

**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

Костукайло К.Н. « » _____ 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:

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

class Pair:

    def __init__(self, first, second):
        self.first = first
        self.second = second

    @classmethod
    def read(cls):
        line = input("Введите коэффициент А (Пример: 2/3): ")
        line2 = input("Введите коэффициент В (Пример: 2/3): ")

        parts = list(map(int, line.split("/", maxsplit=1)))
        parts2 = list(map(int, line2.split("/", maxsplit=1)))

        if parts[1] == 0 and parts2[1] == 0:
            raise ValueError()

        return cls(parts, parts2)

    def display(self):
        print(f"Y = {self.first[0]}/{self.first[1]}*X + {self.second[0]}/{self.second[1]}")

    def sol_lin_equ(self):
        return (self.first[1]/self.first[0])*-1.0 / (self.second[1]/self.second[0])

    def __eq__(self, other):
        return self.sol_lin_equ() == other.sol_lin_equ()

    def __ne__(self, other):
        return self.sol_lin_equ() != other.sol_lin_equ()

    def __add__(self, other):
        return self.sol_lin_equ() + other.sol_lin_equ()

    def __sub__(self, other):
        return self.sol_lin_equ() - other.sol_lin_equ()

    def __truediv__(self, other):
        return self.sol_lin_equ() / other.sol_lin_equ()
```

```

def __float__(self):
    self.first[0] /= self.first[1]
    self.second[0] /= self.second[1]
    return self

if __name__ == "__main__":
    pair = Pair.read()
    pair2 = Pair.read()
    pair.display()
    pair2.display()
    print(pair.sol_lin_equ())
    print(pair2.sol_lin_equ())
    # Перегрузка оператора ==
    print(pair == pair2)
    # Перегрузка оператора !=
    print(pair != pair2)
    # Перегрузка оператора +
    print(pair + pair2)
    # Перегрузка оператора -
    print(pair - pair2)
    # Перегрузка оператора /
    print(pair / pair2)

```

Результат работы программы:

```

Введите коэффициент А (Пример: 2/3): 1/2
Введите коэффициент В (Пример: 2/3): 2/3
Введите коэффициент А (Пример: 2/3): 2/1
Введите коэффициент В (Пример: 2/3): 3/2
Y = 1/2*X + 2/3
Y = 2/1*X + 3/2
-1.3333333333333333
-0.75
False
True
None
-0.5833333333333333
1.7777777777777777

```

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.

Карточка персоны содержит фамилию и дату рождения. Реализовать класс ListPerson для работы с картотекой персоналий. Класс должен содержать список карточек персон. Реализовать методы добавления и удаления карточек персон, а также метод доступа к карточке по фамилии. Фамилии в списке должны быть уникальны. Реализовать операции объединения двух картотек, операцию пересечения и вычисления разности. Реализовать метод, выдающий по фамилии знак зодиака. Для этого в классе должен быть объявлен список словарей Zodiac с ключами: название знака зодиака, дата начала и дата окончания периода.

Код программы:

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

class Triad:
    def __init__(self, a, b, c):
        self.a = a
        self.b = b
        self.c = c

    def compare(self, other):
        # Сравниваем значения полей троек чисел
        if self.a == other.a and self.b == other.b and self.c == other.c:
            return "Триады равны"
        elif (
            self.a > other.a
            or (self.a == other.a and self.b > other.b)
            or (self.a == other.a and self.b == other.b and self.c > other.c)
        ):
            return "Первая триада больше второй"
        else:
            return "Вторая триада больше первой"

class Time(Triad):
    def __init__(self, hour, minute, second):
        super().__init__(hour, minute, second)

    def compare(self, other):
        # Сравниваем значения полей моментов времени
        if (
            self.a > other.a
            or (self.a == other.a and self.b > other.b)
            or (self.a == other.a and self.b == other.b and self.c > other.c)
        ):
            return "Первый момент времени больше второго"
        elif self.a == other.a and self.b == other.b and self.c == other.c:
            return "Моменты времени равны"
        else:
            return "Второй момент времени больше первого"

if __name__ == "__main__":
```

```
# Создаем объекты класса Triad
triad1 = Triad(1, 2, 3)
triad2 = Triad(4, 5, 6)
print(triad1.compare(triad2)) # Сравниваем тройки чисел

# Создаем объекты класса Time
time1 = Time(10, 20, 30)
time2 = Time(10, 20, 30)
print(time1.compare(time2)) # Сравниваем моменты времени
```

Результат работы программы:

```
Выберите действие:
1. Добавить пользователя
2. Удалить пользователя
3. Вывести список пользователей
4. Выйти
5. Узнать знак зодиака по имени пользователя
Введите номер действия: 1
Введите фамилию пользователя: Malzew
Введите дату рождения пользователя (в формате mm.dd): 02.23
Пользователь Malzew успешно добавлен
Введите номер действия: 5
Введите имя: Malzew
Рыбы
Введите номер действия: 4

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 работы были приобретены навыки по перегрузке операторов при написании программ с использованием языка программирования Python версии 3.x.