فرادرس
فراتر از یک کلاس درس
www.faradars.org

الگوهای طراحی در پایتون

مدرس:
پژمان اقبالی
دانشجوی کارشناسی ارشد مهندسی مکانیک
دانشگاه علم و صنعت ایران

The Singleton Design Patterns

DRY → Don't repeat yourself

- Critical
- Error
- Warning
- Info
- Debug

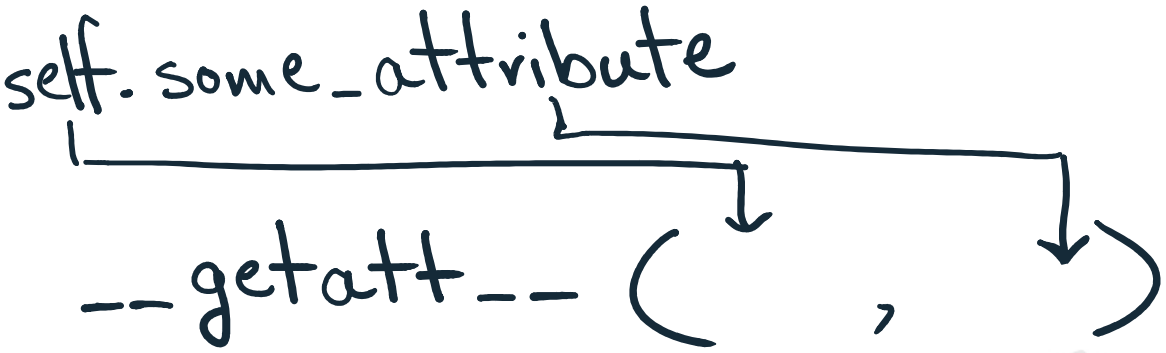
attributes and methods

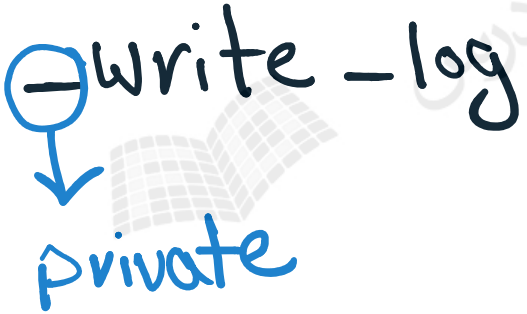
attributes
↓
داده‌های همراه یک شیء را در کلاس تعریف می‌کنند.

methods
↓
توابعی هستند که روی این داده‌ها عملیات انجام می‌دهند.

```
some_method(self)
critical(self)
```

```
self.some_method()
some_method()
```





```
hasattr ( obj , att )
hasattr (cls, 'instance')
```

```
class MyParentClass( ):
    def __init__(self):
        pass

class SubClass(MyParentClass):
    def __init__(self):
        MyParentClass.__init__(self)
```

Python 2

```
super(SubClass, self).__init__
```

Python 3

```
super()
```

Lazy instantiation

class method :

```
classmethod(function)
@classmethod
def func(cls, args...)
```

importing modules

↓

Singleton

GoF

Singleton → یک و تنها یک نمونه از یک class موجود باشد

same state → حالت → ⇒ Monostate design pattern

```
{'1': '2'}
+
{'x': '150', '1': '2'}
```

* پایتون از __dict__ برای ذخیره‌ی state (حالت) هر obj استفاده می‌کند.

روش

__new__ → obj instance

ایجاد کننده

__new__(2, 'cat', a1='dog', a2=8)

non-keyword

keyword

*args , **kwargs

الزامی — الزامی

func(—, —, —)

نوشتن args , kwargs بصورت توافقی است.

اما * , ** الزامی است.

*var **kwvar

Metaclasses

یک Metaclass از class برای class دیگر است.

```
Myclass
metaclass : MyKls
```

```
class A(object):
```

```
A = type(name, bases, dict)
```

- name : اسم کلاس
- bases: سوپرکلاس‌های کلاس A
- dict: attributes

metaclass : MetaKls

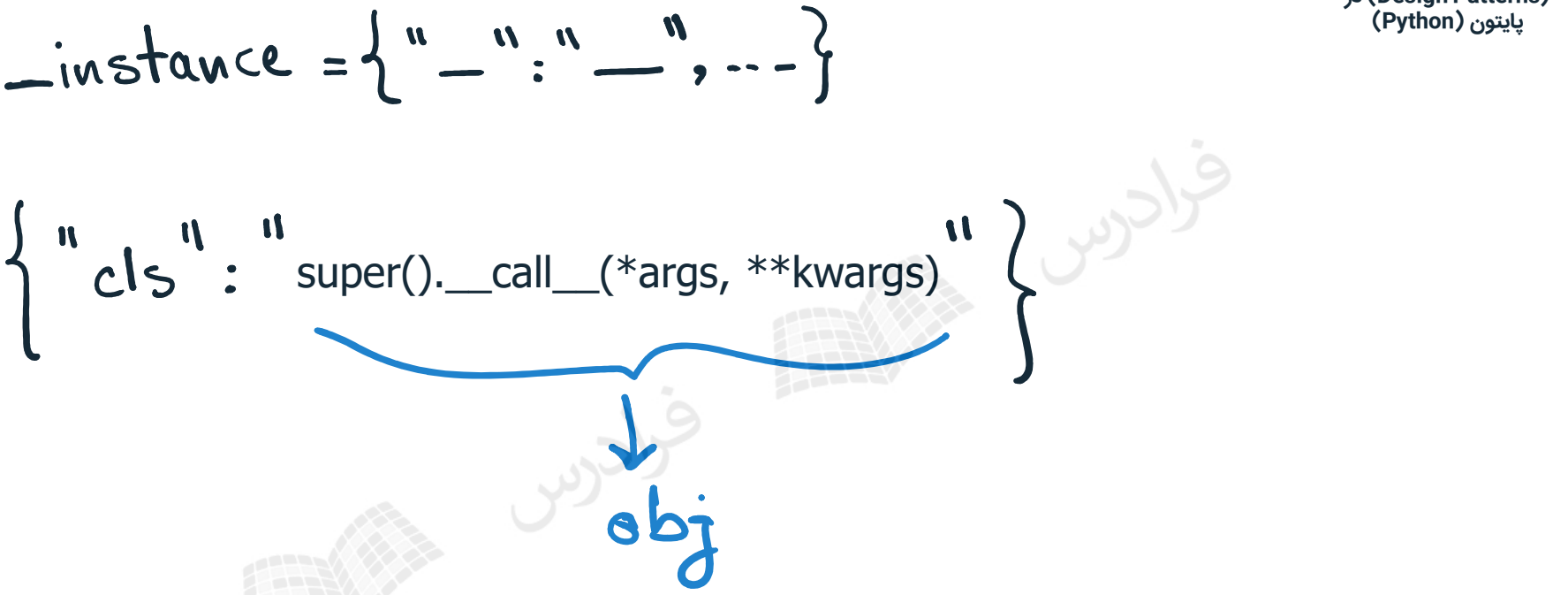
```
A = MetaKls(name , bases , dict)
```

type :

1) type (نوع) obj را مشخص می‌کند.
(نوع کلاس)

2) type (نوع) یک obj را برمی‌گرداند.

3) می‌تواند کلاس‌های جدید ایجاد کند.



تفاوت __init__ و __call__

__call__ => instance را فراخوانی نمود. (__call__ نمود)

n(1,2,3)

__init__ => مقداردهی اولیه instance نمود. (__init__ نمود)

v = A(1,2,3)

تفاوت __new__ و __init__

__new__ => وقتی بخواهیم ایجاد instance جدید را کنترل کنیم

اولین قدم در ساخت instance جدید و مسئول برگرداندن instance جدید می باشد.

__init__ => برای کنترل مقداردهی اولیه استفاده می شود و چیزی را برنمی گرداند

__new__
__call__
} => class method
(cls)

__init__ => Obj method
(self)

```
class C(object):
    pass
```

`c = C()` ⟶ `c = type(C).__call__(...)`

`c = type.__call__(...)`

`type(type(obj)).__call__(...)`

```
class C(object):
    pass
```

اینجا C یک class object و

type(C) → type.C

یک instance از type است.

```
class M(type):
    pass
```

اینجا M یک class object و

type(M) → type

چون superclass این M و type برابر.

در این حالت M ⇐ Metaclass obj

```
class D(metaclass=M):
    pass
```

اینجا D یک class obj

type(D) → M.D

M.D یک instance از M.D و M.D یک

instance از type میباشد نیز M یک subclass از

type است.

این اسلاید ها بر مبنای نکات مطرح شده در فرادرس «آموزش الگوهای طراحی (Design Patterns) در پایتون (Python)» تهیه شده است.

برای کسب اطلاعات بیشتر در مورد این آموزش به لینک زیر مراجعه نمایید.

faradars.org/fvpht96081