

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

**Кафедра инфокоммуникаций**

**Отчет по лабораторной работе №4.2  
Перегрузка операторов в языке Python  
по дисциплине «Объектно-ориентированное программирование»**

Выполнил студент группы ИВТ-б-о-21-1

Лысенко И.А. « » \_\_\_\_\_ 20\_\_ г.

Подпись студента \_\_\_\_\_

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

Проверил Воронкин Р.А. \_\_\_\_\_

(подпись)

Ставрополь 2023

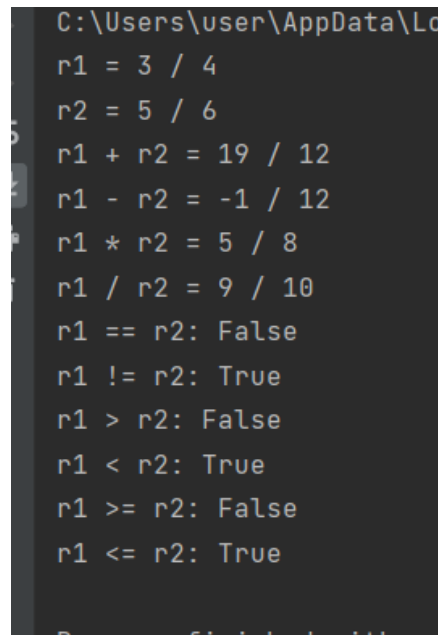
**Цель работы:** приобретение навыков по перегрузке операторов при написании программ с помощью языка программирования Python версии 3.x.

**Ход работы:**

1. Создал репозиторий на GitHub:

[https://github.com/IsSveshuD/OOP\\_Lab\\_4.2.git](https://github.com/IsSveshuD/OOP_Lab_4.2.git) .

2. Проработал пример:



```
C:\Users\user\AppData\Local\Microsoft\Windows\Terminal\
r1 = 3 / 4
r2 = 5 / 6
r1 + r2 = 19 / 12
r1 - r2 = -1 / 12
r1 * r2 = 5 / 8
r1 / r2 = 9 / 10
r1 == r2: False
r1 != r2: True
r1 > r2: False
r1 < r2: True
r1 >= r2: False
r1 <= r2: True
```

Рисунок 1 – Пример

3. Выполнил индивидуальное задание 1.

```

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

class Equation:
    def __init__(self, a, b):
        self.first = a
        self.second = b

    def read(self):
        self.first = float(input("Введите значение коэффициента A: "))
        self.second = float(input("Введите значение коэффициента B: "))

    def display(self):
        print(f"Уравнение y = {self.first}x + {self.second}")

    def function(self, x):
        return self.first * x + self.second

    def __str__(self):
        return f"Equation({self.first}, {self.second})"

    def __eq__(self, other):
        if isinstance(other, Equation):
            return self.first == other.first and self.second == other.second
        return False

    def __add__(self, other):
        if isinstance(other, Equation):
            return Equation(self.first + other.first, self.second + other.second)
        elif isinstance(other, (int, float)):
            return Equation(self.first, self.second + other)
        else:
            raise ValueError("Illegal type of the argument")

    def __sub__(self, other):
        if isinstance(other, Equation):
            return Equation(self.first - other.first, self.second - other.second)
        elif isinstance(other, (int, float)):
            return Equation(self.first, self.second - other)
        else:
            raise ValueError("Illegal type of the argument")

if __name__ == "__main__":
    eq1 = Equation(2, 3)
    eq2 = Equation(1, 4)

    print(eq1 + eq2)
    print(eq1 - eq2)
    print(eq1 + 5)
    print(eq1 == eq2)
    print(str(eq1))

```

Рисунок 2 – Индивидуальное задание1

#### 4. Получил следующий результат работы индивидуального задания

1.

```
C:\Users\User\AppData\Local\Programs\
Equation(3, 7)
Equation(1, -1)
Equation(2, 8)
False
Equation(2, 3)

Process finished with exit code 0
```

Рисунок 3 – Результат работы задания 1

5. Выполнил индивидуальное задание 1.

```
class Test:
    def __init__(self, question, options, correct_answer, points):
        self.question = question
        self.options = options
        self.correct_answer = correct_answer
        self.points = points

class TestContent:
    # Максимальный размер списка
    Max_size = 100

    def __init__(self, size=Max_size):
        self.questions = []
        self.size = size
        self.count = 0

    def add_question(self, question):
        if self.count < self.size:
            if question not in self.questions:
                self.questions.append(question)
                self.count += 1
            else:
                print("Вопрос уже есть в списке.")
        else:
            print("Достигнут максимально возможный размер списка.")

    def remove_question(self, question):
        if question in self.questions:
            self.questions.remove(question)
            self.count -= 1
        else:
```

```

        print("Вопрос не найден в списке.")

    def get_question(self, index):
        if 0 <= index - 1 < self.count:
            return self.questions[index - 1]
        else:
            print("Неверный номер вопроса.")

    def merge_tests(self, other_test):
        new_test = TestContent(size=self.size)
        new_test.questions = self.questions + other_test.questions
        new_test.count = len(new_test.questions)
        return new_test

    def intersect_tests(self, other_test):
        new_test = TestContent(size=self.size)
        new_test.questions = list(set(self.questions)
                                   & set(other_test.questions))
        new_test.count = len(new_test.questions)
        return new_test

    def diff_tests(self, other_test):
        new_test = TestContent(size=self.size)
        new_test.questions = list(set(self.questions)
                                   - set(other_test.questions))
        new_test.count = len(new_test.questions)
        return new_test

    def generate_test(self, k):
        import random
        if k > self.size - self.count:
            print("Запрошенное количество вопросов больше,"
                  " чем доступное количество для добавления.")
            return None
        else:
            selected_questions = random.sample(self.questions, k)
            new_test = TestContent(size=self.size)
            new_test.questions = selected_questions
            new_test.count = len(selected_questions)
            return new_test

    def size(self):
        return self.size

if __name__ == "__main__":
    q1 = Test("Какой язык программирования вы изучаете?",
              ["Python", "Java", "C++", "JavaScript", "Ruby"], 0, 10)
    q2 = Test("Что такое ООП?",
              ["Объектно-ориентированное программирование",
               "Объектно-ориентированный протокол",
               "Объективно-ориентированное программирование",
               "Объектно-ориентированный процессор",
               "Объектно-ориентированная парадигма"], 0, 15)

    test_content1 = TestContent(size=5)
    test_content1.add_question(q1)
    test_content1.add_question(q2)

    # Печать размера и количества элементов

```

```

print(f"Размер: {test_content1.size}, \
      Количество элементов: {test_content1.count}")

question3 = Test("Что такое GIT?",
                 ["Глобальный информационный трекер",
                  "Графический интернет-трансфер",
                  "Общий интернет-текст",
                  "GNU интерактивные инструменты",
                  "Объектно-ориентированное программирование"], 4, 20)
test_content1.add_question(question3)

# Пример добавления существующего вопроса
test_content1.add_question(q2)

# Пример получения вопроса по номеру
get_q = test_content1.get_question(1)
print(get_q.question)

# Пример удаления вопроса
test_content1.remove_question(q1)

# Печать размера и количества элементов после изменений
print(f"Размер: {test_content1.size}, \
      Количество элементов: {test_content1.count}")

# Примеры слияния, пересечения и вычитания списков
q4 = Test("Столица России?",
          ["Лондон", "Воронеж", "Москва",
           "Ставрополь", "Пекин"], 2, 50)
test_content2 = TestContent(size=5)
test_content2.add_question(q1)
test_content2.add_question(q4)
test_content1.merge_tests(test_content2)
test_content1.intersect_tests(test_content2)
test_content1.diff_tests(test_content2)

```

Рисунок 4 – Индивидуальное задание 2

6. Получил следующий результат работы индивидуального задания 2:

```

C:\Users\user\AppData\Local\Programs\Python\Python3
Размер: 5,          Количество элементов: 2
Вопрос уже есть в списке.
Какой язык программирования вы изучаете?
Размер: 5,          Количество элементов: 2
Process finished with exit code 0

```

Рисунок 5 – Результат работы задания 2.

### Ответы на вопросы:

1. Какие средства существуют в Python для перегрузки операций?

В Python для перегрузки операций используются специальные методы, начинающиеся и заканчивающиеся двумя символами подчеркивания.

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

Арифметические операции:

- `__add__` - сложение
- `__sub__` - вычитание
- `__mul__` - умножение
- `__truediv__` - деление
- `__floordiv__` - целочисленное деление
- `__mod__` - остаток от деления
- `__pow__` - возведение в степень

Операции отношения:

- `__eq__` - равенство
- `__ne__` - неравенство
- `__lt__` - меньше
- `__le__` - меньше или равно
- `__gt__` - больше
- `__ge__` - больше или равно

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

`__add__` вызывается для объекта при использовании оператора `+`.

Например:

```
class MyClass:
    def __init__(self, value):
        self.value = value
    def __add__(self, other):
        return MyClass(self.value + other.value)

obj1 = MyClass(1)
obj2 = MyClass(2)
result = obj1 + obj2
```

`__iadd__` вызывается для объекта при использовании оператора `+=`.

Меняет сам объект. Например:

```
class MyClass:
    def __init__(self, value):
        self.value = value
    def __iadd__(self, other):
        self.value += other.value
        return self
obj1 = MyClass(1)
obj2 = MyClass(2)
obj1 += obj2
```

`__radd__`: Вызывается, когда оператор `+` применяется к объекту, и левый операнд не поддерживает эту операцию. Например:

```
class MyClass:
    def __init__(self, value):
        self.value = value
    def __radd__(self, other):
        return MyClass(self.value + other)
result = 1 + MyClass(2)
```

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

`__new__` вызывается для создания нового экземпляра объекта до его инициализации методом `__init__`. Он часто переопределяется в случаях, когда требуется управление процессом создания объекта, например, для создания объекта неизменяемого типа.

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

`__str__`: Возвращает строковое представление объекта, предназначенное для удобства чтения человеком. Вызывается функцией `str()` или встроенной функцией `print()`.



`__repr__`: Возвращает строковое представление объекта, предназначенное для воспроизведения объекта. Если метод `__str__` не определен, то вызывается он. Вызывается функцией `repr()` или при использовании функции `eval()`

**Вывод:** в результате выполнения лабораторной работы были приобретены навыки по перегрузке операторов при написании программ с помощью языка программирования Python.