Питання для закріплення

1. Як відбувається форматування даних про час та дату у Python?
2. Читання та запис файлів в Python?
3. Як відбувається взаємодія із зовнішніми додатками у Python?
4. Для чого використовується Jupyter Notebook?
5. Для чого використовується модуль *subprocess* у Python?

Список рекомендованої літератури

1. datetime – Basic date and time // Електронний ресурс. Режим доступу: https://docs.python.org/3/library/datetime.html
2. Python – Object Oriented // Електронний ресурс. Режим доступу: https://www.tutorialspoint.com/python/python_classes_objects.htm
3. IoT Fundamentals: Big Data & Analytics // Електронний ресурс. Режим доступу: https://www.netacad.com/courses/iot/big-data-analytics
4. Jupyter Notebook // Електронний ресурс. Режим доступу: https://jupyter.org/

Лекція 4.
Програмування Python та SQLite.
Призначення утиліти csvsql

План лекції

4.1. Основні операції SQL.
4.2. Робота Python з SQLite.
4.3. Призначення утиліти csvsql та метод execute().

4.1. Основні операції SQL

Існує багато способів роботи із зовнішніми файлами в Python. SQLite – це реалізація SQL, яка добре працює з Python. Замість того, щоб використовувати метод роботи з клієнтським сервером, використовуються з'єднання, встановлені між Python та базою даних SQL, створюючи об’єкт підключення SQL. Цей об’єкт матиме пов'язані з ним методи. Після створення об’єкта для з'єднання використовується метод створення об’єкта курсору. Об'єкт курсору має методи SQLite, доступні для виконання операцій SQL в базі даних.

SQL – мова для взаємодії з базами даних та таблицями. Існує ряд діалектів SQL, однак деякі основні операції є стандартними, вони працюють аналогічно, як у SQLite, MySQL чи інших реалізаціях SQL. SQLite може працювати в

інтерактивному режимі з командного рядка. Мова програмування Python, може взаємодіяти з SQLite через імпорт модулів.

SQL є мовою, що складається з трьох мов спеціального призначення. Перша – мова визначення даних (data definition language). Вона використовується для створення та маніпулювання структурою баз даних та таблиць SQL (табл. 4.1).

Табл. 4.1. Деякі загальні команди data definition language SQL

Команда	Пояснення
ALTER TABLE	Змінює структуру існуючої таблиці
CREATE DATABASE	Створює нову порожню базу даних
CREATE TABLE	Створює таблицю в рамках існуючої бази даних
DESCRIBE	Відображає структуру таблиці
DROP DATABASE	Повністю видаляє цілу базу даних
DROP TABLE	Видаляє таблицю з бази даних
USE	Відкриває базу даних, з якою слід працювати

Друга – мова маніпулювання даними (data manipulation language). Вона використовується для додавання, видалення або перетворення даних, які є в таблицях даних (табл. 4.2).

Табл. 4.2. Деякі загальні команди data manipulation language SQL

Команда	Пояснення
DELETE	Видаляє наявні дані
INSERT	Додає нові дані
REPLACE	Замінює записи, які мають дублікати даних, на записи, які потрібно вставити
UPDATE	Замінює значення у стовпцях даних новими значеннями залежно від зазначеного критерію

Третя є мовою запитів даних (data query language), яка використовується для доступу до даних у таблицях даних з метою генерування інформації (наприклад, команда SELECT отримує доступ до даних на основі заданого набору критеріїв).

4.2. Робота Python з SQLite

Розглянемо послідовність команд, які ілюструють основи операцій SQLite. По-перше, в операційну систему потрібно встановити зовнішній інструмент під назвою csvkit, щоб файл csv можна було імпортувати в базу даних SQLite. Нижченаведені команди відображають етапи створення бази даних SQLite, імпорту даних CSV в базу даних, виконання запиту в таблиці даних та перегляду результатів запиту.

```
import sqlite3 as sq1
```

Створюємо нову базу даних та встановлюємо з'єднання з нею:

```
conn = sq1.connect('logins.db')
```

```
!csvsql --db sqlite://///logins.db --insert logins.csv
```

Створюємо об'єкт cursor для запитів SQL:

```
cur = conn.cursor()
```

Створюємо запит SQL як рядковий об'єкт:

```
query = 'SELECT * FROM logins LIMIT 5'
```

Виконуємо запит SQL, об'єкт cursor тепер містить результат запиту:

```
cur.execute(query)
```

Створюємо цикл, який перебирає кількість рядків в об'єкті cursor та друкує їх вміст:

```
for row in cur:
    print(row)
```

```
(u'John', u'2016-12-29', u'1970-01-01 14:24:13.000000')
(u'Allan', u'2016-12-29', u'1970-01-01 03:16:54.000000')
(u'Robert', u'2016-12-30', u'1970-01-01 04:54:25.000000')
(u'Eve', u'2016-12-30', u'1970-01-01 08:32:14.000000')
(u'Leslie', u'2016-12-30', u'1970-01-01 20:34:54.000000')
```

Команда "!csvsql --db ..." може бути виконана як перша команда. Це зовнішній інструмент, який потрібно встановити в ОС. Для виконання цього командного рядка можна використовувати командний рядок (Linux CLI), але для спрощення речей цю зовнішню команду можна виконати безпосередньо з ноутбука шляхом префіксації команди "!" [1].

4.3. Призначення утиліти csvsql

Sqlcsv – простий інструмент командного рядка, який можна використовувати для вибірки дані з бази даних та експорту результат як CSV та вставки даних в базу даних із CSV. Працює лише з Python 3 [2]. У наведених нижче прикладах використовується наступна схема таблиці з MySQL:

```
CREATE TABLE testtable(
  id INT AUTO_INCREMENT PRIMARY KEY,
  int_col INT,
  float_col FLOAT,
  varchar_col VARCHAR(255) )
```

Розглянемо, як отримати доступ до бази даних із мови програмування Python. В інших мовах використовується майже однакова модель: імена бібліотек та функцій можуть відрізнятися, але концепції однакові. Ось коротка програма Python, яка вибирає широти та довготи з бази даних SQLite, що зберігається у файлі під назвою survey.db:

```
1.    import sqlite3
2.    connection = sqlite3.connect("survey.db")
3.    cursor = connection.cursor()
4.    cursor.execute("SELECT Site.lat, Site.long FROM Site;")
5.    results = cursor.fetchall()
6.    for r in results:
7.    print(r)
8.    cursor.close()
9.    connection.close()
```

Результат виконання програми:

```
(-49.85, -128.57)
(-47.15, -126.72)
(-48.87, -123.4)
```

Програма починається з імпорту sqlite3 бібліотеки. Якби ми підключалися до MySQL або іншої бази даних, ми імпортували б іншу бібліотеку, але всі вони надають однакові функції, так що решта нашої програми не повинна змінюватись (принаймні, не сильно), якщо ми перемикаємося з однієї бази даних на іншу. Рядок 2 встановлює підключення до бази даних. Оскільки ми використовуємо SQLite, все, що нам потрібно вказати, це ім'я файлу бази даних. Інші системи можуть вимагати від нас надання імені користувача та пароля. Потім рядок 3 використовує це з’єднання для створення об’єкту cursor. Подібно курсору в редакторі, його роль полягає у відстеженні того, де ми знаходимось у базі даних. У рядку 4 ми використовуємо цей курсор, щоб попросити базу даних виконати для нас запит. Запит пишеться в SQL і передається cursor.execute як рядок. Нам потрібно переконатися, що SQL правильно відформатований; якщо це не так, або якщо щось виконується не так, коли вона виконується, база даних повідомляє про помилку. База даних повертає результати запиту у відповідь на cursor.fetchall на рядок 5. Цей результат – це список із одним записом для кожного запису в наборі результатів; якщо ми прокрутимо цей список (рядок 6) і надрукуємо ці записи списку (рядок 7), ми побачимо, що кожен із них – кортеж з одним елементом для кожного поля, про яке ми просили. Нарешті, рядки 8 і 9 закривають наш курсор і наше з’єднання, оскільки база даних може одночасно тримати відкритою лише обмежену їх кількість.

Оскільки встановлення з'єднання вимагає часу, однак, ми не повинні відкривати з'єднання, виконувати одну операцію, потім закривати з'єднання, лише щоб знову відкрити його через кілька мікросекунд, щоб виконати іншу операцію. Натомість краще створити одне з’єднання, яке залишатиметься відкритим протягом усього терміну дії програми [3].

Висновок до лекції 4

SQLite – це реалізація SQL, яка добре працює з Python. Замість того, щоб використовувати метод роботи з клієнтським сервером, використовуються з'єднання, встановлені між Python та базою даних SQL, створюючи об’єкт підключення SQL. Після створення об’єкта з'єднання використовується метод створення об’єкта курсору. Об'єкт курсору має методи SQLite, доступні для виконання операцій SQL в базі даних. SQLite може працювати в інтерактивному режимі з командного рядка, мова Python також може взаємодіяти з SQLite через імпорт модулів.

Питання для закріплення

1. З яких трьох мов спеціального призначення складається SQL?
2. Наведіть основні команди мови SQL.
3. Яке призначення утиліти csvsql?
4. Як отримати доступ до баз даних із програм, написаних на Python?

Список рекомендованої літератури

1. IoT Fundamentals: Big Data & Analytics // Електронний ресурс. Режим доступу: https://www.netacad.com/courses/iot/big-data-analytics
2. sqlcsv // Електронний ресурс. Режим доступу: https://pypi.org/project/sqlcsv/
3. Programming with Databases – Python // Електронний ресурс. Режим доступу: https://swcarpentry.github.io/sql-novice-survey/10-prog/index.html

Лекція 5.
Процедура імпорту даних із файлів у Pandas.
Імпорт даних з мережі Інтернет.
Засоби для кореляційного аналізу в Pandas

План лекції

5.1. Статистичні підходи до аналітики великих даних.
5.2. Використання Pandas.
5.3. Імпорт даних з файлів.
5.4. Імпорт даних з мережі Інтернет.
5.5. Описова статистика в Pandas.
5.6. Засоби для кореляційного аналізу в Pandas.