

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 №26
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: Наследование и полиморфизм в языке Python

Цель работы: приобретение навыков по созданию иерархии классов 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

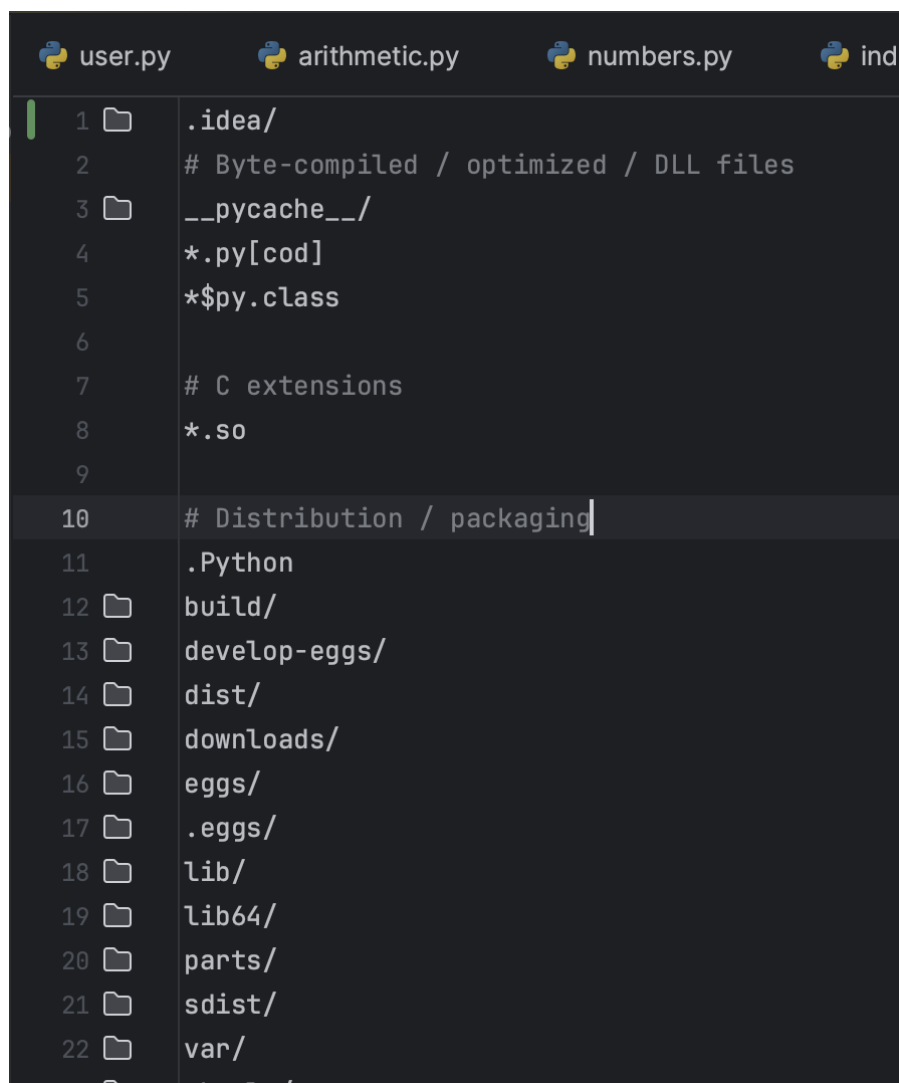
Рисунок 1 – Создание репозитория

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 image 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 a '.gitignore' file. The file is numbered from 1 to 22. The content is as follows:

```
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 примеров из лабораторной работы.

```
3/4
Введите обыкновенную дробь: 2/3
2/3
17/12
-1/12
1/2
8/9
```

Рисунок 4 – Демонстрация работы примера №1

```
I have 3 sides
I have 4 sides
I have 5 sides
I have 6 sides
```

Рисунок 5 – Демонстрация работы примера №2

```
I can walk and run
I can crawl
I can bark
I can roar
```

Рисунок 6 – Демонстрация работы примера №3

5. Выполнение задания №1.

```
В команде "А" находится 43 солдат
В команде "В" находится 57 солдат
2 3
```

Рисунок 7 – Демонстрация работы задания №1

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

```

if __name__ == "__main__":
    # Тестирование класса Triad
    triad1 = Triad(a: 1, b: 2, c: 3)
    triad2 = Triad(a: 1, b: 2, c: 3)

    assert triad1 == triad2, "Triad1 должен быть равен Triad2"

    # Тестирование класса Date
    date1 = Date(year: 2020, month: 1, day: 1)
    date2 = Date(year: 2020, month: 1, day: 2)
    date3 = Date(year: 2020, month: 1, day: 1)
    date4 = Date(year: 2021, month: 1, day: 1)

    assert date1 < date2, "date1 должна быть меньше date2"

    assert date1 <= date3, "date1 должна быть меньше или равна date3"
    assert date1 <= date2, "date1 должна быть меньше date2"

    assert date1 == date3, "date1 должна быть равна date3"

    assert date1 != date4, "date1 не должна быть равна date4"

    assert date4 > date3, "date4 должна быть больше date3"

    assert date4 >= date1, "date4 должна быть больше или равна date1"
    assert date1 >= date3, "date1 должна быть больше или равна date3"

    print("Все тесты пройдены успешно!")

```

Рисунок 8 – Тестирующий код программы

```

C:\Users\Sashka\Documents\GitHub
Все тесты пройдены успешно!

```

Рисунок 9 – Демонстрация работы программы

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

```

Triad after increment: 2020, 2, 1
Triad after increment: 0, 0, 0
date1 < date2: False
date1 > date2: False

```

Рисунок 10 – Демонстрация работы программы

Контрольные вопросы:

1. Наследование в Python — это механизм, позволяющий одному классу (дочернему или производному) наследовать атрибуты и методы другого класса (родительского или базового). Это позволяет избежать дублирования кода и способствует повторному использованию кода. Наследование реализуется с помощью определения класса, где в скобках указывается родительский класс.

2. Полиморфизм в Python — это принцип, согласно которому различные классы могут быть использованы с одним и тем же интерфейсом. Это означает, что функции могут использовать объекты разных классов, если эти классы реализуют определенные методы или атрибуты. В Python полиморфизм обычно реализуется через динамическую типизацию и утиную типизацию.

3. "Утиная" типизация (duck typing) — это стиль типизации, используемый в динамических языках, таких как Python, где тип объекта определяется его текущим набором методов и свойств, а не явным наследованием от какого-либо конкретного класса. Принцип "утиная" типизация можно описать фразой: "Если это крикает как утка и плавает как утка, то это, вероятно, утка".

4. Модуль abc (Abstract Base Classes) в Python предназначен для создания абстрактных базовых классов. Он предоставляет метакласс ABC и декоратор `abstractmethod`, которые используются для определения абстрактных методов и свойств в классах. Это позволяет создать класс, который не может быть инстанцирован напрямую, и устанавливает контракт для дочерних классов, которые должны реализовать абстрактные методы.

5. Чтобы сделать метод класса абстрактным, его нужно объявить с декоратором `abstractmethod`. Это требует также наследования от класса ABC или использования метакласса `ABCMeta`.

6. Чтобы сделать свойство класса абстрактным, можно использовать `abc.abstractmethod` вместе с `property`.

7. Функция `isinstance` используется для проверки, принадлежит ли объект к определенному классу или его подклассам. Это полезно для проверки типов объектов во время выполнения.