

РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образо-
вательное 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
Богадуров Василий Игоревич.

« » _____ 20__г.

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 используется 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 *

 ItsMyLife1337 ▾

Repository name *

OOP_4.1

✓ OOP_4.1 is available.

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

Description (optional)

Лабораторная работа 4.1 Элементы объектно-ориентированного программирования в языке Python

Рисунок 1 – Создание репозитория

2. Выполнение клонирования созданного репозитория.

```
C:\Users\Admin>cd /d c:\users\admin\desktop\git
c:\Users\Admin\Desktop\git>git clone https://github.com/ItsMyLife1337/OOP_4.1.git
Cloning into 'OOP_4.1'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.
c:\Users\Admin\Desktop\git>cd /d c:\users\admin\desktop\git\OOP_4.1
```

Рисунок 2 – Клонирование репозитория

3. Организация репозитория в соответствии с моделью ветвления git-flow.

```

c:\Users\Admin\Desktop\git\OOP_4.1>git flow init

Which branch should be used for bringing forth production releases?
- main
Branch name for production releases: [main]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [C:/Users/Admin/Desktop/git/OOP_4.1/.git/hooks]

c:\Users\Admin\Desktop\git\OOP_4.1>

```

Рисунок 3 – Ветвление по модели git-flow

4. Проработка примера лабораторной работы.

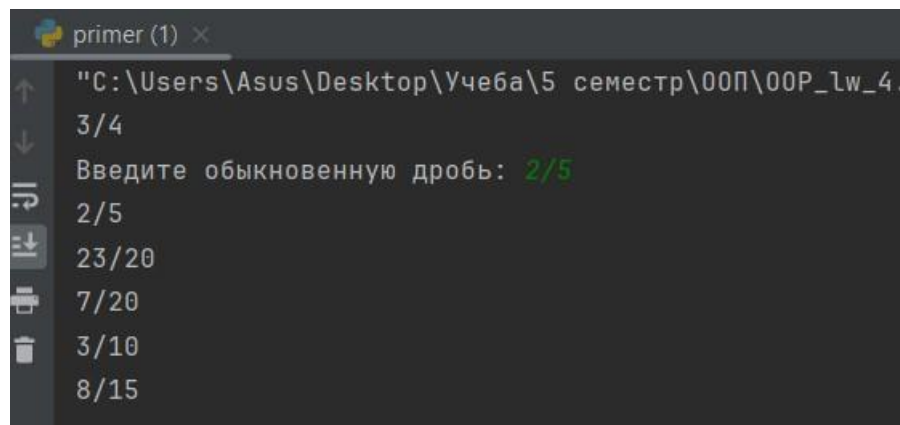


Рисунок 4 – Результат выполнения примера

5. Выполнение индивидуального задания.

Задание 1. Вариант 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.

```

PS C:\Users\Admin> & C:/Users/Admin/AppData/Local/Programs/Python/Python311/python.exe "c:/Users/Admin/Desktop/git/OOP_4.1/My project/ind_1.py"
Введите степень: 8 ^ 3
8.0 ^ 3
512.0
PS C:\Users\Admin>

```

Рисунок 5 – Результат выполнения индивидуального задания
№1

Задание 2. Вариант 1

1. Создать класс `Vector3D`, задаваемый тройкой координат. Обязательно должны быть реализованы: сложение и вычитание векторов, скалярное произведение векторов, умножение на скаляр, сравнение векторов, вычисление длины вектора, сравнение длины векторов.

```
ПРОБЛЕМА    выходные данные    КОНСОЛЬ ОТЛАДКИ    ТЕР
(6.0, 6.0, 6.0)
Разность векторов:
(2.0, 0.0, -2.0)
Скалярное произведение векторов: 25.0
Умножение вектора на скаляр:
(8.0, 6.0, 4.0)
Сравнение векторов (равенство): False
Длина вектора v1: 5.385164807134504
Длина вектора v2: 5.385164807134504
Сравнение длины векторов (v1 > v2): False
Сравнение длины векторов (v1 < v2): False
PS C:\Users\Admin>
```

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 внутрикласса.

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.spec, 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: в результате выполнения лабораторной работы были приобретены навыки по работе с классами и объектами при написании программ с помощью языка программирования Python версии 3.x.