

Лабораторная работа 2.21

Тема: Взаимодействие с базами данных SQLite3 с помощью языка программирования Python.

Цель работы: приобретение навыков по взаимодействию с базами данных SQLite3 с помощью языка программирования Python версии 3.x.

Порядок выполнения работы


1. Создаём аккаунт в GitHub. Затем создаём новый общедоступный репозиторий, в котором будет использована лицензия MIT и язык программирования Python.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (*).

Owner *

 RomanGorchakov ▾

Repository name *

/ Test

✔ Test is available.

Great repository names are short and memorable. Need inspiration? How about [automatic-octo-chainsaw](#) ?

Description (optional)

☒  **Public**

Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**

You choose who can see and commit to this repository.

Initialize this repository with:

☐ Add a README file

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: Python ▾

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)


Choose a license



License: MIT License ▾

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

③ You are creating a public repository in your personal account.


Create repository




Test
Public

 Pin
  Unwatch
 1


main
1 branch
0 tags

Go to file
 Add file
 <> Code



















RomanGorchakov Initial commit
87a4818 now
1 commit

	.gitignore	Initial commit	now
	LICENSE	Initial commit	now

Help people interested in this repository understand your project by adding a README.
 Add a README


© 2023 GitHub, Inc.
Terms
Privacy
Security
Status
Docs
Contact GitHub
Pricing
API
Training

2. Теперь необходимо дополнить файл `.gitignore` с необходимыми правилами для языка программирования Python. Для этого переходим по ссылке «<https://github.com/github/gitignore>» и скачиваем оттуда файл «Python.gitignore».

	Phalcon.gitignore	Remove trailing asterisks in Phalcon rules	10 years ago
	PlayFramework.gitignore	Added /project/project to PlayFramework.gitignore	7 years ago
	Plone.gitignore	Covered by global vim template	10 years ago
	Prestashop.gitignore	Update for Prestashop 1.7 (#3261)	3 years ago
	Processing.gitignore	Ignore transpiled .java and .class files (#3016)	4 years ago
	PureScript.gitignore	Update PureScript adding .spago (#3278)	3 years ago
	Python.gitignore	Update Python.gitignore	last year
	Qooxdoo.gitignore	Add gitignore for qooxdoo apps	13 years ago
	Qt.gitignore	Remove trailing whitespace	2 years ago
	R.gitignore	Merge pull request #3792 from jl5000/patch-1	2 years ago
	README.md	Merge pull request #3854 from AnilSeervi/patch-1	2 years ago
	ROS.gitignore	Added ignore for files created by <code>catkin_make_isolated</code>	6 years ago
	Racket.gitignore	Update Racket.gitignore	2 years ago
	Rails.gitignore	Ignore Rails .env according recommendations	2 years ago
	Raku.gitignore	Changes the name of Perl 6 to Raku (#3312)	3 years ago
	RhodesRhomobile.gitignore	Add Rhodes mobile application framework gitignore	13 years ago
	Ruby.gitignore	Ruby: ignore RuboCop remote inherited config files (#3197)	4 years ago

```

1  # Byte-compiled / optimized / DLL files
2  __pycache__/
3  *.py[cod]
4  *$py.class
5
6  # C extensions
7  *.so
8
9  # Distribution / packaging
10 .Python
11 build/
12 develop-eggs/
13 dist/
14 downloads/
15 eggs/
16 .eggs/
17 lib/
18 lib64/
19 parts/
20 sdist/
21 var/
22 wheels/
23 share/python-wheels/
24 *.egg-info/
25 .installed.cfg
26 *.egg
27 MANIFEST
28

```

3. Теперь создаём файл «README.md», где вносим информацию о своей группе и ФИО. Сохраняем набранный текст через кнопку «Commit changes».

RomanGorchakov / Py21

Search: Type to search

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Py21 / README.md in main

Cancel changes Commit changes...

Edit Preview Code 55% faster with GitHub Copilot Spaces 2 Soft wrap

```

1  Горчаков Роман Владимирович. Вариант 4
2  # Лабораторная работа 2.21. Взаимодействие с базами данных SQLite3 с помощью языка программирования Python.
3
4  Сами по себе СУБД редко используются для работы с базами данных. В том смысле, что в реальных проектах связки БД + СУБД бывает недостаточно. Обычно с СУБД работают через какой-
5  либо язык программирования. Это позволяет более гибко принимать запросы, обрабатывать ответы перед передачей их куда-либо далее. Ведь у императивного, а не декларативного как
6  SQL, языка программирования средств для работы с данными больше, да и логика богаче.
7
8  Чтобы использовать SQLite3 в Python, прежде всего, вам нужно будет импортировать модуль sqlite3, а затем создать объект соединения, который соединит нас с базой данных и позволит
9  нам выполнять операторы SQL. Объект соединения создается с помощью функции connect(). Будет создан новый файл под названием "mydatabase.db", в котором будет храниться наша база
10  данных.
11
12  Для взаимодействия с базой данных SQLite3 в Python необходимо создать объект cursor. Вы можете создать его с помощью метода cursor(). Курсор SQLite3 - это метод объекта
13  соединения. Для выполнения инструкций SQLite3 сначала устанавливается соединение, а затем создается объект курсора с использованием объекта соединения следующим образом. При
14  создании соединения с SQLite3 автоматически создается файл базы данных, если он еще не существует. Этот файл базы данных создается на диске, мы также можем создать базу данных в
15  оперативной памяти с помощью функции :memory: with the connect. Такая база данных называется базой данных в памяти.
16
17  Сначала импортируется модуль sqlite3, а затем определяется функция с именем sql_connection. Внутри функции у нас есть блок try, где функция connect() возвращает объект
18  соединения после установления соединения. В случае возникновения ошибок при установке соединения с базой данных выполняется оператор блока except, в котором в данном случае

```

4. В окне «Codespace» выбираем опцию «Create codespace on main». Откроется терминал, куда мы введём команду «git clone», чтобы клонировать свой репозиторий. После этого организуем репозиторий в соответствие с моделью ветвления Git-flow. Для этого введём в терминал команды: «git checkout –b develop» для создания ветки разработки; «git branch feature_branch» для создания ветки функций; «git branch release/1.0.0» для создания ветки релиза; «git checkout main» и «git branch hotfix» для создания веток hotfix. Устанавливаем библиотеки isort, black и flake8 и создаём файлы .pre-commit-config.yaml и environment.yml.

```
Welcome to Codespaces! You are on our default image.
- It includes runtimes and tools for Python, Node.js, Docker, and more. See the full list here: https://aka.ms/ghcs-default-image
- Want to use a custom image instead? Learn more here: https://aka.ms/configure-codespace

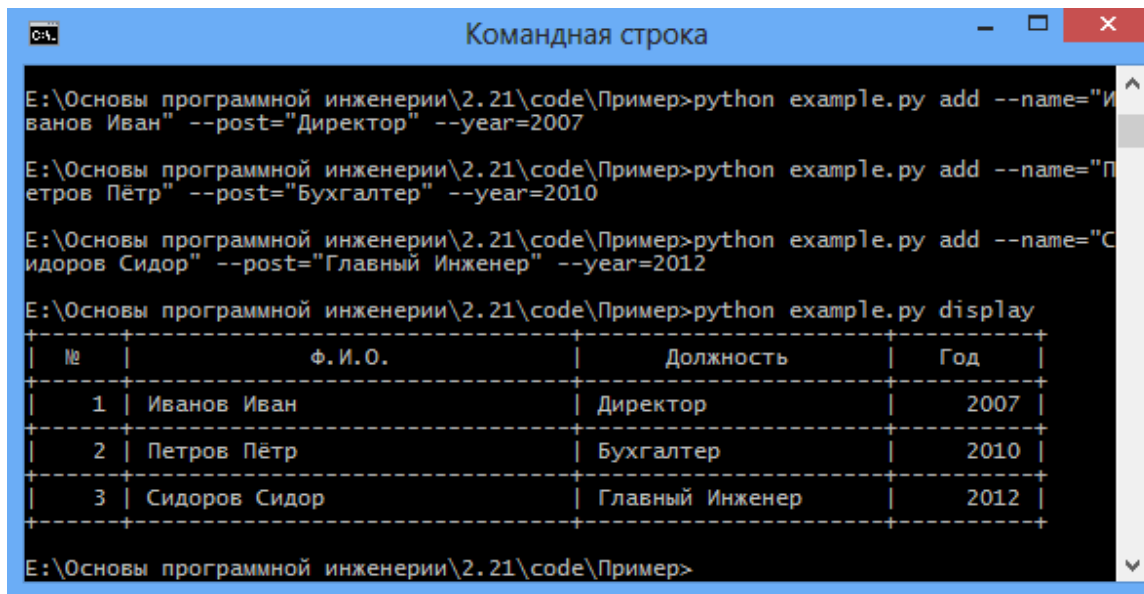
To explore VS Code to its fullest, search using the Command Palette (Cmd/Ctrl + Shift + P or F1).

Edit away, run your app as usual, and we'll automatically make it available for you to access.

@RomanGorchakov → /workspaces/Py21 (main) $ git checkout -b develop
Switched to a new branch 'develop'
@RomanGorchakov → /workspaces/Py21 (develop) $ git branch feature_branch
@RomanGorchakov → /workspaces/Py21 (develop) $ git branch release/1.0.0
@RomanGorchakov → /workspaces/Py21 (develop) $ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
@RomanGorchakov → /workspaces/Py21 (main) $ git branch hotfix
@RomanGorchakov → /workspaces/Py21 (main) $ git checkout develop
Switched to branch 'develop'
```

```
Successfully installed cfgv-3.4.0 distlib-0.3.8 identify-2.5.36 nodeenv-1.9.0 pre-commit-3.7.1 virtualenv-20.26.2
@RomanGorchakov → /workspaces/Py21 (develop) $ pip install isort
Collecting isort
  Downloading isort-5.13.2-py3-none-any.whl.metadata (12 kB)
    Downloading isort-5.13.2-py3-none-any.whl (92 kB)
      92.3/92.3 kB 2.2 MB/s eta 0:00:00
Installing collected packages: isort
Successfully installed isort-5.13.2
@RomanGorchakov → /workspaces/Py21 (develop) $ pip install black
Collecting black
  Downloading black-24.4.2-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (77 kB)
    Downloading black-24.4.2-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.8 MB)
      77.1/77.1 kB 1.9 MB/s eta 0:00:00
Collecting click>=8.0.0 (from black)
  Downloading click-8.1.7-py3-none-any.whl.metadata (3.0 kB)
Collecting mypy_extensions>=0.4.3 (from black)
  Downloading mypy_extensions-1.0.0-py3-none-any.whl.metadata (1.1 kB)
Requirement already satisfied: packaging>=22.0 in /home/codespace/.local/lib/python3.10/site-packages (from black) (24.0)
Collecting pathspec>=0.9.0 (from black)
  Downloading pathspec-0.12.1-py3-none-any.whl.metadata (21 kB)
Requirement already satisfied: platformdirs>=2 in /home/codespace/.local/lib/python3.10/site-packages (from black) (4.2.1)
Requirement already satisfied: tomli>=1.1.0 in /home/codespace/.local/lib/python3.10/site-packages (from black) (2.0.1)
Requirement already satisfied: typing_extensions>=4.0.1 in /home/codespace/.local/lib/python3.10/site-packages (from black) (4.11.0)
    Downloading black-24.4.2-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.8 MB)
      1.8/1.8 MB 28.5 MB/s eta 0:00:00
    Downloading click-8.1.7-py3-none-any.whl (97 kB)
      97.9/97.9 kB 2.5 MB/s eta 0:00:00
    Downloading mypy_extensions-1.0.0-py3-none-any.whl (4.7 kB)
    Downloading pathspec-0.12.1-py3-none-any.whl (31 kB)
Installing collected packages: pathspec, mypy_extensions, click, black
Successfully installed black-24.4.2 click-8.1.7 mypy_extensions-1.0.0 pathspec-0.12.1
@RomanGorchakov → /workspaces/Py21 (develop) $ pip install flake8
Collecting flake8
  Downloading flake8-7.0.0-py2.py3-none-any.whl.metadata (3.8 kB)
Collecting mccabe<0.8.0,>=0.7.0 (from flake8)
  Downloading mccabe-0.7.0-py2.py3-none-any.whl.metadata (5.0 kB)
Collecting pycodestyle<2.12.0,>=2.11.0 (from flake8)
  Downloading pycodestyle-2.11.1-py2.py3-none-any.whl.metadata (4.5 kB)
Collecting pyflakes<3.3.0,>=3.2.0 (from flake8)
  Downloading pyflakes-3.2.0-py2.py3-none-any.whl.metadata (3.5 kB)
    Downloading flake8-7.0.0-py2.py3-none-any.whl (57 kB)
      57.6/57.6 kB 1.3 MB/s eta 0:00:00
    Downloading mccabe-0.7.0-py2.py3-none-any.whl (7.3 kB)
    Downloading pycodestyle-2.11.1-py2.py3-none-any.whl (31 kB)
    Downloading pyflakes-3.2.0-py2.py3-none-any.whl (62 kB)
      62.7/62.7 kB 1.5 MB/s eta 0:00:00
Installing collected packages: pyflakes, pycodestyle, mccabe, flake8
Successfully installed flake8-7.0.0 mccabe-0.7.0 pycodestyle-2.11.1 pyflakes-3.2.0
@RomanGorchakov → /workspaces/Py21 (develop) $ pre-commit sample-config > .pre-commit-config.yaml
@RomanGorchakov → /workspaces/Py21 (develop) $ conda env export > environment.yml
@RomanGorchakov → /workspaces/Py21 (develop) $
```

5. Создаём файл «example.py», в котором нужно реализовать возможность хранения данных в базе данных SQLite3 для примера в лабораторной работе 2.17.



```
E:\Основы программной инженерии\2.21\code\Пример>python example.py add --name="Иванов Иван" --post="Директор" --year=2007
E:\Основы программной инженерии\2.21\code\Пример>python example.py add --name="Петров Пётр" --post="Бухгалтер" --year=2010
E:\Основы программной инженерии\2.21\code\Пример>python example.py add --name="Сидоров Сидор" --post="Главный Инженер" --year=2012
E:\Основы программной инженерии\2.21\code\Пример>python example.py display
+-----+-----+-----+-----+
| № | Ф.И.О. | Должность | Год |
+-----+-----+-----+-----+
| 1 | Иванов Иван | Директор | 2007 |
+-----+-----+-----+-----+
| 2 | Петров Пётр | Бухгалтер | 2010 |
+-----+-----+-----+-----+
| 3 | Сидоров Сидор | Главный Инженер | 2012 |
+-----+-----+-----+-----+
E:\Основы программной инженерии\2.21\code\Пример>
```

6. Создаём файл «individual.py», в котором нужно реализовать хранение данных в базе данных SQLite3. Написать программу, выполняющую следующие действия: ввод с клавиатуры данных в список, состоящий из словарей заданной структуры; записи должны быть упорядочены по возрастанию номера рейса; вывод на экран номеров рейсов и типов самолетов, вылетающих в пункт назначения, название которого совпало с названием, введенным с клавиатуры; если таких рейсов нет, выдать на дисплей соответствующее сообщение. Необходимо также проследить за тем, чтобы файлы генерируемый этой программой не попадали в репозиторий лабораторной работы.

```
Командная строка

E:\Основы программной инженерии\2.21\code\Индивидуальные задания>python individual.py add --race="Kazan" --number=3780 --type=1

E:\Основы программной инженерии\2.21\code\Индивидуальные задания>python individual.py add --race="Moscow" --number=8850 --type=9

E:\Основы программной инженерии\2.21\code\Индивидуальные задания>python individual.py add --race="Novoalexandrovsk" --number=3234 --type=62

E:\Основы программной инженерии\2.21\code\Индивидуальные задания>python individual.py add --race="Saint-Petersburg" --number=9864 --type=48

E:\Основы программной инженерии\2.21\code\Индивидуальные задания>python individual.py add --race="Stavropol" --number=9120 --type=82

E:\Основы программной инженерии\2.21\code\Индивидуальные задания>python individual.py display
+-----+-----+-----+-----+
| No | Destination | Race number | Plane type |
+-----+-----+-----+-----+
| 1 | Kaliningrad | 4449 | 90 |
+-----+-----+-----+-----+
| 2 | Kazan | 3780 | 1 |
+-----+-----+-----+-----+
| 3 | Moscow | 8850 | 9 |
+-----+-----+-----+-----+
| 4 | Novoalexandrovsk | 3234 | 62 |
+-----+-----+-----+-----+
| 5 | Saint-Petersburg | 9864 | 48 |
+-----+-----+-----+-----+
| 6 | Stavropol | 9120 | 82 |
+-----+-----+-----+-----+

E:\Основы программной инженерии\2.21\code\Индивидуальные задания>
```

7. Выполняем коммит файлов в репозиторий Git в ветку разработки, сливаем её с веткой main и отправляем изменения на сервер GitHub.

```
@RomanGorchakov → /workspaces/Py21 (develop) $ git add .
@RomanGorchakov → /workspaces/Py21 (develop) $ git commit -m "SQL databases"
[develop cc32d83] SQL databases
5 files changed, 617 insertions(+)
create mode 100644 .pre-commit-config.yaml
create mode 100644 environment.yml
create mode 100644 "\320\230\320\275\320\264\320\270\320\262\320\270\320\264\321\203\320\260\320\273\321\214\320\275\320\276\320\265\320\267\320\260\320\264\320\260\320\275\320\270\320\265\individual.py"
create mode 100644 "\320\236\321\202\321\207\321\221\321\202\320\233\320\2402.19\320\223\320\276\321\200\321\207\320\260\320\272\320\276\320\262\320\240\320\222.pdf"
create mode 100644 "\320\237\321\200\320\270\320\274\320\265\321\200\example.py"
@RomanGorchakov → /workspaces/Py21 (develop) $ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
@RomanGorchakov → /workspaces/Py21 (main) $ git merge develop
Updating 41dd812..cc32d83
Fast-forward
 .pre-commit-config.yaml
 environment.yml
 .../individual.py
 .../\320\233\320\2402.19\320\223\320\276\321\200\321\207\320\260\320\272\320\276\320\262\320\240\320\222.pdf"
 "\320\237\321\200\320\270\320\274\320\265\321\200\example.py"
 5 files changed, 617 insertions(+)
 create mode 100644 .pre-commit-config.yaml
 create mode 100644 environment.yml
 create mode 100644 "\320\230\320\275\320\264\320\270\320\262\320\270\320\264\321\203\320\260\320\273\321\214\320\275\320\276\320\265\320\267\320\260\320\264\320\260\320\275\320\270\320\265\individual.py"
 create mode 100644 "\320\236\321\202\321\207\321\221\321\202\320\233\320\2402.19\320\223\320\276\321\200\321\207\320\260\320\272\320\276\320\262\320\240\320\222.pdf"
 create mode 100644 "\320\237\321\200\320\270\320\274\320\265\321\200\example.py"
@RomanGorchakov → /workspaces/Py21 (main) $ git push -u
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 2 threads
Compressing objects: 100% (8/8), done.
Writing objects: 100% (10/10), 554.90 KiB | 19.82 MiB/s, done.
Total 10 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/RomanGorchakov/Py21
 41dd812..cc32d83 main -> main
branch 'main' set up to track 'origin/main'.
@RomanGorchakov → /workspaces/Py21 (main) $
```

main
1 Branch
0 Tags
Go to file
Add file
Code

RomanGorchakov SQL databases
cc32d83 · 1 minute ago
5 Commits

Индивидуальное задание	SQL databases	1 minute ago
Отчёт	SQL databases	1 minute ago
Пример	SQL databases	1 minute ago
.gitignore	Create .gitignore	3 days ago
.pre-commit-config.yaml	SQL databases	1 minute ago
LICENSE	Create LICENSE	3 days ago
README.md	Update README.md	14 hours ago
environment.yml	SQL databases	1 minute ago

README
MIT license

Горчаков Роман Владимирович. Вариант 4

Лабораторная работа 2.21. Взаимодействие с базами данных SQLite3 с помощью языка программирования Python.

Сами по себе СУБД редко используются для работы с базами данных. В том смысле, что в реальных проектах связи БД + СУБД бывает недостаточно. Обычно с СУБД работают через какой-либо язык программирования. Это позволяет более гибко принимать запросы, обрабатывать ответы перед передачей их куда-либо далее. Ведь у императивного, а не декларативного как SQL, языка программирования средств для работы с данными больше, да и логика богаче.

Чтобы использовать SQLite3 в Python, прежде всего, вам нужно будет импортировать модуль sqlite3, а затем создать объект соединения, который соединит нас с базой данных и позволит нам выполнять операторы SQL. Объект соединения создается с помощью функции connect(). Будет создан новый файл под названием

Контрольные вопросы

1. Каково назначение модуля sqlite3?

Модуль sqlite3 – API к СУБД SQLite. Своего рода адаптер, который переводит команды, написанные на Питоне, в команды, которые понимает SQLite. Как и наоборот, доставляет ответы от SQLite в python-программу.

2. Как выполняется соединение с базой данных SQLite3? Что такое курсор базы данных?

Вызов функции `connect()` приводит к созданию объекта-экземпляра от класса `Connection`. Этот объект обеспечивает связь с файлом базы данных, представляет конкретную БД в программе.

Курсор `SQLite3` – метод объекта соединения. Для выполнения инструкций `SQLite3` сначала устанавливается соединение, а затем создается объект курсора с использованием объекта соединения.

3. Как подключиться к базе данных `SQLite3`, находящейся в оперативной памяти компьютера?

При создании соединения с `SQLite3` автоматически создается файл базы данных, если он еще не существует. Этот файл базы данных создается на диске, мы также можем создать базу данных в оперативной памяти с помощью функции `:memory: with the connect`.

4. Как корректно завершить работу с базой данных `SQLite3`?

Для того, чтобы корректно завершить работу с базой данных, надо применить изменения (выполнить транзакцию) и разорвать соединение.

5. Как осуществляется вставка данных в таблицу базы данных `SQLite3`?

Чтобы вставить данные в таблицу, используется оператор `INSERT INTO`.

6. Как осуществляется обновление данных таблицы базы данных `SQLite3`

Чтобы обновить данные в таблице, просто создайте соединение, затем создайте объект курсора с помощью соединения и, наконец, используйте оператор `UPDATE` в методе `execute ()`.

7. Как осуществляется выборка данных из базы данных `SQLite3`?

Оператор `SELECT` используется для выбора данных из определенной таблицы

8. Каково назначение метода `rowcount`?

`SQLite3 rowcount` используется для возврата количества строк, которые были затронуты или выбраны последним выполненным SQL-запросом.

9. Как получить список всех таблиц базы данных `SQLite3`?

Чтобы перечислить все таблицы в базе данных SQLite3, вы должны запросить данные из таблицы `sqlite_master`, а затем использовать `fetchall()` для получения результатов из инструкции `SELECT`.

10. Как выполнить проверку существования таблицы как при ее добавлении, так и при ее удалении?

Чтобы проверить, не существует ли таблица уже, мы используем `IF NOT EXISTS` с оператором `CREATE TABLE` следующим образом: `CREATE TABLE IF NOT EXISTS table_name (column1, column2, ..., columnN)`.

Аналогично, чтобы проверить, существует ли таблица при удалении, мы используем `IF EXISTS` с оператором `DROP TABLE` следующим образом: `DROP TABLE IF EXISTS table_name`.

11. Как выполнить массовую вставку данных в базу данных SQLite3?

Метод `executemany` можно использовать для вставки нескольких строк одновременно.

12. Как осуществляется работа с датой и временем при работе с базами данных SQLite3?

В базе данных Python SQLite3 мы можем легко хранить дату или время, импортируя модуль `datetime`.