# Coding competitions with PyPy aka "Python for the win!"

Alessandro Amici <a.amici@bopen.eu>

B-Open Solutions – http://bopen.eu

# Python wins competitions already!

Google Code Jam 2011 – Round 3 – user: linguo

## Language Popularity

| | Problem A | | Problem B | | Problem C | | Problem D | | Totals | |
|---|---|---|---|---|---|---|---|---|---|---|
| Language | S | L | S | L | S | L | S | L | Sets | People ▲ |
| C++ | 273 | 256 | 264 | 214 | 162 | 68 | 244 | | 1481 | 317 / 19 |
| Java | 48 | 48 | 43 | 31 | 32 | 16 | 45 | | 263 | 54 / 6 |
| Python | 16 | 15 | 9 | 8 | 4 | 2 | 6 | 1 | 61 | 17 / 1 |
| C# | 9 | 9 | 8 | 6 | 5 | 3 | 9 | | 49 | 13 |

Google Code Jam 2013 – Round 2 – user: bmerry

## Language Popularity

| | Problem A | | Problem B | | Problem C | | Problem D | | Totals | |
|---|---|---|---|---|---|---|---|---|---|---|
| Language | S | L | S | L | S | L | S | L | Sets | People ▲ |
| C++ | 1130 | 617 | 841 | 673 | 298 | 153 | | | 3712 | 1315 / 393 |
| Java | 202 | 87 | 134 | 116 | 38 | 16 | | | 593 | 233 / 61 |
| Python | 142 | 70 | 113 | 99 | 10 | 6 | 1 | 1 | 442 | 197 / 45 |
| C# | 29 | 8 | 18 | 17 | 6 | 1 | | | 79 | 37 / 3 |

# Coding Competitions 101

"The aim is to write source code of computer programs which are able to solve given problems. Most problems appearing in programming contests are mathematical or logical in nature." (Wikipedia)

- Short competions – few hours:
  - International Olympiad in Informatics, ACM International Collegiate Programming Contest, Google Code Jam, Facebook Haker Cup, TopCoder Algorithm Open
  - TopCoder SRM, Sphere Online Judge, Codeforces
- Long competions – from a few days to several months:
  - TopCoder Marathon Match, Kaggle, Google AI Challenge, Al Zimmermann's Programming Contests

# Coding Competitions 101

What does it look like?

- Google Code Jam – 2 ½ hours, ranking by score and time
  - Problem statement including limits with test input dataset and output
  - Model, code, test, debug, tune for performance...
  - Download the input dataset and start the clock
  - Run your code on your computer and get an output
  - Upload the output within 4 minutes
  - The online judge declares it correct or incorrect
  - Score points or try again with a different dataset

# Coding Competitions 201

Contraints and assets

- Constraints → Execution time and used memory
  - CPU (speed and cores), RAM (size), Storage (size and speed)
- Assets → Time
  - Modeling time
  - Coding time
  - Testing time
  - Debugging time
  - Performance-tuning time

# Problem: find all prime numbers up to `n`

Definition: A prime number is a positive integer which has exactly two distinct positive integer divisors: 1 and itself.

- Trial division with all numbers – one-liner good up to `n`=100.000:
  - `P = [p for p in range(2,n+1) if all(p%i!=0 for i in range(2,p))]`
- Trial division with primes
- Sieve of Eratosthenes

> “Sift the Two's and Sift the Three's,
> The Sieve of Eratosthenes.
> When the multiples sublime,
> The numbers that remain are Prime.”
>
> Anonymous

# Problem: find all prime numbers up to `n`

Plain Python implementation without any performance tuning:

```
1  def primes_loop(n):
2      P = range(n + 1)
3      P[1] = 0
4      for p in range(2, int(n ** 0.5) + 1):
5          if P[p]:
6              for m in range(p * p, n + 1, p):
7                  P[m] = 0
8      return [p for p in P if p]
```

Plain Python implementation with some performance tuning:

```
1  def primes_assign_list(n):
2      P = range(n + 1)
3      P[1] = 0
4      for p in range(2, int(n ** 0.5) + 1):
5          if P[p]:
6              P[p * p::p] = [0] * (1 + ((n - p * p) // p))
7      return [p for p in P if p]
```

# Problem: find all prime numbers up to `n`

Numpy Python implementation that returns `int`'s:

```
1  def primes_numpy(n):
2      P = np.arange(n + 1, dtype='uint32')
3      P[1] = 0
4      for p in range(2, int(n ** 0.5) + 1):
5          if P[p]:
6              P[p * p::p] = 0
7      return [int(p) for p in P if p]
```

Numpy Python implementation that returns `numpy.uint32`'s:

```
1  def primes_all_numpy(n):
2      P = np.arange(n + 1, dtype='uint32')
3      P[1] = 0
4      for p in range(2, int(n ** 0.5) + 1):
5          if P[p]:
6              P[p * p::p] = 0
7      return P[P > 0]
```

# When speed matters

PyPy

| | max | RAM | CPU |
|---|---|---|---|
| primes_loop | 56 M | 620 Mb | 2.3 s |
| primes_assign_list | 56 M | 890 Mb | 2.5 s |
| primes_numpy | 56 M | 290 Mb | 2.4 s |
| primes_all_numpy | 56 M | 300 Mb | 2.5 s |

CPython

| | max | RAM | CPU |
|---|---|---|---|
| primes_loop | 56 M | 2.6 Gb | 18 s |
| primes_assign_list | 56 M | 2.2 Gb | 6.8 s |
| primes_numpy | 56 M | 330 Mb | 6.8 s |
| primes_all_numpy | 56 M | 290 Mb | 1.4 s |

# When size matters

PyPy

| | max | RAM | CPU |
|---|---|---|---|
| primes_loop | 320 M | 2.9 Gb | 13 s |
| primes_assign_list | 320 M | 3.3 Gb | 33 s |
| primes_numpy | 560 M | 2.6 Gb | 25 s |
| primes_all_numpy | 560 M | 2.8 Gb | 26 s |

CPython

| | max | RAM | CPU |
|---|---|---|---|
| primes_loop | 56 M | 2.6 Gb | 18 s |
| primes_assign_list | 100 M | 3.4 Gb | 38 s |
| primes_numpy | 560 M | 3.1 Gb | 68 s |
| primes_all_numpy | 560 M | 2.8 Gb | 15 s |

# Programming languages guide

Programming languages strenghts and weaknesses

- C++ with Standard Template Library → compiled
  - Modeling, Coding, Testing, Performance-tuning → Good
  - Debugging → Horrible
  - Speed, Memory → Excellent
- Java → interpreted with a JIT
  - Modeling, Performance-tuning → Good
  - Coding, Testing, Debugging → Well... it's Java :-P
  - Speed, Memory → Good

# Programming languages guide

Programming languages strenghts and weaknesses

- Python without performance-tuning → interpreted without a JIT
  - Modeling, Coding, Testing, Debugging → Excellent
  - Speed, Memory → Horrible
- Python with performance-tuning → interpreted without a JIT
  - Modeling, Coding, Testing, Debugging → Form Bad to Good
  - Speed, Memory → From Bad to Good
- PyPy → interpreted with a JIT
  - Modeling, Coding, Testing, Debugging → Excellent
  - Performance-tuning → Good
  - Speed, Memory → Good

# GCJ 2012 – Round 1A – Problem B. Kingdom Rush

Problem statement: http://goo.gl/DZRMD and analysis: http://goo.gl/ptgyT

```
from sys import stdin
for t in xrange(1, int(stdin.next().strip())+1):
    N, S = int(stdin.next()), [map(int, stdin.next().split()) for i in xrange(N)]
    r, st = 0, 0
    while len(S) > 0:
        T = [s for s in S if s[1]<=st]
        if len(T) > 0:
            S = [s for s in S if s[1]>st]
            r += len(T)
            st += 2 * len(T) - sum(t[0] > 2001 for t in T)
            continue
        A = sorted([s for s in S if s[0]<=st], lambda x,y: y[1]-x[1])
        if len(A) == 0:
            r = 'Too Bad'
            break
        r += 1
        st += 1
        A[0][0] = 2002
    print 'Case #%d: %s' % (t, r)
```

5x faster with PyPy

# Al Zimmerman's Programming Contests Factorials

Problem statement: http://www.azspcs.net/Contest/Factorials

```
[...]
while len(slp):
        current = slp[-1]
        for i, other in enumerate(slp):
            for j in [0, 1, 2]:
                if j==0:
                    if i==0: continue
                    next = current * other
                elif j==1:
                    if i==len(slp)-1: continue
                    next = current + other
                elif j==2:
                    if i==len(slp)-1: continue
                    next = max(current, other) - min(current, other)
                if lcm % next != 0:
                    continue
                if next in slp or next in nexts[-1]:
                    continue
                if next < current and test_slp_next(slp[:-1], next):
                    continue
[...]
```

5x faster with PyPy

# GCJ 2012 – Round 1B
# Problem B. Tide Goes In, Tide Goes Out

Problem statement: http://goo.gl/7qRzA and analysis: http://goo.gl/JEcSM

```
from sys import stdin
import heapq as hp
for tc in range(1, int(stdin.next())+1):
    H, N, M = map(int, stdin.next().split())
    CH = [map(int, stdin.next().split()) for i in range(N)]
    CL = [map(int, stdin.next().split()) for i in range(N)]
    T = [[2**31]*M for i in xrange(N)]
    T[0][0] = 0.
    F = [(T[0][0], 0, 0)]
    while len(F):
        t, j, i = hp.heappop(F)
        if j==N-1 and i==M-1:
            break
        for jj, ii in [(j-1,i), (j,i-1), (j+1,i), (j,i+1)]:
            if not (0<=jj<N and 0<=ii<M):
                continue
            if min(CH[j][i],CH[jj][ii]) - max(CL[j][i],CL[jj][ii]) < 50:
                continue
            ts = max(t, (H + 50 - CH[jj][ii])/10.)
            if ts > 0.:
                ts += 1. if (H-10*ts-CL[j][i]) >= 20 else 10.
            if ts < T[jj][ii]:
                T[jj][ii] = ts
                hp.heappush(F, (ts, jj, ii))
    print 'Case #%s: %s' % (tc, T[-1][-1])
```

2x SLOWER with PyPy

# PyPy competition setup

Setup a clean virtualenv with the latest PyPy realease with:

- IPython – http://ipython.org/
- PyFlake – https://pypi.python.org/pypi/pyflakes
- NumPyPy – in the PyPy distribution
- NetworkX – Graph library – http://networkx.github.io/
- PIL – Image processing – http://www.pythonware.com/products/pil/

Own library of algorithms:

- PriorityDictionary – partially ordered dict – http://goo.gl/aWg6r
- Dijkstra Shortest Path Algorithm – http://goo.gl/pQaLo
- GCD, LCM, binom, isqrt, etc...

# PyPy competition limitations

A few libraries are not ported to CFFI yet:

- NumPy – http://www.numpy.org/ (NumPyPy is a partial reimplementation)
- SciPy – Scientific computing library – http://www.scipy.org/
- Gmpy2 – Numerical library – http://code.google.com/p/gmpy/

Small tasks don't perform well:

- Fast tasks suffer from the warm-up slowdown
- Small memory tasks suffer the bigger memory footprint of PyPy

# Take away lessons

PyPy advantages over Python

- Modeling time
  - can code at low level when needed, much like C++
- Coding time and Testing time
  - simple code usually runs fast enough
- Performance-tuning time
  - can skip several optimization techniques
  - usually good speed and memory performance for heavy tasks
- Debugging time
  - simple code + less optimization == easier debugging

# Thanks.

Alessandro Amici <a.amici@bopen.eu>

http://linkedin.com/in/alexamici

search me on Google+

B-Open Solutions – http://bopen.eu