

**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 № 4.1**  
**по дисциплине «Объектно-ориентированное программирование»**

Выполнил студент группы ИВТ-б-о-21-1

Шайдеров Дмитрий Викторович.

«10» октября 2023г.

Подпись студента \_\_\_\_\_

Работа защищена «   » \_\_\_\_\_ 20\_\_ г.

Проверил Воронкин Р.А. \_\_\_\_\_  
(подпись)

Ставрополь 2023

**Цель работы:** приобретение навыков по работе с классами и объектами 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.

**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 \*


 dshayderov / OOP\_lw\_4.1

✔ OOP\_lw\_4.1 is available.

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

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

Рисунок 1 - Создание репозитория

2. Выполните клонирование созданного репозитория.

```
C:\Users\Asus\Desktop\Учеба\5 семестр\ООП>git clone https://github.com/dshayderov/OOP_lw_4.1.git
Cloning into 'OOP_lw_4.1'...
remote: Enumerating objects: 11, done.
remote: Counting objects: 100% (11/11), done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 11 (delta 2), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (11/11), 4.12 KiB | 843.00 KiB/s, done.
Resolving deltas: 100% (2/2), done.
```

Рисунок 2 - Клонирование репозитория

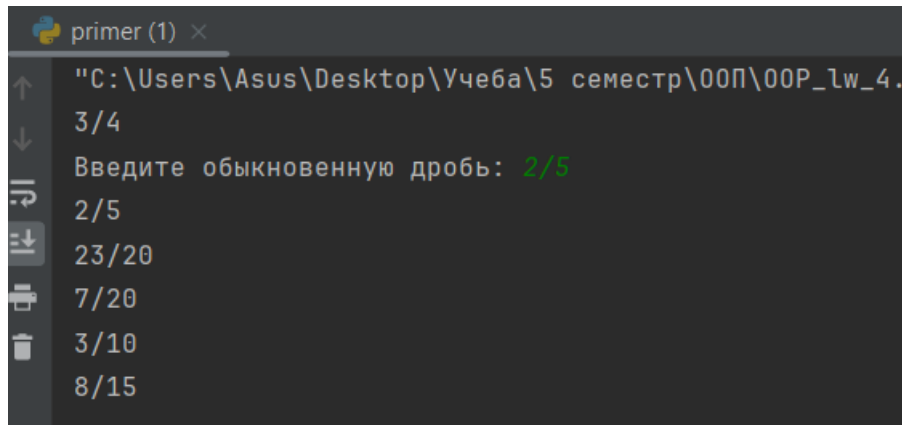
3. Организуйте свой репозиторий в соответствии с моделью ветвления git-flow.

```
C:\Users\Asus\Desktop\Учеба\5 семестр\ООП\ООР_lw_4.1>git checkout -b develop
Switched to a new branch 'develop'

C:\Users\Asus\Desktop\Учеба\5 семестр\ООП\ООР_lw_4.1>
```

Рисунок 3 - Ветвление по модели git-flow

4. Проработать примеры лабораторной работы.



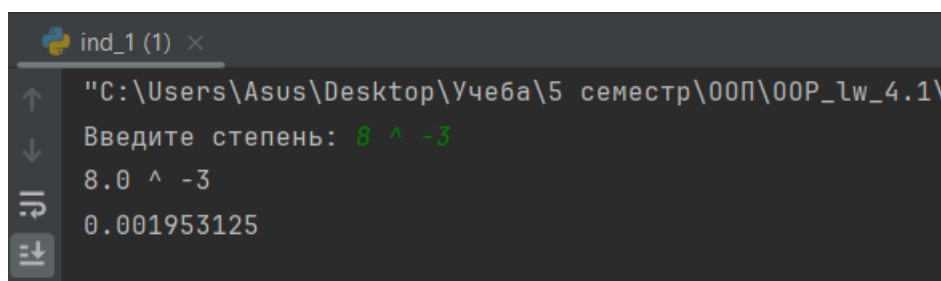
```
primer (1) x
"C:\Users\Asus\Desktop\Учеба\5 семестр\ООП\ООР_lw_4.1\
3/4
Введите обыкновенную дробь: 2/5
2/5
23/20
7/20
3/10
8/15
```

Рисунок 4 - Результат выполнения примера

5. Выполнить индивидуальные задания.

### Задание 1. Вариант 21(1)

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.

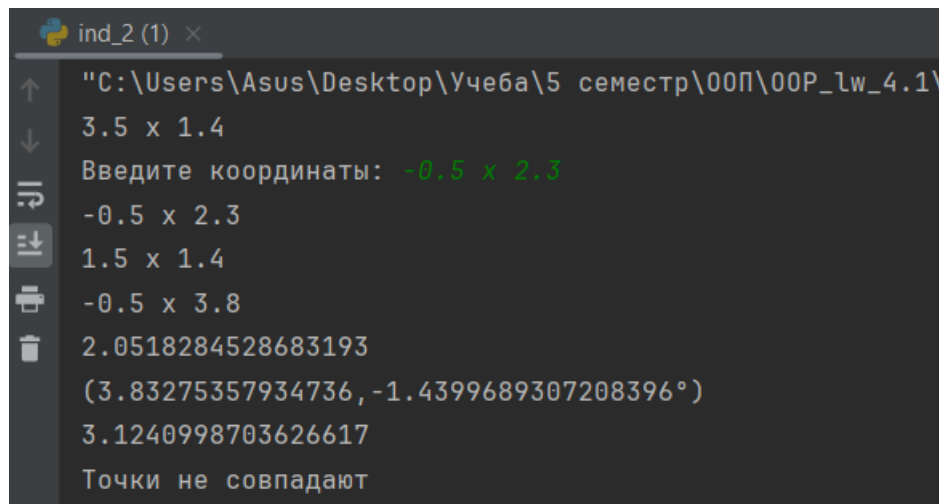


```
ind_1 (1) x
"C:\Users\Asus\Desktop\Учеба\5 семестр\ООП\ООР_lw_4.1\
Введите степень: 8 ^ -3
8.0 ^ -3
0.001953125
```

Рисунок 5 - Результат выполнения индивидуального задания 1

### Задание 2. Вариант 21(6)

Создать класс Point для работы с точками на плоскости. Координаты точки — декартовы. Обязательно должны быть реализованы: перемещение точки по оси X, перемещение по оси Y, определение расстояния до начала координат, расстояния между двумя точками, преобразование в полярные координаты, сравнение на совпадение и несовпадение.



```
ind_2 (1) ×
"C:\Users\Asus\Desktop\Учеба\5 семестр\00П\00P_lw_4.1\
3.5 x 1.4
Введите координаты: -0.5 x 2.3
-0.5 x 2.3
1.5 x 1.4
-0.5 x 3.8
2.0518284528683193
(3.83275357934736, -1.4399689307208396°)
3.1240998703626617
Точки не совпадают
```

Рисунок 6 - Результат выполнения индивидуального задания 2

### Контрольные вопросы:

#### 1. Как осуществляется объявление класса в языке Python?

Классы объявляются с помощью ключевого слова `class` и имени класса:

# class syntax

`class MyClass:`

`var = ... # некоторая переменная`

`def do_smt(self):`

`# какой-то метод`

#### 2. Чем атрибуты класса отличаются от атрибутов экземпляра?

Атрибут класса - это атрибут, общий для всех экземпляров класса. Атрибуты класса определены внутри класса, но вне каких-либо методов. Их значения одинаковы для всех экземпляров этого класса. Так что вы можете рассматривать их как тип значений по умолчанию для всех наших объектов.

Атрибуты экземпляра определяются в методах и хранят информацию, специфичную для экземпляра.

#### 3. Каково назначение методов класса?

Классы позволяют определять данные и поведение похожих объектов. Поведение описывается методами. Метод похож на функцию тем, что это блок кода, который имеет имя и выполняет определенное действие. Методы,

однако, не являются независимыми, поскольку они определены внутри класса.

#### **4. Для чего предназначен метод `__init__()` класса?**

Метод `__init__` является конструктором. Конструкторы - это концепция объектно-ориентированного программирования. Класс может иметь один и только один конструктор. Если `__init__` определен внутри класса, он автоматически вызывается при создании нового экземпляра класса.

Метод `__init__` указывает, какие атрибуты будут у экземпляров нашего класса.

#### **5. Каково назначение `self` ?**

Аргумент `self` представляет конкретный экземпляр класса и позволяет нам получить доступ к его атрибутам и методам. Важно использовать параметр `self` внутри метода, если мы хотим сохранить значения экземпляра для последующего использования.

В большинстве случаев нам также необходимо использовать параметр `self` в других методах, потому что при вызове метода первым аргументом, который ему передается, является сам объект.

#### **6. Как добавить атрибуты в класс?**

Например, мы хотим видеть информацию о всех видах наших питомцев. Мы могли бы записать ее в самом классе с самого начала или создать переменную следующим образом:

```
Pet.all_specs = [tom.сpec, avocado.spec, ben.spec]
```

#### **7. Как осуществляется управление доступом к методам и атрибутам в языке Python?**

Хорошим тоном считается, что для чтения/изменения какого-то атрибута должны использоваться специальные методы, которые называются `getter/setter`, их можно реализовать, но ничего не мешает изменить атрибут напрямую. При этом есть соглашение, что метод или атрибут, который начинается с нижнего подчеркивания, является скрытым, и снаружи класса трогать его не нужно (хотя сделать это можно).

## **8. Каково назначение функции `isinstance` ?**

Встроенная функция `isinstance(obj, Cls)` , используемая при реализации методов арифметических операций и операций отношения, позволяет узнать что некоторый объект `obj` является либо экземпляром класса `Cls` либо экземпляром одного из потомков класса `Cls`.

**Вывод:** были приобретены навыки по работе с классами и объектами при написании программ с помощью языка программирования Python версии 3.x.