# 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.__getattribute__

PyObject_GenericGetAttr (Objects/object.c)

object.__getattribute__

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

type.__getattribute__ (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

- data vs non-data?
- supports wrapping other descriptors?
- triggered on class vs. on instance?
- cooperative (external state) vs. isolated?
- static (ignore obj, cls)?
- resolve on cls? on obj?

rawattr (staticmethod)

classattr

classonly

nondata

classresolved (classmethod)

classunresolved

# Wrapper Descriptors

| | cls | obj | data |
|---|---|---|---|
| rawattr | unresolved | unresolved | no |
| classattr | class-resolved | class-resolved | no |
| classonly | class-resolved | AttributeError | no |
| nondata | resolved | resolved | no |
| classresolved | class-as-object | class-as-object | no |
| classunresolved | descriptor | resolved | no |

```
class rawattr:
    def __init__(self, w):
        self.w = w
    def __get__(self, obj, cls):
        return self.w
```

```
class Spam:
    @rawattr
    @property
    def eggs(self):
        return None

assert isinstance(Spam.eggs, property)
spam = Spam()
assert isinstance(spam.eggs, property)
```

```
class classattr:
    def __init__(self, w):
        self.w = w
    def __get__(self, obj, cls):
        if hasattr(self.w, '__get__'):
            return self.w.__get__(None, cls)
        else:
            return self.w
```

```
class Spam:
    @classattr
    @property
    def eggs(self):
        return None

assert isinstance(Spam.eggs, property)
spam = Spam()
assert isinstance(spam.eggs, property)
```

```
class classonly:
    def __init__(self, w):
        if isinstance(w, functype):
            w = classmethod(w)
        self.w = w
    def __get__(self, obj, cls):
        if obj is None:
           return self.w.__get__(None, cls)
        else:
            raise AttributeError
```

```
class Spam:
    @classonly
    def eggs(cls):
        return cls.__name__

assert Spam.eggs == 'Spam'
Spam().eggs  # AttributeError
```

```
class nondata:
    def __init__(self, w):
        self.w = w
    def __get__(self, obj, cls):
        if hasattr(self.w, '__get__'):
            return self.w.__get__(obj, cls)
        else:
            return self.w
```

```
class Spam:
    @nondata
    @property
    def eggs(self):
        return 'spamspamspam'
assert isinstance(Spam.eggs, property)
spam = Spam()
assert spam.eggs == 'spamspamspam'
spam.eggs = 42
assert spam.eggs == 42
```

```
class classresolved:
    def __init__(self, w):
        self.w = w
    def __get__(self, obj, cls):
        if hasattr(self.w, '__get__'):
            return self.w.__get__(cls, meta)
        else:
            return self.w
```

```
class Spam:
    @classresolved
    @property
    def eggs(cls):
        return cls.__name__

assert Spam.eggs == 'Spam'
assert Spam().eggs == 'Spam'
```

```
class classunresolved:
    def __init__(self, w):
        self.w = w
    def __get__(self, obj, cls):
        if obj is None:
            return self.w
        else:
            return self.w.__get__(obj, cls)
```

```
class Spam:
    @classunresolved
    @classmethod
    def eggs(cls):
        return cls.__name__

assert isinstance(Spam.eggs, classmethod)
assert Spam().eggs == 'Spam'
```

# Bonus: Descriptor Examples

lazy attributes

reverse name binding

placeholders

# Summary

- Descriptors invoked by object.__getattribute__
- __get__ called for class AND object access
- many kinds of descriptor

# Questions

Eric Snow

ericsnowcurrently@gmail.com

http://goo.gl/UGjKPL

(https://bitbucket.org/ericsnowcurrently/presentations/src/default/utpy-may2015)