Introduction, Python Setup, Variables

Python for Ecologists

Tom Purucker, Tao Hong, Chance Pascale

Ecological Society of America Workshop
Portland, OR

purucker.tom@gmail.com

August 4, 2012

Notes

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

Notes

Your presenters

- Tom Purucker
- Tao Hong
- Chance Pascale

Notes

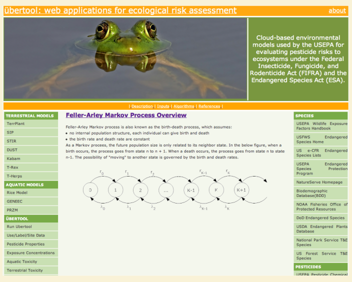
Why bother with Python?

- 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

Notes

übertool Python project

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


übertool: web applications for ecological risk assessment
about
| Description | Inputs | Algorithms | References |
TERRESTRIAL MODELS
TerrPlant
SIP
STIR
DUST
Kabam
T-Rex
T-Herps
AQUATIC MODELS
Rice Model
GENEEC
PRZM
ÜBERTOOL
Run Ubertool
Use/Label/Site Data
Pesticide Properties
Exposure Concentrations
Aquatic Toxicity
Terrestrial Toxicity
SPECIES
USEPA Wildlife Exposure Factors Handbook
USFWS Endangered Species Home
US e-CFR Endangered Species Lists
USEPA Endangered Species Protection Program
NatureServe Homepage
Biodemographic Database(BDD)
NOAA Fisheries Office of Protected Resources
DoD Endangered Species
USDA Endangered Plants Database
National Park Service T&E Species
US Forest Service T&E Species
PESTICIDES
USEPA Pesticide Chemical
0
1
2
...
K-1
K
K+1
r_0
r_1
r_2
r_{K-1}
r_K
λ_0
λ_1
λ_2
λ_{K-1}
λ_K

Cloud-based environmental models used by the USEPA for evaluating pesticide risks to ecosystems under the Federal Insecticide, Fungicide, and Rodenticide Act (FIFRA) and the Endangered Species Act (ESA).

Feller-Arley Markov Process Overview

Feller-Arley Markov process is also known as the birth-death process, which assumes:

- no internal population structure, each individual can give birth and death
- the birth rate and death rate are constant

As a Markov procees, the future population size is only related to its neighbor state. In the below figure, when a birth occurs, the process goes from state n to n + 1. When a death occurs, the process goes from state n to state n-1. The possibility of "moving" to another state is governed by the birth and death rates.

Figure: übertool ecological risk web application

Notes

Getting setup

- We will use Python 2.7 (not 3)
  - http://www.python.org/getit/
- For Windows users
  - http://portablepython.com/wiki/Download

Notes

Some extra libraries to install

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

Notes

Download the exercise scripts for this class

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

Notes

Opening a shell and running Python

- Mac- Spotlight and type ’terminal’

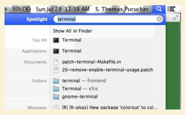

97% Sun Jul 29 11:38 AM S. Thomas Purucker
Spotlight
terminal
Show All in Finder
Top Hit
Terminal
Applications
Terminal
Documents
patch-terminal-Makefile.in
20-remove-enable-terminal-usage.patch
Folders
terminal — frontend
Terminal — xfce
gnome-terminal
Messages
[R] [R-pkgs] New package 'colorout' to col...

Figure: Opening terminal in OS X

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

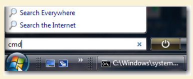


Figure: Opening the command prompt in Windows 7

Notes

Check Python installation

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

```
import sys
sys.version
import numpy
import scipy
import matplotlib
quit()
```

Notes

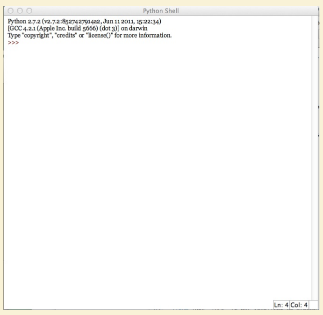
Run a script at the command line

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

Notes

Run IDLE

- IDLE is the "Interactive DeveLopment Environment" bundled with Python
- Type ’IDLE’ in Mac Spotlight or Windows search window



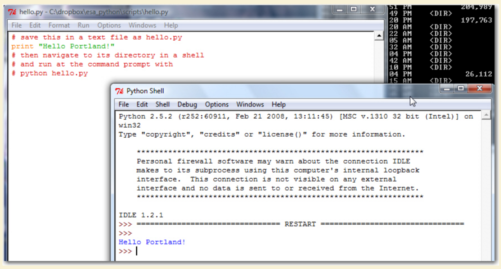
```
Python 2.7.2 (v2.7.2:8527427914a2, Jun 11 2011, 15:22:34)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>>
```

Figure: IDLE in OS X

Notes

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



hello.py - C:\dropbox\esa_python\scripts\hello.py

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

Python Shell

```
Python 2.5.2 (r252:60911, Feb 21 2008, 13:11:45) [MSC v.1310 32 bit (Intel)] on
win32
Type "copyright", "credits" or "license()" for more information.

    ****************************************************************
    Personal firewall software may warn about the connection IDLE
    makes to its subprocess using this computer's internal loopback
    interface.  This connection is not visible on any external
    interface and no data is sent to or received from the Internet.
    ****************************************************************

IDLE 1.2.1
>>> ================================ RESTART ================================
>>>
Hello Portland!
>>> 
```

```
51 PM              204,989
49 PM     <DIR>
20 PM              197,763
20 AM     <DIR>
22 AM     <DIR>
05 AM     <DIR>
32 AM     <DIR>
04 PM     <DIR>
42 AM     <DIR>
10 PM     <DIR>
04 PM               26,112
15 AM     <DIR>
```

File Edit Format Run Options Windows Help
File Edit Shell Debug Options Windows Help

Figure: Result of running hello.py with IDLE

Notes

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)
```

Notes

Basic math operations

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

Notes

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.

Notes

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/

Notes

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

Notes

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()
```

Notes

Notes