

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 №2
дисциплины «Объектно-ориентированное программирование»

Выполнил:
Евдаков Евгений Владимирович
3 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и
вычислительная техника»,
направленность (профиль)
«Программное обеспечение средств
вычислительной техники и
автоматизированных систем», очная
форма обучения

(подпись)

Проверил:
Воронкин Р. А., доцент департамента
цифровых, робототехнических систем
и электроники института
перспективной инженерии

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2024 г.

Тема: перегрузка операторов в языке Python.

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

Ход работы:

Задание 1. Создал общедоступный репозиторий на GitHub, в котором использована лицензия MIT и язык программирования Python, также добавил файл .gitignore с необходимыми правилами. Клонировал свой репозиторий на свой компьютер. Организовал свой репозиторий в соответствии с моделью ветвления git-flow, появилась новая ветка develop в которой буду выполнять дальнейшие задачи.

```
C:\Users\Gaming-PC>git clone https://github.com/EvgenyEvdakov/Laba_4.2.git
Cloning into 'Laba_4.2'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (5/5), done.
```

Рисунок 1. Клонирование репозитория

Задание 2. Создал виртуальное окружение conda и активировал его, также установил необходимые пакеты isort, black, flake8.

```
(base) PS C:\Users\Gaming-PC> cd C:\Users\Gaming-PC\Laba_4.2
(base) PS C:\Users\Gaming-PC\Laba_4.2> conda create -n 4.2 python=3.10
Retrieving notices: ...working... done
Collecting package metadata (current_repodata.json): done
Solving environment: done

==> WARNING: A newer version of conda exists. <==
  current version: 23.1.0
  latest version: 24.9.1

Please update conda by running

    $ conda update -n base -c defaults conda

Or to minimize the number of packages updated during conda update use

    conda install conda=24.9.1

## Package Plan ##

  environment location: C:\Users\Gaming-PC\.conda\envs\4.2

  added / updated specs:
    - python=3.10

The following NEW packages will be INSTALLED:
```

Рисунок 2. Создание виртуального окружения

Задание 3. Создал проект PyCharm в папке репозитория. Приступил к работе с примером. Добавил новый файл primer1.py.

Условие примера: Изменить класс Rational из примера 1 лабораторной работы 4.1, используя перегрузку операторов.

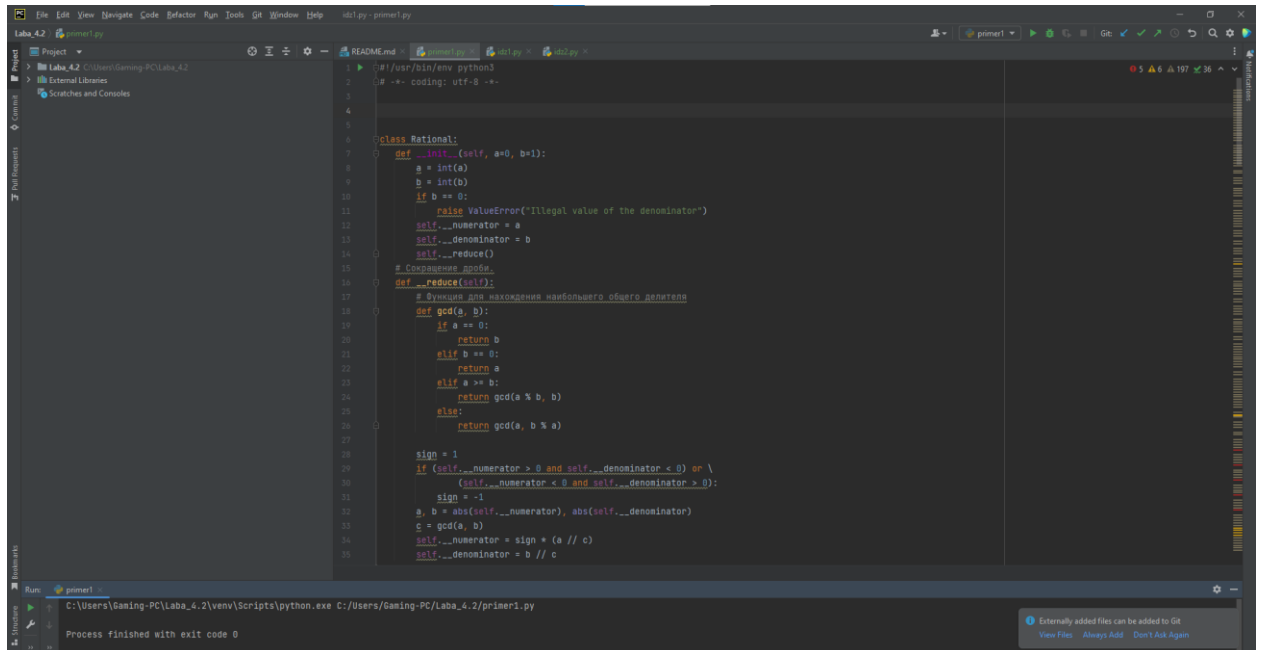


Рисунок 3. Выполнение первого примера

Задание 4.

Индивидуальное задание

Вариант 6

Создал новый файл под названием `idz1.py`.

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

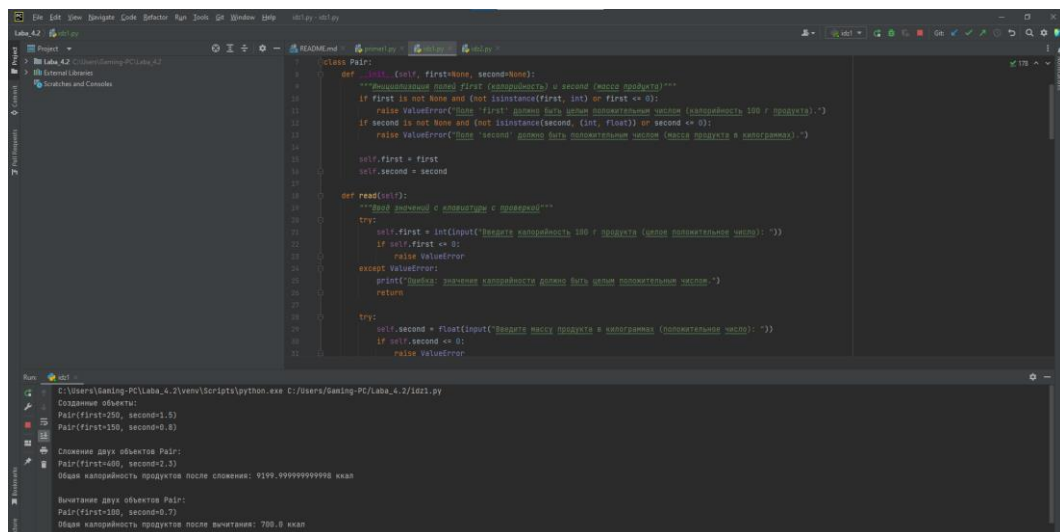
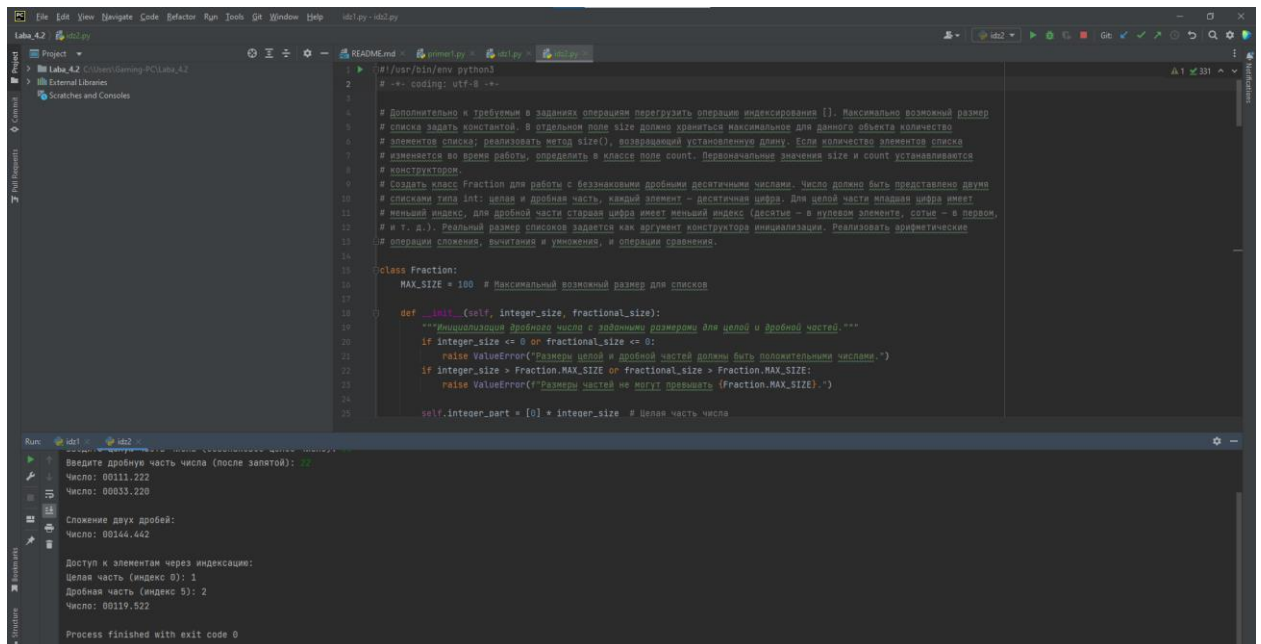


Рисунок 4. Выполнение первого индивидуального задания

Создал новый файл под названием idz2.py.

Условие задания: 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.

Создать класс Fraction для работы с беззнаковыми дробными десятичными числами. Число должно быть представлено двумя списками типа int: целая и дробная часть, каждый элемент — десятичная цифра. Для целой части младшая цифра имеет меньший индекс, для дробной части старшая цифра имеет меньший индекс (десятые — в нулевом элементе, сотые — в первом, и т. д.). Реальный размер списков задается как аргумент конструктора инициализации. Реализовать арифметические операции сложения, вычитания и умножения, и операции сравнения.



```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 # Дополнительно к требуемым в заданиях операциям перегрузить операцию индексирования []. Максимально возможный размер
5 # списка задать константой. В отдельном поле size должно храниться максимальное для данного объекта количество
6 # элементов списка; реализовать метод size(), возвращающий установленную длину. Если количество элементов списка
7 # изменяется во время работы, определить в классе поле count. Первоначальные значения size и count устанавливаются
8 # конструктором.
9
10 # Создать класс Fraction для работы с беззнаковыми дробными десятичными числами. Число должно быть представлено двумя
11 # списками типа int: целая и дробная часть, каждый элемент — десятичная цифра. Для целой части младшая цифра имеет
12 # меньший индекс, для дробной части старшая цифра имеет меньший индекс (десятые — в нулевом элементе, сотые — в первом,
13 # и т. д.). Реальный размер списков задается как аргумент конструктора инициализации. Реализовать арифметические
14 # операции сложения, вычитания и умножения, и операции сравнения.
15
16 class Fraction:
17     MAX_SIZE = 100 # Максимальный возможный размер для списков
18
19     def __init__(self, integer_size, fractional_size):
20         """Инициализация дробного числа с заданными размерами для целой и дробной частей."""
21         if integer_size <= 0 or fractional_size <= 0:
22             raise ValueError("Размеры целой и дробной частей должны быть положительными числами.")
23         if integer_size > Fraction.MAX_SIZE or fractional_size > Fraction.MAX_SIZE:
24             raise ValueError("Размеры частей не могут превышать (Fraction.MAX_SIZE).")
25
26         self.integer_part = [0] * integer_size # Целая часть числа
27
28
29 # Ввод данных
30 integer_size = int(input("Введите размер целой части (максимум 100): "))
31 fractional_size = int(input("Введите размер дробной части (максимум 100): "))
32
33 # Создание объекта Fraction
34 f = Fraction(integer_size, fractional_size)
35
36 # Ввод цифр
37 print("Введите цифры для целой части:")
38 for i in range(integer_size):
39     f.integer_part[i] = int(input(f"Цифра {i}: "))
40
41 print("Введите цифры для дробной части:")
42 for i in range(fractional_size):
43     f.integer_part[-i-1] = int(input(f"Цифра {i}: "))
44
45 # Вывод результата
46 print("Число: ", end="")
47 for i in range(integer_size):
48     print(f.integer_part[i], end="")
49 print(".", end="")
50 for i in range(fractional_size):
51     print(f.integer_part[-i-1], end="")
52 print()
53
54 # Тестирование
55 print("Сложение двух дробей:")
56 f1 = Fraction(1, 2)
57 f2 = Fraction(1, 2)
58 f3 = f1 + f2
59 print("Число: ", end="")
60 for i in range(f3.integer_size):
61     print(f3.integer_part[i], end="")
62 print(".", end="")
63 for i in range(f3.fractional_size):
64     print(f3.integer_part[-i-1], end="")
65 print()
```

Рисунок 5. Выполнение второго индивидуального задания

Задание 5.

После выполнения работы на ветке develop, слил ее с веткой main и отправил изменения на удаленный сервер. Создал файл environment.yml и деактивировал виртуальное окружение.

```
(4.2) PS C:\Users\Gaming-PC\Laba_4.2> conda env export > environment
(4.2) PS C:\Users\Gaming-PC\Laba_4.2> conda deactivate
```

Рисунок 6. Деактивация ВО

Ссылка: https://github.com/EvgenyEvdakov/Laba_4.2

Ответы на контрольные вопросы:

1. Какие средства существуют в Python для перегрузки операций?

В Python поддерживается перегрузка операций с помощью специальных методов (также называемых магическими методами). Эти методы начинаются и заканчиваются двумя подчеркиваниями, и они позволяют переопределять поведение операторов, таких как +, -, *, ==, <, и т.д., для объектов пользовательских классов.

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

Для перегрузки арифметических операций используются следующие методы:

- `__add__(self, other)` — перегрузка оператора + (сложение)
- `__sub__(self, other)` — перегрузка оператора - (вычитание)
- `__mul__(self, other)` — перегрузка оператора * (умножение)
- `__truediv__(self, other)` — перегрузка оператора / (деление)
- `__floordiv__(self, other)` — перегрузка оператора // (целочисленное деление)
- `__mod__(self, other)` — перегрузка оператора % (остаток от деления)
- `__pow__(self, other)` — перегрузка оператора ** (возведение в степень)
- `__and__(self, other)` — перегрузка оператора & (логическое "И")
- `__or__(self, other)` — перегрузка оператора | (логическое "ИЛИ")

- `__xor__(self, other)` — перегрузка оператора \wedge (логическое исключающее "ИЛИ")

Методы для перегрузки операций сравнения (отношения):

- `__eq__(self, other)` — перегрузка оператора `==`
- `__ne__(self, other)` — перегрузка оператора `!=`
- `__lt__(self, other)` — перегрузка оператора `<`
- `__le__(self, other)` — перегрузка оператора `<=`
- `__gt__(self, other)` — перегрузка оператора `>`
- `__ge__(self, other)` — перегрузка оператора `>=`

3. В каких случаях будут вызваны следующие методы: `__add__`, `__iadd__` и `__radd__`? Приведите примеры.

`__add__` (сложение): Этот метод вызывается, когда используется оператор `+` для объектов одного и того же типа (например, `obj1 + obj2`).

`__iadd__` (сложение с присваиванием): Этот метод вызывается при использовании оператора `+=`. Он изменяет объект на месте, если это возможно.

`__radd__` (реверсивное сложение): Этот метод вызывается, когда левый операнд не поддерживает сложение с правым операндом, и происходит попытка обратного вызова. Например, `other + obj`, где `other` не поддерживает `+` с типом `obj`.

4. Для каких целей предназначен метод `__new__`? Чем он отличается от метода `__init__`?

Метод `__new__` отвечает за создание нового экземпляра класса и вызывается до `__init__`. Он используется для управления процессом создания объектов, особенно при наследовании от `immutable` (неизменяемых) типов, таких как `int`, `str`, `tuple`.

- `__new__` создаёт объект и возвращает его.
- `__init__` инициализирует уже созданный объект.

5. Чем отличаются методы `__str__` и `__repr__`?

- `__str__` возвращает "человеко-читаемое" строковое представление объекта. Этот метод вызывается, когда используется функция `str()`, или когда объект выводится с помощью `print()`. Цель — сделать вывод более понятным для конечного пользователя.

- `__repr__` возвращает "машиночитаемое" строковое представление объекта. Этот метод должен возвращать строку, которая (по возможности) позволяет восстановить объект при использовании функции `eval()`. Если метод `__str__` не определён, для вывода объекта вызывается `__repr__`.

Вывод: приобрел навыки по перегрузке операторов при написании программ с помощью языка программирования Python версии 3.x.