# Python modules and packages

Reuven M. Lerner, PhD
reuven@lerner.co.il

# Modules

- Modules allow you to put together functionality
- Modules are used when defining objects
  - (We'll get to this later)
- Act as namespaces, to avoid collisions

# Using modules

- Very simple:

```
import modulename
```

- Now foo from modulename is available to your program as

```
modulename.foo
```

- To import names into the current namespace:

```
from modulename import *
```

# Aliasing a module

- Instead of

```
import numpy
```

- you can say

```
import numpy as np
```

- And then, instead of numpy.arange, you can use np.arange

# Example

```
import os

dir(os)     # List of items in os

print os.getuid()

getuid()    # results in an error

from os import getuid

print getuid()
```

# Aliasing functions

```
from os import getuid as GGUU

print getuid()   # Error

print GGUU()     # Success!

import os

GGUU == os.getuid # True

GGUU is os.getuid # True
```

# What namespace am I in?

- You can find out by checking __name__
- If you're at the top level, you're in the namespace named '__main__'
- If your module wants to execute something when executed (not imported) check if __name__ == '__main__'
- Note: __name__ is an object, but '__main__' is a string

# Modules are objects!

```
>>> import os

>>> type(os)

<type 'module'>
```

# Writing a module

- Create a file with a .py extension.
- Put some code into that file
- Voila!  You have a module!
- From another program/file, type

```
import mymod
```

- Access variables and functions as mymod.X

# dir(module)

- To learn what names are available for a module, use dir()
- You can actually use dir() to find attributes of any named object

# Module search path

- Where does Python look for modules when you import them?
  - Home directory of the program
  - PYTHONPATH directories
  - Standard library directories
  - Contents of .pth files
- In other words: sys.path

# .pth files?

- File has the form NAME.pth
- It must be in one of the default directories that Python searches
- Unix:

```
sys.prefix + lib/site-python

sys.prefix + lib/pythonX.Y/site-packages
```

# .pth files

- The file must then contain one or more directories to be added to sys.path
- Directories that don't exist are ignored
- You can accidentally include files — don't!
- This is all handled by the standard "site" module, which is included at startup

# Packages

- Modules can exist in a directory structure
- This is called a “package”
- there is an __init__.py in the top-level directory
- __init__.py can import what it wants
- You can then have one or more sub-packages

# Relative imports

- From within a package, you can also say

```
from .modname import thing
```

- That looks in the directory above modname
- You shouldn't really use this unless you have to
- You cannot use this except within a package

# Namespaces

- Remember that every package or module is a namespace!
- So if function C is in module B in package A, you can probably “import A”, but invoke the function as A.B.C( ).

# Python standard library

- Python comes with lots of modules and packages
- Use them!  Don’t re-invent the wheel.

# Example modules

- os — Access the filesystem and OS-specific data, such as environment variables
- sys — interpreter system, including version number, system arguments, and even loaded modules
- time, StringIO, textwrap, math, urllib2, tarfile — and much, much more
- http://docs.python.org/library/

# Installing third-party modules

- Download from the Web
- Open the tarfile/zipfile
- Enter the directory

```
python setup.py install
```

# Pypi

- Also known as the “Cheese Shop”
- Online repository of Python modules
- All can be downloaded and installed using the regular setup system

# Pip

- A replacement for easy_install
  - “Pip installs packages”
- Search function!
- And of course, install:

```
pip install pyreadline
```

# Modern PIP installation

- pip is included in Python 3.4 and 2.7.10
- Running an earlier version?  Your best bet is to upgrade to Python 2.7.9, which includes pip!
- On Windows, look for "pip" in the "scripts" directory

```
c:\python27\scripts\pip
```

# Upgrade

- Upgrade one:

```
pip install -U pyreadline
```

- Upgrade all:

```
pip freeze | cut -d = -f 1  | xargs pip install -U
```

# localshop

- Provides a local version of the Cheese Shop (PyPi)
- Upload, manage your packages locally, without exposing them to the outside world
- It also acts as a caching server — so if you want something from PyPi, you can get it
- Pip can be configured to grab things from here, rather than

# .piprc with localshop

```
[distutils]

     index-servers =   local

[local]

    username: myusername

    password: mysecret

    repository: http://localhost:8000/simple/
```