Inplace object/func update (e.g. patch for UT)

```
obj.__class__ = mock.__class__
obj.__dict__ = mock.__dict__


func1.func_code = func2.func_code
func1.func_defaults = func2.func_defaults
```

zip-zip

```
keys = dict.keys()
values = dict.keys()

items = zip(keys, values)
keys, values = zip(*items)
```

Runtime code inspection

```
1 import IPython
2 IPython.embed()
3 #TODO: any way to make this remote?
```

Hide data in object

```
obj.__dict__[0] = ....
```

Make uniq key

```
def get_uniq_key():
    return type('Key', (object,), {})
```

Passing method from other object

```
x = {}
class X(object):
    __getattr__ = x.__getitem__

attrs = {nama:getattr(obj, name) for name in dir(obj)}
x = type("_", (object,), attrs)()
```

List of all objs in current module

```
keys = global().keys()
for name in keys:
    pass
```

Faked module

```
class MyModule(object):
    pass
import sys
mmod = MyModule()
```

```
5 sys.modules['mod1'] = mmod
6
7 import mod1
8 assert mod1 is mmod
```

Faked module

```
1       x = X()
2       x = X.__class__.__call__(X)
3       x.test(12) == x.__class__.test.__get__(x)(12)
```