

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №8
дисциплины «Программирование на языке Python»

Вариант_15_

Выполнила:
Маньшина Дарья Алексеевна
2 курс, группа ИТС-б-о-22-1,
11.03.02 «Инфокоммуникационные
технологии и системы связи»,
направленность (профиль)
«Инфокоммуникационные системы и
сети», очная форма обучения

(подпись)

Руководитель практики:
Воронкин Р. А., канд. тех. наук,
доцент кафедры
инфокоммуникаций

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

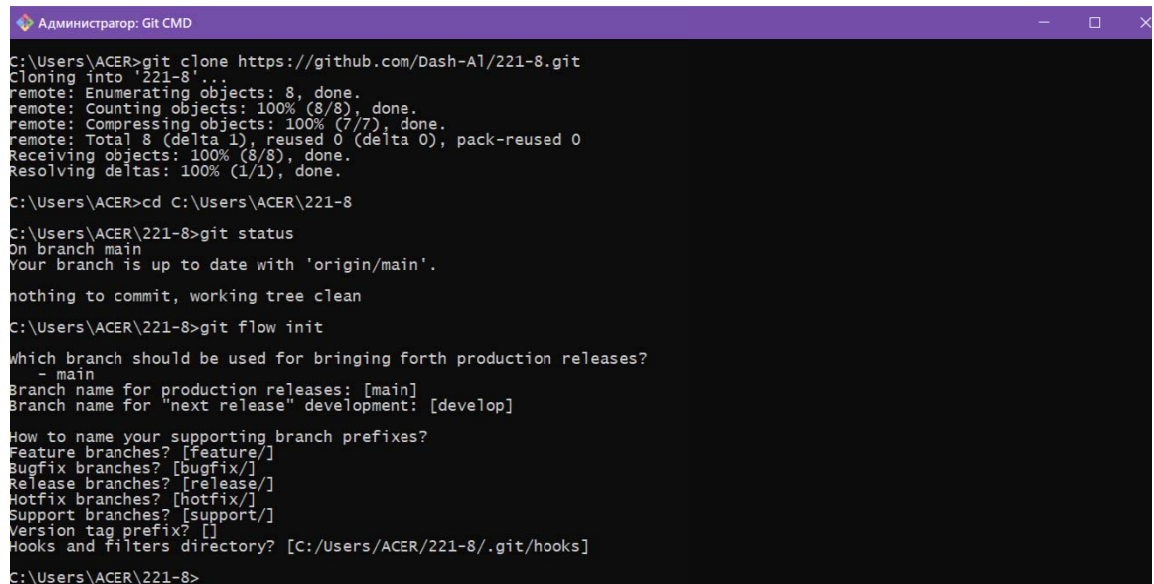
Ставрополь, 2023 г.

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

Цель: приобретение навыков по работе с базами данных SQLite3 с помощью языка программирования Python.

Ход работы:

1. Создадим новый репозиторий. Клонировем его и сделаем способ ветвления git-flow. Также установим пакеты: black, isort, flake8



```
Администратор: Git CMD
C:\Users\ACER>git clone https://github.com/Dash-A1/221-8.git
Cloning into '221-8'...
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 8 (delta 1), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (8/8), done.
Resolving deltas: 100% (1/1), done.

C:\Users\ACER>cd C:\Users\ACER\221-8

C:\Users\ACER\221-8>git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

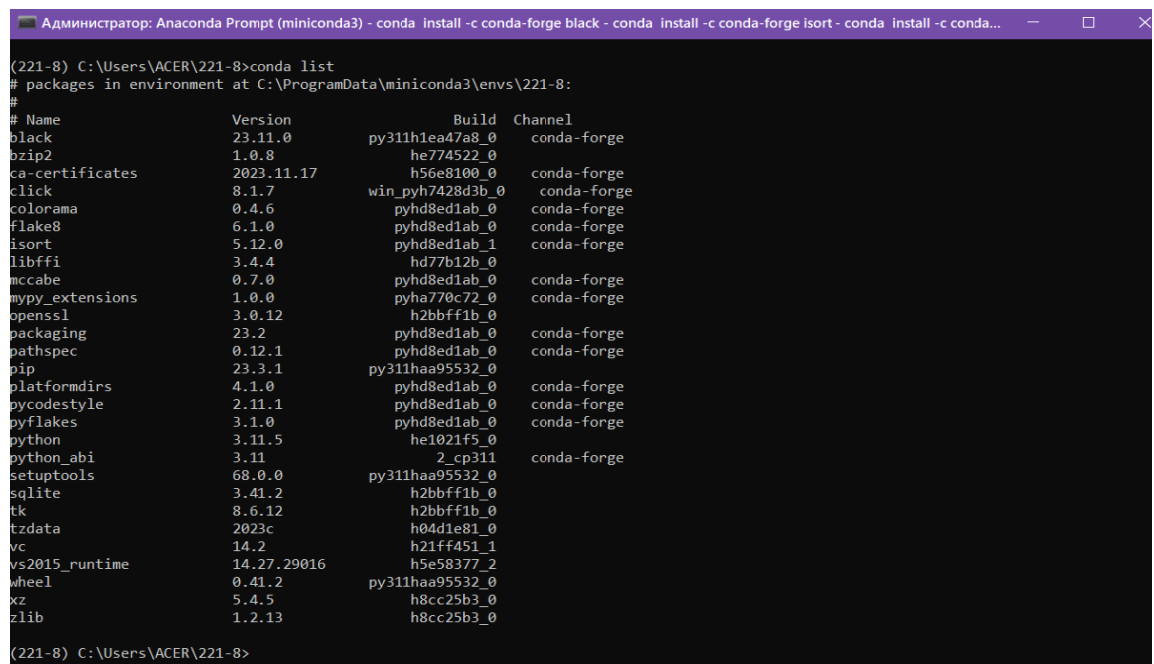
C:\Users\ACER\221-8>git flow init

which branch should be used for bringing forth production releases?
- main
Branch name for production releases: [main]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [C:/Users/ACER/221-8/.git/hooks]

C:\Users\ACER\221-8>
```

Рисунок 1. Клонирование репозитория



```
Администратор: Anaconda Prompt (miniconda3) - conda install -c conda-forge black - conda install -c conda-forge isort - conda install -c conda-forge flake8

(221-8) C:\Users\ACER\221-8>conda list
# packages in environment at C:\ProgramData\miniconda3\envs\221-8:
#
# Name                    Version            Build                Channel
black                     23.11.0            py311h1ea47a8_0     conda-forge
bzip2                     1.0.8              he774522_0          conda-forge
ca-certificates           2023.11.17         h56e8100_0          conda-forge
click                     8.1.7              win_pyh7428d3b_0   conda-forge
colorama                  0.4.6              pyhd8ed1ab_0        conda-forge
flake8                    6.1.0              pyhd8ed1ab_0        conda-forge
isort                      5.12.0             pyhd8ed1ab_1        conda-forge
libffi                     3.4.4              hd77b12b_0          conda-forge
mccabe                    0.7.0              pyhd8ed1ab_0        conda-forge
mypy_extensions           1.0.0              pyha770c72_0        conda-forge
openssl                    3.0.12             h2bbff1b_0          conda-forge
packaging                  23.2               pyhd8ed1ab_0        conda-forge
pathspec                   0.12.1             pyhd8ed1ab_0        conda-forge
pip                        23.3.1             py311haa95532_0     conda-forge
platformdirs              4.1.0              pyhd8ed1ab_0        conda-forge
pycodestyle                2.11.1             pyhd8ed1ab_0        conda-forge
pyflakes                   3.1.0              pyhd8ed1ab_0        conda-forge
python                     3.11.5             he1021f5_0          conda-forge
python_abi                 3.11               2_cp311             conda-forge
setuptools                 68.0.0             py311haa95532_0     conda-forge
sqlite                     3.41.2             h2bbff1b_0          conda-forge
tk                          8.6.12            h2bbff1b_0          conda-forge
tzdata                     2023c              h04d1e81_0          conda-forge
vc                          14.2               h21ff451_1          conda-forge
vs2015_runtime             14.27.29016        h5e58377_2          conda-forge
wheel                      0.41.2             py311haa95532_0     conda-forge
xz                          5.4.5              h8cc25b3_0          conda-forge
zlib                       1.2.13             h8cc25b3_0          conda-forge

(221-8) C:\Users\ACER\221-8>
```

Рисунок 2. Установка пакетов

Выполним пример, предоставленный в методическом указании.

Условие:

Программа:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import argparse
import sqlite3
import typing as t
from pathlib import Path

def display_workers(staff: t.List[t.Dict[str, t.Any]]) -> None:
    # Проверить, что список работников не пуст.
    if staff:
        # Заголовок таблицы.
        line = '+-{}-{}-{}-{}-{}-{}-+'.format(
            '-' * 4,
            '-' * 30,
            '-' * 20,
            '-' * 8
        )
        print(line)
        print(
            '| {:^4} | {:^30} | {:^20} | {:^8} |'.format(
                "№",
                "Ф.И.О.",
                "Должность",
                "Год"
            )
        )
        print(line)

        # Вывести данные о всех сотрудниках.
        for idx, worker in enumerate(staff, 1):
            print(
                '| {:>4} | {:<30} | {:<20} | {:>8} |'.format(
                    idx,
                    worker.get('name', ''),
                    worker.get('post', ''),
                    worker.get('year', 0)
                )
            )
            print(line)
    else:
        print("Список работников пуст.")

def create_db(database_path: Path) -> None:
    conn = sqlite3.connect(database_path)
    cursor = conn.cursor()

    # Создать таблицу с информацией о должностях.
```

```

cursor.execute(
)

# Создать таблицу с информацией о работниках.
cursor.execute(
)

conn.close()

def add_worker(
    database_path: Path,
    name: str,
    post: str,
    year: int
) -> None:
    conn = sqlite3.connect(database_path)
    cursor = conn.cursor()

    # Получить идентификатор должности в базе данных.
    # Если такой записи нет, то добавить информацию о новой должности.
    cursor.execute(
        """
        SELECT post_id FROM posts WHERE post_title = ?
        """,
        (post,)
    )
    row = cursor.fetchone()
    if row is None:
        cursor.execute(
            '
            (post,)
            )
        post_id = cursor.lastrowid

    else:
        post_id = row[0]

    # Добавить информацию о новом работнике.
    cursor.execute(
        (name, post_id, year)
    )

    conn.commit()
    conn.close()

def select_all(database_path: Path) -> t.List[t.Dict[str, t.Any]]:
    conn = sqlite3.connect(database_path)
    cursor = conn.cursor()
    cursor.execute(
    )
    rows = cursor.fetchall()

    conn.close()
    return [
    {

```

```

"name": row[0],
"post": row[1],
"year": row[2],
}
for row in rows
]
def select_by_period(
    database_path: Path, period: int
) -> t.List[t.Dict[str, t.Any]]:
    conn = sqlite3.connect(database_path)
    cursor = conn.cursor()
    cursor.execute(
        ,
        (period,)
    )
    rows = cursor.fetchall()
    conn.close()
    return [
        {
            "name": row[0],
            "post": row[1],
            "year": row[2],
        }
        for row in rows
    ]

def main(command_line=None):
    # Создать родительский парсер для определения имени файла.
    file_parser = argparse.ArgumentParser(add_help=False)
    file_parser.add_argument(
        "--db",
        action="store",
        required=False,
        default=str(Path.home() / "workers.db"),
        help="The database file name"
    )
    # Создать основной парсер командной строки.
    parser = argparse.ArgumentParser("workers")
    parser.add_argument(
        "--version",
        action="version",
        version=f"%(prog)s 0.1.0"
    )
    subparsers = parser.add_subparsers(dest="command")
    # Создать субпарсер для добавления работника.
    add = subparsers.add_parser(
        "add",
        parents=[file_parser],
        help="Add a new worker"
    )
    add.add_argument(
        "-n",
        "--name",
        action="store",
        required=True,
        help="The worker's name"
    )

```

```

)
add.add_argument(
    "-p",
    "--post",
    action="store",
    help="The worker's post"
)
add.add_argument(
    "-y",
    "--year",
    action="store",
    type=int,
    required=True,
    help="The year of hiring"
)
# Создать субпарсер для отображения всех работников.
_ = subparsers.add_parser(
    "display",
    parents=[file_parser],
    help="Display all workers"
)
# Создать субпарсер для выбора работников.
select = subparsers.add_parser(
    "select",
    parents=[file_parser],
    help="Select the workers"
)
select.add_argument(
    "-p",
    "--period",
    action="store",
    type=int,
    required=True,
    help="The required period"
)
# Выполнить разбор аргументов командной строки.
args = parser.parse_args(command_line)
# Получить путь к файлу базы данных.
db_path = Path(args.db)
create_db(db_path)
# Добавить работника.
if args.command == "add":
    add_worker(db_path, args.name, args.post, args.year)
# Отобразить всех работников.
elif args.command == "display":
    display_workers(select_all(db_path))
# Выбрать требуемых работников.
elif args.command == "select":
    display_workers(select_by_period(db_path, args.period))
pass
if __name__ == "__main__":
    main()

```

Результат:

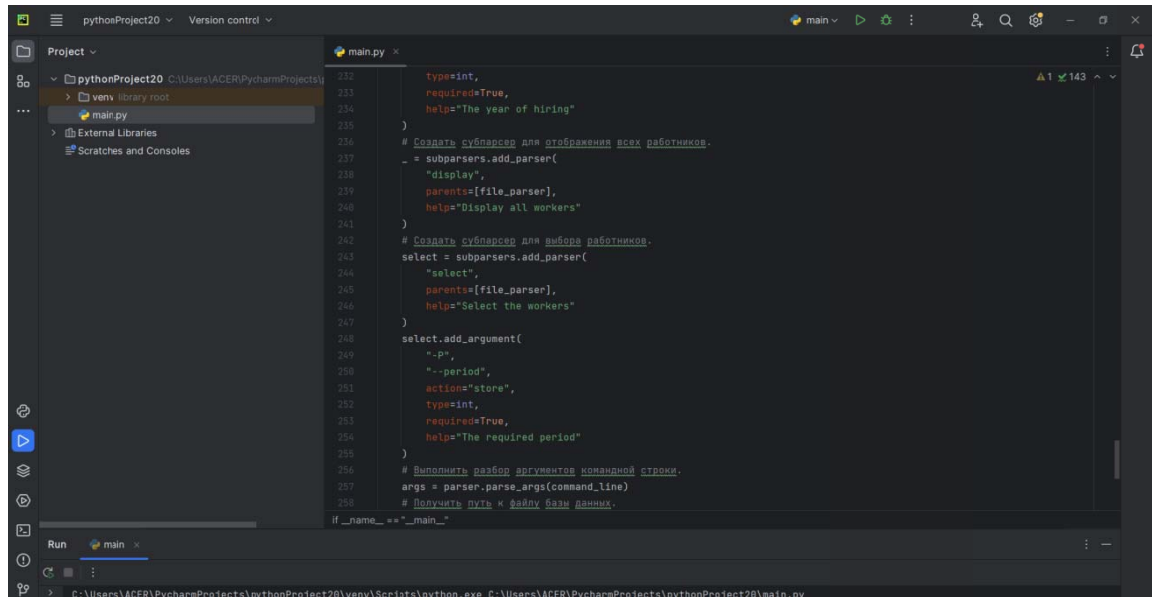


Рисунок №3. Выполнение примера №1

Выполним индивидуальное задание.

Условие: # использовать словарь, содержащий следующие ключи: фамилия, имя; знак Зодиака; дата рождения (список из трёх чисел). Написать программу, выполняющую следующие действия: ввод с клавиатуры данных в список, состоящий из словарей заданной структуры; записи должны быть упорядочены по датам рождения; вывод на экран информацию о людях, родившихся под знаком, название которого введено с клавиатуры; если таких нет, выдать на дисплей соответствующее сообщение. Оформив каждую команду в виде отдельной функции. Дополнительно реализовать сохранение и чтение данных из файла формата JSON. Дополнительно реализовать интерфейс командной строки (CLI). Необходимо реализовать хранение данных в базе данных SQLite3. Информация в базе данных должна храниться не менее чем в двух таблицах.

Программа:

- # использовать словарь, содержащий следующие ключи: фамилия, имя; знак Зодиака; дата рождения (список из трёх чисел).
- # Написать программу, выполняющую следующие действия:
- # ввод с клавиатуры данных в список, состоящий из словарей заданной структуры;
- # записи должны быть упорядочены по датам рождения; вывод на экран информацию о людях, родившихся под знаком, название которого введено с клавиатуры;

если таких нет, выдать на дисплей соответствующее сообщение.
Оформи каждую команду в виде отдельной функции.
Дополнительно реализовать сохранение и чтение данных из файла формата JSON.
Дополнительно реализовать интерфейс командной строки (CLI).
Необходимо реализовать хранение данных в базе данных SQLite3. Информация в базе данных должна храниться не менее чем в двух таблицах.

```
#!/usr/bin/env python3  
# -*- coding: utf-8 -*-
```

```
import json  
import sqlite3
```

```
# Функция для ввода данных с клавиатуры  
def input_data():  
    last_name = input("Введите фамилию: ")  
    first_name = input("Введите имя: ")  
    zodiac_sign = input("Введите знак Зодиака: ")  
    day = int(input("Введите день рождения: "))  
    month = int(input("Введите месяц рождения: "))  
    year = int(input("Введите год рождения: "))  
    return {  
        'фамилия': last_name,  
        'имя': first_name,  
        'знак Зодиака': zodiac_sign,  
        'дата рождения': [day, month, year]  
    }
```

```
# Функция для сортировки списка по датам рождения  
def sort_by_birthday(people):  
    return sorted(people, key=lambda x: x['дата рождения'])
```

```
# Функция для вывода информации о людях с заданным знаком Зодиака  
def print_people_by_zodiac(people, zodiac):  
    found = False  
    for person in people:  
        if person['знак Зодиака'] == zodiac:  
            print(person)  
            found = True  
    if not found:  
        print("Нет людей с таким знаком Зодиака")
```

```
# Функция для сохранения данных в файл формата JSON  
def save_to_json(people, file_name):  
    with open(file_name, 'w') as file:  
        json.dump(people, file)
```

```
# Функция для чтения данных из файла формата JSON  
def load_from_json(file_name):  
    with open(file_name, 'r') as file:  
        return json.load(file)
```

```
# Функция для создания таблиц в базе данных SQLite3  
def create_tables(conn):  
    c = conn.cursor()  
    c.execute("CREATE TABLE IF NOT EXISTS People  
        (last_name text, first_name text, zodiac_sign text, day integer, month integer, year integer)")
```



```

c.execute("""CREATE TABLE IF NOT EXISTS Zodiacs
(zodiac_sign text)""")

# Функция для сохранения данных в базу данных SQLite3
def save_to_database(people, conn):
    c = conn.cursor()
    c.execute('DELETE FROM People')
    c.execute('DELETE FROM Zodiacs')
    for person in people:
        c.execute('INSERT INTO People VALUES (?, ?, ?, ?, ?)', (
            person['фамилия'], person['имя'], person['знак Зодиака'],
            person['дата рождения'][0], person['дата рождения'][1], person['дата рождения'][2]))
        c.execute('INSERT OR IGNORE INTO Zodiacs VALUES (?)', (person['знак Зодиака'],))
    conn.commit()

# Функция для чтения данных из базы данных SQLite3
def load_from_database(conn):
    c = conn.cursor()
    people = []
    for row in c.execute('SELECT * FROM People'):
        person = {
            'фамилия': row[0],
            'имя': row[1],
            'знак Зодиака': row[2],
            'дата рождения': [row[3], row[4], row[5]]
        }
        people.append(person)
    return people

# Функция для работы с интерфейсом командной строки
def cli():
    conn = sqlite3.connect('database.db')
    create_tables(conn)

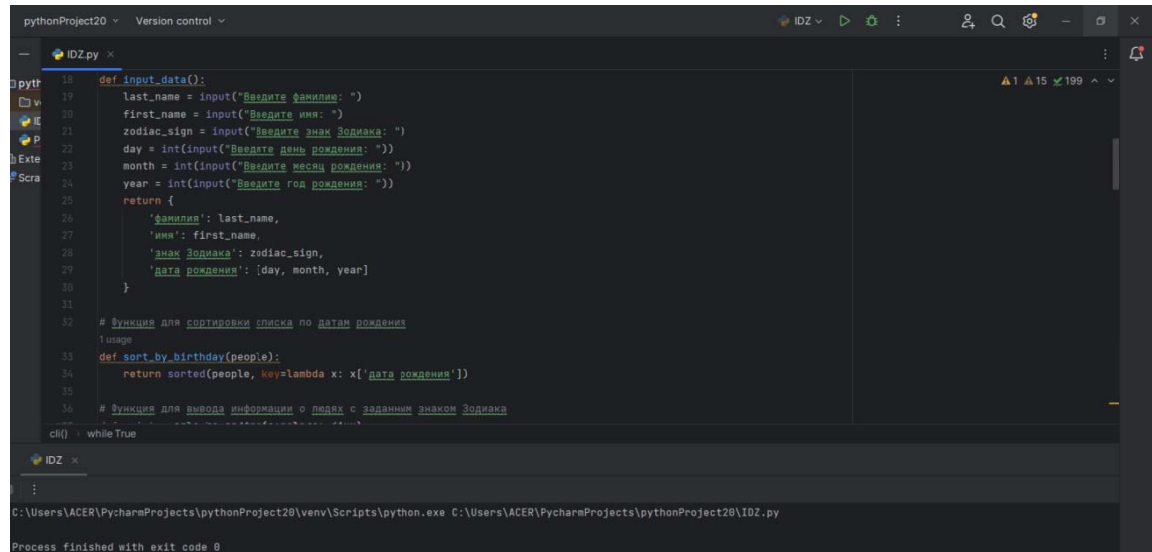
    while True:
        print("1. Ввести данные в список")
        print("2. Сохранить данные в файл JSON")
        print("3. Загрузить данные из файла JSON")
        print("4. Вывести людей с заданным знаком Зодиака")
        print("5. Вывести сохраненные людей с заданным знаком Зодиака из базы данных")
        print("6. Выход")

        choice = input("Выберите действие: ")

        if choice == '1':
            people = []
            count = int(input("Введите количество людей: "))
            for _ in range(count):
                people.append(input_data())
            people = sort_by_birthday(people)
            save_to_database(people, conn)
        elif choice == '2':
            file_name = input("Введите имя файла: ")
            people = load_from_database(conn)
            save_to_json
        if __name__ == "__main__":
            main()

```

Результат:



```
pythonProject20  Version control
IDZ.py
def input_data():
    last_name = input("Введите фамилию: ")
    first_name = input("Введите имя: ")
    zodiac_sign = input("Введите знак Зодиака: ")
    day = int(input("Введите день рождения: "))
    month = int(input("Введите месяц рождения: "))
    year = int(input("Введите год рождения: "))
    return {
        'фамилия': last_name,
        'имя': first_name,
        'знак Зодиака': zodiac_sign,
        'дата рождения': [day, month, year]
    }

# Функция для сортировки списка по датам рождения
usage
def sort_by_birthday(people):
    return sorted(people, key=lambda x: x['дата рождения'])

# Функция для вывода информации о людях с заданным знаком Зодиака
cli() while True
```

Process finished with **exit** code 0

Рисунок №4. Решение ИДЗ

Вывод: в ходе лабораторной работыприобрела навыки по работес базами данных SQLite3 с помощью языка программирования Python.

Контрольные вопросы

1. Каково назначение модуля sqlite3?

Модуль sqlite3 предназначен для работы с базами данных SQLite. Он предоставляет интерфейс для выполнения SQL-запросов и работы с данными.

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

Соединение с базой данных выполняется методом connect. Курсор базы данных - это объект, который используется для обработки результатов SQL-запроса.

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

Для подключения к базе данных в оперативной памяти нужно использовать URI “file::memory:”.

4. Как корректно завершить работу с базой данных SQLite3?

Корректное завершение работы с базой данных включает в себя закрытие соединения, освобождение курсоров и освобождение транзакций.

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

Вставка данных в таблицу выполняется методом execute с SQL-инструкцией INSERT.

6. Как осуществляется обновление данных таблицы базы данных SQLite3?

Обновление данных таблицы выполняется методом execute с SQL-инструкцией UPDATE.

7. Как осуществляется выборка данных из базы данных SQLite3?

Выборка данных выполняется методом executequery с SQL-запросом.

8. Каково назначение метода rowcount?

Метод rowcount возвращает количество измененных строк при выполнении запроса.

9. Как получить список всех таблиц базы данных SQLite3?

Список таблиц можно получить с помощью метода tables.

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

Проверка существования таблицы может быть выполнена с помощью метода exists.

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

Массовая вставка данных может быть выполнена с использованием SQL-инструкций `INSERT INTO ... VALUES(...)` или `INSERT INTO ... SELECT ...`

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

Работа с датой и временем в SQLite3 осуществляется с использованием встроенных функций даты и времени.

#