Introduction to Python
More OO

Christopher Barker

UW Continuing Education

November 19, 2013

Table of Contents

Review of Previous Class

- .
- .
- .

Lightning Talks

Lightning talks today:

Homework review

Homework Questions?

My Solution

special methods

Python's Duck typing:

Defining special (or magic) methods in your classes is how you make your class act like standard classes

special methods

We've seen at least one:

`__init__`

it's all in the double underscores...

Pronounced "dunder" (or "under-under")

try: `dir(2)` or `dir(list)`

Emulating Numeric types

```
object.__add__(self, other)
object.__sub__(self, other)
object.__mul__(self, other)
object.__floordiv__(self, other)
object.__mod__(self, other)
object.__divmod__(self, other)
object.__pow__(self, other[, modulo])
object.__lshift__(self, other)
object.__rshift__(self, other)
object.__and__(self, other)
object.__xor__(self, other)
object.__or__(self, other)
```

special methods

Emulating container types:

```
object.__len__(self)
object.__getitem__(self, key)
object.__setitem__(self, key, value)
object.__delitem__(self, key)
object.__iter__(self)
object.__reversed__(self)
object.__contains__(self, item)
object.__getslice__(self, i, j)
object.__setslice__(self, i, j, sequence)
object.__delslice__(self, i, j)
```

special methods

Example – to define addition:

```
def __add__(self, v):
    """
    redefine + as element-wise vector sum
    """
    assert len(self) == len(v)
    return vector([x1 + x2 for x1, x2 in zip(self, v)])
```

(from a nice complete example in `code/vector.py`)

special methods

You get the idea...

You only need to define the ones that are going to get used

But you probably want to define at least these:

`object.__str__`: Called by the str() built-in function and by the print statement to compute the informal string representation of an object.

`object.__repr__`: Called by the repr() built-in function and by string conversions (reverse quotes) to compute the official string representation of an object.

special methods

When you want your class to act like a "standard" class in some way:

Look up the magic methods you need and define them

`http://docs.python.org/reference/datamodel.html#special-method-names`

`http://www.rafekettler.com/magicmethods.html`

LAB

Write a “Circle” class:

A Circle has a radius and can compute its area:

```
In [2]: c = Circle(3)
In [3]: c.radius
Out[3]: 3
In [4]: c.get_area()
Out[4]: 28.274333882308138
In [5]: print c
Circle Object with radius: 3.000000
```

Write an `__add__` method so you can add two circles

Have `__str__` and `__repr__` methods

Extra credit: also compare them... (`c1 > c2`, etc)

`code/circle.py` and `code/test_circle.py`

LAB

Some lab excercises

Lightning Talk

Lightning Talks:

person 1

person 2

Wrap Up

Thinking OO in Python:

Think about what makes sense for your code:

- Code re-use
- Clean APIs
- ...

Don't be a slave to what OO is *supposed* to look like.

Let OO work for you, not *create* work for you

Wrap Up

OO in Python:

The Art of Subclassing: Raymond Hettinger

`http://pyvideo.org/video/879/the-art-of-subclassing`

"classes are for code re-use – not creating taxonomies"

Stop Writing Classes: Jack Diederich

`http://pyvideo.org/video/880/stop-writing-classes`

"If your class has only two methods – and one of them is `__init__` – you don't need a class "

Homework

Finish the labs.
You should have a good start on your project by the end of this week

Homework

Recommended Reading:

- some stuff

Do:

- Some things