

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

Кафедра инфокоммуникаций

Лабораторная работа 2

Перегрузка операторов в языке Python

Выполнил студент группы ИВТ-б-о-20-1

Пентухов С. А. « » _____ 2022г.

Подпись студента _____

Работа защищена « » _____ 2022г.

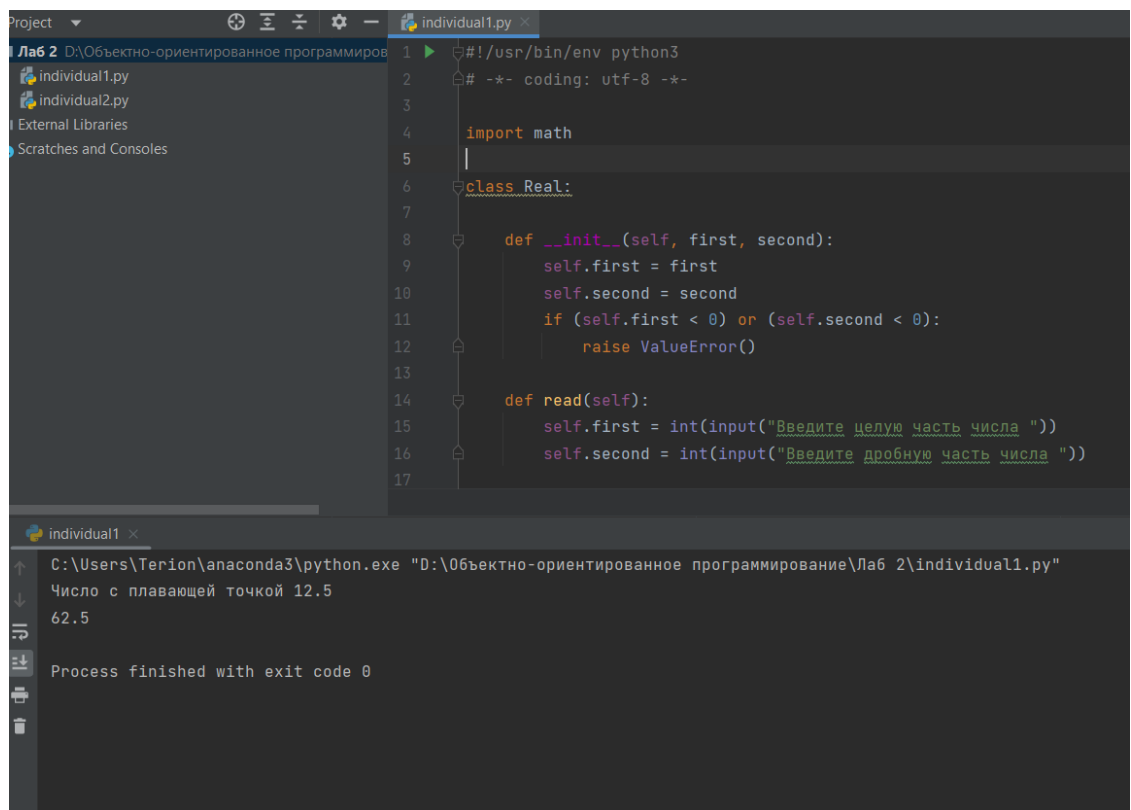
Проверил Воронкин Р.А. _____

(подпись)

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

Задание 1

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



```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import math
5
6 class Real:
7
8     def __init__(self, first, second):
9         self.first = first
10        self.second = second
11        if (self.first < 0) or (self.second < 0):
12            raise ValueError()
13
14    def read(self):
15        self.first = int(input("Введите целую часть числа "))
16        self.second = int(input("Введите дробную часть числа "))
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

individual1

C:\Users\Terion\anaconda3\python.exe "D:\Объектно-ориентированное программирование\Лаб 2\individual1.py"

Число с плавающей точкой 12.5

62.5

Process finished with exit code 0

Рисунок 1 – Результат выполнения

Задание 2

Нагрузка преподавателя за учебный год представляет собой список дисциплин, преподаваемых им в течение года. Одна дисциплина представляется информационным словарем с ключами: название дисциплины,

The image shows a Python IDE with a dark theme. The top pane displays a Python script with the following code:

```
17 for number in discipline:
18     if number not in unique:
19         unique.append(number)
20
21 return unique
22
23
24 def all_hours(self):
25     all_hours = 0
26     for i in self.discipline:
27         all = i['hours_pract'] + i['hours_lekt']
28         i['summ_hours'] = all
29         all_hours += all
30         if i['cont'] == 'exam':
31             i['hours_for_student'] = 0.5
32         else:
33             i['hours_for_student'] = 0.35
34     return all_hours
35
36 if __name__ == '__main__': > 'name'
```

The bottom pane shows the output of the script execution:

```
individual2 ×
C:\Users\Terion\anaconda3\python.exe "D:\Объектно-ориентированное программирование\Лаб 2\individual2.py"
Список предметов [{ 'name': 'Algebra', 'term': 2, 'students': 45, 'hours_lekt': 20, 'hours_pract': 40, 'cont': 'exam' }
4
Список предметов [{ 'name': 'Algebra', 'term': 2, 'students': 45, 'hours_lekt': 20, 'hours_pract': 40, 'cont': 'exam' }
4
Process finished with exit code 0
```

Рисунок 2 – Результат выполнения

Ответы на контрольные вопросы

1. Перегрузка операторов — один из способов реализации полиморфизма, когда мы можем задать свою реализацию какого-либо метода в своём классе.

2. `__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 ($x \wedge y$).
`__or__(self, other)` - битовое ИЛИ ($x | y$).

3. операция $x + y$ будет сначала пытаться вызвать `x.__add__(y)`, и только в том случае, если это не получилось, будет пытаться вызвать `y.__radd__(x)`. Аналогично для остальных методов.

4. `__new__(cls[, ...])` — 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. `__str__(self)` - вызывается функциями `str`, `print` и `format`. Возвращает строковое представление объекта.