МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ» ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ

Отчет о лабораторной работе №2.21 по дисциплине основы программнойинженерии

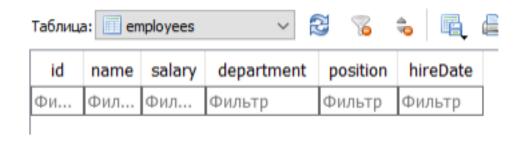
Выполнил: XXX XXX XXX, 2 курс, группа ПИЖ-б-о-20-1,

Проверил: Доцент кафедры прикладной математики и компьютерной безопасности, Воронкин Р.А.

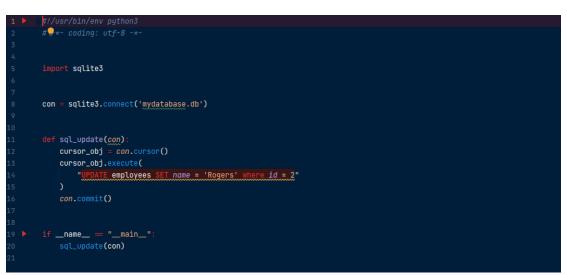
Отчет защищен с оценкой	Дата защиты

ВЫПОЛНЕНИЕ:

Примеры из МУ:



```
def sql_insert(con, entities):
    cursor_obj = con.cursor()
    cursor_obj.execute(
    INSERT INTO employees VALUES(1, 'John', 700, 'HR', 'Manager', '2017 - 01 - 04')
   if __name__ = "__main__":
    entities = (2, 'Andrew', 800, 'IT', 'Tech', '2018-02-86')
    sql_insert(con, entities)
                                                                                                    .
                                                        2
                                                                                  4
Таблица: employees
        id
                 name
                               salary
                                            department
                                                                   position
                                                                                       hireDate
     Фи...
                             Фил...
                                           Фильтр
                Фильтр
                                                                  Фильтр
                                                                                  Фильтр
                                                                  Manager
            1 John
                                700.0 HR
                                                                                  2017 - 01 - 04
 1
```



Tech

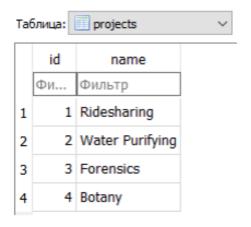
2018-02-06

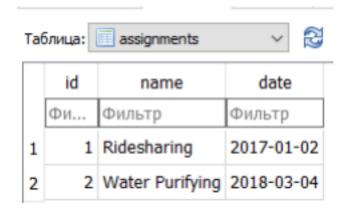
2

2 Andrew

800.0 IT

	id	name	salary	department	position	hireDate
	Фи	Фил	Фил	Фильтр	Фильтр	Фильтр
1	1	John	700.0	HR	Manager	2017 - 01 - 04
2	2	Rogers	800.0	Π	Tech	2018-02-06





```
#!/usr/bin/env python3
import argparse import sqlite3
import typing as t
from pathlib import Path
def display_workers(staff: t.List[t.Dict[str, t.Any]]) \rightarrow None:
          line = '+-{}-+-{}-+-{}-+'.format(
               '-' * 20,
'-' * 8
          print(line)
          print(
                    "No",
                    "Ф.И.О.",
                    "Должность",
                    "Год"
          print(line)
          for idx, worker in enumerate(staff, 1):
               print(
                         worker.get('name', ''),
                        worker.get('post', ''),
worker.get('year', 0)
               print(line)
          print("Список работников пуст.")
{\tt def\ create\_db}({\tt database\_path}\colon {\tt Path}) \,\to\, {\tt None}\colon
     conn = sqlite3.connect(database_path)
     cursor = conn.cursor()
     cursor.execute(
  CREATE TABLE IF NOT EXISTS posts (

post_id INTEGER PRIMARY KEY AUTOINCREMENT,

post_title TEXT NOT NULL
```

```
cursor.execute(
      EATE TABLE IF NOT EXISTS WORKERS (
worker id INTEGER PRIMARY KEY AUTOINCREMENT,
        worker_name TEXT NOT
       post_id INT
       worker_year INTEGER NOT NU
               conn.close()
       database_path: Path,
       post: str,
   conn = sqlite3.connect(database_path)
   cursor = conn.cursor()
    cursor.execute(
            CT post_id FROM posts WHERE post_title = ?
   row = cursor.fetchone()
        cursor.execute(
       INSERT INTO posts (post_title) VALUES (?)
            (post,)
        post_id = cursor.lastrowid
       post_id = row[0]
    cursor.execute(
    """
INSERT INTO workers (worker_name, post_id, worker_year)
        (name, post_id, year)
   conn.commit()
   conn.close()
def select_all(database\_path: Path) \rightarrow t.List[t.Dict[str, t.Any]]:
   conn = sqlite3.connect(database_path)
   cursor = conn.cursor()
   cursor.execute(
```

```
workers.worker_name, posts.post_title, workers.worker_year
             workers
                   N posts ON posts.post_id = workers.post_id
    rows = cursor.fetchall()
    conn.close()
             "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(
               workers.worker_name, posts.post_title, workers.worker_year
            1 workers
              | JOIN posts ON posts.post_id = workers.post_id
| (strftime('%Y', date('now')) - workers.worker_year) | > ?
        (period,)
    rows = cursor.fetchall()
    conn.close()
             "name": row[0],
            "post": row[1]
            "year": row[2],
        for row in rows
def main(command_line=None):
    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"
    parser = argparse.ArgumentParser("workers")
    parser.add_argument(
       "--version",
        action="version"
        version="%(prog)s 0.1.0"
    subparsers = parser.add_subparsers(dest="command")
```

```
parents=[file_parser],
               help="Add a new worker"
            add.add_argument(
               action="store".
               required=True
              help="The worker's name"
               action="store"
               help="The worker's post"
           add.add_argument(
              type=int,
               help="The year of hiring"
           _ = subparsers.add_parser(
   "display",
           "display",
parents=[file_parser],
               help="Display all workers"
          select = subparsers.add_parser(
               parents=[file_parser],
               help="Select the workers"
           select.add_argument(
               required=True
               help="The required period"
           args = parser.parse_args(command_line)
           db_path = Path(args.db)
           create_db(db_path)
          if args.command = "add":
               add_worker(db_path, args.name, args.post, args.year)
          elif args.command = "display":
              display_workers(select_all(db_path))
           # Выбрать требуемых рааботников.
elif args.command = "select":
            display_workers(select_by_period(db_path, args.period))
       if __name__ = "__main__":
PS C:\Users\student-09-525\PycharmProjects\OPI_9\Examples> python .\example.py add --db="workers.db" --name="Mqbcjb" --post="QEkhds" -
year="2018"
PS C:\Users\student-09-525\PycharmProjects\OPI_9\Examples> python .\example.py add --db="workers.db" --name="Knqmcn" --post="Ejhiqjxk"
--vear="2015"
PS C:\Users\student-09-525\PycharmProjects\OPI_9\Examples> python .\example.py display --db="workers.db"
No | Ф.И.О.
                                 | Должность | Год |
 1 | Mabcib
                                 | Ejhiqjxk | 2015 |
 2 | Knamen
PS C:\Users\student-09-525\PycharmProjects\OPI_9\Examples> python .\example.py select --db="workers.db" --period=12
Список работников пуст.
PS C:\Users\student-09-525\PycharmProjects\OPI_9\Examples> python .\example.py select --db="workers.db" --period=5
No | Ф.И.О. | Должность | Год |
                                 | Ejhiqjxk | 2015 |
 1 | Kngmcn
PS C:\Users\student-09-525\PycharmProjects\OPI_9\Examples>
```

```
14 🛦 1 🗶
import argparse
import sqlite3
import typing as t
from pathlib import Path
def display_flights(flights: t.List[t.Dict[str, t.Any]]) \rightarrow None:
    Отобразить список рейсов.
    if flights:
         line = '+-{}-+-{}-+-{}-+-{}-+'.format(
         print(line)
         print(
              '| {:^4} | {:^30} | {:^20} | {:^15} |'.format(
                   "No",
                   "Номер рейса",
                   "Тип самолёта"
         print(line)
         for idx, flight in enumerate(flights, 1):
              print(
                        idx,
                        flight.get('flight_destination', ''),
                        flight.get('flight_number', ''),
flight.get('airplane_type', 0)
              print(line)
         print("Список рейсов пуст.")
def create_db(database_path: Path) \rightarrow None:
    conn = sqlite3.connect(database_path)
    cursor = conn.cursor()
    cursor.execute(
     CREATE TABLE IF NOT EXISTS flight_numbers (
    num_id INTEGER PRIMARY KEY AUTOINCREMENT,
    num_title TEXT NOT NULL
    cursor.execute(
   CREATE TABLE IF NOT EXISTS flights (
flight_id INTEGER PRIMARY KEY AUTOINCREMENT,
```

```
flight_destination TEXT NOT NUL
         num_id ]
        airplane_type TEXT NOT NULL,
FOREIGN KEY(num_id) REFERENCES flight_numbers(num_id)
    conn.close()
def add_flight(
        database_path: Path,
        flight_destination: str,
        flight_number: str,
        airplane_type: int
    conn = sqlite3.connect(database_path)
    cursor = conn.cursor()
    cursor.execute(
              CT num_id FROM flight_numbers WHERE num_title = ?
         .
(flight_number,)
    row = cursor.fetchone()
    if row is None:
        cursor.execute(
                    INTO flight_numbers (num_title) VALUES (?)
             (flight_number,)
        num_id = cursor.lastrowid
        num\_id = row[0]
    cursor.execute(
      INSERT INTO flights (flight_destination, num_id, airplane_type)

VALUES (?, ?, ?)
        (flight_destination, num_id, airplane_type)
    conn.commit()
    conn.close()
\texttt{def} \ \texttt{select\_all}(\textit{database\_path} \colon \texttt{Path}) \ \to \ \texttt{t.List[t.Dict[str, t.Any]]} \colon
    conn = sqlite3.connect(database_path)
    cursor = conn.cursor()
    cursor.execute(
     """SELECT flights.flight_destination, flight_numbers.num_title,
        flights.airplane_type FROM flights I
                                                          N flight_numbers
        flight_numbers.num_id = flights.num_id """
    rows = cursor.fetchall()
    conn.close()
```

```
"flight_destination": row[0],
                                                                                                               14 🗚 1 💥
            "flight_number": row[1],
            "airplane_type": row[2],
        for row in rows
def select_flights(
       database_path: Path, air_type: str
) → t.List[t.Dict[str, t.Any]]:
    conn = sqlite3.connect(database_path)
    cursor = conn.cursor()
    cursor.execute(
              flights.flight_destination, flight_numbers.num_title,
        flights.airplane_type
           OM flights
                  flight_numbers ON flight_numbers.num_id = flights.num_id
             flights.airplane_type = ?
        (air_type,)
   rows = cursor.fetchall()
    conn.close()
            "flight_destination": row[0],
            "flight_number": row[1],
            "airplane_type": row[2]
        for row in rows
def main(command_line=None):
    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"
    parser = argparse.ArgumentParser("workers")
    parser.add_argument(
       action="version"
       version="%(prog)s 0.1.0"
    subparsers = parser.add_subparsers(dest="command")
    add = subparsers.add_parser(
        parents=[file_parser],
       help="Add a new flight"
    add.add_argument(
        "--dest",
```

```
required=True
              help="Flight destination"
           add.add_argument(
              "--flight_num",
              action="store"
          add.add_argument(
             action="store",
              required=True
              help="The airplane type"
          _ = subparsers.add_parser(
              "display"
              parents=[file_parser],
              help="Display all flights"
          select = subparsers.add_parser(
              parents=[file_parser],
          select.add_argument(
            "-T",
"--type",
              action="store",
              required=True
              help="The required type"
          args = parser.parse_args(command_line)
          # Поличить пить к файли базы данных.
          db_path = Path(args.db)
           create_db(db_path)
           if args.command = "add":
              add_flight(db_path, args.dest, args.flight_num, args.type)
          elif args.command = "display"
              display_flights(select_all(db_path))
           elif args.command = "select":
            display_flights(select_flights(db_path, args.type))
       if __name__ = "__main__":
PS C:\Users\student-09-525\PycharmProjects\OPI_9\Individual> python .\individual.py add --db="flights.db" --dest="Chicago" --flight_nu
m="N14G" --type="Military"
PS C:\Users\student-09-525\PycharmProjects\OPI_9\Individual> python .\individual.py add --db="flights.db" --dest="Chicago" --flight_nu
m="N146" --type="Passenger"
PS C:\Users\student-09-525\PycharmProjects\OPI_9\Individual> python .\individual.py add --db="flights.db" --dest="Chicago" --flight_nu
m="F1346" --type="Passenger
| No | Пункт назначения
                               | Номер рейса | Тип самолёта |
  1 | Miami
                                 M1245
                                                    | Passenger
+-----
                                             -------
                                 | EG436
                                                    | Sanitary
  2 | Egypt
                                 N146
   3 | Chicago
                                                    | Military
   4 | Chicago
                                | N14G
                                                   Passenger
  5 | Chicago
                                | F1346
                                                    | Passenger
PS C:\Users\student-09-525\PycharmProjects\OPI_9\Individual> python .\individual.py select --db="flights.db" --type="Passenger"
                               | Номер рейса
| No | Пункт назначения
                                                   I Тип самолёта I
+-----
  1 | Miami
                                 M1245
                                                    | Passenger
  2 | Chicago
                                N146
                                                    | Passenger
                                | F1346
  3 | Chicago
                                                   Passenger
PS C:\Users\student-09-525\PycharmProjects\OPI_9\Individual>
```

1. Каково назначение модуля sqlite3?

Непосредственно модуль sqlite3 — это 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 руthоn-программу. Модуль sqlite3 содержит много классов, функций и констант. Их перечень можно посмотреть с помощью функции dir().

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

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(). Вызов функции connect() приводит к созданию объекта-экземпляра от класса Connection. Этот объект обеспечивает связь с файлом базы данных, представляет конкретную БД в программе. 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 соединения. Для выполнения инструкций SQLite3 сначала устанавливается соединение, а затем создается объект курсора с использованием объекта соединения следующим образом

- 3. Как подключиться к базе данных SQLite3, находящейся в оперативной памяти компьютера? Мы также можем создать базу данных в оперативной памяти с помощью функции :memory: with the connect. Такая база данных называется базой данных в памяти.
- 4. Как корректно завершить работу с базой данных SQLite3? con.close() для высвобождения памяти.
- 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 (). Предположим, что мы хотим обновить имя сотрудника, чей идентификатор равен 2. Для обновления мы будем использовать оператор UPDATE, а для сотрудника, чей идентификатор равен 2. Мы будем использовать предложение WHERE в качестве условия для выбора этого сотрудника.

```
def sql_update(con):
    cursor_obj = con.cursor()
    cursor_obj.execute(
"UPDATE employees SET name = 'Rogers' where id = 2"
)
    con.commit()
```

- 7. Как осуществляется выборка данных из базы данных SQLite3? Оператор SELECT используется для выбора данных из определенной таблицы. Если вы хотите выбрать все столбцы данных из таблицы, вы можете использовать звездочку (*).
- 8. Каково назначение метода rowcount?
- SQLite3 rowcount используется для возврата количества строк, которые были затронуты или выбраны последним выполненным SQL-запросом. Когда мы используем rowcount с оператором SELECT, будет возвращено значение -1, поскольку количество выбранных строк неизвестно до тех пор, пока все они не будут извлечены.
- 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 . sqlite_master это главная таблица в SQLite3, которая хранит все таблицы.
- 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)
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.

```
cursor_obj.execute(
"CREATE TABLE IF NOT EXISTS projects(id INTEGER, name TEXT)"
)
data = [
(1, "Ridesharing"),
(2, "Water Purifying"),
(3, "Forensics"),
(4, "Botany")
]
cursor_obj.executemany("INSERT INTO projects VALUES (?, ?)", data)
con.commit()
```

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 .
cursor_obj.execute(
"""

CREATE TABLE IF NOT EXISTS assignments(
id INTEGER, name TEXT, date DATE
)
"""
)
data = [
(1, "Ridesharing", datetime.date(2017, 1, 2)),
(2, "Water Purifying", datetime.date(2018, 3, 4))
]
cursor_obj.executemany("INSERT INTO assignments VALUES(?, ?, ?)",
data)
con.commit()
```