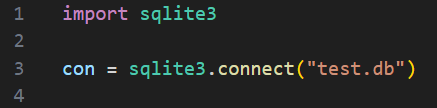
**SQL-запросы в Python**

Для того, чтобы управлять базой данных через Python скрипт, нам надо подключить встроенную библиотеку в самом начале файла:

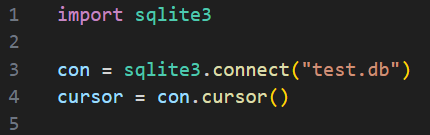


Дальше нам надо подключиться к нашей базе данных:



В кавычках указывается название файла базы данных. Если такого файла не существует, то Python автоматически создаст его.

После подключения нам надо получить так называемый “курсор” для взаимодействия с нашей базой данных:

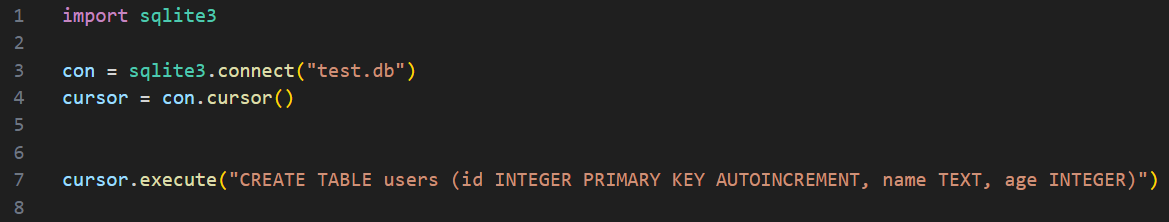


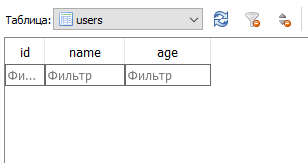
4 строка всегда одинаковая вне зависимости от названия базы данных.

Для того чтобы управлять базой данных у нас существует функция cursor.execute(“SQL-запрос”).

С помощью этой функцией можно написать любой SQL-запрос, но у разных запросов есть свои отличия.

Для того чтобы создать базу данных мы используем знакомый запрос CREATE TABLE:





В кавычках мы пишем такой же запрос, как и в приложении DB Browser for SQLite.

Но для читабельности мы можем использовать тройные кавычки:

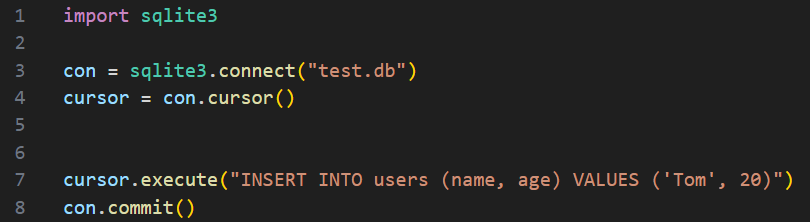


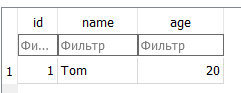
Тройные кавычки показывают Python, что все переносы относятся к строке, а не к коду.

Похожим образом используем команду DROP TABLE.

Если таблица уже существует, то вам выдаст ошибку. Чтобы этого избежать, используйте try, except.

Теперь давайте рассмотрим команду INSERT:





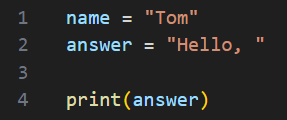
В команде INSERT работают всё те же правила, как и в приложении DB Browser for SQLite. Если в SQL-запросе надо вставить значение в столбец с типом данных TEXT, то значение надо обозначить через ‘

Но для того, чтобы завершить запрос надо дописать con.commit(). Он сохранит нашу базу данных, и запрос выполнится успешно.

con.commit() используется в командах INSERT, DELETE, UPDATE.

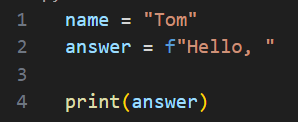
**Интерполяция строк:**

Интерполяция строк позволяет нам упростить вставку переменных в строку:

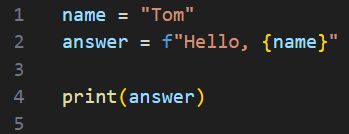


В переменную answer нам надо добавить значение перемой name. Для того, чтобы удобнее было вставлять как раз используется интерполяция строк.

Перед началом, нам надо, что строка интерполируется. Для этого перед началом кавычек ставим букву f:

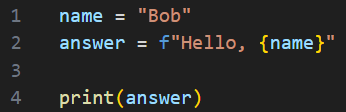


Далее в нужном месте мы должны поставить {}, и в этих скобках указываем переменную:



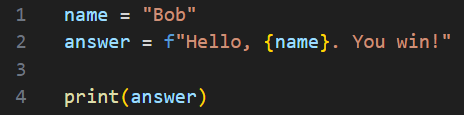


Если мы изменим значение переменной name, то и в строке answer вставится другое значение:



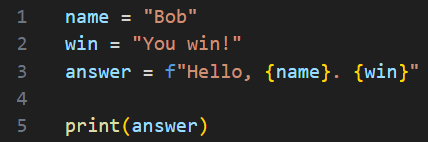


Можно после вставки, продолжить строку:





Можно вставлять несколько переменных:



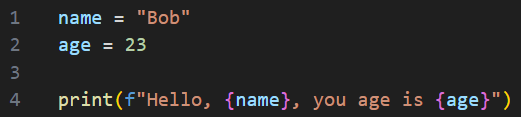


Можно вставлять другие типы данных:



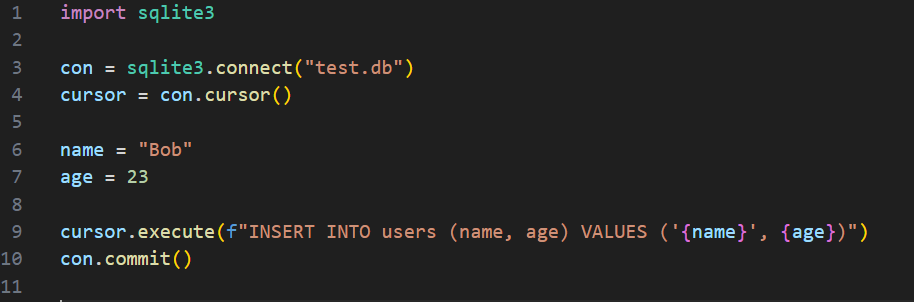


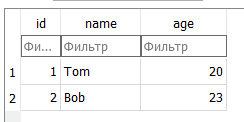
Можно использовать интерполяция не толька для переменных, а везде, где нужна строка (предложение в кавычках), но перед кавычками обязательно ставим f:





Теперь давайте попробуем использовать интерполяцию в нашем запросе:

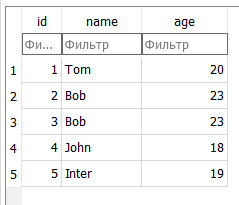




Важно следить за апострофами при вставке. Если мы вставляем значение в бд в столбец со значением TEXT, то до и после фигурных скобок ставим апостроф ‘

Если надо вставить сразу несколько строк одновременно, то не обязательно постоянно писать commit(). Эту функцию надо писать после завершения изменений в бд:

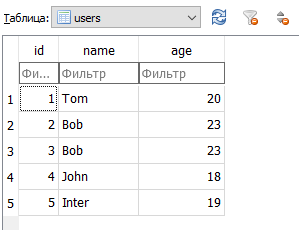




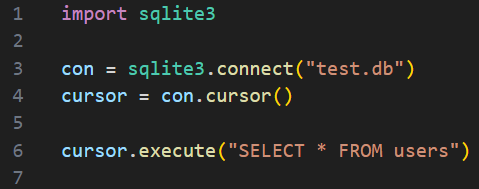
Аналогично пишутся SQL-запросы с UPDATE, DELETE, а именно execute() и commit(). Не забывайте, что вы можете использовать WHERE, OR, AND и т.д.  
Всё то, что есть в SQL-запросах.

Давайте теперь рассмотрим, как получать данные из бд.

У нас есть бд:



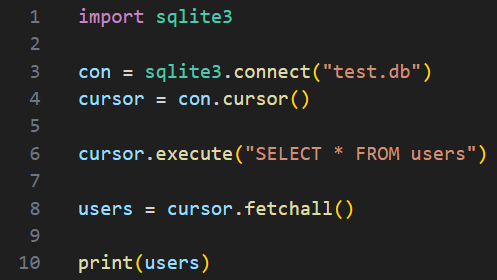
Для того, чтобы получить данные из бд, мы используем SELECT. Например:



Как мы помним – этот запрос позволит вывести все данные из бд.

Обратите внимание, что и тут мы используем execute() потому, что он необходим, чтобы выполнить запрос.

Но если мы запустим скрипт, то ничего не произойдёт, т.к. нам куда-то надо присвоить значения в нашем коде. Для этого есть функция cursor.fetchall():

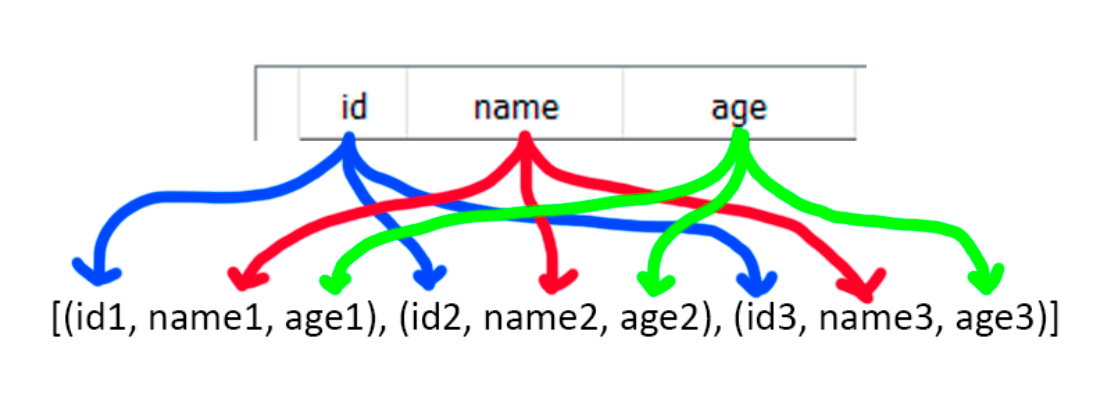




cursor.fetchall() необходимо использовать всегда, когда мы хотим достать данные.

Но если мы посмотрим на вывод, то мы увидим много скобочек, и как в них разобраться, что нам необходимо?

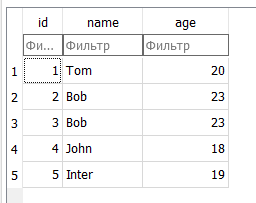
В Python квадратными скобочками обозначаются списки, а круглыми – кортежи. То есть это список с кортежами, но как нам понять, какие данные соответствуют столбцам в нашей бд:



Получается так, что в кортеже значения идут в таком же порядке, как идут наши столбцы в бд. В данном случае в кортеже сначала идёт id, потом name, потом age. Если мы столбцы name и age при создании поменяем местами, то в кортеже это будет выглядеть так: (id, age, name).

Но как же нам обратиться к элементам нашего списка и вывести определенное значение?

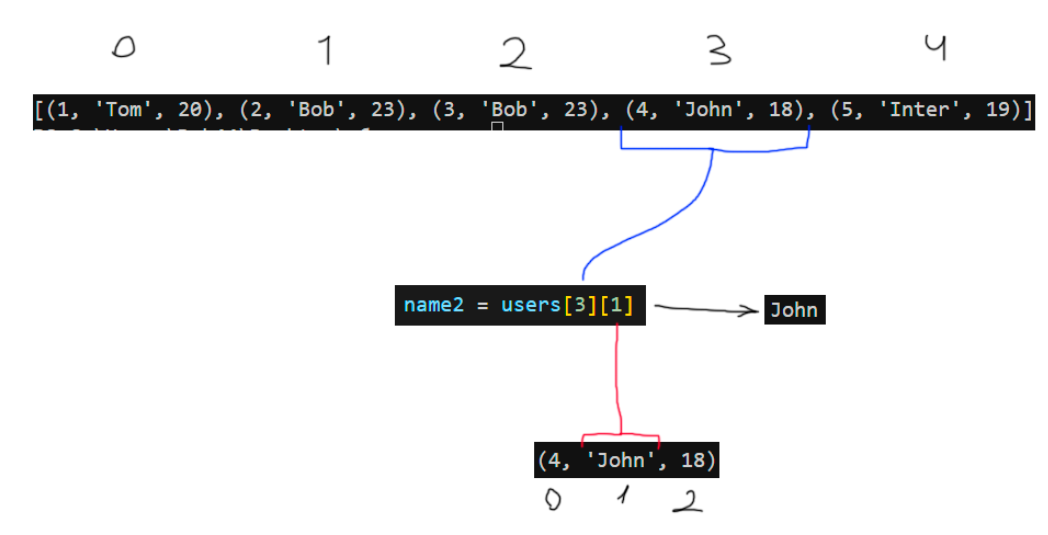






К элементам списка и кортежа можно обратиться через индекс. Поскольку список и кортеж — это контейнер и у нас контейнер в контейнере, то мы указываем в первых квадратных скобках индекс элемента, т.е. какой кортеж мы будем рассматривать, а в вторых квадратных скобка мы указываем уже конкретное значение ячейки.

Схема будет выглядеть примерно так:



Не забываем, что индексы начинаются с 0.

Для SELECT работают всё те же правила, что и в DB Browser for SQLite (WHERE, определённые столбцы, AND, OR и т.д.).

Важно подметить, что, если вы раз использовали cursor.fetchall(), то при повторном использовании у вас ничего не будет. Вам придётся заново использовать cursor.execute().

Разница типов данных:

|  |  |  |
| --- | --- | --- |
| **Python** | **SQLite** | **Представление** |
| None | NULL |  |
| int | INTEGER | 425 |
| float | REAL | 425,123 |
| str | TEXT | Hello World |
| bool | Отсутствует\* | True, False |

\* В SQLite отсутствует True и False. Чтобы имитировать True, False, надо использовать тип данных INTEGER, где используется только 2 значения: 1 – True, 0 – False.

**Задание 1:**

С помощью Python сделайте скрипт, который:

1. Создаёт файл person1DB.db
2. Создаёт таблицу persons в файле personDB.db:

id – число, первичный ключ, автоматическое увеличение

name – текст

email – текст

hours – число

itemsPerHours – число

1. Заполните бд 10-ю строками
2. Достать из бд: id, name, email, где hours \* itemsPerHours больше 14
3. Вывести в консоль данные из 4 пункта и оформить вывод

**Задание 2:**

С помощью Python сделайте скрипт, который:

1. Создайте файл person2DB.db
2. Создаёт таблицу persons в файле personDB.db:

id - число, первичный ключ, автоматическое увеличение

name – текст

age – число

weight – число

height – число

bmi - десятичная дробь (число с запятой)

1. Заполните эту бд на 1000 строк, НО вы должны их заполнить случайными значениями. Т.е. вам надо написать код, которые случайно генерирует name, age, weight, height (используйте библиотеку random).   
   id всегда идёт по порядку и нет необходимости его заполнять.  
   bmi (индекс массы тела) надо рассчитать по формуле:
2. Достать данные из бд: id, name, bmi, где bmi меньше 20 и bmi больше 30
3. Вывести в консоль данные из 4 пункта и оформить вывод

**Творческое задание:**

Создать Python скрипт, по управлению товаров на складе.

ТЗ:

1. Реализовать возможность просмотра всех товаров на складе (айди, название, количество, цена за 1 шт., сумма за вид товара, сумму стоимости всех товаров за все виды на складе)
2. Реализовать возможность добавления нового товара на склад
3. Реализовать возможность изменения количества товаров на складе в двух вариациях:
   1. Напрямую изменить количество
   2. Увеличить (+) или уменьшить (-) количество
4. Реализовать возможность удаления товара из бд
5. Оформить понятный и удобный интерфейс (в рамках терминала Python)