


Программирование, лекция 10. Объектно-ориентированное программирование



Кафедра ИУ7 МГТУ им. Н. Э. Баумана,
2022 год



Знак _ в Python

1. Хранение значения последнего выражения в интерпретаторе
2. Игнорирование некоторых значений (при разыменовании кортежей и т. д.)
3. Задание специальных значений для имен переменных или функций (`_name`, `__name`, `__name__`)

Декораторы @

Декоратор — это функция, которая позволяет “обернуть” другую функцию для расширения её функциональности без непосредственного изменения её кода.

```
@decorator_function
```

```
def some_decorated_function():
```

```
    ...
```

Объектно-ориентированное программирование

Объектно-ориентированное программирование — методология, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определённого класса.

Класс — некоторый шаблон для создания объектов, обеспечивающий начальные значения состояния: инициализация полей-переменных и реализация поведения методов.

Объект — это экземпляр с собственным состоянием этих свойств.

Поле - некоторое «свойство», или атрибут, какого-то объекта (переменная, являющаяся его частью). Объявляется в классе.

Метод - функция объекта, которая имеет доступ к его состоянию (полям). Реализуется в классе.

Принципы ООП

- **абстракция.** Абстрагирование означает выделение значимой информации и исключение из рассмотрения незначимой;
- **инкапсуляция** — свойство системы, позволяющее объединить данные и методы, работающие с ними, в классе;
- **наследование** — свойство системы, позволяющее описать новый класс на основе уже существующего с частично или полностью заимствованной функциональностью;
- **полиморфизм** — свойство системы, позволяющее использовать объекты с одинаковым интерфейсом без информации о типе и внутренней структуре объекта.

Ключевое слово class

```
class A:
```

```
    # поля и методы класса A
```

```
class B(ParentClass1, ParentClass2, ...):
```

```
    # поля и методы класса B
```

```
class C:
```

```
    attr1 = 42
```

```
    attr2 = "Hello, World"
```

```
    def m1(self, x):
```

```
        # код метода
```

Статические методы, методы классов

Статический метод - метод, не имеющий доступа к состоянию (полям) объекта.

Декораторы `@staticmethod`, `@classmethod`

Конструктор, деструктор

Метод `__init__` вызывается при создании объекта

Метод `__del__` вызывается при удалении объекта

Работа с методами и полями

Первый обязательный параметр метода (неявный) - объект.

Для методов класса первый параметр - класс.

Для статических методов такого параметра нет.

Поля объекта определяются как поля `self`...

Наследование классов. Функция super()

```
class Rectangle:
    def __init__(self, length, width):
        self.length = length
        self.width = width

    def area(self):
        return self.length * self.width

class Square(Rectangle):
    def __init__(self, length):
        super().__init__(length, length)
```

Множественное наследование. Интерфейсы, “примеси” (миксины, mixin)

Множественное наследование - использование нескольких базовых классов. При этом поля и методы объединяются.

Проблема - сложные правила вычисления того, какие из одноимённых методов будут использоваться в классе-наследнике.

Миксин (mixin) - способ выделить часть логики в специальный, не полностью функциональный класс, для упрощения наследования.

Паттерны (шаблоны) проектирования

Паттерн проектирования - повторяемая архитектурная конструкция, представляющая собой решение регулярно возникающей проблемы проектирования. Выделяют:

- порождающие шаблоны проектирования (singleton - “одиночка”, внедрение зависимости, ...);
- структурные шаблоны проектирования (фасад, адаптер, декоратор, ...);
- поведенческие шаблоны проектирования.