

Introduction to Python Lambda, and Functional Programming

Christopher Barker

UW Continuing Education

November 5, 2013

Table of Contents

- 1 Review/Questions
- 2 Lambda
- 3 Functional Programming
- 4 Object Oriented Programming
- 5 Python Classes

Review of Previous Class

- Unicode?
- Keyword arguments?
- Comprehensions?
- Unit testing?

Lightning Talks

Lightning talks today:

Lawrence Chan

Kimberly Colwell

Maria Petrova

Homework review

Homework Questions?

lambda

“Anonymous” functions

```
In [171]: f = lambda x, y: x+y
```

```
In [172]: f(2,3)
```

```
Out[172]: 5
```

Can only be an expression – not a statement

lambda

Called “Anonymous”: it doesn't need a name.
It's a python object, it can be stored in a list or other container

```
In [7]: l = [lambda x, y: x+y]
```

```
In [8]: type(l[0])
```

```
Out[8]: function
```

And you can call it:

```
In [9]: l[0](3,4)
```

```
Out[9]: 7
```

functions as first class objects

You can do that with “regular” functions too:

```
In [12]: def fun(x,y):  
        ....:     return x+y  
        ....:
```

```
In [13]: l = [fun]
```

```
In [14]: type(l[0])  
Out[14]: function
```

```
In [15]: l[0](3,4)  
Out[15]: 7
```


map

map “maps” a function onto a sequence of objects –
It applies the function to each item in the list,
returning another list

```
In [23]: l = [2, 5, 7, 12, 6, 4]
```

```
In [24]: def fun(x):  
         return x*2 + 10
```

```
In [25]: map(fun, l)
```

```
Out[25]: [14, 20, 24, 34, 22, 18]
```

But if you only need that function once:

```
In [26]: map(lambda x: x*2 + 10, l)
```

```
Out[26]: [14, 20, 24, 34, 22, 18]
```

filter

`filter` “filters” a sequence of objects with a boolean function – It keeps only those for which the function is `True`

To get only the even numbers

```
In [27]: l = [2, 5, 7, 12, 6, 4]
```

```
In [28]: filter(lambda x: not x%2, l)
```

```
Out[28]: [2, 12, 6, 4]
```

reduce

reduce “reduces” a sequence of objects to a single object with a function that combines two arguments

To get the sum:

```
In [30]: l = [2, 5, 7, 12, 6, 4]
```

```
In [31]: reduce(lambda x,y: x+y, l)
```

```
Out[31]: 36
```

To get the product:

```
In [32]: reduce(lambda x,y: x*y, l)
```

```
Out[32]: 20160
```

comprehensions

Couldn't you do all this with comprehensions?

Yes:

```
In [33]: [x+2 + 10 for x in l]
```

```
Out[33]: [14, 17, 19, 24, 18, 16]
```

```
In [34]: [x for x in l if not x%2]
```

```
Out[34]: [2, 12, 6, 4]
```

Except Reduce

But Guido thinks almost all uses of reduce are really `sum()`

functional programming

Comprehensions and map, filter, reduce are all “functional programming” approaches

map, filter and reduce pre-date comprehensions in Python’s history

Some people like that syntax better

And “map-reduce” is a big concept these days for parallel processing of “Big Data” in NoSQL databases.

(Hadoop, MongoDB, etc.)

lambda

Can also use keyword arguments

```
In [186]: l = []  
In [187]: for i in range(3):  
    l.append(lambda x, e=i: x**e)  
    .....:  
In [189]: for f in l:  
    print f(3)  
1  
3  
9
```

Note when the keyword argument is evaluated: this turns out to be handy

LAB

- Write a function that returns a list of n functions, such that each one, when called, will return the input value, incremented by an increasing number.
- you should use a for loop, lambda, and a keyword argument

`code/lambda/lambda_keyword.html(rst)`

`code/lambda/lambda_keyword.py`

`code/lambda/test_lambda_keyword.py`

Lightning Talks

Lightning Talks:

Lawrence Chan

Kimberly Colwell

Object Oriented Programming

More about Python implementation than OO design/strengths/weaknesses

One reason for this:
Folks can't even agree on what OO “really” means

The Quarks of Object-Oriented Development - Deborah J. Armstrong:
<http://agp.hx0.ru/oop/quarks.pdf>

Object Oriented Programming

Is Python a “True” Object-Oriented Language?

(Doesn't support full encapsulation, doesn't require objects, etc...)

Object Oriented Programming

I don't Care!

Good software design is about code re-use, clean separation of concerns, refactorability, testability, etc...

OO can help with all that, but:

- It doesn't guarantee it
- It can get in the way

Object Oriented Programming

Python is a Dynamic Language

That clashes with “pure” OO

Think in terms of what makes sense for your project
– not any one paradigm of software design.

Object Oriented Programming

OO for this class:

“Objects can be thought of as wrapping their data within a set of functions designed to ensure that the data are used appropriately, and to assist in that use”

http://en.wikipedia.org/wiki/Object-oriented_programming

Object Oriented Programming

Even simpler:

Objects are data and the functions that act on them in one place.

In Python: just another namespace.

Object Oriented Programming

The OO buzzwords:

- data abstraction
- encapsulation
- modularity
- polymorphism
- inheritance

Object Oriented Programming

You can do OO in C

(see the GTK+ project)

“OO languages” give you some handy tools to make it easier (and safer).

- polymorphism (duck typing gives you this anyway)
- inheritance

Object Oriented Programming

OO is the dominant model for the past couple decades

You will need to use it:

- It's a good idea for a lot of problems
- You'll need to work with OO packages

Object Oriented Programming

Some definitions

- class** A category of objects: particular data and behavior:
A “circle” (same as a type in python)
- instance** A particular object of a class: a specific circle
- object** The general case of a instance – really any value
(in Python anyway)
- attribute** Something that belongs to an object (or class) –
generally thought of as a variable, or single object, as
opposed to a ...
- method** A function that belongs to a class

Python Classes

The class statement

class creates a new type object:

```
In [4]: class C(object):  
        pass
```

```
...:
```

```
In [5]: type(C)
```

```
Out[5]: type
```

It is created when the statement is run – much like def

(note on “new style” classes)

Python Classes

Note about the book (TP):

Chapters 15 and 16 use a style that generally isn't recommended:

```
In [6]: class Point(object):  
...:     pass  
In [7]: p = Point()  
In [8]: p.x = 4  
In [9]: p.y = 2
```

Python is Dynamic – you can do this, but you generally want more structure, defaults, etc.

(it used to be a quick and dirty "struct"
– but use a named tuple now)

Python Classes

About the simplest class:

```
>>> class Point(object):  
...     x = 1  
...     y = 2  
>>> Point  
<class __main__.Point at 0x2bf928>  
>>> Point.x  
1  
>>> p = Point()  
>>> p  
<__main__.Point instance at 0x2de918>  
>>> p.x  
1
```

Python Classes

Basic Structure of a real class:

```
class Point(object):  
    # everything defined in here is in the class namespace  
    def __init__(self, x, y):  
        self.x = x  
        self.y = y  
## create an instance of that class  
p = Point(3,4)  
  
## access the attributes  
print "p.x is:", p.x  
print "p.y is:", p.y  
  
see: code/simple_class
```

Python Classes

The Initializer

The `__init__` special method is called when a new instance of a class is created.

You can use it to do any set-up you need

```
class Point(object):  
    def __init__(self, x, y):  
        self.x = x  
        self.y = y
```

It gets the arguments passed to the class constructor

Python Classes

self

The instance of the class is passed as the first parameter for every method.

“self” is only a convention – but you DO want to use it.

```
class Point(object):  
    def a_function(self, x, y):  
    ...
```

Does this look familiar from C-style procedural programming?

Python Classes

```
class Point(object):  
    def __init__(self, x, y):  
        self.x = x  
        self.y = y
```

Anything assigned to a `self.` attribute is kept in the instance name space

That's where all the instance-specific data is.

Python Classes

```
class Point(object):  
    size = 4  
    color= "red"  
    def __init__(self, x, y):  
        self.x = x  
        self.y = y
```

Anything assigned in the class scope is a class attribute – every instance of the class shares the same one.

Python Classes

```
class Point(object):  
    size = 4  
    color= "red"  
...  
    def get_color():  
        return self.color  
  
>>> p3.get_color()  
'red'
```

class attributes are accessed with self also..

Python Classes

Typical methods

```
class Circle(object):  
    color = "red"  
    def __init__(self, diameter):  
        self.diameter = diameter  
  
    def grow(self, factor=2):  
        self.diameter = self.diameter * factor
```

methods take some parameters, manipulate the attributes in self

Python Classes

Gotcha!

```
...  
    def grow(self, factor=2):  
        self.diameter = self.diameter * factor  
...
```

```
In [205]: C = Circle(5)
```

```
In [206]: C.grow(2,3)
```

TypeError: grow() takes at most 2 arguments (3 given)

Huh???? I only gave 2

“self” is implicitly passed in...

LAB

We had such a good time last class – we'll do something similar

The goal is to build a set of classes that render an html page: `sample_html.html`

We'll start with a single class, then add some sub-classes to specialize the behavior

More details in `week-06/LAB_instructions.txt`

LAB

Step 1:

- Create an "Element" class for rendering an html element (xml element).
- It should have class attributes for the tag name and the indentation
- the constructor signature should look like:
`Element(content=None)` where content is a string
- It should have an "append" method that can add another string to the content
- It should have a `render(file_out, ind = "")` method that renders the tag and the strings in the content.
`file_out` could be any file-like object.
`ind` is a string with enough spaces to indent properly.

Lightning Talk

Lightning Talk:

Brett

Homework

Recommended Reading:

- some stuff

Do:

- Some things