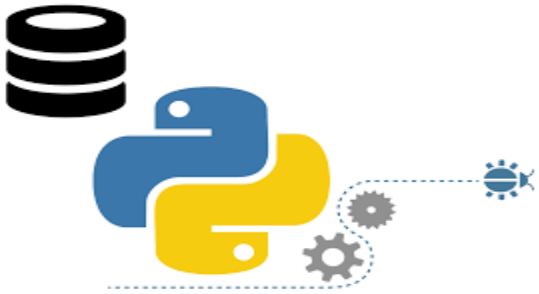
CX_oracle module for database developers working with python application:



CX_oracle is a database API developed to communicate with database objects and perform database operations within python scripting. The API is developed for both python 2.7 and 3.4+ versions.

In order to use this library we can execute the below command if python is installed on our system. We can execute the below command in the cmd prompt and this will install the library from pypi using pip which is a package manager of python.

```
python -m pip install cx_Oracle –upgrade
```

Once the library is installed we can use the library directly by importing it inside our python script using import command. Below is the syntax to import the library which is installed on our server (installed independently or using any python distribution ex: anaconda distribution).

```
Import cx_Oracle
```

Once we are all set with the installation, now it's time to use the library to interact with database.

Mostly we work on the existing database objects like tables , views , procedures etc. so there is no need to write create commands using cx_oracle in python code but we have the option to create objects in case of any requirement.

Following operations are mostly performed on database using CX_oracle module:

1) INSERT
2) UPDATE
3) DELETE
4) SELECT
5) CALL PROCEDURES
6) CALL FUNCTIONS

As part of this tutorial, all the operations will be performed on the below table created for practice.

```
CREATE TABLE client_details
(
    name        VARCHAR2(100),
    location    VARCHAR2(50),
    turnover    NUMBER
);
```

 # The below code is used to insert data in database using python script and DB API

```
import cx_Oracle
#creating connection object to database using the below statement in which we pass connection strings
con= cx_Oracle.connect("sbxtreme", "sbxtreme","localhost/xe")
#creating cursor using connection object which will execute the sql query
cur=con.cursor()
# creating SQL query to insert data in table
sql_query="insert into sbxtreme.client_details(name,location,turnover) values (:1,:2,:3)"
# execute the sql query created in above step and assigned values to bind variables
cur.execute(sql_query,{"1":'Barclay',"2":'USA',"3":782999779})
# commit on database level
con.commit()
# closing cursor
cur.close()
# closing connection
con.close()
```

 # The below code is used to perform multiple inserts in database using python script and DB API

```
import cx_Oracle
#creating connection object to database using the below statement in which we pass connection strings
con= cx_Oracle.connect("sbxtreme", "sbxtreme","localhost/xe")
#data to insert
name=['CA','IBM','GGL','ADOBE']
location=['UK','US','JPN','UK']
turnover=[1332313,4253524,6313245,5356674]
#creating cursor using connection object which will execute the sql query
cur=con.cursor()
# creating sequence of key value pair in dictionary which is stored in list
dict_seq=[{'1':name[i],'2':location[i],'3':turnover[i]} for i in range (1,len(name))]
# sql query to execute on db
sql_query="insert into sbxtreme.client_details(name,location,turnover) values (:1,:2,:3)"
# preparing sql_query for execution
cur.prepare(sql_query)
# using execute many to replace the bind variable at run time
cur.executemany(None,dict_seq)
# committing the transaction on database level, once commit is done automatically connection will be closed in case of multiple inserts
con.commit()
```

 # The below code is used to update/delete data in database using python script and DB API

 # Snippet for update

```
import cx_Oracle
#creating connection object to database using the below statement in which we pass connection strings
con= cx_Oracle.connect("sbxtreme", "sbxtreme","localhost/xe")
#creating cursor using connection object which will execute the sql query
cur=con.cursor()
# creating sql query
sql_query = 'update sbxtreme.client_details  set turnover = :1 where name = :2'
cur.execute(sql_query,{'1':93766274,'2':'IBM'})
con.commit()
```

 # Snippet for delete operation

```
sql_query = 'delete from sbxtreme.client_details where name = :1'
cur.execute(sql_query,{'1':'IBM'})
con.commit()
```

Calling procedures and functions:

Using Cx_oracle library we can also call procedures and functions in our python scripts. Usually it's always better to implement the business logic on database level in procedures and functions rather than on scripting level but again it depends on the requirement and application architecture.

While calling procedures we write the database interaction code in the form of python classes and methods to make code better to read and modular. Also in case of any out parameters used in the procedure, it can be handled easily using the class and method approach which performs individual tasks of connecting database, calling procedures and closing connections. The in parameters can easily be passed in the procedure and we can capture the outparam returned from procedure using class /method approach.

While calling functions the same approach can be followed which helps in capturing the output of functions and pass with in python script to be used for some operation in the script.

Below is the standard approach mentioned in Oracle documentation for calling stored procedure and functions using cx_Oracle DB API in python script.

http://www.oracle.com/technetwork/articles/prez-stored-proc-084100.html