Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт перспективной инженерии

Департамент цифровых, робототехнических систем и электроники

**ОТЧЕТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ №4.2**

**дисциплины «Объектно-ориентированное программирование»**

|  |  |  |
| --- | --- | --- |
|  | Выполнил:  Бересланов Рамазан  3 курс, группа ПИЖ-б-о-22-1,  09.03.04 «Программная инженерия», направленность (профиль) «Разработка и сопровождение программного обеспечения», очная форма обучения  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | |
|  | Руководитель практики:  Хацукова А.И., ассистент департамента цифровых, робототехнических систем и электроники  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | |
|  |  |  |

Отчет защищен с оценкой \_\_\_\_\_\_\_\_\_\_\_ Дата защиты\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ставрополь, 2024 г.

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

Ход работы:

1. Изучил теоретический материал.
2. Создал общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и язык программирования Python.

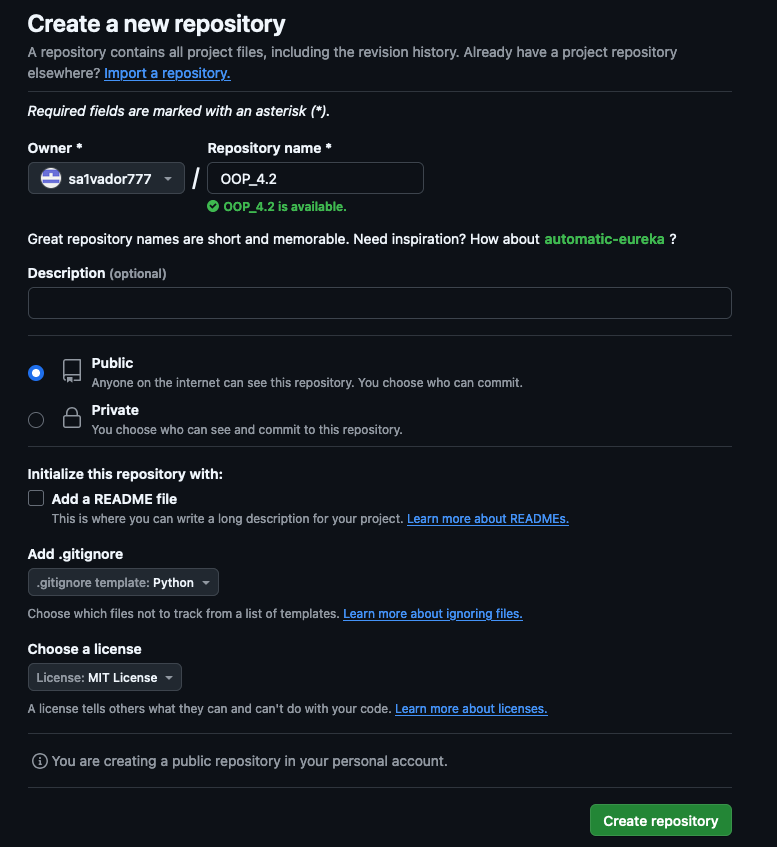


Рисунок 1 – Создание репозитория

1. Склонировал репозиторий на ПК.

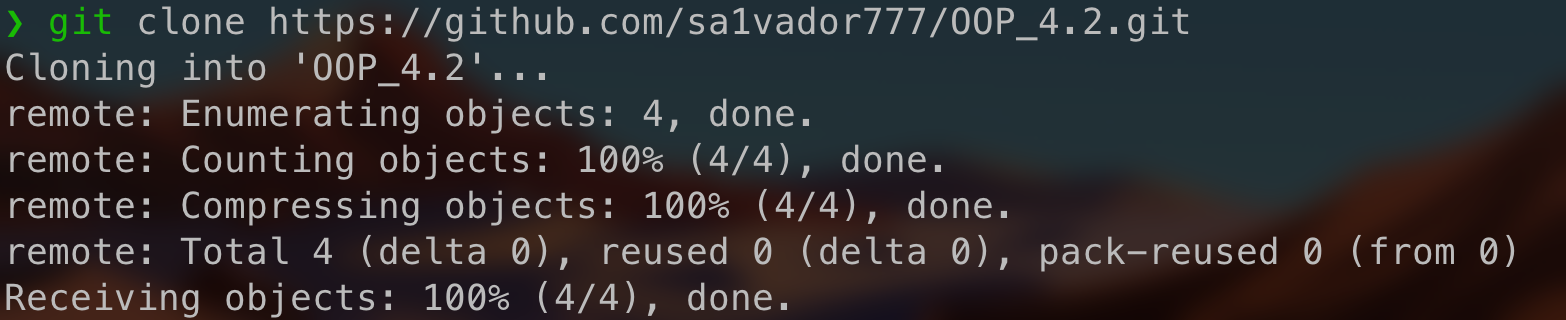


Рисунок 2 – Клонирование репозитория

1. Инициализировал модель ветвления git flow.

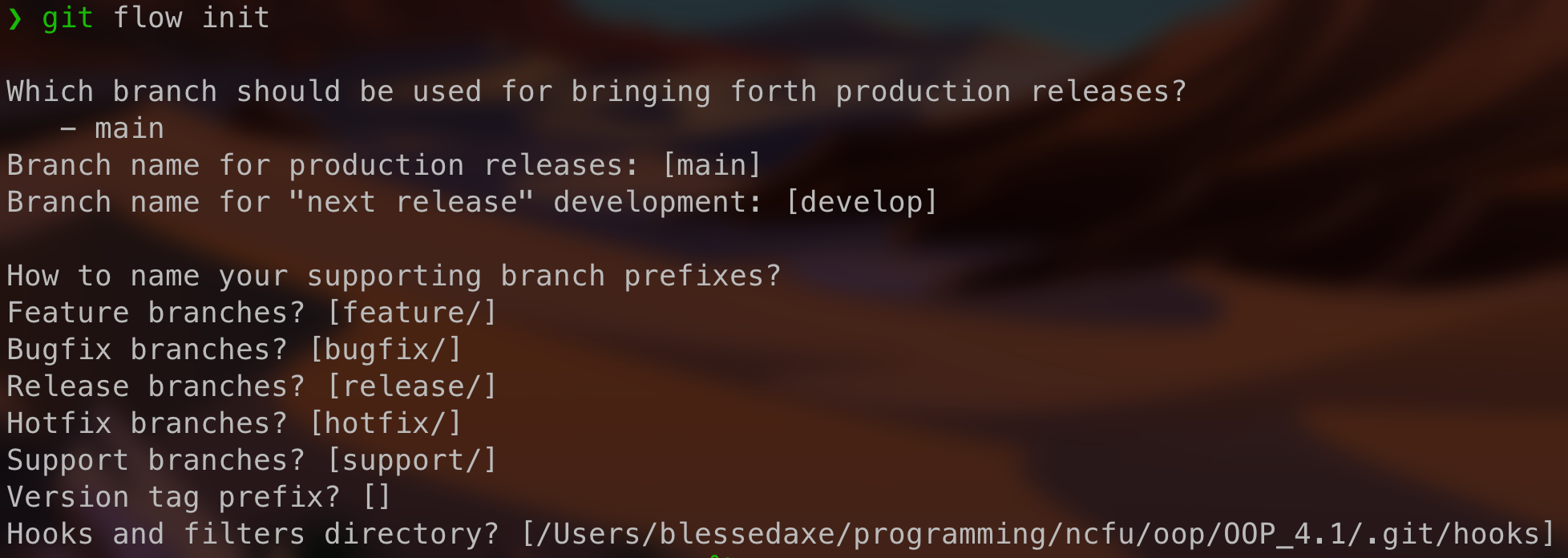


Рисунок 3 – Инициализация git-flow

1. Создал виртуальное окружение poetry

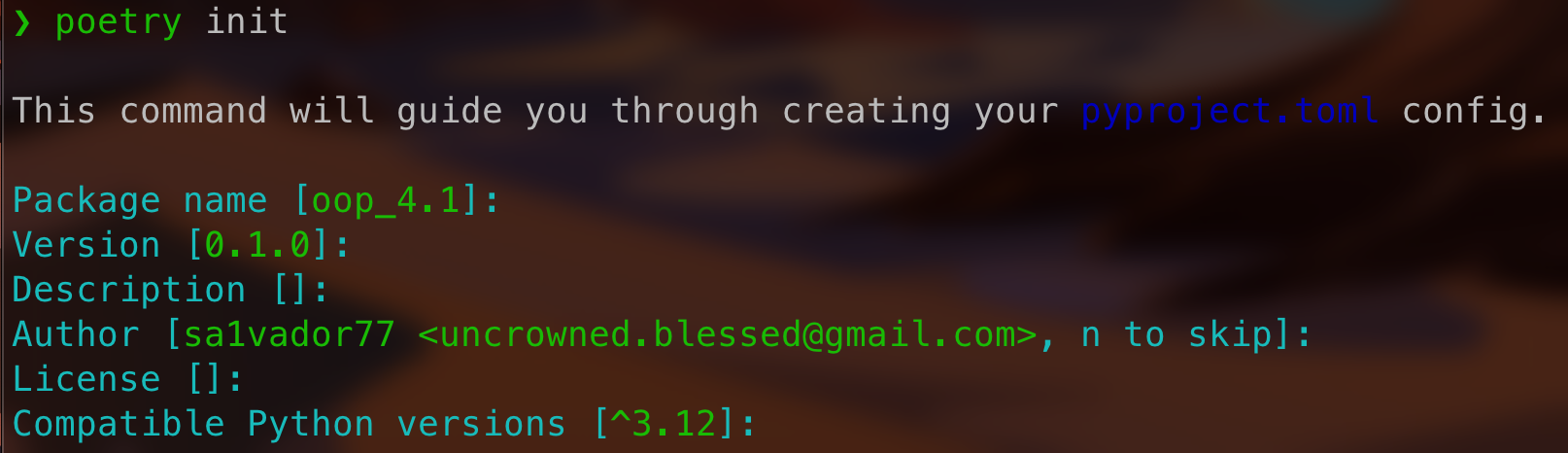


Рисунок 4 – Создание окружения poetry

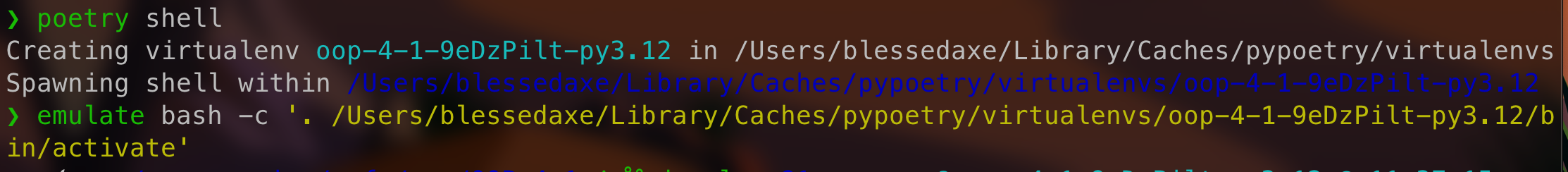


Рисунок 5 – Активация окружения

1. **Задание 1.**

Парой называется класс с двумя полями, которые обычно имею т имена **first** и **second**. Требуется реализовать тип данных с помощью такого класса. Во всех заданиях обязательно должны присутствовать:

* метод инициализации \_\_**init**\_\_; метод должен контролировать значения аргументов накорректность;
* ввод с клавиатуры **read**;
* вывод на экран **display**.

Реализовать внешнюю функцию с именем **make**\_**тип**() , где тип — тип реализуемой структуры. Функция должна получать в качестве аргументов значения для полей структуры и возвращать структуру требуемого типа. При передаче ошибочных параметров следует выводить сообщение и заканчивать работу. Номер варианта необходимо уточнить у преподавателя. В раздел программы, начинающийся после инструкции **if \_\_name\_\_ = '\_\_main\_\_'**: добавить код, демонстрирующий возможности разработанного класса.

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**.

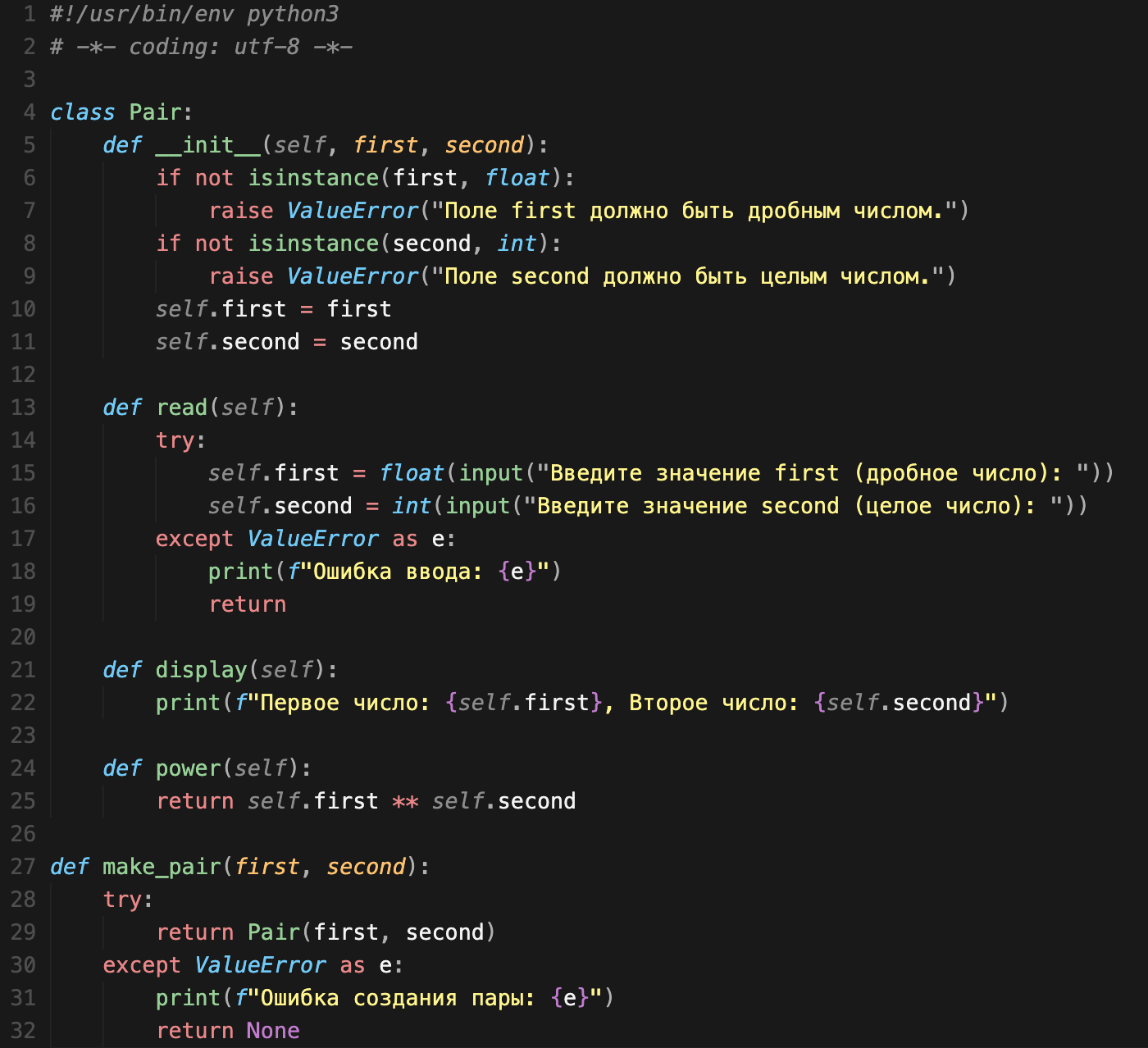


Рисунок 6 – Основной код индивидуального задания №1

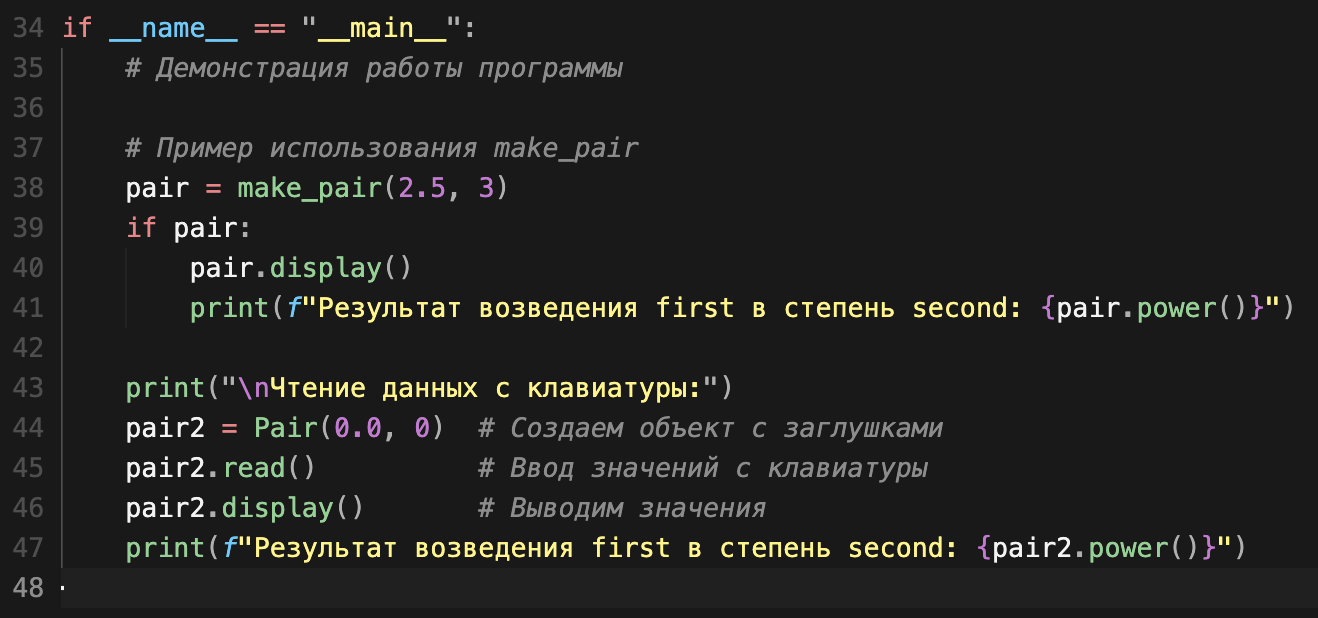


Рисунок 7 – Демонстрация работы программы

1. **Задание 2.**

Составить программу с использованием классов и объектов для решения задачи. Во всех заданиях, помимо указанных в задании операций, обязательно должны быть реализованы следующие методы:

* метод инициализации \_\_**init**\_\_;
* ввод с клавиатуры **read**;
* вывод на экран **display**.

Номер варианта необходимо уточнить у преподавателя. В раздел программы, начинающийся после инструкции **if** \_\_**name**\_\_ = '\_\_**main**\_\_': добавить код, демонстрирующий возможности разработанного класса.

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.



Рисунок 8 – Основной код индивидуального задания №2



Рисунок 9 – Демонстрация работы программы

**Ответы на контрольные вопросы:**

1. **Объявление** **класса** в Python осуществляется с использованием ключевого слова `class`, за которым следует имя класса и двоеточие. Пример: `class MyClass:`.

2. **Атрибуты** **класса** принадлежат всему классу и одинаковы для всех экземпляров, а **атрибуты** **экземпляра** уникальны для каждого объекта и задаются через `self` в методах.

3. **Назначение** **методов** **класса** — определять поведение объектов класса. Методы описывают действия, которые объекты могут выполнять.

4. **Метод** `\_\_**init**\_\_()` используется для инициализации экземпляра класса, задавая начальные значения атрибутов при создании объекта.

5. `**self**` — это ссылка на текущий экземпляр класса. Используется для доступа к атрибутам и методам объекта внутри класса.

6. **Атрибуты** **добавляются** в класс через метод `\_\_**init**\_\_()` или напрямую в классе (для атрибутов класса). Внутри методов используется `self`.

7. **Управление** **доступом** к методам и атрибутам осуществляется с помощью соглашений: имена, начинающиеся с `\_`, считаются защищёнными, а с `\_\_` — частными (но это лишь условность).

8. **Функция** `**isinstance**()` проверяет, является ли объект экземпляром указанного класса или его подклассов. Пример: `**isinstance**(**obj**, **MyClass**)` возвращает `**True**` или `**False**`.