Студент группы ИС-25 Дудков Д.Ю.

Практическое занятие № 15

Тема: составление программ для работы с базами данных в IDE PyCharm Community.

Цель: закрепить усвоенные знания, понятия, алгоритмы, основные принципы составления программ, работы с БД в IDE PyCharm Community.

Постановка задачи.
Приложение ЗАКАЗЫ ТОВАРОВ для автоматизированного контроля заказов торговой фирмы. Таблица Заказы должна содержать информацию о товарах со следующей структурой записи: Код товара, Наименование товара, Заказчик (наименование организации, возможны повторяющиеся значения), Дата заказа, Срок исполнения (от 1 до 10 дней), Стоимость заказа.

Текст программы:

```
import sqlite3
from random import randint, choice as random_choice



DB_FILE = './db/db.sqlite3'



class ProductOrdersDB:
    def __init__(self, db_file_path: str, print_errors: bool) -> None:
        self.db_file_path = db_file_path
        self.print_errors = print_errors

    def clear_db(self) -> bool:
        """Удаление всех таблиц из БД"""

        drop_order_table_command = """
            DROP TABLE IF EXISTS product_order
        """
        try:
            with sqlite3.connect(self.db_file_path) as connection:
                cursor = connection.cursor()
                cursor.execute(drop_order_table_command)
            return True
        except Exception as error:
            if self.print_errors:
                print(error)
            return False

    def create_tables(self) -> bool:
        """Создание всех таблиц в БД"""
```

```
    create_order_table_command = """
    CREATE TABLE IF NOT EXISTS product_order (
        id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
        code VARCHAR(30) NOT NULL UNIQUE,
        title VARCHAR(100) NOT NULL UNIQUE,
        buyer VARCHAR(150) NOT NULL,
        order_datetime TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
        perform_days INTEGER NOT NULL,
        price INTEGER NOT NULL
    );
    """
    try:
        with sqlite3.connect(self.db_file_path) as connection:
            cursor = connection.cursor()
            cursor.execute(create_order_table_command)
        return True
    except Exception as error:
        if self.print_errors:
            print(error)
        return False

def insert_order(self, code: str, title: str, buyer: str, perform_days: int, price: int) -> bool:
    """Добавление заказа в БД"""

    insert_order_command = f"""
    INSERT INTO product_order
        (code, title, buyer, perform_days, price)
        VALUES
        ("{code}", "{title}", "{buyer}", {perform_days}, {price});
    """
    try:
        with sqlite3.connect(self.db_file_path) as connection:
            cursor = connection.cursor()
            cursor.execute(insert_order_command)
            connection.commit()
        return True
    except Exception as error:
        if self.print_errors:
            print(error)
        return False

def delete_orders(self, field: str, value: str | int) -> bool:
    """Удаление заказов из БД по значению value поля field"""

    if field not in ['id', 'code', 'title', 'buyer', 'perform_days', 'price']:
        if self.print_errors:
            print(f'Поля с именем "{field}" нет в таблице "product_order"')
        return False

    value_to_command = f"'{value}'" if isinstance(value, str) else value
    delete_order_command = f"""
    DELETE FROM product_order
        WHERE {field} = {value_to_command};
    """
    try:
```

```
            with sqlite3.connect(self.db_file_path) as connection:
                cursor = connection.cursor()
                cursor.execute(delete_order_command)
                connection.commit()
            return True
        except Exception as error:
            if self.print_errors:
                print(error)
            return False

    def get_orders(self, *args) -> list | bool:
        """Получение заказов из БД по фильтрам *args (OR и/или AND указывать в фильтре)"""

        select_orders_command = "SELECT * FROM product_order;"
        if args:
            select_orders_command = select_orders_command[:-1] + f" WHERE {' '.join(args)};"

        try:
            with sqlite3.connect(self.db_file_path) as connection:
                cursor = connection.cursor()
                cursor.execute(select_orders_command)
                return cursor.fetchall()
        except Exception as error:
            if self.print_errors:
                print(error)
            return False

    def update_orders(self, where_filter: str, update_set: str) -> bool:
        """Обновление значений заказов update_set в БД по фильтру where_filter"""

        update_orders_command = f"UPDATE product_order SET {update_set} WHERE {where_filter};"

        try:
            with sqlite3.connect(self.db_file_path) as connection:
                cursor = connection.cursor()
                cursor.execute(update_orders_command)
                connection.commit()
            return True
        except Exception as error:
            if self.print_errors:
                print(error)
            return False


if __name__ == '__main__':
    db_model = ProductOrdersDB(db_file_path=DB_FILE, print_errors=True)

    db_model.clear_db()
    db_model.create_tables()

    for i in range(10):
```

```
        db_model.insert_order(
            code=f'58675699{i}',
            title=random_choice([f'Мыло {i}', f'Картофель {i}', f'Кресло офисное {i}']),
            buyer=random_choice(['ООО что-то', 'ПАО Ростелеком', 'ИП Дерево']),
            perform_days=randint(1, 10),
            price=randint(1000, 100000),
        )


    db_model.delete_orders(
        field='id',
        value=1
    )


    list_orders = db_model.get_orders()
    print(list_orders)
    list_orders_filtered = db_model.get_orders("title LIKE '%Картофель%' OR", "buyer =
'ПАО Ростелеком'")
    print(list_orders_filtered)


    print(db_model.get_orders("buyer = 'ИП Дерево'"))
    db_model.update_orders(where_filter="buyer = 'ИП Дерево'", update_set="price =
price + 1")
    print(db_model.get_orders("buyer = 'ИП Дерево'"))
```

Протокол работы программы:

all: [(1, '586756990', 'Мыло 0', 'ИП Дерево', '2024-04-05 19:51:10', 8, 30897), (2, '586756991', 'Мыло 1', 'ИП Дерево', '2024-04-05 19:51:10', 5, 34872), (3, '586756992', 'Картофель 2', 'ООО что-то', '2024-04-05 19:51:10', 10, 54687), (4, '586756993', 'Мыло 3', 'ПАО Ростелеком', '2024-04-05 19:51:10', 3, 17983), (5, '586756994', 'Кресло офисное 4', 'ПАО Ростелеком', '2024-04-05 19:51:10', 10, 82904), (6, '586756995', 'Картофель 5', 'ПАО Ростелеком', '2024-04-05 19:51:10', 2, 78103), (7, '586756996', 'Картофель 6', 'ИП Дерево', '2024-04-05 19:51:10', 2, 47615), (8, '586756997', 'Картофель 7', 'ООО что-то', '2024-04-05 19:51:10', 10, 47848), (9, '586756998', 'Мыло 8', 'ПАО Ростелеком', '2024-04-05 19:51:10', 2, 82568), (10, '586756999', 'Мыло 9', 'ПАО Ростелеком', '2024-04-05 19:51:10', 8, 20875)]

title LIKE '%Картофель%' OR buyer = 'ПАО Ростелеком': [(3, '586756992', 'Картофель 2', 'ООО что-то', '2024-04-05 19:51:10', 10, 54687), (4, '586756993', 'Мыло 3', 'ПАО Ростелеком', '2024-04-05 19:51:10', 3, 17983), (5, '586756994', 'Кресло офисное 4', 'ПАО Ростелеком', '2024-04-05 19:51:10', 10, 82904), (6, '586756995', 'Картофель 5', 'ПАО Ростелеком', '2024-04-05 19:51:10', 2, 78103), (7, '586756996', 'Картофель 6', 'ИП Дерево', '2024-04-05 19:51:10', 2, 47615), (8, '586756997', 'Картофель 7', 'ООО что-то', '2024-04-05 19:51:10', 10, 47848), (9,

'586756998', 'Мыло 8', 'ПАО Ростелеком', '2024-04-05 19:51:10', 2, 82568), (10, '586756999', 'Мыло 9', 'ПАО Ростелеком', '2024-04-05 19:51:10', 8, 20875)]

buyer = 'ИП Дерево' (before update): [(1, '586756990', 'Мыло 0', 'ИП Дерево', '2024-04-05 19:51:10', 8, 30897), (2, '586756991', 'Мыло 1', 'ИП Дерево', '2024-04-05 19:51:10', 5, 34872), (7, '586756996', 'Картофель 6', 'ИП Дерево', '2024-04-05 19:51:10', 2, 47615)]

buyer = 'ИП Дерево' (after update): [(1, '586756990', 'Мыло 0', 'ИП Дерево', '2024-04-05 19:51:10', 8, 30898), (2, '586756991', 'Мыло 1', 'ИП Дерево', '2024-04-05 19:51:10', 5, 34873), (7, '586756996', 'Картофель 6', 'ИП Дерево', '2024-04-05 19:51:10', 2, 47616)]

Process finished with exit code 0

Студент группы ИС-25 Дудков Д.Ю.

Постановка задачи.

Тип алгоритма: ветвление.

Блок-схема алгоритма:

Студент группы ИС-25 Дудков Д.Ю.

Текст программы:

Протокол работы программы:

Process finished with exit code 0

Вывод: в процессе выполнения практического занятия закрепил усвоенные знания, понятия, алгоритмы, основные принципы составления программ, приобрёл навыки составления программ ветвящейся структуры в IDE PyCharm Community. Были использованы языковые конструкции def, if.
Выполнены разработка кода, отладка, тестирование, оптимизация программного кода.
Готовые программные коды выложены на GitHub.