

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

```
1 obj.__class__ = mock.__class__
2 obj.__dict__ = mock.__dict__
3
4
5 func1.func_code = func2.func_code
6 func1.func_defaults = func2.func_defaults
```

zip-zip

```
1 keys = dict.keys()
2 values = dict.keys()
3
4 items = zip(keys, values)
5 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

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

Make uniq key

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

Passing method from other object

```
1 x = {}
2 class X(object):
3     __getattr__ = x.__getitem__
4
5 attrs = {name: getattr(obj, name) for name in dir(obj)}
6 x = type("_", (object,), attrs)()
```

List of all objs in current module

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

Faked module

```
1 class MyModule(object):
2     pass
3 import sys
4 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)
```