

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
- ▶ Chance Pascale


Why bother with Python when I have R?

- ▶ A scripting language (like R) but also,
- ▶ A high level programming language
- ▶ Designed to produce readable code
- ▶ Cross-platform

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



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).

[Description](#) | [Inside](#) | [Algorithms](#) | [References](#)

TERRESTRIAL MODELS

- TerPlant
- SIP
- STIR
- DUST
- Kabam
- T-Rex
- T-Herps

AQUATIC MODELS

- Rice Model
- GENESC
- PRZM

ÜBERTOOL


- Run Übertool
- Use/Label/State Data
- Pesticide Properties
- Exposure Concentrations
- Aquatic Toxicity
- Terrestrial Toxicity

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 process, 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.



SPECIES

- USEPA Wildlife Exposure Factors Handbook
- USFWS Endangered Species Home
- US e-CHR Endangered Species List
- 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 Chemicals

Figure: übertool ecological risk web application

Getting setup

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

Some extra libraries to install

- ▶ numpy- <http://sourceforge.net/projects/numpy/>
- ▶ scipy- <http://sourceforge.net/projects/scipy/files/>

Need a text editor

- ▶ Linux-/
 - ▶ Mac- TextWrangler, Smultron, TextEdit (already installed)
 - ▶ Windows- Notepad (already installed), Notepad++, TextPad

Python IDLE IDE

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

Python objects

- ▶ Everything in Python is an object with these properties
 - ▶ an identity (id)
 - ▶ a type (type)
 - ▶ a value (mutable or immutable)

Each Python object has an id

```
>>> n_predators = 12  
>>> id(n_predators)  
4298191056
```

Each Python object has a type

```
>>> n_predators = 12  
>>> type(n_predators)  
<type 'int'>
```

Each Python object has a value

- ▶ String, integer, and tuple object values are *immutable*

```
>>> n_prej = 88
>>> id(n_prej)
4298193184
>>> n_prej = 96
>>> id(n_prej)
4298192992 # id for n_prej has changed
```

- ▶ Dictionary and list items are *mutable*

```
>>> birds = ["cardinal", "oriole"]
>>> id(birds)
4332756000
>>> birds.append("gnatcatcher")
>>> id(birds)
4332756000 # id is still the same
```

Variables

```
pop_size = 112 # integer  
pop_density2 = 4 # still an integer  
pop_density = 4. # float  
species_name = "Oedipina_complex" # string  
species_name = "4" # still a string
```

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/>

Exercise 1

`variables.py`

Hello world! (from an interpreter)

```
>>> python
```

```
>>> print "Hello_world!"
```

```
Hello world!
```


Hello world! (from a script)

save this in a text file as hello.py

```
print "Hello_world!"
```

then run at the command prompt with 'python hello.py'

Hello world! (as an executable)

```
#!/usr/bin/env python  
# the above line automatically invokes Python  
# save this in a text file as hello_exe.py  
print "Hello_world!"  
# we can run this with simply "hello"
```