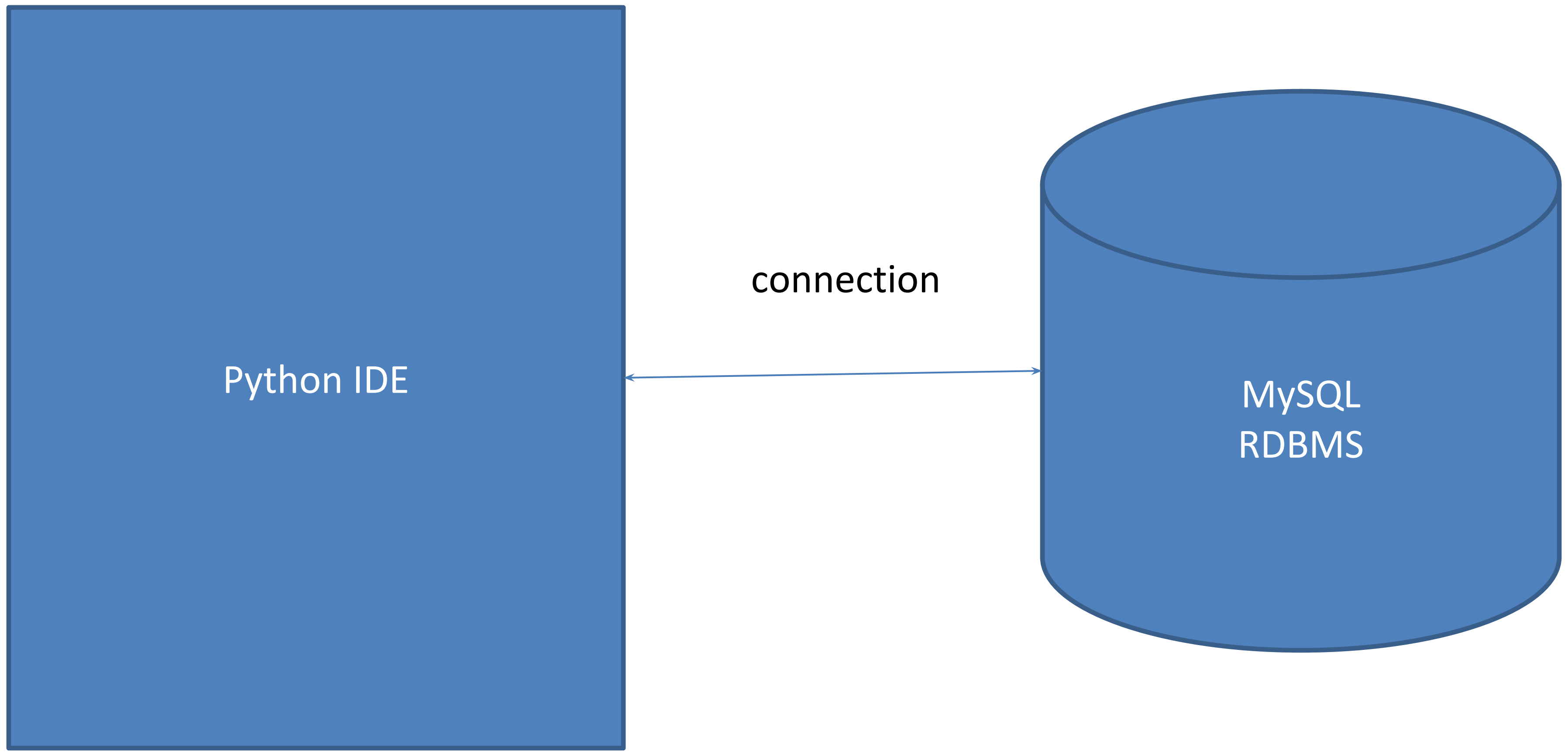
Python Programming - X

By Nimesh Kumar Dagur


Python IDE
connection
MySQL
RDBMS

Introduction to database programming in Python

- The Python standard for database interfaces is the Python DB-API.
- Most Python database interfaces adhere to this standard.
- You can choose the right database for your application.
- Python Database API supports a wide range of database servers such as −
- MySQL
- PostgreSQL
- Microsoft SQL Server
- Informix
- Oracle
- Sybase

Introduction to database programming in Python

- You must download a separate DB API module for each database you need to access.
- For example, if you need to access an Oracle database as well as a MySQL database, you must download both the Oracle and the MySQL database modules.

Python DB API

- 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

Working with MySQL Database in Python

- Install MySql Server
- Install MySql Connector for Python

Install MySql Connector for Python

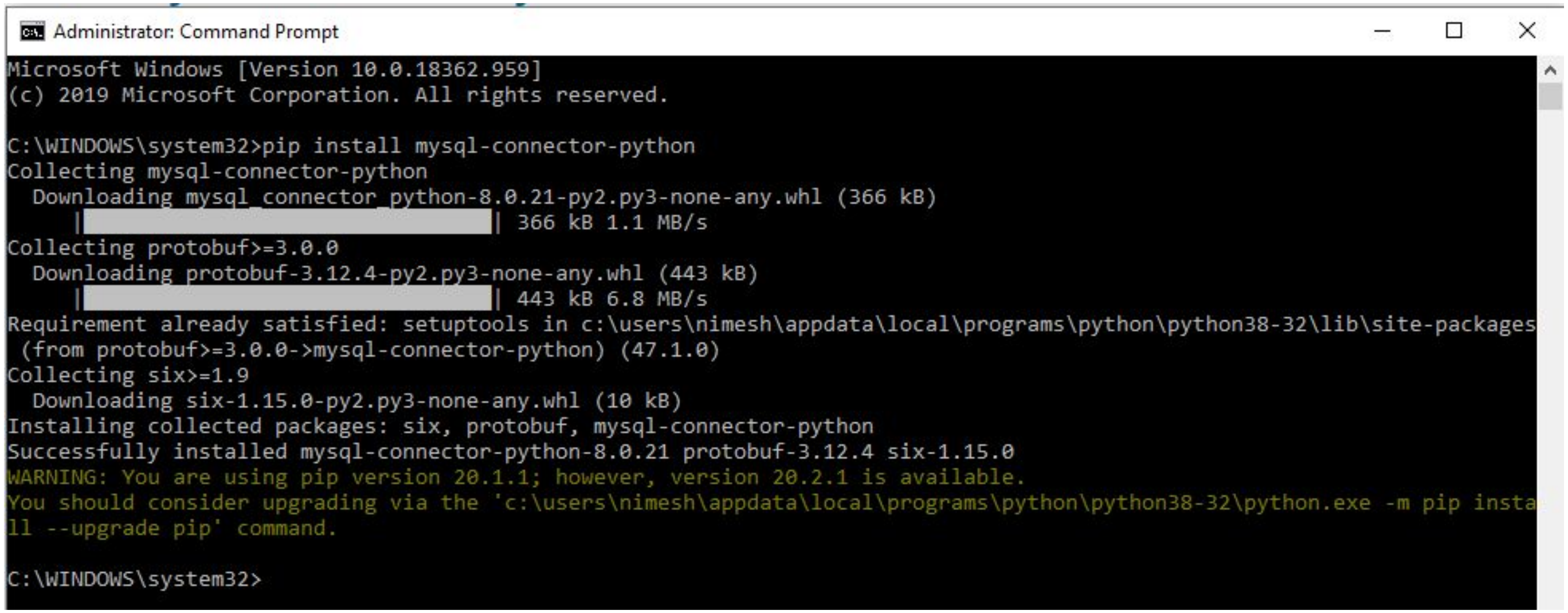
pip install mysql-connector-python

Or

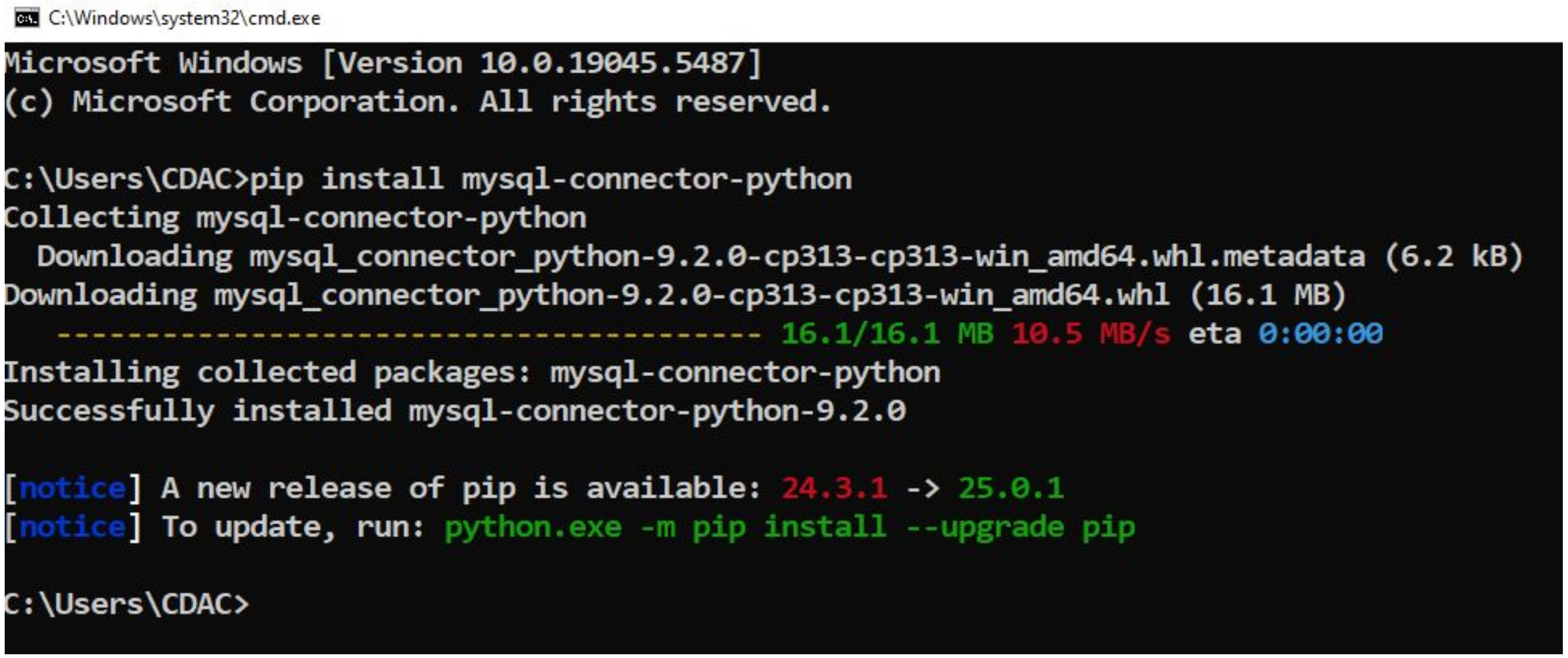
pip install mysql-connector-python==8.0.11

Or

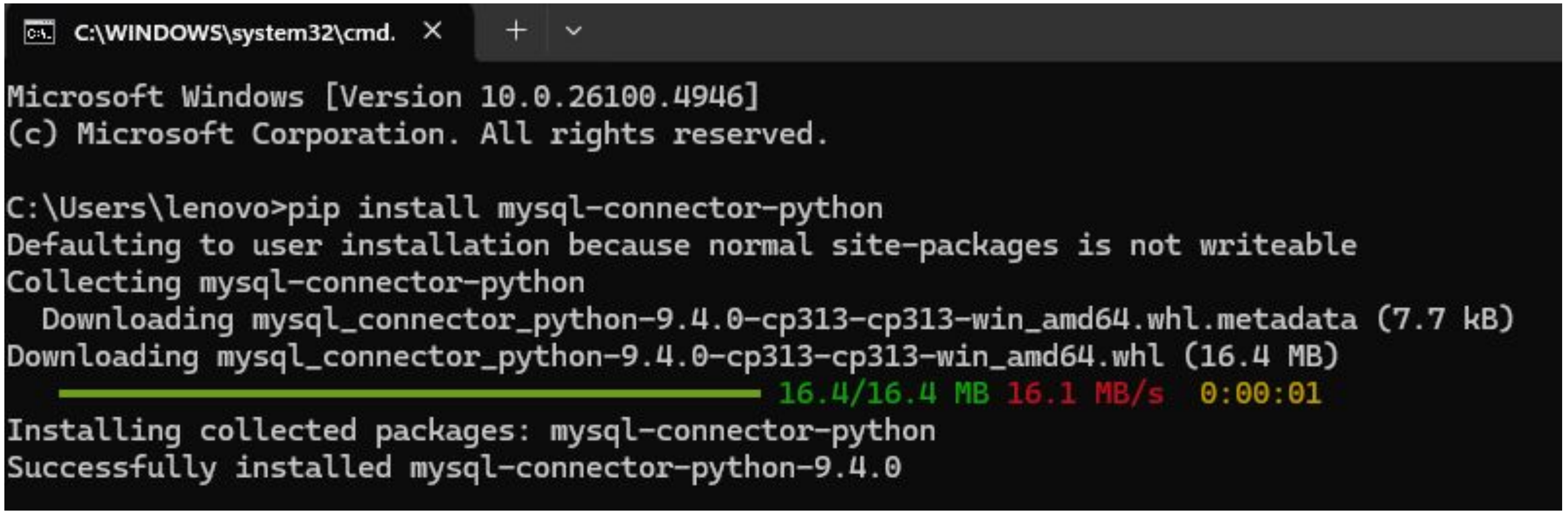
python -m pip install --trusted-host pypi.org --trusted-host files.pythonhosted.org --trusted-host pypi.python.org mysql-connector-python



```
Microsoft Windows [Version 10.0.18362.959]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>pip install mysql-connector-python
Collecting mysql-connector-python
  Downloading mysql_connector_python-8.0.21-py2.py3-none-any.whl (366 kB)
     |████████████████████████████████| 366 kB 1.1 MB/s
Collecting protobuf>=3.0.0
  Downloading protobuf-3.12.4-py2.py3-none-any.whl (443 kB)
     |████████████████████████████████| 443 kB 6.8 MB/s
Requirement already satisfied: setuptools in c:\users\nimesh\appdata\local\programs\python\python38-32\lib\site-packages
 (from protobuf>=3.0.0->mysql-connector-python) (47.1.0)
Collecting six>=1.9
  Downloading six-1.15.0-py2.py3-none-any.whl (10 kB)
Installing collected packages: six, protobuf, mysql-connector-python
Successfully installed mysql-connector-python-8.0.21 protobuf-3.12.4 six-1.15.0
WARNING: You are using pip version 20.1.1; however, version 20.2.1 is available.
You should consider upgrading via the 'c:\users\nimesh\appdata\local\programs\python\python38-32\python.exe -m pip insta
ll --upgrade pip' command.

C:\WINDOWS\system32>
```



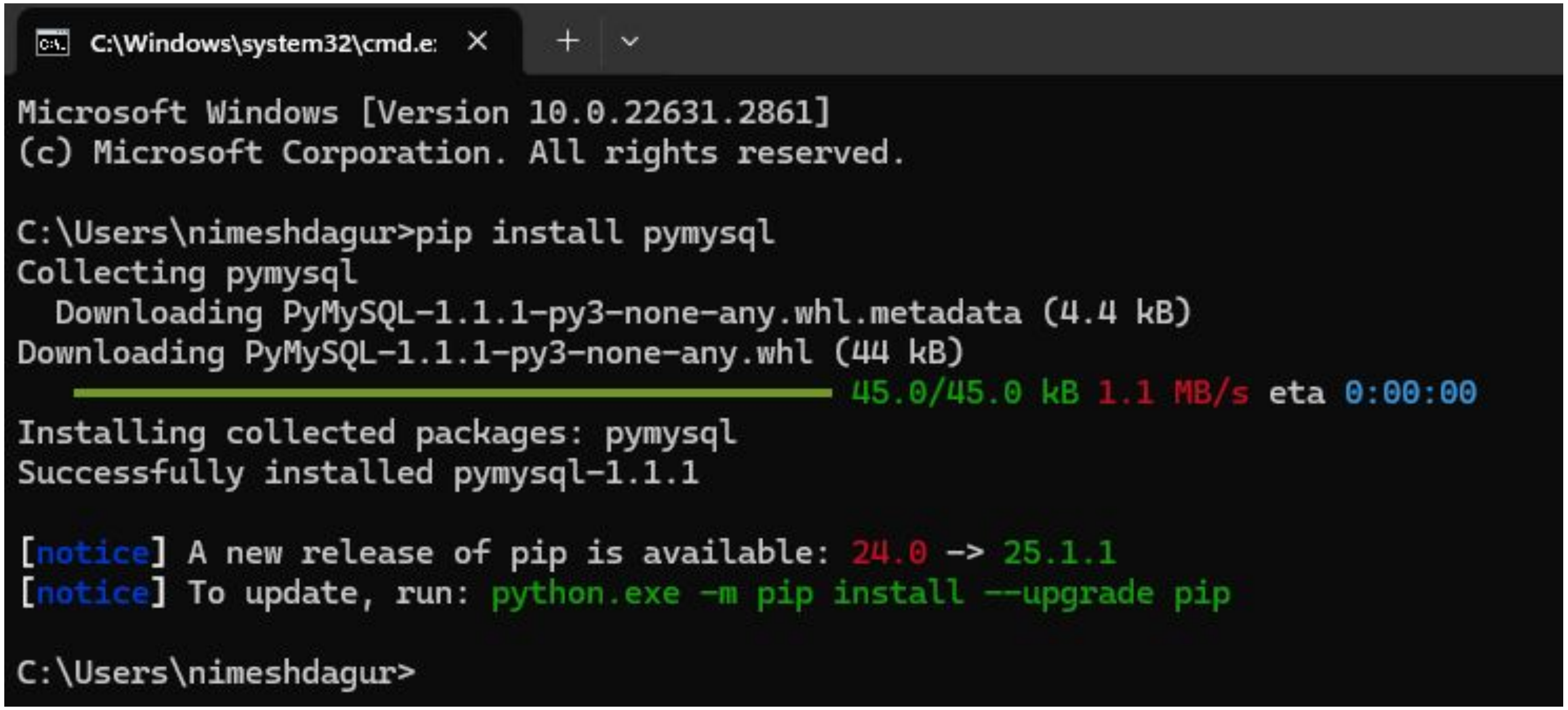
```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.19045.5487]
(c) Microsoft Corporation. All rights reserved.

C:\Users\CDAC>pip install mysql-connector-python
Collecting mysql-connector-python
  Downloading mysql_connector_python-9.2.0-cp313-cp313-win_amd64.whl.metadata (6.2 kB)
Downloading mysql_connector_python-9.2.0-cp313-cp313-win_amd64.whl (16.1 MB)
   ---------------------------------------- 16.1/16.1 MB 10.5 MB/s eta 0:00:00
Installing collected packages: mysql-connector-python
Successfully installed mysql-connector-python-9.2.0

[notice] A new release of pip is available: 24.3.1 -> 25.0.1
[notice] To update, run: python.exe -m pip install --upgrade pip

C:\Users\CDAC>
```



```
C:\WINDOWS\system32\cmd.
Microsoft Windows [Version 10.0.26100.4946]
(c) Microsoft Corporation. All rights reserved.

C:\Users\lenovo>pip install mysql-connector-python
Defaulting to user installation because normal site-packages is not writeable
Collecting mysql-connector-python
  Downloading mysql_connector_python-9.4.0-cp313-cp313-win_amd64.whl.metadata (7.7 kB)
Downloading mysql_connector_python-9.4.0-cp313-cp313-win_amd64.whl (16.4 MB)
   ---------------------------------------- 16.4/16.4 MB 16.1 MB/s  0:00:01
Installing collected packages: mysql-connector-python
Successfully installed mysql-connector-python-9.4.0
```

Alternatively, you can use pymysql connector also to connect with MySQL server through python

Step1: install pymysql



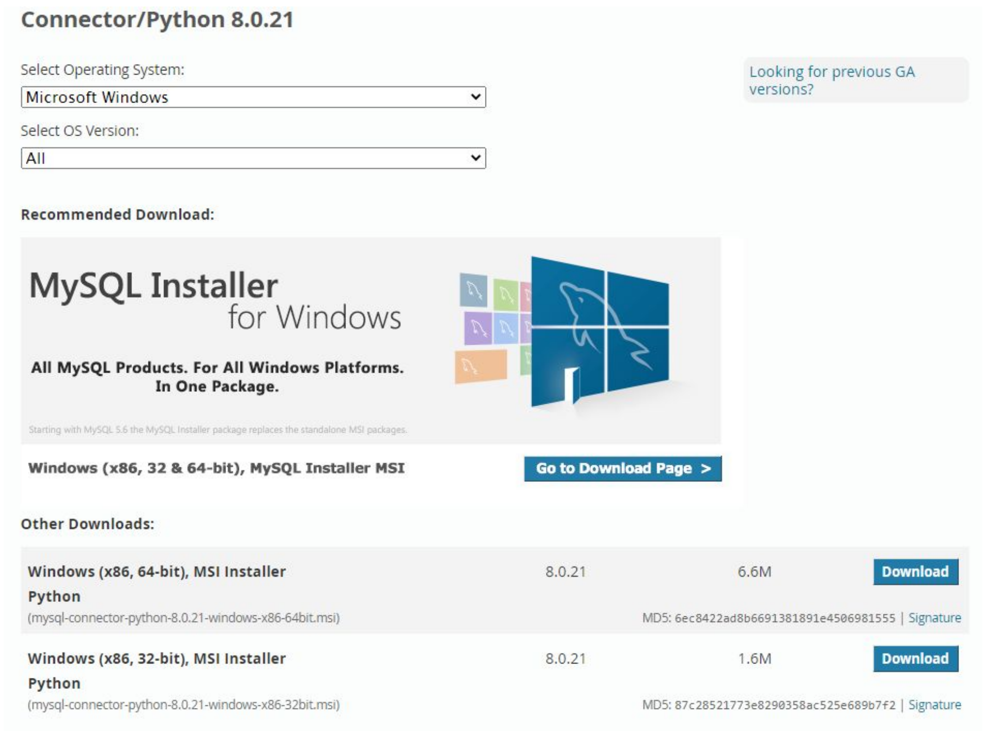
```
Microsoft Windows [Version 10.0.22631.2861]
(c) Microsoft Corporation. All rights reserved.

C:\Users\nimeshdagur>pip install pymysql
Collecting pymysql
  Downloading PyMySQL-1.1.1-py3-none-any.whl.metadata (4.4 kB)
Downloading PyMySQL-1.1.1-py3-none-any.whl (44 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 45.0/45.0 kB 1.1 MB/s eta 0:00:00
Installing collected packages: pymysql
Successfully installed pymysql-1.1.1

[notice] A new release of pip is available: 24.0 -> 25.1.1
[notice] To update, run: python.exe -m pip install --upgrade pip

C:\Users\nimeshdagur>
```

Step2:   import pymysql

Install MySql Connector for Python

Download and Install MySQL Connector Python on Windows from following url:

https://dev.mysql.com/downloads/connector/python/



Connector/Python 8.0.21

Select Operating System:
Microsoft Windows

Select OS Version:
All

Looking for previous GA versions?

Recommended Download:

MySQL Installer for Windows

All MySQL Products. For All Windows Platforms. In One Package.

Starting with MySQL 5.6 the MySQL Installer package replaces the standalone MSI packages.

Windows (x86, 32 & 64-bit), MySQL Installer MSI

Other Downloads:

Windows (x86, 64-bit), MSI Installer Python (mysql-connector-python-8.0.21-windows-x86-64bit.msi)	8.0.21	6.6M	Download MD5: 6ec8422ad8b6691381891e4506981555 | Signature
Windows (x86, 32-bit), MSI Installer Python (mysql-connector-python-8.0.21-windows-x86-32bit.msi)	8.0.21	1.6M	Download MD5: 87c28521773e8290358ac525e689b7f2 | Signature

Connect MySQL from Python

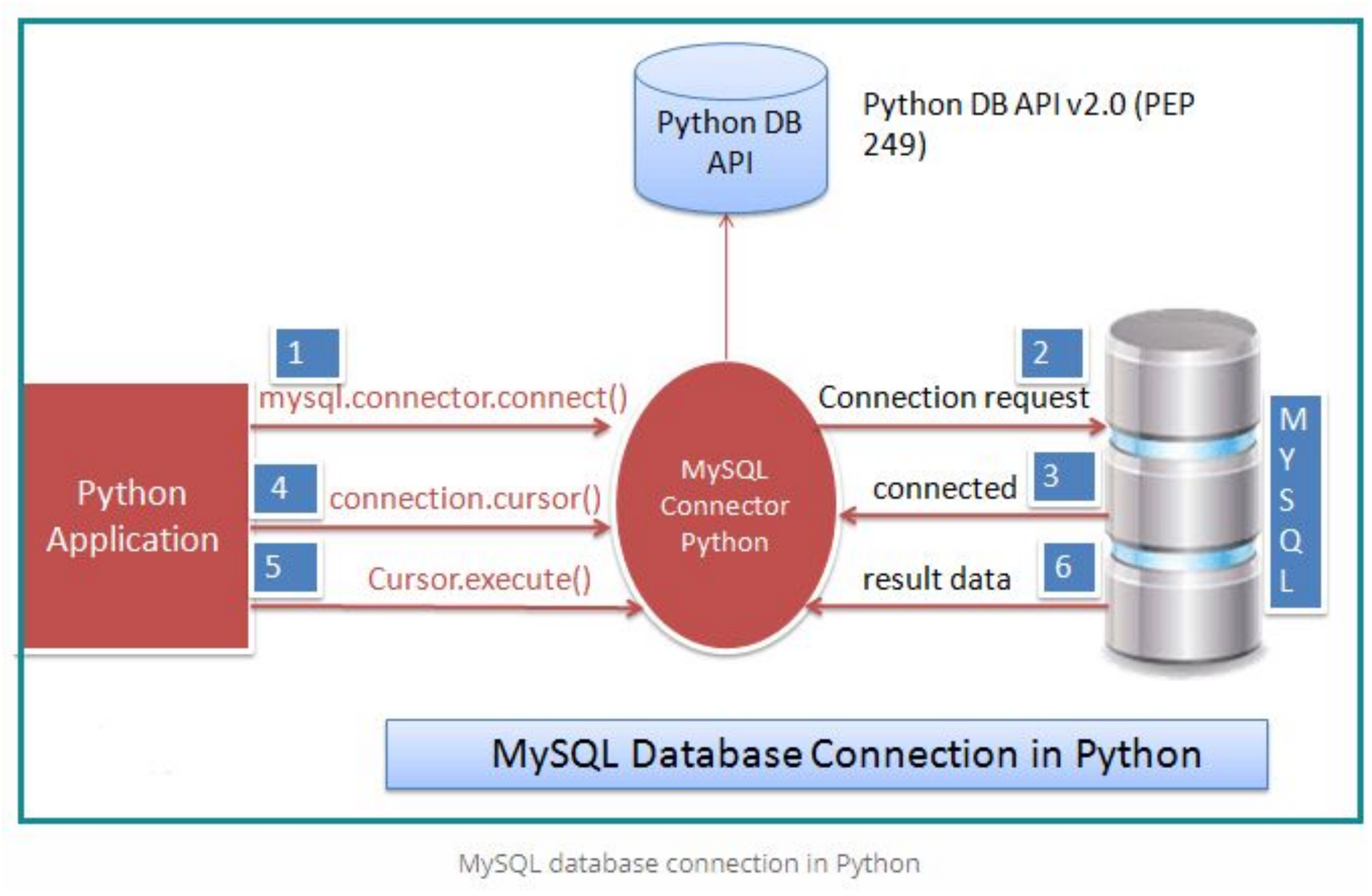
- You need to know the following detail of the MySQL server to perform the connection from Python.
- **Username** – i.e., the username that you use to work with MySQL Server. The default username for the MySQL database is a **root**
- **Password** – Password is given by the user at the time of installing the MySQL database. If you are using root then you won't need the password.
- **Host Name** – is the server name or Ip address on which MySQL is running. if you are running on localhost, then you can use localhost, or it's IP, i.e. 127.0.0.1
- **Database Name** – Database name to which you want to connect. Here we are using Database named '**cdacdb**' because we have already created this for our example.

Steps to connect MySQL database in Python using MySQL Connector Python

- Install MySQL Connector Python using pip.
- Use the **mysql.connector.connect()** method of MySQL Connector Python with required parameters to connect MySQL.
- Use the connection object returned by a connect() method to create a **cursor** object to perform Database Operations.
- The **cursor.execute()** to execute SQL queries from Python.
- Close the Cursor object using a **cursor.close()** and MySQL database connection using **connection.close()** after your work completes.
- Catch Exception if any that may occur during this process.

Note: if you are using pymysql connector then use pymysql in place of mysql.connector in the mentioned code

Steps to connect MySQL database in Python using MySQL Connector Python



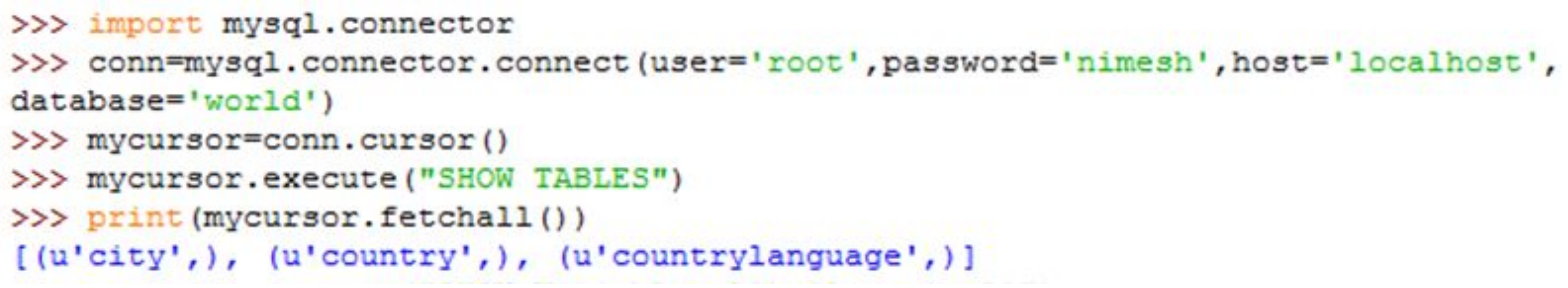
MySQL database connection in Python

Example: Python code to connect MySQL Database

```python
import mysql.connector
from mysql.connector import Error

try:
    connection = mysql.connector.connect(host='localhost',
                                         database='cdacdb',
                                         user='root',
                                         password='cdac123')
    if connection.is_connected():
        db_Info = connection.get_server_info()
        print("Connected to MySQL Server version ", db_Info)
        cursor = connection.cursor()
        cursor.execute("select database();")
        record = cursor.fetchone()
        print("You're connected to database: ", record)

except Error as e:
    print("Error while connecting to MySQL", e)
finally:
    if (connection.is_connected()):
        cursor.close()
        connection.close()
        print("MySQL connection is closed")
```

```
Connected to MySQL Server version  5.7.13-log
You're connected to database:  ('cdacdb',)
MySQL connection is closed
```

Importing MySQL API module & connecting with the database



```
>>> import mysql.connector
>>> conn=mysql.connector.connect(user='root',password='nimesh',host='localhost',
database='world')
>>> mycursor=conn.cursor()
>>> mycursor.execute("SHOW TABLES")
>>> print(mycursor.fetchall())
[(u'city',), (u'country',), (u'countrylanguage',)]
```

READ Operation

- READ Operation on any database means to fetch some useful information from the database.
- Once our database connection is established, you are ready to make a query into this database.
- You can use either **fetchone()** method to fetch single record or **fetchall()** method to fetech multiple values from a database table.
- **fetchone():** It fetches the next row of a query result set.
- A result set is an object that is returned when a cursor object is used to query a table.
- **fetchall():** It fetches all the rows in a result set.
- If some rows have already been extracted from the result set, then it retrieves the remaining rows from the result set.
- **rowcount:** This is a read-only attribute and returns the number of rows that were affected by an execute() method.

Performing Transactions

- Transactions are a mechanism that ensures data consistency.
- Transactions have the following four properties:
- **Atomicity:** Either a transaction completes or nothing happens at all.
- **Consistency:** A transaction must start in a consistent state and leave the system in a consistent state.
- **Isolation:** Intermediate results of a transaction are not visible outside the current transaction.
- **Durability:** Once a transaction was committed, the effects are persistent, even after a system failure.
- The Python DB API provides two methods to either ***commit*** or ***rollbac****k* a transaction.

COMMIT Operation

- Commit is the operation, which gives a green signal to database to finalize the changes, and after this operation, no change can be reverted back.
- Syntax: to call **commit** method

```
db.commit()
```

ROLLBACK Operation

- If you are not satisfied with one or more of the changes and you want to revert back those changes completely, then use **rollback()** method.
- Syntax: to call **rollback()** method.

```
db.rollback()
```

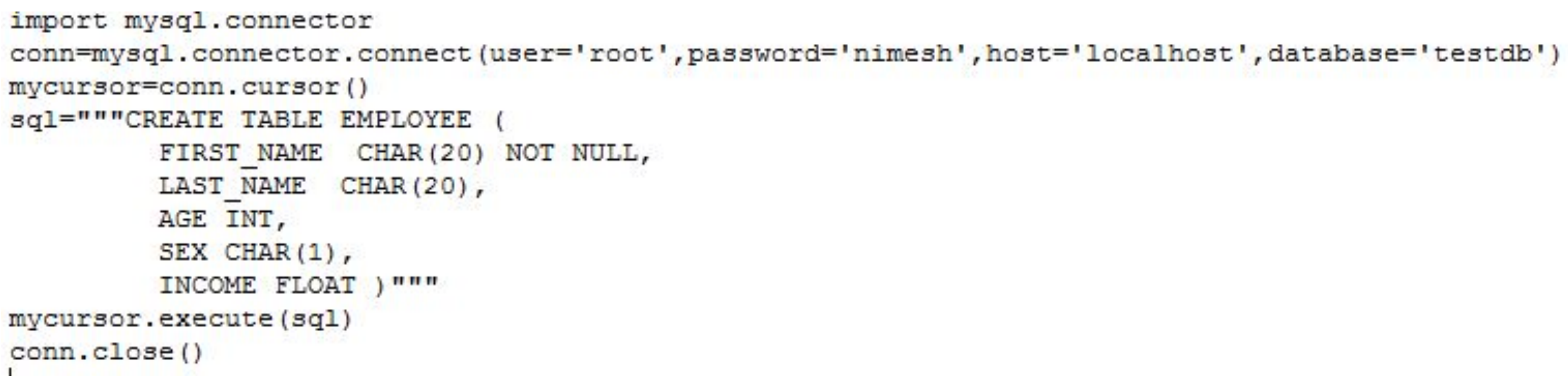
Disconnecting Database

- To disconnect Database connection, use close() method.

```
db.close()
```

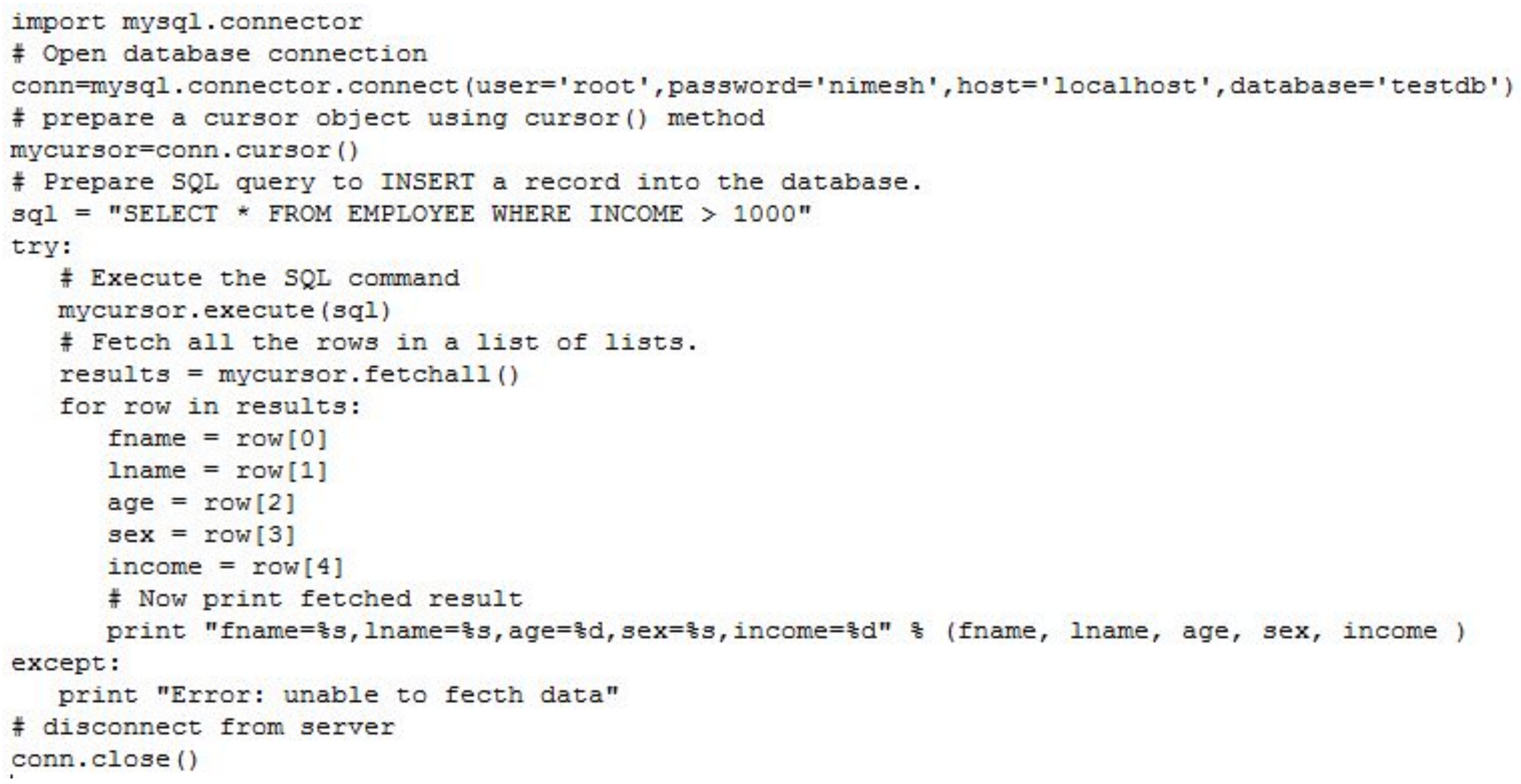
- If the connection to a database is closed by the user with the close() method, any outstanding transactions are rolled back by the DB.

Create MySql Table Using Python

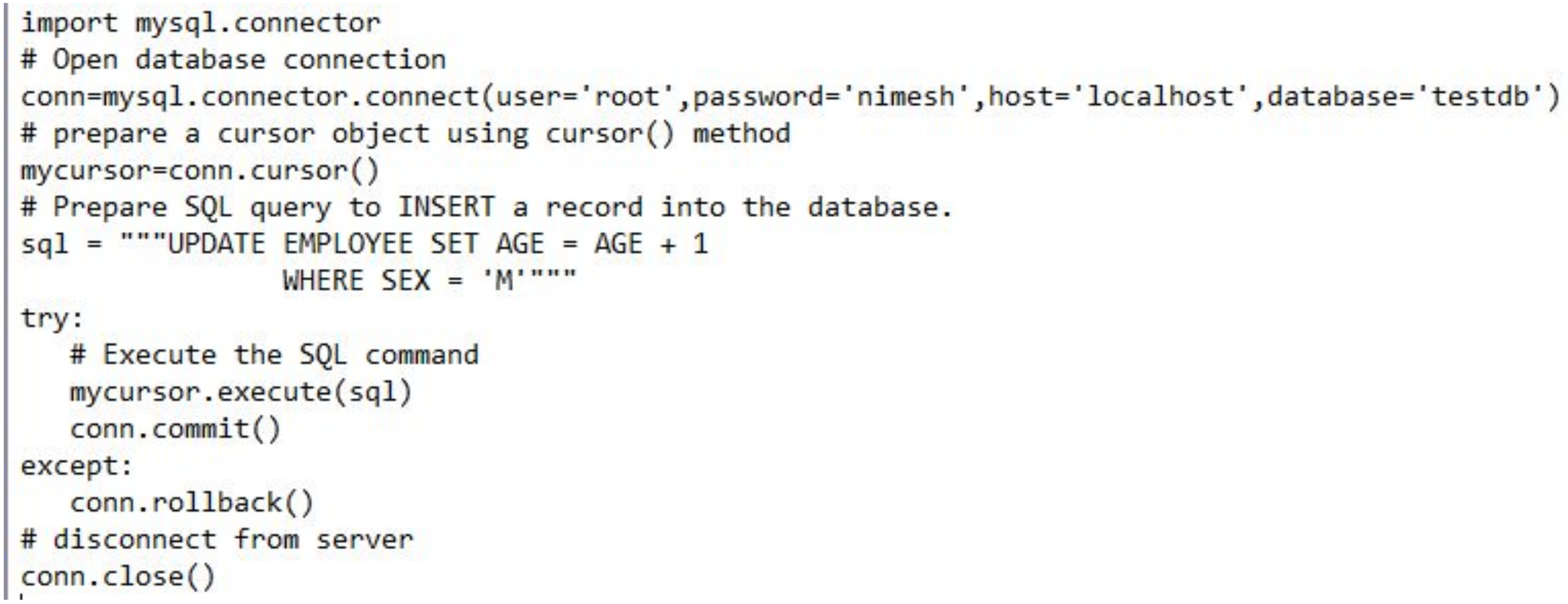


```
import mysql.connector
conn=mysql.connector.connect(user='root',password='nimesh',host='localhost',database='testdb')
mycursor=conn.cursor()
sql="""CREATE TABLE EMPLOYEE (
         FIRST_NAME  CHAR(20) NOT NULL,
         LAST_NAME  CHAR(20),
         AGE INT,
         SEX CHAR(1),
         INCOME FLOAT )"""
mycursor.execute(sql)
conn.close()
```

```
import mysql.connector
conn=mysql.connector.connect(user='root',password='nimesh',host='localhost',database='testdb')
mycursor=conn.cursor()
sql="""INSERT INTO EMPLOYEE(FIRST_NAME,
         LAST_NAME, AGE, SEX, INCOME)
         VALUES ('Nimesh', 'Dagur', 29, 'M', 37500)"""
try:
   # Execute the SQL command
   mycursor.execute(sql)
   # Commit your changes in the database
   conn.commit()
except:
   # Rollback in case there is any error
   conn.rollback()
# disconnect from server
conn.close()
```

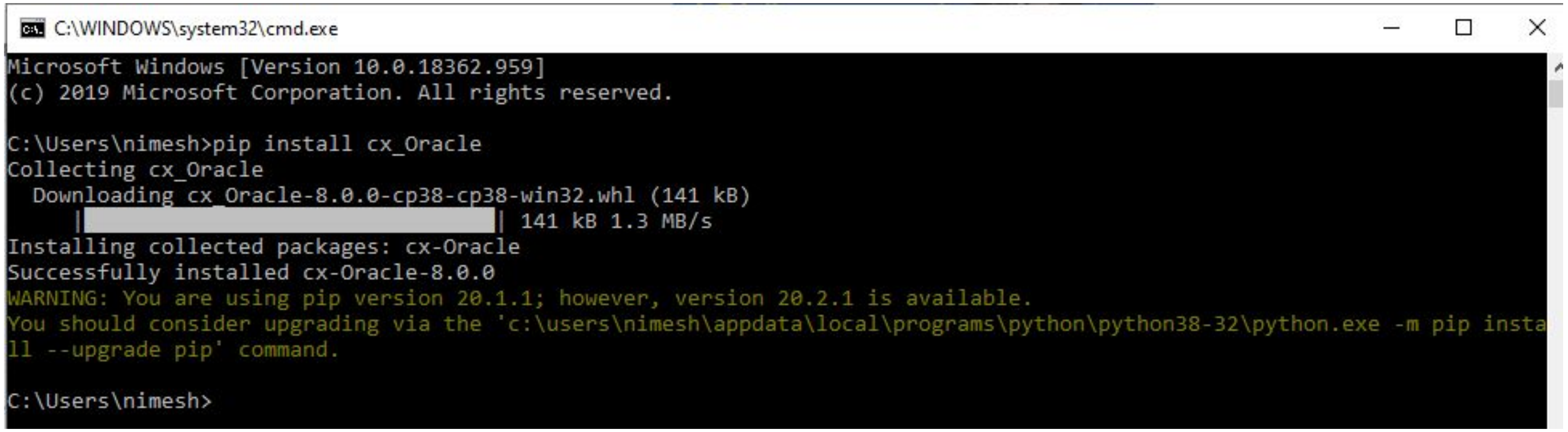


```
import mysql.connector
# Open database connection
conn=mysql.connector.connect(user='root',password='nimesh',host='localhost',database='testdb')
# prepare a cursor object using cursor() method
mycursor=conn.cursor()
# Prepare SQL query to INSERT a record into the database.
sql = "SELECT * FROM EMPLOYEE WHERE INCOME > 1000"
try:
   # Execute the SQL command
   mycursor.execute(sql)
   # Fetch all the rows in a list of lists.
   results = mycursor.fetchall()
   for row in results:
      fname = row[0]
      lname = row[1]
      age = row[2]
      sex = row[3]
      income = row[4]
      # Now print fetched result
      print "fname=%s,lname=%s,age=%d,sex=%s,income=%d" % (fname, lname, age, sex, income )
except:
   print "Error: unable to fecth data"
# disconnect from server
conn.close()
```



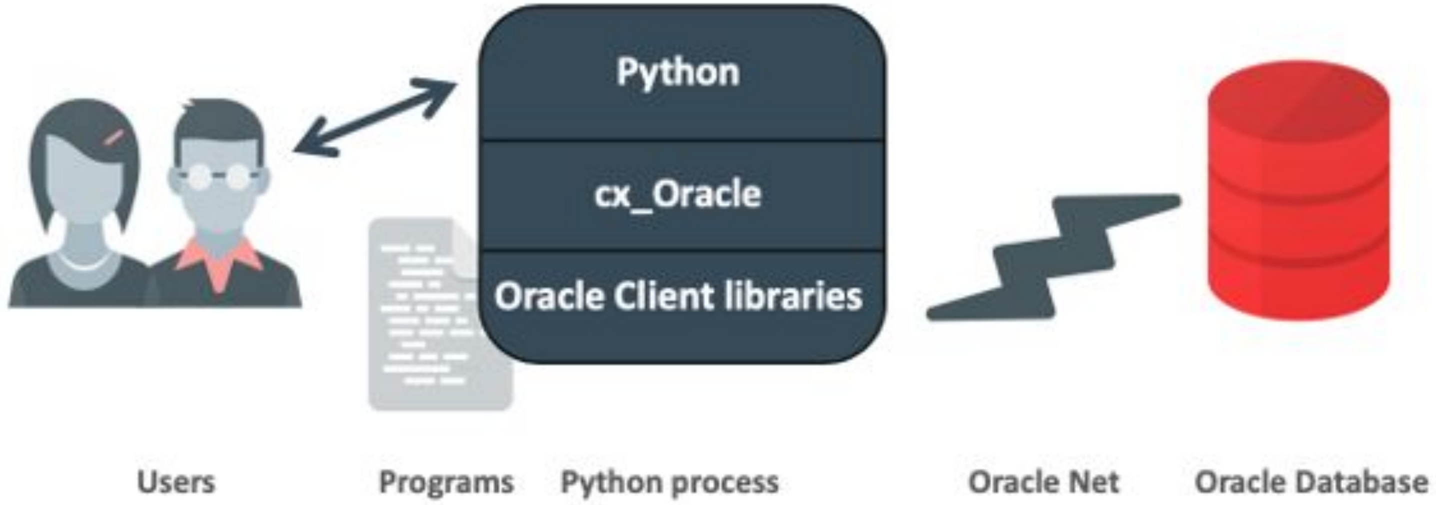
```
import mysql.connector
# Open database connection
conn=mysql.connector.connect(user='root',password='nimesh',host='localhost',database='testdb')
# prepare a cursor object using cursor() method
mycursor=conn.cursor()
# Prepare SQL query to INSERT a record into the database.
sql = """UPDATE EMPLOYEE SET AGE = AGE + 1
                WHERE SEX = 'M'"""
try:
   # Execute the SQL command
   mycursor.execute(sql)
   conn.commit()
except:
   conn.rollback()
# disconnect from server
conn.close()
```

Install oracle Connector for Python



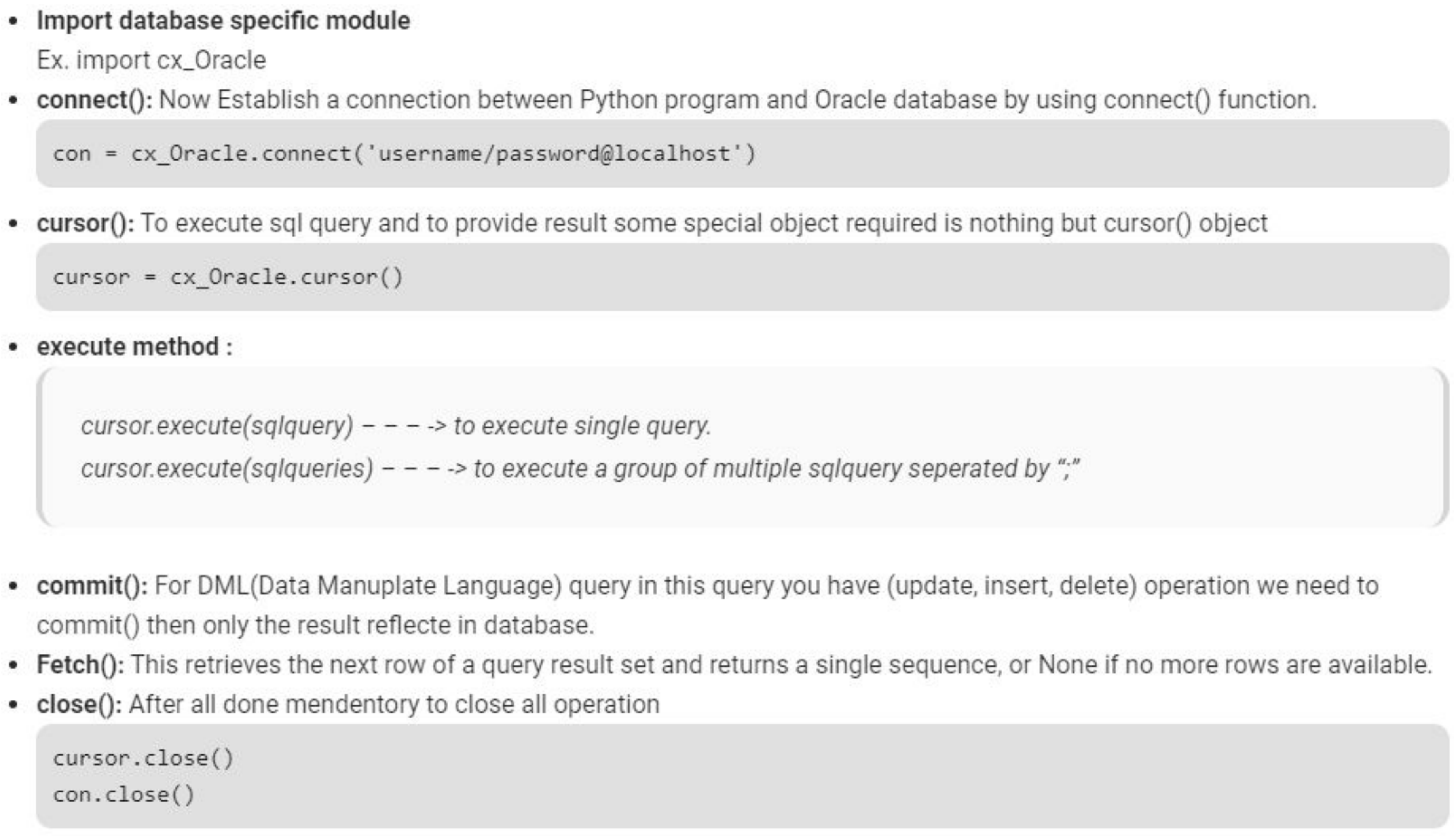
C:\WINDOWS\system32\cmd.exe

```
Microsoft Windows [Version 10.0.18362.959]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\nimesh>pip install cx_Oracle
Collecting cx_Oracle
  Downloading cx_Oracle-8.0.0-cp38-cp38-win32.whl (141 kB)
     |████████████████████████████████| 141 kB 1.3 MB/s
Installing collected packages: cx-Oracle
Successfully installed cx-Oracle-8.0.0
WARNING: You are using pip version 20.1.1; however, version 20.2.1 is available.
You should consider upgrading via the 'c:\users\nimesh\appdata\local\programs\python\python38-32\python.exe -m pip insta
ll --upgrade pip' command.

C:\Users\nimesh>
```



- The **cx_Oracle** module loads Oracle Client libraries which communicate over Oracle Net to an existing database.
- Oracle Net is not a separate product: it is how the Oracle Client and Oracle Database communicate.
- **Oracle Client libraries :** These can be from the free Oracle Instant Client, or those included in Oracle Database if Python is on the same machine as the database.
- Oracle client libraries versions 19, 18, 12, and 11.2 are supported on Linux, Windows and macOS.
- Use the latest client possible: Oracle's standard client-server version interoperability allows connection to both older and newer databases.

Steps to connect Oracle database in Python using Oracle Connector Python



- **Import database specific module**
  Ex. import cx_Oracle
- **connect():** Now Establish a connection between Python program and Oracle database by using connect() function.

```
con = cx_Oracle.connect('username/password@localhost')
```

- **cursor():** To execute sql query and to provide result some special object required is nothing but cursor() object

```
cursor = cx_Oracle.cursor()
```

- **execute method :**

> *cursor.execute(sqlquery) – – – -> to execute single query.*
> *cursor.execute(sqlqueries) – – – -> to execute a group of multiple sqlquery seperated by ";"*

- **commit():** For DML(Data Manuplate Language) query in this query you have (update, insert, delete) operation we need to commit() then only the result reflecte in database.
- **Fetch():** This retrieves the next row of a query result set and returns a single sequence, or None if no more rows are available.
- **close():** After all done mendentory to close all operation

```
cursor.close()
con.close()
```

```
# importing module
import cx_Oracle


# Create a table in Oracle database
try:

    con = cx_Oracle.connect("system/Cdac1234@localhost:1522/orcl1")

    # Now execute the sqlquery
    cursor = con.cursor()

    # Creating a table srollno heading which is number
    cursor.execute("create table student11(srollno number, name varchar2(10), efees number(10,2))")

    print("Table Created successful")

except cx_Oracle.DatabaseError as e:
    print("There is a problem with Oracle", e)

# by writing finally if any error occurs
# then also we can close the all database operation
finally:
    if cursor:
        cursor.close()
    if con:
        con.close()
```