

Лекция 1: Введение в ООП. Классы и объекты


Цель лекции: Познакомить студентов с парадигмой объектно-ориентированного программирования (ООП), объяснить ключевые понятия — класс, объект, атрибут, метод, конструктор. Научить создавать простые классы и работать с их экземплярами.

План лекции

1. Что такое ООП и зачем оно нужно?
 2. Основные принципы ООП (кратко).
 3. Класс как шаблон, объект как экземпляр.
 4. Синтаксис: `class`, `__init__`, `self`.
 5. Атрибуты и методы.
 6. Создание и использование объектов.
 7. Пример: класс `Dog`.
 8. Практическая демонстрация.
 9. Выводы и ответы на вопросы.
-

1. Что такое ООП?

Объектно-ориентированное программирование (ООП) — это парадигма программирования, в которой программа структурируется вокруг **объектов**, а не действий и логики.

 Вместо того чтобы писать функции, которые работают с данными, мы **объединяем данные и функции в один объект**.

Почему ООП?

- Упрощает моделирование реального мира.
 - Повышает читаемость и структурированность кода.
 - Облегчает повторное использование кода.
 - Упрощает поддержку и масштабирование проектов.
-

2. Основные принципы ООП (введение)

На этой лекции мы просто перечислим их — подробно разберём позже:

1. **Инкапсуляция** — сокрытие внутренней реализации, предоставление интерфейса.
 2. **Наследование** — возможность создавать новые классы на основе существующих.
 3. **Полиморфизм** — возможность объектов разных классов реагировать на одинаковые методы по-разному.
 4. **Абстракция** — выделение существенных характеристик объекта и игнорирование второстепенных.
-



3. Класс и объект

- **Класс** — это шаблон или чертёж. Описывает, какие **атрибуты** (данные) и **методы** (функции) будут у объектов этого класса.
- **Объект (экземпляр)** — это конкретная реализация класса в памяти программы.

Например:

Класс — `Car` (описывает, что у машины есть цвет, марка, метод “завести двигатель”).

Объект — `my_car = Car("Toyota", "red")` — конкретная машина.



4. Синтаксис: `class`, `__init__`, `self`

Объявление класса:

```
class ClassName:
    # тело класса
```

Конструктор `__init__`

- Метод, который вызывается **автоматически** при создании объекта.
- Используется для **инициализации атрибутов** объекта.
- Первый параметр — всегда `self` — ссылка на текущий объект.

```
class Dog:
    def __init__(self, name, age):
        self.name = name    # атрибут экземпляра
        self.age = age      # атрибут экземпляра
```

🔍 `self` — это не ключевое слово, но **обязательное соглашение**. Без него Python не поймёт, к какому объекту относятся атрибуты.



5. Атрибуты и методы

- **Атрибуты** — переменные, принадлежащие объекту (например, `name`, `age`).
- **Методы** — функции, определённые внутри класса, которые могут работать с атрибутами объекта.

```
class Dog:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def bark(self): # метод
        print(f"{self.name} says: Woof!")
```

🔗 Метод всегда принимает `self` первым параметром — даже если других параметров нет.



6. Создание и использование объектов

Создание объекта — это вызов класса как функции:

```
dog1 = Dog("Buddy", 5)
dog2 = Dog("Max", 3)
```

Теперь можно вызывать методы и обращаться к атрибутам:

```
print(dog1.name)      # Buddy
dog1.bark()           # Buddy says: Woof!
print(dog2.age)       # 3
dog2.bark()           # Max says: Woof!
```



7. Полный пример: класс Dog

```
class Dog:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def bark(self):
        print(f"{self.name} says: Woof!")

    def get_info(self):
        return f"Dog named {self.name}, age {self.age}"

# Создание объектов
dog1 = Dog("Buddy", 5)
dog2 = Dog("Max", 3)

# Использование
dog1.bark()           # Buddy says: Woof!
print(dog2.get_info()) # Dog named Max, age 3
```



8. Демонстрация на занятии

Продемонстрируйте вживую:

1. Создайте класс `Cat` с атрибутами `name`, `color`, методом `meow()`.
2. Создайте 2 объекта.

3. Вызовите методы, выведите атрибуты.

4. Измените атрибут объекта вручную: `cat1.color = "black"` — покажите, что это возможно.

✓ 9. Выводы

- ООП позволяет структурировать код вокруг объектов.
- Класс — шаблон, объект — его конкретный экземпляр.
- `__init__` — конструктор, инициализирует объект.
- `self` — обязательный параметр методов, ссылается на текущий объект.
- Атрибуты хранят данные, методы — поведение.

🔗 Главное — начать думать категориями объектов: кто они, что умеют, как взаимодействуют.

? Вопросы для обсуждения

1. Чем объект отличается от класса?
 2. Зачем нужен `self`?
 3. Можно ли создать объект без `__init__`?
 4. Что будет, если забыть `self` в методе?
 5. Можно ли добавить атрибут объекту после создания?
-

📝 Домашнее задание (необязательное)

Создайте класс `Phone` с атрибутами `brand`, `model`, `battery_level` и методом `call(contact)`, который выводит:

```
Calling {contact}... Battery: {battery_level}%
```

Создайте 2 объекта и "позвоните" с каждого.