the form objects, processed and whose results stored Permanently.

=>In Industry, we have two ways two store the Data Permanently. They are

1) By using Files

2) By Using DataBase Softwares (Oracle, MySQL, MongoDB, DB2, PostgreySQL, SQL Server,SQLITE3..etc)

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

Data Persistenecy by Files of Python

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

-------------------

Def. of File:

-----------------------------------------------------

=>A File is a collection of Records.

=>Files Resides in Secondary Memory (HDD).

=>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.

----------------------------------------------------------------------------------------------------------------------------------

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

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 are 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

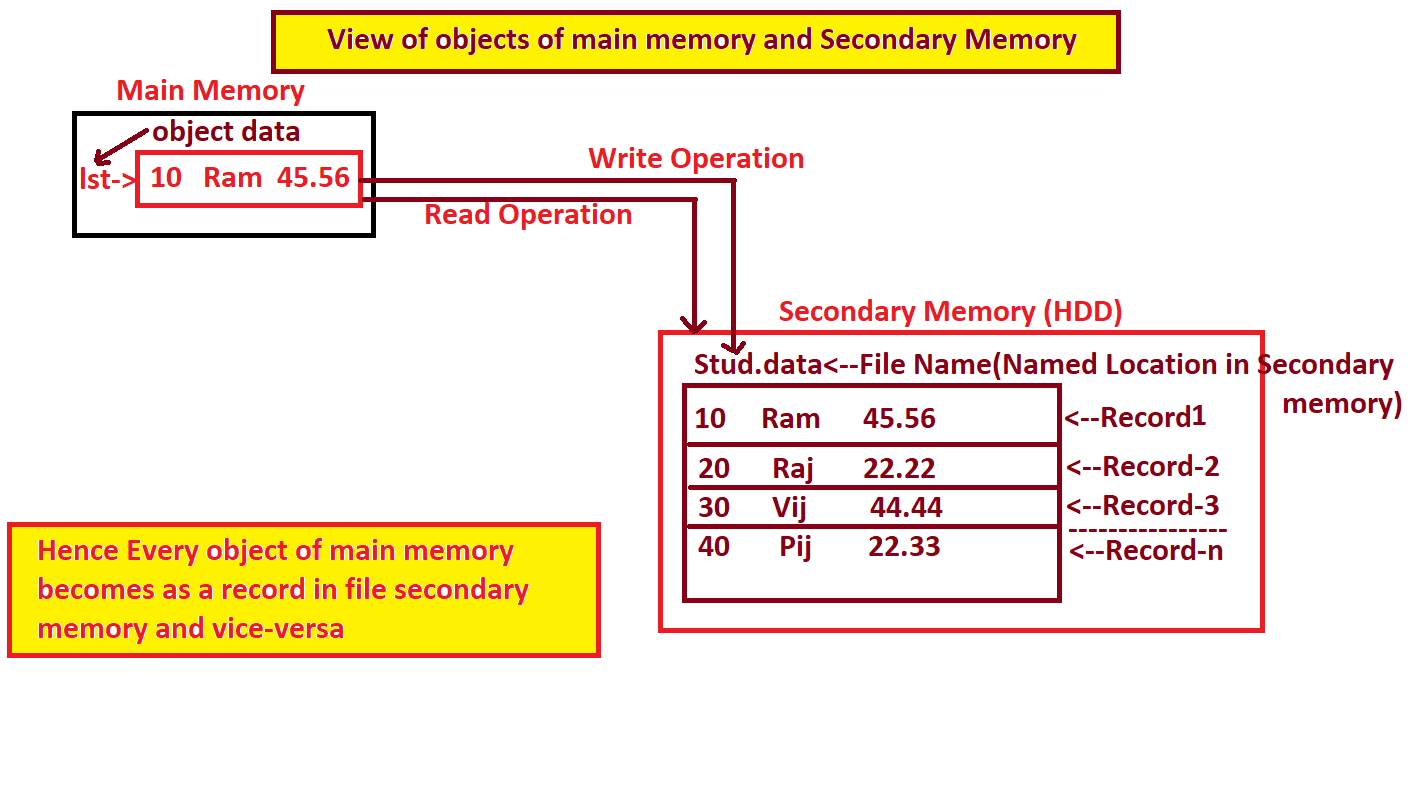
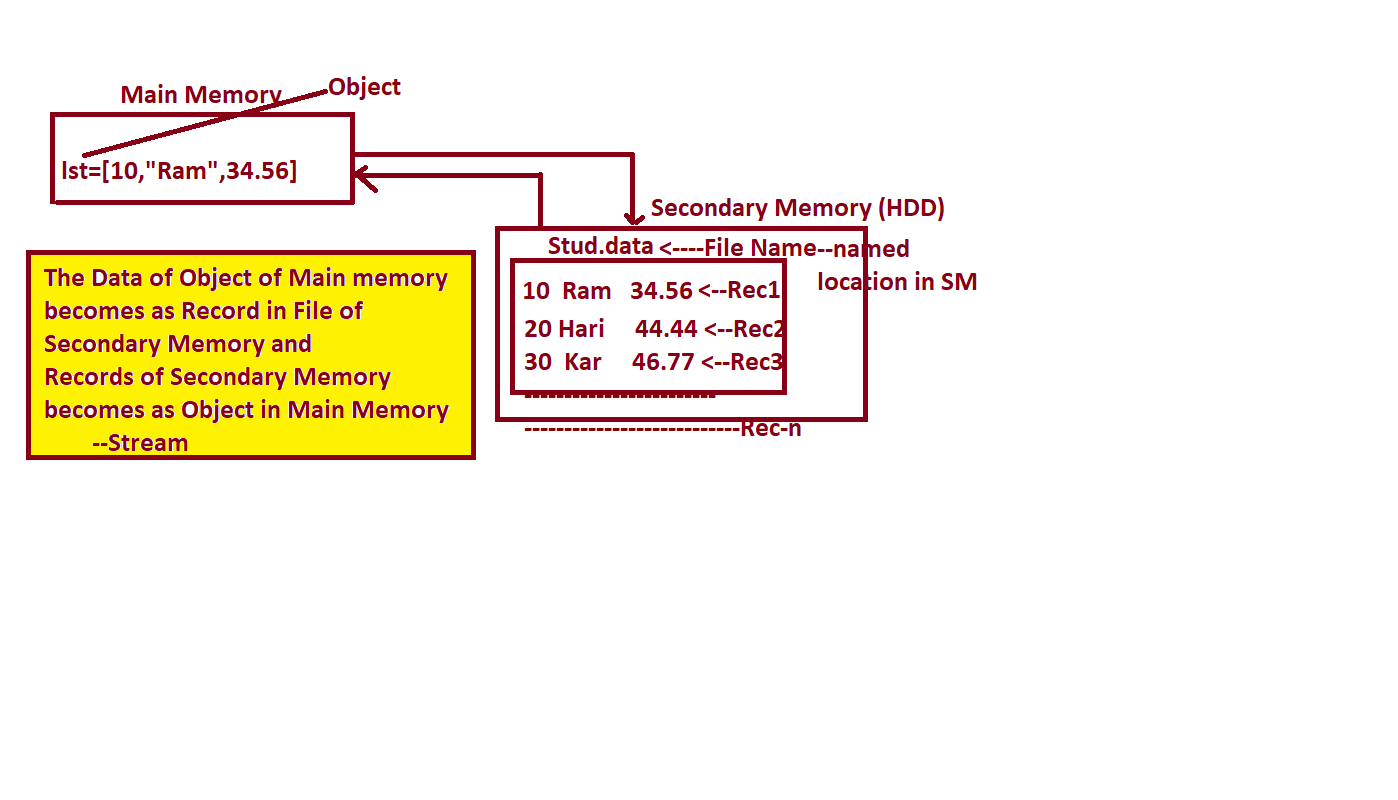
c) Perform cycle of read operations.

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

a) FileNotFoundError

b) EOFError

================================x=========================================

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

File Opening Modes

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

=>The purpose of File Opening Modes is that "In which type file mode we are opening the file".

=>In Python Programming, we have 8 File Opening Modes. They are

1. r 2. w 3. a

4. r+ 5. w+ 6. a+

7. x 8. x+

-----------------------------------------------------------------------------------------------

1) r

-----------------------------------------------------------------------------------------------

=>This mode is used for opening the file name in READ Mode.

=>If we open the file name in 'r' mode and if the file name does not exist then we get

FileNotFoundError.

=>"r" mode is default mode of all file opening modes.

-----------------------------------------------------------------------------------------------

2) w

-----------------------------------------------------------------------------------------------

=>This mode is used for Creating File Name and Opening the file name in WRITE Mode always.

=>If we are opening NEW FILE in “w " mode then that File Created Newly and Opened in write mode

=>If we are opening EXISTING FILE in "w" then EXISTING FILE Data OVERLAPPED with new Data.

-----------------------------------------------------------------------------------------------

3) a

-----------------------------------------------------------------------------------------------

=>This mode is used for Creating File Name and Opening the file name in WRITE Mode always.

=>If we are opening NEW FILE in "a" mode then that File Created Newly and Opened in write mode

=>If we are opening EXISTING FILE in "a" then EXISTING FILE Data APPENDED with new Data.

-----------------------------------------------------------------------------------------------

4) r +

-----------------------------------------------------------------------------------------------

=>This mode is used for opening the file name in READ Mode.

=>If we open the file name in 'r+' mode and if the file name does not exist then we get

FileNotFoundError.

=>With "r+" mode, first we Must read the data and Later we can perform Write Operation.

----------------------------------------------------------------------

5) w +

-----------------------------------------------------------------------

=>This mode is used for Creating File Name and Opening the file name in WRITE Mode always.

=>If we are opening NEW FILE in “w" mode then that File Created Newly and Opened in write mode

=>If we are opening EXISTING FILE in "w" then EXISTING FILE Data OVERLAPPED with new Data.

=>With "w+" mode, additionally we can perform Read Operation After Performing Write Operation

-----------------------------------------------------------------------

6) a+

-----------------------------------------------------------------------

=>This mode is used for Creating File Name and Opening the file name in WRITE Mode always.

=>If we are opening NEW FILE in “a+" mode then that File Created Newly and Opened in write mode

=>If we are opening EXISTING FILE in "a+" then EXISTING FILE Data APPENDED with new Data.

=>With "a+" mode, additionally we can perform Read Operation After Performing Write Operation.

-----------------------------------------------------------------------

7) x

-----------------------------------------------------------------------

=>This mode is used for Creating File Name and exclusively Opening the file name in WRITE Mode always.

=>Once we open File Name in "x" mode, we can Perform Write Operations only.

=>If we open existing file in "x" mode then we get FileExistError

-----------------------------------------------------------------------

8) x+

----------------------------------------------------------------------

=>This mode is used for Creating File Name and exclusively Opening the file name in WRITE Mode always.

=>Once we open File Name in "x+" mode, we can Perform Write Operations First and Letter we can Perform Read Operations also

=>If we open existing file in "x+” then we get FileExistError

-----------------------------------------------------------------------

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

Opening Files

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

=>To perform any type of Operation, we must open the file.

=>In Python Programming, to open the file, we have two approaches. They are

1. By using open ()

2. By using " with open() as "

-------------------------------------------------------------------------------------------------------------------

1) By using open ()

-------------------------------------------------------------------------------------------------------------------

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

--------------------

Explanation

--------------------

=>varname represents File Pointer and always points to the content of file and it is an object of type TextIOWrapper

=>open () is one of the pre-defined Function, which is used for opening the specified File Name in Specified File Mode

=>File Name represents Name of the file

=>File Mode represents r,w,a,r+,w+,a+,x,x+

=>When we open the file with open() then it is recommended to close the file manually by using close(). In otherwords open() does not provide auto-closability of Files.

-------------------------------------------------------------------------------------------------------------------

2) By using " with open() as "

-------------------------------------------------------------------------------------------------------------------

Syntax:- with open("File Name","File Mode") as Varname:

----------------------------------------------------------

----------------------------------------------------------

----------------------------------------------------------

----------------------------------------------------------

-------------------------------------------------------------------

Other statemenets out-of "with open() as " statements

--------------------------------------------------------------------

Explanation:

-------------------

=>Here "with" and "as" are are the key words

=>open() is one of the pre-defined Function, which is used for opening the specified File Name in Specified File Mode

=>varname represents File Pointer and always points to the content of file and it is an object

of type TextIOWrapper

=>File Name represents Name of the file

=>File Mode represents r,w,a,r+,w+,a+,x,x+

=>The advantage of " with open() as " is that "Auto-Closability of File. In Otherwords, as long as PVM is executing Indentation block of " with open() as " then File Name is Active(Open) and Once PVM comes out-of Indentation block of " with open() as" then File Name closed automatically (no need to use close() ).

==============================X========================================

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

Writing the data to the File

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

=>To write the data to the file, we have 2 pre-defined Functions in file pointer object. They are

1) write ()

2) writelines ()

-----------------------------------------------------------------------

1)write ()

-----------------------------------------------------------------------

=>Syntax: filepointerobj.write(strdata)

=>This Function is used for writing any type of Data to the file in the form of str

=>Examples: refer FileWriteEx1.py

------------------------------------------------------------------------------------------------------------------

2)writelines ()

------------------------------------------------------------------------------------------------------------------

=>Syntax: filepointerobj.writelines(Iterable-object)

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

=>Examples: refer FileWriteEx2.py

------------------------------------------------------------------------------------------------------------------

#Program for demonstarting to open the file name in different modes

#FileOpenEx1.py

try:

fp=open("stud1.data") # Here we ar eopening "stud.data" in read mode.

except FileNotFoundError:

print ("File does not exist")

else:

print("File Opened Successfully in Read Mode:")

print("Type of fp=",type(fp)) # Type of fp= <class '\_io.TextIOWrapper'>

print("Is File Closed--else block:", fp.closed)

print("File Mode Name=",fp.mode)

finally:

print("\nI am from finally block:")

fp.close() # manual closing of file

print("Is File Closed in finally block:", fp.closed)

#Program for demonstarting to open the file name in different modes

#FileOpenEx2.py

fp=open("stud1.data","w")

print("File Created Successfully in Write Mode:")

print("Type of fp=",type(fp))

#Program for demonstarting to open the file name in different modes

#FileOpenEx3.py

try:

with open("stud3.data","r") as fp:

print ("File Opened Successfully in Read Mode:")

print ("Type of fp=",type(fp)) # Type of fp= <class '\_io.TextIOWrapper'>

print("Is File Closed--within with-open() as Indentation block:", fp.closed) # False

print("\ni am out-of with-open() as Indentation block:")

print("Is File Closed--out-of with-open() as Indentation block:", fp.closed) # True

except FileNotFoundError:

print("File does not exist:")

#Program for demonstarting to open the file name in different modes

#FileOpenEx4.py

with open("stud4.data","a+") as fp:

print("-"\*50)

print("File Opened in Write Mode:")

print("Name of File =",fp.name)

print("File Opening Mode=",fp.mode)

print("Is File Writable=",fp.writable())

print("Is File Readable=",fp.readable())

print("-"\*50)

#Program for demonstarting to open the file name in different modes

#FileOpenEx5.py

try:

with open("kvr.data","x") as fp:

print("-"\*50)

print("File Opened in Write Mode:")

print("Name of File =",fp.name)

print("File Opening Mode=",fp.mode)

print("Is File Writable=",fp.writable())

print("Is File Readable=",fp.readable())

print("-"\*50)

except FileExistsError:

print("File Name alerady Exist:")

#Program for demonstarting to open the file name in different modes

#FileOpenEx6.py

try:

with open("hyd.data","x+") as fp:

print("-"\*50)

print("File Opened in Write Mode:")

print("Name of File =",fp.name)

print("File Opening Mode=",fp.mode)

print("Is File Writable=",fp.writable())

print("Is File Readable=",fp.readable())

print("-"\*50)

except FileExistsError:

print("File Name alerady Exist:")

#Program for writing Address of a Person to file file---write()

#FileWriteEx1.py

with open("Addr1.data","a") as fp:

fp.write("Travis El Olihpant\n")

fp.write("FNO:13-14, Hill Side\n")

fp.write("Numpy Foundation\n")

fp.write("Nethet Lands-56\n")

print ("Data Written to the file")

#Program for writing Address of a Person to file file---writelines()

#FileWriteEx2.py

obj={10,20,30,10,20,30}

with open("Addr2.data","a") as fp:

fp.writelines(str(obj)+"\n")

print ("Data Written to the file")

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

Reading the Data from File

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

=>To read the data from the file, we have 2 pre-defined functions present File Pointer object. They are

1) read ()

2) readlines()

-----------------------------------------------------------------------

1) read ()

--------------------------------------------------------------------------------------------------------------------------

=>Syntax: - varname=filepointerobj.read()

=>This Function is used for reading the entire data from file and placed in LHS Varname in the form of str.

----------------

Examples:

----------------

--------------------------------------------------------------------------------------------------------------------------

2) readline()

--------------------------------------------------------------------------------------------------------------------------

=>Syntax: - varname=filepointerobj.readlines()

=>This Function is used for reading the entire data from file the form of Line by line and placed in LHS Varname in the form of list object.

------------------

Examples:

------------------

--------------------------------------------------------------------------------------------------------------------------

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

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 with Images.

#Write a python program which will count number of lines , number of words and number of characters in any file.

#FileCountInfo.py

try:

fname=input("Enter File Name:")

nl=0

nw=0

nc=0

with open(fname,"r") as fp:

lines=fp.readlines()

for line in lines:

print(line,end="")

nl=nl+1

nw=nw+len(line.split())

nc=nc+len(line)

else:

print('-'\*40)

print('Number of Lines:{}'.format(nl))

print("Number of Words:{}".format(nw))

print("Number of Chars:{}".format(nc))

print('-'\*40)

except FileNotFoundError:

print("File does not Exist");

#Write a python program which will copy an image

#ImageFileCopyEx1.py

sfile=input ("Enter Source File:")

try:

with open(sfile,"rb") as rp:

dfile=input("Enter Destination File:")

with open(dfile,"wb") as wp:

sfiledata=rp.read()

wp.write(sfiledata)

print("\nSource File Data Copied into Destination File:")

except FileNotFoundError:

print("File does not Exist");

#Write a python program which will copy the content of one file into another file (text file)

#TextFileCopyEx1.py

sfile=input("Enter Source File:")

try:

with open(sfile,"r") as rp:

dfile=input("Enter Destination File:")

with open(dfile,"a") as wp:

sfiledata=rp.read()

wp.write(sfiledata)

print("\nSource File Data Copied into Destination File:")

except FileNotFoundError:

print("File does not Exist");

# Write a python program which will read the data from the file and display on the console

#FileReadEx1.py-----read()

try:

filename=input("Enter any file name:")

with open(filename) as fp:

print("Initila Position of fp=",fp.tell())

print("-"\*50)

filedata=fp.read()

print(filedata)

print("-"\*50)

print("Now Position of fp=",fp.tell())

except FileNotFoundError:

print("File does not Exist");

# Write a python program which will read the data from the file and display on the console

#FileReadEx2.py-----readline()

try:

filename=input("Enter any file name:")

with open(filename) as fp:

print("Initial Position of fp=",fp.tell())

print("-"\*50)

lines=fp.readlines()

for line in lines:

print(line,end="")

print("-"\*50)

print("Now Position of fp=",fp.tell())

except FileNotFoundError:

print("File does not Exist");

# Write a python program which demonstartes tell() and seek()

#RandomFileAccessEx.py----tell() and seek()

try:

with open("addr1.data") as fp:

print("Initila Position of fp before reading=",fp.tell()) # 0

print(fp.read())

print("Position of fp after reading=",fp.tell()) # ----163

print("-"\*40)

fp.seek(6)

print("Now after Seek--Position of fp before reading=",fp.tell()) #

print(fp.read(10))

print("Position of fp after reading=",fp.tell()) # 16

except FileNotFoundError:

print("File does not Exist");

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

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 () saves 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 transfer the entire record content from file of secondary memory, if we use read(), 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(filepointer)

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

#WAPP which will read emp no, emp name and salary and save then in a file---Program--(A)

#EmpPickEx1.py

import pickle,sys

with open("emp.data","ab") as fp:

while(True):

print("-"\*50)

#read emp values from key board

eno=int(input("Enter Employee Number:"))

ename=input("Enter Employee Name:")

sal=float(input("Enter Emp Salary:"))

#place emp values in Iterable Object--list

l1=list() #empty list

l1.append(eno)

l1.append(ename)

l1.append(sal)

#Save list object data into file

pickle.dump(l1,fp)

print("-"\*50)

print("\tEmplyee Record Saved Successfully in File:")

print("-"\*50)

ch=input("Do u want to insert another record(yes/no):")

if(ch.lower()=="no"):

print("Thx for using this program")

sys.exit()

#WAPP which will read emp records from the file----Program--(B)

#EmpUnPick.py

import pickle

try:

with open("emp.data","rb") as fp:

print("-"\*50)

print("\tEmpno\tName\tSal")

print("-"\*50)

while(True):

try:

obj=pickle.load(fp) # Which ever object was pickled and same type of object would be un-pickled

for val in obj:

print("\t{}".format(val),end="")

print()

except EOFError:

print("-"\*50)

break

except FileNotFoundError:

print("File does not exist")

#Write a pyhton program which will accept number of student in the college.Accept individual details of student such as student number, student name and marks save the student details as a record in a file

#StudPick.py

import pickle

def savestudrecords():

nos=int(input("Enter How many students u have:"))

if(nos<=0):

print("{} is invalid number of students:".format(nos))

else:

with open ("stud.data","ab") as fp:

for i in range(1,nos+1):

print("-"\*50)

print("Enter {} Student Information:".format(i))

print("-"\*50)

sno=int(input("Enter Student Number:"))

sname=input("Enter Student Name:")

marks=float(input("Enter Student Marks:"))

l=list();

l.append(sno)

l.append(sname)

l.append(marks)

pickle.dump(l,fp);

print("-"\*50)

print("{} Student Record Saved in a File".format(i))

#main program

savestudrecords()

#write a python program which will read the student record from the file

#StudUnpick.py

import pickle,sys

try:

with open("stud.data","rb") as fp:

print("-"\*50)

print('\tStno\tName\tMarks')

print("-"\*50)

while(True):

try:

studrec=pickle.load(fp)

for val in studrec:

print("\t{}".format(val),end="")

else:

print()

except EOFError:

print("-"\*50)

break

except FileNotFoundError:

print("File does not exist")

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

Working with CSV Files in Python

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

=>CSV stannds for Comma Separated Values.

=>A CSV File is one of the is a simple file format used to store tabular data, such as a

spreadsheet or database.

=>A CSV file stores tabular data (numbers and text) in plain text.

=>Each line of the CSV file is a data record. Each record consists of one or more fields,

separated by commas.

=>Python provides an in-built module called csv to work with CSV files.

=>There are 2 classes provided by this module for writing the data to CSV File. They are

1) By using Using csv.writer class object

2) By Using csv.DictWriter class object

---------------------------------------------------

1) By using 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)

# importing the csv module

#csvdictwriteex1.py

import csv

# my data rows as dictionary objects

mydict =[ {'branch': 'COE', 'cgpa': '9.0', 'name': 'Nikhil', 'year': '2'},

{'branch': 'COE', 'cgpa': '9.1', 'name': 'Sanchit', 'year': '2'},

{'branch': 'IT', 'cgpa': '9.3', 'name': 'Aditya', 'year': '2'},

{'branch': 'SE', 'cgpa': '9.5', 'name': 'Sagar', 'year': '1'},

{'branch': 'MCE', 'cgpa': '7.8', 'name': 'Prateek', 'year': '3'},

{'branch': 'EP', 'cgpa': '9.1', 'name': 'Sahil', 'year': '2'} ]

# field names

csvfields = ['name', 'branch', 'year', 'cgpa']

# name of csv file

filename = "univ2.csv"

# writing to csv file

with open(filename, 'w') as fp:

# creating a csv dict writer object

dictwriter = csv.DictWriter(fp, fieldnames = csvfields)

# writing headers (field names)

dictwriter.writeheader()

# writing data rows

dictwriter.writerows(mydict)

print("\nDict Data Written to the csv file--verify")

#Program for Reading the Data from CSV File

#CsvReadEx1.py

try:

with open("E:\\KVR-PYTHON-11AM\\FILES\\NOTES\\student.csv","r") as fp:

csvdata=fp.read()

print(csvdata)

except FileNotFoundError :

print("CSV File does not exist:")

#Note: This Type of Reading the data from File is called NON-CSV Reading

# Python program to demonstrate writing to CSV File

#csvwriteex1.py

import csv

# field names OR Header Names

recfields = ['Name', 'Branch', 'Year', 'CGPA']

# data rows of csv file

rows = [ ['Nikhil', 'COE', '2', '9.0'],

['Sanchit', 'COE', '2', '9.1'],

['Aditya', 'IT', '2', '9.3'],

['Sagar', 'SE', '1', '9.5'],

['Prateek', 'MCE', '3', '7.8'],

['Sahil', 'EP', '2', '9.1'] ]

# name of csv file

csvfilename = "univ1.csv"

# writing data to csv file

with open(csvfilename, 'w') as fp:

# creating a csv writer object

csvwriter = csv.writer(fp)

# writing the fields

csvwriter.writerow(recfields)

# writing the data rows

csvwriter.writerows(rows)

print("\nCSV file Created and Verify")

# Python program to demonstrate to write single record

# writing single record to CSV file

#csvwriteex2.py

import csv

# data record of csv file

row = ('Rajan', 'ECE', '3', '9.9')

# name of csv file

filename = "univ1.csv"

# writing to csv file

with open(filename, 'a') as fp:

# creating a csv writer object

cw = csv.writer(fp)

# writing the data row to the csv file

cw.writerow(row)

print("\nSingle Record Written to the CSV File:")

#Program for Reading the Data from CSV File

#To read the data from csv file, we must create an object csv.reader object

# csv.reader(filepointer) which gives an object of csv.reader

#PureCsvReadEx1.py

import csv

try:

with open("E:\\KVR-PYTHON-11AM\\FILES\\NOTES\\student.csv","r") as fp:

csvr=csv.reader(fp)

for record in csvr:

for val in record:

print("\t{}".format(val),end="")

print()

except FileNotFoundError :

print("CSV File does not exist:")

#Program for Reading the Data from CSV File

#To read the data from csv file, we must create an object csv.reader object

# csv.reader(filepointer) which gives an object of csv.reader

#PureCsvReadEx2.py

import csv

try:

with open("univ1.csv","r") as fp:

csvr=csv.reader(fp)

for record in csvr:

for val in record:

print("\t{}".format(val),end="")

print()

except FileNotFoundError :

print("CSV File does not exist:")

#Program for Reading the Data from CSV File

#To read the data from csv file, we must create an object csv.reader object

# csv.reader(filepointer) which gives an object of csv.reader

#PureCsvReadEx3.py

import csv

try:

with open("univ2.csv","r") as fp:

dictcsvr=csv.DictReader(fp)

for record in dictcsvr:

for key,val in record.items():

print("\t{}-->{}".format(key,val))

print()

except FileNotFoundError :

print("CSV File does not exist:")

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

Python DataBase Communication ( PDBC )

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

=>Even we acheived the Data Persistency by using Files, Files has the following Limitations.

1. Files of any language does not contain security bcoz Files are unable to provide security in the form of User Name and Password.

2. Files are unable to store large amount of data

3. File are differing from One OS to another OS (Files are OS depenedent)

4. Querying and Processing the data from Files is Very Complex bcoz file data is organized w.r.t Indices and idenfying the indices is very complex.

5. Files does not contain Column Names (Except CSV Files)

=>To Overcome the limitation of files and to acheive the Data Persistency, we must use the concept of any RDBMS DataBase Softwares ( Oracle, MYSQL, Mongo DB, DB2, SQL Server, Postgey SQL, SQLITE3...........etc).

1. All RDBMS DataBase Softwares Provides Security bcoz RDBMS DataBase

Softwares considers User names and Password.

2. All RDBMS DataBase Softwares stores large amount of data

3. All RDBMS DataBase Softwares Arch Remains Same on all types of OSes ( OS

Independent)

4. Querying and Processing the data from All RDBMS DataBase Softwares is Very

Simple bcoz data of All RDBMS DataBase Softwares oranganized in the of Tables with Column Names.

5. The Data Present in any RDBMS DataBase Softwares oranganized in the of Tables

with Column Names.

=>If Python Program want to communicate with any RDBMS DataBase Softwares then we must use a PRE-DEFINED MODULE and such PRE-DEFINED MODULE does not exist in Python Software.

=>Some Third Party Software Vendors(Ex: "Anthony Tuininga") developed a Module for Python Programmers to communicate with RDBMS DataBase Softwares and placed in github and Third Party Software Modules must be installed.

=>To install any Third Party Software Modules in python , we use a tool called pip and it is present in C:\Users\KVR\AppData\Local\Programs\Python\Python310\Scripts folder.

=>Syntax: pip install Module Name (at any Windows command prompt)

=>If Python Program want to communicate with Oracle Database, then we must install

cx\_Oracle Module.

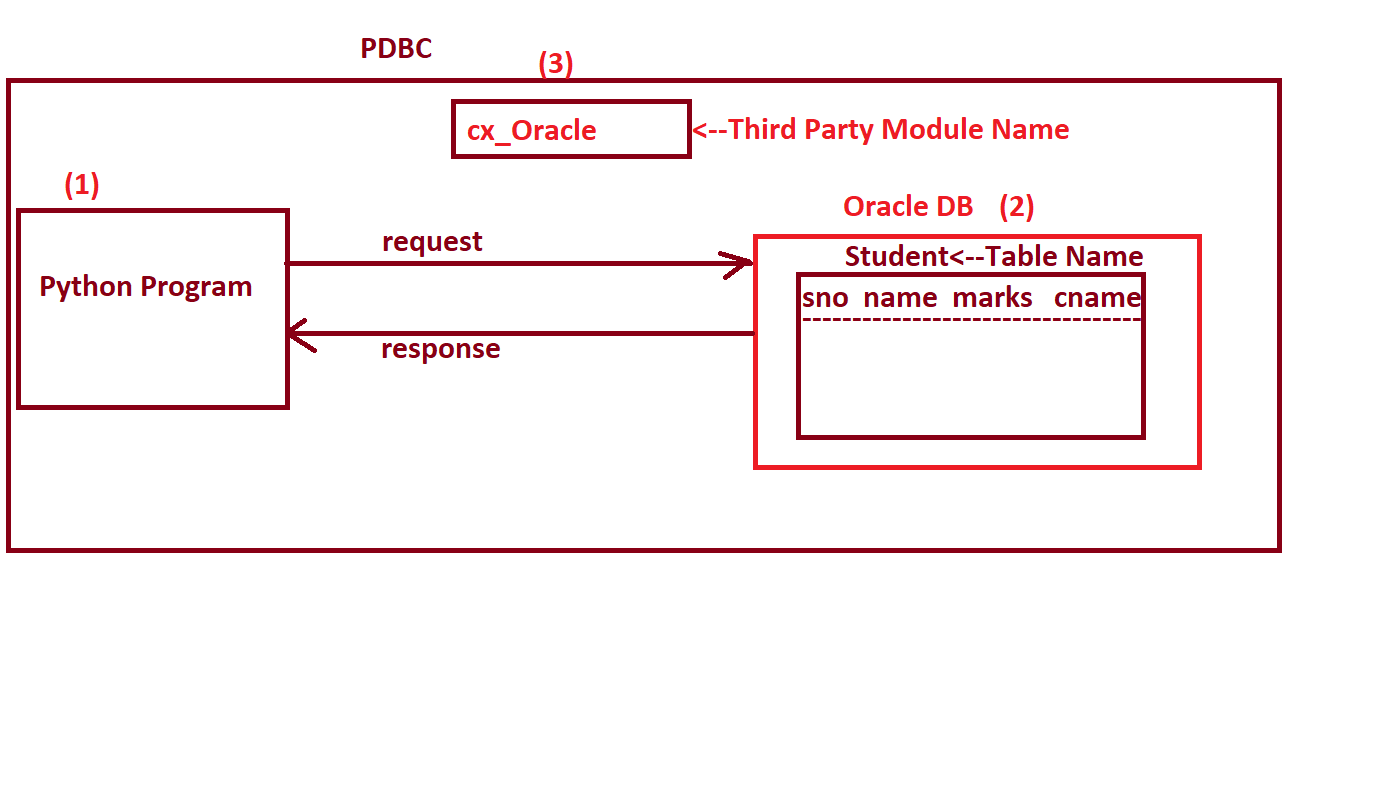
=>Examples : pip install cx\_Oracle

=>If Python Program want to communicate with MySQL Database, then we must install

mysql-connector or mysql-connector-python Module.

=>Examples: pip install mysql-connector

=>Examples: pip install mysql-connector-python



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

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 of 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 Name 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 from Python Program, 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 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 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.

#Write a python program which will obtains the connection from oracle database.

#TestOracleConEx1.py

import cx\_Oracle # step-1

try:

con=cx\_Oracle.connect("scott1/tiger@localhost/orcl") # Step-2

print("\ntype of con=",type(con))

print("Python Program obtains connection from Oracle DB")

except cx\_Oracle.DatabaseError as d:

print(d)

#Write a python program which will obtains the connection from oracle database.

#TestOracleConEx2.py

import cx\_Oracle # step-1

try:

con=cx\_Oracle.connect("scott/tiger@127.0.0.1/orcl") # Step-2

print("\ntype of con=",type(con))

print("Python Program obtains connection from Oracle DB")

except cx\_Oracle.DatabaseError as d:

print(d)

#Write a python program which will create an object of cursor.

#CursorEx1.py

import cx\_Oracle # 1

try:

con=cx\_Oracle.connect("scott/tiger@localhost/orcl") # 2

print("\nConnection Obtained from Oracle DB")

cur=con.cursor() # 3

print("\ntype of cur=",type(cur))

print("Cursor object created:")

except cx\_Oracle.DatabaseError as db:

print(db)

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

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

#Write a python program which will add new column name called company name to employee table

#AlterwithAddEx.py

import cx\_Oracle

def alterwithadd():

try:

con=cx\_Oracle.connect("scott/tiger@127.0.0.1/orcl")

cur=con.cursor()

aq="alter table employee add(cname varchar2(10) not null)"

cur.execute(aq)

print("Employee Table Altered")

except cx\_Oracle.DatabaseError as db:

print("Problem in Database:",db)

#main programs

alterwithadd()

#Write a python Program Which will Change the column sizes of employee table(eno,ename)

#AlterwithModifyEx.py

import cx\_Oracle

def alterwithmodify():

try:

con=cx\_Oracle.connect("scott/tiger@127.0.0.1/orcl")

cur=con.cursor()

aq="alter table employee modify(eno number(3),name varchar2(15))"

cur.execute(aq)

print("Employee Table Altered")

except cx\_Oracle.DatabaseError as db:

print("Problem in Database:",db)

#main programs

alterwithmodify()

#Write a python program which will create employee table in oracle database.

#TableCreateEx1.py

import cx\_Oracle

def tablecreate():

try:

con=cx\_Oracle.connect("scott/tiger@localhost/orcl")

cur=con.cursor()

#prepare the query and execute

ctq="create table employee(eno number(2) primary key, name varchar2(10) not null,sal number(5,2) not null )"

cur.execute(ctq)

print("Emplyee Table Created Successfully")

except cx\_Oracle.DatabaseError as db:

print("Problem in Database:",db)

#main program

tablecreate()

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

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 for column-n)

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;

#Write a python program which will access employees number employees name,salary and disignation from keyboard and insert as a record in employee table

#RecordInsertEx1.py

import cx\_Oracle

def insertRecord():

try:

con=cx\_Oracle.connect("scott/tiger@localhost/orcl")

cur=con.cursor()

iq="insert into employee values(40,'Kinney',1.2,'Pandas')"

cur.execute(iq)

con.commit()

print("Record Inserted Successfully in Employee Table")

except cx\_Oracle.DatabaseError as db:

print("Problem in Database:",db)

#main program

insertRecord()

#Write a python program which will access employees number employees name,salary and disignation from keyboard and insert as a record in employee table

#RecordInsertEx2.py

import cx\_Oracle

def insertRecord():

try:

con=cx\_Oracle.connect("scott/tiger@localhost/orcl")

cur=con.cursor()

#accept employee values from KBD

eno=int(input("Enter Employee Number:"))

ename=input("Enter Employee Name:")

empsal=float(input("Enter Employee Salary:"))

cname=input("Enter Employee Company Name:")

#Prepare the query and execute

iq="insert into employee values(%d,'%s',%f,'%s' ) "

cur.execute("insert into employee values(%d,'%s',%f,'%s' ) " %(eno,ename,empsal,cname) )

#OR cur.execute(iq %(eno,ename,empsal,cname) )

con.commit()

print("{} Record Inserted Successfully in Employee Table".format(cur.rowcount ))

except cx\_Oracle.DatabaseError as db:

print("Problem in Database:",db)

except ValueError:

print("Don't eneter alnums,strs and symbols for empno, salary")

#main program

insertRecord()

#Write a python program which will access employees number employees name,salary and disignation from keyboard and insert as a record in employee table

#RecordInsertEx3.py

import cx\_Oracle,sys

def insertRecord():

while(True):

try:

con=cx\_Oracle.connect("scott/tiger@localhost/orcl")

cur=con.cursor()

print("-"\*50)

#accept employee values from KBD

eno=int(input("Enter Employee Number:"))

ename=input("Enter Employee Name:")

empsal=float(input("Enter Employee Salary:"))

cname=input("Enter Employee Company Name:")

#Prepare the query and execute

iq="insert into employee values(%d,'%s',%f,'%s' ) "

cur.execute("insert into employee values(%d,'%s',%f,'%s' ) " %(eno,ename,empsal,cname) )

#OR cur.execute(iq %(eno,ename,empsal,cname) )

con.commit()

print("-"\*50)

print("{} Record Inserted Successfully in Employee Table".format(cur.rowcount ))

print("-"\*50)

ch=input("Do u want to insert another record(yes/no):")

if(ch.lower()=="no"):

print("Thx for using this program")

sys.exit()

print("-"\*50)

except cx\_Oracle.DatabaseError as db:

print("Problem in Database:",db)

except ValueError:

print("Don't eneter alnums,strs and symbols for empno, salary")

#main program

insertRecord()

#Write a python program which will delete the record from employee table based employee number

#RecordDeleteEx.py

import cx\_Oracle,sys

def deleteRecord():

while(True):

try:

con=cx\_Oracle.connect("scott/tiger@localhost/orcl")

cur=con.cursor()

print("-"\*50)

#accept employee values from KBD

empno=int(input("Enter Employee Number for deleting the Record:"))

#Prepare the query and execute

cur.execute("delete from employee where eno=%d" %empno)

con.commit()

if(cur.rowcount>0):

print("{} Record deleted Succeessfully".format(cur.rowcount))

else:

print("Employee Record does not exist:")

print("-"\*50)

ch=input("Do u want to delete another record(yes/no):")

if(ch.lower()=="no"):

print("Thx for using this program")

sys.exit()

print("-"\*50)

except cx\_Oracle.DatabaseError as db:

print("Problem in Database:",db)

except ValueError:

print("Don't eneter alnums,strs and symbols for empno, salary")

#main program

deleteRecord()

#Write a python program which will update salary by 50% for those employee whose salary is less than 1.0

#RecordUpdateEx.py

import cx\_Oracle

def updateRecord():

try:

con=cx\_Oracle.connect("scott/tiger@localhost/orcl")

cur=con.cursor()

uq="update employee set sal=sal+sal\*(50/100) where sal<1.0"

cur.execute(uq)

con.commit()

print("{} Record updated Successfully in Employee Table".format(cur.rowcount ))

except cx\_Oracle.DatabaseError as db:

print("Problem in Database:",db)

#main program

updateRecord()

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

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 for column-n)

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, where 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.

#program for selecting or reading records from employee table---fetchone()

#SelectRecordEx1.py

import cx\_Oracle

def selectrecords():

con=cx\_Oracle.connect("scott/tiger@localhost/orcl")

cur=con.cursor()

cur.execute("select \* from employee")

print("-"\*50)

while(True):

rec=cur.fetchone()

if(rec!=None):

for val in rec:

print("\t{}".format(val),end="")

print()

else:

print("-"\*50)

break

#main program

selectrecords()

#program for selecting or reading records from employee table---fetchmany()

#SelectRecordEx2.py

import cx\_Oracle

def selectrecords():

con=cx\_Oracle.connect("scott/tiger@localhost/orcl")

cur=con.cursor()

cur.execute("select \* from employee order by name ASC") # No. records in employee tab=6

print ("-"\*50)

records=cur.fetchmany()

for record in records:

for val in record:

print("\t{}".format(val),end="")

print()

print("-"\*50)

#main program

selectrecords()

#program for selecting or reading records from employee table---fetchall()

#SelectRecordEx3.py

import cx\_Oracle

def selectrecords():

con=cx\_Oracle.connect("scott/tiger@localhost/orcl")

cur=con.cursor()

cur.execute("select \* from employee order by name ") # No. records in employee tab=6

print("-"\*50)

records=cur.fetchall()

for record in records:

for val in record:

print("\t{}".format(val),end="")

print()

print("-"\*50)

#main program

selectrecords()

#program for displaying col Names of a table

#ColNamesEx1.py

import cx\_Oracle

def colnames():

con=cx\_Oracle.connect("scott/tiger@localhost/orcl")

cur=con.cursor()

cur.execute("select \* from employee")

colnamesinfo=cur.description

print("="\*50)

for colname in colnamesinfo:

print(colname[0],end="\t")

print()

print("="\*50)

#main program

colnames()

#program for displaying col Names of a table

#ColNamesEx2.py

import cx\_Oracle

def colnames():

con=cx\_Oracle.connect("scott/tiger@localhost/orcl")

cur=con.cursor()

cur.execute("select \* from emp")

print("="\*80)

for cname in [colname[0] for colname in cur.description]:

print("\t{}".format(cname),end="")

print()

print("="\*80)

#main program

colnames()

#Write a python program which will accept any table name and print all the records along with column names.

#CompleteTable.py

import cx\_Oracle

def tablerecords():

try:

con=cx\_Oracle.connect("scott/tiger@localhost/orcl")

cur=con.cursor()

#accept the table name from keybord

tname=input("Enter table Name:")

cur.execute("select \* from %s" %tname)

#Dipplay column Names

print("="\*80)

for cname in [colname[0] for colname in cur.description]:

print("\t{}".format(cname),end="")

print()

print("="\*80)

#display records

records=cur.fetchall()

for record in records:

for val in record:

print("\t{}".format(val),end="")

print()

print("="\*80)

except cx\_Oracle.DatabaseError as db:

print("prob in DB:",db)

#main program

tablerecords()

#Write a python program which will obtain connection from MySQL databases

#MySQLConnTest.py

import mysql.connector

con=mysql.connector.connect(host="localhost",

user="root",

passwd="root")

print("Type of con=",type(con))

print("Python Program Obtains Connection from MySQL")

#Write a python program which will obtain connection and cusrsor objects in MySQL databases

#MySQLCursorTest.py

import mysql.connector

try:

con=mysql.connector.connect(host="localhost",

user="root",

passwd="root")

print ("\nType of con= “, type(con))

print ("Python Program Obtains Connection from MySQL")

cur=con.cursor()

print ("\nType of cus= “, type(cur))

print ("Python Program Created Cursor object:")

except mysql.connector.DatabaseError as db:

print ("Prob inb MySQL DB:”, db)

#Write a Python program which will create a database in the name of batch 11am in my SQL database.

#mysqldbcreate.py

import mysql.connector

try:

con=mysql.connector.connect(host="localhost",

user="root",

passwd="root")

print("Python Program obtains connection MYSQL")

cur=con.cursor()

#create a database name --->batch11am

dq="create database HydPython"

cur.execute(dq)

print("Database created Successfully")

except mysql.connector.DatabaseError as db:

print("Prob inb MySQL DB:",db)

#Write a python program which will create employee table in batch11am database of MySQL

#TableCreateEx.py

import mysql.connector

def createtable():

try:

con=mysql.connector.connect(host="localhost",

user="root",

passwd="root",

database="batch11am")

cur=con.cursor()

#prepare the query and execute

tq="create table stuident(sno int primary key, name varchar(10) not null, marks float not null)"

cur.execute(tq)

print("Table created successfully")

except mysql.connector.DatabaseError as db:

print("Prob inb MySQL DB:",db)

#main program

createtable()

#Write a python program which will insert employee records in MYSQL by reading the values from keyboard

#RecordInsertEx.py

import mysql.connector,sys

def insertRecord():

while(True):

try:

con=mysql.connector.connect(host="localhost",

user="root",

passwd="root",

database="batch11am")

cur=con.cursor()

print("-"\*50)

#accept employee values from KBD

eno=int(input("Enter Employee Number:"))

ename=input("Enter Employee Name:")

empsal=float(input("Enter Employee Salary:"))

#Prepare the query and execute

cur.execute("insert into employee values(%d,'%s',%f) " %(eno,ename,empsal) )

con.commit()

print("-"\*50)

print("{} Record Inserted Successfully in Employee Table".format(cur.rowcount ))

print("-"\*50)

ch=input("Do u want to insert another record(yes/no):")

if(ch.lower()=="no"):

print("Thx for using this program")

sys.exit()

print("-"\*50)

except mysql.connector.DatabaseError as db:

print("Problem in Database:",db)

except ValueError:

print("Don't eneter alnums,strs and symbols for empno, salary")

#main program

insertRecord()

#WAPP which will Delete a record based on employee number from employee table of mysql

#RecordDeleteEx.py

import mysql.connector,sys

def deleteRecord():

while(True):

try:

con=mysql.connector.connect(host="localhost",

user="root",

passwd="root",

database="batch11am")

cur=con.cursor()

print("-"\*50)

#accept employee values from KBD

empno=int(input("Enter Employee Number for deleting the Record:"))

#Prepare the query and execute

cur.execute("delete from employee where eno=%d" %empno)

con.commit()

if(cur.rowcount>0):

print("{} Record deleted Succeessfully".format(cur.rowcount))

else:

print("Employee Record does not exist:")

print("-"\*50)

ch=input("Do u want to delete another record(yes/no):")

if(ch.lower()=="no"):

print("Thx for using this program")

sys.exit()

print("-"\*50)

except mysql.connector.DatabaseError as db:

print("Problem in Database:",db)

except ValueError:

print("Don't eneter alnums,strs and symbols for empno, salary")

#main program

deleteRecord()

#write a python program which will update name of the employee and give 50% hike for that employee based on employee number

#RecordUpdateEx.py

import mysql.connector

def updateRecord():

try:

con=mysql.connector.connect(host="localhost",

user="root",

passwd="root",

database="batch11am")

cur=con.cursor()

eno=int(input("Enter Employee Number for updating Name and Sal:"))

empname=input("Enter Ur Correct Name:")

uq="update employee set name='%s' , sal=sal+sal\*(50/100) where eno=%d"

cur.execute(uq %(empname,eno))

con.commit()

if(cur.rowcount>0):

print("{} Record updated Successfully in Employee Table".format(cur.rowcount ))

else:

print("Employee Record does not exist:")

except mysql.connector.DatabaseError as db:

print("Problem in Database:",db)

#main program

updateRecord()

#Write a python program which will accept any table name and print all the records along with column names.

#CompleteTable.py

import mysql.connector

def tablerecords():

try:

con=mysql.connector.connect(host="localhost",

user="root",

passwd="root",

database="batch11am")

cur=con.cursor()

#accept the table name from keybord

tname=input("Enter table Name:")

cur.execute("select \* from %s" %tname)

#Dipplay column Names

print("="\*80)

for cname in [colname[0] for colname in cur.description]:

print("\t{}".format(cname),end="")

print()

print("="\*80)

#display records

records=cur.fetchall()

for record in records:

for val in record:

print("\t{}".format(val),end="")

print()

print("="\*80)

except mysql.connector.DatabaseError as db:

print("prob in DB:",db)

#main program

tablerecords()