Python MySQL

Python can be used in database applications.

One of the most popular databases is MySQL.

MySQL Database

To be able to experiment with the code examples in this tutorial, you should have MySQL installed on your computer.

You can download a MySQL database at https://www.mysql.com/downloads/.

Install MySQL Driver

Python needs a MySQL driver to access the MySQL database.

In this tutorial we will use the driver "MySQL Connector".

We recommend that you use PIP (*pip* is the package installer for *Python*)to install "MySQL Connector".

PIP is most likely already installed in your Python environment.

Navigate your command line to the location of PIP, and type the following:

Download and install "MySQL Connector":

C:\Users*Your Name*\AppData\Local\Programs\Python\Python36-32\Scripts>python -m pip install mysql-connector-python

Now you have downloaded and installed a MySQL driver.

Test MySQL Connector

To test if the installation was successful, or if you already have "MySQL Connector" installed, create a Python page with the following content:

demo_mysql_test.py:

```
import mysql.connector
```

If the above code was executed with no errors, "MySQL Connector" is installed and ready to be used.

Create Connection

Start by creating a connection to the database.

Use the username and password from your MySQL database:

demo_mysql_connection.py:

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="yourusername",
  password="yourpassword"
)

print(mydb)
```

op: <mysql.connector.connection.MySQLConnection object ar 0x016645F0>

Now you can start querying the database using SQL statements.

Creating a Database

To create a database in MySQL, use the "CREATE DATABASE" statement:

Example

create a database named "mydatabase":

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="yourusername",
  password="yourpassword"
)

mycursor = mydb.cursor()

mycursor.execute("CREATE DATABASE mydatabase")
```

If the above code was executed with no errors, you have successfully created a database.

The MySQLCursor of mysql-connector-python (and similar libraries) is used to execute statements to communicate with the MySQL database.

Using the methods of it you can execute SQL statements, fetch data from the result sets, call procedures and many more. The cursor object consists various methods and properties to modify MySQL database.

You can create **Cursor** object using the cursor() method of the Connection object/class.

A cursor is an object which helps to execute the query and fetch the records from the database. The cursor plays a very important role in executing the query.

We can create the cursor object through the mysql.

Create a cursor object:

```
1       #python cursor_object.py
2
3       #import the library
4       import mysql.connector
5
6       # creating connection
7       conn = mysql.connector.connect(
8       host="localhost",
9       user="sammy",
10      password="password"
11      )
12      #print the connection
13      print(conn)
14      # import the cursor from the connection (conn)
15
16      mycursor = conn.cursor()
17      #print the mycursor
18      print(mycursor)
```

Output: python cursor_object.py
<mysql.connector.connection_cext.CMySQLConnection object at 0x7f520da04be0>
CMySQLCursor: (Nothing executed yet)

Line 4: We import the connector class from MySql.

Line 7 to 11: We access the connect method through the connector class, which we already import into our program. Now, we are passing our connection parameters to the connect method. The user name and password will be different according to your installation process.

Line 16: We imported the cursor method from the established connection (conn) object and created the cursor object (mycursor).

Line 18: Now, we just print this mycursor which we created on line 16, and the output shows that CMySQLCursor: (Nothing executed yet).

Method cursor.execute ():

The execute () method helps us to execute the query and return records according to the query. The syntax of the execute () function is:

execute (query, args = None)

Parameters:

- **query:** This should be a string type.
- **Arguments:** By default, the arguments are **None** because sometimes we can pass only a query like a **SELECT** query which fetches the records and does not require any values. So that's the reason for the **args=None** by default. But if we want to pass the values in the case of the **INSERT** query, then the type of the arguments must be a tuple, list, or dict only.

Returns:

- It will return the count of the numbers of rows affected during the query.

Return Type:

- The return type will be an integer (**int**).

Examples**********

Example_1: use execute () method only for query

```
#python simple_execute_function.py

#import the library
import mysql.connector

# creating connection
conn = mysql.connector.connect(
 host="localhost",
 user="sammy",
 password="password",
 database ="dbTest"
)

# import the cursor from the connection (conn)
mycursor = conn.cursor()

mycursor.execute("SELECT * FROM MOVIE")

# iterate over the result
for row in mycursor:
 print(row)

# we close the cursor and conn both
mycursor.close()
conn.close()
```

Output: python simple_execute_function.py

(1, 'Bruce Almighty', 2003)
(2, 'Kung Fu panda', 2014)
(3, 'Kung Fu panda', 2014)
(4, 'Frozen', 2014)
(5, 'Frozen2', 2020)
(6, 'Iron Man', 2013)

Line 11: We added one more parameter name to the database. Now, our python code will try to connect with this MySql database (dbTest) only.

Line 15: We created a cursor object (mycursor).

Line 17: We run a simple query SELECT through the execute function.

Line 20 to 21: We iterated over the results fetched by the cursor object and noticed that all records are returned in tuples.

Check if Database Exists

You can check if a database exist by listing all databases in your system by using the "SHOW DATABASES" statement:

Example

Return a list of your system's databases:

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="yourusername",
  password="yourpassword"
)

mycursor = mydb.cursor()

mycursor.execute("SHOW DATABASES")

for x in mycursor:
  print(x)
```

op:

('information_scheme',)
('mydatabase',)

('performance_schema',)
('sys',)

Or you can try to access the database when making the connection:

Example

Try connecting to the database "mydatabase":

```
import mysql.connector

mydb = mysql.connector.connect(
 host="localhost",
 user="yourusername",
 password="yourpassword",
 database="mydatabase"
)
```

Creating a Table

To create a table in MySQL, use the "CREATE TABLE" statement.

Make sure you define the name of the database when you create the connection

Example

Create a table named "customers":

```
import mysql.connector

mydb = mysql.connector.connect(
 host="localhost",
 user="yourusername",
 password="yourpassword",
 database="mydatabase"
)

mycursor = mydb.cursor()

mycursor.execute("CREATE TABLE customers (name VARCHAR(255), address VARCHAR(255))")
```

If the above code was executed with no errors, you have now successfully created a table.

Check if Table Exists

You can check if a table exist by listing all tables in your database with the "SHOW TABLES" statement:

Example

Return a list of your system's databases:

```
import mysql.connector

mydb = mysql.connector.connect(
 host="localhost",
 user="yourusername",
 password="yourpassword",
 database="mydatabase"
)

mycursor = mydb.cursor()

mycursor.execute("SHOW TABLES")

for x in mycursor:
 print(x)
```

op:

('customers',)

Primary Key

When creating a table, you should also create a column with a unique key for each record.

This can be done by defining a PRIMARY KEY.

We use the statement "INT AUTO_INCREMENT PRIMARY KEY" which will insert a unique number for each record. Starting at 1, and increased by one for each record.

Example

Create primary key when creating the table:

```
import mysql.connector

mydb = mysql.connector.connect(
 host="localhost",
 user="yourusername",
 password="yourpassword",
 database="mydatabase"
)
```

```
mycursor = mydb.cursor()

mycursor.execute("CREATE TABLE customers (id INT AUTO_INCREMENT PRIMARY KEY, name VARCHAR(255), address VARCHAR(255))")
```

If the table already exists, use the ALTER TABLE keyword:

Example

Create primary key on an existing table:

```
import mysql.connector

mydb = mysql.connector.connect(
 host="localhost",
 user="yourusername",
 password="yourpassword",
 database="mydatabase"
)

mycursor = mydb.cursor()

mycursor.execute("ALTER TABLE customers ADD COLUMN id INT AUTO_INCREMENT PRIMARY KEY")
```

Insert Into Table

To fill a table in MySQL, use the "INSERT INTO" statement.

Example

Insert a record in the "customers" table:

```
import mysql.connector

mydb = mysql.connector.connect(
 host="localhost",
 user="yourusername",
 password="yourpassword",
 database="mydatabase"
)

mycursor = mydb.cursor()

sql = "INSERT INTO customers (name, address) VALUES (%s, %s)"
val = ("John", "Highway 21")
mycursor.execute(sql, val)

mydb.commit()

print(mycursor.rowcount, "record inserted.")
```

op: 1 record inserted.

Insert Multiple Rows

To insert multiple rows into a table, use the `executemany()` method.

The second parameter of the `executemany()` method is a list of tuples, containing the data you want to insert:

Example

Fill the "customers" table with data:

```
import mysql.connector

mydb = mysql.connector.connect(
 host="localhost",
 user="yourusername",
 password="yourpassword",
 database="mydatabase"
)

mycursor = mydb.cursor()

sql = "INSERT INTO customers (name, address) VALUES (%s, %s)"
val = [
 ('Peter', 'Lowstreet 4'),
 ('Amy', 'Apple st 652'),
 ('Hannah', 'Mountain 21'),
 ('Michael', 'Valley 345'),
 ('Sandy', 'Ocean blvd 2'),
 ('Betty', 'Green Grass 1'),
 ('Richard', 'Sky st 331'),
 ('Susan', 'One way 98'),
 ('Vicky', 'Yellow Garden 2'),
 ('Ben', 'Park Lane 38'),
 ('William', 'Central st 954'),
 ('Chuck', 'Main Road 989'),
('Viola', 'Sideway 1633')
]

mycursor.executemany(sql, val)

mydb.commit()

print(mycursor.rowcount, "was inserted.")
```

op : 13 record was inserted.

Get Inserted ID

You can get the id of the row you just inserted by asking the cursor object.

Note: If you insert more than one row, the id of the last inserted row is returned.

Example

Insert one row, and return the ID:

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="yourusername",
  password="yourpassword",
  database="mydatabase"
)

mycursor = mydb.cursor()

sql = "INSERT INTO customers (name, address) VALUES (%s, %s)"
val = ("Michelle", "Blue Village")
mycursor.execute(sql, val)

mydb.commit()

print("1 record inserted, ID:", mycursor.lastrowid)
```

op: 1 record inserted, ID: 15

Select From a Table

To select from a table in MySQL, use the "SELECT" statement:

Example

Select all records from the "customers" table, and display the result:

```
import mysql.connector

mydb = mysql.connector.connect(
 host="localhost",
 user="yourusername",
 password="yourpassword",
 database="mydatabase"
)

mycursor = mydb.cursor()
```

```
mycursor.execute("SELECT * FROM customers")

myresult = mycursor.fetchall()

for x in myresult:
 print(x)
```

op:
(1, 'John', 'Highway 21')
(2, 'Peter', 'Lowstreet 27')
(3, 'Amy', 'Apple st 652')
(4, 'Hannah', 'Mountain 21')
(5, 'Michael', 'Valley 345')
(6, 'Sandy', 'Ocean blvd 2')
(7, 'Betty', 'Green Grass 1')
(8, 'Richard', 'Sky st 331')
(9, 'Susan', 'One way 98')
(10, 'Vicky', 'Yellow Garden 2')
(11, 'Ben', 'Park Lane 38')
(12, 'William', 'Central st 954')
(13, 'Chuck', 'Main Road 989')
(14, 'Viola', 'Sideway 1633')
(15, 'Michelle', 'Blue Village')

Note: We use the `fetchall()` method, which fetches all rows from the last executed statement.

Selecting Columns

To select only some of the columns in a table, use the "SELECT" statement followed by the column name(s):

Example

Select only the name and address columns:

```
import mysql.connector

mydb = mysql.connector.connect(
 host="localhost",
 user="yourusername",
 password="yourpassword",
 database="mydatabase"
)

mycursor = mydb.cursor()
```

```
mycursor.execute("SELECT name, address FROM customers")

myresult = mycursor.fetchall()

for x in myresult:
 print(x)
```

op:

```
('John', 'Highway 21')
('Peter', 'Lowstreet 27')
('Amy', 'Apple st 652')
('Hannah', 'Mountain 21')
('Michael', 'Valley 345')
('Sandy', 'Ocean blvd 2')
('Betty', 'Green Grass 1')
('Richard', 'Sky st 331')
('Susan', 'One way 98')
('Vicky', 'Yellow Garden 2')
('Ben', 'Park Lane 38')
('William', 'Central st 954')
('Chuck', 'Main Road 989')
('Viola', 'Sideway 1633')
('Michelle', 'Blue Village')
```

Select With a Filter

When selecting records from a table, you can filter the selection by using the "WHERE" statement:

Example

Select record(s) where the address is "Park Lane 38": result:

```
import mysql.connector

mydb = mysql.connector.connect(
 host="localhost",
 user="yourusername",
 password="yourpassword",
 database="mydatabase"
)

mycursor = mydb.cursor()

sql = "SELECT * FROM customers WHERE address ='Park Lane 38'"

mycursor.execute(sql)

myresult = mycursor.fetchall()
```

```
for x in myresult:
 print(x)
```

op:
(11, 'Ben', 'Park Lane 38')

Python MySQL Order By

Sort the Result

Use the ORDER BY statement to sort the result in ascending or descending order.

The ORDER BY keyword sorts the result ascending by default. To sort the result in descending order, use the DESC keyword.

Example

Sort the result alphabetically by name: result:

```
import mysql.connector

mydb = mysql.connector.connect(
 host="localhost",
 user="yourusername",
 password="yourpassword",
 database="mydatabase"
)

mycursor = mydb.cursor()

sql = "SELECT * FROM customers ORDER BY name"

mycursor.execute(sql)

myresult = mycursor.fetchall()

for x in myresult:
 print(x)
```

op: (3, 'Amy', 'Apple st 652')
(11, 'Ben', 'Park Lane 38')
(7, 'Betty', 'Green Grass 1')
(13, 'Chuck', 'Main Road 989')
(4, 'Hannah', 'Mountain 21')
(1, 'John', 'Highway 21')
(5, 'Michael', 'Valley 345')
(15, 'Michelle', 'Blue Village') (2, 'Peter', 'Lowstreet 27')
(8, 'Richard', 'Sky st 331')

(6, 'Sandy', 'Ocean blvd 2')
(9, 'Susan', 'One way 98')
(10, 'Vicky', 'Yellow Garden 2')
(14, 'Viola', 'Sideway 1633')
(12, 'William', 'Central st 954')

Python MySQL Delete From By

You can delete records from an existing table by using the "DELETE FROM" statement:

Example

Delete any record where the address is "Mountain 21":

```
import mysql.connector

mydb = mysql.connector.connect(
 host="localhost",
 user="yourusername",
 password="yourpassword",
 database="mydatabase"
)

mycursor = mydb.cursor()

sql = "DELETE FROM customers WHERE address = 'Mountain 21'"

mycursor.execute(sql)

mydb.commit()

print(mycursor.rowcount, "record(s) deleted")
```

op: 1 record(s) deleted

Important!: Notice the statement: `mydb.commit()`. It is required to make the changes, otherwise no changes are made to the table.
Notice the WHERE clause in the DELETE syntax: The WHERE clause specifies which record(s) that should be deleted. If you omit the WHERE clause, all records will be deleted!

Python MySQL Drop Table

Delete a Table

You can delete an existing table by using the "DROP TABLE" statement:

Example

Delete the table "customers":

```
import mysql.connector

mydb = mysql.connector.connect(
 host="localhost",
 user="yourusername",
 password="yourpassword",
 database="mydatabase"
)

mycursor = mydb.cursor()

sql = "DROP TABLE customers"

mycursor.execute(sql)
```

Python MySQL Update Table

Update Table

You can update existing records in a table by using the "UPDATE" statement:

Example

Overwrite the address column from "Valley 345" to "Canyon 123":

```
import mysql.connector

mydb = mysql.connector.connect(
 host="localhost",
 user="yourusername",
 password="yourpassword",
 database="mydatabase"
)

mycursor = mydb.cursor()

sql = "UPDATE customers SET address = 'Canyon 123' WHERE address = 'Valley 345'"

mycursor.execute(sql)

mydb.commit()

print(mycursor.rowcount, "record(s) affected")
```

Important!: Notice the statement: `mydb.commit()`. It is required to make the changes, otherwise no changes are made to the table.
Notice the WHERE clause in the UPDATE syntax: The WHERE clause specifies which record or records that should be updated. If you omit the WHERE clause, all records will be updated!

Python MySQL Join

Join Two or More Tables

You can combine rows from two or more tables, based on a related column between them, by using a JOIN statement.

Consider you have a "users" table and a "products" table:

users

{ id: 1, name: 'John', fav: 154},
{ id: 2, name: 'Peter', fav: 154},
{ id: 3, name: 'Amy', fav: 155},
{ id: 4, name: 'Hannah', fav:},
{ id: 5, name: 'Michael', fav:}

products

{ id: 154, name: 'Chocolate Heaven' },
{ id: 155, name: 'Tasty Lemons' },
{ id: 156, name: 'Vanilla Dreams' }

These two tables can be combined by using users' `fav` field and products' `id` field.

Join users and products to see the name of the users favorite product:

```
import mysql.connector

mydb = mysql.connector.connect(
 host="localhost",
 user="yourusername",
 password="yourpassword",
 database="mydatabase"
)

mycursor = mydb.cursor()

sql = "SELECT \
 users.name AS user, \
 products.name AS favorite \
 FROM users \
 INNER JOIN products ON users.fav = products.id"

mycursor.execute(sql)

myresult = mycursor.fetchall()

for x in myresult:
 print(x)
```

op:
('John', 'Chocolate Heaven')
('Peter', 'Chocolate Heaven')
('Amy', 'Tasty Lemon')

- Amarjeet Mohanty
Department of CSE