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 лабораторно-практическим работам**

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

|  |  |
| --- | --- |
|  | Выполнил (-а) обучающийся (-аяся):  Луганский Вадим Витальевич |
|  | (Ф.И.О.) |
|  | Специальность:  09.02.07 Информационные системы и программирование |
|  | (код и наименование) |
|  | Курс: 2 |
|  | Группа: ИС-21 184118 |
|  | Преподаватель:  Воронцов Роман Антонович |
|  | (Ф.И.О.) |

Архангельск 2022

**Лабораторная работа №1**

**Работа с классами**

**Цель работы:** Научиться работать с классами.

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

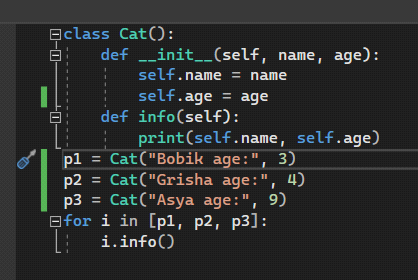


Рисунок 1 – создание класса Cat

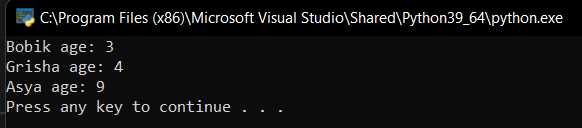


Рисунок 2 – результат программы

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

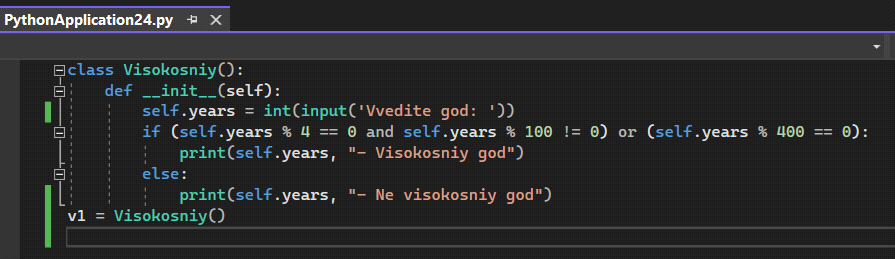


Рисунок 3 – задача с високосным годом

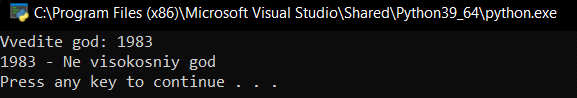


Рисунок 4 – результат программы

Задание по варианту (9 вариант):



Рисунок 1 – листинг кода

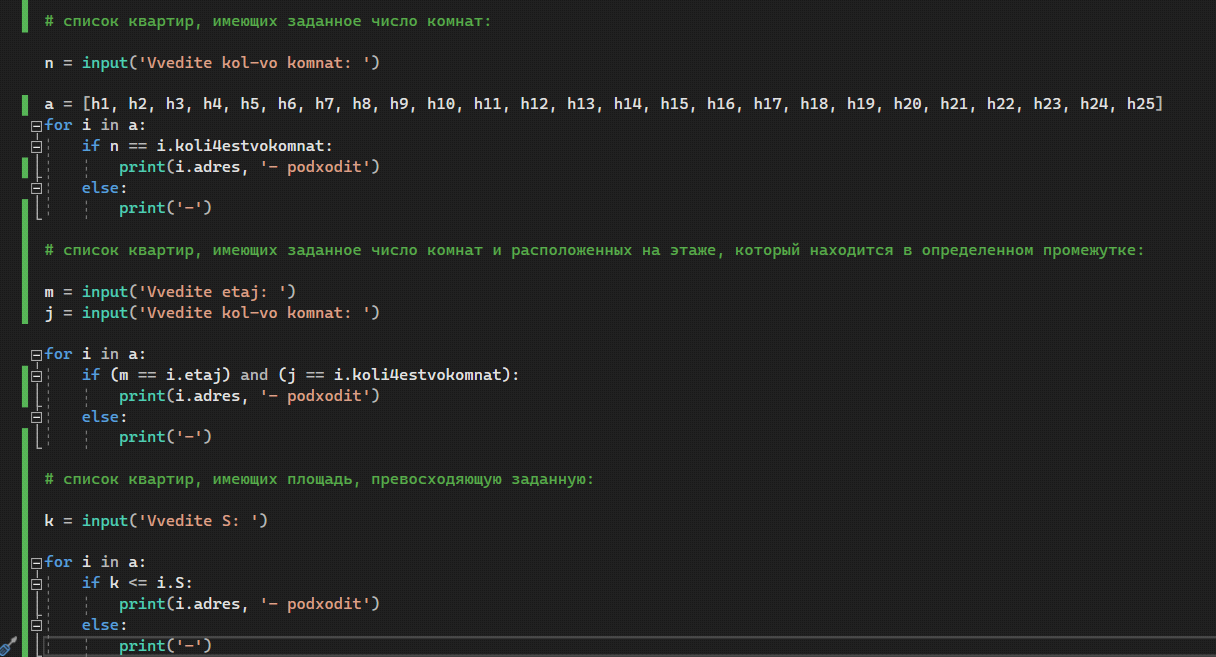


Рисунок 2 – листинг кода

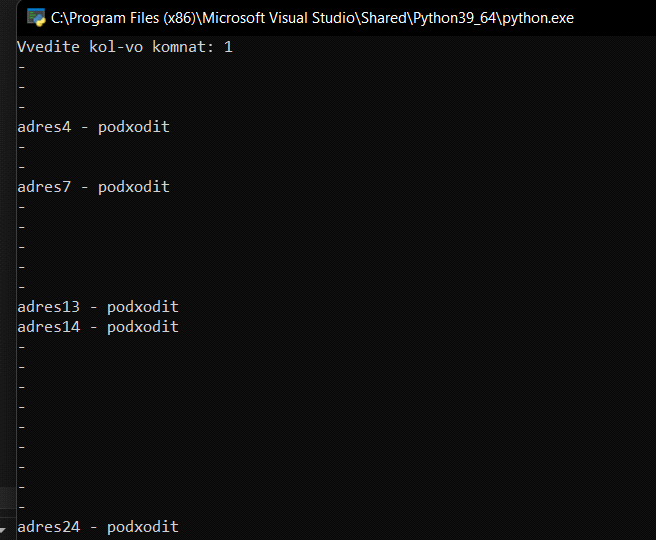


Рисунок 3 – результат программы

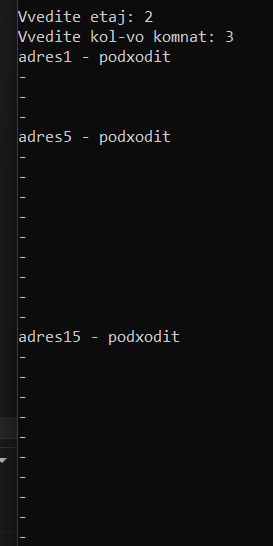


Рисунок 4 – результат программы

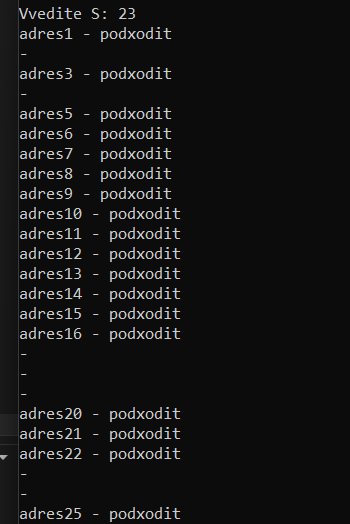


Рисунок 5 – результат программы

**Контрольные вопросы:**

**1. Что представляет собой класс?**

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

**2. Как создаются новые классы?**

Классы создаются с помощью инструкции class, за которой следует произвольное имя класса, после которого ставится двоеточие, далее с новой строки и с отступом реализуется тело класса.

**Вывод**

Я научился работать с классами.

**Лабораторная работа №2**

**Перегрузка методов**

**Цель работы:** Научиться работать с перегрузкой методов.

**Задание:** Выполните перегрузку метода согласно варианта вашего индивидуального задания.

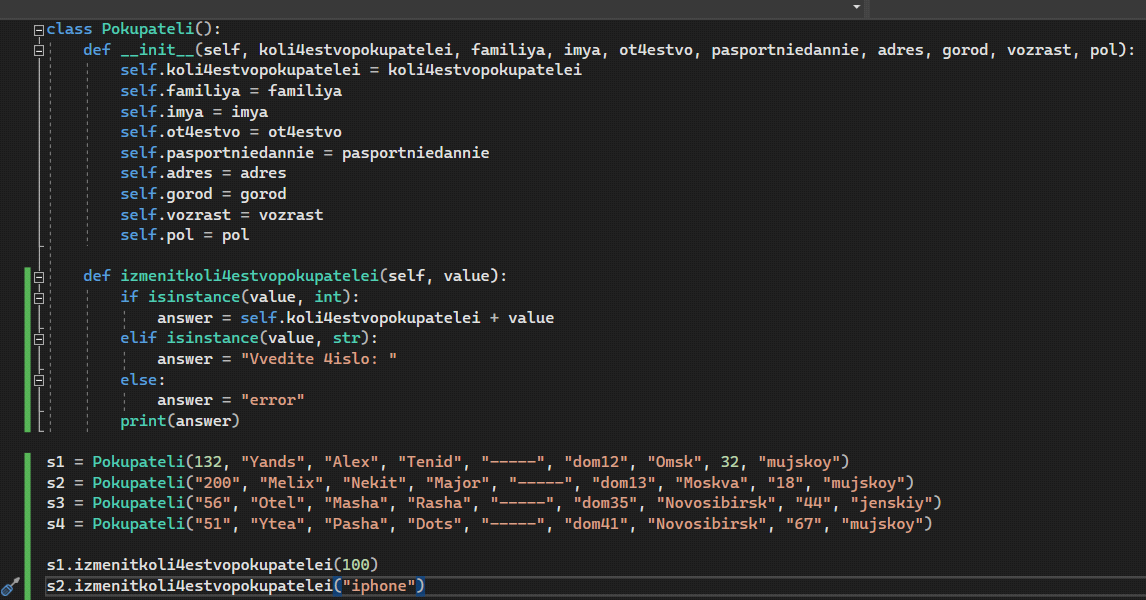


Рисунок 1 – создание перегрузки

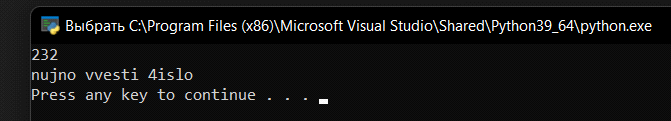


Рисунок 2 – результат программы

**Контрольные вопросы:**

**1. Что такое 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 перегрузка операторов?**

Это возможность одновременного существования в одной области видимости нескольких различных вариантов применения оператора, имеющих одно и то же имя, но различающихся типами параметров, к которым они применяются.

**Вывод**

Я научился работать с перегрузкой методов.

**Лабораторная работа №3**

**Определение операций в классе**

**Цель работы:** Научиться работать в классе.

**Задание:** Выполнить задания из приложенных файлов:

1. ООП. Задание 6. Перегрузка операций. Документирование.



Рисунок 1 – создание документирования

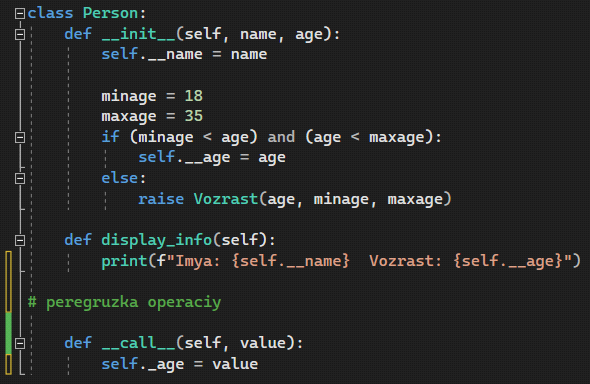


Рисунок 2 – перегрузка операций

2. ООП. Задание 7. Ассоциация. Метапрограммирование.

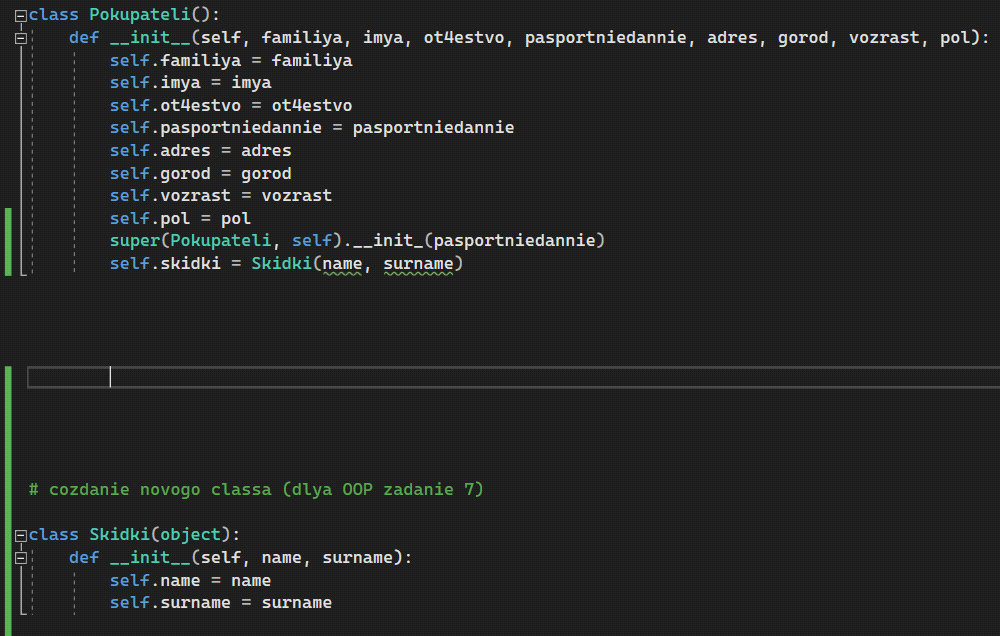


Рисунок 3 – привязка классов

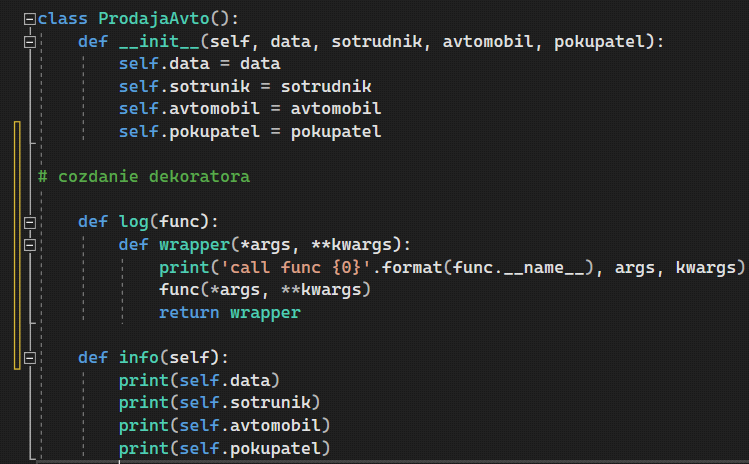


Рисунок 4 – создание декоратора для методов класса

3. ООП. Задание 8. Агрегация. Контейнеры. Итераторы.

**Контрольные вопросы:**

**1. Что такое конструктор?**

Это специальный блок инструкций, вызываемый при создании объекта.

**Вывод**

Я научился работать в классе.

**Лабораторная работа №4**

**Создание наследованных классов.**

**Цель работы:** Научиться работать с наследованными классами.

**Задание:** Выполните задания из файлов:

1. ООП. Задание 2. Конструктор.

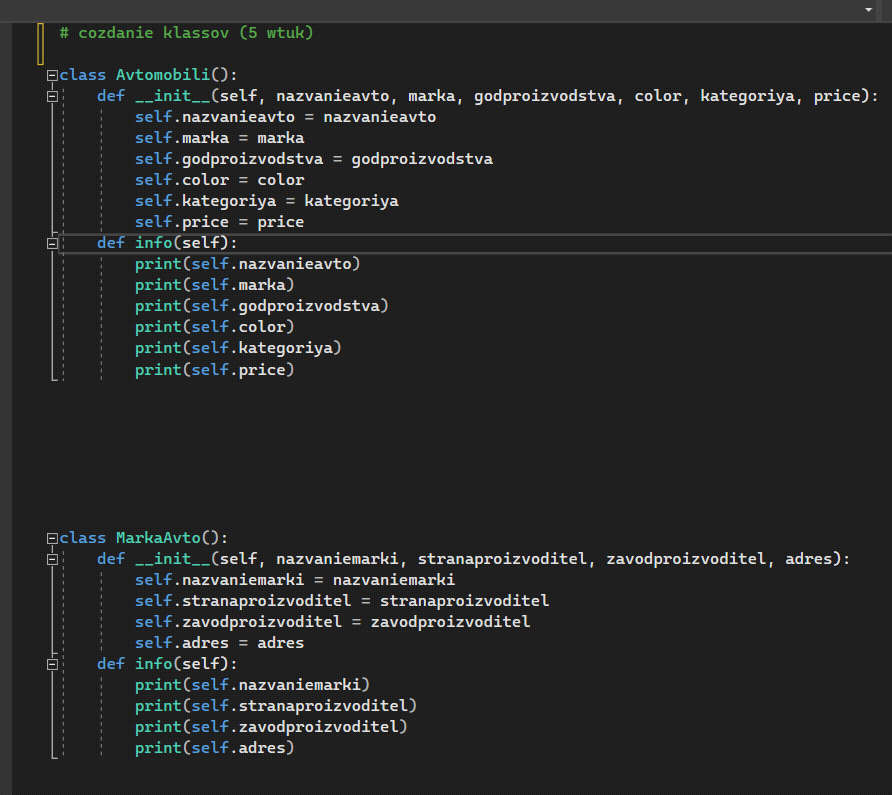


Рисунок 1 – листинг кода (создание классов)

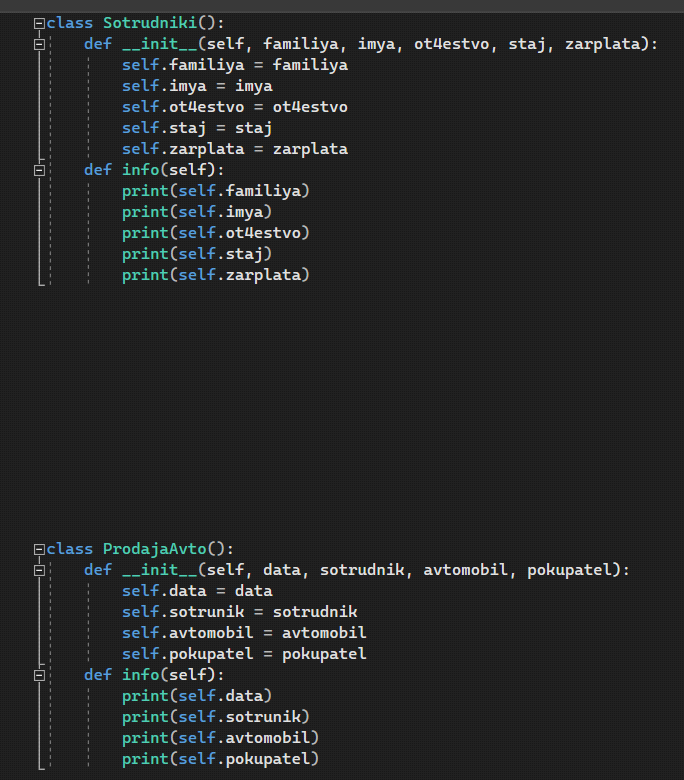


Рисунок 2 – листинг кода (создание классов)

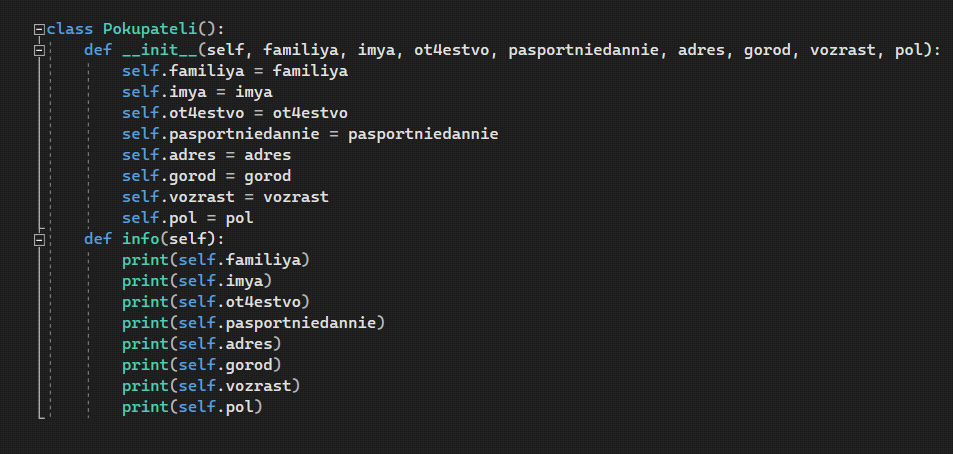


Рисунок 3 – листинг кода (создание классов)

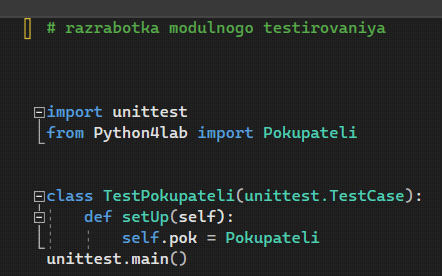


Рисунок 4 – создание модульного тестирования

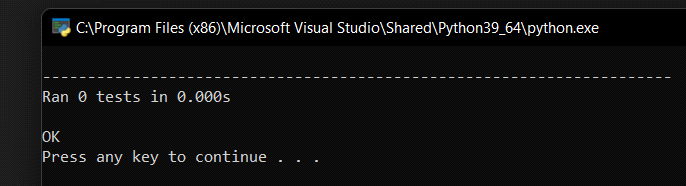


Рисунок 5 – результат тестирования

2. ООП. Задание 3. Деструктор.

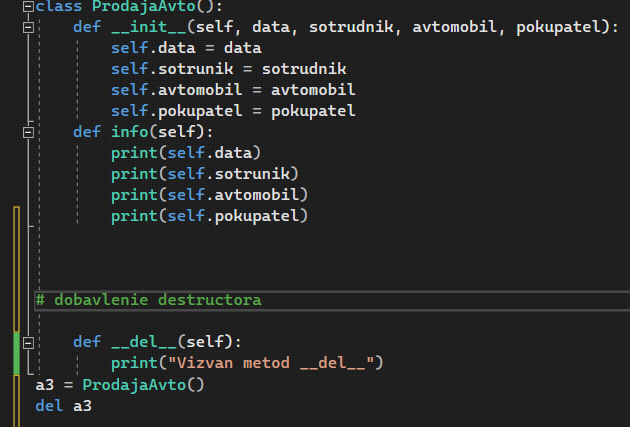


Рисунок 6 – создание деструктора

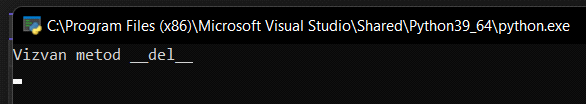


Рисунок 7 – результат деструктора

3. ООП. Задание 4. Наследование. Полиморфизм.

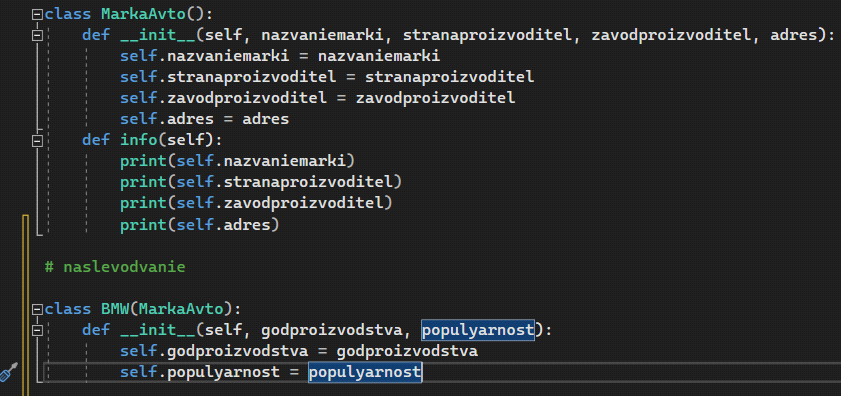


Рисунок 8 – создание наследования класса

**Контрольные вопросы:**

**1. Что такое наследование?**

Наследование — концепция объектно-ориентированного программирования, согласно которой [абстрактный тип данных](https://ru.wikipedia.org/wiki/%D0%90%D0%B1%D1%81%D1%82%D1%80%D0%B0%D0%BA%D1%82%D0%BD%D1%8B%D0%B9_%D1%82%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) может наследовать данные и функциональность некоторого существующего типа, способствуя повторному использованию компонентов [программного обеспечения](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5).

**Вывод**

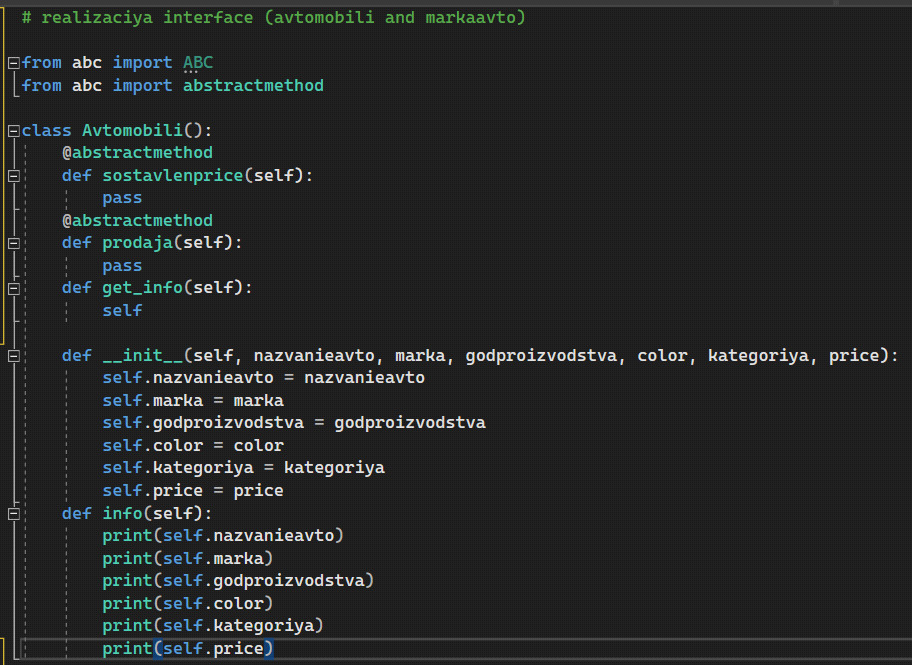
Я научился работать с наследованными классами

**Лабораторная работа №5**

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

**Цель работы:** Научиться работать с интерфейсами.

**Задание:** 1. Выполнить задание из файла «Интерфейсы Python».



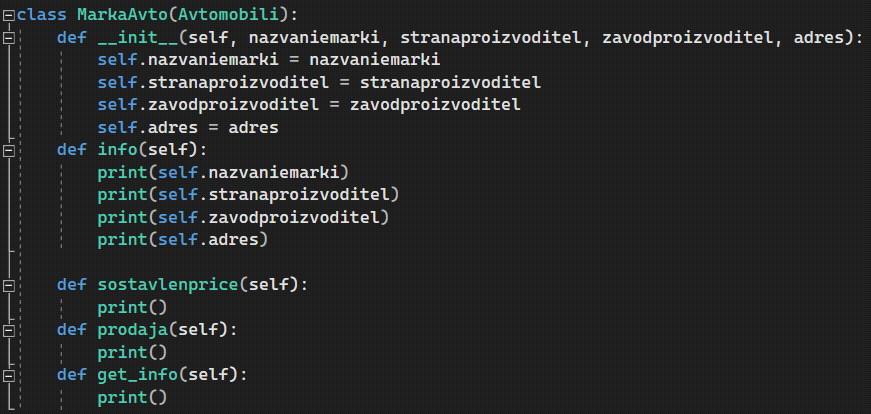


Рисунок 1, 2 – реализация интерфейса

**Контрольные вопросы**:

**1. Что такое интерфейс?**

Интерфейс представляет ссылочный тип, который может определять некоторый функционал - набор методов и свойств без реализации. Затем этот функционал реализуют классы и структуры, которые применяют данные интерфейсы.

**Вывод**

Я научился работать с интерфейсами.

**Лабораторная работа №6**

**Работа с типом данных структура**

**Цель работы:** Научиться работать с типом данных структуры.

**Задание:** 1. Выполнить задание из приложенного файла «Структуры в Python».

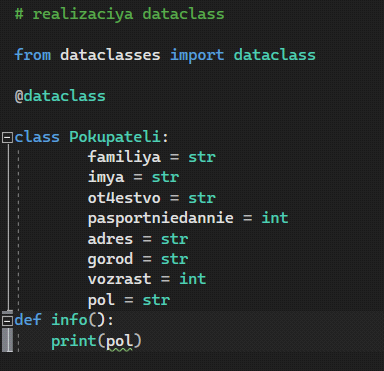


Рисунок 1 – реализация dataclass

**Контрольные вопросы:**

**1. Что такое структура?**

Это совокупность переменных, объединенных одним именем, предоставляющая общепринятый способ совместного хранения информации.

**2. Методы задания структур.**

* графический
* матричный
* аналитический
* множественный

**Вывод**

Я научился работать с типом данных структуры.

**Лабораторная работа №7.1**

**Использование основных шаблонов**

**Цель работы:** Изучить основные шаблоны.

**Задание:** 1. Выполнить задание из файла «Обобщенные классы в Python»

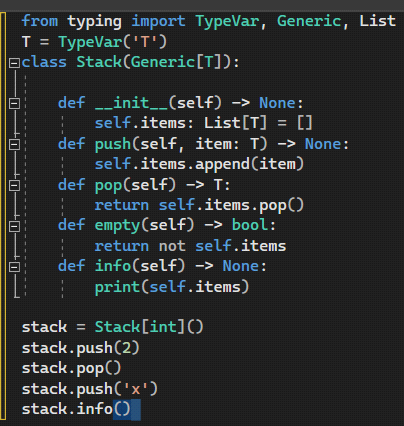


Рисунок 1 – Реализация обобщенного класса

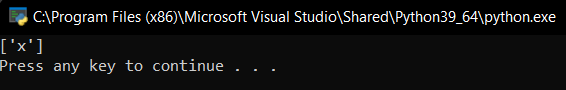


Рисунок 2 – Результат работы программы

**Контрольные вопросы:**

**1. Как вы понимаете шаблонные классы?**

Это альтернатива для создания классов с параметрами.

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

Python работает на основе динамической типизации, то обобщенных классов в нем нет, и дженерики видятся больше приятным дополнением, которое помещено в библиотеку typing. В целом, дженерики применяются больше в контексте «типизированного питона».

**Вывод**

В ходе лабораторной работы я изучил основные шаблоны.

**Лабораторная работа №7**

**Коллекции. Параметризованные классы**

**Цель работы:** Научиться работать с коллекциями. Параметризованными классами.

**Задание:**1. Выполнить задание из файла «Коллекции в Python».

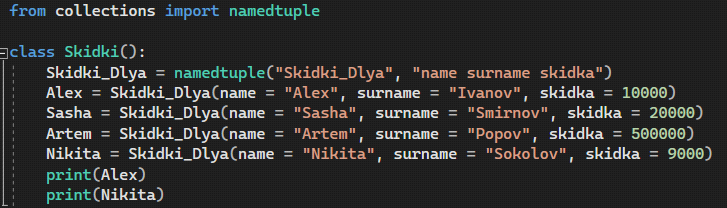


Рисунок 1 – реализация модуля namedtuple

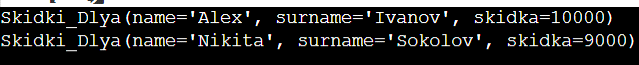


Рисунок 2 – результат программы

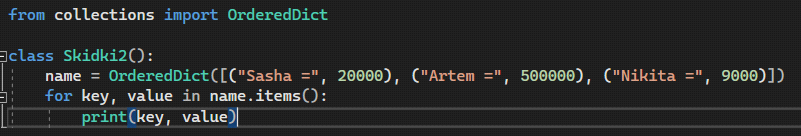


Рисунок 3 – реализация модуля OrderedDict



Рисунок 4 – результат программы

**Контрольные вопросы:**

**1. Основные виды коллекций?**

* defaultdict
* OrderedDict
* counter
* deque
* namedtuple
* enum.Enum (вне модуля)

**2. Применение параметризованных классов.**

Параметризованный класс представляет собой что-то вроде шаблона для построения других классов; шаблон может быть параметризован другими классами, объектами или операциями.

**Вывод**

Я научился работать с коллекциями. Параметризованными классами.

**Лабораторная работа №8**

**Использование регулярных выражений. Операции со списками**

**Цель работы**: научиться использовать регулярные выражения.

**Задание**: выполнить задание из приложенного файла «Регулярные выражения в Python».

Изображение выглядит как текст

Автоматически созданное описание

Рис 1 – регулярное выражение в Python.

Изображение выглядит как текст

Автоматически созданное описание

Рис 2 – результат работы программы.

**Контрольные вопросы:**

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

* Метод RegExp, который выполняет поиск совпадения в строке. Он возвращает массив данных.
* Метод RegExp, который тестирует совпадение в строке. Возвращает либо истину, либо ложь.  
  Метод String, который выполняет поиск совпадения в строке. Он возвращает массив данных либо null если совпадения отсутствуют.
* Метод String, который тестирует на совпадение в строке. Он возвращает индекс совпадения, или -1 если совпадений не будет найдено.
* Метод String, который выполняет поиск совпадения в строке, и заменяет совпавшую подстроку другой подстрокой, переданной как аргумент в этот метод.
* Метод String, который использует регулярное выражение или фиксированную строку чтобы разбить строку на массив подстрок.

**2.Экранирование спец символов.**

Чтобы использовать специальный символ как обычный, добавьте к нему обратную косую черту: \ Это называется «экранирование символа». Если мы ищем обратную косую черту \ ,это специальный символ как в обычных строках, так и в регулярных выражениях, поэтому мы должны удвоить её.

**Вывод**

Я научился использовать регулярные выражения.

**Лабораторная работа №9**

**Инкапсуляция**

**Цель работы:** Научиться работать с механизмом сокрытия компонентов программы.

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

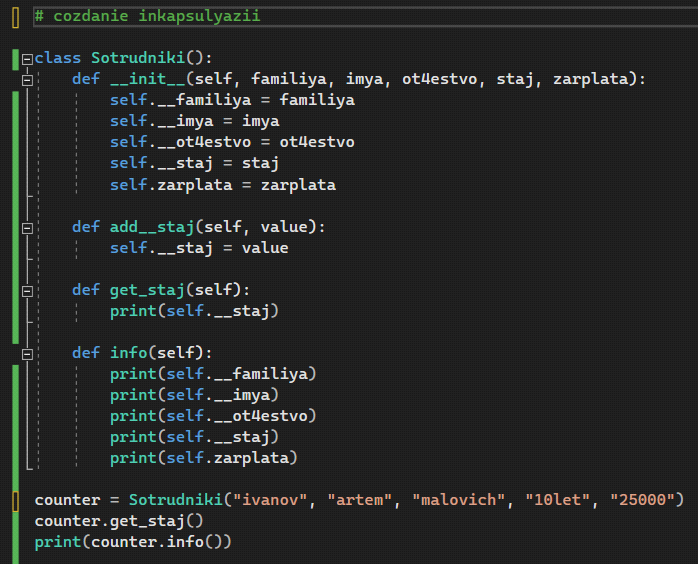


Рисунок 1 – Преобразование атрибутов и методов в приватные (инкапсуляция)

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2 – Результат работы программы

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 3 – Создание свойства

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 4 – Создание класса исключений

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 5 – Создание класса исключений



Рисунок 6 – Результат работы программы

**Контрольные вопросы:**

**1. Что представляет собой инкапсуляция?**

Это объединение функций и данных в рамках одной структуры, внутреннее состояние которой (данные) скрыто от внешнего мира.

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

* private – только внутри методов данного класса;
* protected - внутри методов данного класса и методов всех его потомков;
* public - любое место программы, в том числе во всех классах и их методах;
* internal - во всех файлах, необходимых для компиляции программы.

**3. Каким образом инкапсуляция реализована в языке Python?**

Инкапсуляция в Python работает лишь на уровне соглашения между программистами о том, какие атрибуты являются общедоступными, а какие — внутренними.

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

Инкапсуляция в C# осуществляется при помощи модификаторов доступа (private, public…). Конечным пользователем объекта здесь выступает либо объект наследник, либо программист.

**Вывод**

Я научился работать с механизмом сокрытия компонентов программы.

**Лабораторная работа №10**

**Использование порождающих шаблонов**

**Цель работы:** изучить порождающие шаблоны.

**Оборудование: п**ерсональный компьютер.

Изображение выглядит как текст, монитор, внутренний, снимок экрана

Автоматически созданное описание

Рисунок 1. Демонстрация работы кода

Изображение выглядит как текст, монитор, внутренний, снимок экрана

Автоматически созданное описание

Рисунок 2. Демонстрация работы кода

Изображение выглядит как текст, монитор, внутренний, компьютер

Автоматически созданное описание

Рисунок 3. Демонстрация работы кода

Изображение выглядит как текст, снимок экрана, монитор, внутренний

Автоматически созданное описание

Рисунок 4. Демонстрация работы кода

Изображение выглядит как текст, монитор, внутренний, снимок экрана

Автоматически созданное описание

Рисунок 5. Демонстрация работы кода

Изображение выглядит как текст, внутренний, монитор, электроника

Автоматически созданное описание

Рисунок 6. Демонстрация работы кода

Изображение выглядит как текст, монитор, внутренний, снимок экрана

Автоматически созданное описание

Рисунок 7. Демонстрация работы кода

**Контрольные вопросы:**

**1.Существуют следующие порождающие шаблоны:**

* простая фабрика (Simple Factory)
* фабричный метод (Factory Method)
* абстрактная фабрика (Abstract Factory)
* строитель (Builder) прототип (Prototype)
* одиночка (Singleton)

**2.Шаблоны проектирования — это руководства по решению повторяющихся проблем.**

**Вывод**

Я научился пользоваться порождающими шаблонами

**Лабораторная работа №12**

**Использование поведенческих шаблонов**

**Цель работы**: изучить поведенческие шаблоны.

**Оборудование**: персональный компьютер.

Изображение выглядит как текст, внутренний, монитор, ноутбук

Автоматически созданное описание

Рисунок 1. Пример работы observer

Изображение выглядит как текст, монитор, внутренний, черный

Автоматически созданное описание

Рисунок 2. Пример работы visitor

Изображение выглядит как текст, монитор, внутренний, снимок экрана

Автоматически созданное описание

Рисунок 3. Пример работы шаблонного метода

**Контрольные вопросы:**

**1.Что такое поведенческие шаблоны?**

поведенческие шаблоны — шаблоны проектирования, определяющие алгоритмы и способы реализации взаимодействия различных объектов и классов

**2.Виды поведенческих шаблонов**

Поведенческие шаблоны: цепочка обязанностей (Chain of Responsibility); команда (Command); итератор (Iterator); посредник (Mediator); хранитель (Memento); наблюдатель (Observer); посетитель (Visitor); стратегия (Strategy); состояние (State); шаблонный метод (Template Method).

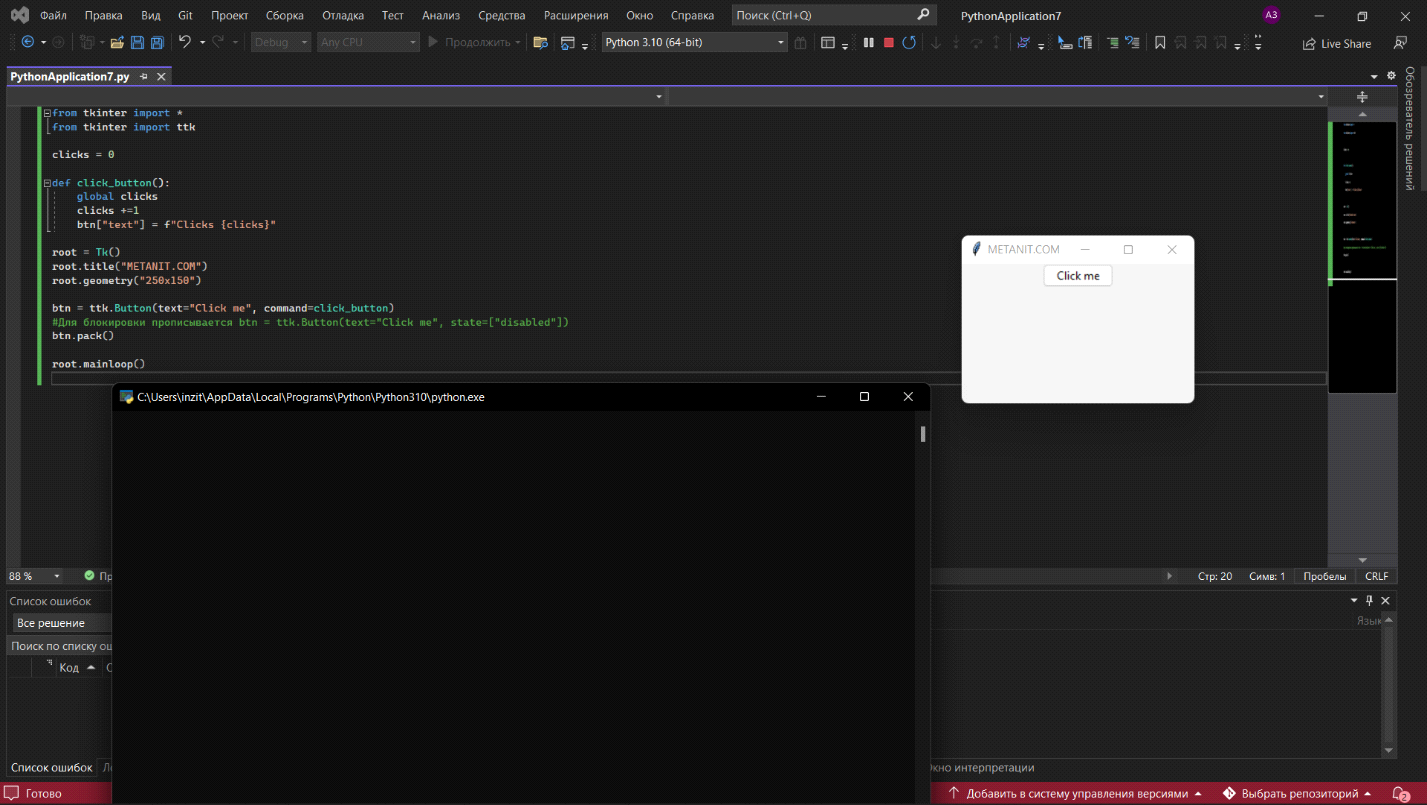
**Вывод**

Я изучил поведенческие шаблоны

**Лабораторная работа №13**

**Разработка приложения с использованием текстовых компонентов**

**Цель работы**: Разработать оконное приложение с использованием текстовых компонентов.



**Контрольные вопросы**

**1.Перечислите основные компоненты интерфейса в tkinter**

* + 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

**2.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.

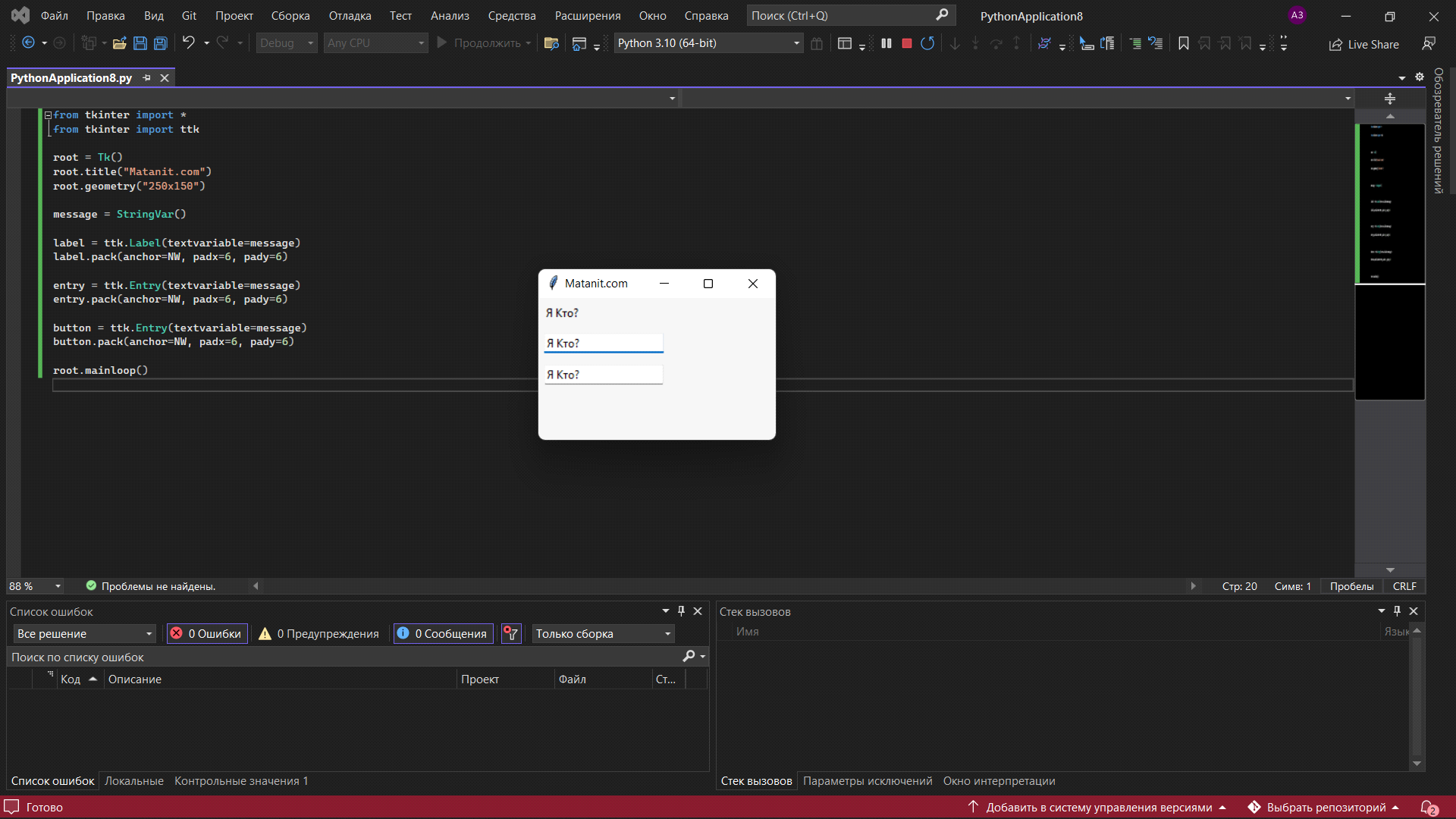
**Вывод**

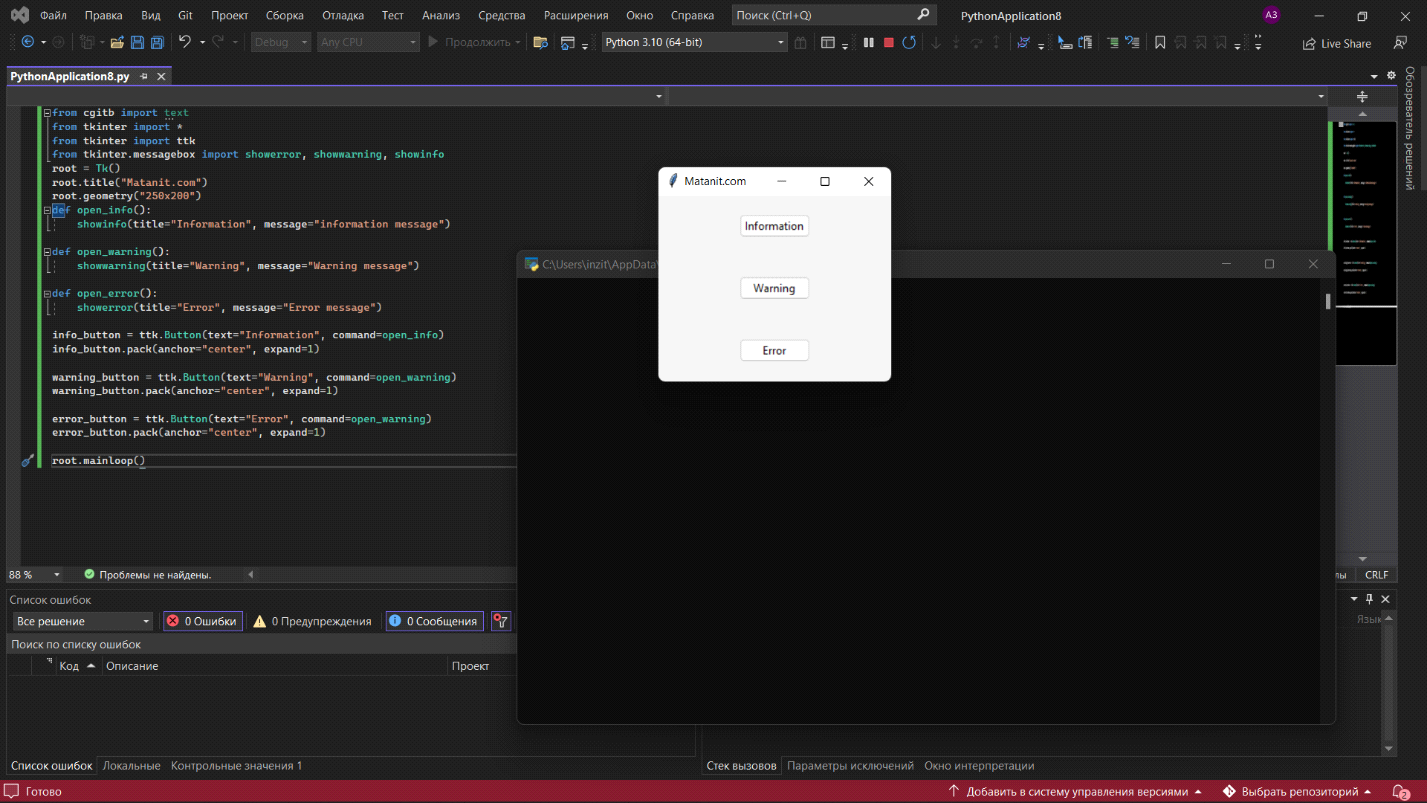
Я разработал оконное приложение с использованием текстовых компонентов.

**Лабораторная работа №14**

**Разработка приложения с несколькими формами**

**Цель работы:** Научиться разрабатывать приложение с несколькими формами.

(Привязка виджетов к переменным)

(MessageBox)

**Контрольные вопросы**

**1. Методы отображения новой формы?**

* anchor: устанавливает позиционирование текста
* background: фоновый цвет
* borderwidth: толщина границы метки
* cursor: курсор указателя мыши при наведении на метку
* font: шрифт текста
* foreground: цвет текста
* height: высота виджета
* image: ссылка на изображение, которое отображается на метке
* justify: устанавливает выравнивание текста. Значение LEFT выравнивает текст по левому краю, CENTER - по центру, RIGHT - по правому краю
* pading: отступы от границ вилжета до его текста
* relief: определяет тип границы, по умолчанию значение FLAT
* text: устанавливает текст метки
* textvariable: устанавливает привязку к элементу StringVar
* underline: указывает на номер символа в тексте кнопки, который подчеркивается. По умолчанию значение -1, то есть никакой символ не подчеркивается
* width: ширина виджета
* wraplength: при положительном значении строки текста будут переносится для вмещения в пространство виджета

**2. С помощью какого метода получается значение поля Entry**

* insert(index, str): вставляет в текстовое поле строку по определенному индексу
* get(): возвращает введенный в текстовое поле текст
* delete(first, last=None): удаляет символ по индексу first. Если указан параметр last, то удаление производится до индекса last. Чтобы удалить до конца, в качестве второго параметра можно использовать значение END.
* focus(): установить фокус на текстовое поле

**Вывод**

Я научился разрабатывать приложение с несколькими формами.

**Лабораторная работа №15**

**Разработка приложения с невизуальными компонентами**

**Цель работы:** Научиться использовать не визуальные компоненты.

**Задания:** Реализуйте программный код из поясняющих примеров, сделайте скриншот результатов и оформите отчет, доработайте интерфейс в рамках вашего индивидуального задания, забавив виджеты кнопок использовав один из методов позиционирования.

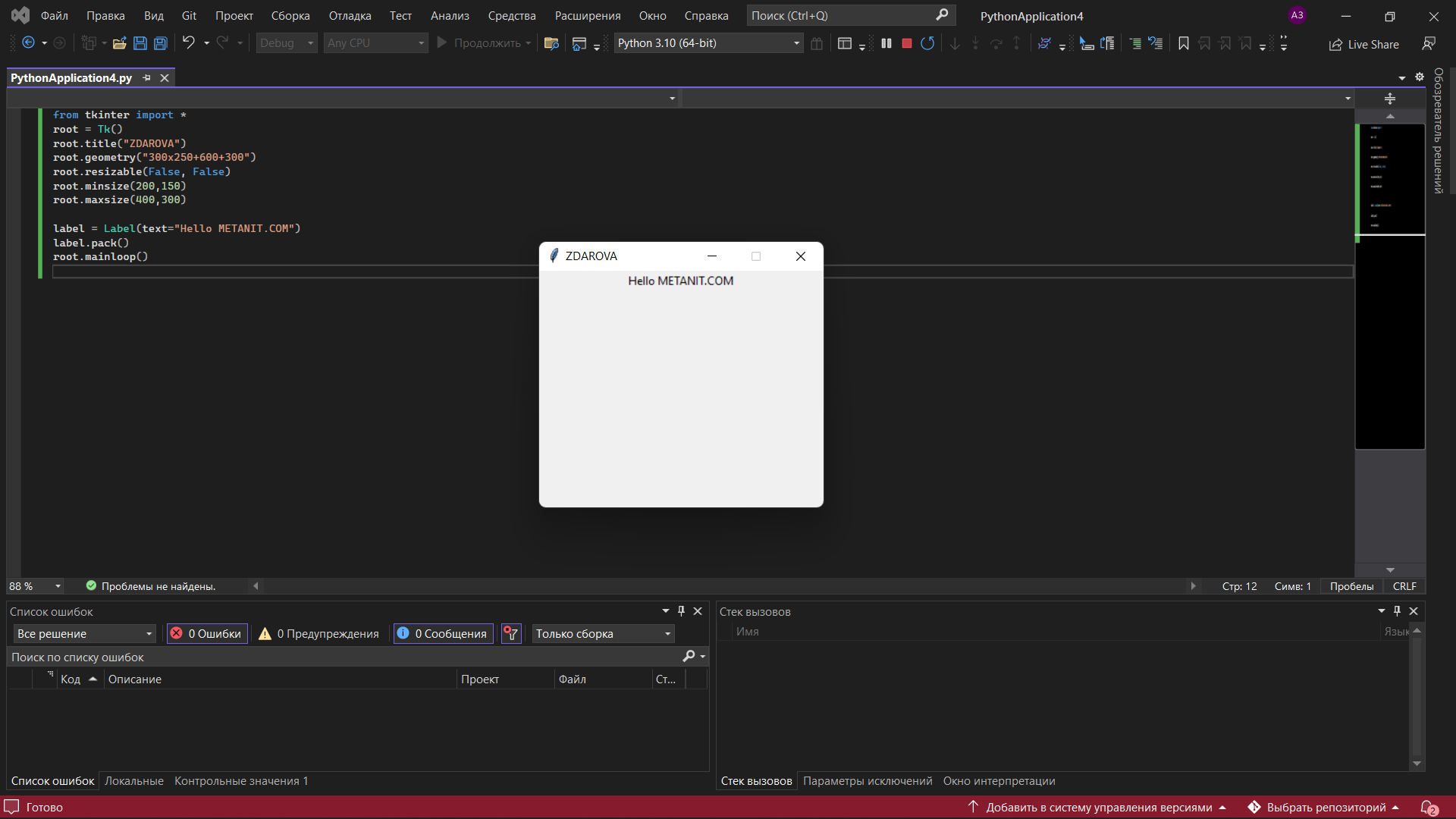


Рисунок 1 – Листинг кода программы

Изображение выглядит как текст, снимок экрана, монитор, внутренний

Автоматически созданное описание

Рисунок 2 – Результат работы программы

**Контрольные вопросы:**

**1.Перечислите основные способы позиционирования элементов интерфейса**

Статическое позиционирование — это умолчание, которое получает каждый элемент, что всего лишь значит "поставить элемент в его нормальное положение в потоке макета документа — ничего особенного для рассмотрения". Относительное позиционирование - модифицирование окончательное положение позиционируемого объекта, занявшего своё место в макете нормального потока. [Абсолютное позиционирование](https://developer.mozilla.org/ru/docs/Learn/CSS/CSS_layout/Positioning#%D0%B0%D0%B1%D1%81%D0%BE%D0%BB%D1%8E%D1%82%D0%BD%D0%BE%D0%B5_%D0%BF%D0%BE%D0%B7%D0%B8%D1%86%D0%B8%D0%BE%D0%BD%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) -  элемента «отделяется» от потока документа и перестает в нем существовать.

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

Параметр expand.

**Вывод**

Я научился использовать не визуальные компоненты.

**Лабораторная работа №22**

**Разработка интерфейса пользователя**

**Цель работы**: Получение навыков разработки хранилищ данных с использованием реляционных СУБД.

Изображение выглядит как текст, монитор, снимок экрана, внутренний

Автоматически созданное описание

**Контрольные вопросы:**

1.Пользовательский интерфейс — средства, позволяющие пользователю эффективно взаимодействовать с устройствами компьютера достаточно удобным для себя образом. Визуально, пользовательский интерфейс представляет собой набор пиктограмм (иконок) для различных программ, кнопки для различных действий и окна. в которых отображаются данные при работе с той или иной программой.

**Вывод**

Яполучил 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.

**Оборудование:** персональный компьютер.

Изображение выглядит как текст, монитор, снимок экрана, компьютер

Автоматически созданное описание

Рисунок 1. Демонстрация кода

Изображение выглядит как текст, монитор, снимок экрана, электроника

Автоматически созданное описание

Рисунок 2. Демонстрация кода

Изображение выглядит как текст, монитор, внутренний, снимок экрана

Автоматически созданное описание

Рисунок 3. Демонстрация работы кода

**Контрольные вопросы**

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

Компактная встраиваемая СУБД

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

MySQL. Она предоставляет отличный набор инструментов для более простой (в сравнении с серверными БД) обработки любых видов данных.

**Вывод**

Я получил навыки разработки хранилищ данных с использованием реляционных СУБД

**Практическая работа №1**

**Изучение и настройка системы контроля версий**

**Цель работы:** Научиться работать и настраивать систему контроля версий

**Задание:**

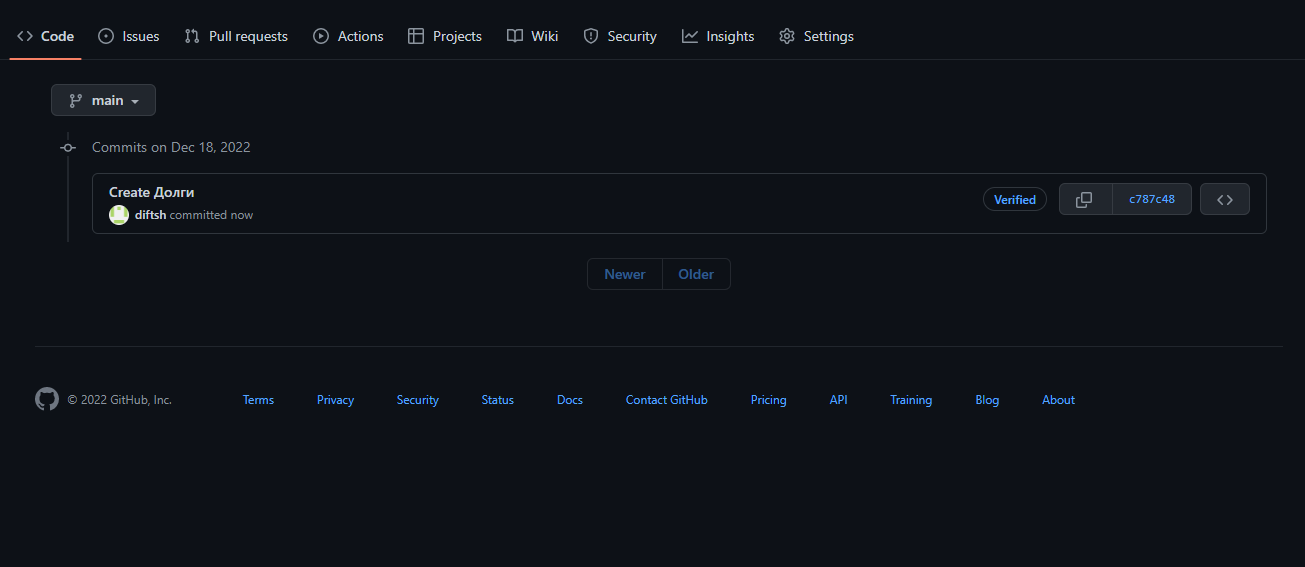


Рисунок 1 – Репозиторий

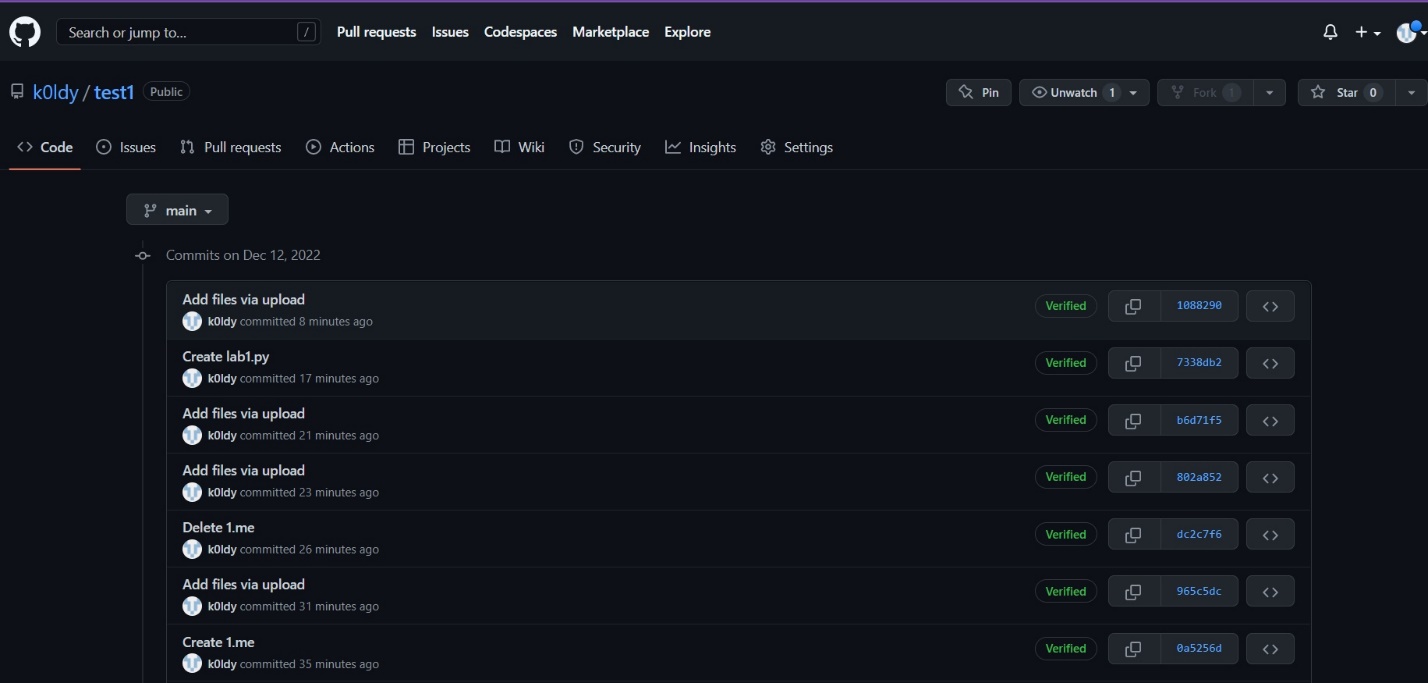


Рисунок 2 – Коммиты

**Контрольные вопросы**

**1.Принцип работы системы контроля версий?**

Система контроля версий страхует от ошибок и возвращает код в то состояние, когда всё работало.

Контрольные точки называются коммитами. Один коммит — это пакет изменений, хранящий информацию с добавленными, отредактированными или удалёнными файлами кода. В один коммит принято добавлять не более десяти изменений — так получается длинная история версий, которая позволяет в случае ошибки откатиться с минимальной потерей работоспособного кода.

**Вывод**

Научился работать и настраивать систему контроля версий

**Практическая работа №3**

**Оценка сложности алгоритмов сортировки. Оценка сложности алгоритма поиска**

**Цель работы:** Научиться разрабатывать алгоритмы сортировки и поиска, а также оценивать их сложность.

**Задание:** 1. Разработать алгоритм сортировки и оценить его сложность.

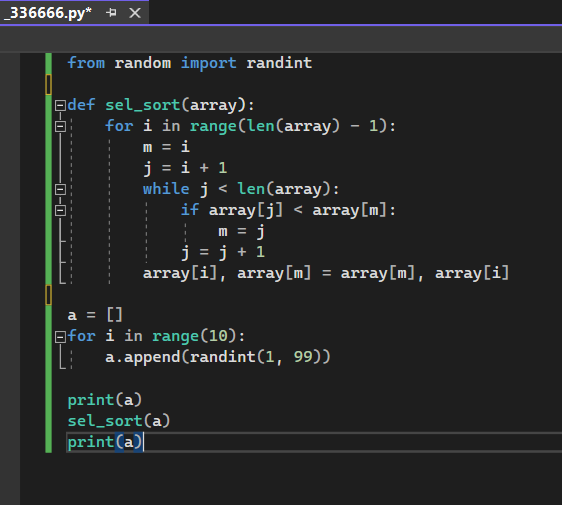


Рисунок 1 – алгоритм сортировки

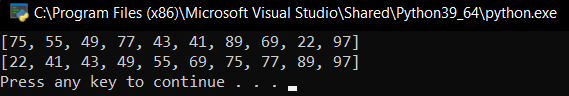


Рисунок 2 – результат алгоритма

Сложность O(n^2)

2. Разработать алгоритм поиска и оценить его сложность.

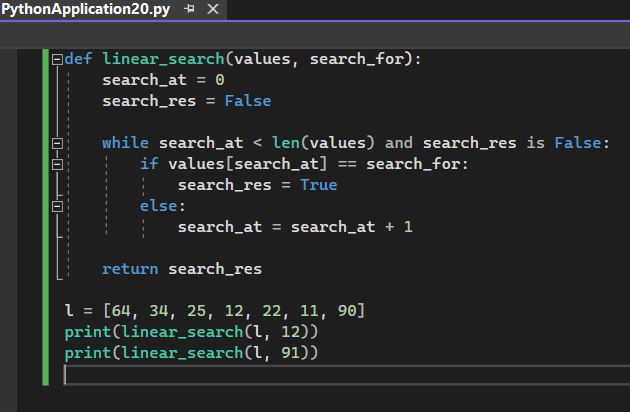


Рисунок 3 – алгоритм поиска

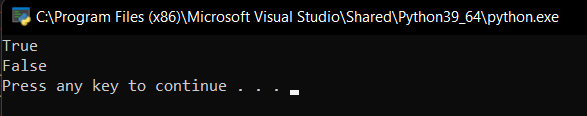


Рисунок 4 – результат алгоритма

Сложность O(n)

**Контрольные вопросы:**

**1. Что такое алгоритм сортировки?**

Это алгоритм для упорядочивания элементов в массиве. В случае, когда элемент в массиве имеет несколько полей, поле, служащее критерием порядка, называется ключом сортировки. На практике в качестве ключа часто выступает число, а в остальных полях хранятся какие-либо данные, никак не влияющие на работу алгоритма.

**2. Что такое алгоритм поиска?**

Это любой алгоритм, который решает задачу поиска, а именно извлекает информацию, хранящуюся в некоторой структуре данных или вычисляемую в пространстве поиска проблемной области, либо с дискретными, либо с непрерывными значениями.

**Вывод**

В ходе практической работы я научился разрабатывать алгоритмы сортировки и поиска, а также оценивать их сложность.

**Практическая работа №4**

**Оценка сложности эвристических алгоритмов**

**Цель работы:** Научиться оценивать сложность эвристических алгоритмов.

**Задание:** 1. Оценить сложность эвристического алгоритма.

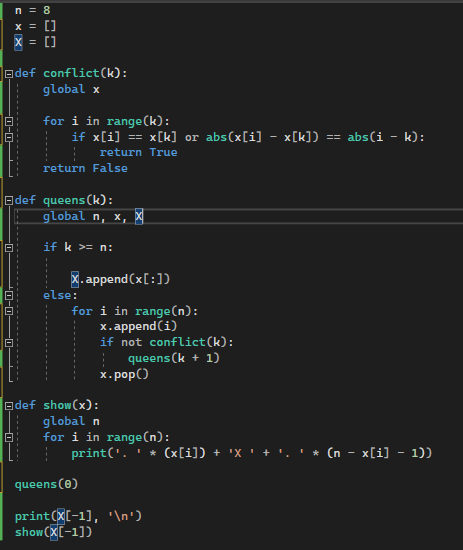


Рисунок 1 – эвристический алгоритм

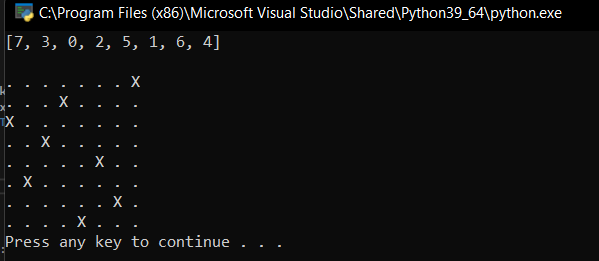


Рисунок 2 – результат кода

Сложность O(n^2)

**Контрольные вопросы:**

**1. Что такое эвристический алгоритм?**

Это алгоритм решения задачи, правильность которого для всех возможных случаев не доказана, но про который известно, что он даёт достаточно хорошее решение в большинстве случаев.

**2. Что такое формулы оценки сложности алгоритмов?**

Они позволяют ориентироваться, какие структуры данных и архитектуры стоит использовать в тех или иных ситуациях.

**Вывод**

В ходе практической работы я научился оценивать сложность эвристических алгоритмов.

**Практическая работа №5**

**Оценка сложности рекурсивных алгоритмов**

**Цель работы:** Научиться разрабатывать рекурсивные алгоритмы и оценить их сложность.

**Задание:**

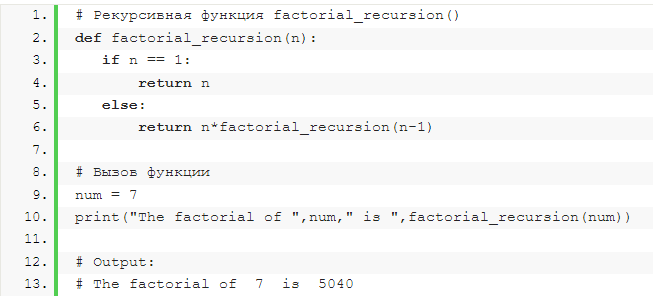


Рисунок 1 – Рекурсивный алгоритм (Сложность O(n))

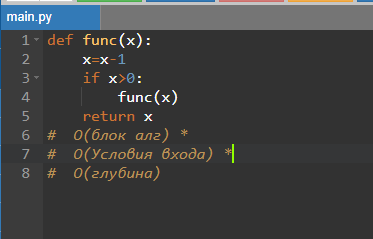


Рисунок 2 – Схема

**Контрольные вопросы:**

**1.Что такое рекурсивный алгоритм?**

Это алгоритм, в описании которого прямо или косвенно содержится обращение к самому себе.

**Практическая работа №6**

**Оформление документации на программные средства**

**Цель работы:** Научиться оформлять документацию на программные средства.

**Задание:** Оформить документацию на программный код в соответствии с основными форматами документирования: Строки документации Google, reStructuredText, Строки документации NumPy/SciPy, Epytext.

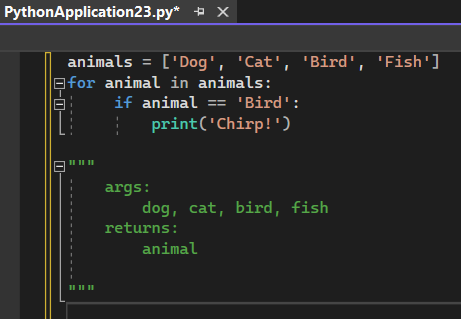


Рисунок 1 – пример GOOGLE DOCSTRINGS

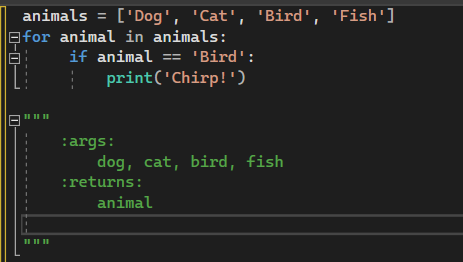


Рисунок 2 – пример RESTRUCTUREDTEXT EXAMPLE



Рисунок 3 – пример NUMPY/SCIPY DOCSTRINGS EXAMPLE

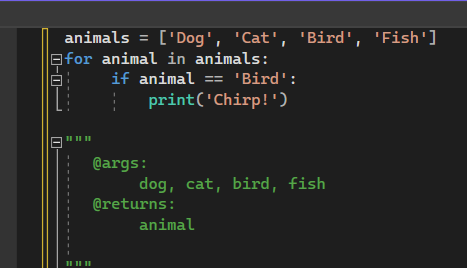


Рисунок 4 – пример EPYTEXT EXAMPLE

**Контрольные вопросы:**

**1. Что такое стандарт PEP8 и PEP257?**

PEP8 – это документ, описывающий соглашение о том, как писать код для языка Python, включая стандартную библиотеку, входящую в состав Python.

PEP 257 – описывает соглашения, связанные со строками документации Python, рассказывает о том, как нужно документировать Python код.

**2. Перечислите основные форматы документирования кода?**

* строки документации Google.
* reStructuredText.
* строки документации NumPy/SciPy.
* epytext.

**Вывод**

Я научился оформлять документацию на программные средства.