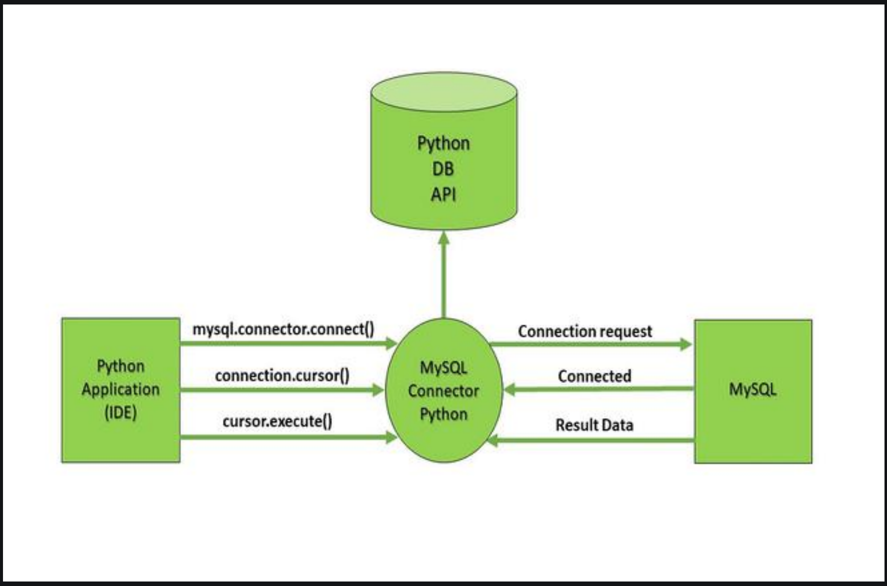
How to Connect Python with SQL Database?

Python is a high-level, general-purpose, and very popular programming language. Basically, it was designed with an emphasis on code readability, and programmers can express their concepts in fewer lines of code. We can also use Python with SQL. In this article, we will learn how to connect SQL with Python using the 'MySQL Connector Python module. The diagram given below illustrates how a connection request is sent to MySQL connector Python, how it gets accepted from the database and how the cursor is executed with result data.



SQL connection with Python

Connecting MySQL with Python

To create a connection between the MySQL database and Python, the **connect()** method of **mysql.connector** module is used. We pass the database details like HostName, username, and the password in the method call, and then the method returns the connection object.

The following steps are required to connect SQL with Python:

Step 1: Download and Install the free MySQL database from here.

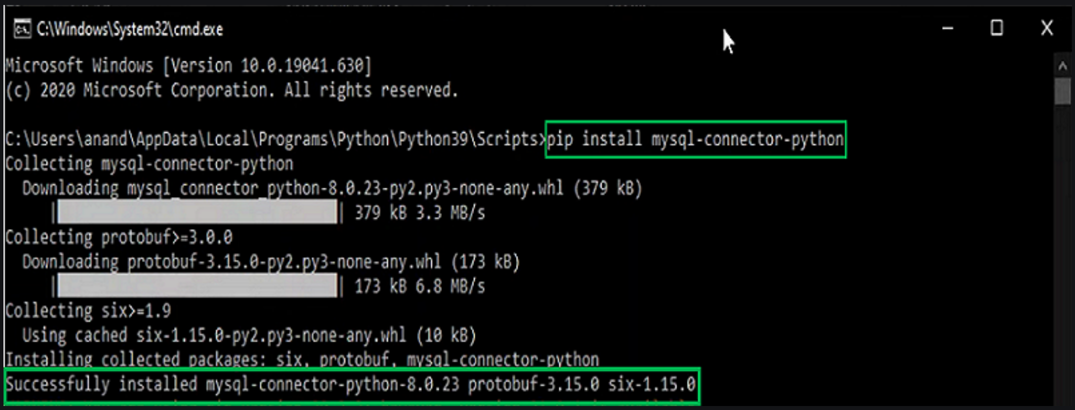
Step 2: After installing the MySQL database, open your Command prompt.

Step 3: Navigate your Command prompt to the location of PIP. Click here to see, How to install PIP?

Step 4: Now run the commands given below to download and install "MySQL Connector". Here, mysql.connector statement will help you to communicate with the MySQL database.

Download and install "MySQL Connector"

```
pip install mysql-connector-python
```

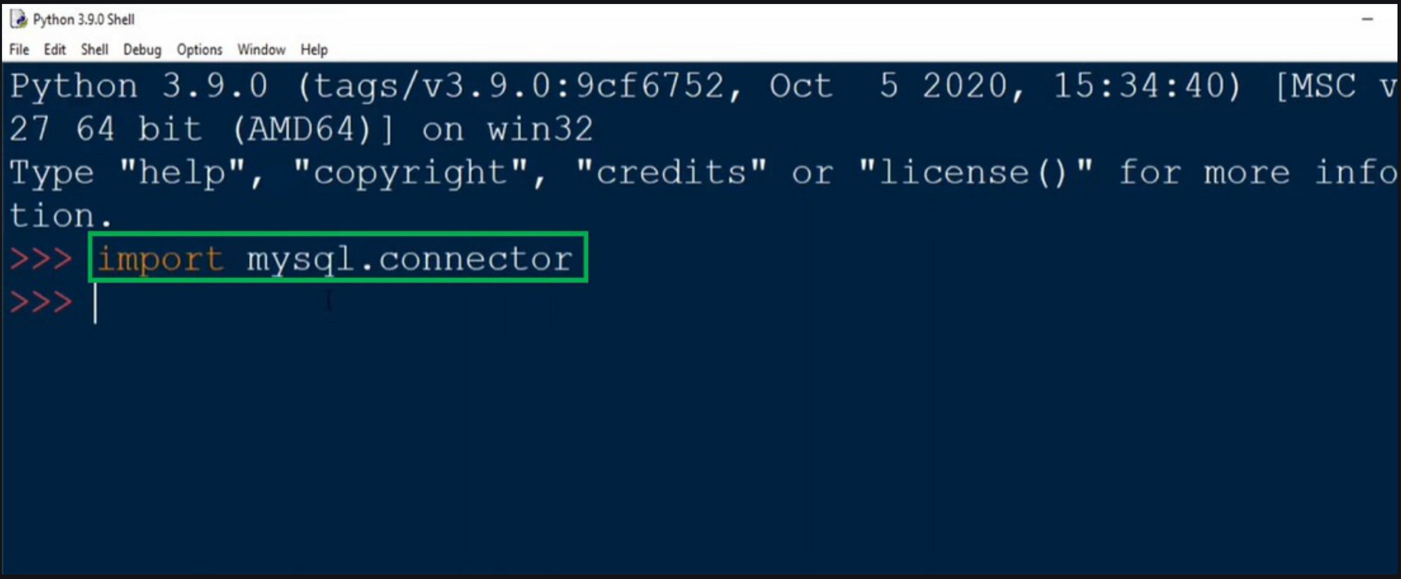


```
Microsoft Windows [Version 10.0.19041.630]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\anand\AppData\Local\Programs\Python\Python39\Scripts>pip install mysql-connector-python
Collecting mysql-connector-python
  Downloading mysql_connector_python-8.0.23-py2.py3-none-any.whl (379 kB)
     |████████████████████████████████| 379 kB 3.3 MB/s
Collecting protobuf>=3.0.0
  Downloading protobuf-3.15.0-py2.py3-none-any.whl (173 kB)
     |████████████████████████████████| 173 kB 6.8 MB/s
Collecting six>=1.9
  Using cached six-1.15.0-py2.py3-none-any.whl (10 kB)
Installing collected packages: six, protobuf, mysql-connector-python
Successfully installed mysql-connector-python-8.0.23 protobuf-3.15.0 six-1.15.0
```

Step 5: Test MySQL Connector

To check if the installation was successful, or if you already installed "MySQL Connector", go to your IDE and run the given below code :

```
import mysql.connector
```



```
Python 3.9.0 (tags/v3.9.0:9cf6752, Oct  5 2020, 15:34:40) [MSC v
27 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more info
tion.
>>> import mysql.connector
>>> 
```

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

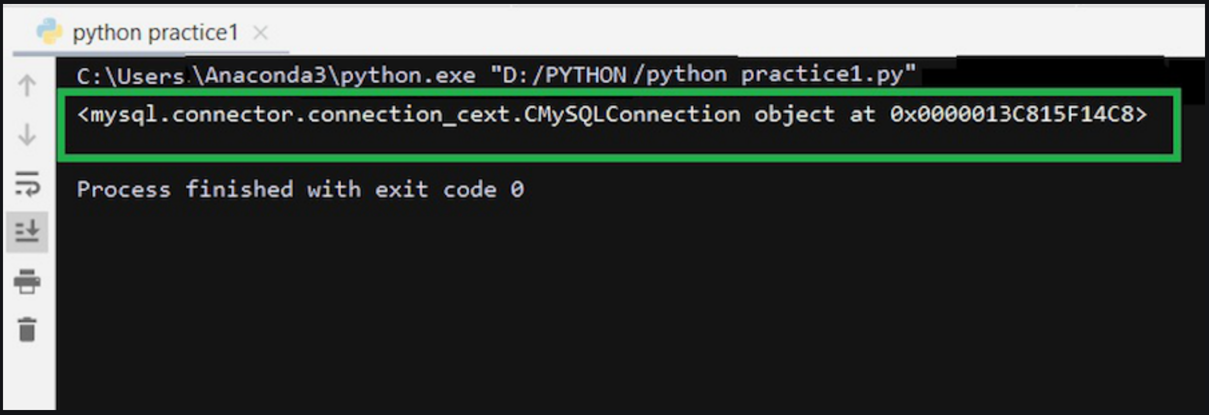
Step 6: Create Connection

Now to connect SQL with Python, run the code given below in your IDE.

Python3

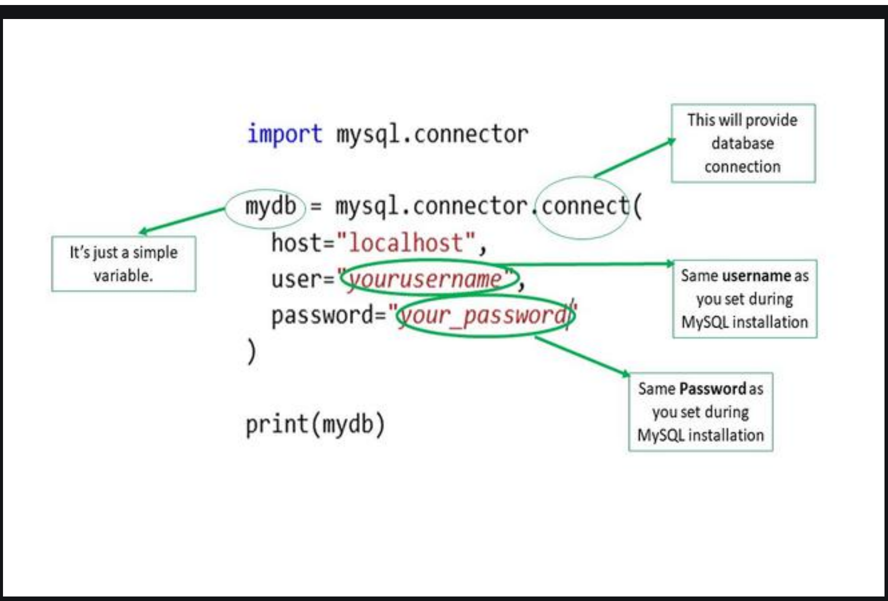
```
# Importing module
import mysql.connector

# Creating connection object
mydb = mysql.connector.connect(
    host = "localhost",
    user = "yourusername",
    password = "your_password"
)

# Printing the connection object
print(mydb)
```

Output:



```
C:\Users\Anaconda3\python.exe "D:/PYTHON /python practice1.py"
<mysql.connector.connection_cext.CMySQLConnection object at 0x0000013C815F14C8>

Process finished with exit code 0
```

Here, in the above code:



```
import mysql.connector

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

print(mydb)
```

This will provide database connection
It's just a simple variable.
Same username as you set during MySQL installation
Same Password as you set during MySQL installation

Code Info

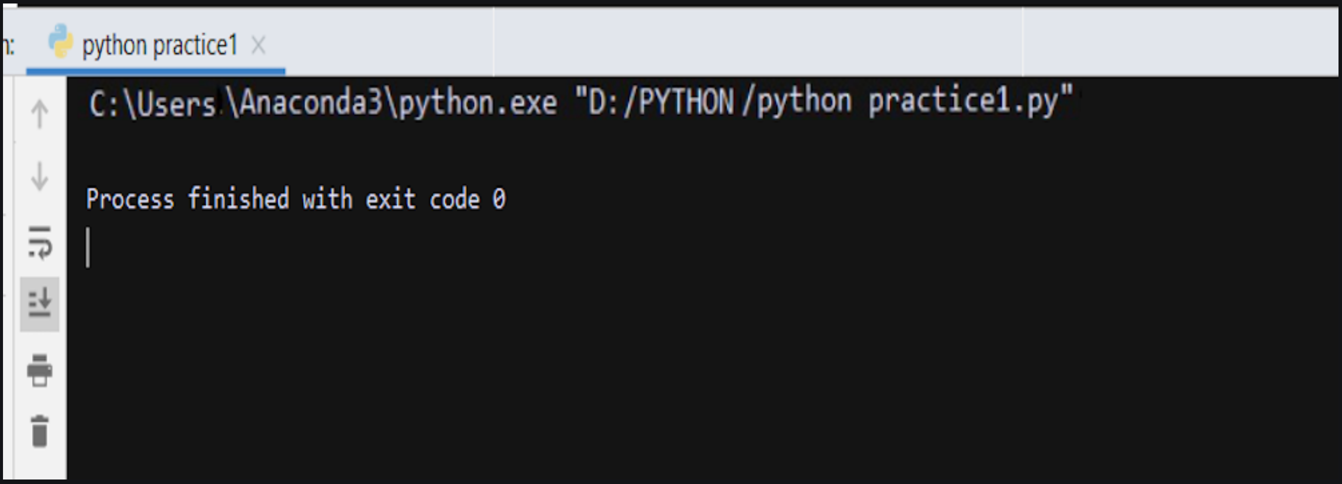
Creating MySQL Database

To create a database, we will use CREATE DATABASE database_name statement and we will execute this statement by creating an instance of the 'cursor' class.

Python3

```python
import mysql.connector

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

# Creating an instance of 'cursor' class
# which is used to execute the 'SQL'
# statements in 'Python'
cursor = mydb.cursor()

# Creating a database with a name
# 'geeksforgeeks' execute() method
# is used to compile a SQL statement
# below statement is used to create
# the 'geeksforgeeks' database
cursor.execute("CREATE DATABASE geeksforgeeks")
```

Output:



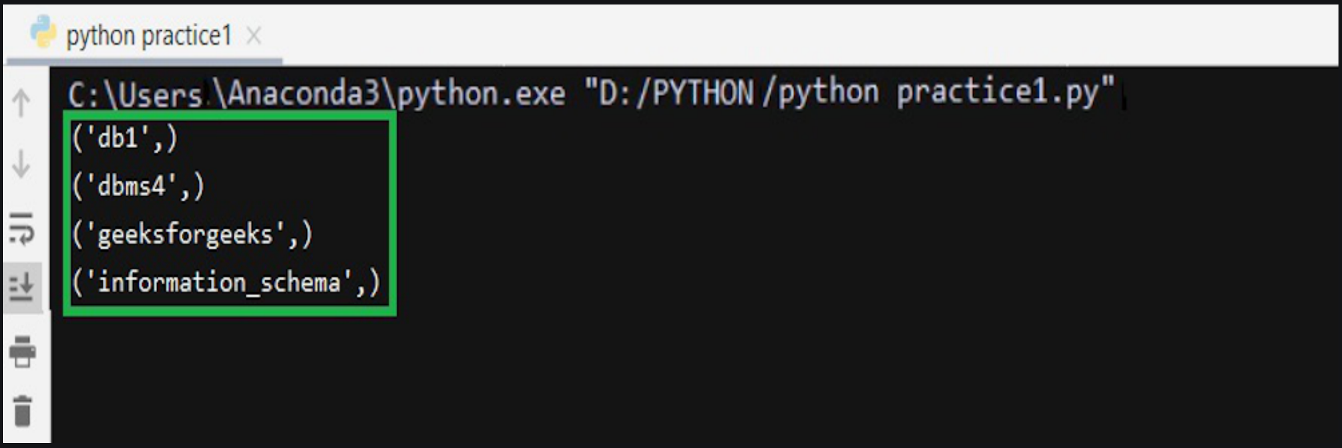
```
C:\Users \Anaconda3\python.exe "D:/PYTHON /python practice1.py"

Process finished with exit code 0
```

If the database with the name 'geeksforgeeks' already exists then you will get an error, otherwise no error. So make sure that the new database that you are creating does not have the same name as the database already you created or exists previously. Now to check the databases that you created, use *"SHOW DATABASES" - SQL* *statement* i.e. cursor.execute("SHOW DATABASES")

Python3

```
import mysql.connector

mydb = mysql.connector.connect(
    host = "localhost",
    user = "root",
    password = "1234"
)

# Creating an instance of 'cursor' class
# which is used to execute the 'SQL'
# statements in 'Python'
cursor = mydb.cursor()

# Show database
cursor.execute("SHOW DATABASE")

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

Output:



```
C:\Users\Anaconda3\python.exe "D:/PYTHON/python practice1.py"
('db1',)
('dbms4',)
('geeksforgeeks',)
('information_schema',)
```

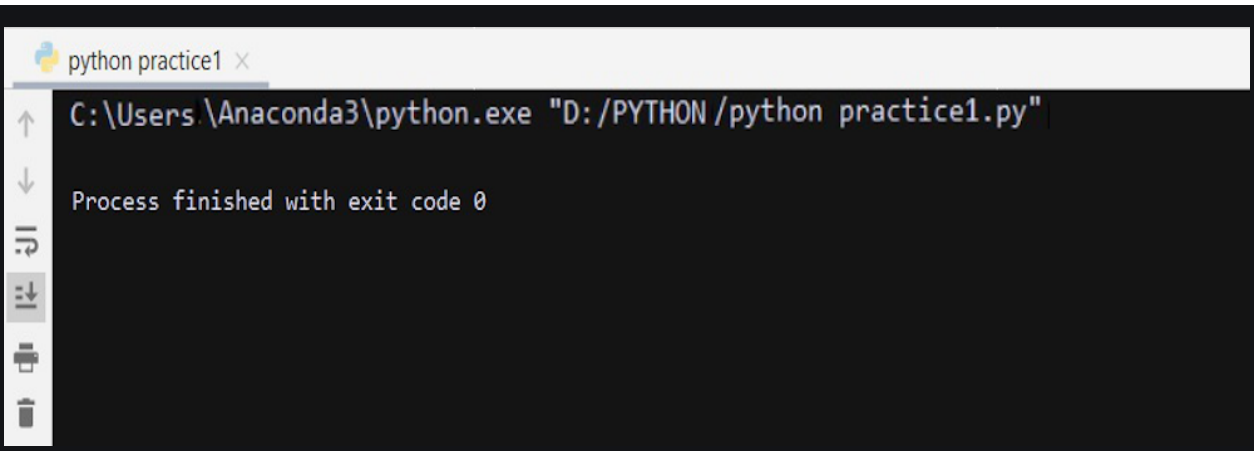
Creating Tables

Now to create tables in a database, first, we have to select a database and for that, we will pass *database* = "*NameofDatabase*" as your fourth parameter in connect() function. Since we have created a database with the name 'geekforgeeks' above, so we will use that and create our tables. We will use *CREATE TABLE gfg (variableName1 datatype, variableName2 datatype)* statement to create our table with the name 'gfg'.

Python3

```
import mysql.connector

mydb = mysql.connector.connect(
    host = "localhost",
    user = "yourusername",
    password = "your_password",
    database = "geeksforgeeks"


cursor = mydb.cursor()

# Creating a table called 'gfg' in the
# 'geeksforgeeks' database
cursor.execute("CREATE TABLE gfg (name VARCHAR(255), user_name VARCHAR(255))")
```

Output:



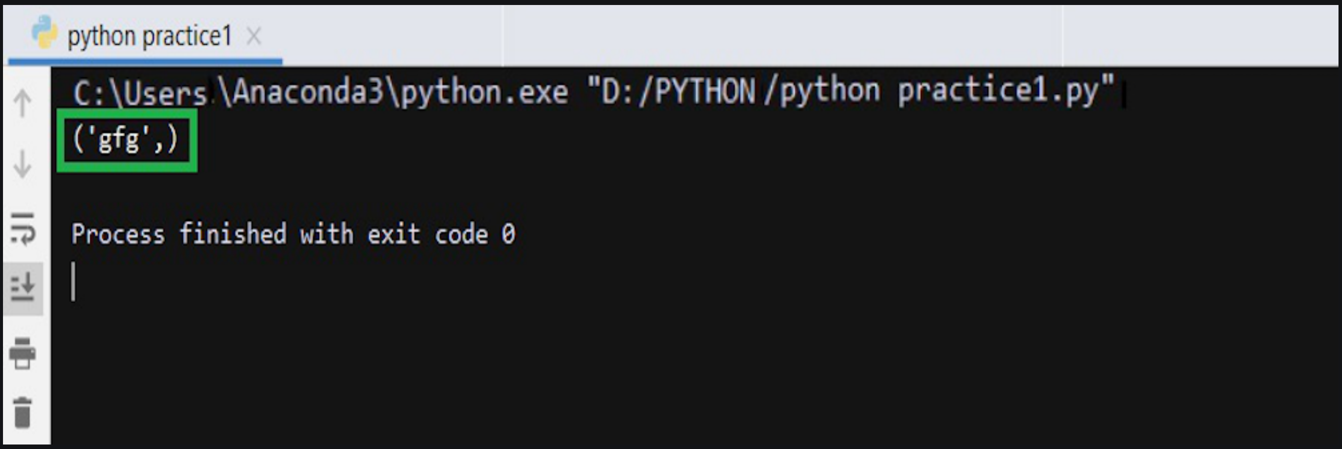
```
C:\Users\Anaconda3\python.exe "D:/PYTHON /python practice1.py"

Process finished with exit code 0
```

If the table with the name 'gfg' already exists, you will get an error, otherwise no error. So make sure that the new table that you are creating does not have the same name as the table already you created or exists previously. Now to check tables that you created, use *"SHOW TABLES" - SQL statement* i.e. cursor.execute("SHOW TABLES").

Python3

```
import mysql.connector

mydb = mysql.connector.connect(
    host = "localhost",
    user = "root
    password = "1234",
    database = "geeksforgeeks"
)

cursor = mydb.cursor()

# Show existing tables
cursor.execute("SHOW TABLES")

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

Output:



```
C:\Users\Anaconda3\python.exe "D:/PYTHON /python practice1.py"
('gfg',)

Process finished with exit code 0
```

Notes:

mysql.connector allows Python programs to access MySQL databases.
connect() method of the MySQL Connector class with the arguments will connect to MySQL and would return a MySQLConnection object if the connection is established successfully.

- **user = "yourusername"**, here "yourusername" should be the same username as you set during MySQL installation.
- **password = "your_password"**, here "your_password" should be the same password as you set during MySQL installation.
- **cursor()** is used to execute the SQL statements in Python.
- **execute()** method is used to compile a SQL statement.