

РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Кафедра инфокоммуникаций

**«Элементы объектно-ориентированного программирования в языке
Python»**

Отчет по лабораторной работе № 4.2
по дисциплине «Объектно-ориентированное программирование»

Выполнил студент группы ИВТ-б-о-23-2.

Сумин Никита Сергеевич

Подпись студента _____

Работа защищена « » _____ 2025г.

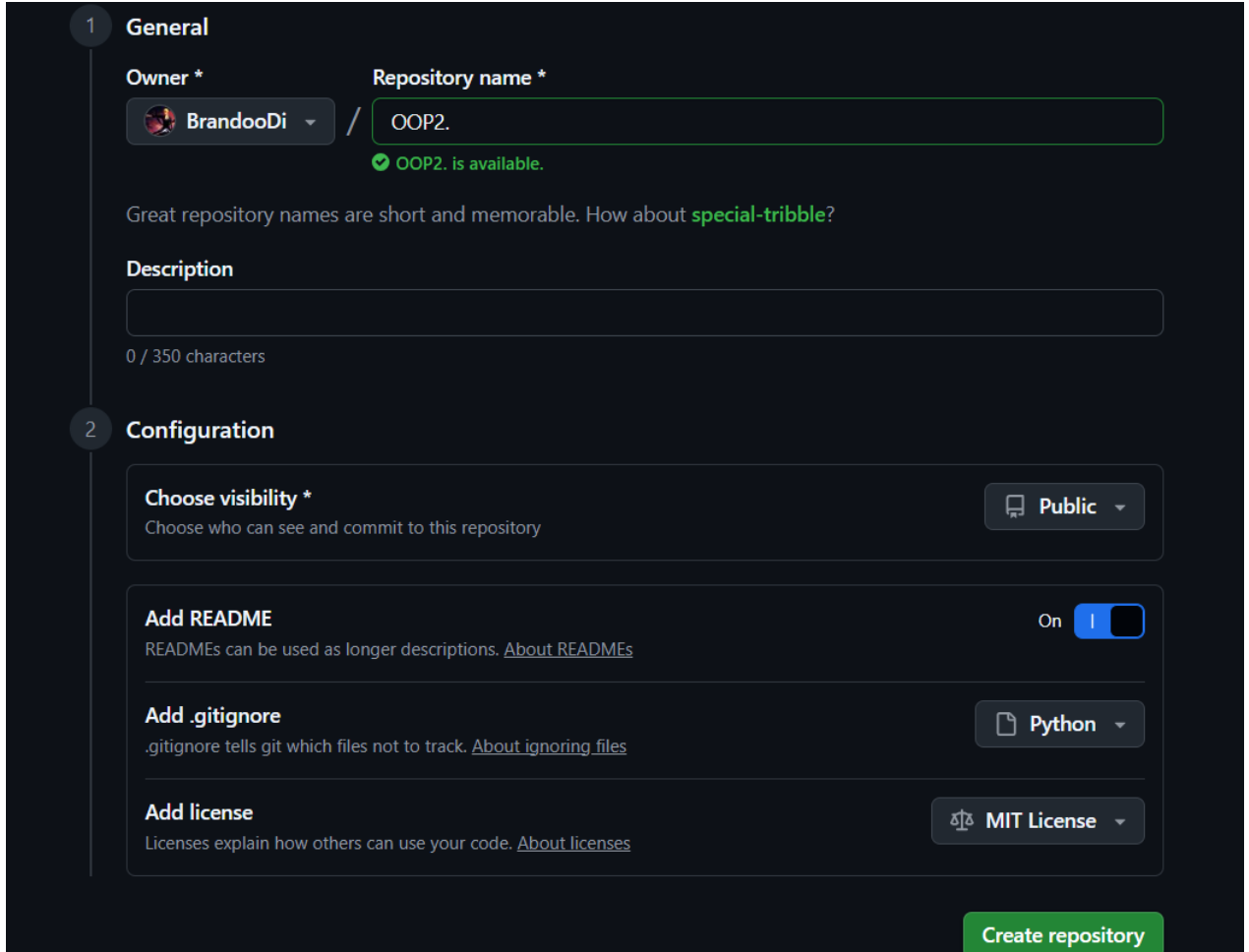
Проверил Воронкин Р.А. _____
(подпись)

Ставрополь 2025

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

Порядок выполнения работы:

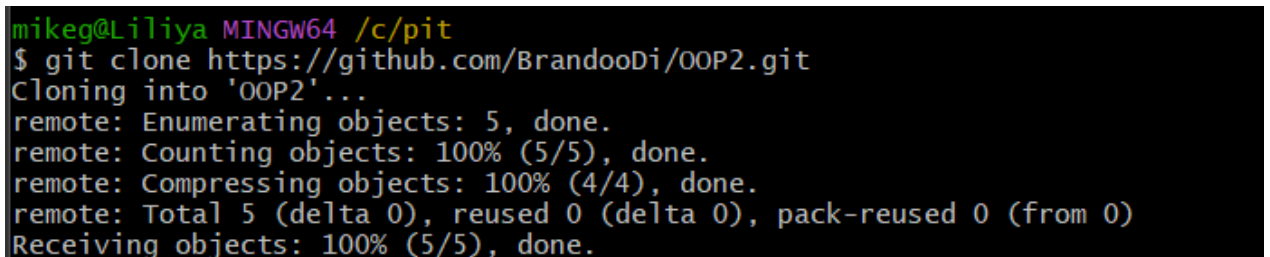
1. Создал общедоступный репозиторий на GitHub, в котором использована лицензия MIT и язык программирования Python.



The screenshot shows the GitHub repository creation process. The 'General' tab is selected, displaying the repository name 'OOP2' and the owner 'BrandooDi'. A green checkmark indicates 'OOP2. is available.' Below this, there is a description field and a 'Description' label. The 'Configuration' tab is also visible, showing options for visibility (Public), adding a README, .gitignore, and license (MIT License). A 'Create repository' button is at the bottom right.

Рисунок 1 - Создание репозитория

2. Выполните клонирование созданного репозитория.



```
mikeg@Liliya MINGW64 /c/pit
$ git clone https://github.com/BrandooDi/OOP2.git
Cloning into 'OOP2'...
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 (from 0)
Receiving objects: 100% (5/5), done.
```

Рисунок 2 - Клонирование репозитория

3. Проработать примеры лабораторной работы.

```
C:\pit\00P2\.venv\Scripts\python.exe C:\pit\00P2\pr1.py
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
```

Рисунок 3 - Результат выполнения примера

4. Выполнить индивидуальные задания.

Задание 1.

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

Поле `first` — дробное число; поле `second` — целое число, показатель степени. Реализовать метод `power()` — возведение числа `first` в степень `second`. Метод должен правильно работать при любых допустимых значениях `first` и `second`.

```
1. Ввод/вывод с помощью операторов << и >>:  
Введите степень в формате 'число ^ степень': 7 ^ 3  
7.0 ^ 3  
  
2. Использование оператора @ для вычисления:  
Результат: 343.0  
  
3. Использование оператора вызова ():  
Результат вызова: 343.0  
  
4. Создание новых объектов с помощью | и &:  
Исходный: 7.0 ^ 3  
exm1 | 5: 7.0 ^ 5  
exm1 & 10.5: 10.5 ^ 3  
  
5. Арифметические операции:  
2.0 ^ 3 + 3.0 ^ 2 = 17.0  
2.0 ^ 3 - 3.0 ^ 2 = -1.0  
2.0 ^ 3 * 3.0 ^ 2 = 72.0  
2.0 ^ 3 / 3.0 ^ 2 = 0.89
```

Рисунок 4 - Результат выполнения индивидуального задания 1

Задание 2.

Прайс-лист компьютерной фирмы включает в себя список моделей продаваемых компьютеров. Одна позиция списка (Model) содержит марку компьютера, тип процессора, частоту работы процессора, объем памяти, объем жесткого диска, объем памяти видеокарты, цену компьютера в условных единицах и количество экземпляров, имеющихся в наличии. Реализовать класс PriceList, полями которого являются дата его создания, номинал условной единицы в рублях и список продаваемых моделей компьютеров. В списке не должно быть двух моделей одинаковой марки. В классе PriceList реализовать методы добавления, изменения и удаления записи о модели, метод поиска информации о модели по марке компьютера, по объему памяти, диска и видеокарты (равно или не меньше заданного), а также метод подсчета общей суммы. Реализовать методы объединения и пересечения прайс-листов. Методы

добавления и изменения принимают в качестве входного параметра объект класса Model. Метод поиска возвращает объект класса Model в качестве результата.

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.

```
Добавлена модель TREIDCOMPUTERS: Intel Core i5 4570/3200 МГц/16 Гб/480 Гб/256 Мб/17.28 ус. ед./12 шт.

Добавлена модель Raskat Strike 520: Intel Core i5-10400F/2900 МГц/16 Гб/1024 Гб/8192 Мб/59.184 ус. ед./6 шт.

Добавлена модель TREIDCOMPUTERS: Intel Core i5 4570/3200 МГц/16 Гб/480 Гб/256 Мб/17.28 ус. ед./12 шт.

Добавлена модель ASUS G10DK-A3400G0320: Intel Core i5-12400F/2500 МГц/16 Гб/1024 Гб/12288 Мб/81.216 ус. ед./3 шт.

Данной модели нет в списке
Добавлена модель Raskat Strike 520: Intel Core i5-10400F/2900 МГц/16 Гб/1024 Гб/8192 Мб/59.184 ус. ед./6 шт.

Данные о модели изменены
Список 1:
Список 09-01-24: 1000 руб./ед:
[TREIDCOMPUTERS: Intel Core i5 4570/3200 МГц/16 Гб/480 Гб/256 Мб/17.28 ус. ед./12 шт.
, Raskat Strike 520: Intel Core i5-10400F/2900 МГц/16 Гб/1024 Гб/8192 Мб/59.184 ус. ед./6 шт.
, Raskat Strike 520: Intel Core i5-10400F/2900 МГц/16 Гб/1024 Гб/8192 Мб/59.184 ус. ед./6 шт.
]

Список 2:
Список 11-12-23: 1000 руб./ед:
```

Рисунок 5 - Результат выполнения индивидуального задания 2

Контрольные вопросы:

1. Какие средства существуют в Python для перегрузки операций?

В Python для перегрузки операций используются магические методы (или "dunder" методы), начинающиеся и заканчивающиеся двумя подчеркиваниями (например, __add__, __sub__, __eq__ и т.д.). Эти методы позволяют объектам классов вести себя определенным образом при использовании операторов.

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

- Арифметические операции: `__add__` (сложение), `__sub__` (вычитание), `__mul__` (умножение), `__truediv__` (деление), и т.д.
- 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 при использовании оператора `+`. Например, `a + b`.
- `__iadd__` (in-place addition): Вызывается при использовании оператора `+=`. Например, `a += b`.
- `__radd__` (right-side addition): Вызывается при использовании оператора `+`, когда левый операнд не поддерживает операцию, но правый операнд поддерживает. Например, `b + a`, если для `b` не определен метод `__add__`.

```
class Example:
    def __add__(self, other):
        return Example(self.value + other.value)
```

```
def __iadd__(self, other):
    self.value += other.value
    return self
```

```
def __radd__(self, other):
    return Example(self.value + other)
```

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

`__new__` отвечает за создание нового экземпляра объекта перед его инициализацией (`__init__`). Метод `__new__` чаще всего используется в неизменяемых типах данных, таких как строки и кортежи. Отличие от `__init__`: `__new__` вызывается перед `__init__` и имеет возможность изменить создаваемый объект или даже вернуть другой объект вместо создаваемого.

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

- `__str__`: Вызывается функцией `str()`. Предназначен для представления объекта в человекочитаемой форме. Если `__str__` отсутствует, вызывается `__repr__`.

- `__repr__`: Вызывается функцией `repr()` и используется для представления объекта в форме, пригодной для воспроизведения (по возможности). Если `__repr__` отсутствует, вызывается `__str__`.

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