**SQLite command line**

The following methods are defined in the connection class :

Cursor (): - Returns a Cursor object which uses this Connection.

Commit (): - Explicitly commits any pending transactions to the database

Rollback (): - Causes a transaction to be rolled back to the starting point. This method is optional since not all databases provide transaction support.

Close (): - Closes the connection to the database permanently. Attempts to use the connection after calling this will raise a DB-API Error.

Further reading: <https://www.python.org/dev/peps/pep-0249/#connection-methods>

**Creating a database from Python - Practice**

import sqlite3

MySchool=sqlite3.connect('schooltest.db')

curschool=MySchool.cursor()

curschool.execute('''CREATE TABLE student (

StudentID INTEGER PRIMARY KEY AUTOINCREMENT,

name TEXT (20) NOT NULL,

age INTEGER,

marks REAL);''')

MySchool.close()

|  |
| --- |
| 1. ****Inserting a new Record**** |

In the previous section, you learned how to create a database and a table within it. Now, let's see how to create a new record in the existing table, student.

**Example 1:** To add the following record:

    Name = Sherlock

    House = Slytherin

    Marks = 65

1. Assuming that the database MySchool is created and contains the table student, we start by creating a connection:

import sqlite3

MySchool=sqlite3.connect('schooltest.db')

curschool=MySchool.cursor()

2. To add a new record to the table, we execute the INSERT query.

import sqlite3

MySchool=sqlite3.connect('schooltest.db')

curschool=MySchool.cursor()

curschool.execute("INSERT INTO student (StudentID, name, house, marks) VALUES (5,'Sherlock',32,50);")

3. We now commit the changes to confirm them.

import sqlite3

MySchool=sqlite3.connect('schooltest.db')

curschool=MySchool.cursor()

curschool.execute("INSERT INTO student (StudentID, name, house, marks) VALUES (5,'Sherlock',32,50);")

MySchool.commit()

The new record is added to the table. You can verify this from SQLite Studio.  
You can refer to the helper text for the relevant screenshot.

**Example 2:** To accept user input for the values in the table:

1. Instead of adding known values, you can also accept user input for these values. Assuming that the database MySchool is created and contains the table student, we start by creating a connection:

import sqlite3

MySchool=sqlite3.connect('schooltest.db')

curschool=MySchool.cursor()

2. To accept user input, we use variables to store each of the values.

import sqlite3

MySchool=sqlite3.connect('schooltest.db')

curschool=MySchool.cursor()

mysid= int(input("Enter ID: "))

myname=input("Enter name: ")

myhouse=int(input("Enter house: "))

mymarks=float(input("Enter marks: "))

3. We now replaces the fixed VALUES in the INSERT query with the variables, mysid, myname, myhouse and mymarks. To do this, we use the DB-API’s parameter substitution. We put a ? as a placeholder wherever we want to use a value and then give a tuple of values as the second argument to the cursor’s execute() method.

import sqlite3

MySchool=sqlite3.connect('schooltest.db')

curschool=MySchool.cursor()

mysid= int(input("Enter ID: "))

myname=input("Enter name: ")

myhouse=int(input("Enter house: "))

mymarks=float(input("Enter marks: "))

curschool.execute("INSERT INTO student (StudentID, name, house, marks) VALUES (?,?,?,?);", (mysid,myname,myhouse,mymarks))

4. We now commit the changes.

import sqlite3

MySchool=sqlite3.connect('schooltest.db')

curschool=MySchool.cursor()

mysid= int(input("Enter ID: "))

myname=input("Enter name: ")

myhouse=int(input("Enter house: "))

mymarks=float(input("Enter marks: "))

curschool.execute("INSERT INTO student (StudentID, name, house, marks) VALUES (?,?,?,?);", (mysid,myname,myhouse,mymarks))

MySchool.commit()

|  |
| --- |
| Error handling in Python |

In Python, there are (at least) two distinguishable kinds of errors: syntax errors and exceptions. Syntax errors, also known as parsing errors, are errors in the programming syntax. In the following example, the quotation mark is missing at the end of the string, hello world. This is a syntax error.

**Example 1**

>>> print("Hello World)

SyntaxError: EOL while scanning string literal

>>>

Error handling in Python is done through the use of exceptions that are caught in try blocks and handled in except blocks.  
Let us look an example of how this is used. The following code not only accepts a user input and adds a new record but also displays a message if the operation was successful or not.

**Example 2**

Even if a statement or expression is syntactically correct, it may cause an error when an attempt is made to execute it. Errors detected during execution are called exceptions. In the following example, division by zero is an exception.

>>> 10 \* (1/0)

Traceback (most recent call last):

File "<pyshell#0>", line 1, in <module>

10 \* (1/0)

ZeroDivisionError: division by zero

>>>

Error handling in Python is done through the use of exceptions that are caught in try blocks and handled in except blocks.  
Let us look an example of how this is used. The following code not only accepts a user input and adds a new record but also displays a message if the operation was successful or not.

import sqlite3

MySchool=sqlite3.connect('schooltest.db')

mysid= int(input("Enter ID: "))

myname=input("Enter name: ")

myhouse=int(input("Enter house: "))

mymarks=float(input("Enter marks: "))

#try block to catch exceptions

try:

curschool=MySchool.cursor()

curschool.execute("INSERT INTO student (StudentID, name, house, marks) VALUES (?,?,?,?)", (mysid, myname, myhouse, mymarks))

MySchool.commit()

print ("One record added successfully.")

#except block to handle exceptions

except:

print ("Error in operation.")

MySchool.rollback()

MySchool.close()

The connection class defines the commit() and rollback() methods. Changes in database are finalised only if the execute() method runs successfully by commit() method. Otherwise, any changes are undone by the rollback() method. You can try this yourself by saving this code as a .py file and executing it.

**Try and Except**

The try statement works as follows.

* First, the try clause (the statement(s) between the try and except keywords) is executed.
* If no exception occurs, the except clause is skipped and execution of the try statement is finished
* If an exception occurs during execution of the try clause, the rest of the clause is skipped. Then the except clause is executed, and then execution continues after the try statement.
* If an exception occurs which does not match the exception named in the except clause, it is passed on to outer try statements; if no handler is found, it is an unhandled exception and execution stops with a message.

Further Reading: <https://docs.python.org/3/tutorial/errors.html>

|  |
| --- |
| Retrieving a new Record |

The SELECT query forms a result set containing all records returned as a response to a query. The execute() method uses a string representing the SELECT query statement. There are two prominent methods as per DB-API standard. The below two methods are used:

**fetchone (): -** This method fetches the next available record from the result set. It is a tuple consisting of values of each column of the fetched record. The Following code snippet retrieves and prints one record at a time till the result set is exhausted.

import sqlite3

MySchool=sqlite3.connect('schooltest.db')

sql="SELECT \* from student;"

curschool=MySchool.cursor()

curschool.execute(sql)

while True:

record=curschool.fetchone()

if record==None:

break

print (record)

**fetchall (): -** This method fetches all the remaining records in the form of a list of tuples. Each tuple corresponds to one record and contains values of each column in the table. The following code snippet fetches all records and prints them one at a time by using the 'for' statement.

import sqlite3

MySchool=sqlite3.connect('schooltest.db')

sql="SELECT \* from student;"

curschool=MySchool.cursor()

curschool.execute(sql)

result=curschool.fetchall()

for record in result:

print (record)

|  |
| --- |
| Updating a Record - Example |

import sqlite3

MySchool=sqlite3.connect('schooltest.db')

nm=input("enter name: ")

sql="SELECT \* from student WHERE name='"+nm+"';"

curschool=MySchool.cursor()

curschool.execute(sql)

record=curschool.fetchone()

print (record)

m=float(input("enter new marks: "))

sql="UPDATE student SET marks='"+str(m)+"' WHERE name='"+nm+"';"

try:

curschool.execute(sql)

MySchool.commit()

print ("record updated successfully")

except:

print ("error in update operation")

MySchool.rollback()