Lecture 1 – 4/2/13

[boverly@pce.uwedu](mailto:boverly@pce.uwedu)

Need a project proposal by next week.

<https://github.com/PythonCHB> - main python course code

ssh [hhoaglundbiron@teampython.west.isilon.com](mailto:hhoaglundbiron@teampython.west.isilon.com) , password = a

Unit Testing

<http://docs.python.org/2/library/unittest.html> or Nose

A string under the name of a function can be accessed with <function\_name>.\_\_doc\_\_

Unit tests can be added to a test class (no matter the name), as long as the test/function has “test” in the name.

Lecture 2 – 4/9/13

Project GUI library possibility: tk (python version of tcl)

Project Design: Could be just empty classes and methods, could be lines and bubbles

Make sure pdb is installed in Cygwin before next week

More Unit Testing

\_\_init\_\_.py – Read up on it! He glossed over it a bit…

Nose and pytest notes will be up on the website

Decorators

Applications:

* Profiling
* Memoization (caching)
* Frameworks
  + Testing
  + Web (pylons!)

Functions are first class objects!

* Python namespace is a dictionary – assigning a new function to a function called “foo” does not erase its value in the namespace, just assigns the key (“foo”) a different value. If something still references the original value, it **doesn’t get garbage-collected**.
* Functions can contain classes and other functions, or even return an inner function. When returning an inner function, calling that returned function actually calls the inner function, not the outer. In addition, this inner function is re-defined every time the outer function is run.
  + - A function (its code) and context is called a **closure**.

Decorator

* A decorator passes the defined function under it into the decorator function, so that when the defined function is called, it is actually calling the decorator function. It’s a wrapper!
* Multiple decorators can be used at once, but **order matters**.

Lecture 3 – 4/16/13

Lambda functions

Example:

def fact\_with\_lambda(n):

return reduce(lambda x, y: x \* y, range(2, n+1), 1)

Same as:

def fact\_no\_lambda(n):

return reduce(mul, range(2, n+1), 1)

Python debugging and PDB

\_\_repr\_\_

Can be used if you want a class to be represented as a particular string in debugging.

sys module

sys.exc\_info(): information about current excepton in processor

sys.\_getframe(depth\_int): get stack frame at specified depth

sys.settrace(my\_trace\_func): set tracing function, can have certain things happening when certain events fire, specifically:

* + - enter (call)
    - leave (return)
    - next line
    - exception
    - etc.

PDB

Run with (one option of many):

python –m pdb *thing\_to\_be\_debugged*

Interesting commands:

**list**: gives the source code

Any python code works here!