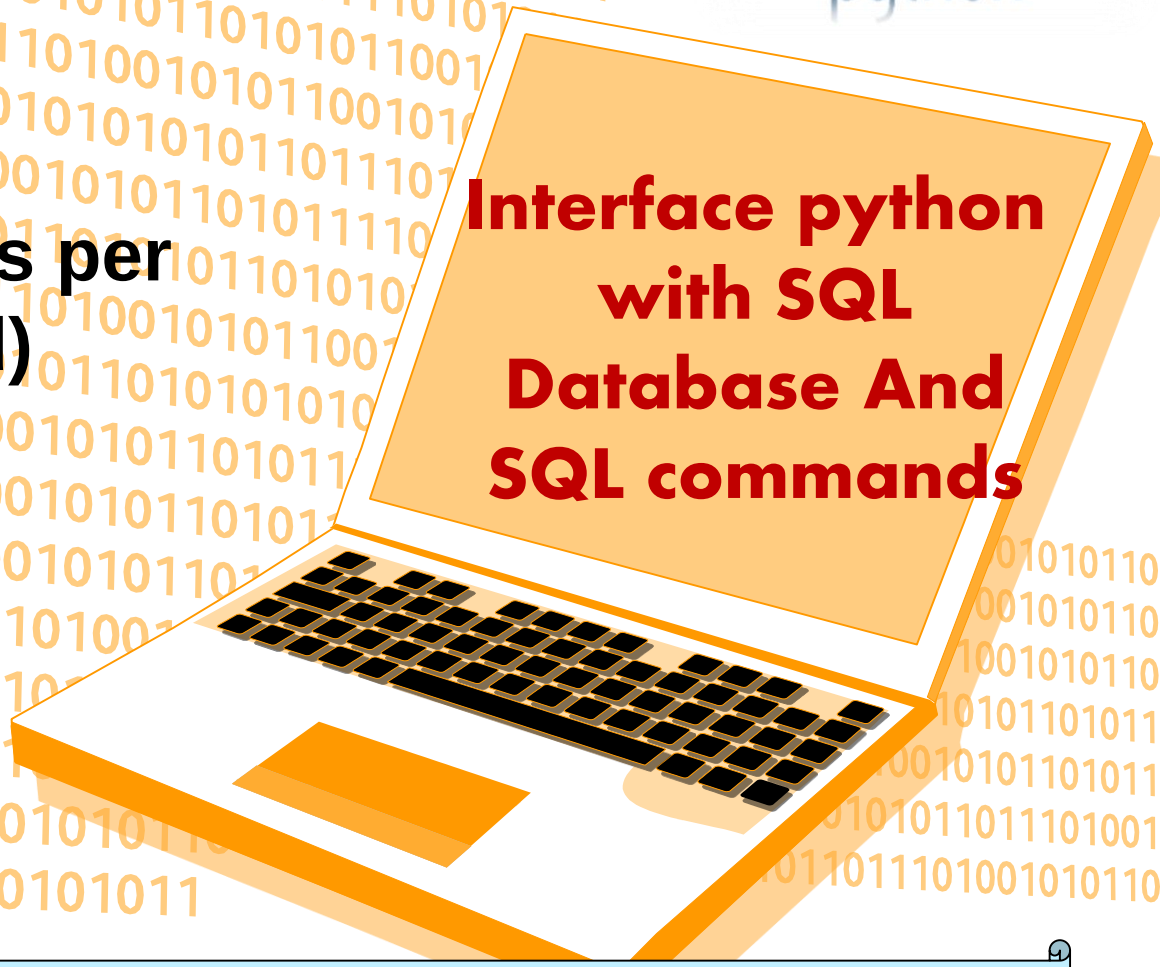
Chapter 13 :

Computer Science

Class XII (As per CBSE Board)

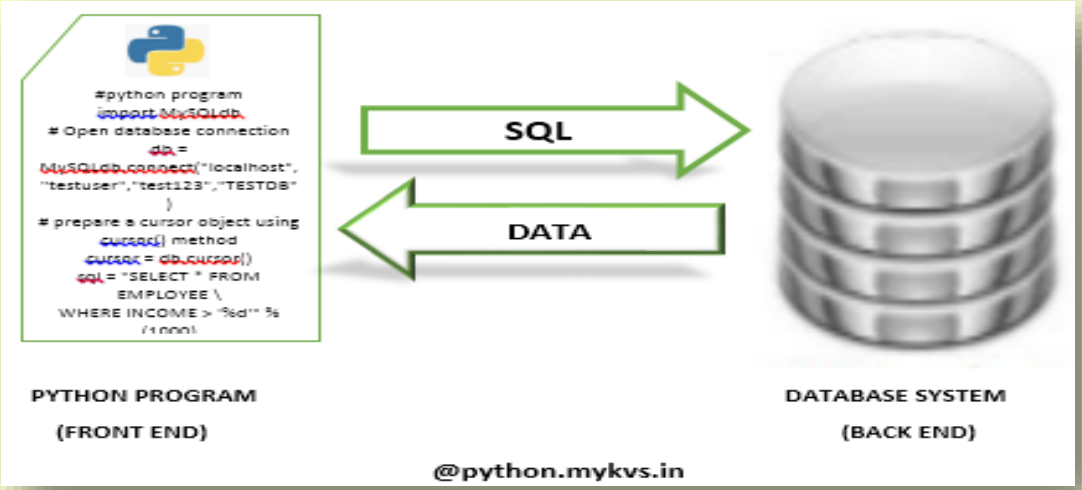


New Syllabus 2019-20

Visit : python.mykvs.in for regular updates

Interface python with SQL Database

A database is nothing but an organized collection of data. Data is organized into rows, columns and tables and it is indexed to make it easier to find relevant information. All companies whether large or small use databases. So it become necessary to develop project/software using any programming language like python in such a manner which can interface with such databases which support SQL.Generalised form of Interface of python with SQL Database can be understood with the help of this diagram.



Form/any user interface designed in any programming language is **Front End** where as data given by database as response is known as **Back-End** database.

SQL is just a query language, it is not a database. To perform SQL queries, we need to install any database for example Oracle, MySQL, MongoDB, PostGres SQL, SQL Server, DB2 etc.

Using SQL in any of the dbms ,databases and table can be created and data can be accessed, updated and maintained. The Python standard for database interfaces is the **Python DB-API**. Python Database API supports a wide range of database servers, like msql , mysql, postgressql, Informix, oracle, Sybase etc.

Interface python with SQL Database

Why choose Python for database programming

Following are the reason to choose python for database programming

- **Programming more efficient and faster compared to other languages.**
- **Portability of python programs.**
- **Support platform independent program development.**
- **Python supports SQL cursors.**
- **Python itself take care of open and close of connections.**
- **Python supports relational database systems.**
- **Porting of data from one dbms to other is easily possible as it support large range of APIs for various databases.**

Interface python with SQL Database

SQL Connectors

We must download a separate DB API module for each database we need to access. Suppose we need to access an Oracle database as well as a MySQL database, we must download both the Oracle and the MySQL database modules .

The **DB API** provides a minimal standard for working with databases using Python structures and syntax wherever possible.

This API includes the following –

- Importing the API module.
- Acquiring a connection with the database.
- Issuing SQL statements and stored procedures.
- Closing the connection

Interface python with SQL Database

Here we are using mysql as back end database because of it is open source,free and portable and widely used. Any one of mysql-connector or MySQLdb can be used for database programming.

1. mysql-connector

MySQL-Connector enables Python programs to access MySQL databases, using an API that is compliant with the Python Database API Specification v2.0 (PEP 249). It is written in pure Python and does not have any dependencies except for the Python Standard Library.

Steps to use mysql-connector

1. Download Mysql API ,exe file and install it.(click here to download)
2. Install Mysql-Python Connector (Open command prompt and execute command) >**pip install mysql-connector**
3. Now connect Mysql server using python
4. Write python statement in python shell **import mysql.connector**

   If no error message is shown means mysql connector is properly installed

Interface python with SQL Database

2. MySQLdb

MySQLdb is an interface for connecting to a MySQL database server from Python. It implements the Python Database API v2.0 and is built on top of the MySQL C API.

Steps to use mysqlclient

1. First Upgrade pip command through > python –m pip install –upgrade pip
2. Install mysqlclient through **pip install mysqlclient**
3. After successful installation check through **import mysqldb**
4. If it is installed no error will be displayed otherwise error message will be displayed

To install MySQLdb module, use the following command −

For Ubuntu, use the following command -

$ sudo apt-get install python-pip python-dev libmysqlclient-dev

For Fedora, use the following command -

$ sudo dnf install python python-devel mysql-devel redhat-rpm-config gc c

For Python command prompt, use the following command -

pip install MySQL-python

Note − Make sure you have root privilege to install above module

Interface python with SQL Database

Establish connection

For database interface/database programming ,connection must be established.Before establishing connection there must be mysql installed on the system and a database and table is already created.In following way we can establish a connection with mysql database through mysql.connector.

```
import mysql.connector
mydb=mysql.connector.connect(host="localhost",user="root",passwd="root“,database
=“school”)
print(mydb)
```

Alternatively we can write the following statement if we are using mysqldb

```
import MySQLdb
mydb = MySQLdb.connect("localhost",“root",“root",“school" )
print(mydb)
```

In both way we are specifying host,user,password and database name as arguments.database is optional argument if we want to create database through programming later on.

After successful execution of above statements in python following out will be displayed

```
<mysql.connector.connection.MySQLConnection object at 0x022624F0>
```

Otherwise an error message will be shown.

Interface python with SQL Database

Cursor object :

The MySQLCursor class instantiates objects that can execute operations such as SQL statements. Cursor objects interact with the MySQL server using a MySQLConnection object.

How to create cursor object and use it

```
import mysql.connector
mydb=mysql.connector.connect(host="localhost",user="root",passwd="root")
mycursor=mydb.cursor()
mycursor.execute("create database if not exists school")
mycursor.execute("show databases")
for x in mycursor:
    print(x)
```

Through line 4 we are creating a database named school if it is already not created with the help of cursor object.

Line 5 executes the sql query show databases and store result in mycursor as collection ,whose values are being fetched in x variable one by one.

On execution of above program school database is created and a list of available databases is shown.

Interface python with SQL Database

How to create table at run time

Table creation is very easy ,if we are already well versed in sql table creation then we have to just pass the create table query in execute() method of cursor object. But before table creation we must open the database.Here we are opening database school(through connect() method) before student table creation.

```
import mysql.connector
mydb=mysql.connector.connect(host="localhost",user="root",passwd="root"
,database="school")
mycursor=mydb.cursor()
mycursor.execute("create table student(rollno int(3) primary key,name
varchar(20),age int(2))")
```

On successful execution of above program a table named student with three fields rollno,name,age will be created in school database.

We can check student table in mysql shell also,if required.

Interface python with SQL Database

How to change table structure/(add,edit,remove colum of a table) at run time

To modify the structure of the table we just have to use alter table query.Below program will add a column mark in the student table.

```
import mysql.connector
mydb=mysql.connector.connect(host="localhost",user="root",passwd="root"
,database="school")
mycursor=mydb.cursor()
mycursor.execute("alter table student add (marks int(3))")
mycursor.execute("desc student")
for x in mycursor:
    print(x)
```

Above program will add a column marks in the table student and will display the structure of the table

Interface python with SQL Database

How to search records of a table at run time

Below statement demonstrate the use of select query for searching specific record from a table.

```
import mysql.connector
mydb=mysql.connector.connect(host="localhost",user="root",passwd="root"
,database="school")
mycursor=mydb.cursor()
nm=input("enter name")
mycursor.execute("select * from student where name='"+nm+"'")
for x in mycursor:
    print (x)
```

Above statements will prompt a name from user,as user type the name ,that name is searched into the table student with the help of select query .result will be shown with the help of mycursor collection.

Interface python with SQL Database

How to fetch all records of a table at run time

```
import mysql.connector
mydb=mysql.connector.connect(host="localhost",user="root",passwd="root"
,database="school")
mycursor=mydb.cursor()
mycursor.execute("select * from student")
myrecords=mycursor.fetchall()
for x in myrecords:
    print (x)
```

MySQLCursor.fetchall() Method

The method fetches all (or all remaining) rows of a query result set and returns a list of tuples. If no more rows are available, it returns an empty list.

Interface python with SQL Database

How to fetch one record of a table at run time

```
import mysql.connector
mydb=mysql.connector.connect(host="localhost",user="root",passwd="root"
,database="school")
mycursor=mydb.cursor()
mycursor.execute("select * from student")
row=mycursor.fetchone()
while row is not None:
  print(row)
  row = mycursor.fetchone()
```

MySQLCursor.fetchone() Method

This method retrieves the next row of a query result set and returns a single sequence, or None if no more rows are available. By default, the returned tuple consists of data returned by the MySQL server, converted to Python objects.

MySQLCursor.fetchmany() Method

```
rows = cursor.fetchmany(size=1)
```

This method fetches the next set of rows of a query result and returns a list of tuples. If no more rows are available, it returns an empty list.

Interface python with SQL Database

How to delete record of a table at run time

```
import mysql.connector
mydb=mysql.connector.connect(host="localhost",user="root",passwd="root"
,database="school")
mycursor=mydb.cursor()
mycursor.execute("delete from student where rollno=1")
mydb.commit()
```

In above program delete query will delete a record with rollno=1.commit() method is necessary to call for database transaction.

How to update record of a table at run time

```
import mysql.connector
mydb=mysql.connector.connect(host="localhost",user="root",passwd="root"
,database="school")
mycursor=mydb.cursor()
mycursor.execute("update student set marks=99 where rollno=2")
mydb.commit()
```

In above program update query update the marks with 99 of rollno=2

Students are advised to develop menu driven program using above concepts for better understating of python mysql database interface.

Interface python with SQL Database

Manage Database Transaction

Database transaction represents a single unit of work. Any operation which modifies the state of the MySQL database is a transaction.

Python MySQL Connector provides the following method to manage database transactions.

commit – MySQLConnection.commit() method sends a COMMIT statement to the MySQL server, committing the current transaction.
rollback – MySQLConnection.rollback revert the changes made by the current transaction.
AutoCommit – MySQLConnection.autocommit value can be assigned as True or False to enable or disable the auto-commit feature of MySQL. By default its value is False.

Interface python with SQL Database

Manage Database Transaction

```
try:
   conn = mysql.connector.connect(host='localhost',
                          database='school',
                          user='root',
                          password='root')
   conn.autocommit = false
   cursor = conn.cursor()
   sql_update_query = """Update student set marks = 95 where rollno = 2"""
   cursor.execute(sql_update_query)
   print ("Record Updated successfully ")
   #Commit your changes
   conn.commit()
except mysql.connector.Error as error :
    print("Failed to update record to database rollback: {}".format(error))
    #reverting changes because of exception
    conn.rollback()
finally:
    #closing database connection.
    if(conn.is_connected()):
        cursor.close()
        conn.close()
        print("connection is closed")
```

In above program if update query is successfully executed then commit() method will be executed otherwise exception error part will be executed where revert of update query will be done due to error.At finally we are closing cursor as well as connection.To rollback or commit we have to set autocommit=false,just like **conn.autocommit = false** in above program otherwise rollback will not work

SQL Commands

Grouping Records in a Query

- Some time it is required to apply a Select query in a group of records instead of whole table.
- We can group records by using GROUP BY <column> clause with Select command. A group column is chosen which have non-distinct (repeating) values like City, Job etc.
- Generally, the following Aggregate Functions [MIN(), MAX(), SUM(), AVG(), COUNT()] etc. are applied on groups.

Name	Purpose
SUM()	Returns the sum of given column.
MIN()	Returns the minimum value in the given column.
MAX()	Returns the maximum value in the given column.
AVG()	Returns the Average value of the given column.
COUNT()	Returns the total number of values/ records as per given column.

SQL Commands

Aggregate Functions & NULL

Consider a table Emp having following records as-

Null values are excluded while (avg)aggregate function is used

Emp		
Code	Name	Sal
E1	Mohak	NULL
E2	Anuj	4500
E3	Vijay	NULL
E4	Vishal	3500
E5	Anil	4000

SQL Queries

Result of query

```
mysql> Select Sum(Sal) from EMP;          12000
mysql> Select Min(Sal) from EMP;          3500
mysql> Select Max(Sal) from EMP;          4500
mysql>  Select Count(Sal) from EMP;       3
mysql>  Select Avg(Sal) from EMP;         4000
mysql>  Select Count(*) from EMP;         5
```

SQL Commands

Aggregate Functions & Group

An Aggregate function may applied on a column with DISTINCT or ALL keyword. If nothing is given ALL is assumed.

Using SUM (<Column>)

This function returns the sum of values in given column or expression.

```
mysql> Select Sum(Sal) from EMP;
mysql> Select Sum(DISTINCT Sal) from EMP;
mysql> Select Sum (Sal) from EMP where City='Jaipur';
mysql> Select Sum (Sal) from EMP Group By City;
mysql> Select Job, Sum(Sal) from EMP Group By Job;
```

Using MIN (<column>)

This functions returns the Minimum value in the given column.

```
mysql> Select Min(Sal) from EMP;
mysql> Select Min(Sal) from EMP Group By City;
mysql> Select Job, Min(Sal) from EMP Group By Job;
```

SQL Commands

Aggregate Functions & Group

Using MAX (<Column>)

This function returns the Maximum value in given column.

```
mysql> Select Max(Sal) from EMP;
mysql> Select Max(Sal) from EMP where City='Jaipur';
mysql> Select Max(Sal) from EMP Group By City;
```

Using AVG (<column>)

This functions returns the Average value in the given column.

```
mysql> Select AVG(Sal) from EMP;
mysql> Select AVG(Sal) from EMP Group By City;
```

Using COUNT (<*|column>)

This functions returns the number of rows in the given column.

```
mysql> Select Count (*) from EMP;
mysql> Select Count(Sal) from EMP Group By City;
mysql> Select Count(*), Sum(Sal) from EMP Group By Job;
```

SQL Commands

Aggregate Functions & Conditions

You may use any condition on group, if required. HAVING <condition> clause is used to apply a condition on a group.

```
mysql> Select Job,Sum(Pay) from EMP
Group By Job HAVING Sum(Pay)>=8000;

mysql> Select Job, Sum(Pay) from EMP
Group By Job HAVING Avg(Pay)>=7000;

mysql> Select Job, Sum(Pay) from EMP
Group By Job HAVING Count(*)>=5;

mysql> Select Job, Min(Pay),Max(Pay), Avg(Pay) from EMP Group
       By Job HAVING Sum(Pay)>=8000;

mysql> Select Job, Sum(Pay) from EMP Where City='Jaipur'
```

Note :- Where clause works in respect of whole table but Having works on Group only. If Where and Having both are used then Where will be executed first.

SQL Commands

Ordering Query Result – ORDER BY Clause

A query result can be orders in ascending (A-Z) or descending (Z-A)

order as per any column. Default is Ascending order.

mysql> SELECT * FROM Student ORDER BY City;

To get descending order use DESC key word.

mysql> SELECT * FROM Student ORDER BY City DESC;

mysql> SELECT Name, Fname, City FROM Student Where Name LIKE ‘R%’ ORDER BY Class;