

Виды методов класса

Принципы ООП

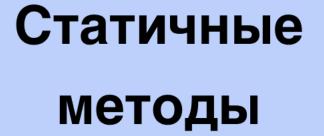


Абстрактные методы



Магические методы





Классовые методы

Классовые методы

Классовые методы – такие методы не работают с объектом класса, а работают с самим классом. Если представлять класс как чертёж дома, а объекты как реализованные дома, то классовые методы относятся к чертежу, например, из какой бумаги состоит чертеж либо каким цветом такой чертёж нарисован.

Для создания такого метода используется декоратор @classmethod. В такой метод обязательно передается аргумент cls, он, как и self (ссылка на объект), только хранит ссылку на класс.

Классовые методы

Вывод:

```
Dom.dom_params()
```

Статичные методы

Статичные методы – такие методы не работают ни с объектом класса, ни с классом. Они являются обычной функцией, только **реализованной в классе**.

Такие методы позволяют делать функции, которые являются независимыми, но являются именно методом класса и предполагается, что по смысловой нагрузке, тоже будут относится к классу.

Статичные методы

Для создания такого метода используется декоратор @staticmethod. @staticmethod – это что-то вроде обычной функции, определенной внутри класса, которая не имеет доступа к экземпляру, поэтому её можно вызывать без создания экземпляра класса.

Вывод:

Dom.info()

Абстрактные методы

Понятие абстрактного метода вытекает из понятия абстрактного класса. Если в классе существует хоть 1 абстрактный метод, то такой класс называется абстрактным.

Абстрактный метод – это метод, который определяет своё существование не вдаваясь в подробности реализации. Иными словами – это нереализованный метод.

```
from abc import abstractmethod

class Hello:
    @abstractmethod
    def hello(self):
        pass
```

Существование магических методов связано с упрощением разработки. Это стандартные методы, которые выполняют какие-либо инструкции быстро и просто. Магические они потому, что работают для всех классов, реализованных и стандартных в Python.

- __init__(self) конструктор класса. Используется при определении объектов.
- __del__(self) это метод класса, высвобождающий всю память при окончании работы программы. Вызывается автоматически сборщиком мусора, практически никогда не используется, за исключением, когда пользователя необходимо предупредить о незакрытых дескрипторах.
- __str__(self) позволяет переопределить вывод информации об объекте.

- __init__(self) конструктор класса. Используется при определении объектов.
- __del__(self) это метод класса, высвобождающий всю память при окончании работы программы. Вызывается автоматически сборщиком мусора, практически никогда не используется, за исключением, когда пользователя необходимо предупредить о незакрытых дескрипторах.
- __str__(self) позволяет переопределить вывод информации об объекте.

```
class Person:
    def __str__(self):
        return 'class Person'

class Person1:
    pass

print(Person(), '|', Person1())
```

• __len__(self) — метод позволяет выводить длину определенного поля вашего класса, при обращении к нему.

```
class Person:
    def __init__(self, peoples):
        self.peoples = peoples

    def __len__(self):
        return len(self.peoples)

people = Person(['Bob', 'Kevin', 'Sam', 'Rob'])
print(len(people))
```

- __iter__(self) создаёт итератор, при обращении к классу.
- __add__(self) позволяет проводить математические операции, при обращении к классу.

```
class Person:
    def __init_(self, age):
        self.age = age

    def __add__(self, other):
        return self.age + other.age

people = Person(10)
people2 = Person(22)
print(people+people2)
```