Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт перспективной инженерии

Департамент цифровых, робототехнических систем и электроники

**ОТЧЕТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ №4.2**

**дисциплины «Объектно-ориентированное программирование»**

|  |  |  |
| --- | --- | --- |
|  | Выполнил:  Бересланов Рамазан  3 курс, группа ПИЖ-б-о-22-1,  09.03.04 «Программная инженерия», направленность (профиль) «Разработка и сопровождение программного обеспечения», очная форма обучения  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | |
|  | Руководитель практики:  Воронкин Р.А., доцент департамента цифровых, робототехнических систем и электроники  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | |
|  |  |  |

Отчет защищен с оценкой \_\_\_\_\_\_\_\_\_\_\_ Дата защиты\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ставрополь, 2024 г.

**Цель работы**: приобретение навыков по перегрузке операторов при написании программ с помощью языка программирования Python версии 3.x.

Ход работы:

1. Изучил теоретический материал.
2. Создал общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и язык программирования Python.

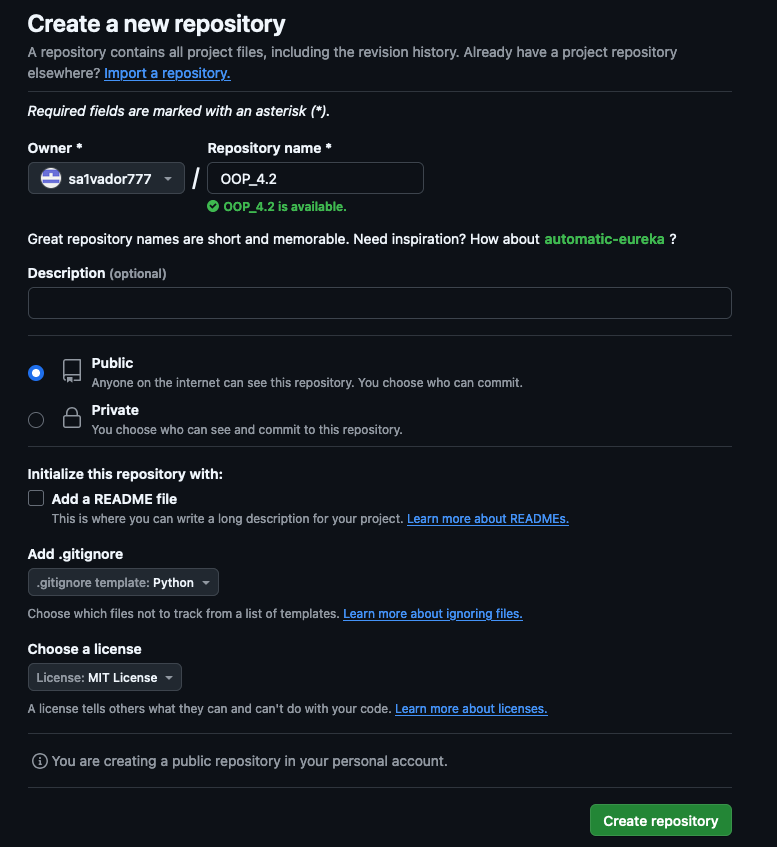


Рисунок 1 – Создание репозитория

1. Склонировал репозиторий на ПК.

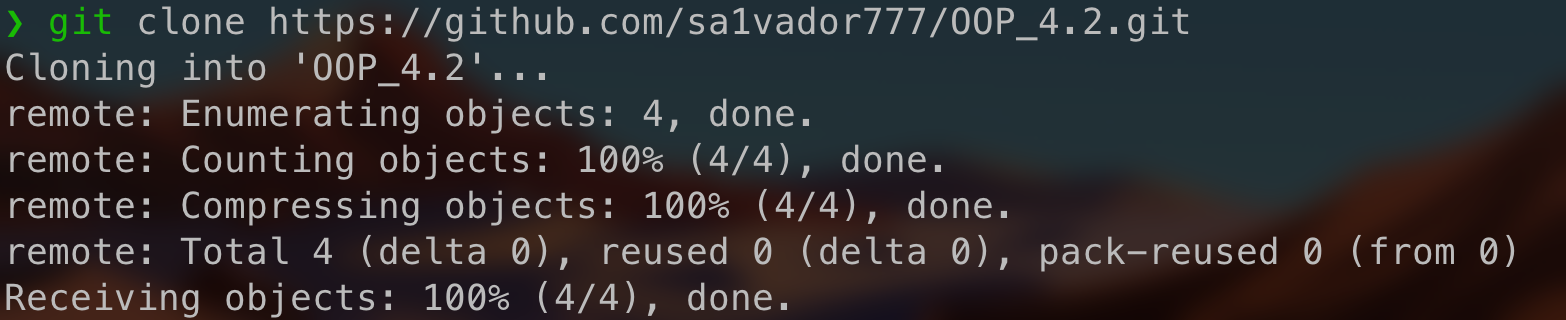


Рисунок 2 – Клонирование репозитория

1. Инициализировал модель ветвления git flow.

Изображение выглядит как текст, снимок экрана, Цвет электрик

Автоматически созданное описание

Рисунок 3 – Инициализация git-flow

1. Создал виртуальное окружение poetry

Изображение выглядит как текст, снимок экрана, Цвет электрик, Шрифт

Автоматически созданное описание

Рисунок 4 – Создание окружения poetry

Изображение выглядит как текст, снимок экрана, Цвет электрик, Шрифт

Автоматически созданное описание

Рисунок 5 – Активация окружения

1. **Задание 1.**

Выполнить индивидуальное задание 1 лабораторной работы 4.1, максимально задействовав имеющиеся в Python средства перегрузки операторов.

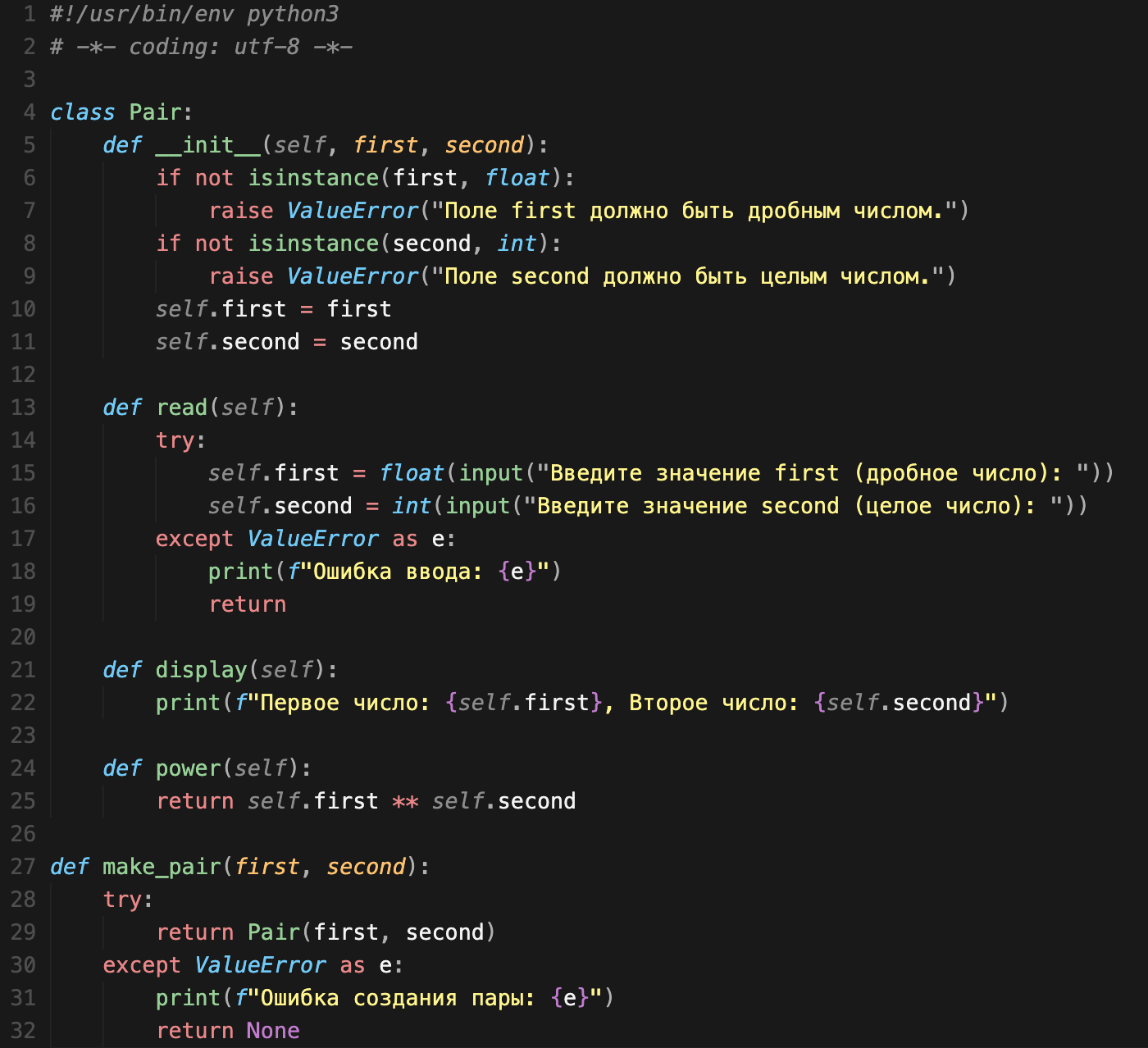


Рисунок 6 – Основной код индивидуального задания №1 лабораторной работы 4.1

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 7 – Добавленные перегрузки операторов

1. **Задание 2. Вариант - 1**

Создать класс Decimal для работы с беззнаковыми целыми десятичными числами, используя для представления числа список из 100 элементов типа int, каждый из которых является десятичной цифрой. Младшая цифра имеет меньший индекс (единицы — в нулевом элементе списка). Реальный размер списка задается как аргумент конструктора инициализации. Реализовать арифметические операции, аналогичные встроенным для целых и операции сравнения.

Дополнительно к требуемым в заданиях операциям перегрузить операцию индексирования []. Максимально возможный размер списка задать константой. В отдельном поле size должно храниться максимальное для данного объекта количество элементов списка; реализовать метод size(), возвращающий установленную длину. Если количество элементов списка изменяется во время работы, определить в классе поле count. Первоначальные значения size и count устанавливаются конструктором. В тех задачах, где возможно, реализовать конструктор инициализации строкой.

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 8 – Основной код второго индивидуального задания

­­­ Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 9 – Перегрузка основных операторов­

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 10 – Перегрузка операторов сравнения и приведение к строке

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунки 11, 12 – Демонстрация работы программы

**Ответы на контрольные вопросы:**

1. **Какие средства существуют в Python для перегрузки операций?**

В Python перегрузка операций осуществляется с помощью **магических методов** (или методов "dunder" — double underscore). Эти методы имеют специальный синтаксис и позволяют изменять поведение стандартных операторов для пользовательских классов. Например:

* + \_\_add\_\_(self, other) — перегрузка оператора сложения +
  + \_\_sub\_\_(self, other) — перегрузка оператора вычитания -
  + \_\_mul\_\_(self, other) — перегрузка оператора умножения \*

1. **Какие существуют методы для перегрузки арифметических операций и операций отношения в языке Python?**

Методы для перегрузки арифметических операций:

* + \_\_add\_\_(self, other) — сложение
  + \_\_sub\_\_(self, other) — вычитание
  + \_\_mul\_\_(self, other) — умножение
  + \_\_truediv\_\_(self, other) — деление
  + \_\_floordiv\_\_(self, other) — целочисленное деление
  + \_\_mod\_\_(self, other) — остаток от деления
  + \_\_pow\_\_(self, other) — возведение в степень

Методы для перегрузки операций отношения:

* + \_\_eq\_\_(self, other) — равно ==
  + \_\_ne\_\_(self, other) — не равно !=
  + \_\_gt\_\_(self, other) — больше >
  + \_\_lt\_\_(self, other) — меньше <
  + \_\_ge\_\_(self, other) — больше или равно >=
  + \_\_le\_\_(self, other) — меньше или равно <=

1. **В каких случаях будут вызваны следующие методы: \_\_add\_\_, \_\_iadd\_\_ и \_\_radd\_\_? Приведите примеры.**

* \_\_add\_\_(self, other): вызывается при использовании оператора + между двумя объектами. Например: **result = obj1 + obj2**
* \_\_iadd\_\_(self, other): вызывается при использовании оператора +=. Например: **obj1 += obj2**
* \_\_radd\_\_(self, other): вызывается, когда оператор + применяется с объектом, у которого нет метода \_\_add\_\_. Например, если other (правый операнд) имеет метод \_\_radd\_\_, он будет вызван, если obj1 (левый операнд) не имеет метода \_\_add\_\_.

1. **Для каких целей предназначен метод \_\_new\_\_? Чем он отличается от метода \_\_init\_\_?**

* Метод \_\_new\_\_ предназначен для создания нового экземпляра класса. Он вызывается перед \_\_init\_\_ и возвращает новый объект. Используется, когда нужно управлять созданием объекта, например, в случае с метаклассами или синглтонами.
* Метод \_\_init\_\_ — это конструктор, который инициализирует уже созданный объект. В \_\_init\_\_ задаются атрибуты экземпляра.

В общем, \_\_new\_\_ отвечает за создание, а \_\_init\_\_ — за инициализацию.

1. **Чем отличаются методы \_\_str\_\_ и \_\_repr\_\_?**

* Метод \_\_str\_\_ предназначен для создания строкового представления объекта, которое будет читабельно для пользователя. Используется, когда вызывается print() или str().
* Метод \_\_repr\_\_ предназначен для создания представления объекта, которое может быть использовано для восстановления объекта. Обычно это более детальное и формальное представление, подходящее для разработчиков. Используется при вызове repr() и в интерактивной среде Python.