# DATABASE CONNECTIVITY

## TOPICS

1. TYPES OF STORAGE UNITS
    a. TEMPORARY STORAGE UNIT
        i. List
        ii. Variables
        iii. Arrays etc
    b. PERMANENT STORAGE UNIT
        i. Files
        ii. Database
        iii. Bigdata
        iv. Cloud computing
2. DRAWBACK OF FILES
    a. We can’t store the huge amount of data
    b. data will be stored in unstructured format
    c. retrieving of information is difficult
    d. we can’t provide the security in the file system
    e. we can’t backup and restore
3. DATABASE INTRODUCTION
    a. Database is used to store the huge amount of data
    b. Data will be stored in structured format i.e., tables
    c. Retrieving of information is easy and fast
    d. We can provide the security to database
    e. We can backup and restore

4. STEPS TO CONNECT PYTHON TO THE DATABASE
    a. IMPORT SQLITE3 MODULE
        i. It is the built in module in python
    b. CREATE THE CONNECTION OBJECT
        i. Create the connection object by using connect() method which is available under the sqlite3 module
        ii. connect() method is used to establish the connection between the python software and the database
        iii. syntax:

```
conobj=sqlite3.connect(database_name)
```

        ⇨ it returns the connection object
    c. CREATE THE CURSOR OBJECT
        i. Create the cursor object by using cursor() method of the database connection object
        ii. Syntax:

```
curobj=conobj.cursor()
```

        ⇨ Cursor object is used to execute the queries on the database
    d. EXECUTE THE QUERIES
        - After creating the cursor object, we can execute the queries by using cursor object
        - Syntax:
            - curobj.execute(query) – it is used to execute only single query
            - curobj.execute(queries) – it is used to execute a string of sql queries separated by semicolon
            - curobj.executemany() – it is used to execute parameterized queries

e. COMMIT THE CHANGES
   - Save the changes to the database by using commit() method of connection object after performing any operaion on the database
   - Syntax:

```
conobj.commit()
```

f. CLOSE THE DATABASE CONNECTION
   a. We must close the database connection after saving the changes to the database
   b. We can close the database connection by using close() method of the connection object
   c. Once we connect to the database we must disconnect from the database
   d. Syntax:

```
conobj.close()
```

CREATING THE TABLE

⇨ When you are giving the database name
   - If database exists – it connects to it
   - If database doesn't exist – it will create new database

ex:

```
import sqlite3
conobj=sqlite3.connect("sampledatabase.db")
curobj=conobj.cursor()
curobj.execute("create table student(name varchar(20)
not null,id int primary key)")
conobj.commit()
conobj.close()
```

INSERTING THE DATA INTO THE TABLE

Ex:

```
import sqlite3
conobj=sqlite3.connect("sampledatabase.db")
curobj=conobj.cursor()
curobj.execute("insert into student values('raju',20)")
conobj.commit()
conobj.close()
```

5. EXECUTING MULTIPLE QUERIES

1. By using executescript(queries)

Syntax:

```
curobj.executescript(queries)
```

ex:

```
import sqlite3
conobj=sqlite3.connect("sampledatabase5.db")
curobj=conobj.cursor()
curobj.executescript("create table anime(SNO int
primary key,name varchar(20) not null,release_date
int );insert into anime
values(1,'demonslayer',2006);insert into anime
values(2,'naruto',2000)")
conobj.commit()
conobj.close()
```

2. By using executemany()

Syntax:

```
curobj.executemany(sql_query,values)
```

ex:

```
import sqlite3
conobj=sqlite3.connect("sampledatabase5.db")
curobj=conobj.cursor()
sql_query="insert into anime(SNO,name,release_date)
values(?,?,?)"
values=[(4,"attackontitan",2007),(5,"jujustukaisen",20
08)]
curobj.executemany(sql_query,values)
conobj.commit()
conobj.close()
```

6. METHODS OF FETCHING
    1. fetchall():
        a. this method is used to fetch the all the records/rows from the table
        b. it returns the list of strings, in this each string is considered as an one row in the table
        c. syntax:

```
curobj.fetchall()
```

ex:

```
import sqlite3
conobj=sqlite3.connect("sampledatabase5.db")
curobj=conobj.cursor()
curobj.execute("select * from anime")
records=curobj.fetchall()
for i in records:
    print(i)
conobj.commit()
conobj.close()
```

2. fetchone()
   a. this method is used to fetch only single record/row from the table
   b. syntax:

```
curobj.fetchone()
```

ex:

```
import sqlite3
conobj=sqlite3.connect("sampledatabase5.db")
curobj=conobj.cursor()
curobj.execute("select * from anime")
record=curobj.fetchone()
print(record)
conobj.commit()
conobj.close()
```

3. fetchmany(n)

a. this method is used to fetch n no.of records/rows from starting record of the table

b. n – it represents how many records to fetch from starting record

c. syntax:

```
curobj.fetchmany(n)
```

ex:

```
import sqlite3
conobj=sqlite3.connect("sampledatabase5.db")
curobj=conobj.cursor()
curobj.execute("select * from anime")
records=curobj.fetchmany(4)
for i in records:
        print(i)
conobj.commit()
conobj.close()
```