InnoDom

Классы и объекты

ООП (объектно-ориентированное программирование)

ещё одна концепция в разработке, где решение проблемы происходит через создание класса и вызовов объекта этого класса. Объект всегда имеет свои атрибуты и свои методы.

Когда используется ООП

- когда в решении можно выделить **сущность класса**;
- когда сложную логику в функциональности можно описать легко в ООП;
- в ООП существуют **шаблоны**, которыми можно описать сложную логику, благодаря чему не нужно ничего придумывать;
- **повторное** использование кода становится проще;
- классы **удобнее** читать и использовать другим разработчикам;
- для того, чтобы изменить поведение во всех объектах, достаточно изменить только класс, что позволяет **быстро** и **надежно** исправлять ошибки и поддерживать код.

Когда не используется ООП

Бывают ситуации, когда нужно сделать что-то **быстро** и **одноразово**. НЕ нужно тратить время на продумывание объектов и разнесение логики. Сделайте всё в одном файле. Это будет быстрее и проще для изменения. Удалить один файл легче, чем «выковыривать» потом классы по всему проекту.

Классы и объекты

В Python почти всё, что вы используете является **объектом какого-то класса** (даже типы данных в Python реализованы на ООП).

Классы – это **функции** (методы) и **переменные** (данные), которые описывают определённую, независимую, понятную **сущность** (объект). Иными словами класс – это шаблон, которому подчиняется объект.

Класс состоит из:

- конструктора
- статических и динамических переменных
- методов

Классы и объекты

Объект (экземпляр класса) – это сущность, предмет или явление (процесс), имеющий чётко выраженные границы, индивидуальность и поведение. Создаётся на **основе класса** и выполняет инструкции, описанные в классе.

Объекты класса можно создавать в **неограниченном** количестве.

Классы и объекты

Создание класса:

```
class Person
    pass
```

Вызов объекта этого класса:

```
people = Person()
```

Конструктор класса

Конструктор – это место, где инициализируются динамические атрибуты класса.
Методы – это функции, только они находятся в классе и используются для объектов.

`self` – ключевое слово, которое является **ссылкой на объек**т и используется для вызова методов и атрибутов определённого объекта в классе. Такая ссылка передаётся вместе со всеми аргументами, т.к. понимает, к какому объекту вы обращаетесь.

Конструктор класса

Создание конструктора:

```
class Person:

    def __init__(self):
        pass
```

Создание метода:

```
class Person:

    def __init__(self):
        pass

    def method1(self):
        pass
```

Свойства класса

Свойства (атрибуты класса) – это все переменные в классе, которые могут использовать объекты.

Свойства бывают:

- **Статические** – объявляются в классе и на протяжении всего класса являются одинаковым значением.
- **Динамические** – задают начальное состояние определённому объекту и меняются у каждого объекта при выполнении какого-либо действия с ним.

Все динамические свойства инициализируется в **конструкторе** класса и являются *начальным состоянием* объекта.

Свойства класса

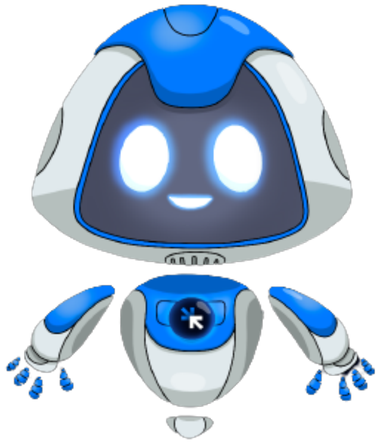
Создание свойств:

```
class Person:
    count = 0

    def __init__(self, name):
        self.name = name


people = Person('Katya')
print(people.name)
people.count = 20
print(people.count, Person.count)
```

Это *InnoBot* (**сущность**) ⟶



Выделим его **свойства**:

- имя
- возраст
- завод, на котором был изготовлен

Выделим его **методы**:

- посмотреть информацию
- переименовать
- изменить возраст

Описываем класс, который будет *создавать* наших роботов:

```
class InnoBot: # создаем класс InnoBot
    factory = 'InnoDom' #создаем 1 статическое свойство

    def __init__(self, name, age): # создаем конструктор класса и инициализируем 2 динамические переменные
        self.name = name
        self.age = age

    def info(self): # создаем метод, выводящий информацию об объекте
        print(f"InnoBot: \nИмя: {self.name}\nВозраст: {self.age}")

    def rename(self): # создаем метод, позволяющий переименовывать объект.
        self.name = input('Введите инноботу новое имя: ')
```