

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:
Матвеев Александр Иванович
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: Перегрузка операторов в языке Python.

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

Ход работы.

1. Создание нового репозитория с лицензией MIT.

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 * SashkaHacker / **Repository name *** laba10
✔ laba10 is available.

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

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.](#)

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

Create repository

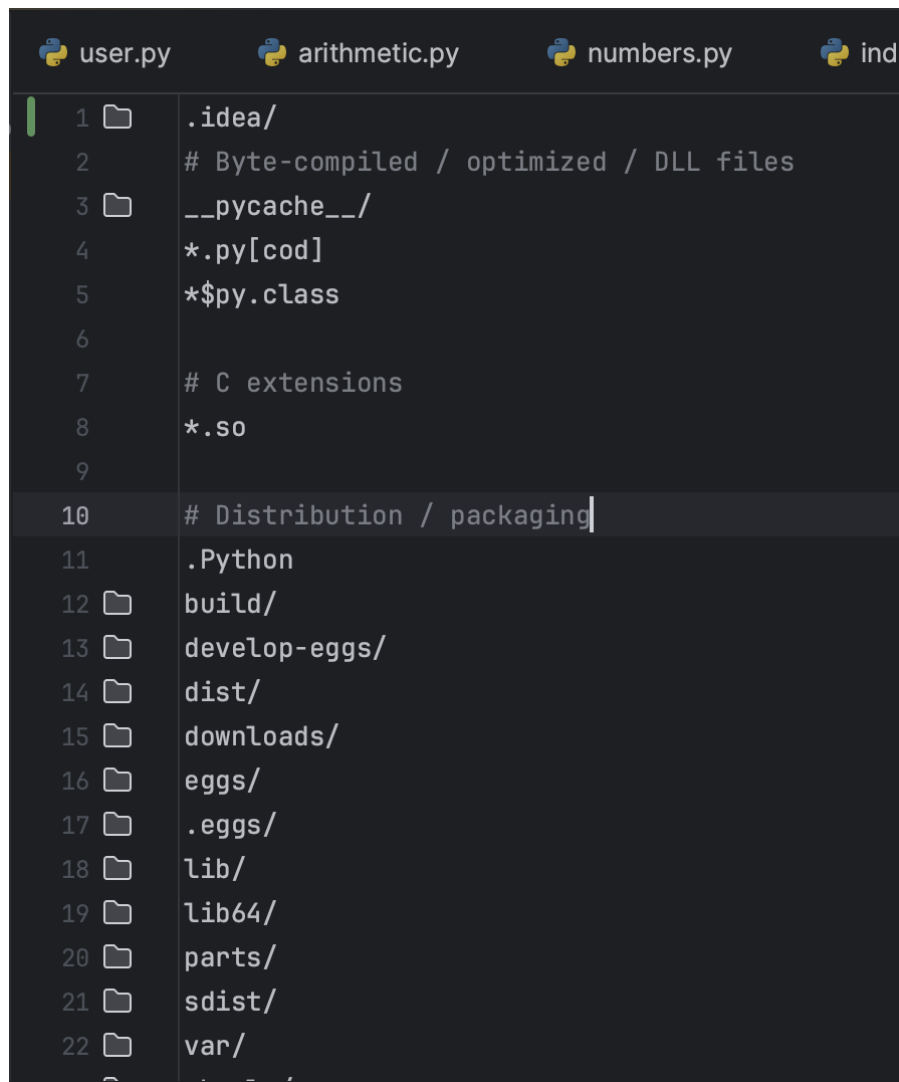
Рисунок 1 – Создание репозитория

2. Клонировал репозиторий на рабочий ПК.

```
[→ GitHub git clone https://github.com/SashkaHacker/laba16.git
Cloning into 'laba16'...
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.
```

Рисунок 2 – Клонирование репозитория

3. Дополнил файл .gitignore необходимыми инструкциями.



The screenshot shows a code editor with a dark theme. At the top, there are four tabs: 'user.py', 'arithmetic.py', 'numbers.py', and 'ind'. The active tab is 'user.py'. The editor displays the content of the '.gitignore' file, which is numbered from 1 to 22. The content includes comments and file/folder names to be ignored:

```
1  .idea/
2  # Byte-compiled / optimized / DLL files
3  __pycache__/
4  *.py[cod]
5  *$py.class
6
7  # C extensions
8  *.so
9
10 # Distribution / packaging
11 .Python
12 build/
13 develop-eggs/
14 dist/
15 downloads/
16 eggs/
17 .eggs/
18 lib/
19 lib64/
20 parts/
21 sdist/
22 var/
```

Рисунок 3 – Файл .gitignore

4. Проработка примера №1 из лабораторной работы.

```
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
```

Рисунок 4 – Демонстрация работы примера №1

5. Выполнение индивидуального задания №1 (Вариант - 10).

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

# Вариант 10

class LinearEquation:
    def __init__(self, first, second):
        if not isinstance(first, (int, float)) or not isinstance(second, (
            int, float)):
            raise ValueError("Коэффициенты должны быть числами.")
        self.first = first
        self.second = second

    def read(self):
        self.first = float(input("Введите коэффициент a: "))
        self.second = float(input("Введите коэффициент b: "))

    # Заменяет метод display на __repr__
    def __repr__(self):
        return f"Линейное уравнение: {self.first}x + {self.second} = 0"

    # Заменяет метод function на __call__
    def __call__(self, x):
        return self.first * x + self.second

if __name__ == '__main__':
    equation1 = LinearEquation(first=1, second=2)
    print(equation1)
    print(equation1(10))

    equation2 = LinearEquation(first=0, second=0)
    equation2.read()
    print(equation2)
    x = float(input("Введите значение x для вычисления функции: "))
    print(f"Значение функции: {equation2(x)}")
```

Рисунок 5 – Код программы

```
Линейное уравнение: 1x + 2 = 0
12
Введите коэффициент a: 1
Введите коэффициент b: 2
Линейное уравнение: 1.0x + 2.0 = 0
Введите значение x для вычисления функции: 10
Значение функции: 12.0
```

Рисунок 6 – Демонстрация работы программы

6. Выполнение индивидуального задания №2 (Вариант - 10).

```
{'вопрос': 'Вопрос 1', 'ответы': ['a', 'b', 'c', 'd', 'e'], 'правильный': 1, 'баллы': 2}, {'вопрос': 'Вопрос 2', 'ответы': ['a', 'b', 'c', 'd', 'e'], 'правильный': 2, 'баллы': 2}]
Такой вопрос уже есть в тесте.
2
1
1
1
1
```

Рисунок 7 – Демонстрация работы программы

Контрольные вопросы:

1. В Python для перегрузки операций используются специальные методы, называемые "магическими" методами или методами двойного подчеркивания. Они имеют имена вида `__op__`, где `op` обозначает операцию. Например, `__add__` для сложения, `__eq__` для проверки равенства.

2. Методы для перегрузки арифметических операций включают `__add__` (сложение), `__sub__` (вычитание), `__mul__` (умножение), `__truediv__` (деление), `__floordiv__` (целочисленное деление), `__mod__` (остаток от деления), `__pow__` (возведение в степень). Для перегрузки операций отношения используются `__eq__` (равно), `__ne__` (не равно), `__lt__` (меньше), `__le__` (меньше или равно), `__gt__` (больше), `__ge__` (больше или равно).

3. `__add__` вызывается при использовании оператора `+` между экземплярами класса, например, `a + b`. `__iadd__` вызывается для операции `+=`, например, `a += b`, и обычно модифицирует объект на месте. `__radd__` вызывается, если левый операнд не поддерживает сложение с правым, например, если `b` не имеет метода `__add__` для сложения с `a`, Python попытается вызвать `a.__radd__(b)`.

4. Метод `__new__` используется для создания нового объекта перед инициализацией. Он отличается от `__init__`, который используется для инициализации объекта после его создания. `__new__` является статическим методом и возвращает экземпляр класса, а `__init__` не возвращает значение и называется конструктором экземпляра.

5. Метод `__str__` предназначен для возвращения понятного человеку представления объекта, часто используется для преобразования объекта в строку, например, при печати или преобразовании в `str`. Метод `__repr__` предназначен для возвращения официального строкового представления объекта, которое, если возможно, должно быть допустимым выражением Python, с помощью которого можно воссоздать объект с теми же данными. `__repr__` предназначен скорее для разработчика, чтобы понять, как объект устроен внутри..