InnoDom

Ещё про принципы ООП

Принципы ООП

Полиморфизм

это принцип ООП, позволяющий представлять одинаковые методы с **разной реализацией**. В Python полиморфизм реализовывается через наследование и переопределение методов в дочерних классах.

Полиморфизм

Создадим три класса, первый - это класс, в котором существует метод **make_dound(self)**, второй и третий класс переопределяют первый. Как видим, классы-наследники переопределяют родительский метод, вызов метода для каждого класса имеет одинаковое название однако разную реализацию.

```
class Animal:
    def __init_(self, name):
        self.name = name

    def make_sound(self):
        return "Animal"

class Dog(Animal):
    def make_sound(self):
        return 'Woof!'

class Cat(Animal):
    def make_sound(self):
        return 'Meow!'
```

```
def animal_sounds(animals):
    for animal in animals:
        print(animal.make_sound())


dog = Dog('Rex')
cat = Cat('Whiskers')
animals = [dog, cat]

animal_sounds(animals)
```

Абстракция

это принцип ООП, который позволяет реализовывать сложные иерархии классов, благодаря **абстрактным классам**. Класс называется абстрактным, когда в нём находится хотя бы один абстрактный метод.

Абстрактный метод - это метод, в котором отсутствует реализация, но он обязан быть реализован во всех наследниках.

Также благодаря абстракции существует множество паттернов программирования – это реализованные шаблоны, для описания сложных процессов простым кодом.

Абстракция

Допустим, существует класс *animal* и от него наследуется множество дочерних классов разных животных, для простоты реализации, чтобы не забыть какой-либо метод, можно использовать абстрактный, т.к. он существенно упростит реализацию дочерних классов.

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

    @abstractmethod
        def make_sound(self):
            pass

class Dog(Animal):
    def make_sound(self):
        return 'Woof!'

class Cat(Animal):
    def make_sound(self):
        return 'Meow!'

dog = Dog('Rex')
cat = Cat('Whiskers')

print(dog. make_sound())
print(cat.make_sound())
```

Абстракция

Мы создали класс-родитель, в нём находится 1 абстрактный метод, далее 2 класса-наследника, если мы их наследуем от абстрактного класса, то обязательно должны реализовать все абстрактные методы в нём.

Таким образом мы получаем класс-шаблон, на основе которого не забываем реализовать всю логику класса-наследника. Помним, что на основе абстрактного класса нельзя создать экземпляр