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

Стригалов Д.М. « »_____20__г.

Подпись студента_____

Работа защищена «
»_____20__г.

Проверил доцент
Кафедры инфокоммуникаций, старший преподаватель Воронкин Р.А.

_____
(подпись)

Ставрополь 2023

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 MyPair:

    def __init__(self, value1=0.0, value2=0.0):
        value1 = float(value1)
        value2 = float(value2)

        self.__value1 = value1
        self.__value2 = value2

    @property
    def value1(self):
        return self.__value1

    @property
    def value2(self):
        return self.__value2

    # Прочитать значение
    def read(self):
        self.__value1 = float(input("Введите значение для первого поля
(дробное число): "))
        self.__value2 = float(input("Введите значение для второго поля
(положительное дробное число): "))
        if self.__value2 <= 0:
            raise ValueError("Значение второго поля должно быть положительным
дробным числом")

    # Вывести на экран
    def display(self):
        print(f"Value1: {self.__value1}, Value2: {self.__value2}")

    # Умножение на произвольное дробное число
    def multiply(self, factor):
        self.__value1 *= factor
        self.__value2 *= factor

    def __eq__(self, other):
        return self.__value1 == other.__value1 and self.__value2 ==
other.__value2

    def __ne__(self, other):
        return self.__value1 != other.__value1 or self.__value2 !=
other.__value2
```

```
    def __mul__(self, factor):
        return MyPair(self.__value1 * factor, self.__value2 * factor)


def make_pair(first, second):
    """
    Функция создания экземпляра класса MyPair, принимая значения полей как
аргументы
    """
    return MyPair(first, second)


if __name__ == '__main__':
    pair = MyPair()
    pair.read()
    pair.display()

    factor = float(input("Введите множитель для умножения: "))
    pair.multiply(factor)
    print("Результат умножения:")
    pair.display()

    other_pair = MyPair()
    other_pair.read()

    # Перегрузка оператора ==
    print(pair == other_pair)

    # Перегрузка оператора !=
    print(pair != other_pair)

    # Перегрузка оператора умножения *
    factor = float(input("Введите множитель для умножения 2-ого экземпляра:
"))
    result_pair = other_pair * factor
    print("Результат умножения 2-ого экземпляра:")
    result_pair.display()
```

Результат работы программы:

```
Введите значение для первого поля (дробное число): 3
Введите значение для второго поля (положительное дробное число): 4
Value1: 3.0, Value2: 4.0
Введите множитель для умножения: 2
Результат умножения:
Value1: 6.0, Value2: 8.0
Введите значение для первого поля (дробное число): 3
Введите значение для второго поля (положительное дробное число): 4
False
True
Введите множитель для умножения 2-ого экземпляра: 4
Результат умножения 2-ого экземпляра:
Value1: 12.0, Value2: 16.0
```

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.

Реализовать класс Money, используя для представления суммы денег список словарей. Словарь имеет два ключа: номинал купюры и количество купюр данного достоинства. Номиналы представить как строку. Элемент списка словарей с меньшим индексом содержит меньший номинал.

Код программы:

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


class Money:
    MAX_SIZE = 10

    def __init__(self):
        self.bills = []
        self.size = self.MAX_SIZE
        self.count = 0

    def get_size(self):
        return self.size

    def add_bill(self, denomination, count):
        if self.count + count > self.size:
            print("Превышен максимальный размер списка")
            return

        for bill in self.bills:
            if bill["denomination"] == denomination:
                bill["count"] += count
                break
        else:
            self.bills.append({"denomination": denomination, "count": count})
        self.count += count

    def total_amount(self):
        total = 0
        for bill in self.bills:
            total += bill["denomination"] * bill["count"]
        return total

    def __add__(self, other):
        if isinstance(other, Money):
```

```
            new_money = Money()
            new_money.bills = [bill.copy() for bill in self.bills]

            for bill in other.bills:
                if new_money.count + bill["count"] > new_money.size:
                    print(f"Превышен максимальный размер списка
({bill['denomination']})")
                    return self

                new_money.add_bill(bill["denomination"], bill["count"])

            return new_money
        else:
            raise TypeError("Неверный тип операнда")

    def __getitem__(self, index):
        if 0 <= index < len(self.bills):
            return self.bills[index]["denomination"]
        else:
            raise IndexError("Индекс вне диапазона")

    def print_money(self, label):
        print(f"\n{label}:")
        print("\n".join([f"{bill['denomination']}: {bill['count']} шт." for
bill in self.bills]))


if __name__ == "__main__":
    money1 = Money()
    money1.add_bill(10, 5)
    money1.add_bill(50, 3)
    money1.add_bill(100, 2)

    money2 = Money()
    money2.add_bill(10, 5)
    money2.add_bill(50, 1)
    money2.add_bill(100, 2)
    money2.add_bill(20, 2)

    money1.print_money("Текущий состав денег (money1)")
    money2.print_money("Текущий состав денег (money2)")

    money3 = money1 + money2
    money3.print_money("Сумма денег (money1 + money2)")

    print(f"\nРазмер списка (money1): {money1.get_size()}")
```

Результат работы программы:

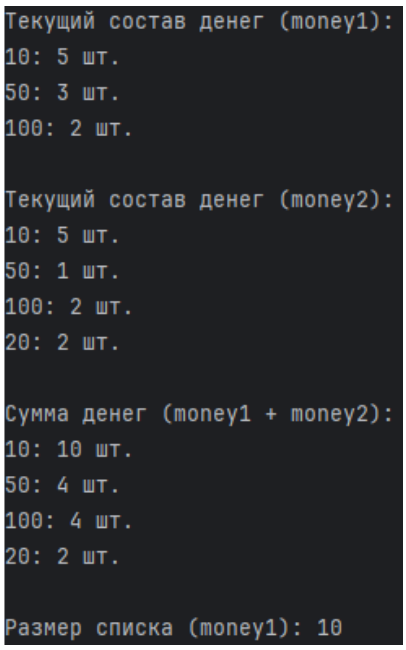

Рисунок 2. Результат работы программы

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, использующиеся 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.