Advanced Python

Whats all this then?

Biology Dept, Duke University
Hunter Matthews
thm@duke.edu

What is Python?

Advanced, Object Oriented scripting language

"Batteries Included" - 6GB of CPAN not required for hello world.

Despite that, there are a large number of add-on modules.

Twisted, BioPython, Numerical Python, SciPy

Database Modules, SAP R/3 modules...

Changes from 1.5 to 2.0

Unicode Support, Augmented assignment, Cycles GC, XML included

String Methods

    foo.strip()   instead of string.strip(foo)

Distutils

    "Build" system for python packages (including C extensions)

print statement change

    print >> sys.stderr, "Warning: action field not supplied"

gettext, pyexpat, zipfile, and _winreg new modules

List Comprehensions

List Comprehensions

```
[ expression for expr in sequence1
            for expr2 in sequence2 ...
            for exprN in sequenceN
            if condition ]
```

```
for expr1 in sequence1:
 for expr2 in sequence2:
 ...
    for exprN in sequenceN:
        if (condition):
       #append value to resulting list
```

For Example:

```
seq1 = 'abc'
seq2 = (1,2,3)
>>> [ (x,y) for x in seq1 for y in seq2]
[('a', 1), ('a', 2), ('a', 3), ('b', 1), ('b', 2), ('b', 3), ('c', 1),
('c', 2), ('c', 3)]
```

Changes from 2.0 to 2.1

Nested Scopes now work as expected

Rich Comparison method names (__lt__, __ne__, etc)

Warning Framework

```
import warnings
warnings.filterwarnings(action = 'ignore',
                        message='.*regex .* deprecated',
                        category=DeprecationWarning,
                        module = '__main__')
```

pydoc, doctest/PyUnit, usability improvements to the time module

__all__ variable

Changes from 2.1 to 2.2

Type and class changes

- You can sub-class builtins now (class MyList(list):)
- New style classes - class New(object): blah blah
- int, float, str, dict, iter and file might as well be keywords now

Diamond Rule for multiple Inheritance

__getattribute__ added (called always, __getattr__ only on dict lookup fail)

__slots__ - way to restrict legal attribute names

ints and long ints are now unified. "L" is now deprecated.

os.stat(filename)[stat.ST_SIZE] ==> file_size = os.stat(filename).st_size

xmlrpclib, hmac, socket is now ipv6, re is faster, smtplib is TLS

Iterators

Made readlines() just as fast as xreadlines() (and deprecated it)

Iterator is (potentially) any object with next() method.
raises StopIteration when its done

L = [1,2,3] ; i = iter(L) ; i.next() == 1; i.next() == 2;
last time StopIteration will be raised.

for key in myDict: print key, myDict[key]

Generators

Basically, a reentrant function that saves locals()

```
def generate_ints(N):
    for i in range(N):
        yield i
```

When you call a generator function, it doesn't return a single value; instead it returns a generator object that supports the iterator protocol.

Someone has even used generators to simplfy multi-threading!

Division is changing.

The age old problem of 1 / 2 = 0 is being fixed.

// will be the "floor" division operator, just like classic /.

In the future, / will perform floating point division.

So 1 / 2 = 0.499999

The future is NOW. (2.3)

Set datatype added. Both mutable and immutable varients.

Universal newline support.

enumerate(L) ==> (0, L[0]), (1, L[1]), (2, L[2]) etc

bool type, FINALLY. bool, True and False are now essentially keywords.

You can import modules directly from zip archives, ala java
    Just add the zip file to the sys.path

```
if 'ab' in 'abcd':    # is now valid sub-string checking
```

modules: csv, pickle, PyBSDDB, bz2, platform, tarfile, testwrap, timeit

[2.3] logging package

```
import logging
logging.debug('Debugging information')
logging.info('Informational message')
logging.warning('Warning:config file %s not found', 'server.conf')
```

Logger can log to files, raw sockets, syslog, email, NTEvents, and more.

Logger is backwards compatible with 1.5, and is really cool.

I wish I could get it to work for me...

optparse

Replacement for getopt, originally developed as Optik.

```
from optparse import OptionParser
op = OptionParser()
op.add_option('-i', '--input',
              action='store', type='string', dest='input',
              help='set input filename')
op.add_option('-l', '--length',
              action='store', type='int', dest='length',
              help='set maximum length of output')
```

Generates help for you, and gives you a single object with the flags as attributes.

Pickling Example

```
import pickle      # never cPickle
obj = random_python_object()
...stuff...

fh = open('saurkraut', 'w')
pickle.dump(obj, fh)
fh.close()

fh2 = open('saurkraut', 'r')
obj2 = pickle.load(fh2)
```

obj2 is now equivalent to obj.

PySqlite

Beware the conn.commit() boojum.

```
import sqlite

conn = sqlite.connect(db="db", mode=077)
cursor = conn.cursor()
cursor.execute("""
   select genus, species, family, category
   from calflora
   order by species, genus
   """)
for row in cursor.fetchall():
   print "%14s, %15s, %19s, %8s, %25s, %i" % ( row.genus,
                                    row.species,
                                    row.family,
                                    row.category)
conn.close()
```

Psyco

Basically, its a JIT compiler, written as a module.
So using it is a lead pipe cinch.

```
import psyco
psyco.full()    # full program is JIT
```

```
import psyco
psyco.bind(myfunction1)    # only these two functions are sped up
psyco.bind(myfunction2)    # useful if these are the core loops.
```

Building psyco is left as an exercise for the programmer.

Writing Extensions

```
#include <Python.h>
static PyObject * spam_system(PyObject *self, PyObject *args){
   char *command;
   int sts;

   if (!PyArg_ParseTuple(args, "s", &command))
       return NULL;
   sts = system(command);
   return Py_BuildValue("i", sts);
}

static PyMethodDef SpamMethods[] = {
   {"system",  spam_system, METH_VARARGS,
    "Execute a shell command."},
   {NULL, NULL, 0, NULL}        /* Sentinel */
};

void initspam(void) {
   (void) Py_InitModule("spam", SpamMethods);
}
```

XMLRPC

```
import xmlrpclib
server = ServerProxy("http://betty.userland.com")

try:
   # Using the server object, call "examples.getStateName()" with
   # an argument of 41
   print server.examples.getStateName(41)
except Error, v:
   print "ERROR", v
```

Resources

The home page of all things pythonic

- http://www.python.org

Python Quick Reference - everything you need, on one page.

- http://www.brunningonline.net/simon/python/PQR.html

Daily Python URL

- http://www.pythonware.com/daily/

Tons of Python apps, modules for python, etc

- http://www.freshmeat.net

Python.faqts

- http://python.faqts.com

Resources

Beginning python

- http://www.uselesspython.com/index.html

Python Blog

- http://www.deadlybloodyserious.com/Python

Introduction to Numerical Python

- http://www.onlamp.com/pub/a/python/2000 /05/03/numerically.html

Psyco (JIT)

- http://psyco.sourceforge.net

Resources

O'Reillys web site (I don't care for this one)

- http://www.onlamp.com/python/

Biology and Python

- http://biopython.org/

All the power of python with all the incompatibility of java.

- http://www.jython.org/

Python makes windows suck a little less

- http://www.python.org/download/ download_windows.html

Resources

Python and GTK (and links to pygnome)

- http://www.daa.com.au/~james/pygtk/

Online Python Journal

- http://pythonjournal.cognizor.com/

Published (like, as in paper) Python Journal

- http://www.pyzine.com/

Python Book Reviews -- Bad HTML, great content

- http://www.awaretek.com/book.html

Development is a state of mind.