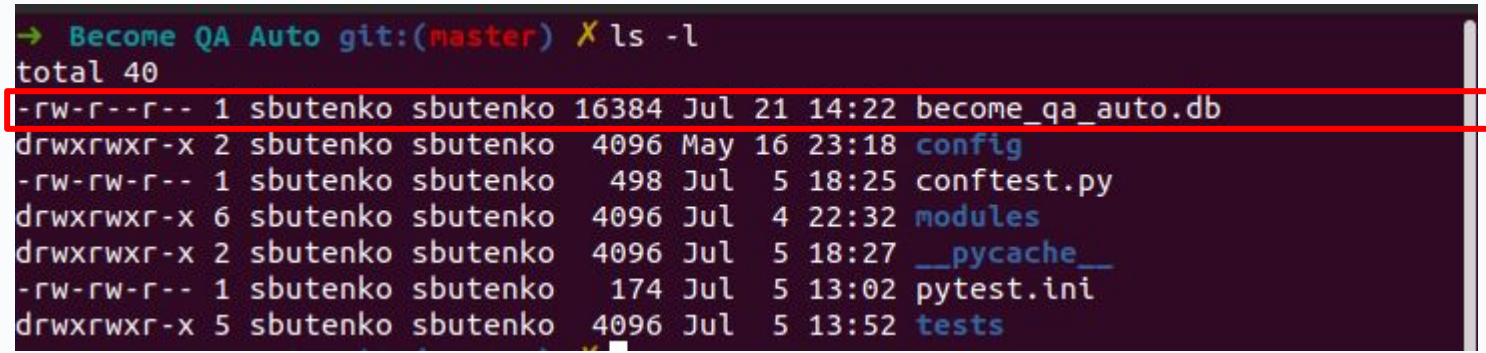
Python. Як приєднатися до бази даних

Бутенко Сергій

План лекції

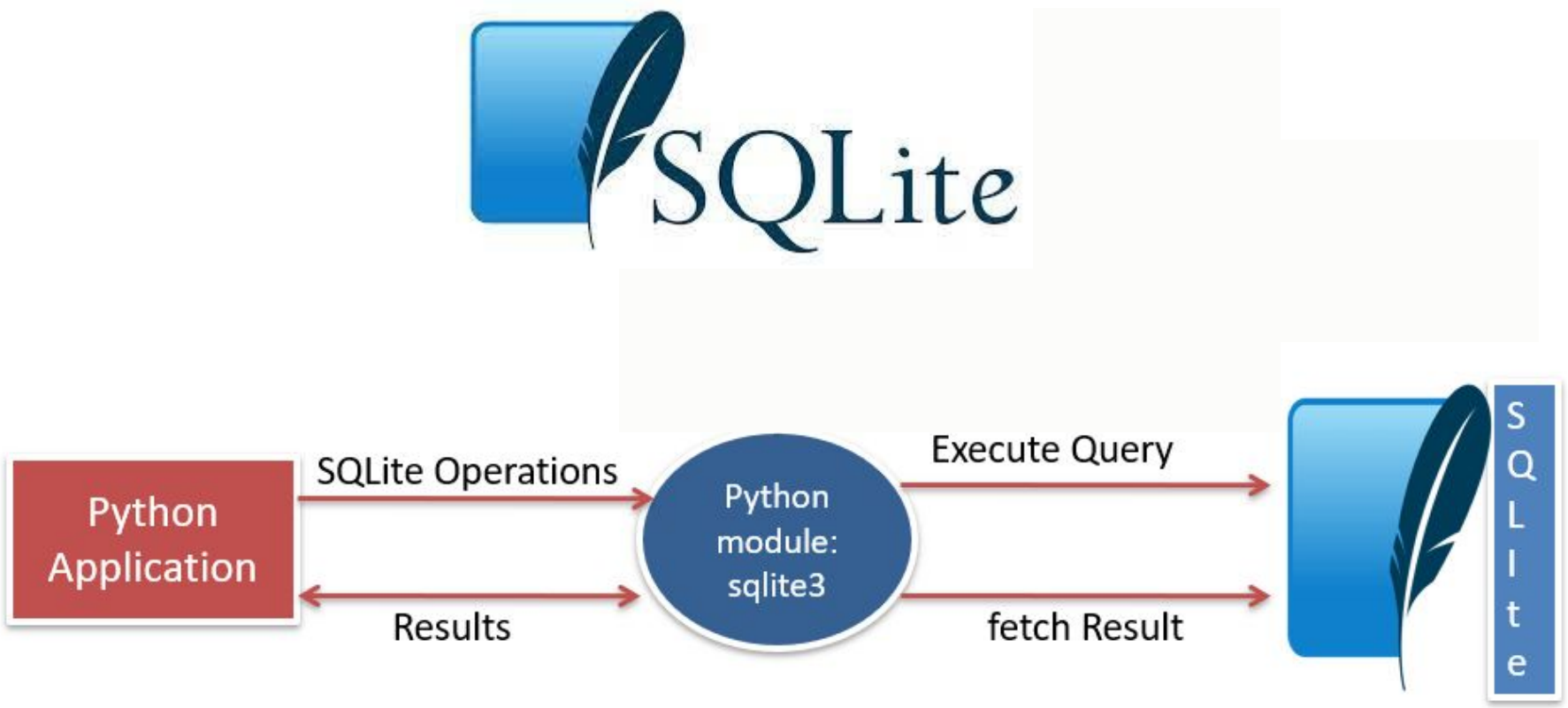
- Як спілкуватися з базою даних?
- Що таке SQL?

◎ Практика


→ Become QA Auto git:(master) ✗ ls -l
total 40
-rw-r--r-- 1 sbutenko sbutenko 16384 Jul 21 14:22 become_qa_auto.db
drwxrwxr-x 2 sbutenko sbutenko 4096 May 16 23:18 config
-rw-rw-r-- 1 sbutenko sbutenko 498 Jul 5 18:25 conftest.py
drwxrwxr-x 6 sbutenko sbutenko 4096 Jul 4 22:32 modules
drwxrwxr-x 2 sbutenko sbutenko 4096 Jul 5 18:27 __pycache__
-rw-rw-r-- 1 sbutenko sbutenko 174 Jul 5 13:02 pytest.ini
drwxrwxr-x 5 sbutenko sbutenko 4096 Jul 5 13:52 tests

Види баз даних

◎ Практика


SQLite
Python Application
SQLite Operations
Results
Python module: sqlite3
Execute Query
fetch Result
SQLite

◎ Практика

Table of Contents

`sqlite3` — DB-API 2.0 interface for SQLite databases

Source code: Lib/sqlite3/

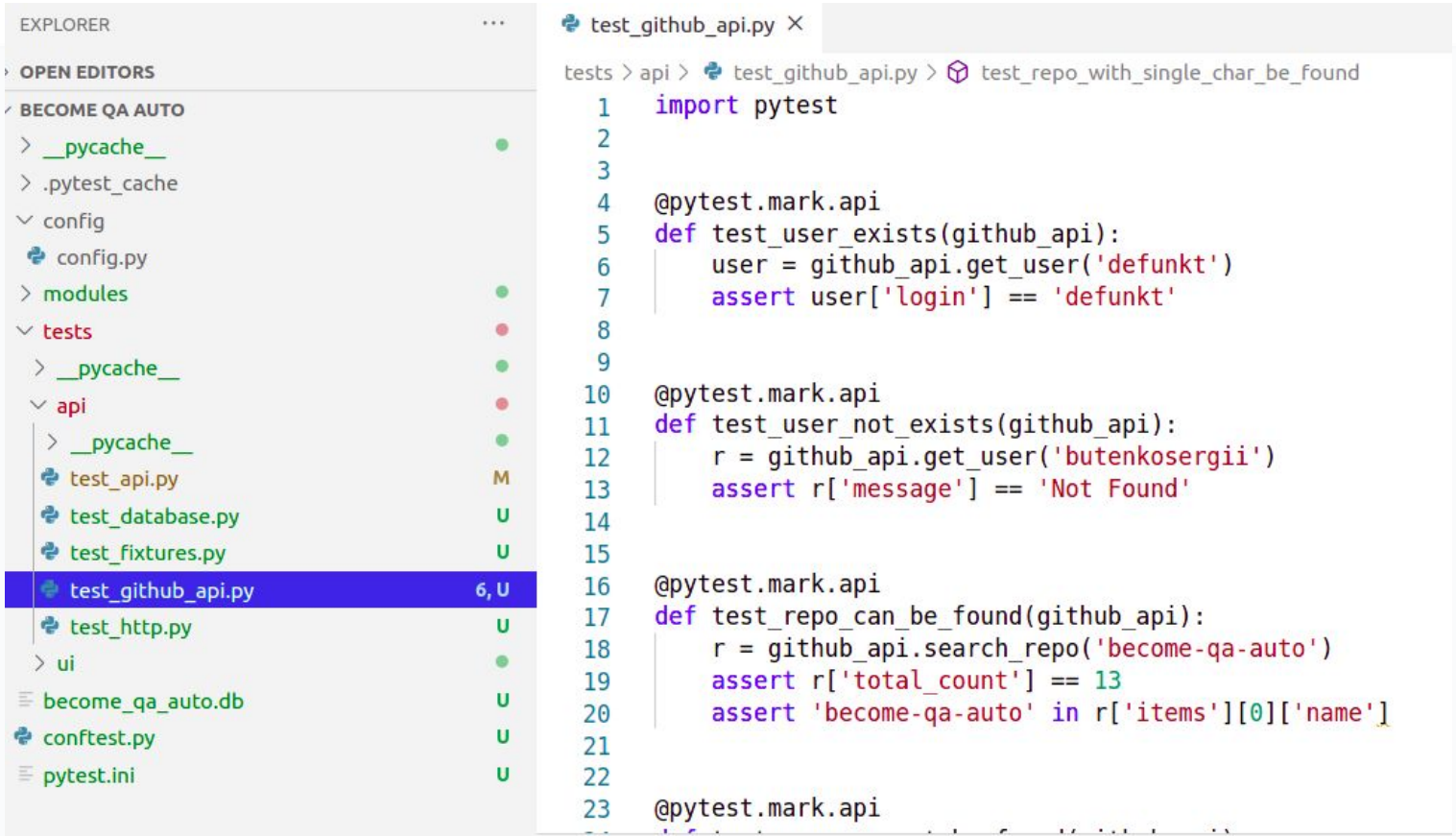
SQLite is a C library that provides a lightweight disk-based database that doesn't require a separate server process and allows accessing the database using a nonstandard variant of the SQL query language. Some applications can use SQLite for internal data storage. It's also possible to prototype an application using SQLite and then port the code to a larger database such as PostgreSQL or Oracle.

The sqlite3 module was written by Gerhard Häring. It provides an SQL interface compliant with the DB-API 2.0 specification described by **PEP 249**, and requires SQLite 3.7.15 or newer.

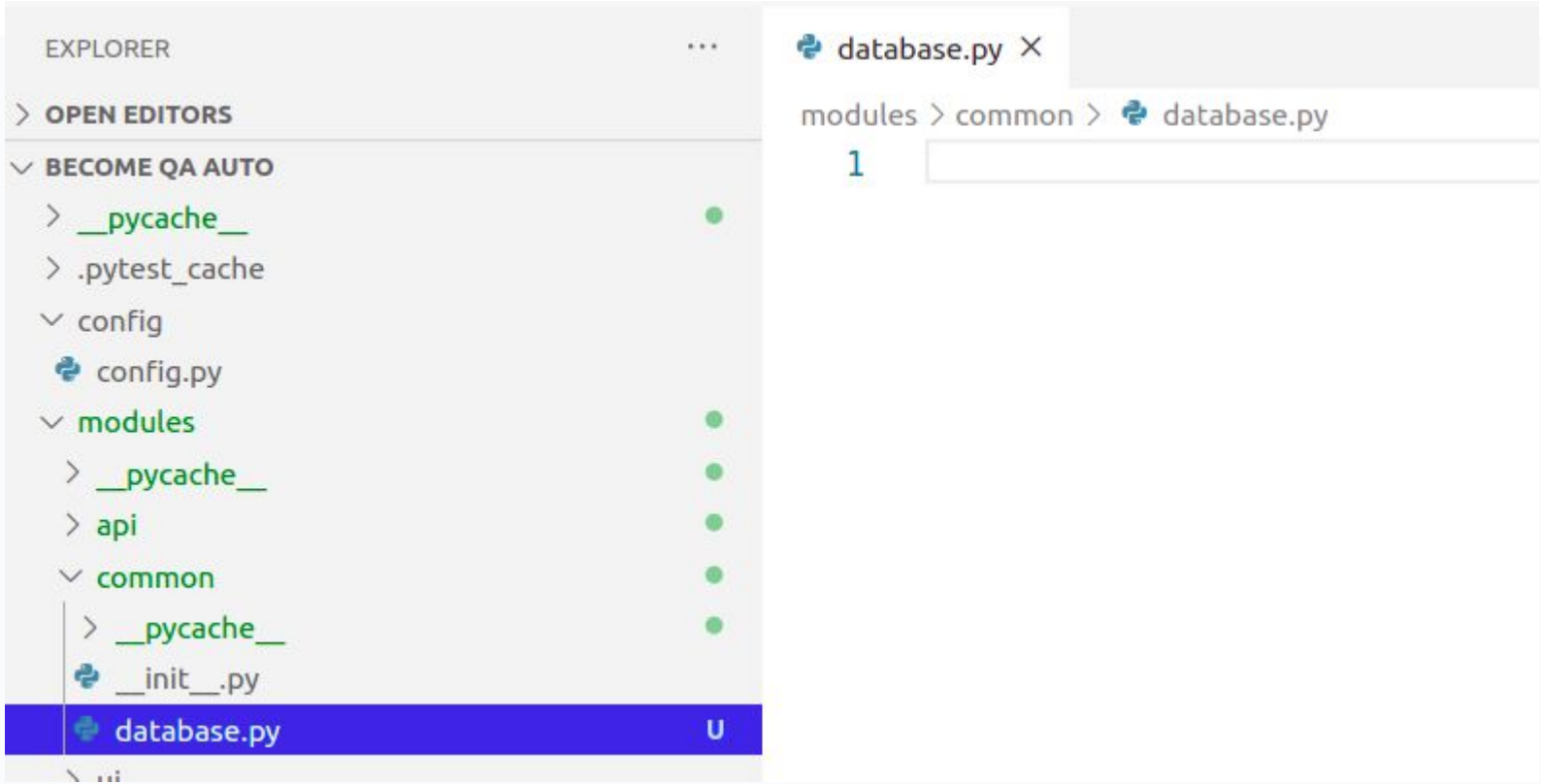
To use the module, start by creating a `Connection` object that represents the database. Here the data will be stored in the `example.db` file:

```
import sqlite3
con = sqlite3.connect('example.db')
```

Практика


EXPLORER
OPEN EDITORS
BECOME QA AUTO
__pycache__
.pytest_cache
config
config.py
modules
tests
__pycache__
api
__pycache__
test_api.py M
test_database.py U
test_fixtures.py U
test_github_api.py 6, U
test_http.py U
ui
become_qa_auto.db U
conftest.py U
pytest.ini U
test_github_api.py
tests > api > test_github_api.py > test_repo_with_single_char_be_found
1 import pytest
2
3
4 @pytest.mark.api
5 def test_user_exists(github_api):
6 user = github_api.get_user('defunkt')
7 assert user['login'] == 'defunkt'
8
9
10 @pytest.mark.api
11 def test_user_not_exists(github_api):
12 r = github_api.get_user('butenkosergii')
13 assert r['message'] == 'Not Found'
14
15
16 @pytest.mark.api
17 def test_repo_can_be_found(github_api):
18 r = github_api.search_repo('become-qa-auto')
19 assert r['total_count'] == 13
20 assert 'become-qa-auto' in r['items'][0]['name']
21
22
23 @pytest.mark.api

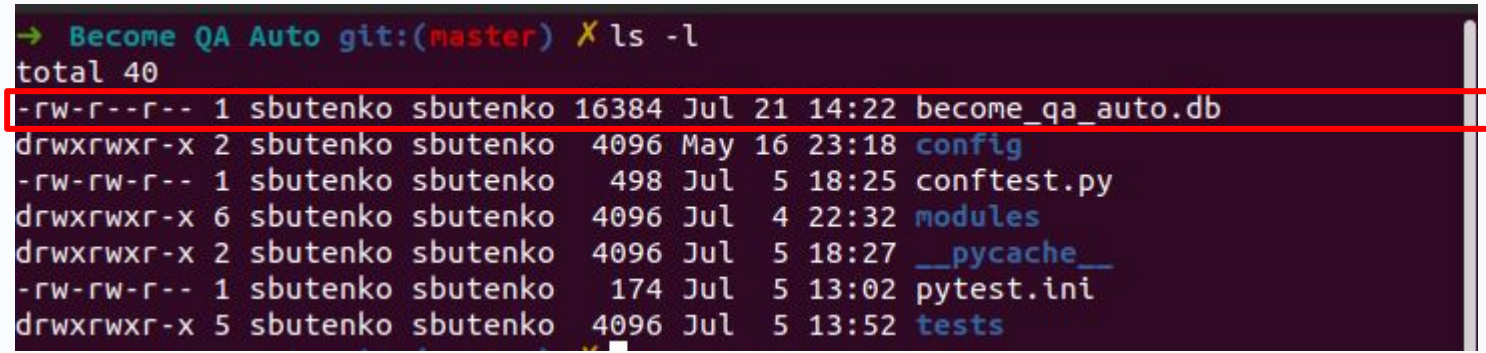
◎ Практика


EXPLORER
OPEN EDITORS
BECOME QA AUTO
__pycache__
.pytest_cache
config
config.py
modules
__pycache__
api
common
__pycache__
__init__.py
database.py
U
database.py
modules > common > database.py
1

◎ Практика

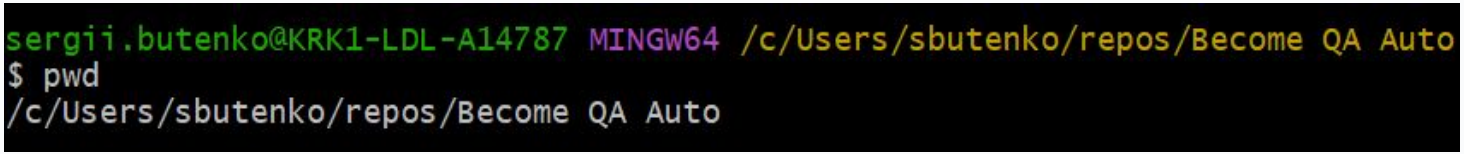
```
1   import sqlite3
2
3
4   class Database():
5
6       def __init__(self):
7           self.connection = sqlite3.connect('/home/sbutenko/repos/LnD/Become QA Auto' + '/become_qa_auto.db')
8           self.cursor = self.connection.cursor()
9
10      def test_connection(self):
11          sqlite_select_Query = "SELECT sqlite_version();"
12          self.cursor.execute(sqlite_select_Query)
13          record = self.cursor.fetchall()
14          print(f"Connected successfully. SQLite Database Version is: {record}")
15
```

◎ Практика


→ Become QA Auto git:(master) ✗ ls -l
total 40
-rw-r--r-- 1 sbutenko sbutenko 16384 Jul 21 14:22 become_qa_auto.db
drwxrwxr-x 2 sbutenko sbutenko 4096 May 16 23:18 config
-rw-rw-r-- 1 sbutenko sbutenko 498 Jul 5 18:25 conftest.py
drwxrwxr-x 6 sbutenko sbutenko 4096 Jul 4 22:32 modules
drwxrwxr-x 2 sbutenko sbutenko 4096 Jul 5 18:27 __pycache__
-rw-rw-r-- 1 sbutenko sbutenko 174 Jul 5 13:02 pytest.ini
drwxrwxr-x 5 sbutenko sbutenko 4096 Jul 5 13:52 tests

◎ Практика


→ Become QA Auto git:(master) ✗ pwd
/home/sbutenko/repos/LnD/Become QA Auto
→ Become QA Auto git:(master) ✗


sergii.butenko@KRK1-LDL-A14787 MINGW64 /c/Users/sbutenko/repos/Become QA Auto
$ pwd
/c/Users/sbutenko/repos/Become QA Auto

◎ Практика

```
1  import sqlite3
2
3
4  class Database():
5
6      def __init__(self):
7          self.connection = sqlite3.connect(r'/home/sbutenko/repos/LnD/Become QA Auto' + r'/become_qa_auto.db')
8          self.cursor = self.connection.cursor()
9
10     def test_connection(self):
11         sqlite_select_Query = "SELECT sqlite_version();"
12         self.cursor.execute(sqlite_select_Query)
13         record = self.cursor.fetchall()
14         print(f"Connected successfully. SQLite Database Version is: {record}")
15
```

◎ Практика

```
1  import sqlite3
2
3
4  class Database():
5
6      def __init__(self):
7          self.connection = sqlite3.connect(r'/home/sbutenko/repos/LnD/Become QA Auto' + r'/become_qa_auto.db')
8          self.cursor = self.connection.cursor()
9
10     def test_connection(self):
11         sqlite_select_Query = "SELECT sqlite_version();"
12         self.cursor.execute(sqlite_select_Query)
13         record = self.cursor.fetchall()
14         print(f"Connected successfully. SQLite Database Version is: {record}")
15
```

◎ Практика

```python
import sqlite3


class Database():

    def __init__(self):
        self.connection = sqlite3.connect(r'C:\\Users\\sbutenko\\repos\\Become QA Auto' + r'\\become_qa_auto.db')
        self.cursor = self.connection.cursor()

    def test_connection(self):
        sqlite_select_Query = "SELECT sqlite_version();"
        self.cursor.execute(sqlite_select_Query)
        record = self.cursor.fetchall()
        print(f"Connected successfully. SQLite Database Version is: {record}")

```

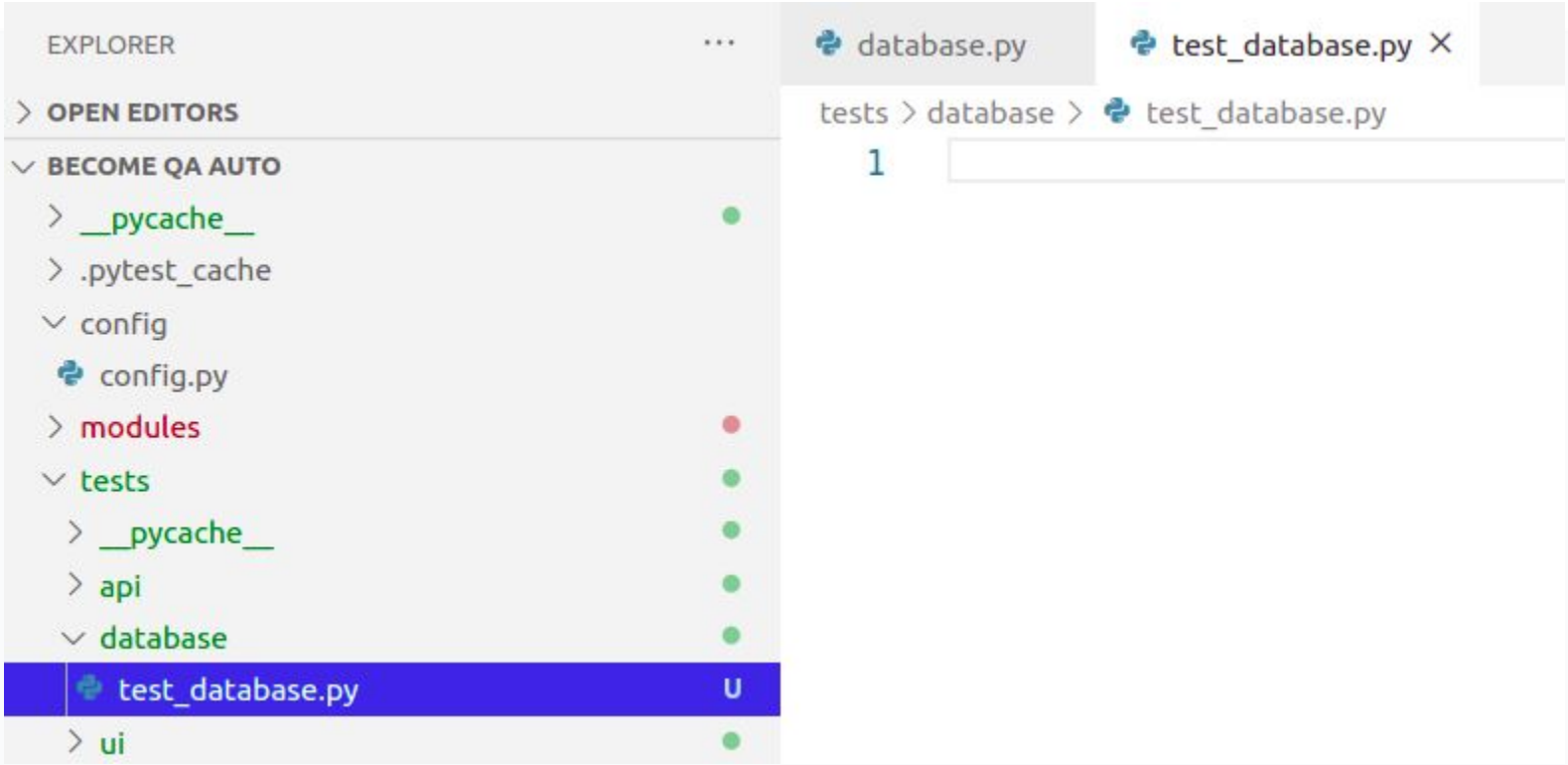
◎ Практика

```python
import sqlite3


class Database():

    def __init__(self):
        self.connection = sqlite3.connect(r'C:\Users\sbutenko\repos\Become QA Auto' + r'\become_qa_auto.db')
        self.cursor = self.connection.cursor()

    def test_connection(self):
        sqlite_select_Query = "SELECT sqlite_version();"
        self.cursor.execute(sqlite_select_Query)
        record = self.cursor.fetchall()
        print(f"Connected successfully. SQLite Database Version is: {record}")

```

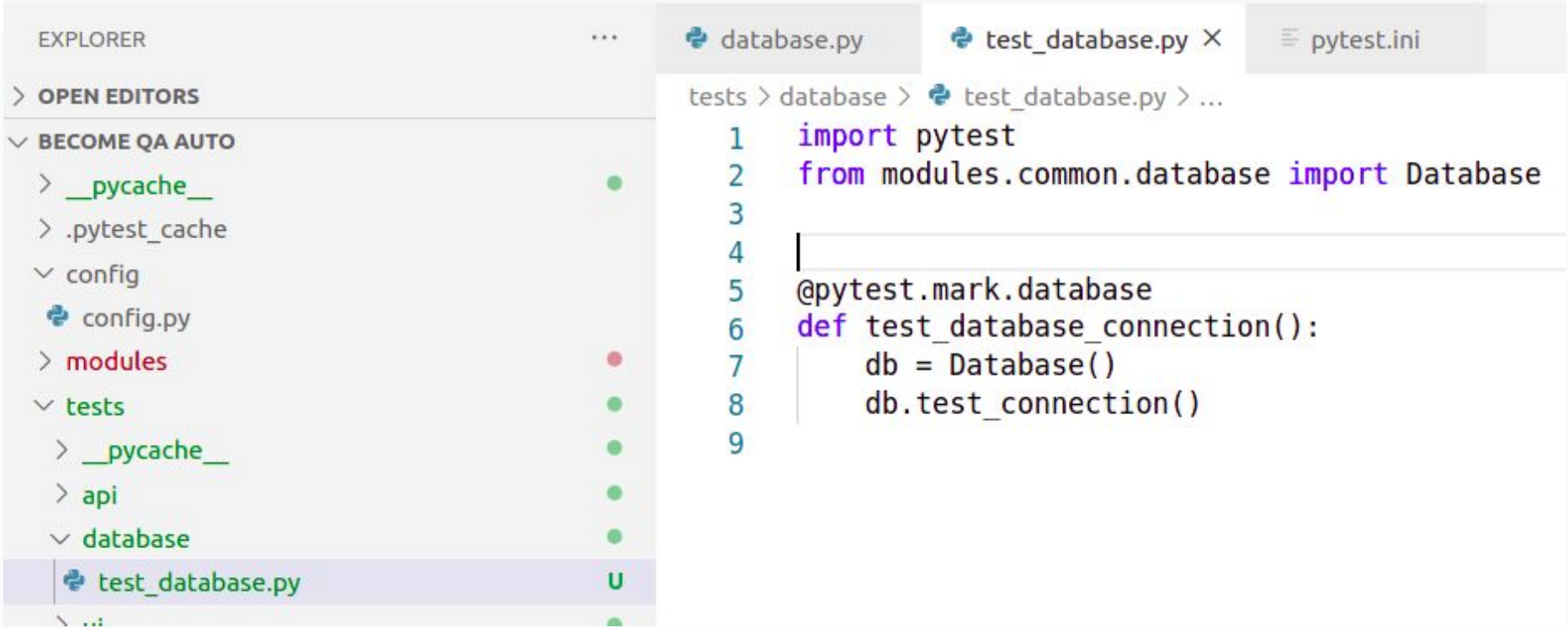
◎ Практика

```
1  import sqlite3
2
3
4  class Database():
5
6      def __init__(self):
7          self.connection = sqlite3.connect(r'/home/sbutenko/repos/LnD/Become QA Auto' + r'/become_qa_auto.db')
8          self.cursor = self.connection.cursor()
9
10     def test_connection(self):
11         sqlite_select_Query = "SELECT sqlite_version();"
12         self.cursor.execute(sqlite_select_Query)
13         record = self.cursor.fetchall()
14         print(f"Connected successfully. SQLite Database Version is: {record}")
15
```

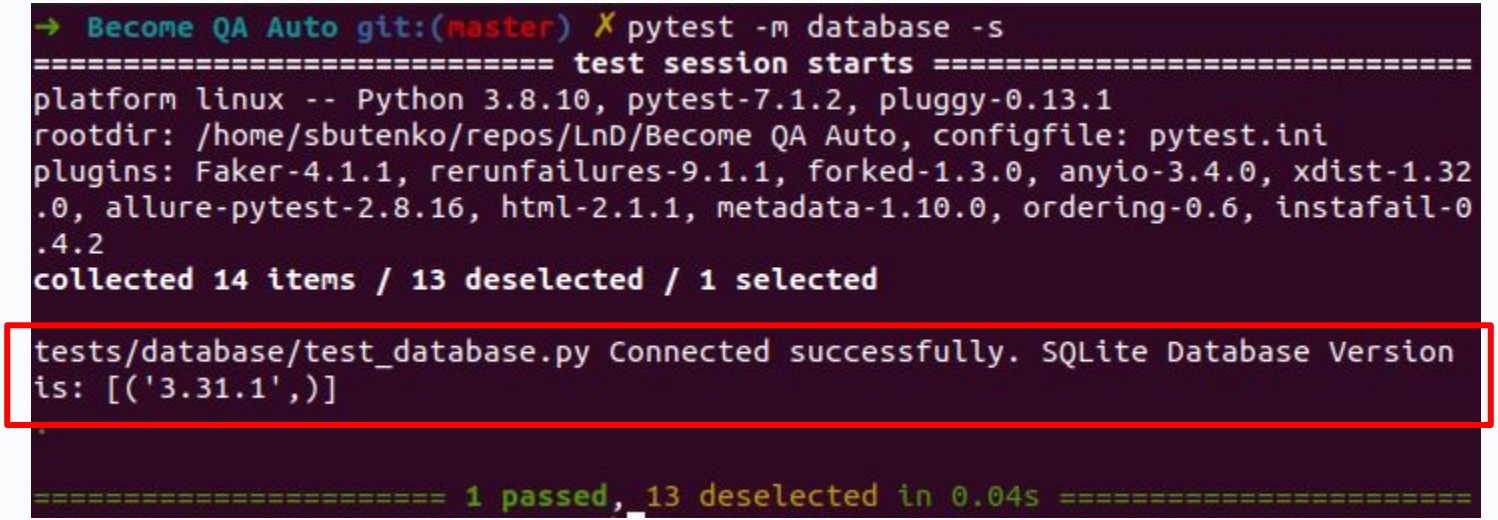
Практика


EXPLORER
database.py
test_database.py
OPEN EDITORS
tests > database > test_database.py
BECOME QA AUTO
1
__pycache__
.pytest_cache
config
config.py
modules
tests
__pycache__
api
database
test_database.py
U
ui

◎ Практика


EXPLORER
OPEN EDITORS
BECOME QA AUTO
__pycache__
.pytest_cache
config
config.py
modules
tests
__pycache__
api
database
test_database.py
U
database.py
test_database.py
pytest.ini
tests > database > test_database.py > ...
1 import pytest
2 from modules.common.database import Database
3
4
5 @pytest.mark.database
6 def test_database_connection():
7 db = Database()
8 db.test_connection()
9

Практика


→ Become QA Auto git:(master) ✗ pytest -m database -s
============================= test session starts ==============================
platform linux -- Python 3.8.10, pytest-7.1.2, pluggy-0.13.1
rootdir: /home/sbutenko/repos/LnD/Become QA Auto, configfile: pytest.ini
plugins: Faker-4.1.1, rerunfailures-9.1.1, forked-1.3.0, anyio-3.4.0, xdist-1.32
.0, allure-pytest-2.8.16, html-2.1.1, metadata-1.10.0, ordering-0.6, instafail-0
.4.2
collected 14 items / 13 deselected / 1 selected
tests/database/test_database.py Connected successfully. SQLite Database Version
is: [('3.31.1',)]
.
======================= 1 passed, 13 deselected in 0.04s ======================

SQL

SQL — це декларативна мова програмування для управління базами даних та для обробки даних за допомогою запитів.

SQL

SQL запит — звернення до бази даних, для створення чи модифікації структури бази даних та дій над даними (додавання, перегляд, зміна видалення)

SQL

```
SELECT name, address, city FROM customers;
```

```
DELETE FROM products WHERE id = 2
```

Підсумки

- Начилися використовувати модуль **`sqlite3`**
- Створили клас для підключення до бази даних
- Написали перший тест для перевірки підключення до бази даних
- Вивчили, що **SQL** — це декларативна мова програмування для управління базами даних та для обробки даних за допомогою запитів

Підсумки

- **SQL запит** — звернення до бази даних, для створення чи модифікації структури бази даних та дій над даними (додавання, перегляд, зміна, видалення)