# Write a server side program which will accept employee number from client, retrieve empname, salary and designation from emp table.

#ServerEmpData.py-----------Program-(A)

import socket

import cx\_Oracle

s=socket.socket()

s.bind(("127.0.0.1",3600))

s.listen(2)

print("SSP is Ready to accept CSP request:")

while(True):

try:

cs,ca=s.accept()

eno=int(cs.recv(1024).decode())

#PDBC

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

print("SSP connectd to Oracle DB")

cur=oracon.cursor()

cur.execute("select name,sal,cname from employee where eno=%d" %eno)

record=cur.fetchone()

if(record==None):

cs.send("Employee Record Does not Exist".encode())

else:

cs.send(str(record).encode())

except ValueError:

cs.send("Don't enter strs,Symbols and alph-numerics for empno".encode() )

except cx\_Oracle.DatabaseError as db:

cs.send("Prob in DB"+str(db).encode())

except :

cs.send("OOOOPs Some went wrong".encode())

#program for getting current working folder ---getcwd()

#getcwdex.py

import os

cwdname=os.getcwd()

print(type(cwdname))

print("current working folder=",cwdname)

#Program for listing the files in a folder

#listdirex.py

import os

try:

files=os.listdir("E:\KVR-PYTHON-6PM\FILES")

print ("Number of files =",len(files))

for file in files:

print(file)

except FileNotFoundError:

print ("File does not exists")

#Program for Creating Folders Hierarchy

#makedirsex.py

import os

try:

os.makedirs("C:\INDIA\BANG\AMPT\INDIA")

print("Folders Hirerarchy created..")

except FileExistsError:

print("Folders Hirerarchy already exist--try with some other")

#Program for creating a folder

#mkdirex.py

import os

try:

os.mkdir("C:\HYD\AMPT")

print("Folder created successfully-verify")

except FileExistsError:

print("Folder already exist--try with another one")

except FileNotFoundError:

print("with mkdir(), we can create Folders Hierarchy")

#program for removing a folders hierarchy

#removedirsex.py

import os

try:

os.removedirs("C:\INDIA\TS\AMPT\python")

print("Folders Hiearchy Removed")

except FileNotFoundError:

print("File does not exists")

except OSError:

print("Folders hierarhcy is not empty-can't remove")

#Program for renaming a folder

#renamefolderex.py

import os

try:

os.rename("E:\KVR-PYTHON-6PM\FILESPROG","E:\KVR-PYTHON-6PM\FILES")

print("Folder renamed")

except FileNotFoundError:

print("Folder does not exists")

#program for removing a folder

#rmdirex.py

import os

try:

os.rmdir("C:\PYTHON")

print("Folder Removed")

except FileNotFoundError:

print("File does not exists")

except OSError:

print("Folder is not empty-can't remove")

#RemoveFileEx.py

import os

try:

os.remove("D:\KVR\human.png")

print("File Name removed Sucessfully")

except FileNotFoundError:

print("File does not exist")

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

Iterators in Python

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

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

Why should WE use Iterators:

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

=>In modern days, we have a lot of data in our hands, and handling this huge amount of data creates problems for everyone who wants to do some sort of analysis with that data.So, If you’ve ever struggled with handling huge amounts of data, and your machine running out of memory, then WE use the concept of Iterators in Python.

=>Therefore, rather than putting all the data in the memory in one step, it would be better if we could work with it in bits or some small chunks, dealing with only that data that is required at that moment. As a result, this would reduce the load on our computer memory tremendously. And this is what exactly the iterators do.

=>Therefore, you can use Iterators to save a ton of memory, as Iterators don’t compute their items when they are generated, but only when they are called upon.

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

=>Iterator in python is an object that is used to iterate over iterable objects like lists, tuples, dicts, and sets.

=>The iterator object is initialized using the iter() method. It uses the next() method for iteration.

=>Here iter() is used for converting Iterable object into Iterator object.

=>next() is used for obtaining next element of iterator object and if no next element then we get an exception called StopIteration.

=>On the object of Iterator, we can't perform Indexing and Slicing Operations bcoz They supply the values on demand .

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

Examples:

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

s = 'Python'

itobj = iter(s)

while True:

try:

item = next(s) # Iterate by calling next

print(item)

except StopIteration: # exception will happen when iteration will over

break

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

mytuple = ("apple", "banana", "cherry")

myit = iter(mytuple)

print(next(myit))

print(next(myit))

print(next(myit))

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

#IteratorEx1.py

lst=["Python","Java","C","C++","DSc"]

print(lst,type(lst))

print ("--------------------------------------------")

lstitr=iter(lst)

print ("Content of Iterators:")

print(next(lstitr))

print(next(lstitr))

print(next(lstitr))

print(next(lstitr))

print(next(lstitr))

print(next(lstitr))

#IteratorEx2.py

tpl=("Python","Java","C","C++","DSc")

print(tpl,type(tpl))

print ("--------------------------------------------")

tplitr=iter(tpl)

print("Content of Iterators:",type(tplitr))

while(True):

try:

print(next(tplitr))

except StopIteration:

break

#IteratorEx3.py

tpl={"Python","Java","C","C++","DSc"}

prin t(tpl,type(tpl))

print("--------------------------------------------")

tplitr=iter(tpl)

print("Content of Iterators:",type(tplitr))

while(True):

try:

print(next(tplitr))

except StopIteration:

break

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

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 a file which contains a JSON object.

Consider a file named employee.json 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)

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

# Python program to convert JSON to Python

#JSONLOADS.py

import json

# JSON string

employee = ' {"id":"09", "name": "Rossum", "department":"IT"} '

# Convert JSON string to Python dict

print("Type employee=",type(employee))

ed = json.loads(employee)

print(ed,type(ed))

for k,v in ed.items():

print("{}-->{}".format(k,v))

# Python program to read json file

#JsonRead.py

import json

# Opening JSON file

try:

with open("sample.json","r") as fp:

# returns JSON object as a dictionary

dictdata = json.load(fp)

# Iterating through the json list

for k,v in dictdata.items():

print("{}--->{}".format(k,v))

except FileNotFoundError:

print("Json File does not exist")

#JsontoDict.py

import json

# JSON string

employee = ' {"id":"09", "name": "Rossum", "department":"IT"} '

print(type(employee))

"""print("Json String data=",employee)

# Convert string to Python dict

employee\_dict = json.loads(employee)

print("Dict Data=",employee\_dict)

for k,v in employee\_dict.items():

print("{}-->{}".format(k,v))"""

# Python program to write JSON to a file

#JsonWrite.py

import json

# Data to be written

dictionary ={"rollno" : 56,"name" : "Rossum", "cgpa" : 8.6}

with open("sample.json", "w") as fp:

json.dump(dictionary, fp)

print("Data written to file--verify")