

Атрибуты класса

```
In [18]: class Planet:

        count = 0

        def __init__(self, name, population=None):
            self.name = name
            self.population = population or []
            Planet.count += 1

In [13]: earth = Planet("Earth")
mars = Planet("Mars")

print(Planet.count)

2

In [14]: mars.count

Out[14]: 2
```

Деструктор экземпляра класса

```
In [162]: class Human:

        def __del__(self):
            print("Goodbye!")

In [163]: human = Human()
# в данном случае деструктор отработает - но все же
# лучше создать метод и вызывать его явно
del human

Goodbye!
```

Словарь экземпляра и класса

```
In [1]: class Planet:

        """This class describes planets"""

        count = 1

        def __init__(self, name, population=None):
            self.name = name
            self.population = population or []

planet = Planet("Earth")

In [2]: planet.__dict__

Out[2]: {'name': 'Earth', 'population': []}

In [3]: planet.mass = 5.97e24

In [4]: planet.__dict__

Out[4]: {'mass': 5.97e+24, 'name': 'Earth', 'population': []}

In [5]: Planet.__dict__

Out[5]: mappingproxy({'__dict__': <attribute '__dict__' of 'Planet' objects>,
    '__doc__': 'This class describes planets',
    '__init__': <function __main__.Planet.__init__>,
    '__module__': '__main__',
    '__weakref__': <attribute '__weakref__' of 'Planet' objects>,
    'count': 1})

In [6]: Planet.__doc__

Out[6]: 'This class describes planets'

In [7]: planet.__doc__

Out[7]: 'This class describes planets'

In [8]: print(dir(planet))

['_class_', '__delattr__', '__dict__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattr__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__le__', '__lt__', '__module__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', '__weakref__', 'count', 'mass', 'name', 'population']

In [9]: planet.__class__

Out[9]: __main__.Planet
```

Конструктор экземпляра класса

```
In [10]: class Planet:

        def __new__(cls, *args, **kwargs):
            print("__new__ called")
            obj = super().__new__(cls)
            return obj

        def __init__(self, name):
            print("__init__ called")
            self.name = name

In [11]: earth = Planet("Earth")

__new__ called
__init__ called
```

То есть при вызове Planet("Earth") произошло примерно следующее:

```
planet = Planet.__new__(Planet, "Earth")

if isinstance(planet, Planet):
    Planet.__init__(planet, "Earth")
```

Мы с вами:

- узнали, что такое атрибут класса
- посмотрели на деструктор и конструктор экземпляра
- поговорили о поиске атрибутов в словаре экземпляра и класса

```
In [ ]: ### Атрибуты класса

class Planet:

    count = 0

    def __init__(self, name, population=None):
        self.name = name
        self.population = population or []
        Planet.count += 1

earth = Planet("Earth")
mars = Planet("Mars")

print(Planet.count)

mars.count

### Деструктор экземпляра класса

class Human:

    def __del__(self):
        print("Goodbye!")

human = Human()
# в данном случае деструктор отработает - но все же
# лучше создать метод и вызывать его явно
del human

### Словарь экземпляра и класса

class Planet:

    """This class describes planets"""

    count = 1

    def __init__(self, name, population=None):
        self.name = name
        self.population = population or []

planet = Planet("Earth")

planet.__dict__

planet.mass = 5.97e24

planet.__dict__

Planet.__dict__

Planet.__doc__

planet.__doc__

print(dir(planet))

planet.__class__

### Конструктор экземпляра класса

class Planet:

    def __new__(cls, *args, **kwargs):
        print("__new__ called")
        obj = super().__new__(cls)
        return obj

    def __init__(self, name):
        print("__init__ called")
        self.name = name

earth = Planet("Earth")

То есть при вызове Planet("Earth") произошло примерно следующее:

<pre>
planet = Planet.__new__(Planet, "Earth")

if isinstance(planet, Planet):
    Planet.__init__(planet, "Earth")
</pre>

Мы с вами:
* узнали, что такое атрибут класса
* посмотрели на деструктор и конструктор экземпляра
* поговорили о поиске атрибутов в словаре экземпляра и класса
```