Python for Optimisation

Toby Davies

Biarri Commercial Mathematics

13 November 2012

Outline

Outline

We all need to implement our mathematics somehow

What we want in a language:

- don't obscure the mathematics
- or the algorithm
- integrates with everything

Python is a fantastic language for implementing formulations and algorithms

I'll go into more depth on python's strengths:

- As a Turing-complete modelling language
  - For **using** algorithms
- As a programming language
  - For **implementing** algorithms
- As a "scripting language" for integrating
  - For **integrating** algorithms with delivery technologies

Outline

Generators make for readable code

```
mincost, choice = min((cost(act), act)
                      for act in actions)
```

Generators (continued)

```
cost, action = min((cost(dx, dy), (dx, dy))
                   for dx in [-1, 0, 1]
                   for dy in [-1, 0, 1]
                   if 0 <= x + dx <= MAXX
                      and
                      0 <= y + dy <= MAXY)
```

Generators make modelling easy

```
import pulp as p
board = {(x, y): p.LpVariable(cat=p.LpBinary)
          for x in range(n) for y in range(n)}
cols = [sum(board[x, y] for y in range(n)) == 1
         for x in range(n)]
rows = [sum(board[x, y] for x in range(n)) == 1
         for y in range(n)]
diag = [ (v1 + v2) <= 1
          for (x1, y1), v1 in board.items()
          for (x2, y2), v2 in board.items()
          if x2 - x1 == y2 - y1]
```

Generators make modelling easy (continued)

```
prob = p.LpProblem()
for constraint in rows + cols + diag:
    prob += constraint
prob.solve()
for y in range(n):
    print ''.join(('x' if p.value(board[x, y]) else '-')
                  for x in range(n))
```

Decorators: Say what you mean, not how to do it

```
@memoize
def fib(x):
    """
    naive, recursive definition.
    dynamic programming implementation.
    """
    if 0 <= x <= 1:
        return 1
    return fib(x-1) + fib(x-2)
```

Michael's code, refactored

```
@memoize
def bestorder(time, stock):
    "Return the best (cost, order) pair"
    if time >= Periods:
        return (0, 0)

    return min((cost(order, stock, time), order)
               for order in range(MaxOrder + 1))
```

Michael's code (continued)

```
def cost(order, stock, t):
    "Return the minimum cost of buying `order` new items"
    unmet = mid(0, (Demand[t] - stock - order), Demand[t])
    new = mid(0, stock + order - Demand[t], MaxStock)

    immediate = ((20 + 5 * order if order > 0 else 0)
                 + new * HoldingCost
                 + unmet * StockOutCost)

    # Recursively calculate the tail cost
    longterm, _nextorder = bestorder(t+1, new)

    return immediate + longterm
```

Outline

Dynamic typing means less boilerplate

- To parse our Excel->C++ (and python->C++) xml format:
  - C++: 250 lines of library, 1600 lines building datastructures
  - Python: 50 lines of library, <10 lines building datastructures
  - Which is the better ratio?

Static code analysis gives you many benefits of compiler warnings

- Dynamic code **is** harder to analyse
  - But far from impossible
- pylint
  - Integrated with my editor - I see most errors before I **save**
  - And it usually tells me when I'm making bad design choices

Strong, Dynamic typing means you still catch errors before you break stuff!

Wait, what's all this "typing" business?

A theory of type systems (an aside)

	Weak	Strong
Dynamic	Values typed Conversions implicit	Values typed Conversions explicit
Static	Variables typed Conversions implicit	Variables typed Conversions explicit

We want to be dynamic to reduce boilerplate, but what about weak vs strong?

Type system exercises

Weak typing can seem like a neat idea:

- "I am " + age
- "0" == 0?

But:

- does "0" + 3 == "03" or 3?
- raw_input() < 0? how about int(raw_input()) < 0
- ("03" + phone for phone in contactnumbers)
- (base + release for base in products for release in ["-current", "-11"])

Strong, Dynamic typing means you still catch errors before you break stuff!

- No masked errors (what if we'd put those wrong contact numbers on our website)
  - Or started optimising around a product "89" instead of "100-11"?
- Being explicit means the language can't guess wrong

Python is "fast enough"

- Efficient dictionaries and sets O(1) amortised lookup, insertion and deletion
- Efficient lists with O(1) random access
- Usually only constant factor slower for same algorithm in C/C++
- Brevity and ease of use make it easier to see and make algorithmic improvements

Python is "fast enough"

- Except when it's not
  - Global Interpreter Lock (GIL) makes using threads a little harder
  - "Boxing" of primitives adds overhead
- Boost.python to the rescue!
  - easily write an interface to C++ code in a syntax very similar to python itself.

Outline

Biarri uses python every day

- For our web server
- As a "what if" tool (Turing-complete Excel)
- To talk to our APIs
- To manage our servers

Python is "Batteries Included"

- Libraries to parse and generate
  - csv
  - XML
  - JSON
- Libraries to use WWW technologies
  - HTTP client (urllib2)
  - HTTP server (Pyramid, Django, CherryPy)
  - HTML generation (mako, cheetah, Django)
  - HTML parsing (Beautiful Soup)

Outline

Where X $\in$ {C, C++, Java}

- Python is faster to develop in
  - and faster to learn
- Cheaper to throw more hardware than more dev time
  - A single day of a developer's time is worth over a year of dedicated use of one of our 32GB RAM, 8-core servers' time

Where X $\in$ {Matlab, Mathematica}

These are pretty similar choices, but I'm pretty sure there is no "Web Server" toolbox

- Python has a huge community, and has been used in mission critical apps for decades
- It's open source "Many eyes, shallow bugs"
- pip and pypi (Python Package Index)
  - Hundreds of thousands of libraries are one 'pip install' away
- Someone, somewhere has integrated python with whatever you are using today

Where X $\in$ {Ruby, high-level scripting languages}

These are as much personal choice as anything else, but:

- Generators make math in python pretty

Outline

Questions

- Questions?