

**Министерство науки и высшего образования Российской Федерации  
Федеральное государственное автономное образовательное учреждение  
высшего образования  
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

Институт цифрового развития  
Кафедра инфокоммуникаций

**ОТЧЕТ  
ПО ЛАБОРАТОРНОЙ РАБОТЕ №2  
по дисциплине «Объектно-ориентированное программирование»  
«Перегрузка операторов в языке Python»  
Вариант 5**

Выполнила:

Диченко Дина Алексеевна  
студентка 3 курса, группы ИВТ-б-о-21-1  
направление подготовки Информатика и  
вычислительная техника, очная форма  
обучения

\_\_\_\_\_  
(подпись)

Проверил:

Воронкин Роман Александрович

\_\_\_\_\_  
(подпись)

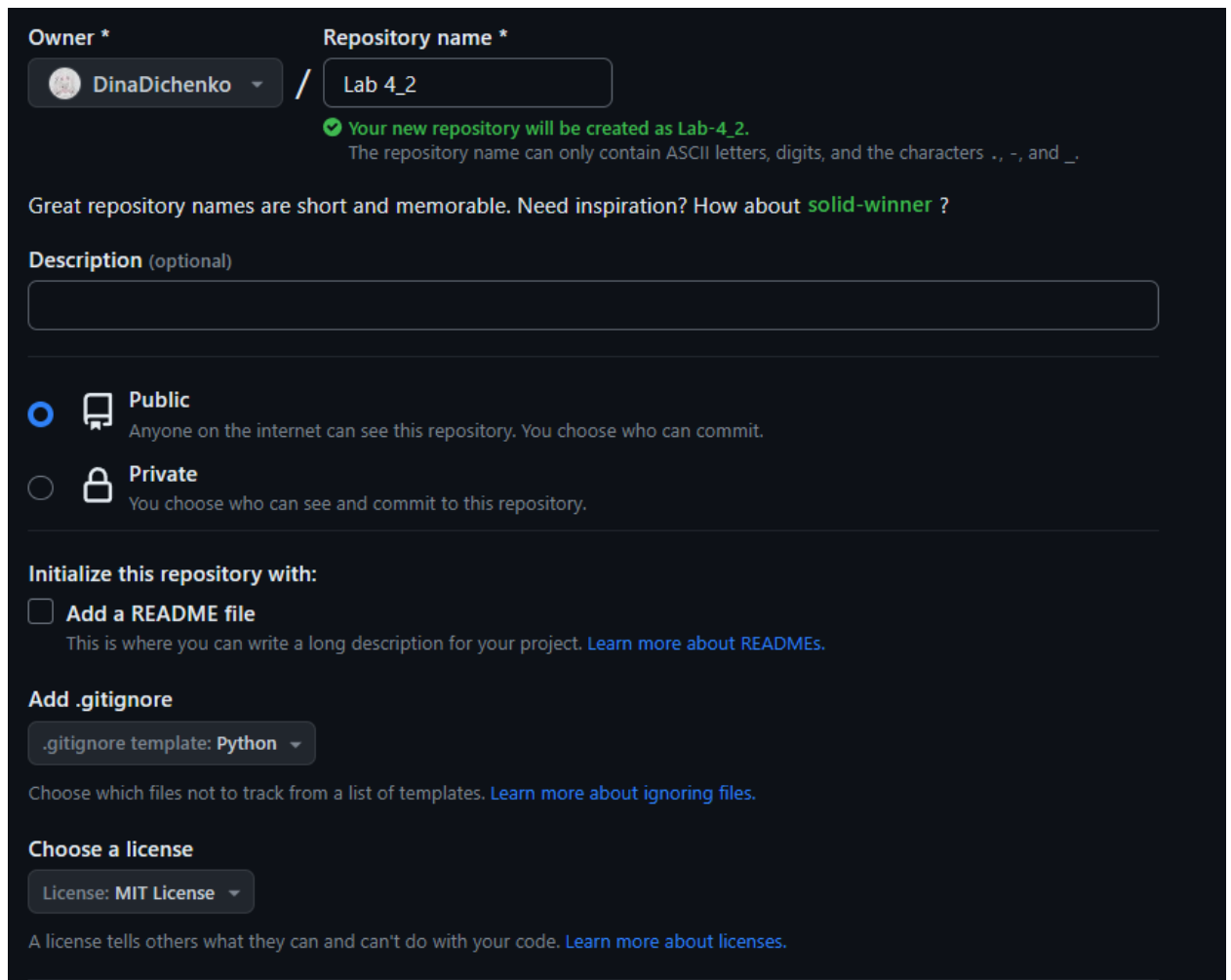
Отчет защищен с оценкой \_\_\_\_\_ Дата защиты \_\_\_\_\_

Ставрополь, 2023 г


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

### Практическая часть:

1. Создала общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и язык программирования Python.




Owner \* / Repository name \*


 DinaDichenko / Lab 4\_2

✔ Your new repository will be created as Lab-4\_2.  
The repository name can only contain ASCII letters, digits, and the characters ., -, and \_.

Great repository names are short and memorable. Need inspiration? How about **solid-winner** ?

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.](#)

Рисунок 1. Создание репозитория

2. Выполнила клонирование созданного репозитория.

```
C:\Users\super\OneDrive\Рабочий стол\DI\ВУЗ\Воронкин х3>git clone https://github.com/DinaDichenko/Lab-4_2.git
Cloning into 'Lab-4_2'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.
```

Рисунок 2. Клонирование репозитория

3. Дополнила файл .gitignore необходимыми правилами для работы с IDE PyCharm.

```
Spyder project settings
spyderproject
spyproject

Rope project settings
ropeproject

mkdocs documentation
site

mypy
mypy_cache/
mypy.json
mypy.json

Pyre type checker
pyre/

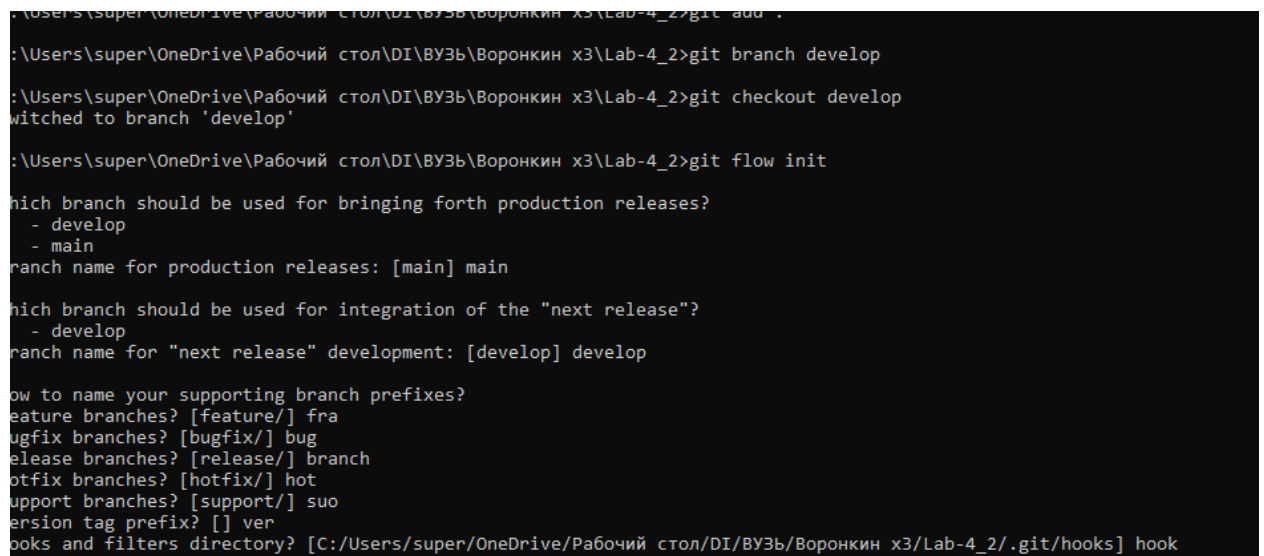
pytype static type analyzer
pytype/

Cython debug symbols
cython_debug/

PyCharm
JetBrains specific template is maintained in a separate JetBrains.gitignore that can
```

Рисунок 3. Изменение файла .gitignore

4. Организовала свой репозиторий в соответствии с моделью ветвления git-flow.



```
C:\Users\super\OneDrive\Рабочий стол\DI\ВУЗ\Воронкин х3\Lab-4_2>git add .
C:\Users\super\OneDrive\Рабочий стол\DI\ВУЗ\Воронкин х3\Lab-4_2>git branch develop
C:\Users\super\OneDrive\Рабочий стол\DI\ВУЗ\Воронкин х3\Lab-4_2>git checkout develop
Switched to branch 'develop'
C:\Users\super\OneDrive\Рабочий стол\DI\ВУЗ\Воронкин х3\Lab-4_2>git flow init
Which branch should be used for bringing forth production releases?
- develop
- main
branch name for production releases: [main] main
Which branch should be used for integration of the "next release"?
- develop
branch name for "next release" development: [develop] develop
How to name your supporting branch prefixes?
feature branches? [feature/] fra
bugfix branches? [bugfix/] bug
release branches? [release/] branch
hotfix branches? [hotfix/] hot
support branches? [support/] suo
version tag prefix? [] ver
Hooks and filters directory? [C:/Users/super/OneDrive/Рабочий стол/DI/ВУЗ/Воронкин х3/Lab-4_2/.git/hooks] hook
```

Рисунок 4. Организация репозитория в соответствии с git-flow

5. Проработала примеры лабораторной работы.

```
PS C:\Users\student-09-510> & C:/ProgramData/anaconda3/python.exe f:/Dinohka/00П/00П_2.1
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
PS C:\Users\student-09-510>
```

Рисунок 5. Результат работы примера

## 6. Выполнила индивидуальные задания.

### Задание 1

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

```
Введите цену и количество товара через пробел: 20 3
Стоимость товара: 60.0
PS C:\Users\super\OneDrive\Рабочий стол\DI\ВУЗ\Воронкин х3\Lab-4.1\prog>
```

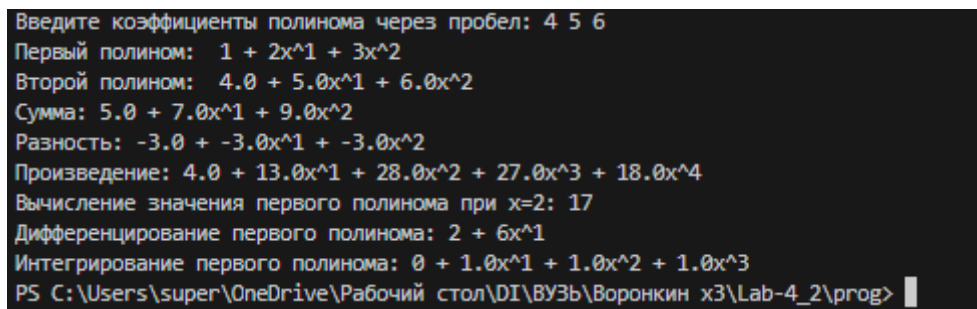
Рисунок 6. Выполнение индивидуального задания 1

### 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.

(вариант 5)

Создать класс `Polinom` для работы с многочленами до 100-й степени. Коэффициенты должны быть представлены списком из 100 элементов — коэффициентов. Младшая степень имеет меньший индекс (нулевая степень — нулевой индекс). Размер списка задается как аргумент конструктора инициализации. Реализовать арифметические операции и операции сравнения, вычисление значения полинома для заданного значения  $x$ , дифференцирование, интегрирование.



```
Введите коэффициенты полинома через пробел: 4 5 6
Первый полином: 1 + 2x^1 + 3x^2
Второй полином: 4.0 + 5.0x^1 + 6.0x^2
Сумма: 5.0 + 7.0x^1 + 9.0x^2
Разность: -3.0 + -3.0x^1 + -3.0x^2
Произведение: 4.0 + 13.0x^1 + 28.0x^2 + 27.0x^3 + 18.0x^4
Вычисление значения первого полинома при x=2: 17
Дифференцирование первого полинома: 2 + 6x^1
Интегрирование первого полинома: 0 + 1.0x^1 + 1.0x^2 + 1.0x^3
PS C:\Users\super\OneDrive\Рабочий стол\DI\ВУЗ\Воронкин x3\Lab-4_2\prog>
```

Рисунок 7. Выполнение индивидуального задания 2

### Контрольные вопросы:

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

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

`__add__(self, other)` - сложение.  $x + y$  вызывает `x.__add__(y)`

`__radd__(self, other)` - то же самое, но для аргументов, находящихся справа, и только в случае, если для левого операнда не определён соответствующий метод.

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

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

`__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. 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, использующиеся для внутреннего представления в python.

**Вывод:** в результате выполнения работы были приобретены навыки по перегрузке операторов при написании программ с помощью языка программирования Python версии 3.x