

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

Отчет по лабораторной работе № 4.2  
Перегрузка операторов в языке Python

Выполнил студент группы ИВТ-б-о-20-1

Ищенко М.А.

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

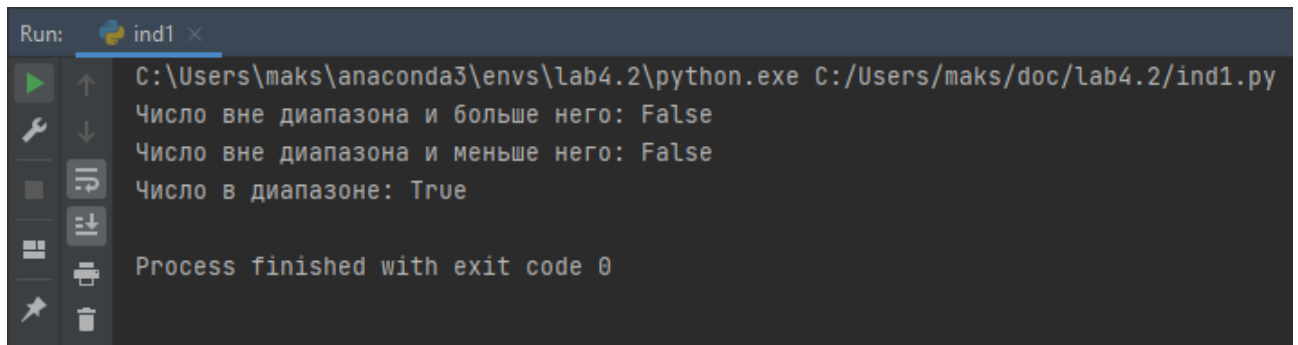
Проверил(а) \_\_\_\_\_

Ставрополь 2022

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

Создан общедоступный репозиторий на GitHub. Дополнен файл .gitignore необходимыми правилами для работы с IDE PyCharm.

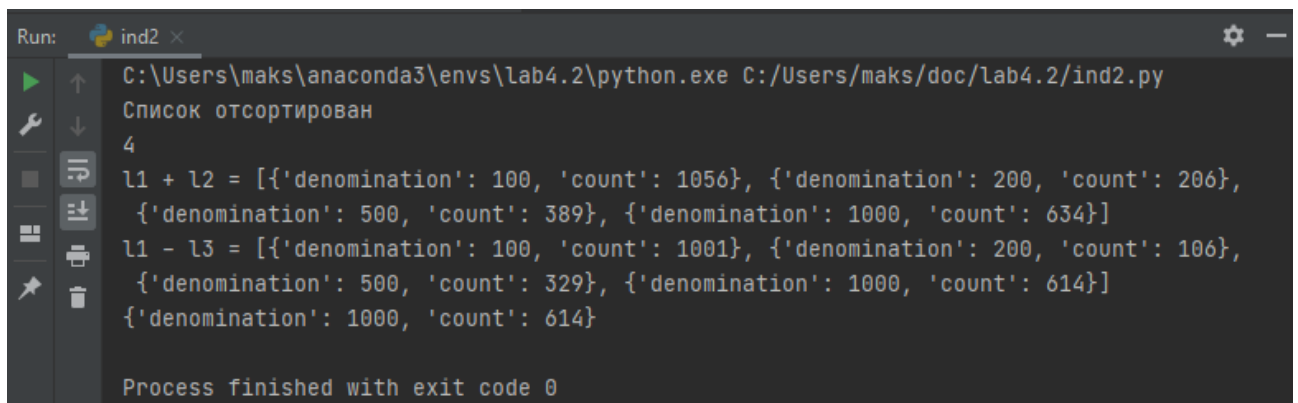
Выполнено первое индивидуальное задание варианта 8, рис. 1



```
Run: ind1 x
C:\Users\maks\anaconda3\envs\lab4.2\python.exe C:/Users/maks/doc/lab4.2/ind1.py
Число вне диапазона и больше него: False
Число вне диапазона и меньше него: False
Число в диапазоне: True
Process finished with exit code 0
```

Рисунок 1

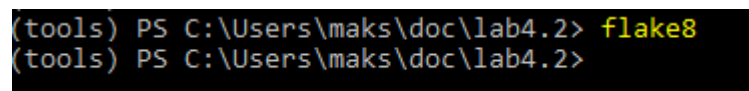
Выполнено второе индивидуальное задание варианта 8, рис. 1



```
Run: ind2 x
C:\Users\maks\anaconda3\envs\lab4.2\python.exe C:/Users/maks/doc/lab4.2/ind2.py
Список отсортирован
4
l1 + l2 = [{'denomination': 100, 'count': 1056}, {'denomination': 200, 'count': 206},
{'denomination': 500, 'count': 389}, {'denomination': 1000, 'count': 634}]
l1 - l3 = [{'denomination': 100, 'count': 1001}, {'denomination': 200, 'count': 106},
{'denomination': 500, 'count': 329}, {'denomination': 1000, 'count': 614}]
{'denomination': 1000, 'count': 614}
Process finished with exit code 0
```

Рисунок 2

Программы проверены на flake8, рис. 3



```
(tools) PS C:\Users\maks\doc\lab4.2> flake8
(tools) PS C:\Users\maks\doc\lab4.2>
```

Рисунок 3

Контрольные вопросы:

1. Какие средства существуют в Python для перегрузки операций?

Перегрузка осуществляется при помощи специальных методов.

Методы группируются по следующим категориям:

- методы для всех видов операций;

- методы перегрузки операторов работы с коллекциями;
- методы для числовых операций в двоичной форме;
- методы для других операций над числами;
- методы для операций с дескрипторами;
- методы для операций, используемых с диспетчерами контекста.

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

`__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 ( $x \wedge y$ ).

`__radd__(self, other)`,

`__rsub__(self, other)`,

`__rmul__(self, other)`,

`__rtruediv__(self, other)`,

`__rmod__(self, other)`,

`__rdivmod__(self, other)`,

`__rpow__(self, other)`,

`__rlshift__(self, other)`,

`__rrshift__(self, other)`,

`__rand__(self, other)`,

`__rxor__(self, other)` ,

`__ror__(self, other)` - 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.

`__iadd__(self, other)` - `+=` .

`__isub__(self, other)` - `-=` .

`__imul__(self, other)` - `*=` .

`__itruediv__(self, other)` - `/=` .

`__ifloordiv__(self, other)` - `//=` .

`__imod__(self, other)` - `%=` .

`__ipow__(self, other[, modulo])` - `**=` .

`__ilshift__(self, other)` - `<<=` .

`__irshift__(self, other)` - `>>=` .

`__iand__(self, other)` - `&=` .

`__ixor__(self, other)` - `^=` .

`__ior__(self, other)` - `|=` .

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

1) `__add__` - `a + b`

2) `__iadd__` - `a += b`

3) `__radd__` - Если не получилось вызвать метод `__add__`

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

Метод `__new__` используется, когда нужно управлять процессом создания нового экземпляра, а `__init__` – когда контролируется его инициализация.

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

\_\_str\_\_ должен возвращать строковый объект, тогда как \_\_repr\_\_ может возвращать любое выражение в Python

Вывод: в ходе занятия были приобретены навыки по перегрузке операторов при написании программ с помощью языка программирования Python версии 3.x.