Introduction, Python Setup, Variables

Python for Ecologists

Tom Purucker, Tao Hong, Jon Flaishans, Marcia Snyder

Ecological Society of America Workshop
Minneapolis, MN

purucker.tom@gmail.com

August 2, 2013

Python for Ecologists

- Assuming not much programming experience
- Immersion approach
  - Short lecture on Python topic
  - Hands-on Python exercises
  - Rinse & repeat
- Will use ecological examples as much as possible

Your presenters

- Tom Purucker
- Tao Hong
- Jon Flaishans
- Marcia Snyder

Why bother with Python?

- A scripting language (like R) but also,
- A high level programming language
- Strong libraries for mathematical sciences, engineering
- Designed to produce readable code
- Cross-platform
- Open source, free
- Plays well with other technologies

übertool Python project

- http://www.ubertool.org
- Created with Python as the science engine
- Integrates easily with web technologies such as HTML, JavaScript, JQuery

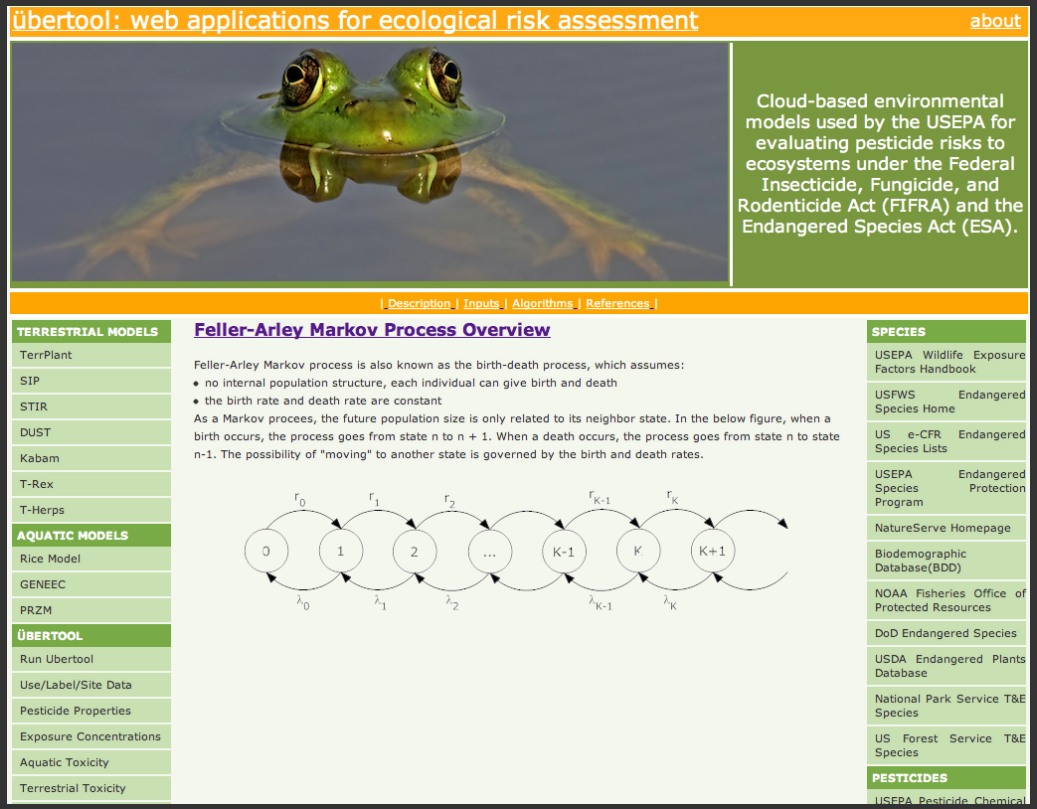

Figure: übertool ecological risk web application

Getting setup

- We will use Python 2.7 (not 3)
  - http://www.python.org/getit/
- For Windows users
  - https://code.google.com/p/pythonxy/wiki/Downloads?tm=2

Some extra libraries to install

- numpy- http://sourceforge.net/projects/numpy/

Download the exercise scripts for this class

- http://www.ubertool.org
- Created with Python as the science engine

Opening a shell and running Python

- Mac- Spotlight and type ’terminal’

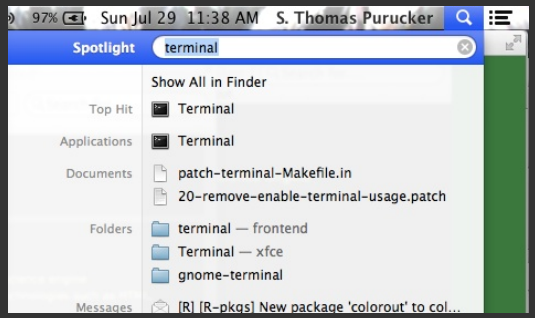

Figure: Opening terminal in OS X

- Windows- Type ’cmd’ in search window for command prompt

Figure: Opening the command prompt in Windows 7

Check Python installation

1. Type ’python’ at the shell prompt
2. Then type at the Python prompt:

```
import sys
sys.version
import numpy
numpy.__version__
quit()
```

Run a script at the command line

```
# save this in a text file as hello.py
print "Hello Minneapolis!"
# then navigate to its directory in a shell
# and run at the command prompt with
# python hello.py
```

Run IDLE

- IDLE is the "Interactive DeveLopment Environment" bundled with Python
- Type ’IDLE’ in Mac Spotlight or Windows search window
- Or type ’idle’ from the python prompt

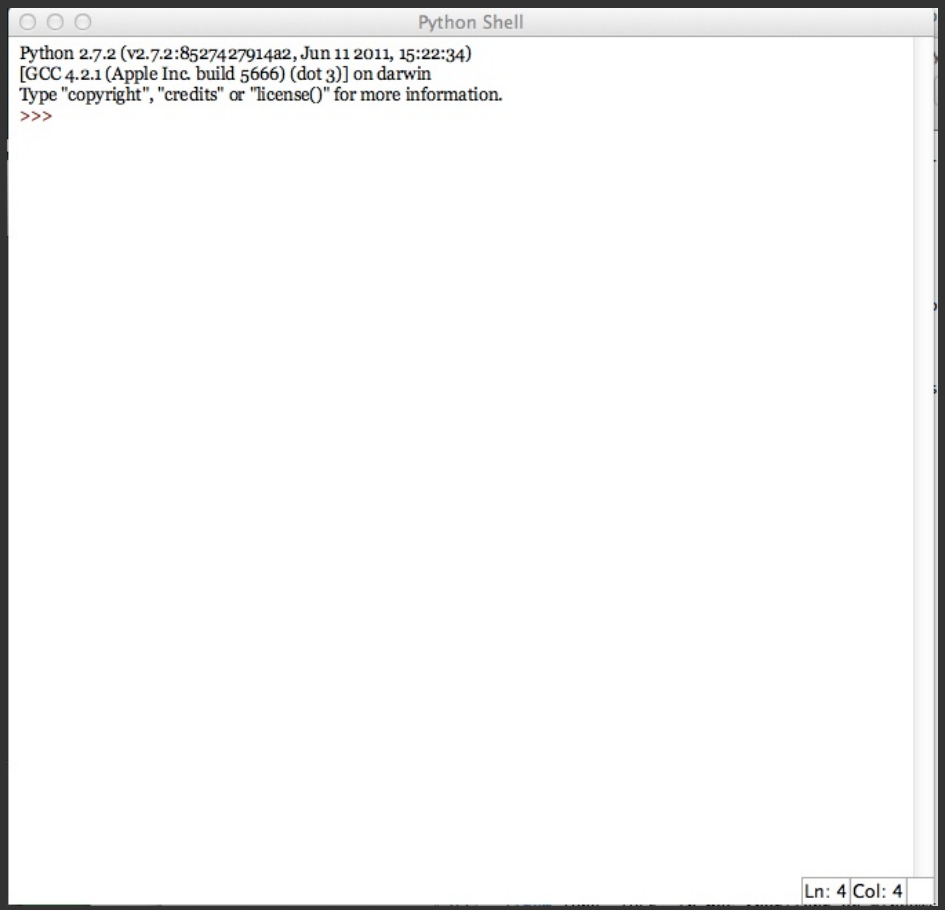

Figure: IDLE in OS X

Run hello.py with IDLE

1. Open hello.py in scripts directory with File -> Open
2. Run hello.py with Run -> Run Module or (fn) F5

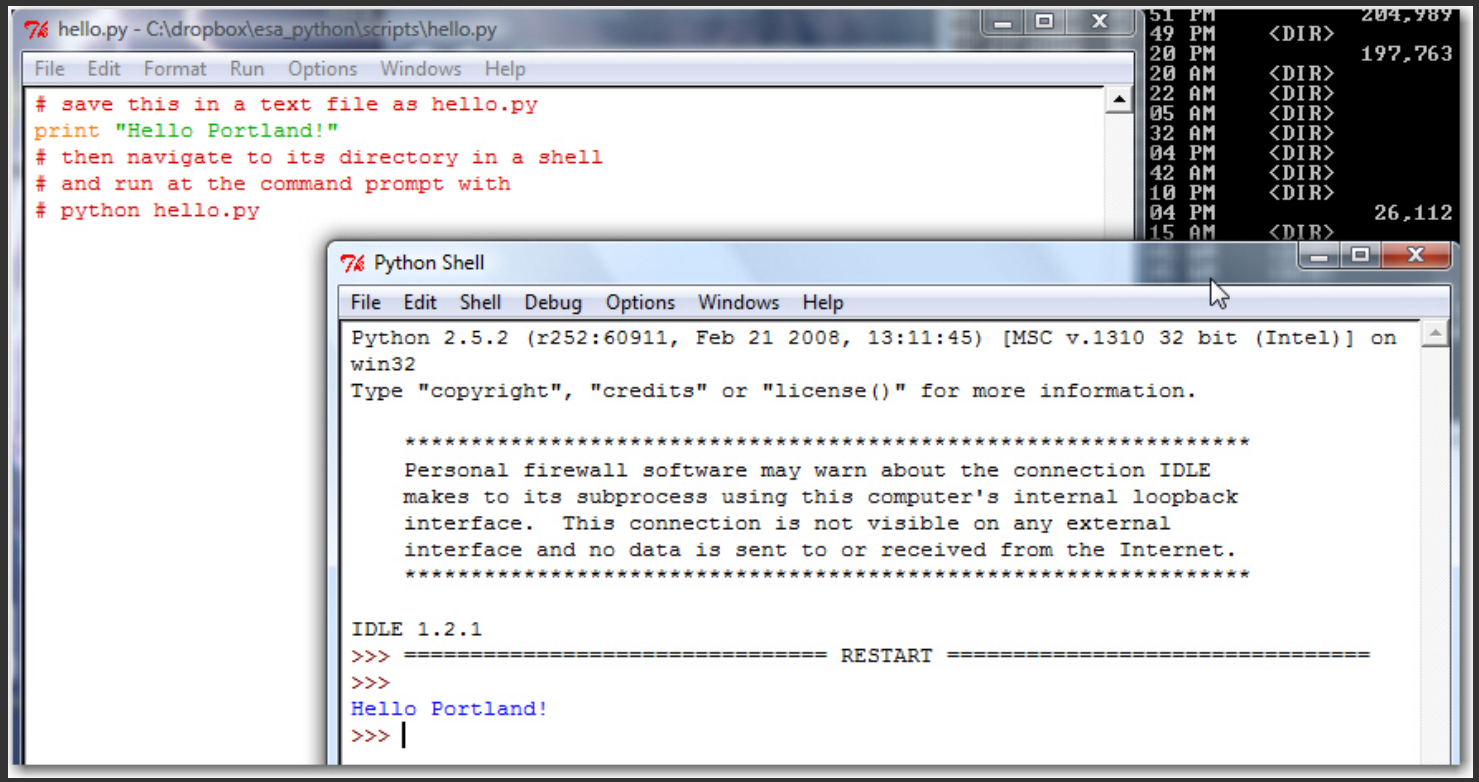

Figure: Result of running hello.py with IDLE

Command Line with IDLE

1. Use the shell window to run commands like at a terminal/cmd prompt
2. Run hello.py with Run -> Run Module or (fn) F5

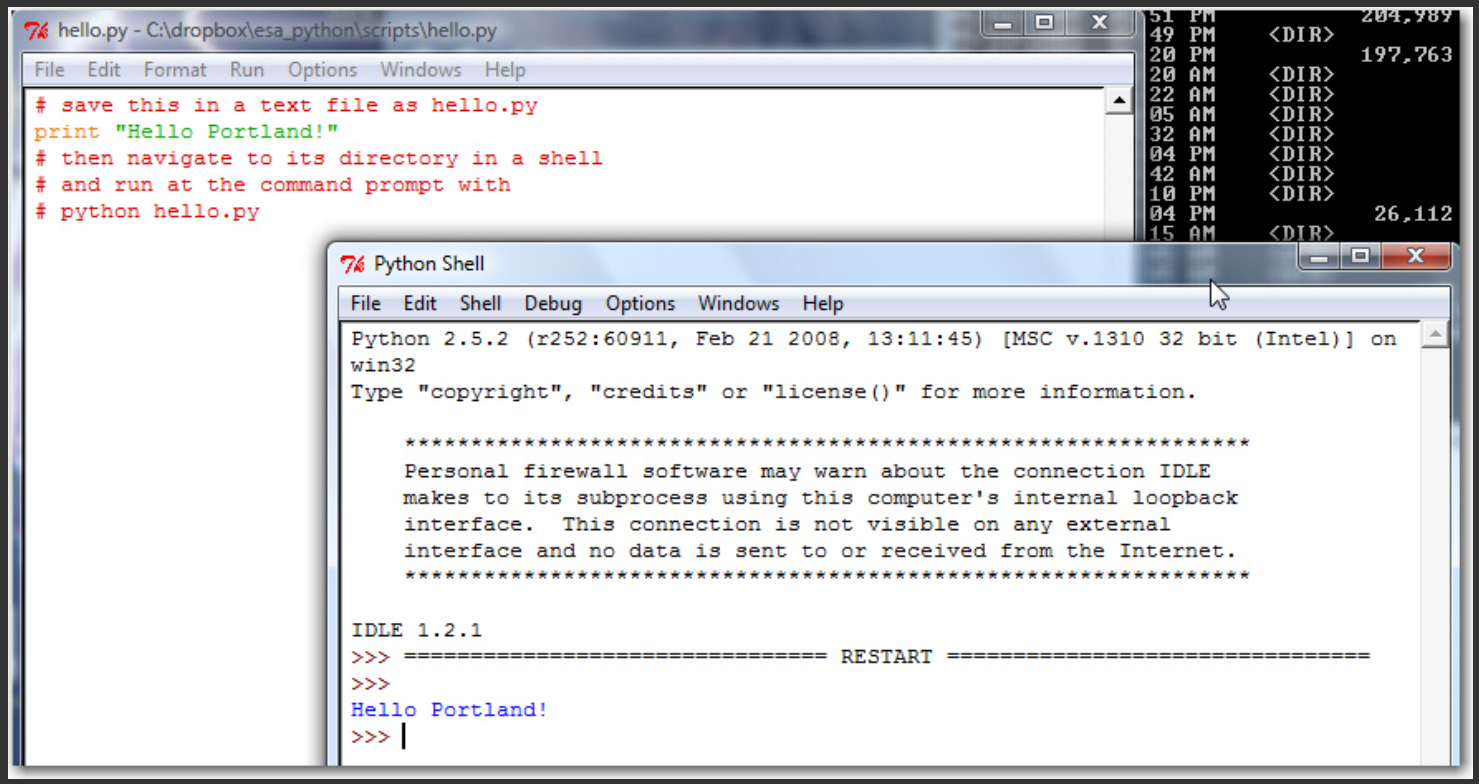

Figure: Result of running hello.py with IDLE

Variables

- No declaration of variables necessary!

```
pop_size = 112 # integer
type(pop_size)
pop_density = 4 # still an integer
type(pop_density)
pop_density = 4. # now its a float
type(pop_density)
species_name = "Oedipina_complex" # string
type(species_name)
species_name = "4" # still a string
type(species_name)
```

Basic math operations

Operation	Sign
Addition	+
Subtraction	-
Multiplication	*
Division	/
Power	**
Modulus	%

Be careful about int v float

```
>>> pop_size = 1086
>>> area = 1254
>>> pop_density = pop_size/area
>>> print(pop_density)
0
>>> type(pop_density)
<type 'int'>
```

Beware

- Declare floats by using a decimal point
- e.g., pop_size = 1086.

Python variable naming conventions

- all lowercase
- cannot start with numbers
- separate_words_with_underscores
- Style Guide for Python:
  - http://www.python.org/dev/peps/pep-0008/

unittest exercises

- Exercise 1 uses the unittest library so you can type code and test the result yourself
  1. Edit the script in IDLE between the # and the selfassert calls
  2. Run it
  3. If it complains, fix it and run it again!

Beware

- Python is very picky about space formatting, start your editing right below each # (8 spaces over)
- Python is case-sensitive- diffusion_rate and Diffusion_rate are different variables

Exercise 1- Run the script exer01_variables.py

```
import unittest

class TestVariables(unittest.TestCase):
    def test_variables(self):
        # create the variable ‘diffusion_rate‘,
        # and assign it a float value of 6.0
        # ***************************************

        self.assertEqual(diffusion_rate, 6.)
        self.assert_(isinstance(diffusion_rate, float))

        # assign ‘‘cohort_size‘‘ to an integer value of 84
        # ***************************************

        self.assertEqual(cohort_size, 84)
        self.assert_(isinstance(cohort_size, int))

        # create a variable ‘species_name‘,
        # and assign it to ’Pieza kake’
        # ***************************************

        self.assertEqual(species_name, "Pieza␣kake")
        self.assertTrue(isinstance(b, str))

if __name__ == ’__main__’:
    unittest.main()
```