МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Кафедра инфокоммуникаций

Отчет по лабораторной работе №2

Перегрузка операторов в языке Python

По дисциплине «Объектно-ориентированное программирование»

Выполнил студент группы ИВТ-б-о-20-1

Галяс Д. И. « »_____20__г.

Подпись студента_____

Работа защищена « »_____20__г.

Проверил Воронкин Р. А. _____

(подпись)

Ставрополь 2022

Цель работы: приобретение навыков по работе с исключениями при написании программ с помощью языка программирования Python версии 3.x.

Ссылка на репозиторий: https://github.com/DIMITRY-GALYAS1/Laba-4.2.git

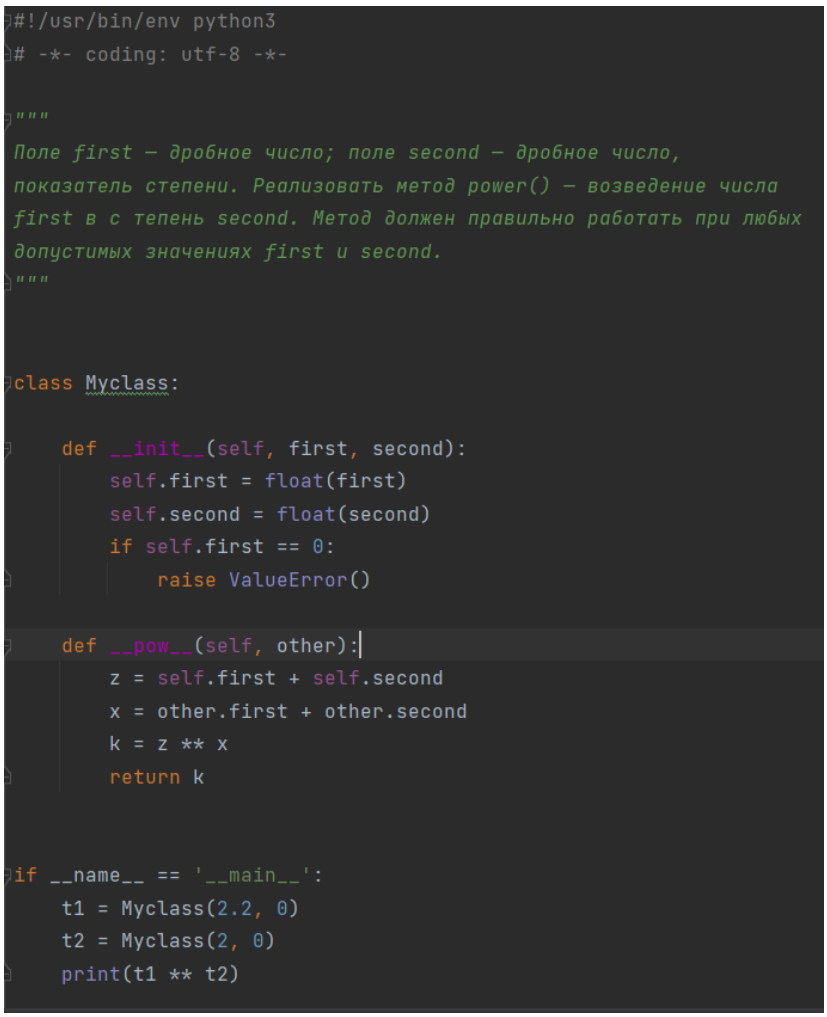
Ход работы:

1. Создал новый репозиторий на github, после клонировал его и создал в папке репозитория новый проект PyCharm.
2. Выполнил пример.

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

class Rational:

    def __init__(self, a=0, b=1):
        a = int(a)
        b = int(b)

        if b == 0:
            raise ValueError("Illegal value of the denominator")

        self.__numerator = a
        self.__denominator = b

        self.__reduce()

    # Сокращение дроби.
    def __reduce(self):
        # Функция для нахождения наибольшего общего делителя
        def gcd(a, b):
            if a == 0:
                return b
            elif b == 0:
                return a
            elif a >= b:
                return gcd(a % b, b)
            else:
                return gcd(a, b % a)

        sign = 1
```

Рисунок 1. Пример 1

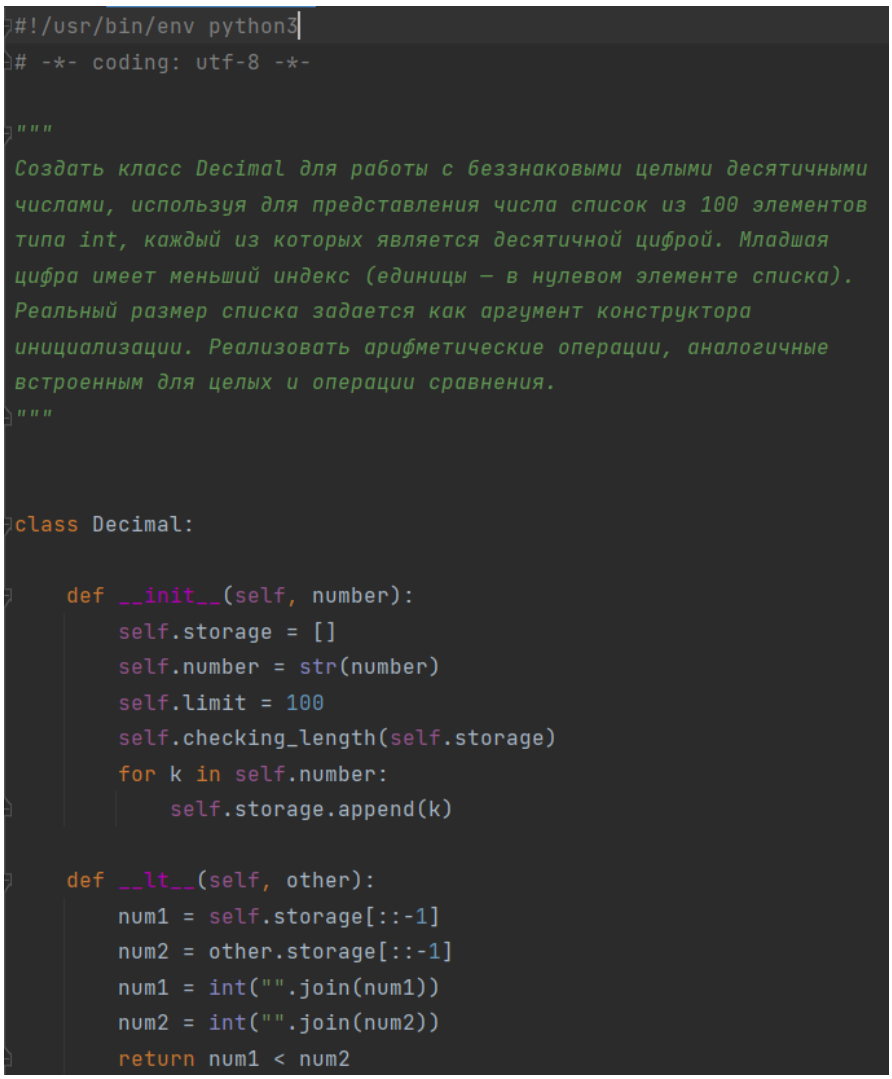
3. Далее приступил к выполнению первого индивидуального задания.



```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

"""
Поле first – дробное число; поле second – дробное число,
показатель степени. Реализовать метод power() – возведение числа
first в с тепень second. Метод должен правильно работать при любых
допустимых значениях first и second.
"""


class Myclass:

    def __init__(self, first, second):
        self.first = float(first)
        self.second = float(second)
        if self.first == 0:
            raise ValueError()

    def __pow__(self, other):
        z = self.first + self.second
        x = other.first + other.second
        k = z ** x
        return k


if __name__ == '__main__':
    t1 = Myclass(2.2, 0)
    t2 = Myclass(2, 0)
    print(t1 ** t2)
```

Рисунок 2. Первое индивидуальное задание

4. Затем выполнил второе индивидуальное задание.



```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

"""
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.
"""


class Decimal:

    def __init__(self, number):
        self.storage = []
        self.number = str(number)
        self.limit = 100
        self.checking_length(self.storage)
        for k in self.number:
            self.storage.append(k)

    def __lt__(self, other):
        num1 = self.storage[::-1]
        num2 = other.storage[::-1]
        num1 = int("".join(num1))
        num2 = int("".join(num2))
        return num1 < num2
```

Рисунок 3. Второе индивидуальное задание

Контрольные вопросы:

1. Какие средства существуют в Python для перегрузки операций? Перегрузка осуществляется при помощи специальных методов. Методы группируются по следующим категориям:

– методы для всех видов операций;

– методы перегрузки операторов работы с коллекциями;

– методы для числовых операций в двоичной форме;

– методы для других операций над числами;

– методы для операций с дескрипторами;

– методы для операций, используемых с диспетчерами контекста.

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

____add__(self, other) - сложение. x + y вызывает x.__add__(y) .

____sub__(self, other) - вычитание (x - y).

____mul__(self, other) - умножение (x * y).

____truediv__(self, other) - деление (x / y).

____floordiv__(self, other) - целочисленное деление (x // y).

____mod__(self, other) - остаток от деления (x % y).

____divmod__(self, other) - частное и остаток (divmod(x, y)).

___pow_(self, other[, modulo]) - возведение в степень (x ** y , pow(x, y[, modulo])).

___lshift_(self, other) - битовый сдвиг влево (x << y).

___rshift_(self, other) - битовый сдвиг вправо (x >> y).

___and_(self, other) - битовое И (x & y).

___xor_(self, other) - битовое ИСКЛЮЧАЮЩЕЕ ИЛИ (x ^ y).

___radd__(self, other) ,

___rsub__(self, other) ,

___rmul_(self, other) ,

__rtruediv_(self, other) ,

___rfloordiv_(self, other) ,

___rmod_(self, other) ,

___rdivmod_(self, other) ,

___rpow_(self, other) ,

___rlshift__(self, other) ,

___rrshift__(self, other) ,

___rand__(self, other) ,

___rxor__(self, other) ,

__ror (self, other) - делают то же самое, что и арифметические операторы, перечисленные выше, но для аргументов, находящихся справа, и только в случае, если длялевого операнда не определён соответствующий метод.

__iadd_(self, other) - += .

___isub_(self, other) - -= .

___imul__(self, other) - *= .

___itruediv__(self, other) - /= .

__ifloordiv__(self, other) - //= .

__imod__(self, other) - %= .

___ipow__(self, other[, modulo]) - **= .

___ilshift__(self, other) - <<= .

___irshift__(self, other) - >>= .

____iand__(self, other) - &= .

___ixor__(self, other) - ^= .

____ior__(self, other) - |= .

3. В каких случаях будут вызваны следующие методы:__add__, ____iadd__и___radd__?

– add_____- a + b

– __iadd___- a += b

____radd_____- Если не получилось вызвать метод__add

4. Для каких целей предназначен метод new? Чем он отличается от метода init?

Метод new используется, когда нужно управлять процессом создания нового экземпляра, а init – когда контролируется его инициализация.

5. Чем отличаются методы str и repr ?

str должен возвращать строковый объект, тогда как repr может возвращать любое выражение в Python.

Вывод: в ходе выполнения лабораторной работы приобрел навыки по работе с исключениями при написании программ с помощью языка программирования Python версии 3.x.