PyCodeConf 2011

What makes Python Awesome ?

by Raymond Hettinger
@raymondh

Context for Success

- License
- Commercial Distributions
- Zen
- Community
- Repository of Modules (PyPi)
- Killer Apps and Success Stores
- Win32
- Books

High level qualities of Python

- Ease of Learning
- Rapid Development Cycle
- Economy of Expression
- Readability and Beauty
- One way to do it
- Interactive Prompt
- Batteries Included
- Protocols -- wsgi, dbapi, …

A bit of awesomeness in five minutes

```
# Search directory tree for all duplicate files

hashmap = {}  # content signature -> list of filenames

for path, dirs, files in os.walk('.'):
    for filename in files:
        fullname = os.path.join(path, filename)
        with open(fullname) as f:
            d = f.read()
        h = hashlib.md5(d).hexdigest()
        filelist = hashmap.setdefault(h, [])
        filelist.append(fullname)

pprint.pprint(hashmap)
```

Why is Python awesome?

Surely, something makes it great?

Aren't all scripting languages the same?

Are there any unique features?

What will propel Python into future?

What will other langauges copy from us?

Winning Language Feature: Indentation

This is how we write our pseudo code

It contributes to Python's clean, uncluttered appearance

It was an audacious move

One of the secrets to debugging C is to run it through a beautifier so the indentation will reflect the actual logic, instead of the programmers intended logic

With Python, the indentation is executable so the visual appearance and actual execution always match.

Indentation: Why you need it

```
for (i=0 ; i<10 ; i++);
    printf("Good morning\n");


if (x < y)
    if (pred(x))
        printf("One");
else if (x == y)
    printf("Two")
else
    printf("Three");
```

Winning Language Feature: Iterator Protocol

- High level glue that holds the language together
- Iterables: strings, lists, sets, dicts, collections, files, open urls, csv readers, itertools, etc
- Things that consume iterators: for-loops, min, max, sorted, sum, set, list, tuple, dict, itertools
- Can be chained together like Unix pipes and filters

Iterators

```
sorted(set('abracadabra'))

sorted(set(open(filename)))

cat filename | sort | uniq
```

Winning Language Feature: List Comprehensions

- Arguably, one of the most loved language features
- Very popular addition to Python
- Derived from notation used in mathematics
- Clean and beautiful
- Much more flexible and expressive than map, filter, and reduce

List Comprehensions

```
[line.lower() for line in open(filename)
              if 'INFO' in line]
```

```
sum([x**3 for x in range(10000)])
```

Generators

Easiest way to write an Iterator

Simple syntax, only adds the YIELD keyword

Remembers state between invocations:

- the stack including open loops and try-statements; the execution pointer; and local variables

Generator Example

```
def pager(lines, pagelen=60):
    for lineno, line in enumerate(lines):
        yield line
        if lineno % pagelen == 0:
            yield FORMFEED
```

Winning Language Features: Genexps, Set comps, and Dict comps

Logical extension of list comprehensions and generators

to unify the language

Genexps Setcomps and Dictcomps

```
sum(x**3 for x in xrange(10000))
```

```
{os.path.splitext(filename)[1]
      for filename in os.listdir('.')}
```

```
{filename: os.path.getsize(filename)
      for filename in os.listdir('.')}
```

Generators that accept inputs

- Generators support send(), throw(), and close()
- Unique to Python
- Makes it possible to implement Twisted's inline deferreds

Two-way generator example

```
@inline_deferred
def session(request, cleared=False):
    while not cleared:
        cleared = yield authenticate(request.user)
    db_result = yield database_query(request.query)
    html = yield format_data(db_result)
    yield post_result(html)
    return end_session()
```

Winning Language Feature: Decorators

- Expressive
- Easy on the eyes
- Works for functions, methods, and classes
- Adds powerful layer of composable tools

Complete web service using Itty

```
from itty import get, post, run_itty
import os, subprocess

@get('/env/(?P<name>\w+)')
def lookup_environ_variable(request, name):
    return os.environ[name]

@get('/freespace')
def compute_free_disk_space(request):
    return subprocess.check_output('df')

@post('/restart')
def test_post(request):
    os.system('restart')

run_itty()
```

Winning Language Feature: With-statement

- Clean, elegant resource management: threads, locks, etc.
- More importantly, it is a tool for factoring code
- Factors-out common setup and teardown code
- Few languages currently have a counterpart to the with-statement

Context mangers are easy to use

```
with locking:
    access_resource()
```

Winning Language Feature: Abstract Base Classes

Uniform definition of what it means to be a sequence, mapping, etc

Ability to override isinstance() and issubclass()

- The new duck-typing, “if it says it’s a duck …”

Mix-in capability

Abstract Base Class: Mix-in

```
class ListBasedSet(collections.Set):

    def __init__(self, iterable):
        self.elements = lst = []
        for value in iterable:
            if value not in lst:
                lst.append(value)

    def __iter__(self):
        return iter(self.elements)

    def __contains__(self, value):
        return value in self.element

    def __len__(self):
        return len(self.elements)
```

Winning Language Features: Summary

- Indentation
- Iterator Protocol
- Generators
- List comps, set comps, dict comps, and genexps
- Two-way generators
- Decorators
- With-statement
- Abstract Base Classes

Anything Else?

Is that all that makes Python awesome?

Time for your thoughts and questions