Интроспекция

- Добраться можно до всего чего угодно (почти) и бОльшую часть этого можно поменять на лету
- DRY

Внутри объектов

```
class T(object):
    val = 1
    def __init__(self, data):
        self.data = data
        self.__x = 1

    def func(self):
        return self.data
t = T(2)
t.__dict__ == {'_T__x': 1, 'data': 1}
t.__class__ is T
```

Внутри объектов

```
x = t.func
x() == 2

y = T.func
y(x) == 2

x2 = y.__get__(t)
x2() == 2
```

Внутри методов

```
1 x = t.func
2 x() == 2
3 x.im_func, x.im_self, x.im_class
4
5 y = T.func
6 y(x) == 2
7
8 x2 = y.__get__(t)
9 x2() == 2
```

Внутри классов

```
T.__dict__ == {'__dict__': <attribute '__dict__' of 'T' objects>
               '__doc__': None,
               '__init__': <function __main__.__init__>,
               '__module__': '__main__',
               '__weakref__': <attribute '__weakref__' of 'T' ob
               'func': <function __main__.func>,
               'val': 1}
```

Внутри функций

```
1  def test(v1, v2=12):
2      return v1 + v2
3
4  test.func_code # code object
5  test.func_defaults == (12, )
6
7  test.x = 0
8  test.__dict__ == {'x': 0}
9
10 #func_closure, func_doc, func_globals, func_name
```

Внутри функций

```
1  import dis
2  import inspect
3
4  dis.dis(test)
5  #   2  0 LOAD_FAST        0 (v1)
6  #      3 LOAD_FAST        1 (v2)
7  #      6 BINARY_ADD
8  #      7 RETURN_VALUE
9
10 inspect.getargspec(test)
11 # ArgSpec(args=['v1', 'v2'], varargs=None, keywords=None, defaul
```

Внутри функций

```
1  import ast
2  print ast.dump(ast.parse("x = 1 + r"))
3
4  Module(body=[
5      Assign(
6          targets=[Name(id='x', ctx=Store())],
7          value=BinOp(
8                  left=Num(n=1),
9                  op=Add(),
10                 right=Name(id='r', ctx=Load())
11             )
12         )
13     ])
```

Пример

Нужно сгенерировать вагончик API к имеющемуся коду. CLI, HTTP, RPC, whatever.

Пример

```
1
2  class Command(objects):
3      class Params:
4          pass
5
6  def create_cli(cmd):
7      pass
8
9  @classmethod
10 def validate(self, **params):
11     pass
```