Review/Questions
Lambda
Functional Programming
Object Oriented Programming
Python Classes
Subclassing/Inheritance

Introduction to Python
Lambda, and Functional Programming

Christopher Barker

UW Continuing Education

November 5, 2013

Review/Questions
Lambda
Functional Programming
Object Oriented Programming
Python Classes
Subclassing/Inheritance

Table of Contents

Review/Questions
Lambda
Functional Programming
Object Oriented Programming
Python Classes
Subclassing/Inheritance

Review of Previous Class

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

Review/Questions
Lambda
Functional Programming
Object Oriented Programming
Python Classes
Subclassing/Inheritance

Lightning Talks

Lightning talks today:

Lawrence Chan

Kimberly Colwell

Maria Petrova

Review/Questions
Lambda
Functional Programming
Object Oriented Programming
Python Classes
Subclassing/Inheritance

Homework review

Homework Questions?

Review/Questions
Lambda
Functional Programming
Object Oriented Programming
Python Classes
Subclassing/Inheritance

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

Review/Questions
Lambda
Functional Programming
Object Oriented Programming
Python Classes
Subclassing/Inheritance

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
```

Review/Questions
Lambda
Functional Programming
Object Oriented Programming
Python Classes
Subclassing/Inheritance

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
```

Review/Questions
Lambda
Functional Programming
Object Oriented Programming
Python Classes
Subclassing/Inheritance

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]
```

Review/Questions
Lambda
Functional Programming
Object Oriented Programming
Python Classes
Subclassing/Inheritance

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]
```

Review/Questions
Lambda
Functional Programming
Object Oriented Programming
Python Classes
Subclassing/Inheritance

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
```

Review/Questions
Lambda
Functional Programming
Object Oriented Programming
Python Classes
Subclassing/Inheritance

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()

Review/Questions
Lambda
Functional Programming
Object Oriented Programming
Python Classes
Subclassing/Inheritance

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.)

Review/Questions
Lambda
Functional Programming
Object Oriented Programming
Python Classes
Subclassing/Inheritance

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

Review/Questions
Lambda
Functional Programming
Object Oriented Programming
Python Classes
Subclassing/Inheritance

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.

- 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
```

Review/Questions
Lambda
Functional Programming
Object Oriented Programming
Python Classes
Subclassing/Inheritance

Lightning Talks

Lightning Talks:
ł

Lawrence Chan

Kimberly Colwell

Review/Questions
Lambda
Functional Programming
Object Oriented Programming
Python Classes
Subclassing/Inheritance

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

Review/Questions
Lambda
Functional Programming
Object Oriented Programming
Python Classes
Subclassing/Inheritance

Object Oriented Programming

Is Python a "True" Object-Oriented
Language?

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

Review/Questions
Lambda
Functional Programming
Object Oriented Programming
Python Classes
Subclassing/Inheritance

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

Review/Questions
Lambda
Functional Programming
Object Oriented Programming
Python Classes
Subclassing/Inheritance

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.

Review/Questions
Lambda
Functional Programming
Object Oriented Programming
Python Classes
Subclassing/Inheritance

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

Review/Questions
Lambda
Functional Programming
Object Oriented Programming
Python Classes
Subclassing/Inheritance

Object Oriented Programming

Even simpler:

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

In Python: just another namespace.

Review/Questions
Lambda
Functional Programming
Object Oriented Programming
Python Classes
Subclassing/Inheritance

Object Oriented Programming

The OO buzzwords:

- data abstraction

- encapsulation

- modularity

- polymorphism

- inheritance

Review/Questions
Lambda
Functional Programming
Object Oriented Programming
Python Classes
Subclassing/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

Review/Questions
Lambda
Functional Programming
Object Oriented Programming
Python Classes
Subclassing/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

Review/Questions
Lambda
Functional Programming
Object Oriented Programming
Python Classes
Subclassing/Inheritance

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

Review/Questions
Lambda
Functional Programming
Object Oriented Programming
Python Classes
Subclassing/Inheritance

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)

Review/Questions
Lambda
Functional Programming
Object Oriented Programming
Python Classes
Subclassing/Inheritance

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)

Review/Questions
Lambda
Functional Programming
Object Oriented Programming
Python Classes
Subclassing/Inheritance

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
```

Review/Questions
Lambda
Functional Programming
Object Oriented Programming
Python Classes
Subclassing/Inheritance

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
```

Review/Questions
Lambda
Functional Programming
Object Oriented Programming
Python Classes
Subclassing/Inheritance

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

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

It gets the arguments passed to the class constructor

Review/Questions
Lambda
Functional Programming
Object Oriented Programming
Python Classes
Subclassing/Inheritance

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?

Review/Questions
Lambda
Functional Programming
Object Oriented Programming
Python Classes
Subclassing/Inheritance

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.

Review/Questions
Lambda
Functional Programming
Object Oriented Programming
Python Classes
Subclassing/Inheritance

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.

Review/Questions
Lambda
Functional Programming
Object Oriented Programming
Python Classes
Subclassing/Inheritance

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..

Review/Questions
Lambda
Functional Programming
Object Oriented Programming
Python Classes
Subclassing/Inheritance

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

Review/Questions
Lambda
Functional Programming
Object Oriented Programming
Python Classes
Subclassing/Inheritance

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...)

Review/Questions
Lambda
Functional Programming
Object Oriented Programming
Python Classes
Subclassing/Inheritance

LAB

Let's say you need to render some html...

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_instuctions.rst(html)`

Review/Questions
Lambda
Functional Programming
Object Oriented Programming
Python Classes
Subclassing/Inheritance

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.

Review/Questions
Lambda
Functional Programming
Object Oriented Programming
Python Classes
Subclassing/Inheritance

Lightning Talk

Lightning Talk:

Maria Petrova

Review/Questions
Lambda
Functional Programming
Object Oriented Programming
Python Classes
Subclassing/Inheritance

Inheritance

In object-oriented programming (OOP), inheritance is a way to reuse code of existing objects, or to establish a subtype from an existing object.

...

objects are defined by classes, classes can inherit attributes and behavior from pre-existing classes called base classes, or super classes.

The resulting classes are known as derived classes or subclasses.

(http://en.wikipedia.org/wiki/Inheritance_%28object-oriented_programming%29)

Review/Questions
Lambda
Functional Programming
Object Oriented Programming
Python Classes
Subclassing/Inheritance

Subclassing

A subclass "inherits" all the attributes (methods, etc) of the
parent class.

You can then change ("override") some or all of the attributes to
change the behavior.

The simplest subclass in Python:

```
class A_Subclass(The_SuperClass):
    pass
```

A_subclass now has exactly the same behavior as
The_SuperClass

Review/Questions
Lambda
Functional Programming
Object Oriented Programming
Python Classes
Subclassing/Inheritance

Overriding attributes

Overriding is as simple as creating a new attribute
with the same name:

```
class Circle(object):
    color = "red"
...
class NewCircle(Circle):
    color = "blue"
>>> nc = NewCircle
>>> print nc.color
blue
```

all the self instances will have the new attribute

Review/Questions
Lambda
Functional Programming
Object Oriented Programming
Python Classes
Subclassing/Inheritance

Overriding methods

Same thing, but with methods

```
class Circle(object):
...
    def grow(self, factor=2):
        """grows the circle's diameter by factor"""
        self.diameter = self.diameter * factor
...
class NewCircle(Circle):
...
    def grow(self, factor=2):
        """grows the area by factor..."""
        self.diameter = self.diameter * math.sqrt(2)
```

all the instances will have the new method

Review/Questions
Lambda
Functional Programming
Object Oriented Programming
Python Classes
Subclassing/Inheritance

"Here's a program design suggestion: whenever you override a method, the interface of the new method should be the same as the old. It should take the same parameters, return the same type, and obey the same preconditions and postconditions. If you obey this rule, you will find that any function designed to work with an instance of a superclass, like a Deck, will also work with instances of subclasses like a Hand or PokerHand. If you violate this rule, your code will collapse like (sorry) a house of cards."

ThinkPython 18.10

Review/Questions
Lambda
Functional Programming
Object Oriented Programming
Python Classes
Subclassing/Inheritance

LAB

Step 2:

- Create a couple subclasses of Element, for a <body> tag and <p> tag. Simply override the tag class attribute.
- Extend the Element.render() method so that it can render other elements inside the tag in addition to strings. Simple recursion should do it. i.e. it can call the render() method of the elements it contains.
- Deal with the content items that could be either simple strings or Elements with render methods...there are a few ways to handle that...

Review/Questions
Lambda
Functional Programming
Object Oriented Programming
Python Classes
Subclassing/Inheritance

LAB

Step 3:

- Create a <head> element – simple subclass.
- Create a OneLineTag subclass of Element: It should override the render method, to render everything on one line – for the simple tags, like:
  <title> PythonClass - Class 6 example </title>
- Create a Title subclass of OneLineTag class for the title.
- You should now be able to render an html doc with a head element, with a title element in that, and a body element with some <P> elements and some text.

Review/Questions
Lambda
Functional Programming
Object Oriented Programming
Python Classes
Subclassing/Inheritance

Homework

Catch Up!

Read up on OO if you haven't already

Finish today's Lab

Finish other Homework / Labs you may not have gotten to.

Come up with a project proposal