

TODO

- `__slots__`
- type conversion
- Метаклассы
- `property`
- Поиск атрибута
- `weakref` + `__weakref__` + имитация `__del__`
- implementing protocols - container, contextmanager, generator
- приведение типов, как `int(x)`
- ABC
- `zope.interfaces`
- PEAK PyProtocols

Внутреннее устройство экземпляра

- Экземпляр состоит из `__dict__` - словаря атрибутов и `__class__` - ссылки на свой класс (тип)
- `a.b == a.__dict__['b'] _or_ a.__class__.__dict__['b'] _or_ ..`

Внутреннее устройство класса

- `class` это синтаксический сахар

```
1  def method(self, val):
2      "method_documentation"
3      self.some_field = val
4      return val ** 0.3
5
6  Simple = type('Simple',
7                (ParentClass,),
8                {'method': method,
9                 '__doc__': "class_documentation"})
10
```

Внутреннее устройство класса

- Блок внутри `class` выполняется и полученный `locals()` становится `__dict__` класса
- Класс может быть модифицирован в любое время и эта модификация тут-же отобразится на всех экземплярах

Множественное Наследование

```
1 class A(object):
2     def draw(self, pt):
3         some_action()
4
5     def clear(self):
6         pass
7
8 class B(A):
9     def draw(self, pt):
10        A.draw(self, pt)
11
12 class C(A):
13     def draw(self, pt):
14        A.draw(self, pt)
15
16     def clear(self):
17        pass
18
19 class D(B, C):
20     def draw(self, pt):
21        B.draw(self, pt)
22        C.draw(self, pt)
```

```
23
24 print A.__mro__
25 # (<class '__main__.A'>, <type 'object'>)
26
27 print B.__mro__
28 # (<class '__main__.B'>, <class '__main__.A'>, <type 'object'>)
29
30 print C.__mro__
31 # (<class '__main__.C'>, <class '__main__.A'>, <type 'object'>)
```

super

```
1 class A(object):
2     def draw(self, pt):
3         some_action()
4
5 class B(A):
6     def draw(self, pt):
7         super(B, self).draw(pt)
8
9 class C(A):
10    def draw(self, pt):
11        super(C, self).draw(pt)
12
13 class D(B, C):
14    def draw(self, pt):
15        super(D, self).draw(pt)
```

```
1 class B(object):
2     def draw(self, pt):
3         pass
4
5 class C(object):
6     def draw(self, pt):
7         pass
8
9 class D(B, C):
10     def draw(self, pt):
11         super(D, self).draw(pt) # ?????
```