МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение

высшего образования

**«Северный (Арктический) федеральный университет имени М.В. Ломоносова»**

**технологический колледж императора петра i**

**ОТЧЁТ**

**по лабораторным работам**

|  |  |  |
| --- | --- | --- |
| По дисциплине/междисциплинарному курсу | | 01.01 Разработка программных модулей |
|  | | |
|  | | |
|  |
|  | | |

|  |  |
| --- | --- |
|  | Выполнил (-а) обучающийся (-аяся):  Зимин Данила Викторович |
|  | (Ф.И.О.) |
|  | Специальность:  09.02.07 Информационные системы и программирование |
|  | (код и наименование) |
|  | Курс: 3 |
|  | Группа: 404017 |
|  | Преподаватель:  Воронцов Роман Антонович |
|  | (Ф.И.О.) |

Архангельск 2022

**Лабораторная работа №1**

Работа с классами

**Цель работы**

Научиться работать с классами.

**Ход работы**

**Вариант 3**

**Задание 1.** Создать класс объект кошки Cat, который будет принимать два аргумента, три метода по примеру выше и создать 2-3 экземпляра класса.

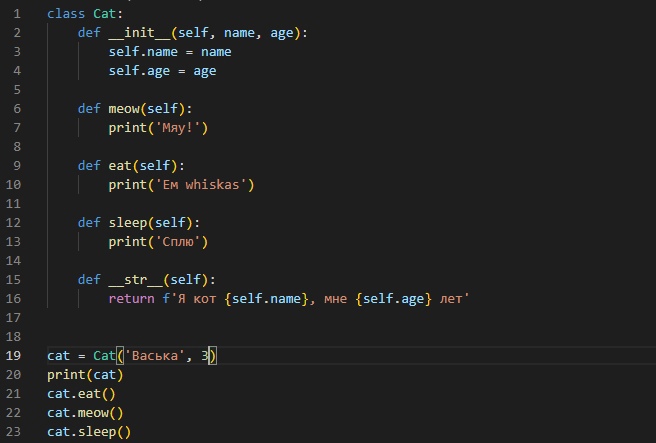
****

Рисунок 1 – Создание класса Cat

**Задание 2.** Решить задачу определения високосный год или нет с применением класса.

****

Рисунок 2 – Вычисление високосного года

**Задание 3.** Создать объект класс Car, который будет принимать несколько аргументов и 3 метода, один из которых будет считать транспортный налог.

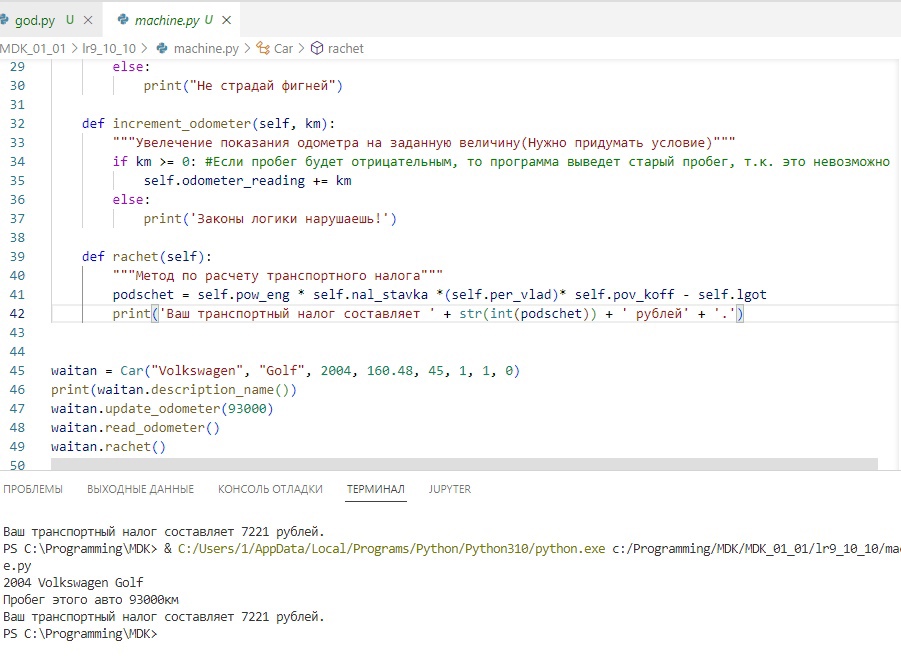
****

Рисунок 3 – Создание класса Car

**Контрольные вопросы**

*1.Что представляет собой класс?*

Класс – в объектно-ориентированном программировании, представляет собой шаблон для создания объектов, обеспечивающий начальные значения состояний: инициализация полей-переменных и реализации поведения функций или методов.

*2. Как создаются новые классы?*

Классы позволяют моделировать практически все что угодно. Начнём с написания простого класса Dog, представляющего собаку – не какую-то конкретную, а собаку.

**Вывод**

На данной лабораторной работе я научился работать с классами.

**Лабораторная работа №2**

Перезагрузка методов

**Цель работы**

Научиться работать с перезагрузкой методов.

**Ход работы**

**Вариант 7**

**Задание 1.** Выполняем перезагрузку эффективным методом с помощью декоратора множественной отправки.

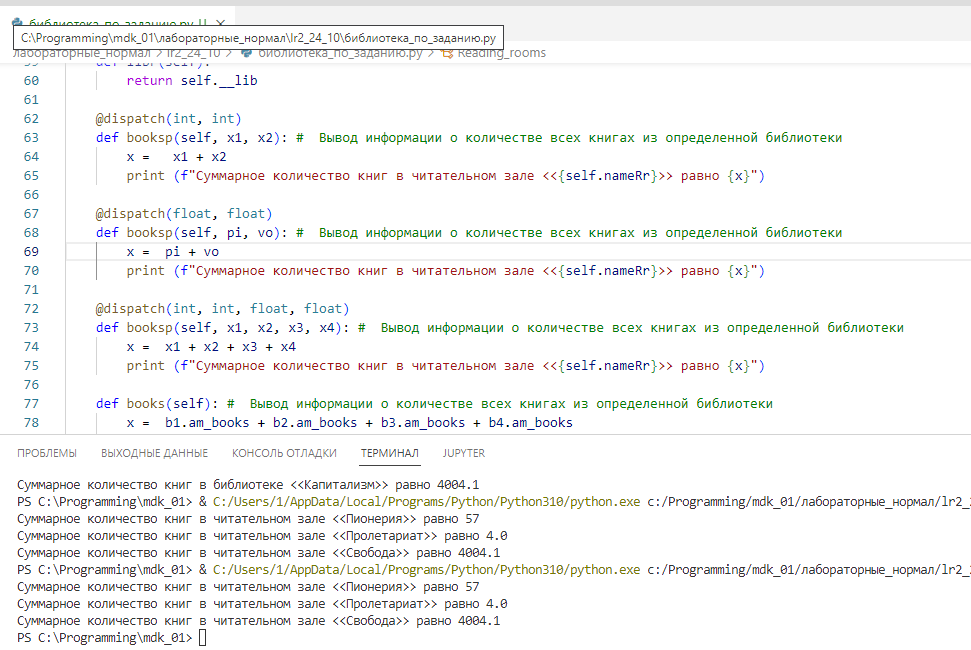


Рисунок 1 – Использование декоратора множественной отправки

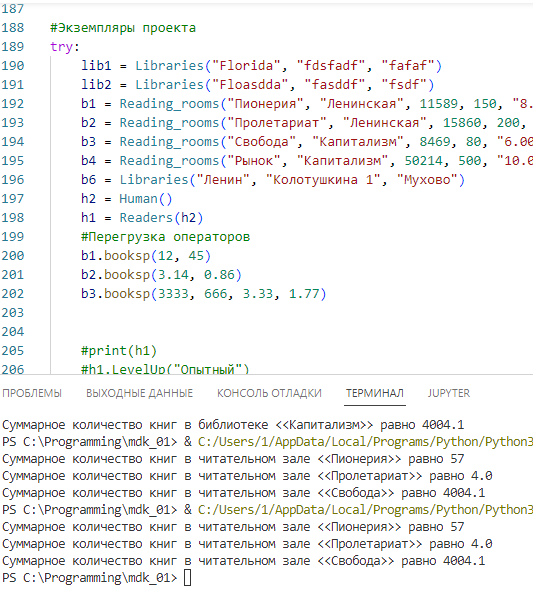


Рисунок 2 – Перегрузка методов с помощью декоратора

**Контрольные вопросы**

*1.Что такое перезагрузка методов?*

Перегрузка методов — это приём программирования, который позволяет разработчику в одном классе для методов с разными параметрами использовать одно и то же имя.

*2.Что такое перезагрузка операторов?*

Перегрузка операторов — один из способов реализации полиморфизма, заключающийся в возможности одновременного существования в одной области видимости нескольких различных вариантов применения операторов, имеющих одно и то же имя, но различающихся типами параметров, к которым они применяются.

**Вывод**

На данной лабораторной работе я научился работать с методами перезагрузки в Python.

**Лабораторная работа №3**

Определение операций в классе

**Цель работы**

Научиться работать в классе.

**Ход работы**

Вариант 7

**Задание 1.** Перезагрузка операций. Документирование. ООП.

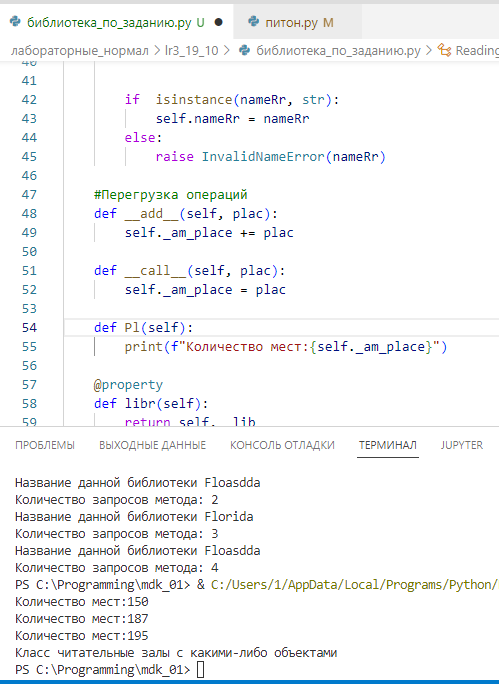


Рисунок 1 – Использование перегрузки операций

**Задание 2.** Ассоциация. Метапрограммирование.

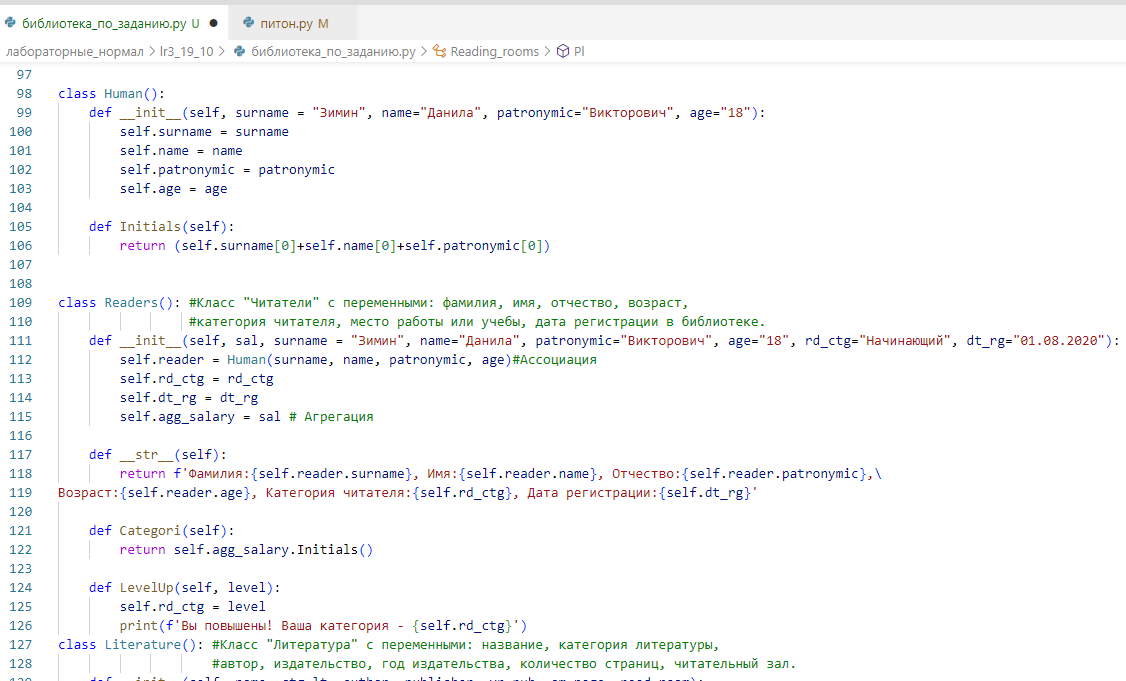


Рисунок 2 – Создание ассоциации

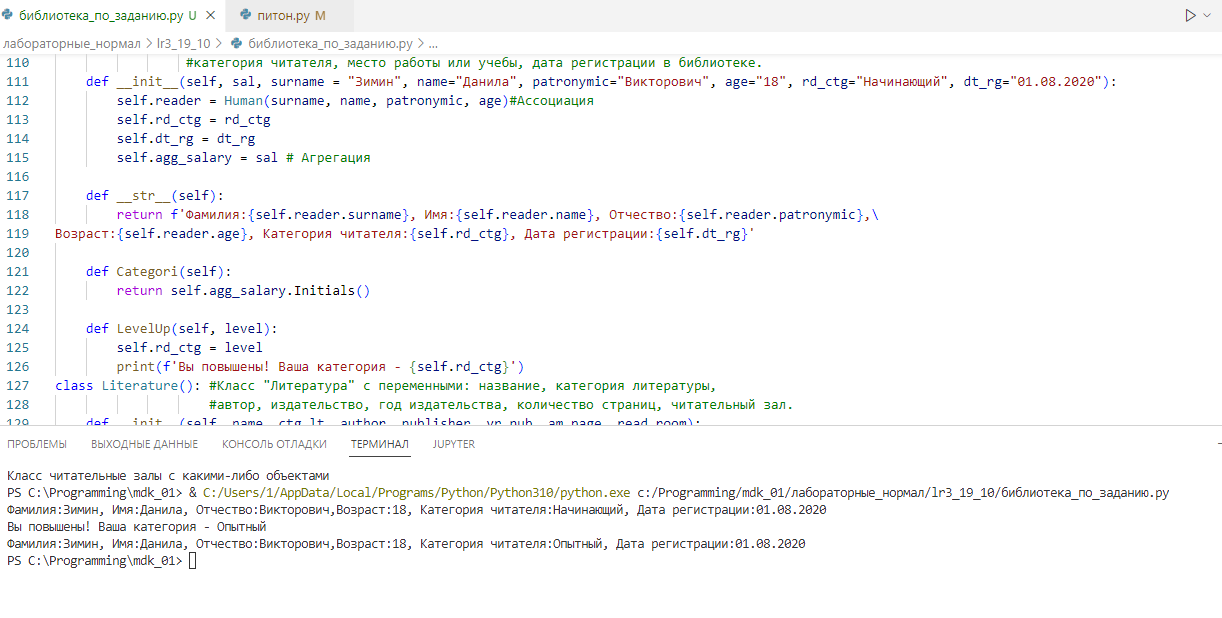


Рисунок 3 – Использование ассоциации

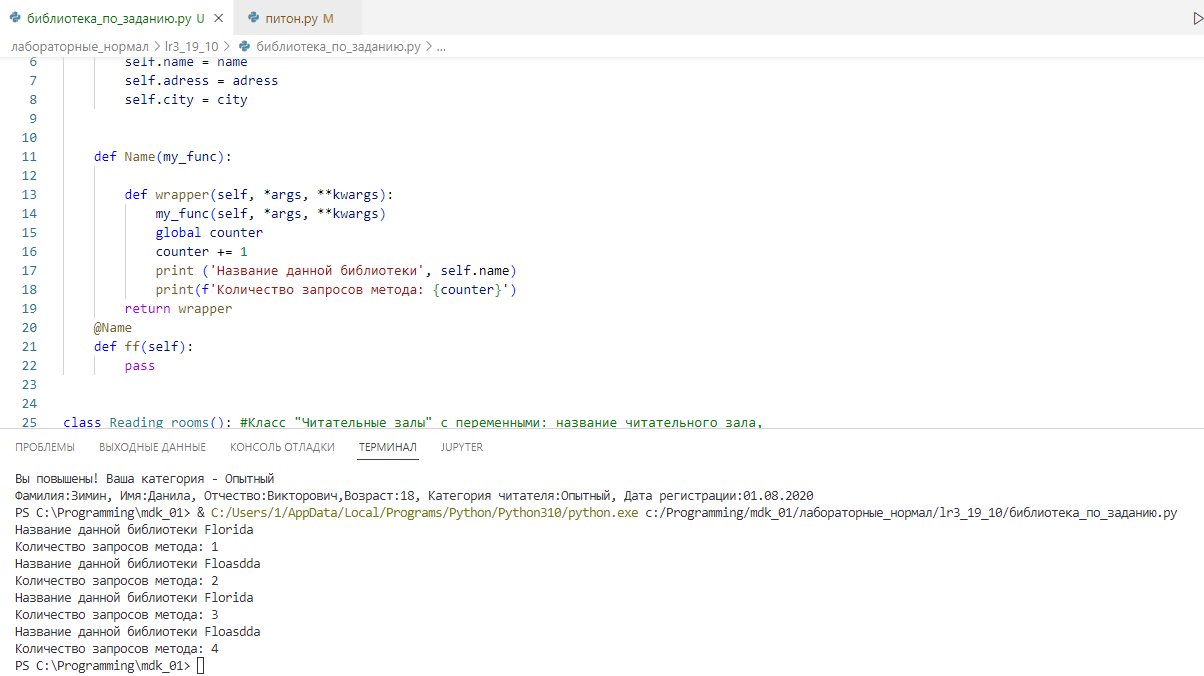


Рисунок 4 – Создание и использование декоратора

**Задание 3.** Агрегация. Контейнеры. Итераторы. ООП.

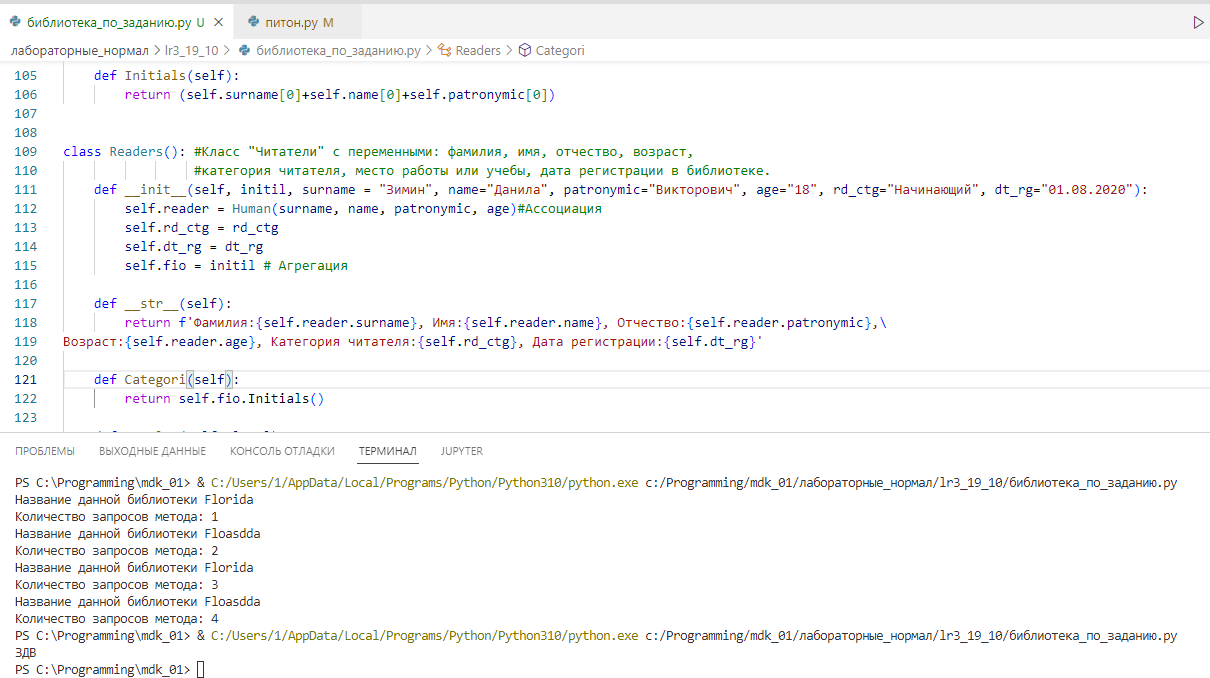


Рисунок 5 – Создание и использование агрегации

**Контрольные вопросы**

1. *Что такое конструктор?*

Конструктор — это особый тип метода (функции), который используется для инициализации членов экземпляра класса.

**Вывод**

На данной лабораторной работе я научился работать в классе.

**Лабораторная работа №4**

Создание наследованных классов

**Цель работы**

Научиться работать с наследованными классами.

**Ход работы**

Вариант 7. Создание класса по предметной области «Библиотеки города».

**Задание 1.** Конструктор. ООП.

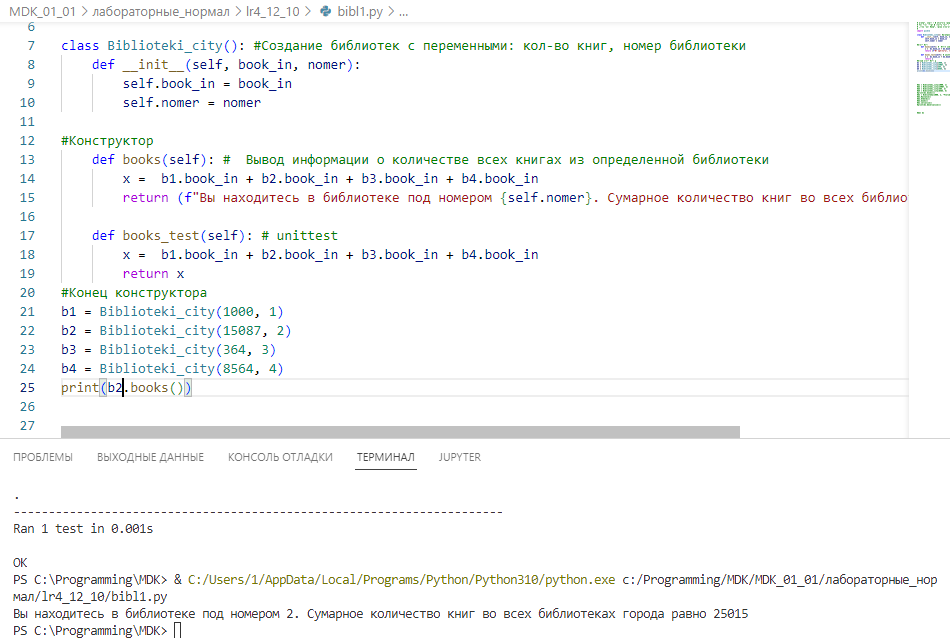


Рисунок 1 – Создание конструктора

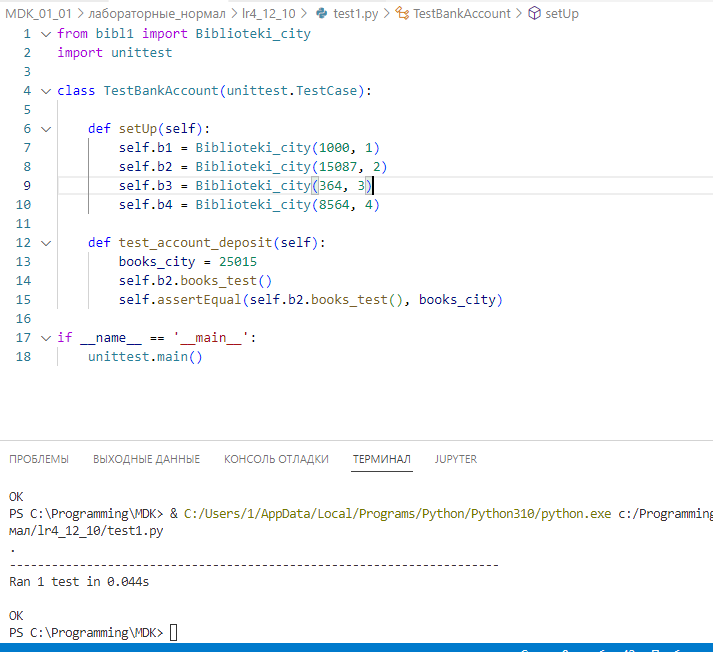


Рисунок 2 – Проведение unit-теста

**Задание 2.** Деструктор. ООП.

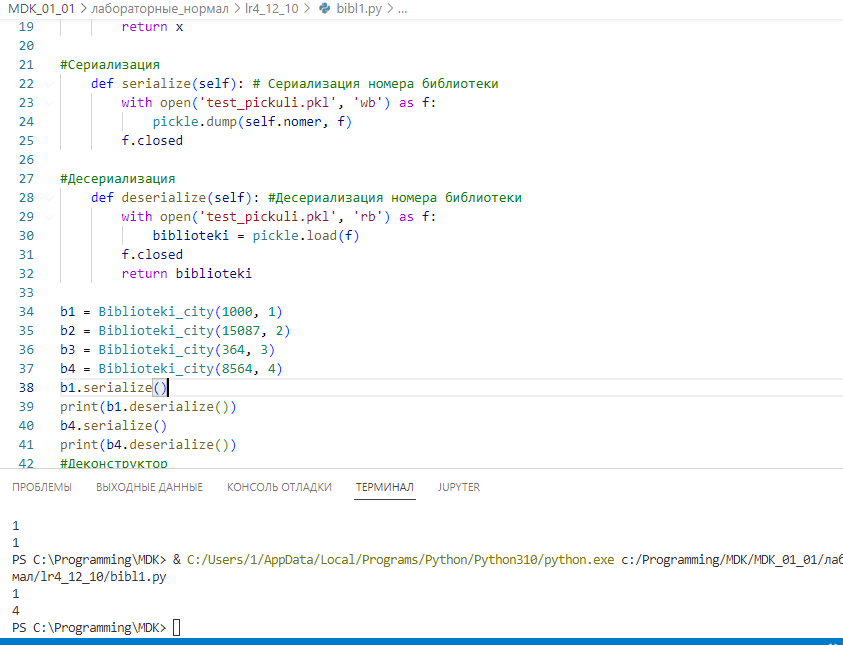


Рисунок 3 – Сериализация

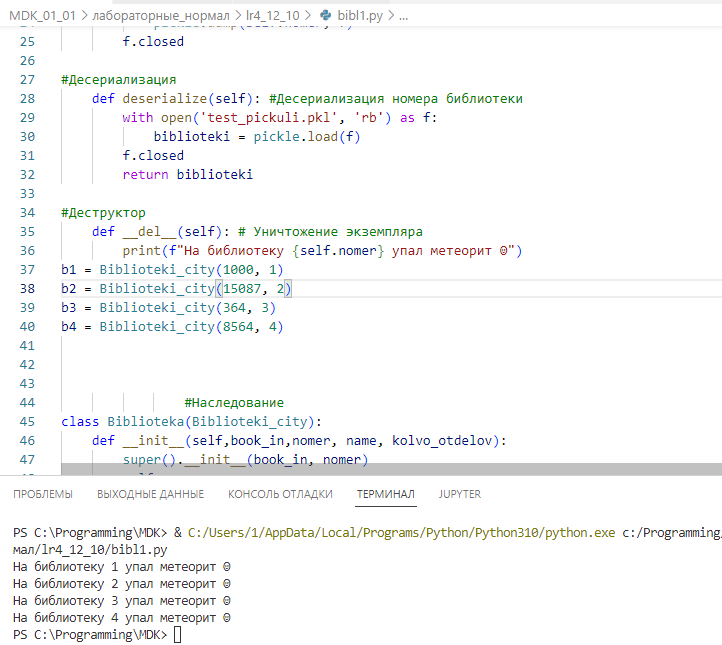


Рисунок 4 – Создание деструктора

**Задание 3.** Наследование. Полиморфизм. ООП.

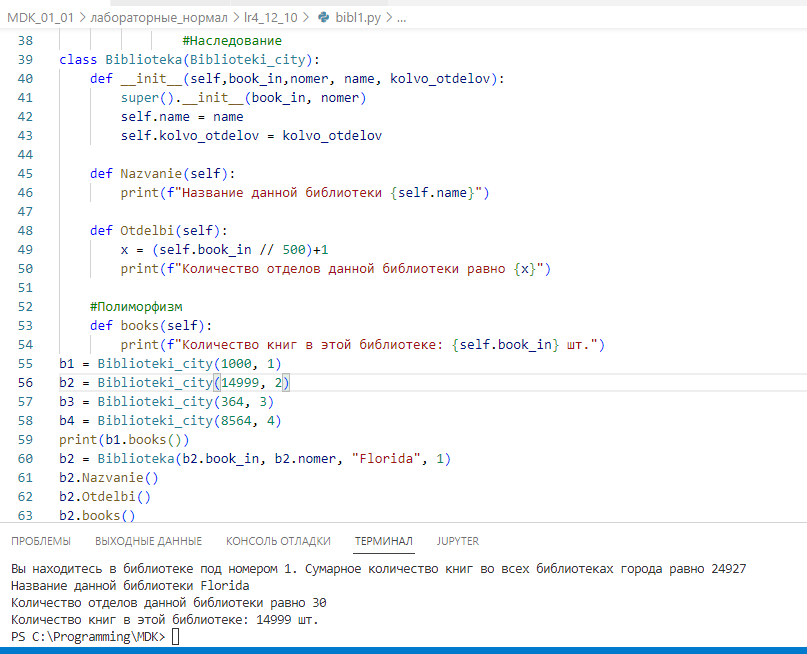


Рисунок 5 – Наследование и полиморфизм

**Контрольные вопросы**

1. *Что такое наследование?*

Наследование — это механизм создания новых классов, призванных настроить или изменить поведение существующего класса.

**Вывод**

На данной лабораторной работе я научился работать с наследованными классами.

**Лабораторная работа №5**

Работа с объектами через интерфейсы. Использование стандартных интерфейсов **Цель работы**

Научиться работать с интерфейсами.

**Ход работы**

**Вариант 7**

**Задание.** Реализуйте интерфейс согласно варианта вашего индивидуального задания.

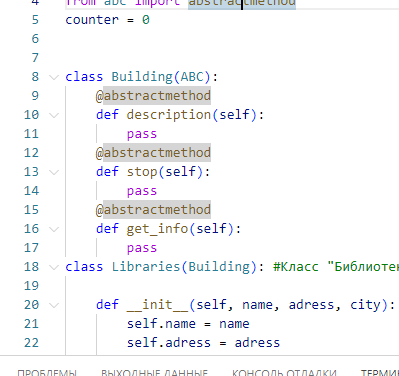


Рисунок 1 – Создание абстрактного класса

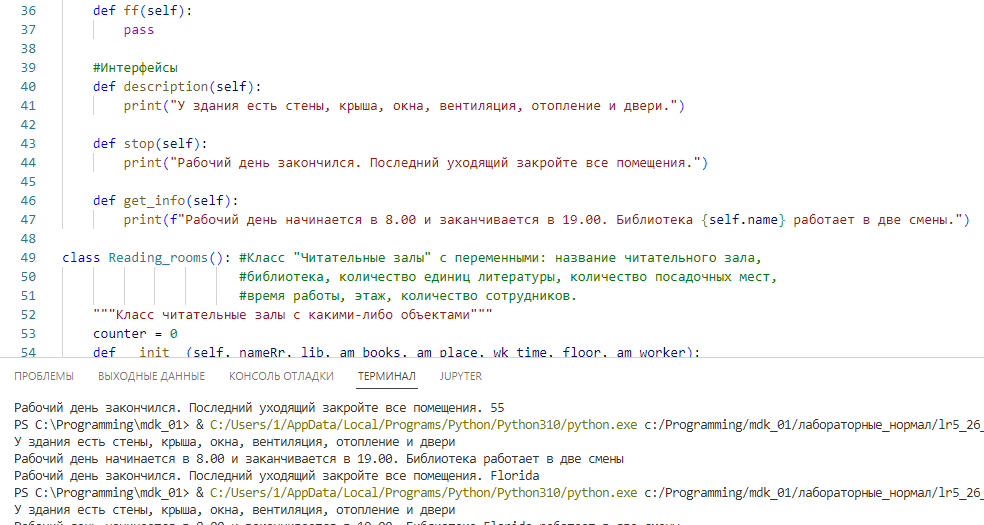


Рисунок 2 – Создание интерфейса

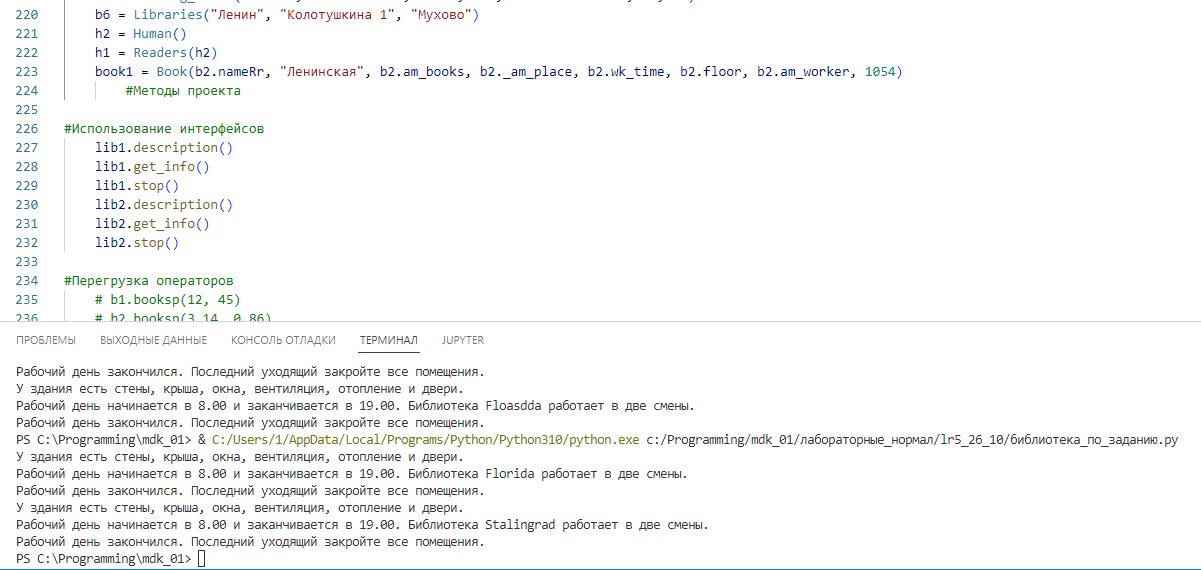


Рисунок 3 – Использование интерфейса

**Контрольные вопросы**

*1.Что такое интерфейс?*

Интерфейс — набор функций (имена и их сигнатуры, то есть количество и типы входящих параметров, а также возвращаемое значение), не зависящих от конкретной реализации.

**Вывод**

На данной лабораторной работе я научился работать с интерфейсами.

**Лабораторная работа №6**

Работа с типом данных структура

**Цель работы**

Научиться работать с типом данных структуры.

**Ход работы**

**Вариант 7**

**Задание.** Реализуйте минимум один датакласс согласно варианта вашего индивидуального задания.

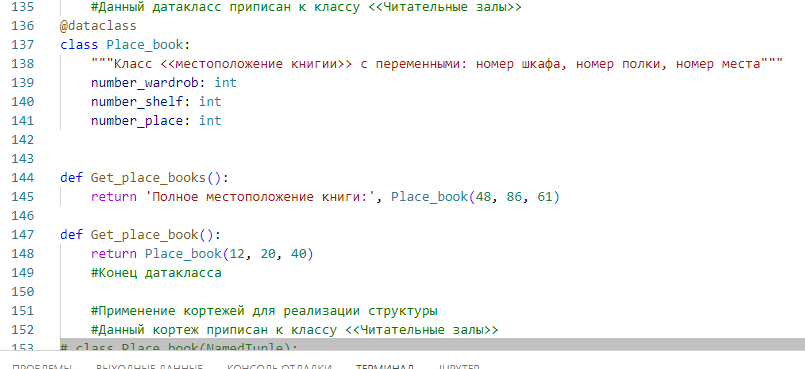


Рисунок 1 – Объявление структуры

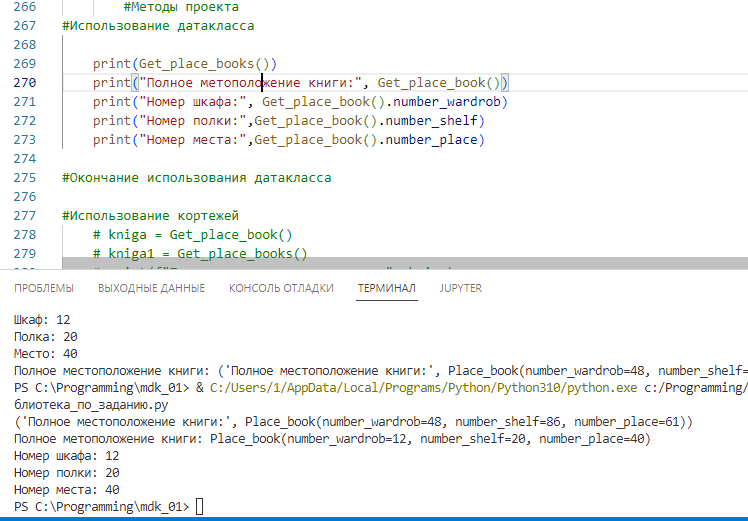


Рисунок 2 – Использование структуры

**Контрольные вопросы**

*1.Что такое структура?*

Структура – это совокупность переменных, объединенных одним именем, предоставляющая общепринятый способ совместного хранения информации.

*2. Методы задания структур.*

Существует три метода задания структуры:

* Именованные кортежи: namedtuple из библиотеки collections или NamedTuple из библиотеки typing;
* Датаклассы;
* Типизированный словарь: TypeDict из библиотеки typing.

Чаще всего используются датаклассы и кортежи.

**Вывод**

На данной лабораторной работе я научился работать с типом данных структуры.

**Лабораторная работа №7**

Коллекции. Параметризованные классы

**Цель работы**

Научиться работать с коллекциями. Параметризованными классами.

**Ход работы**

**Вариант 7**

**Задание.** Продемонстрируйте реализацию структур данных согласно варианта вашего индивидуального задания.



Рисунок 1 – Использование структуры «namedtuple»



Рисунок 2 – Использование структуры «defaultdict»

**Контрольные вопросы**

*1.Основные виды коллекций?*

* Список (list);
* Кортеж (tuple);
* Строка (string);
* Множество (set);
* Неизменяемое множество (frozenset);
* Словарь (dict).

*2. Применение параметризированных классов.*

Параметризированные классы позволяют использовать более гибкую и в то же время достаточно строгую типизацию, что особенно важно при работе с коллекциями.

**Вывод**

На данной лабораторной работе я научился работать с коллекциями и параметризированными классами.

**Лабораторная работа №7.1**

Использование основных шаблонов

**Цель работы**

Изучить основные шаблоны.

**Ход работы**

**Вариант 7**

**Задание 1.** Продемонстрируйте реализацию обобщенного класса согласно варианту вашего индивидуального задания.

**Задание 2.** Попробуйте установить средство проверки статических типов mypy и запустите проверку вашего кода.



Рисунок 1 – Реализация обобщенного класса

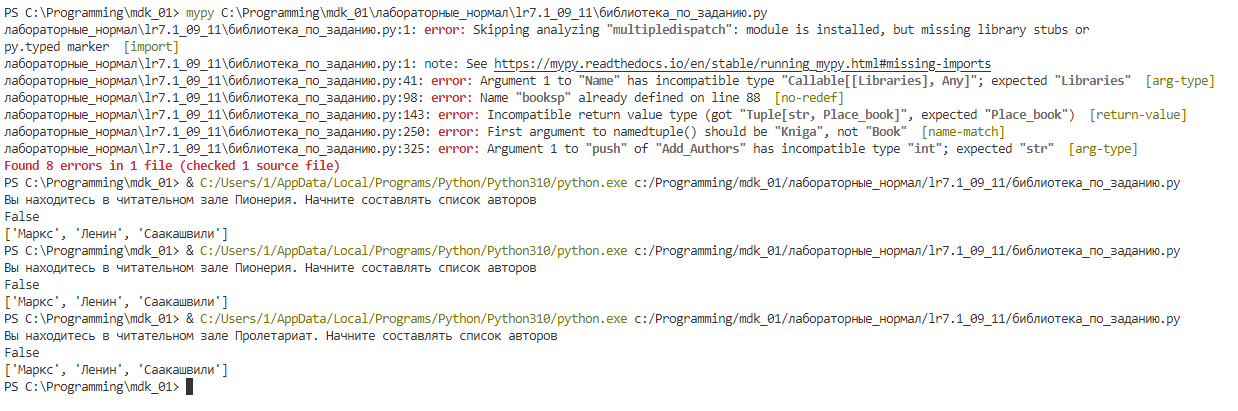


Рисунок 2 – Использование обобщенного класса и проверки mypy

**Контрольные вопросы**

*1.Как вы понимаете шаблонные классы?*

Я понимаю шаблонные классы как структура, в которой можно использовать разные типы данных.

*2. Как реализованы шаблонные классы в языке Python?*

Шаблонные классы в языке Python реализованы при помощи библиотеки typing.

**Вывод**

На данной лабораторной работе я изучил основные шаблоны.

**Лабораторная работа №8**

Использование регулярных выражений. Операции со списками

**Цель работы**

Научиться использовать регулярные выражения.

**Ход работы**

**Вариант 7**

**Задание 1.** Реализовать класс входа в систему с проверкой пароля в рамках вашего индивидуального задания.

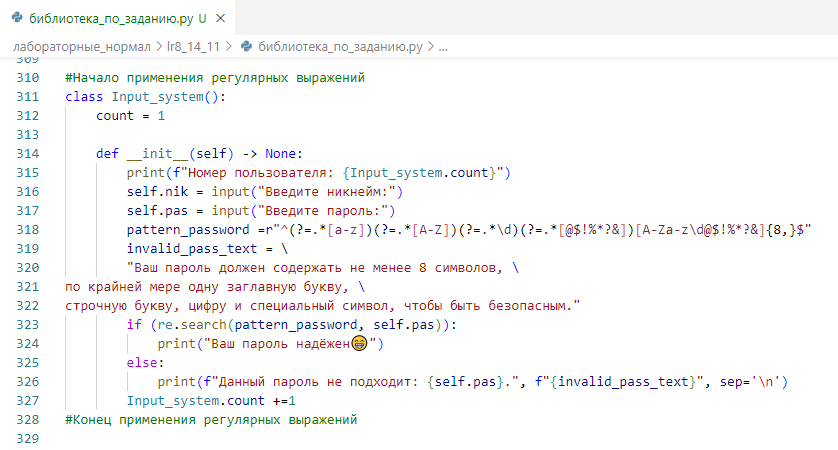


Рисунок 1 – Реализация применения регулярных выражений

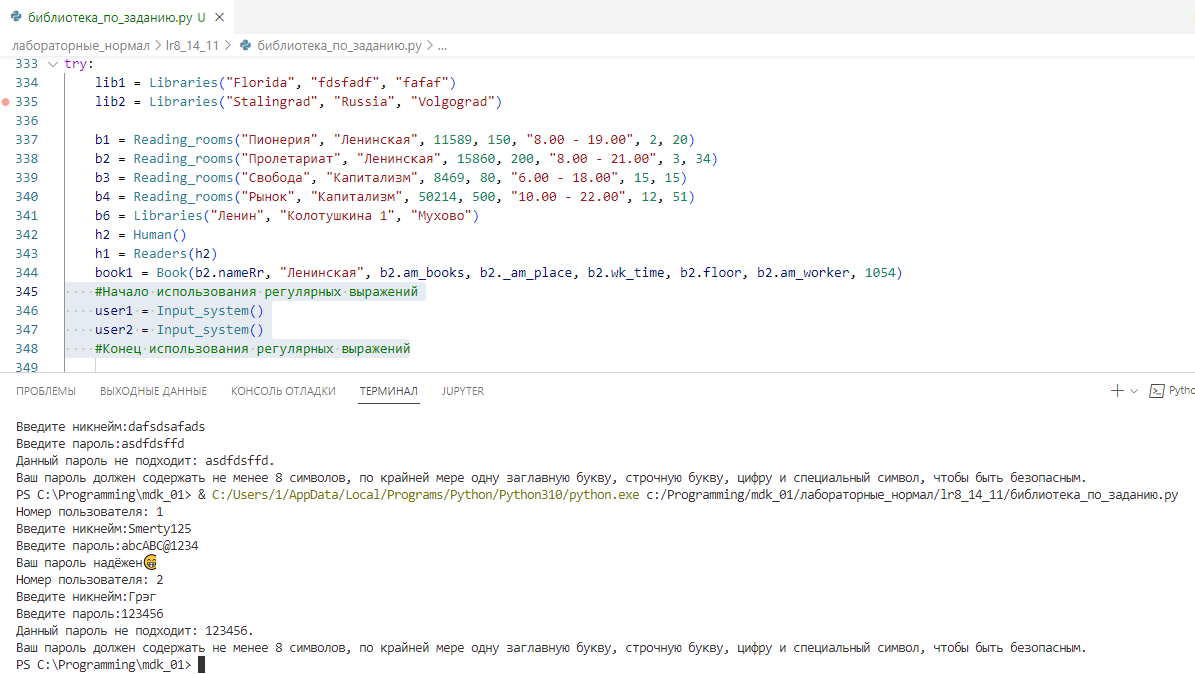


Рисунок 2 – Использование регулярных выражений

**Контрольные вопросы**

*1.* *Основные методы для работы с регулярными выражениями?*

* re.search (pattern, string);
* re.fullmatch (pattern, string);
* re.split (pattern, string, maxsplit=0);
* re.findall (pattern, string);
* re.finditer (pattern, string);
* re.sub (pattern, repl, string, count=0).

*2. Экранирование спец символов?*

Экранирование спец символов происходит с помощью постановки обратного слэша (**\**) перед символом, который необходимо экранировать.

**Вывод**

На данной лабораторной работе я научился использовать регулярные выражения.

**Лабораторная работа №9**

Инкапсуляция

**Цель работы**

Научиться работать с механизмом сокрытия компонентов программы.

**Ход работы**

**Задание 1.** Решите задания из приложенного файла ООП. Задание 5. Инкапсуляция. Исключения.

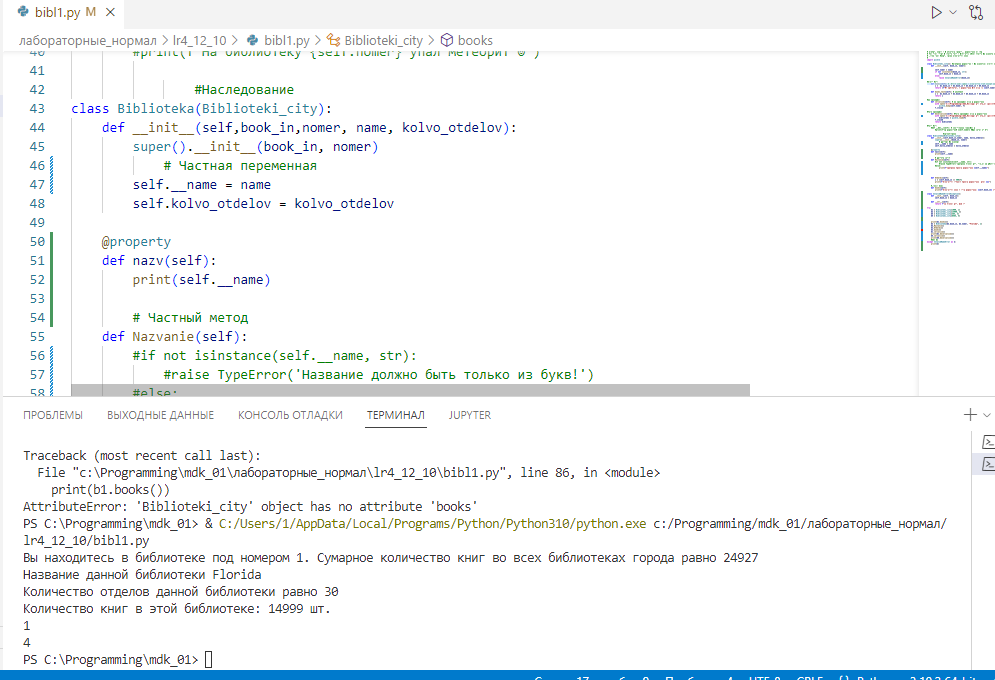


Рисунок 1 – Создание частной переменной

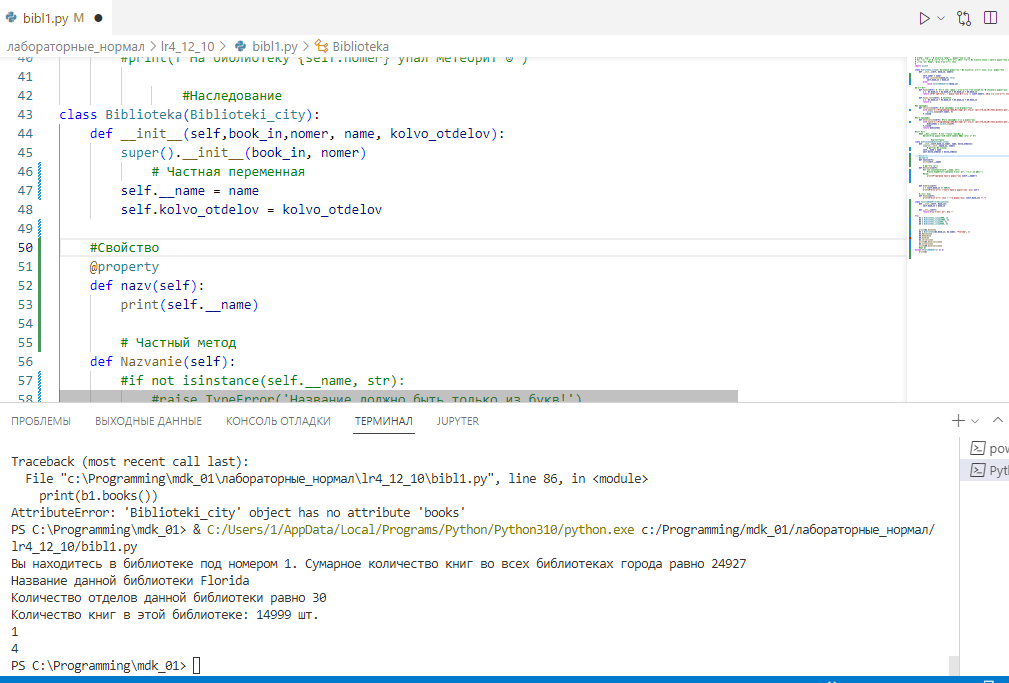


Рисунок 2 – Использование свойств



Рисунок 3 – Условия для возникновения исключения

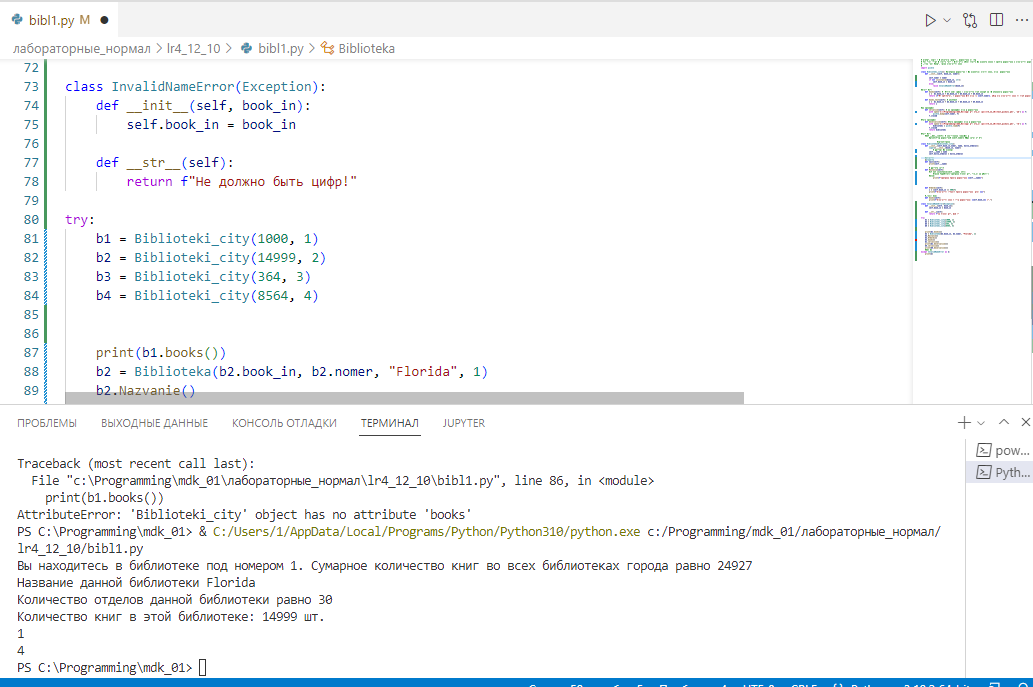


Рисунок 4 – Класс для возникновения исключения

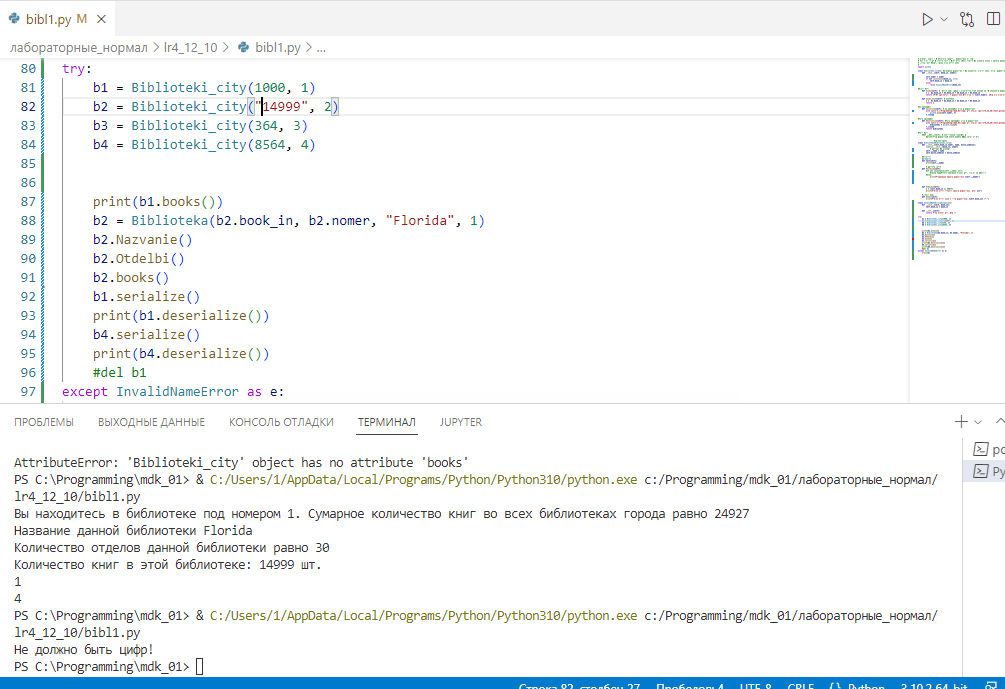


Рисунок 5 – Возникновение исключения

**Контрольные вопросы**

1. *Что представляет собой инкапсуляция?*

Инкапсуляция – это свойство системы, позволяющее объединить данные и методы, работающие с ними, в классе и скрыть детали реализации от пользователя.

1. *Какие уровни сокрытия можно выделить в языках программирования; в чем их различия?*

* Public

Публичный режим доступа.

* Protected

Режим доступа. Служит для обращения внутри класса и во всех его дочерних классах.

* Private

Режим доступа. Служит для обращения только внутри класса.

* *Каким образом инкапсуляция реализована в языке Python?*

Инкапсуляция делает некоторые из компонент доступными только внутри класса. Инкапсуляция в Python работает лишь на уровне соглашения между программистами о том, какие атрибуты являются общедоступными, а какие — внутренними.

* *Каким образом инкапсуляция реализована в языке C#?*

C# обеспечивает полную поддержку объектно-ориентированного программирования, включая абстракцию, инкапсуляцию, наследование и полиморфизм.

**Вывод**

На данной лабораторной работе я научился работать с механизмом сокрытия компонентов программы.

**Лабораторная работа №10**

Использование порождающих шаблонов

**Цель работы**

Изучить порождающие шаблоны.

**Ход работы**

**Задание.** Выполнить задания из приложенных файлов «Одиночка» и «Фабричный метод».



Рисунок 1 – Реализация «Singleton» пример 1

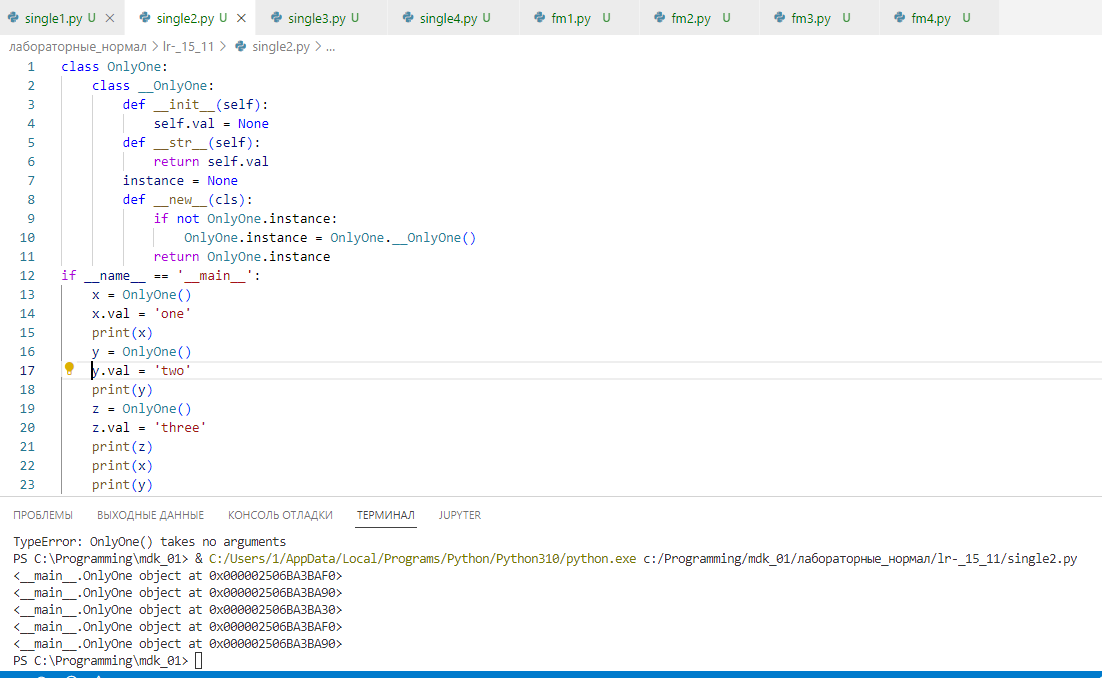


Рисунок 2 – Реализация «Singleton» пример 2



Рисунок 3 – Реализация «Singleton» пример 3

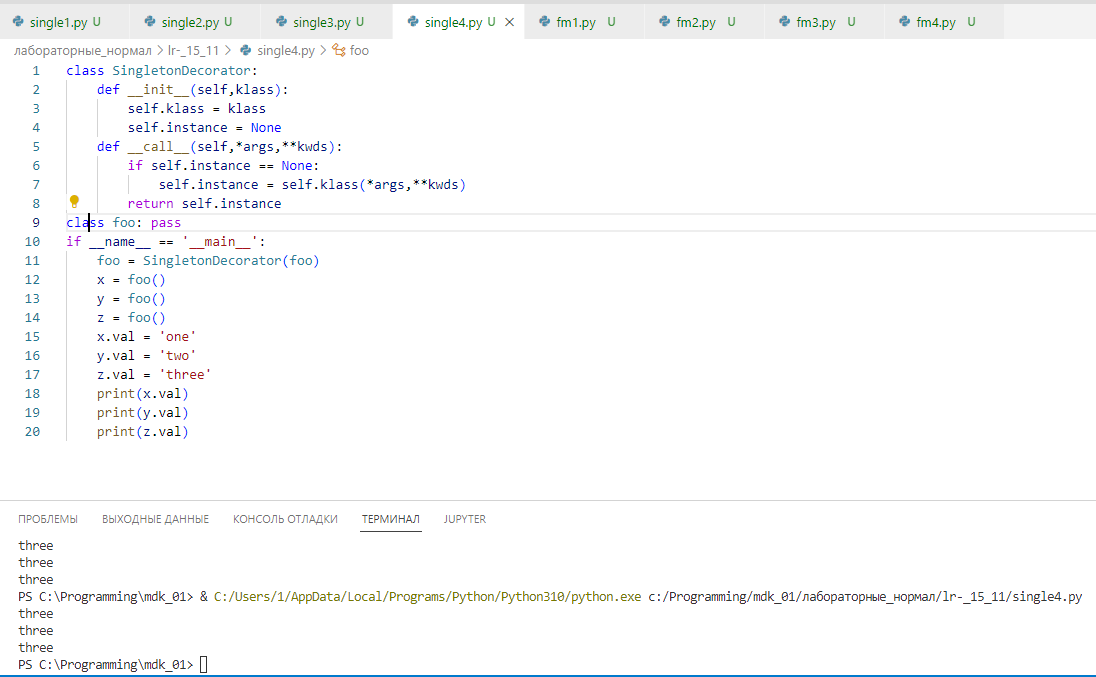


Рисунок 4 – Реализация «Singleton» пример 4



Рисунок 5 – Неудачная реализация «Singleton» пример 5

Результатом неудачи, изображенного на рисунке 5, стал устаревший пример. То есть данный код был реализован на Python 2.2, в связи с чем было принято решение пропустить выполнение данного примера.

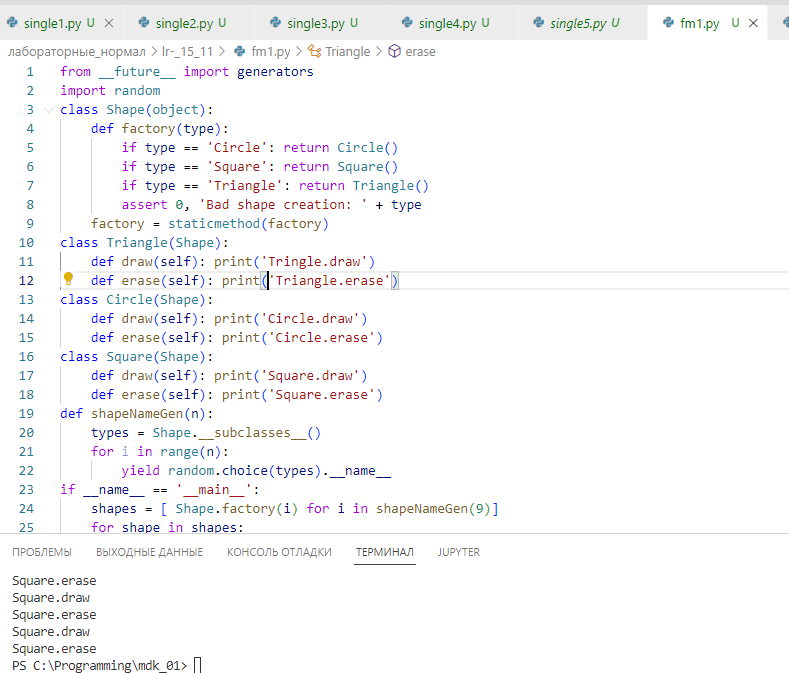


Рисунок 6 – Реализация «Factory method» пример 1 классом «Triangle»

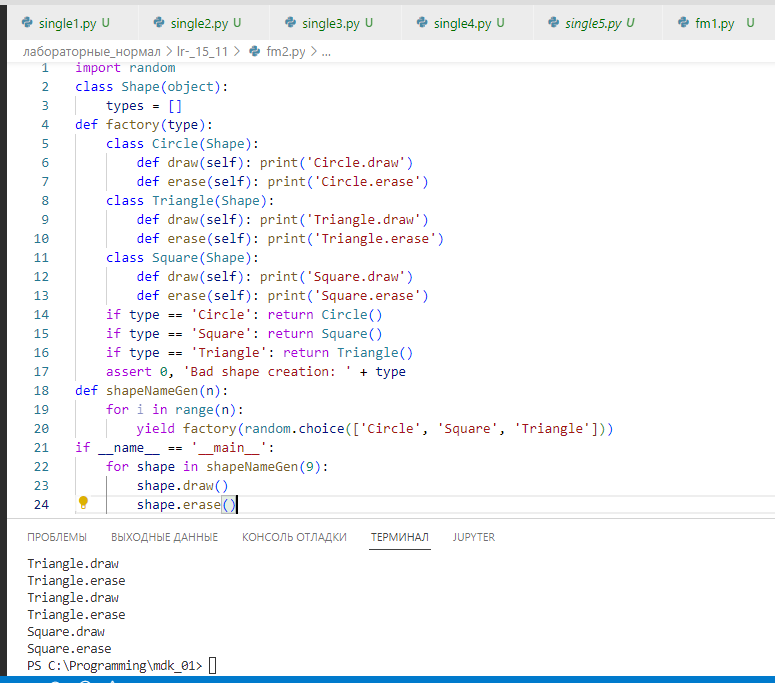


Рисунок 7 – Реализация «Factory method» пример 2 с классом «Triangle»

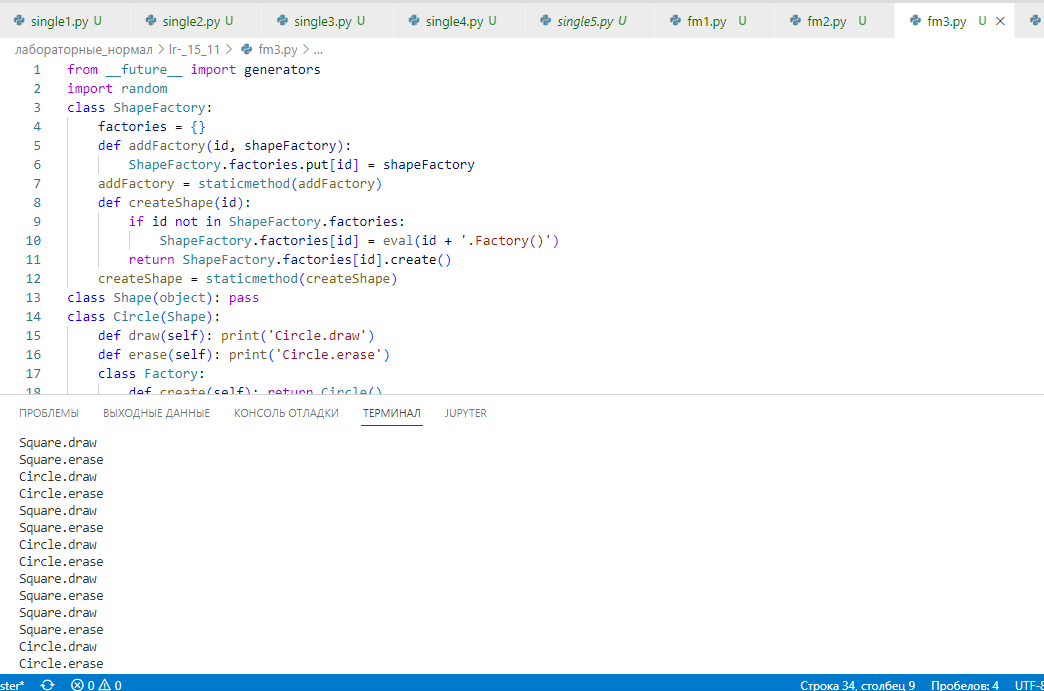


Рисунок 8 – Реализация «Factory method» пример 3

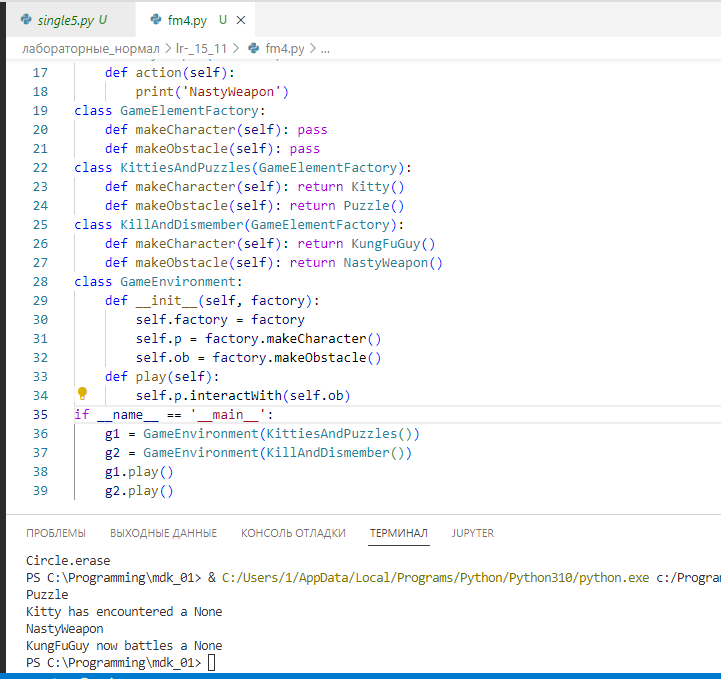


Рисунок 9 – Реализация «Factory method» пример 4

**Контрольные вопросы**

*1.Виды порождающих шаблонов?*

Существуют следующие виды порождающих шаблонов:

* Фабричный метод (Factory Method);
* Абстрактная фабрика (Abstract Factory);
* Строитель (Builder);
* Прототип (Prototype);
* Одиночка (Singleton).

*2.Функции порождающих шаблонов?*

|  |  |
| --- | --- |
| Вид порождающего шаблона | Функция порождающего шаблона |
| Фабричный метод (Factory Method) | Определяет общий интерфейс для создания объектов в  суперклассе, позволяя подклассам изменять тип создаваемых  объектов |
| Абстрактная фабрика (Abstract Factory) | Позволяет создавать семейства связанных объектов, не  привязываясь к конкретным классам создаваемых объектов |
| Строитель (Builder) | Позволяет создавать сложные объекты пошагово. Строитель  даёт возможность использовать один и тот же код  строительства для получения разных представлений объектов. |
| Прототип (Prototype) | Позволяет копировать объекты, не вдаваясь в подробности их  реализации. |
| Одиночка (Singleton) | Гарантирует, что у класса есть только один экземпляр, и  предоставляет к нему глобальную точку доступа. |

**Вывод**

На данной лабораторной работе я изучил порождающие шаблоны.

**Лабораторная работа №11**

Использование структурных шаблонов

**Цель работы**

Изучить структурные шаблоны.

**Ход работы**

**Задание.** Выполнить задания из приложенных файлов «Адаптер» и «Декоратор».

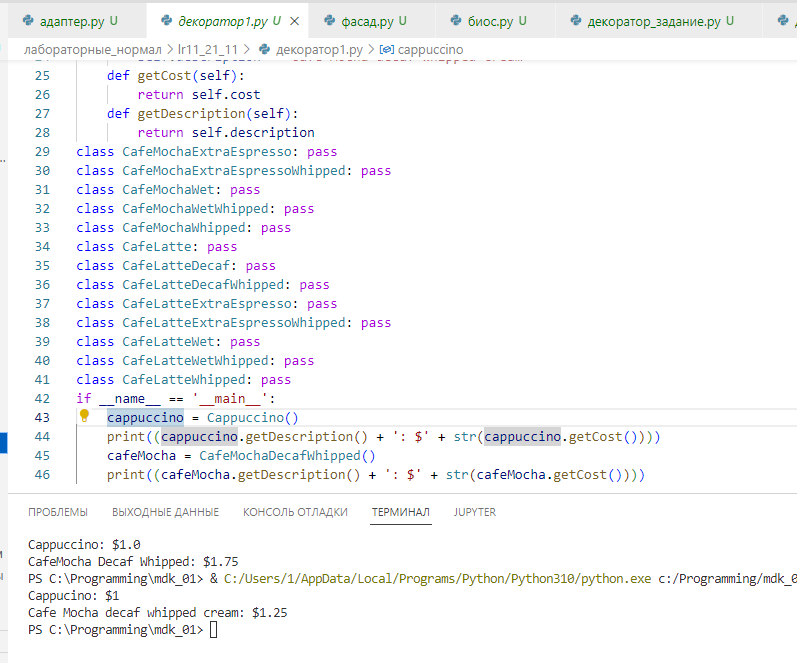


Рисунок 1 – Реализация декоратора пример 1

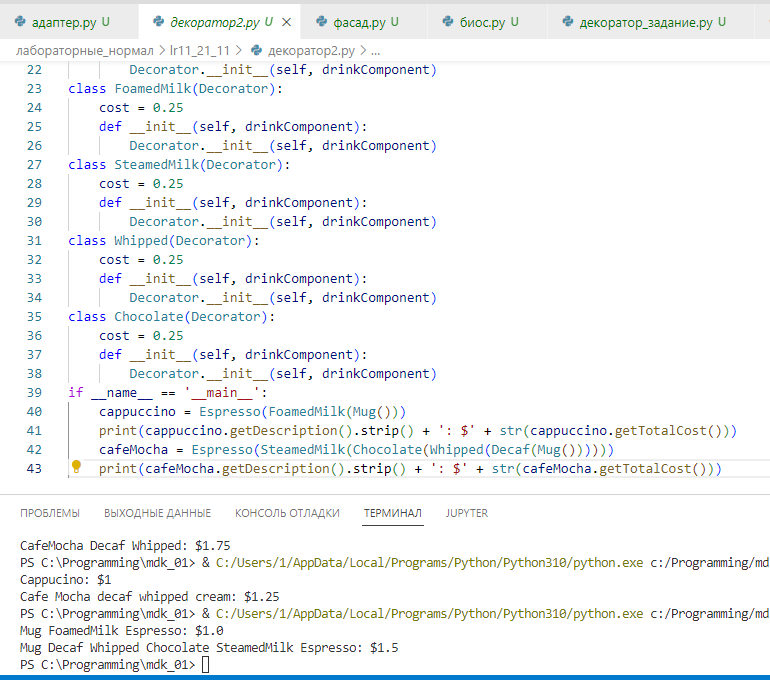


Рисунок 2 – Реализация декоратора пример 2

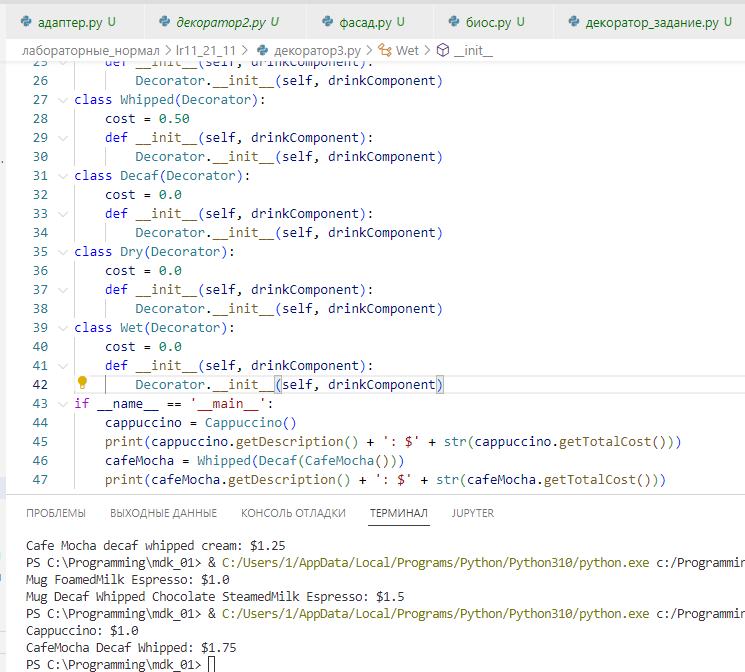


Рисунок 3 – Реализация декоратора пример 3

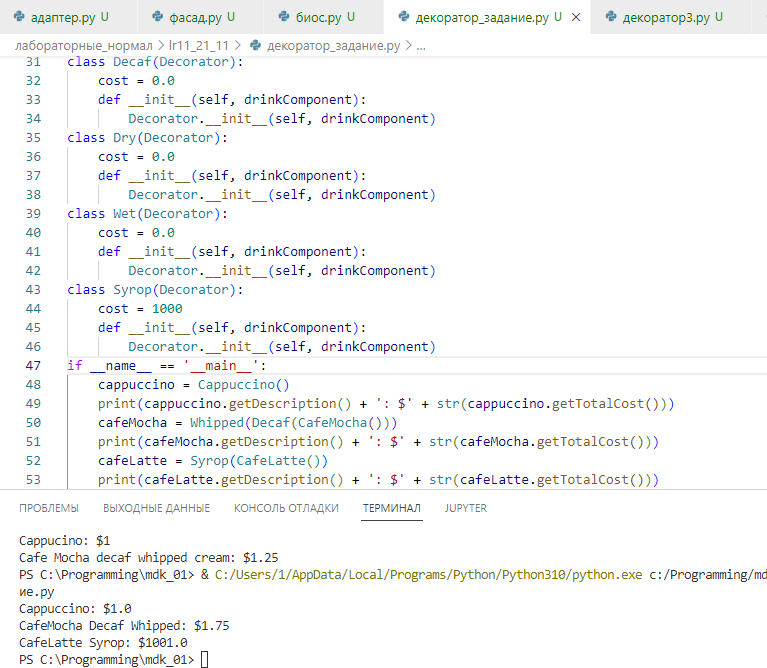


Рисунок 4 – Реализация декоратора

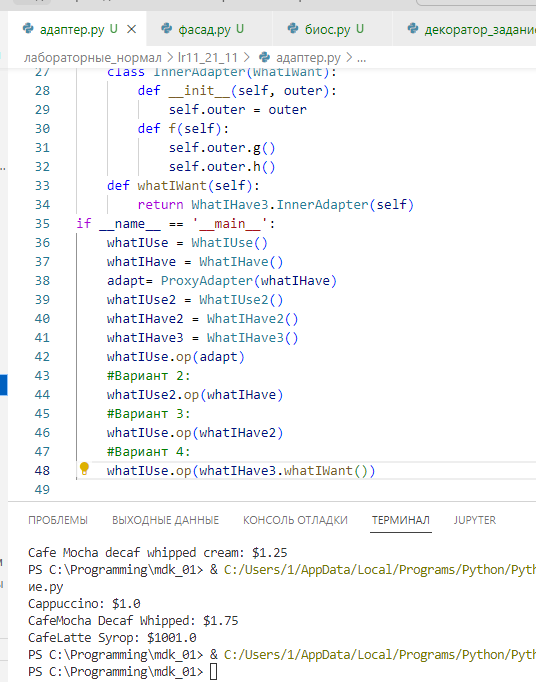


Рисунок 5 – Пример реализации адаптера



Рисунок 6 – Пример реализации фасада



Рисунок 7 – Реализация фасада по заданию

**Контрольные вопросы**

*1.Виды структурных шаблонов?*

Существуют следующие виды структурных шаблонов:

* Адаптер (Adapter);
* Мост (Bridge);
* Компоновщик (Composite);
* Декоратор (Decorator);
* Фасад (Facade)
* Легковес (Flyweight);
* Заместитель (Proxy).

*2.Функции структурных шаблонов?*

|  |  |
| --- | --- |
| Вид порождающего шаблона | Функция порождающего шаблона |
| Адаптер (Adapter) | Позволяет объектам с несовместимыми интерфейсами работать вместе |
| Мост (Bridge) | Разделяет один или несколько классов на две отдельные иерархии — абстракцию и реализацию, позволяя изменять их независимо друг от друга |
| Компоновщик (Composite) | Позволяет сгруппировать объекты в древовидную структуру, а затем работать с ними так, если бы это был единичный объект |
| Декоратор (Decorator) | Позволяет динамически добавлять объектам новую функциональность, оборачивая их в полезные «обёртки». |
| Фасад (Facade) | Предоставляет простой интерфейс к сложной системе классов, библиотеке или фреймворку. |
| Легковес (Flyweight) | Позволяет вместить большее количество объектов в отведённую оперативной память за счёт экономного  разделения общего состояния объектов между собой, вместо хранения одинаковых данных в каждом объекте. |
| Заместитель (Proxy) | Позволяет подставлять вместо реальных объектов специальные объекты-заменители. Эти объекты перехватывают вызовы к оригинальному объекту, позволяя сделать что-то до или после передачи вызова оригиналу. |

**Вывод**

На данной лабораторной работе я изучил структурных шаблоны.

**Лабораторная работа №12**

Использование поведенческих шаблонов

**Цель работы**

Изучить поведенческие шаблоны.

**Ход работы**

**Задание.** Разработать программное обеспечение, содержащее поведенческие шаблоны.

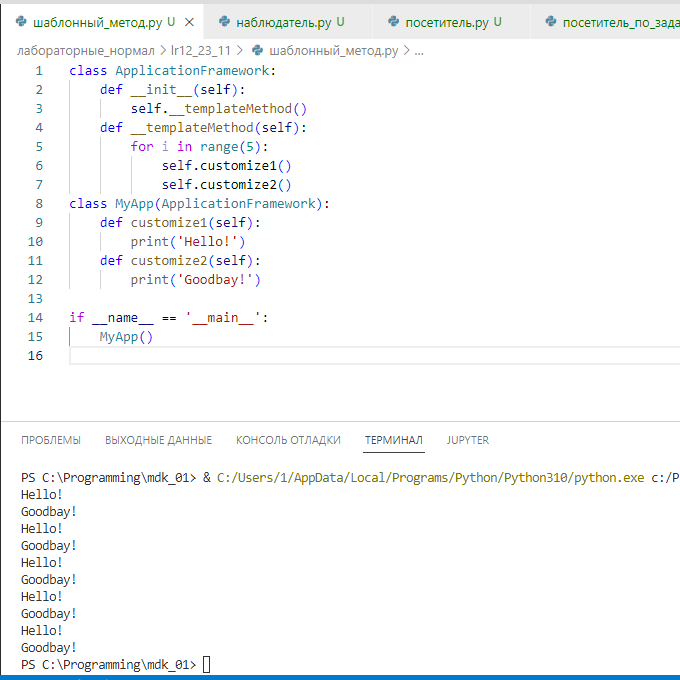


Рисунок 1 – Реализация шаблонного метода

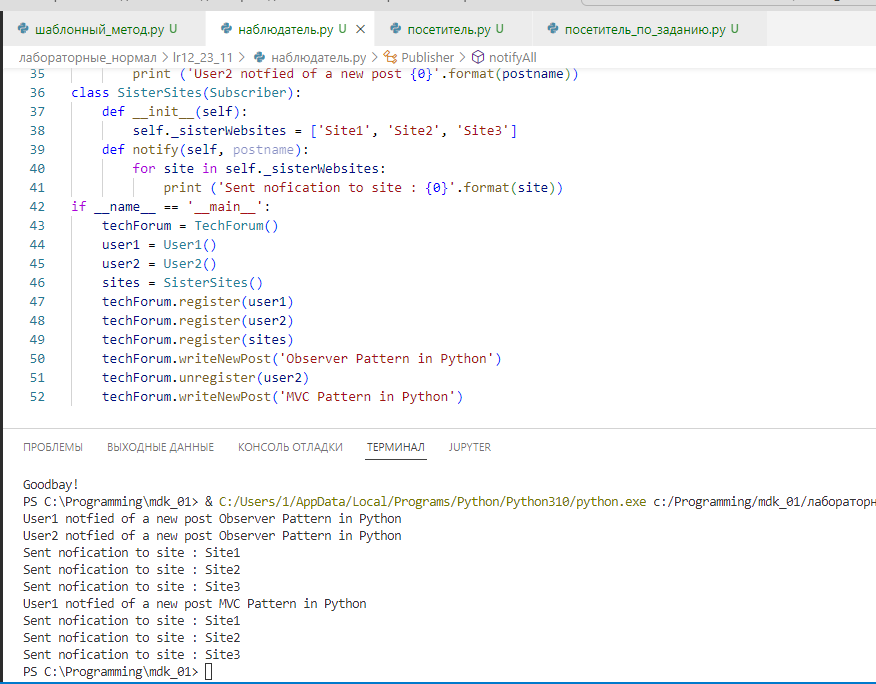


Рисунок 2 – Реализация «Наблюдателя»

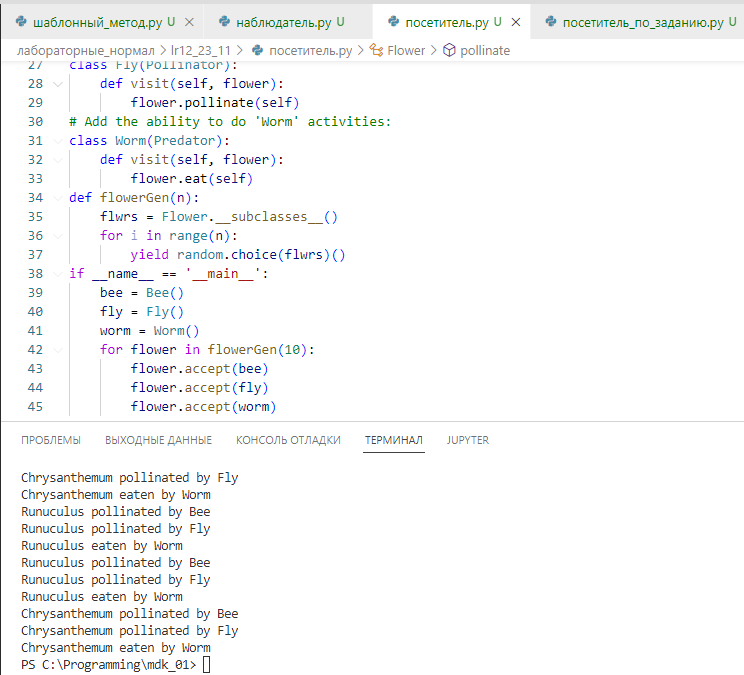


Рисунок 3 – Реализация «Посетителя»



Рисунок 4 – Реализация «Посетителя» в соответствии с заданием

**Контрольные вопросы**

*1.Виды поведенческих шаблонов?*

Существуют следующие виды поведенческих шаблонов:

* Цепочка обязанностей (Chain of Responsibility);
* Команда (Command);
* Итератор (Iterator);
* Посредник (Mediator);
* Снимок (Memento)
* Наблюдатель (Observer);
* Состояние (State);
* Стратегия (Strategy);
* Шаблонный метод (Template method);
* Посетитель (Visitor).

*2.Функции поведенческих шаблонов?*

|  |  |
| --- | --- |
| Вид порождающего шаблона | Функция порождающего шаблона |
| Цепочка обязанностей (Chain of Responsibility) | Позволяет передавать запросы последовательно по цепочке обработчиков. Каждый последующий обработчик решает, может ли он обработать запрос сам и стоит ли передавать запрос дальше по цепи. |
| Команда (Command) | Превращает запросы в объекты, позволяя передавать их как аргументы при вызове методов, ставить запросы в очередь, логировать их, а также поддерживать отмену операций |
| Итератор (Iterator) | Даёт возможность последовательно обходить элементы составных объектов, не раскрывая их внутреннего  представления |
| Посредник (Mediator) | Позволяет уменьшить связанность множества классов между собой, благодаря перемещению этих связей в один класс-посредник. |
| Снимок (Memento) | Позволяет делать снимки состояния объектов, не раскрывая подробностей их реализации. Затем снимки можно использовать, чтобы восстановить прошлое состояние объектов. |
| Наблюдатель (Visitor) | Создаёт механизм подписки, позволяющий одним объектам следить и реагировать на события, происходящие в других объектах |
| Состояние (State) | Позволяет объектам менять поведение в зависимости от своего состояния. Извне создаётся впечатление, что изменился класс объекта |
| Стратегия (Strategy) | Определяет семейство схожих алгоритмов и помещает каждый из них в собственный класс. После чего, алгоритмы можно взаимозаменять прямо во время исполнения программы |
| Шаблонный метод (Template method) | Определяет скелет алгоритма, перекладывая ответственность за некоторые его шаги на подклассы. Паттерн позволяет подклассам переопределять шаги алгоритма, не меняя его общей структуры |
| Посетитель (Visitor) | Позволяет создавать новые операции, не меняя классы объектов, над которыми эти операции могут выполняться |

**Вывод**

На данной лабораторной работе я изучил поведенческие шаблоны.

**Лабораторная работа №13**

Знакомство с Qt designer

**Цель работы**

Разработать оконное приложение с использованием текстовых компонентов

**Задание**

Выполните задание из приложенного файла «Введение в виджеты. Tk и ttk»

**Ход работы**

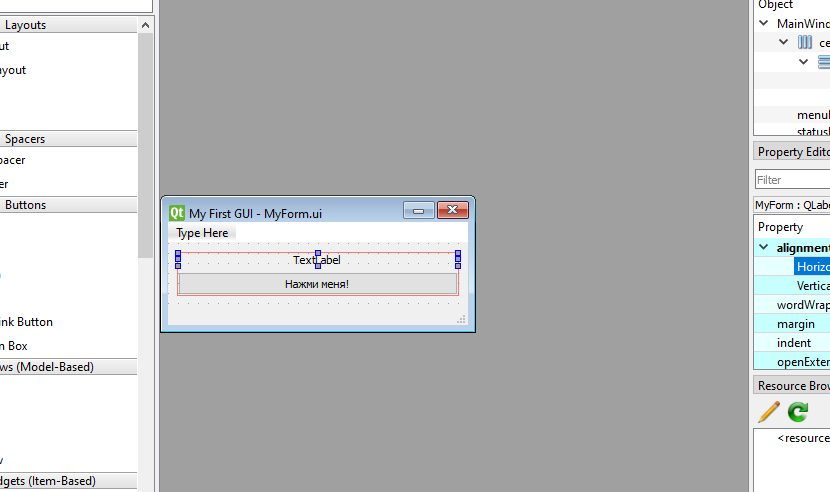


Рисунок 1 – Макет, созданный в дизайнере



Рисунок 2 – Создание макета для приложения «ToDo»

**Контрольные вопросы**

1. *Перечислите основные компоненты интерфейса в tkinter*

В tkinter существуют следующие основные компоненты интерфейса:

* Button: кнопка
* Label: текстовая метка
* Entry: однострочное текстовое поле
* Text: многострочное текстовое поле
* Checkbutton: флажок
* Radiobutton: переключатель или радиокнопка
* Frame: фрейм, который организует виджеты в группы
* Listbox: список
* Combobox: выпадающий список
* Menu: элемент меню
* Scrollbar: полоса прокрутки
* Treeview: позволяет создавать древовидные и табличные элементы
* Scale: текстовая метка
* Spinbox: список значений со стрелками для перемещения по элементам
* Progressbar: текстовая метка
* Canvas: текстовая метка
* Notebook: панель вкладок

1. *Каким образом можно связать нажатие кнопки с выполнением функции?*

Для обработки нажатия на кнопку необходимо установить в конструкторе параметр **command**, присвоив ему ссылку на функцию, которая будет срабатывать при нажатии.

**Вывод**

В данной лабораторной работе было разработано оконное приложение с использованием текстовых компонентов.

**Лабораторная работа №14**

Разработка приложения с несколькими формами

**Цель работы**

Научиться разрабатывать приложение с несколькими формами.

**Задание**

Выполнить задание из приложенного файла «Работа с виджетами и окнами».

**Ход работы**

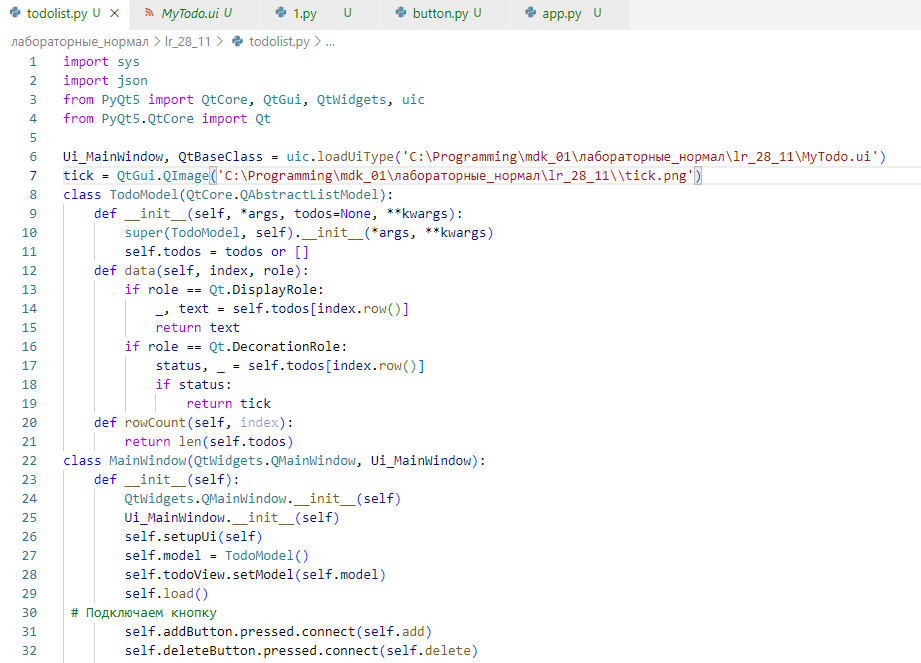


Рисунок 1 – Листинг кода для приложения «ToDo» часть 1



Рисунок 2 – Листинг кода для приложения «ToDo» часть 2

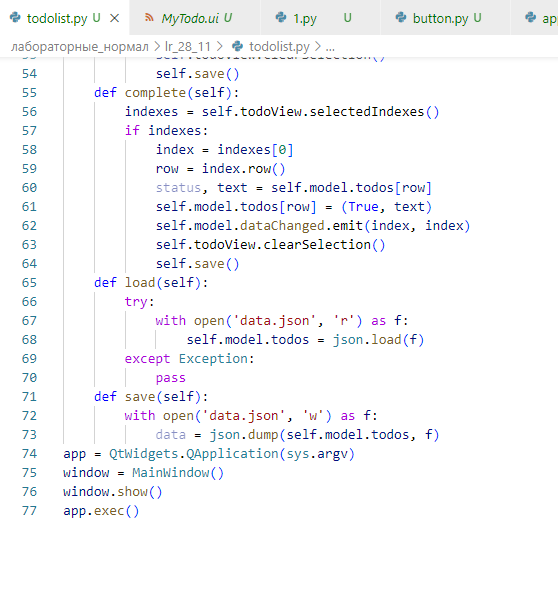


Рисунок 3 – Листинг кода для приложения «ToDo» часть 3

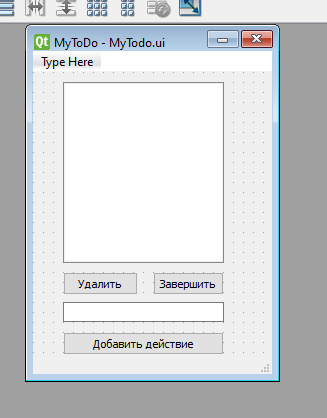


Рисунок 4 – Макет для приложения «ToDo»

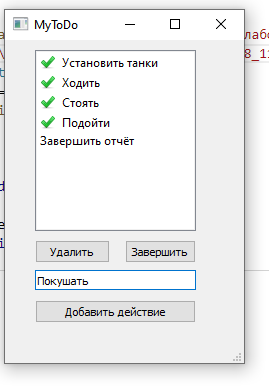


Рисунок 5 – Работа приложения

**Контрольные вопросы**

1. *Методы отображения новой формы?*

Имеются следующие методы отображения новой формы:

* Toplevel;
* MessageBox.

1. *С помощью какого метода получается значение поля Entry?*

С помощью параметра validate конструктора Entry можно задать,

когда проводить валидацию введенного значения. Этот параметр может

принимать следующие значения:

* none: отсутствие валидации, значение по умолчанию
* focus: валидация при получении фокуса
* focusin: валидация при изменении фокуса
* focusout: валидация при потере фокуса
* key: валидация при каждом вводе нового символа
* all: валидация при изменении фокуса и вводе символов в поле

**Вывод**

В данной лабораторной работе был получен навык разработки приложений с несколькими формами.

**Лабораторная работа №15**

Работа Qt designer и PyQt

**Цель работы**

Научиться использовать не визуальные компоненты.

**Задание**

Выполнить задание из приложенного файла, оформить отчет.

**Ход работы**

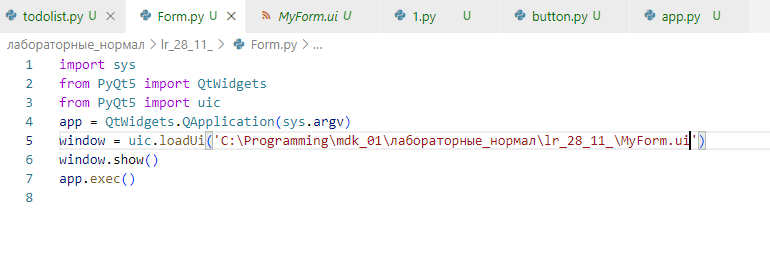


Рисунок 1 – Листинг кода для запуска макета

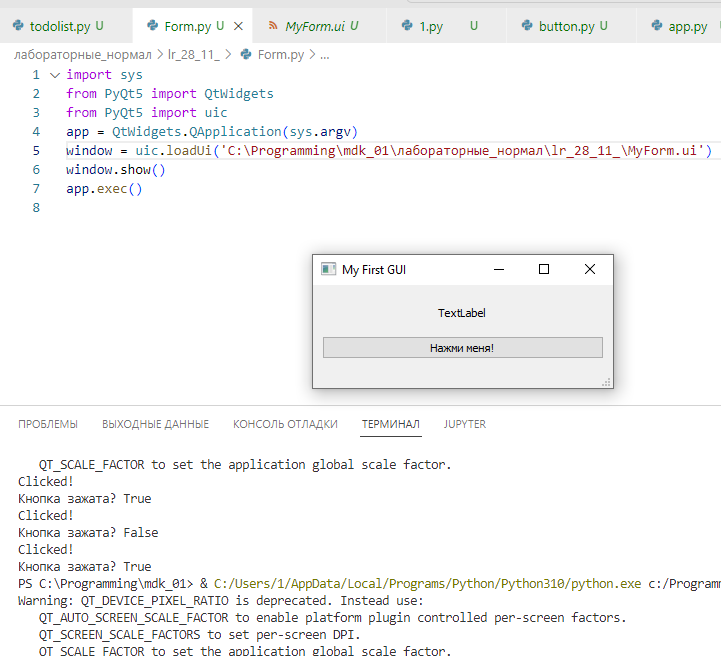


Рисунок 2 – Запуск макета из прошлой работы

**Контрольные вопросы**

1. *Перечислите основные способы позиционирования элементов интерфейса?*

Основные способы позиционирования интерфейса:

* Pack;
* Place;
* Grid.

1. *Какой параметр отвечает за растяжение виджета по свободному пространству?*

За растяжение виджета по свободному пространству отвечает параметр fill.

**Вывод**

В данной лабораторной работе был навык по использованию не визуальных компонентов.

**Лабораторная работа №22**

Знакомство с PyQt

**Цель работы**

Получение навыков разработки хранилищ данных с использованием реляционных СУБД

**Задание**

Выполните задания из приложенного файла «Введение в Tkinter. Первая программа»

**Ход работы**

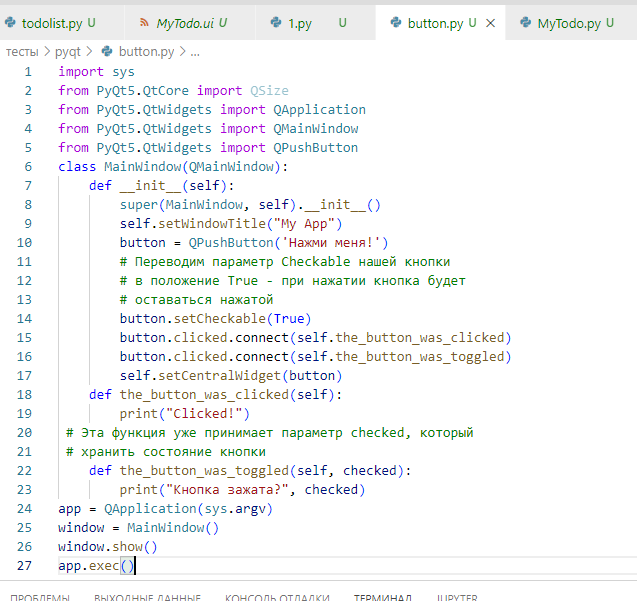


Рисунок 1 – Листинг кода для кнопки

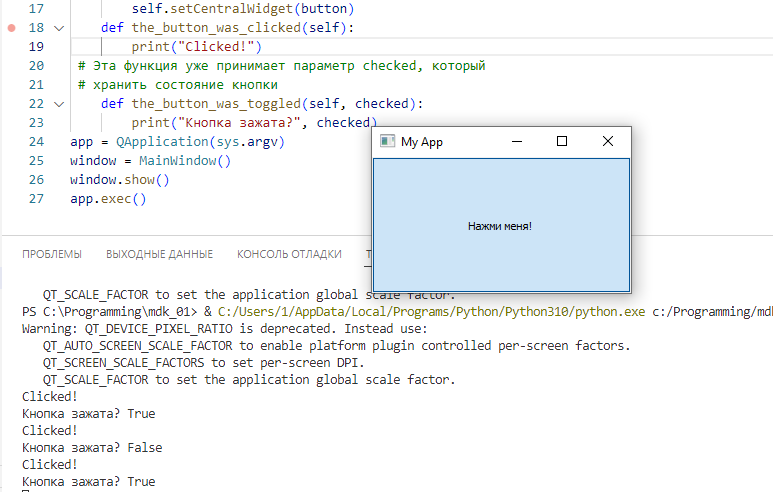


Рисунок 2 – Работа приложения

**Контрольные вопросы**

1. *Что такое пользовательский интерфейс?*

Пользовательский интерфейс - интерфейс, обеспечивающий передачу информации между пользователем-человеком и программно-аппаратными компонентами компьютерной системы.

**Вывод**

В данной лабораторной работе были получены навыки разработки хранилищ данных с использованием реляционных СУБД.

**Лабораторная работа №28**

Создание приложения с БД

**Цель работы**

Получение навыков разработки хранилищ данных с использованием реляционных СУБД

**Задание**

Выполните задание из приложенного файла «Работа с SQLite».

**Ход работы**

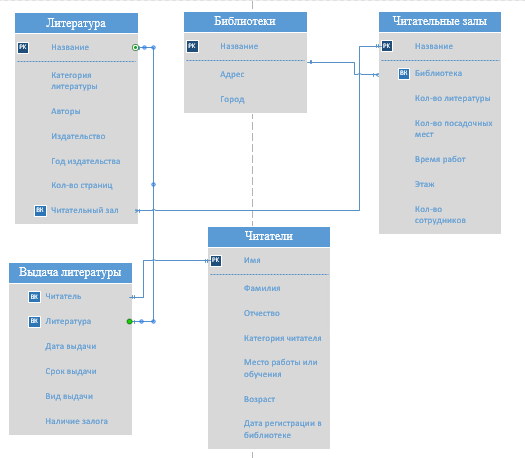


Рисунок 1 – Схема для базы данных

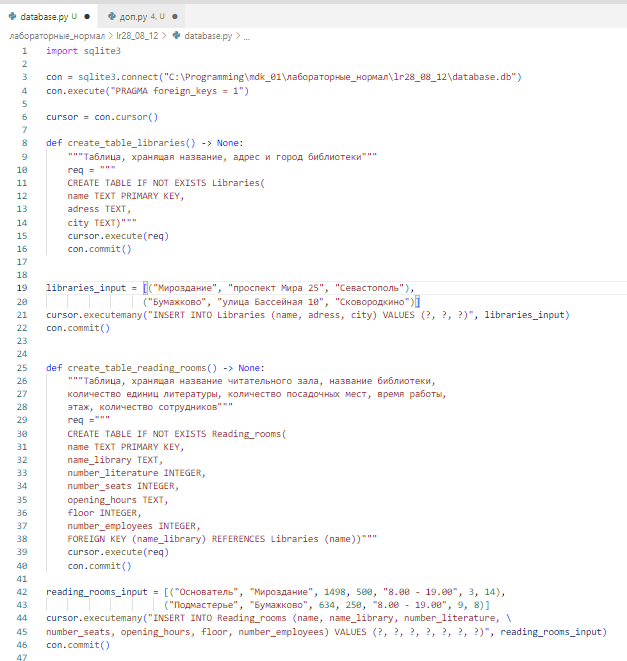


Рисунок 2 – Листинг кода для базы данных часть 1

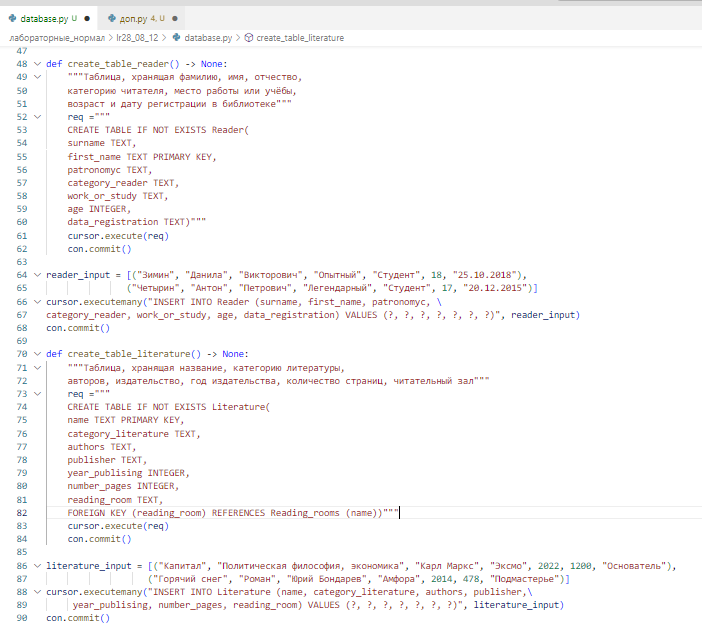


Рисунок 3 – Листинг кода для базы данных часть 2

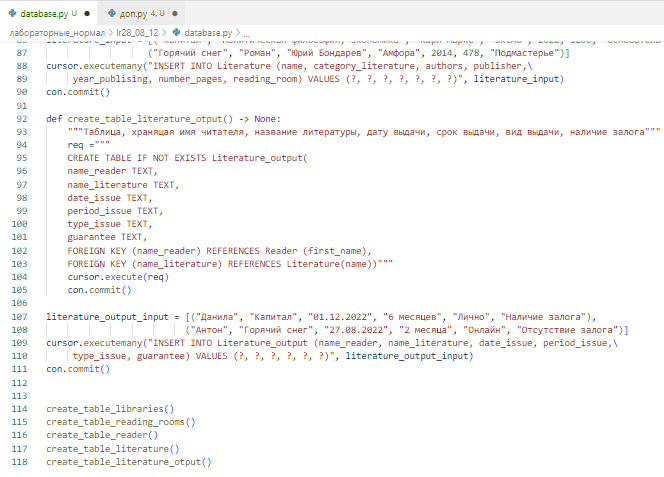


Рисунок 4 – Листинг кода для базы данных часть 3

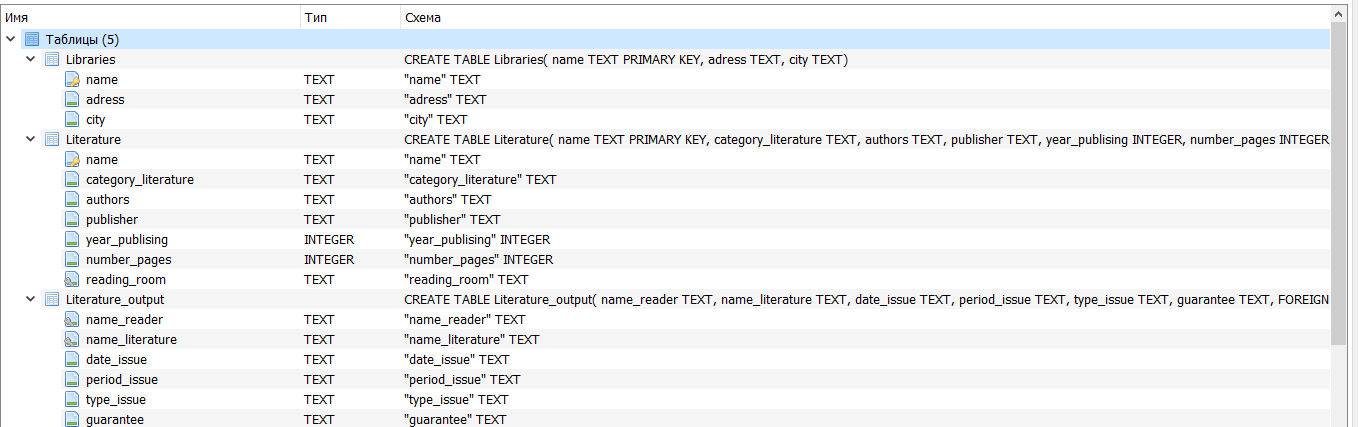


Рисунок 5 – Структура базы данных часть 1

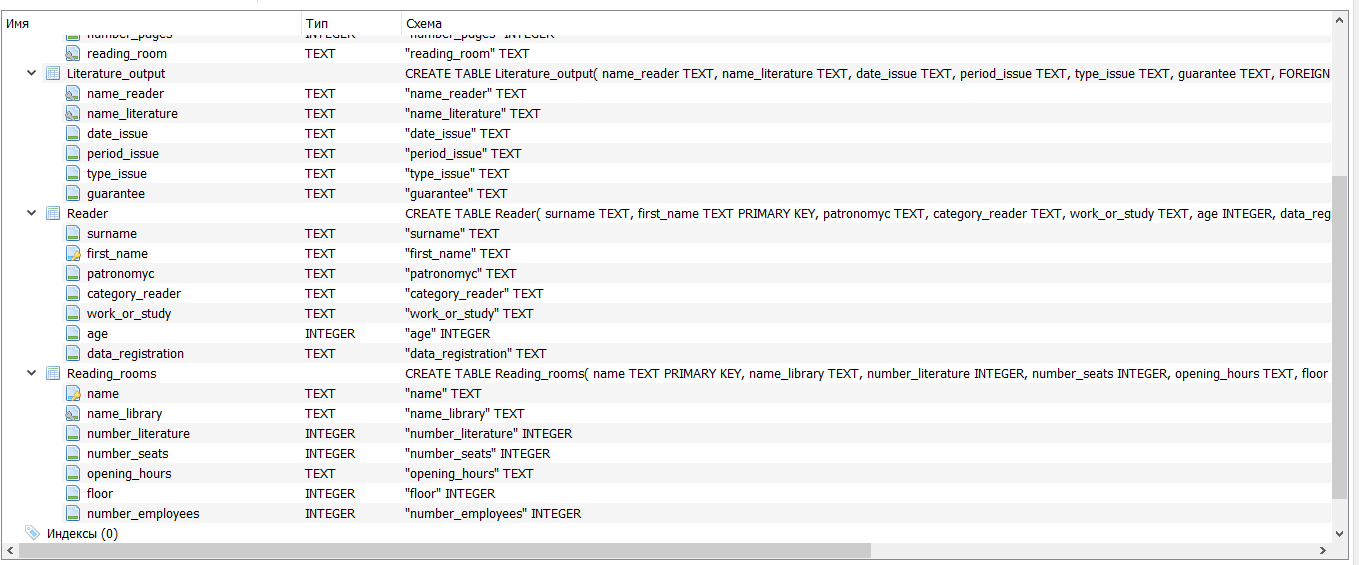


Рисунок 6 – Структура базы данных часть 2

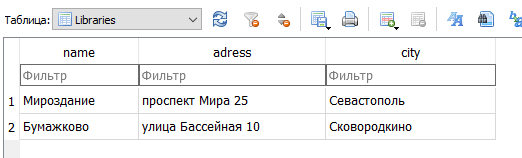


Рисунок 7 – Заполненная таблица «Libraries»

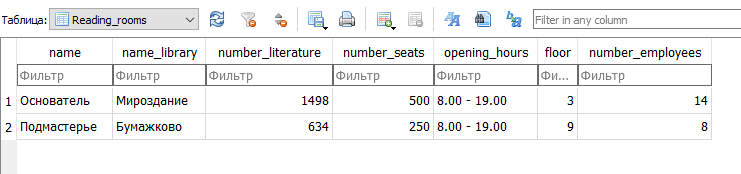


Рисунок 8 – Заполненная таблица «Reading\_rooms»

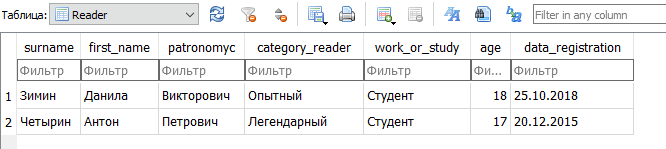


Рисунок 9 – Заполненная таблица «Reader»

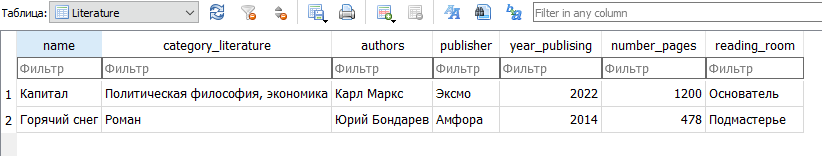


Рисунок 10 – Заполненная таблица «Literature»



Рисунок 11 – Заполненная таблица «Literature\_output»

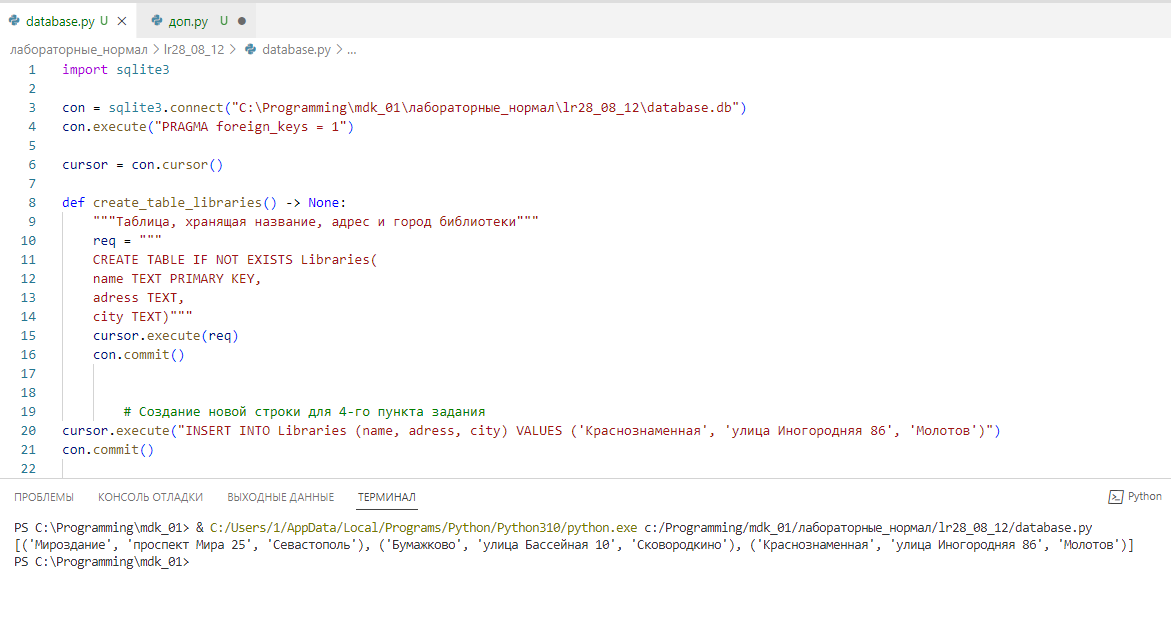


Рисунок 12 – Добавление новой строки в таблицу «Libraries»

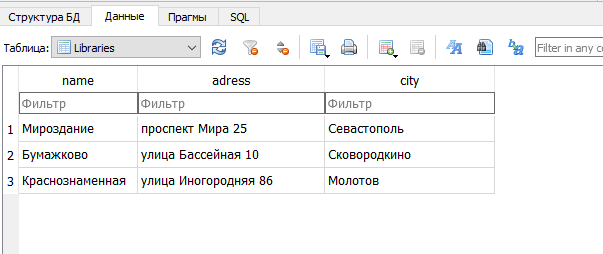


Рисунок 13 – Результат добавления строки

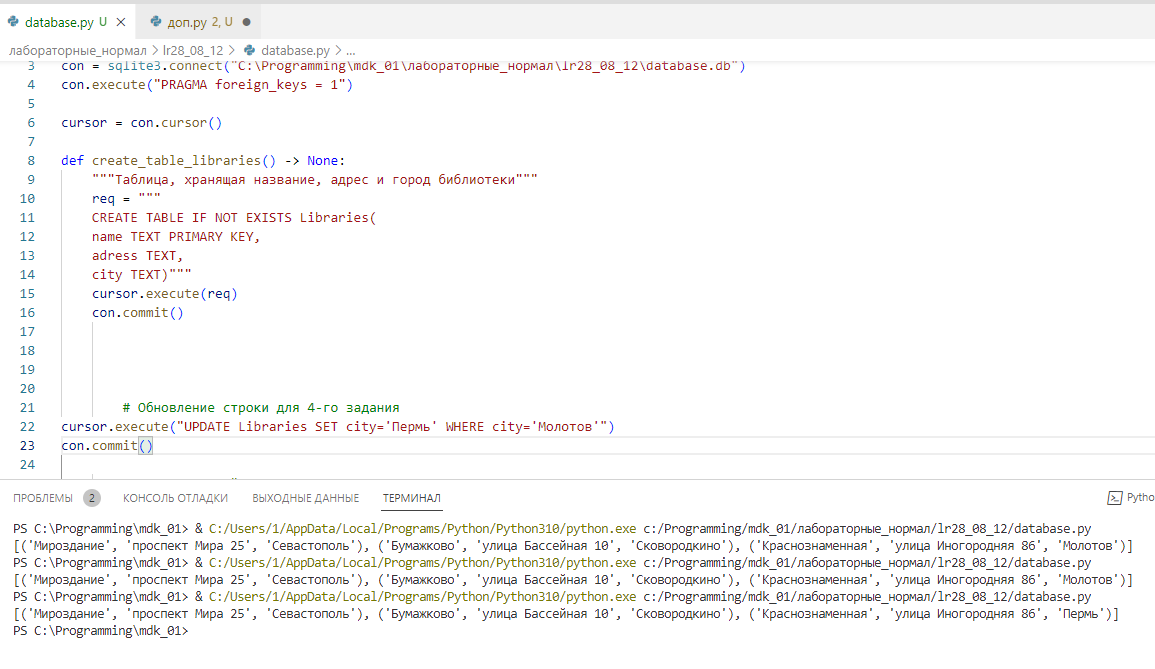


Рисунок 14 – Обновление строки в таблице «Libraries»

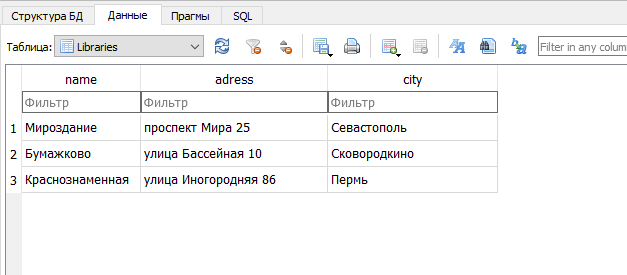


Рисунок 15 – Результат обновления строки

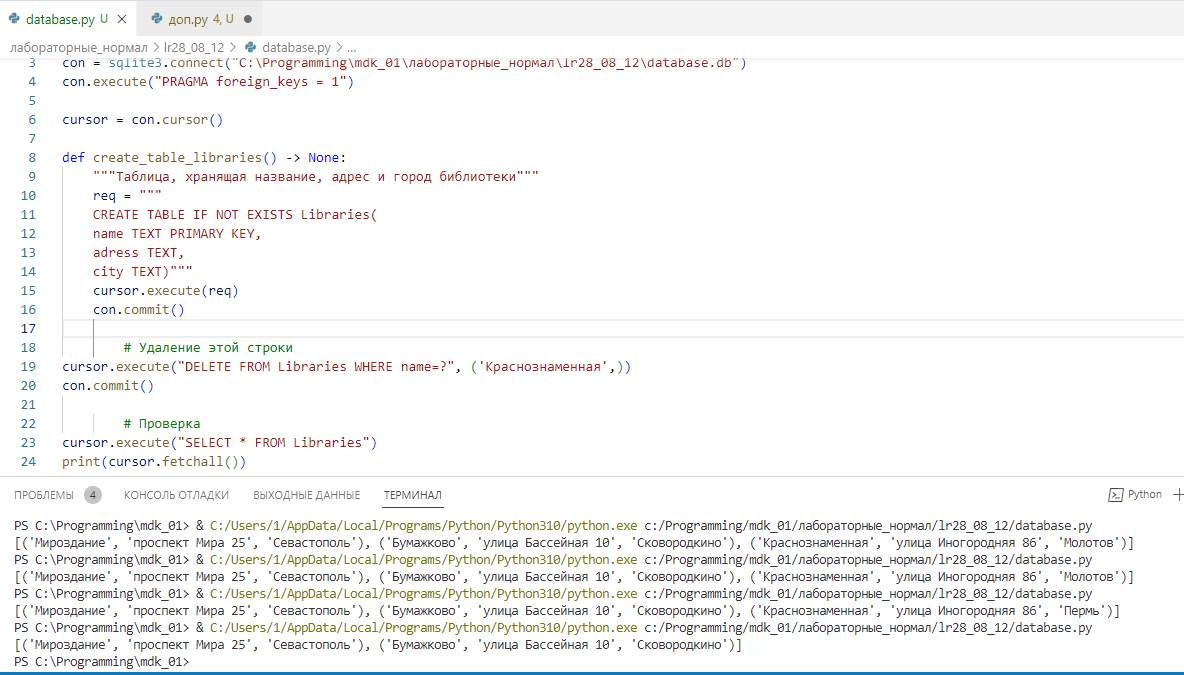


Рисунок 16 – Удаление строки из таблицы «Libraries»

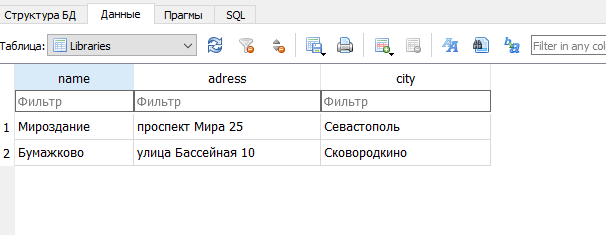


Рисунок 17 – Результат удаления строки

**Контрольные вопросы**

1. *Что представляет из себя база данных SQLite?*

SQLite – это C библиотека, реализующая легковесную дисковую базу данных (БД), не требующую отдельного серверного процесса и позволяющую получить доступ к БД с использованием языка запросов SQL.

1. *Какие СУБД вам известны, в чем их отличия от SQLite?*

В связи с работой только с СУБД MS SQL SERVER, то будет проведено их сравнение в таблице под номером 1.

Таблица 1 – Сравнение SQLite и MS SQL SERVER

|  |  |  |
| --- | --- | --- |
| Номер | SQLite | MS SQL SERVER |
| 1 | Он был разработан Д. Ричардом Хиппом в августе 2000 года | Он был разработан корпорацией Microsoft и первоначально выпущен 24 апреля 1989 года |
| 2 | SQLite разработан только на языке C | MS SQL server написан на C, C ++ и C# |
| 3 | Основной моделью базы данных для SQLite является СУБД | Основной моделью базы данных для MS SQL Server также является СУБД |
| 4 | Лицензия на SQLite с открытым исходным кодом | Лицензия на MS SQL Server является коммерческой |
| 5 | Для SQLite не существует вторичной модели базы данных. | Вторичной моделью базы данных для MS SQL Server являются СУБД Document store и Graph |
| 6 | SQLite не поддерживает формат данных XML | MS SQL Server поддерживает формат данных XML |
|  |  |  |
| Продолжение таблицы 1 | | |
| Номер | SQLite | MS SQL SERVER |
| 7 | В SQLite схема данных является динамической | В MS SQL Server схема данных является фиксированной |
| 8 | Он не поддерживает никаких методов репликации | Он поддерживает методы репликации, но в зависимости от версии SQL-Server |
| 9 | Для работы SQLite не требуется сервер | Серверными операционными системами для MS SQL Server являются Linux и Windows |
| 10 | Он поддерживает возможности в памяти | Он также поддерживает возможности в памяти |
| 11 | SQLite не поддерживает метод Map Reduce | MS SQL Server также не поддерживает метод Map Reduce |

**Вывод**

В данной лабораторной работе были получены навыки разработки хранилищ данных с использованием реляционных СУБД.