

The Many Faces of Python Descriptors

Utah Python Users Group, May 2015
Eric Snow



What Is a Descriptor?



```
__get__(self, obj, cls) <- object.__getattribute__
```

```
__set__(self, obj, value) <- object.__setattr__
```

```
__delete__(self, obj) <- object.__delattr__
```



“data descriptor”

implements both `__get__` and `__set__`

example: `property`

“non-data descriptor”

implements only `__get__`

example: `functions`



“data descriptor”

implements both `__get__` and `__set__`

example: `property`

“non-data descriptor”

implements only `__get__`

example: `functions`



“data descriptor”

implements both `__get__` and `__set__`

example: `property`

“non-data descriptor”

implements only `__get__`

example: `functions`



```
def __get__(self, obj, cls):  
    if obj is None:  
        ...  
    ...
```



```
def __get__(self, obj, cls):  
    if obj is None:  
        return self  
    return do_something_cool(obj)
```




```
def __get__(self, obj, cls):  
    if obj is None:  
        return self  
    return do_something_cool(obj)
```



```
def __get__(self, obj, cls):  
    if obj is None:  
        return self  
    return do_something_cool(obj)
```



Attribute Lookup



`spam.eggs -> getattr(spam, 'eggs')`

“dotted access”

“Attribute references” (language reference)

`object.__getattr__`

`PyObject_GenericGetAttr (Objects/object.c)`



`object.__getattr__`

1. handle “data descriptor” in `type(obj).__dict__`
2. try `obj.__dict__`
3. handle “non-data descriptor”
4. try `type(obj).__gettr__`
5. raise `AttributeError`



`type.__getattr__ (type_getattro)`

1. handle “data descriptor” in `type(cls).__dict__`
2. try `cls.__dict__` (handle descriptor)
3. handle “non-data descriptor”
4. raise `AttributeError`



“special” (“dunder”) method lookup

`_PyType_Lookup` (see `inspect._check_class`):

```
return type(obj).__dict__[name]
```

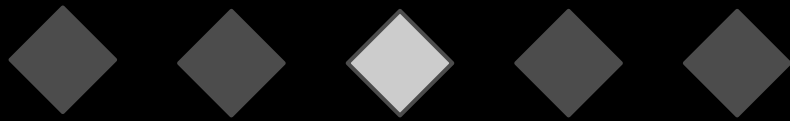


Handling descriptors:

```
return descr.__get__(obj, cls)
```




Descriptors You Use Every Day

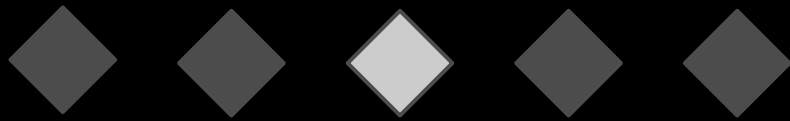


`property(fget, fset, fdel)`

`property.__get__` -> `property.fget`

`property.__set__` -> `property.fset`

`property.__delete__` -> `property.fdel`



```
def __get__(self, obj, cls):  
    if obj is None:  
        return self  
    return self.fget(obj)
```



```
def spam(): ...
```

```
spam.__get__ -> method
```

```
(no __set__ or __delete__)
```

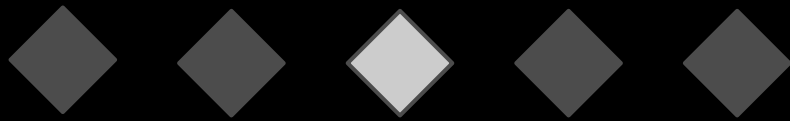


Python 3

```
def __get__(self, obj, cls):  
    if obj is None:  
        return self  
    return types.MethodType(self, obj)
```



```
>>> class Spam:
...     def eggs(self): pass
...
>>> Spam.eggs
<function Spam.eggs at ...>
>>> Spam().eggs
<bound method Spam.eggs of <Spam object at ...>>
```



Python 2

```
def __get__(self, obj, cls):  
    if obj is None:  
        return types.UnboundMethodType(self, cls)  
    return types.MethodType(self, obj)
```



```
>>> class Spam:
...     def eggs(self): pass
...
>>> Spam.eggs
<unbound method Spam.eggs>
>>> Spam().eggs
<bound method Spam.eggs of <Spam object at ...>>
```




Composing New Descriptors



classmethod
staticmethod
classonly



Bonus: Descriptor Examples



lazy attributes

reverse name binding

placeholders

Summary

- Descriptors invoked by object.__getattrute__
- __get__ called for class AND object access
- 6 kinds of descriptor

Questions

Eric Snow

ericsnowcurrently@gmail.com

<http://goo.gl/UGjKPL>

<https://bitbucket.org/ericsnowcurrently/presentations/src/default/utpy-may2015>