6.3 Використання Python для роботи з MySQL / Using Python to work with MySQL

- Python може використовуватися для розробки застосувань для роботи з базами даних
- Однією з найпоширеніших СУБД є саме MySQL

- Python can be used in database applications
- One of the most popular databases is MySQL

- Для роботи з базою даних Python потребує драйвера MySQL
- Будемо використовувати драйвер “MySQL Connector”

- Python needs a MySQL driver to access the MySQL database
- We will use the driver "MySQL Connector"

6.3 Використання Python для роботи з MySQL / Using Python to work with MySQL

- Рекомендується використовувати PIP для встановлення “MySQL Connector”
- Скоріше за все PIP вже встановлений у середовищі Python
- Необхідно перейти у командному рядку до місця розташування PIP та ввести наступну команду:

python -m pip install mysql-connector

- It is recommended to use PIP 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:

python -m pip install mysql-connector

6.3 Використання Python для роботи з MySQL / Using Python to work with MySQL

- Для перевірки успішності інсталяції або в разі наявності встановленого “MySQL Connector”, необхідно створити файл з наступним вмістом
- 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
```

- Якщо даний код буде виконано без помилок, “MySQL Connector” є встановленим та готовим до використання
- If the above code was executed with no errors, "MySQL Connector" is installed and ready to be used

6.3 Використання Python для роботи з MySQL / Using Python to work with MySQL

- Робота з базою даних розпочинається з встановлення з'єднання
- Необхідно вказати ім'я користувача бази даних та пароль

- 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",
  passwd="yourpassword"
)

print(mydb)
```

6.3 Використання Python для роботи з MySQL / Using Python to work with MySQL

CREATE DATABASE

```
import mysql.connector

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

mycursor = mydb.cursor()

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

SHOW DATABASES

```
mycursor.execute("SHOW DATABASES")

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

Підключення до бази даних

Connect to the database

```
import mysql.connector

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

Якщо бази даних не існує, буде отримано помилку

If the database does not exist, you will get an error

6.3 Використання Python для роботи з MySQL / Using Python to work with MySQL

INSERT INTO

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="yourusername",
  passwd="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.")
```

Зверніть увагу на команду mydb.commit(). Вона обов'язкова для виконання операцій з даними, інакше жодних змін не буде зафіксовано.

Notice the statement: mydb.commit(). It is required to make the changes, otherwise no changes are made to the table.

6.3 Використання Python для роботи з MySQL / Using Python to work with MySQL

Вставка декількох записів / Insert multiple rows

```
import mysql.connector

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

mycursor = mydb.cursor()

sql = "INSERT INTO customers (name, address) VALUES (%s, %s)"
val = [
  ('Peter', 'Lowstreet 4'),
  ('Amy', 'Apple st 652'),
  ('Chuck', 'Main Road 989'),
  ('Viola', 'Sideway 1633')
]

mycursor.executemany(sql, val)

mydb.commit()

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

Метод executemany() використовується для виконання декількох запитів. Другим аргументом є список кортежів, які необхідно додати у таблицю.

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.

6.3 Використання Python для роботи з MySQL / Using Python to work with MySQL

- Звернувшись до курсору, можна отримати ID останнього створеного рядку
- You can get the id of the row you just inserted by asking the cursor object

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="yourusername",
  passwd="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)
```

6.3 Використання Python для роботи з MySQL / Using Python to work with MySQL

SELECT fetchall()

```
import mysql.connector

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

mycursor = mydb.cursor()

mycursor.execute("SELECT * FROM customers")

myresult = mycursor.fetchall()

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

fetchone()

```
import mysql.connector

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

mycursor = mydb.cursor()

mycursor.execute("SELECT * FROM customers")

myresult = mycursor.fetchone()

print(myresult)
```

Метод fetchone() поверне перший рядок результуючої вибірки
The fetchone() method will return the first row of the result

6.3 Використання Python для роботи з MySQL / Using Python to work with MySQL

Запобігання SQL-ін'єкціям / Prevent SQL injections (WHERE, DELETE)

```
import mysql.connector

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

mycursor = mydb.cursor()

sql = "SELECT * FROM customers WHERE address = %s"
adr = ("Yellow Garden 2", )

mycursor.execute(sql, adr)

myresult = mycursor.fetchall()

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

```
import mysql.connector

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

mycursor = mydb.cursor()

sql = "DELETE FROM customers WHERE address = %s"
adr = ("Yellow Garden 2", )

mycursor.execute(sql, adr)

mydb.commit()

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

6.3 Використання Python для роботи з MySQL / Using Python to work with MySQL

UPDATE

```
import mysql.connector

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

mycursor = mydb.cursor()

sql = "UPDATE customers SET address = %s WHERE address = %s"
val = ("Valley 345", "Canyon 123")

mycursor.execute(sql, val)

mydb.commit()

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

Хорошою практикою вважається екранування значень у будь-яких запитах.
Це дозволяє запобігти SQL-ін'єкціям, які є поширеною технікою хакінгу з метою зламу або виведення зі строю бази даних.
Модуль mysql.connector використовує символ підстановки %s для екранування значень у запитах.

It is considered a good practice to escape the values of any query, also in update statements.

This is to prevent SQL injections, which is a common web hacking technique to destroy or misuse your database.
The mysql.connector module uses the placeholder %s to escape values in the statement.