

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 №25
дисциплины «Основы программной инженерии»

Выполнил:
Плугатырев Владислав Алексеевич
2 курс, группа ПИЖ-б-о-22-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка и
сопровождение программного
обеспечения», очная форма обучения

(подпись)

Проверил Воронкин Роман Александрович

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

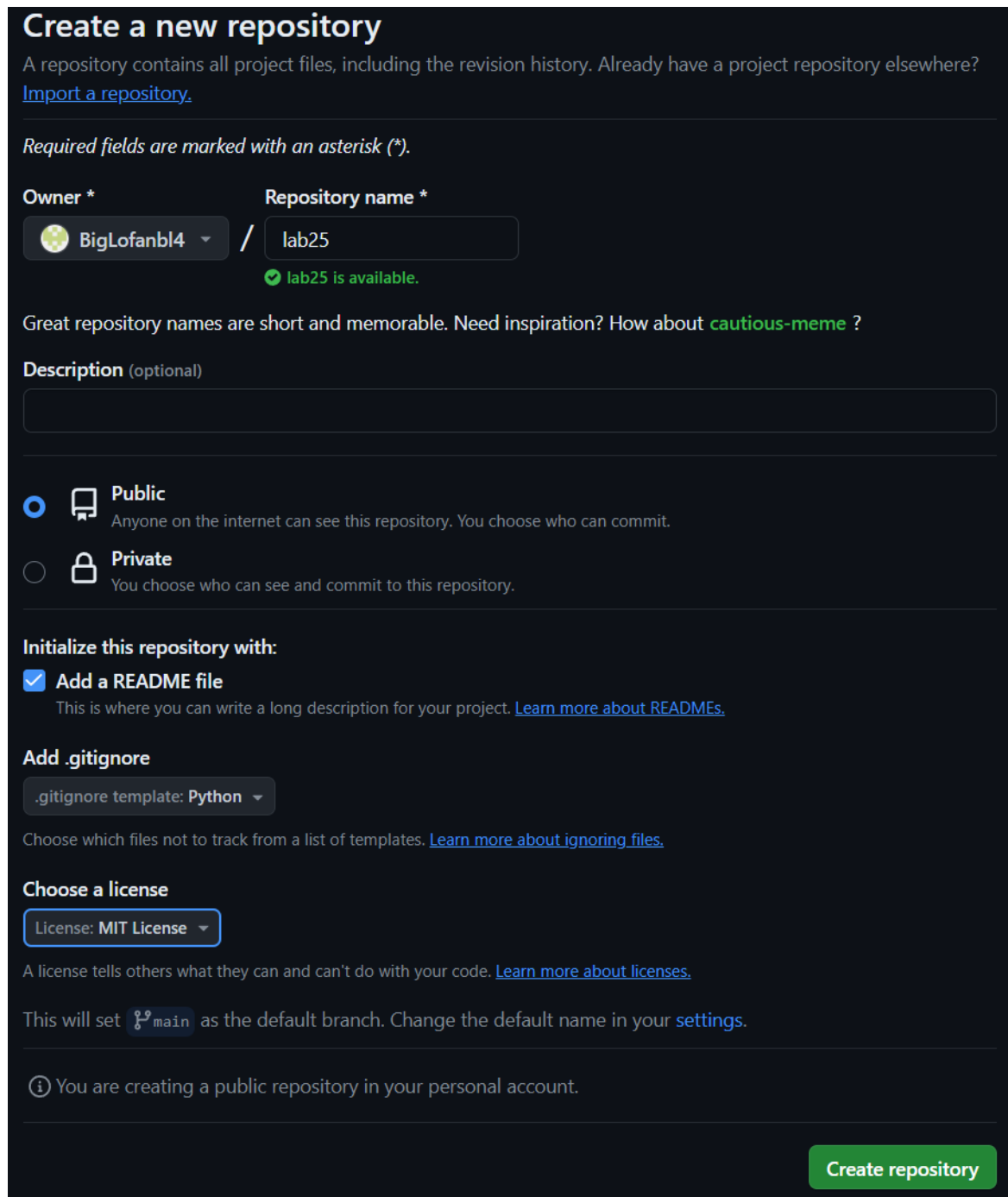
Ставрополь, 2024 г.

Тема: Элементы объектно-ориентированного программирования в языке Python.

Цель работы: приобретение навыков по работе с классами и объектами при написании программ с помощью языка программирования Python версии 3.x.

Ход работы.

1. Создание репозитория.



Create a new repository

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

Required fields are marked with an asterisk ().*

Owner * **Repository name ***

BigLofanbl4 / lab25

✓ lab25 is available.

Great repository names are short and memorable. Need inspiration? How about **cautious-meme** ?

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](#).

i You are creating a public repository in your personal account.

Create repository

Рисунок 1.1 – Создание репозитория

2. Выполнение примеров из лабораторной работы.

```
class Rational:
    def __init__(self, a=0, b=1):
        a = int(a)
        b = int(b)
        if b == 0:
            raise ValueError("Illegal value of the denominator")
        self.__numerator = a
        self.__denominator = b
        self.__reduce()

    # Сокращение дроби.
    def __reduce(self):
        # Функция для нахождения наибольшего общего делителя
        def gcd(a, b):
            if a == 0:
                return b
            elif b == 0:
                return a
            elif a >= b:
                return gcd(a % b, b)
            else:
                return gcd(a, b % a)

        sign = 1
        if (self.__numerator > 0 and self.__denominator < 0) or (
            self.__numerator < 0 and self.__denominator > 0
        ):
            sign = -1
        a, b = abs(self.__numerator), abs(self.__denominator)
        c = gcd(a, b)
        self.__numerator = sign * (a // c)
        self.__denominator = b // c
```

Рисунок 2.1 – Код примера

3. Выполнение первого задания: выполнить индивидуальное задание 1 лабораторной работы 4.1, максимально задействовав имеющиеся в Python средства перегрузки операторов.

```
import math

class RightTriangle:
    def __init__(self, first, second):
        if not isinstance(first, (int, float)) or not isinstance(second, (int, float)):
            raise ValueError("The first and second inputs must be numbers")
        if first <= 0 or second <= 0:
            raise ValueError("The numbers must be > 0")
        self.first = first
        self.second = second

    def read(self):
        try:
            first = float(input("Enter first value > 0: "))
            second = float(input("Enter second value > 0: "))
            if first <= 0 or second <= 0:
                raise ValueError("The numbers must be > 0")
            self.first = first
            self.second = second
        except ValueError as error:
            print(error)
            self.read()

    # заменили метод display на __str__
    def __str__(self):
        return f"The catheter a={self.first}, the catheter b={self.second}"

    # заменили метод hypotenuse на __call__
    def __call__(self):
        return math.sqrt(self.first**2 + self.second**2)

def make_right_triangle(first, second):
    try:
        return RightTriangle(first, second)
    except ValueError as error:
        print(error)

if __name__ == "__main__":
    triangle1 = make_right_triangle(3, 4)
    print(triangle1)
    print(triangle1())

    triangle2 = make_right_triangle(3, 4)
    triangle2.read()
    print(triangle2)
    print(triangle2())
```

Рисунок 3.1 – Код программы

```
The catheter a=3, the catheter b=4
5.0
Enter first value > 0: 1
Enter second value > 0: 4
The catheter a=1.0, the catheter b=4.0
4.123105625617661
```

Рисунок 3.2 – Вывод программы

4. Выполнение второго задания: Информационная запись о книге в библиотеке содержит следующие поля: автор, название, год издания, издательство, цена. Для моделирования учетной карточки абонента реализовать класс `Subscriber`, содержащий фамилию абонента, его библиотечный номер и список взятых в библиотеке книг. Один элемент списка состоит из информационной записи о книге, даты выдачи, требуемой даты возврата и признака возврата. Реализовать методы добавления книг в список и удаления книг из него; метод поиска книг, подлежащих возврату; методы поиска по автору, издательству и году издания; метод вычисления стоимости всех подлежащих возврату книг. Реализовать операцию слияния двух учетных карточек, операцию пересечения и вычисления разности. Реализовать операцию генерации конкретного объекта `Debt` (долг), содержащего список книг, подлежащих возврату из объекта типа `Subscriber`.

```
class Subscriber:
    MAX_BOOKS = 100

    def __init__(self, surname, id, books=[], size=MAX_BOOKS):
        if len(books) > size:
            raise ValueError("Amount of books > max size")
        self.surname = surname
        self.id = id
        self.books = books
        self.__size = size
        self.__count = len(books)

    @property
    def size(self):
        return self.__size

    @property
    def count(self):
        return self.__count

    def add_book(self, book, issue_date, return_date):
        if self.__count < self.__size:
            self.books.append({"book": book, "issue_date": issue_date, "return_date": return_date, "returned": False})
            self.__count += 1
        else:
            print("Reached maximum size")
```

Рисунок 4.1 – Код программы

```
Books to return:
[{'book': Book('Author1', 'Book1', 2020, 'Publisher1', 10.99), 'issue_date': '2024-01-01', 'return_date': '2024-01-31', 'returned': False}, {'book': Book('Author2', 'Book2', 2019, 'Publisher2', 9.99), 'issue_date': '2024-02-01', 'return_date': '2024-02-28', 'returned': False}, {'book': Book('Author2', 'Book2', 2019, 'Publisher2', 9.99), 'issue_date': '2024-02-01', 'return_date': '2024-02-28', 'returned': False}]

Books by author:
[{'book': Book('Author1', 'Book1', 2020, 'Publisher1', 10.99), 'issue_date': '2024-01-01', 'return_date': '2024-01-31', 'returned': False}]

Books by publisher:
[{'book': Book('Author1', 'Book1', 2020, 'Publisher1', 10.99), 'issue_date': '2024-01-01', 'return_date': '2024-01-31', 'returned': False}]

Price of not returned books:
30.97

Merge:
Subscriber(Ascorbin & Lofanb14, 12345 & 54321, [{'book': Book('Author1', 'Book1', 2020, 'Publisher1', 10.99), 'issue_date': '2024-01-01', 'return_date': '2024-01-31', 'returned': False}, {'book': Book('Author2', 'Book2', 2019, 'Publisher2', 9.99), 'issue_date': '2024-02-01', 'return_date': '2024-02-28', 'returned': False}, {'book': Book('Author2', 'Book2', 2019, 'Publisher2', 9.99), 'issue_date': '2024-02-01', 'return_date': '2024-02-28', 'returned': False}, {'book': Book('Author1', 'Book1', 2020, 'Publisher1', 10.99), 'issue_date': '2024-01-01', 'return_date': '2024-01-31', 'returned': False}, {'book': Book('Author2', 'Book2', 2019, 'Publisher2', 9.99), 'issue_date': '2024-02-01', 'return_date': '2024-02-28', 'returned': False}, {'book': Book('Author2', 'Book2', 2019, 'Publisher2', 9.99), 'issue_date': '2024-02-01', 'return_date': '2024-02-28', 'returned': False}])

Intersection:
Subscriber(Ascorbin & Lofanb14, 12345 & 54321, [{'book': Book('Author1', 'Book1', 2020, 'Publisher1', 10.99), 'issue_date': '2024-01-01', 'return_date': '2024-01-31', 'returned': False}, {'book': Book('Author2', 'Book2', 2019, 'Publisher2', 9.99), 'issue_date': '2024-02-01', 'return_date': '2024-02-28', 'returned': False}, {'book': Book('Author2', 'Book2', 2019, 'Publisher2', 9.99), 'issue_date': '2024-02-01', 'return_date': '2024-02-28', 'returned': False}])

Difference:
Subscriber(Ascorbin & Lofanb14, 12345 & 54321, [])

Object Debt:
[{'book': Book('Author1', 'Book1', 2020, 'Publisher1', 10.99), 'issue_date': '2024-01-01', 'return_date': '2024-01-31', 'returned': False}, {'book': Book('Author2', 'Book2', 2019, 'Publisher2', 9.99), 'issue_date': '2024-02-01', 'return_date': '2024-02-28', 'returned': False}, {'book': Book('Author2', 'Book2', 2019, 'Publisher2', 9.99), 'issue_date': '2024-02-01', 'return_date': '2024-02-28', 'returned': False}]
```

Рисунок 4.2 – Вывод программы

Ответы на контрольные вопросы

1. Средства перегрузки операций в Python:

В Python перегрузка операций реализуется через специальные методы, часто называемые "магическими" методами. Эти методы имеют двойные подчеркивания в начале и конце их имен (например, «__add__» для операции сложения, «__eq__» для операции равенства).

2. Методы перегрузки арифметических операций и операций отношения:

- «Арифметические операции»: «__add__» (сложение), «__sub__» (вычитание), «__mul__» (умножение), «__truediv__» (деление), «__floordiv__» (целочисленное деление), «__mod__» (модуль), «__pow__» (возведение в степень), «__neg__» (отрицание), «__pos__» (унарный плюс).

- «Операции отношения»: «__eq__» (равенство), «__ne__» (неравенство), «__lt__» (меньше), «__le__» (меньше или равно), «__gt__» (больше), «__ge__» (больше или равно).

3. Случаи вызова «__add__», «__iadd__», и «__radd__»:

- «__add__(self, other)»: Вызывается, когда объект класса находится в левой части операции сложения (например, «a + b»).

- «__iadd__(self, other)»: Вызывается для реализации операции сложения с присваиванием (например, «a += b»). Обычно метод должен возвращать «self» после модификации.

- «__radd__(self, other)»: Вызывается, когда объект класса находится в правой части операции сложения и левый операнд не поддерживает сложение с правым (например, «b + a», где «b» не имеет метода «__add__» для работы с «a»).

4. Назначение метода «__new__» и его отличия от «__init__»:

«__new__» – это статический метод, который вызывается для создания нового экземпляра класса перед «__init__». Этот метод возвращает новый экземпляр класса (новый объект). Он обычно используется для изменения

процесса создания объекта у неизменяемых типов (как «tuple», «str») или при реализации паттерна Singleton.

«__init__» – метод, который вызывается после создания объекта (через «__new__») для его инициализации с начальными значениями.

5. Отличия между «__str__» и «__repr__»:

«__str__» – возвращает строковое представление объекта, предназначенное для отображения пользователю. Более "дружественное" представление.

«__repr__» – возвращает официальное строковое представление объекта, которое должно быть максимально точным. Идеально, чтобы результат этого метода можно было использовать для воссоздания объекта при его вызове через «eval()».

«__str__» больше ориентирован на конечного пользователя, в то время как «__repr__» предназначен для программистов. Если «__str__» не определен, Python в качестве запасного варианта использует «__repr__».