

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ
АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО
ОБРАЗОВАНИЯ «СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ**

**Отчет по лабораторной работе № 4.2
«Перегрузка операторов в языке Python»**

по дисциплине «Основы программной инженерии»

Выполнила:
Образцова Мария Дмитриевна,
2 курс, группа ПИЖ-б-о-21-1,
Проверил:
Доцент кафедры инфокоммуникаций,
Воронкин Р.А.

Ставрополь, 2023 г.


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

Методика и порядок выполнения работы

1. Изучить теоретический материал работы.
2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и язык программирования Python.

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)


Owner * Repository name *


 obrMaria / OPI_4.2

✔ OPI_4.2 is available.

Great repository names are short and memorable. Need inspiration? How about [vigilant-spork?](#)

Description (optional)

☒  **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**
You choose who can see and commit to this repository.

Initialize this repository with:

☒ **Add a README file**
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore


.gitignore template: Python


Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: MIT License

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

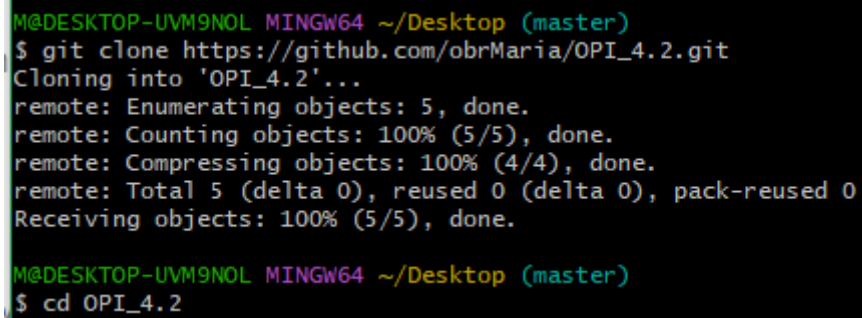
This will set  main as the default branch. Change the default name in your [settings](#).

 You are creating a public repository in your personal account.

[Create repository](#)

Рисунок —создание репозитория

3. Выполните клонирование созданного репозитория. Организуйте свой репозиторий в соответствие с моделью ветвления git-flow.



```
M@DESKTOP-UVM9NOL MINGW64 ~/Desktop (master)
$ git clone https://github.com/obrMaria/OPI_4.2.git
Cloning into 'OPI_4.2'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.

M@DESKTOP-UVM9NOL MINGW64 ~/Desktop (master)
$ cd OPI_4.2
```

Рисунок – клонирование созданного репозитория

4. Проработать примеры лабораторной работы.

The image shows a Python IDE with a file named `pr_1.py` open. The code defines a `Vector2D` class with several methods for vector operations. The `__init__` method sets `self.x` and `self.y`. The `__repr__` method returns a string representation of the vector. The `__str__` method returns a string representation of the vector. The `__add__` method returns a new `Vector2D` object with the sum of the x and y components. The `__iadd__` method increments the x and y components of the current object. The `__name__` attribute is set to `'__main__'`.

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4
5  import math
6
7
8  class Vector2D:
9      def __init__(self, x, y):
10         self.x = x
11         self.y = y
12
13     def __repr__(self):
14         return 'Vector2D({}, {})'.format(
15
16     def __str__(self):
17         return '({}, {})'.format(self.x,
18
19     def __add__(self, other):
20         return Vector2D(self.x + other.x,
21
22     def __iadd__(self, other):
23         self.x += other.x
24         self.y += other.y
25
26 if __name__ == '__main__':
```

The output of the program is shown in the Run window:

```
(-2, -2)
(-3, -4)
(8, 10)
True
False
(0, 0)
```

Рисунок 1 – Перегрузка операторов

5. Выполнить индивидуальные задания.

Задание 1

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

The image shows a Python IDE with two windows. The top window, titled 'pr_2.py', contains a Python script. The script defines a `Number` class with methods `__pow__`, `__eq__`, and `__ne__`. It then uses these methods in a `__main__` block to create two `Number` objects, calculate a power, and compare them. The bottom window, titled 'ind_1', shows the output of the script, which includes the results of the power calculation and the equality/inequality comparisons.

```
21 def __pow__(self, other):
22     return Number(self.first ** other.first)
23
24 def __eq__(self, other):
25     return self.first == other.first and s
26
27 def __ne__(self, other):
28     return not self.__eq__(other)
29
30
31 if __name__ == "__main__":
32     num1 = Number(1.5, 4)
33     num1.display()
34
35     num2 = Number(2.1, 2)
36     num2.display()
37
38     result = num1 ** num2
39     result.display()
40
41     print(f"num1 == num2: {num1 == num2}")
42     print(f"num1 != num2: {num1 != num2}")
```

if __name__ == "__main__"

ind_1

```
C:\Users\M\Desktop\OPI_4.2\PY\venv\Scripts\python.exe
1.5 ^ 4 = 5.0625
2.1 ^ 2 = 4.41
2.3431044239829237 ^ 8 = 908.5171965968312
num1 == num2: False
num1 != num2: True

Process finished with exit code 0
```

Рисунок – Индивидуальное задание №1

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.

В тех задачах, где возможно, реализовать конструктор инициализации строкой.

Создать класс `BitString` для работы с битовыми строками не более чем из 100 бит. Битовая строка должна быть представлена списком типа `int`, каждый элемент которого принимает значение 0 или 1. Реальный размер списка задается как аргумент конструктора инициализации. Должны быть реализованы все традиционные операции для работы с битовыми строками: `and`, `or`, `xor`, `not`. Реализовать сдвиг влево и сдвиг вправо на заданное количество битов.

The image shows a Python IDE with a file explorer on the left and a code editor. The code editor displays a Python script with line numbers 56 to 76. The script defines a class with a `__str__` method and a main block. The main block creates two `BitString` objects, `x` and `y`, and performs various bit operations: setting bits, printing the binary strings, and calculating AND, OR, XOR, NOT, right shift, and left shift. The output window at the bottom shows the execution results, including the binary strings and the results of the operations.

```
56
57 def __str__(self):
58     # Вывод результата в консоль
59     return ''.join(map(str, self.x))
60
61
62 if __name__ == "__main__":
63     x = BitString(8) # Размер списка 1 - 8 бит
64     y = BitString(8) # Размер списка 2 - 8 бит
65
66     x.set(60) # Первая цифра 00111100
67     print(x)
68     y.set(37) # Вторая цифра 00100101
69     print(y)
70
71     print(f'{x} and {y} = {x & y}')
72     print(f'{x} or {y} = {x | y}')
73     print(f'{x} xor {y} = {x ^ y}')
74     print(f'{x} not = {~x}')
75     print(f'{y} >> 1 = {y >> 1}')
76     print(f'{x} << 2 = {x << 2}')
```

Run ind_2

```
C:\Users\M\Desktop\OPI_4.2\PY\venv\Scripts\python.exe C
00111100
00100101
00111100 and 00100101 = 00100100
00111100 or 00100101 = 00111101
00111100 xor 00100101 = 00011001
00111100 not = 11000011
00100101 >> 1 = 00010010
11000011 << 2 = 00001100
```

Рисунок – Индивидуальное задание №2

ВОПРОСЫ

1. Какие средства существуют в Python для перегрузки операций?

Заключение оператора в двойное подчёркивание «`__`» с обеих сторон.

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

`__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 ($x | y$).

`__radd__`(self, other) ,

`__rsub__`(self, other) ,

`__rmul__`(self, other) ,

`__rtruediv__`(self, other) ,

`__rfloordiv__`(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.

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

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

Он 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__(self)` - вызывается функциями `str`, `print` и `format`. Возвращает строковое представление объекта.

`__repr__(self)` - вызывается встроенной функцией `repr`; возвращает "сырые" данные, используемые для внутреннего представления в python.