

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

**Кафедра инфокоммуникаций**

**Объектно-ориентированное программирование**

**Отчет по лабораторной работе №4.2**

Перегрузка операторов в языке Python

Выполнил студент группы

ИВТ-б-о-21-1

Богдашов А.В. « » \_\_\_\_\_ 20\_\_ г.

Подпись студента \_\_\_\_\_

Работа защищена « » \_\_\_\_\_ 20\_\_ г.

Проверил доцент

Кафедры инфокоммуникаций, старший  
преподаватель

Воронкин Р.А.

\_\_\_\_\_  
(подпись)

Ставрополь 2023

## Наследование и полиморфизм в языке Python.

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

### Порядок выполнения работы:

#### Задание 1.

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

#### Код программы:

```
# -*- coding: utf-8 -*-

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

class Pair:
    def __init__(self, first, second):
        if not isinstance(first, int) or not isinstance(second, int):
            raise TypeError("Оба поля (first и second) должны быть целыми
числами")

        if second == 0:
            raise ValueError("Знаменатель (second) не может быть равен нулю")

        self.first = first
        self.second = second

    def __floordiv__(self, other):
        """
        Перегрузка оператора для выделения целой части от деления.
        """
        if isinstance(other, Pair):
            if other.second != 0:
                return self.first // other.second
            else:
                raise ValueError("Невозможно вычислить целую часть при нулевом
знаменателе")
        else:
            raise TypeError("Операнд должен быть объектом класса Pair")

    def __str__(self):
        """
        Перегрузка метода str для вывода дроби.
        """
        return f"({self.first}/{self.second})"

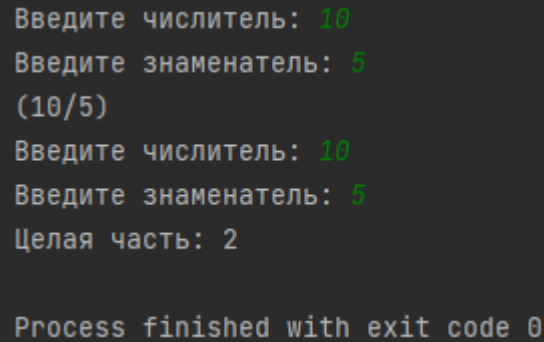
    def display(self):
        print(str(self))

    @classmethod
    def read(cls):
        a = int(input("Введите числитель: "))
        b = int(input("Введите знаменатель: "))
```

```
        return cls(a, b)

if __name__ == "__main__":
    pair = Pair.read()
    pair.display()
    other_pair = Pair.read()
    print("Целая часть:", pair // other_pair)
```

Результат работы программы:



```
Введите числитель: 10
Введите знаменатель: 5
(10/5)
Введите числитель: 10
Введите знаменатель: 5
Целая часть: 2

Process finished with exit code 0
```

Рисунок 1. Результат работы программы

## Задание 2.

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 Hex для работы с беззнаковыми целыми шестнадцатеричными числами, используя для представления числа список из 100 элементов типа int, каждый из которых является шестнадцатеричной цифрой. Младшая цифра имеет меньший индекс. Реальный размер списка задается как аргумент конструктора инициализации. Реализовать арифметические операции, аналогичные встроенным для целых и операции сравнения.

Код программы:

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

"""
Создать класс Hex для работы с беззнаковыми целыми шестнадцатеричными числами,
используя для представления числа список из 100 элементов типа int, каждый из
которых
является шестнадцатеричной цифрой. Младшая цифра имеет меньший индекс. Реальный
размер списка задается как аргумент конструктора инициализации. Реализовать
арифметические операции, аналогичные встроенным для целых и операции сравнения.
"""

class Hex:
    SIZE_LIMIT = 100 # Максимальный размер списка

    def __init__(self, value=0):
        self.count = 0
        self.digits = [0] * self.SIZE_LIMIT
        self.set_value(value)

    def set_value(self, value):
        hex_str = hex(value)[2:].upper()
        self.count = min(len(hex_str), self.SIZE_LIMIT)
        self.digits[:self.count] = [int(d, 16) for d in hex_str[-self.count:][::-1]]

    def get_size(self):
        return self.SIZE_LIMIT

    def __getitem__(self, index):
        if 0 <= index < self.SIZE_LIMIT:
            return self.digits[index]
        else:
            raise IndexError("Index out of range")

    def __setitem__(self, index, value):
        if 0 <= index < self.SIZE_LIMIT:
            self.digits[index] = value
        else:
            raise IndexError("Index out of range")

    def display(self):
        print("0x" + "".join([hex(d)[2:].upper() for d in self.digits[::-1]]))

if __name__ == "__main__":
```

```
hex_num = Hex(255)
hex_num.display()

print("Size:", hex_num.get_size())
print("Element at index 2:", hex_num[2])

hex_num[1] = 10
hex_num.display()
```

```
Результат работы программы:
```

```
0x00000000000000000000000000000000000000000000000000000000000000FF  
Size: 100  
Element at index 2: 0  
  
0x000000000000000000000000000000000000000000000000000000000000AF  
|  
  
Process finished with exit code 0
```

Рисунок 2. Результат работы программы

### Ответы на вопросы:

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

В python имеются методы, которые не вызываются напрямую, а вызываются встроенными функциями или операторами. С их помощью можно перегрузить операции.

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

Пример: `__add__` - сложение, `__sub__` - вычитание, `__mul__` - умножение.

3. В каких случаях будут вызваны следующие методы: `__add__`, `iadd` и `radd`?

`__add__` - вызывается при сложении двух чисел оператором «+». В случае, если это сделать не удаётся, вызываются `__iadd__` и `__radd__`, они делают то же самое, что и арифметические операторы, перечисленные выше, но для аргументов, находящихся справа, и только в случае, если для левого операнда не определён соответствующий метод.

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

Управляет созданием экземпляра. В качестве обязательного аргумента принимает класс (не путать с экземпляром). Должен возвращать экземпляр класса для его последующей его передачи методу `__init__`

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

`__str__` - вызывается функциями `str`, `print` и `format`. Возвращает строковое представление объекта.

`__repr__` - вызывается встроенной функцией `repr`; возвращает "сырые" данные, используемые для внутреннего представления в python.

**Вывод:** в ходе выполнения данной лабораторной работы были приобретены навыки по перегрузке операторов при написании программ с использованием языка программирования Python версии 3.x.