Using PyPy

Two Ways of Using PyPy

- You can build a binary
  - This gives you much less flexibility, and isn't necessarily faster
  - This is mostly useful for creating your own language interpreter
- You can interpret your Python code
  - This is what most people will want to do
  - For pure Python, It's pretty much just a matter of changing your #!

You Can Build a Binary

- This involves RPython
- I'm showing you this mostly to convince you that it's normally not what you want
- Check out PyPy
  - hg clone https://bitbucket.org/pypy/pypy src
- Goals
  - cd src/pypy/translator/goal
  - Select or create a goal, EG pypy or nop (hello world)

Goals

- Building a PyPy binary from the RPython code:
  - python translate.py --opt=jit targetpypystandalone.py
- Building a Hello World from the RPython code:
  - python translate.py targetnopstandalone.py
  - Gives a 171K hello world (targetnopstandalone-c) program
- I briefly tried creating a targetsievestandalone.py, but it (RPython) disliked my use of generators, so I moved on

What “nop” looks like

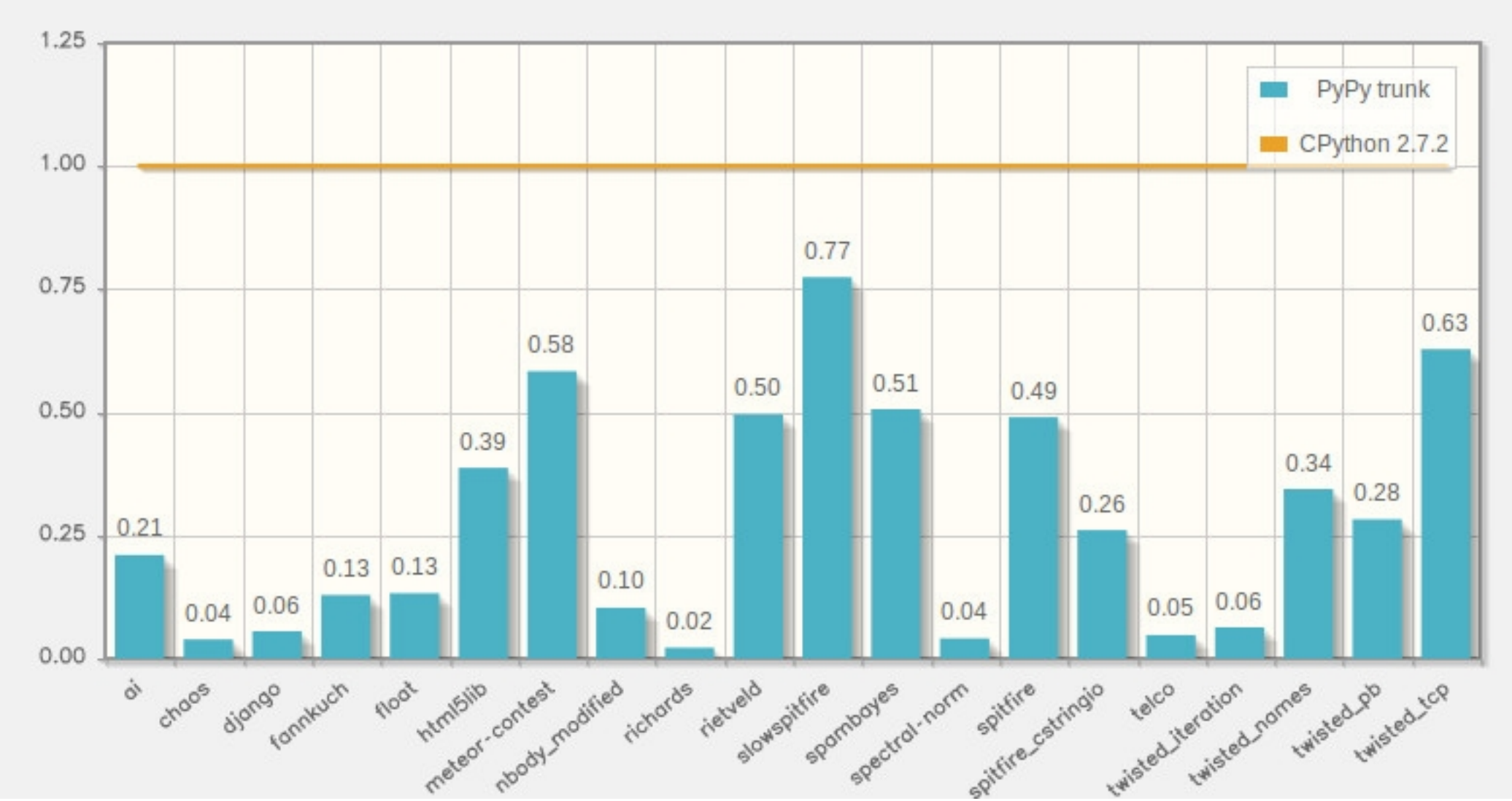
```
def debug(msg):
    print "debug:", msg

# ___________  Entry point  ___________

def entry_point(argv):
    debug("hello world")
    return 0

# _____ Define and setup target ___

def target(*args):
    return entry_point, None
```

You Can Interpret Your Python Code

- This involves full-fledged Python 2.7; no RPython required.
- For most pure Python code just change the #! line to PyPy (*ix)
- Running sieve for primes below 250,000,000 on a 32 bit Linux Mint 14 system:

Interpreter	Duration (low is good)
Pypy 1.9	3m30.553s
Jython 2.5.3	Memory error
CPython 2.5.6	12m33.746s
CPython 2.7.2	12m58.738s
CPython 3.0.1	16m31.780s
CPython 3.3.0	14m22.298s

The PyPy project's idea of their own speed



But I really want to use RPython

- Some reasons not to:
  - Generators get less flexible
  - If you change one module, you have to rebuild your whole project, which can take a while for large projects
  - Implicitly statically typed data
  - You get the speed benefit even without using RPython

What Are PyPy (RPython) Build Times?

Goal	Time
PyPy	145m34.050s
nop	0m27.181s

Barriers to use of PyPy

- C Extension Modules are the single biggest barrier.
  - It usually works best to rewrite your C Extension Modules as Pure Python for PyPy's benefit, so the PyPy JIT can optimize them. You can also preprocess Cython!
  - PyPy 1.9/PyPy 2.0 Beta 1 (current at the time of this writing) and below have Beta support for C extension modules, but they tend to be slow
  - You can write fast C interfacing code using ctypes or cffi

For More Information

- http://doc.pypy.org/en/latest/faq.html
- http://morepypy.blogspot.com/2011/04/tutorial-writing-interpreter-with-pypy.html