

An introduction to Python programming

in 30 minutes...?!

Matt J Williams

FTS Seminar
27 Jan 2010

The aim of this talk...

- A whistle-stop tour of Python
- Introduce basic syntax
- Demonstrate some of the neat features of Python
 - ...but don't worry about learning the exact syntax!
- Highlight Python's strengths and whether you may want to use it
(...or may not want to use it!)

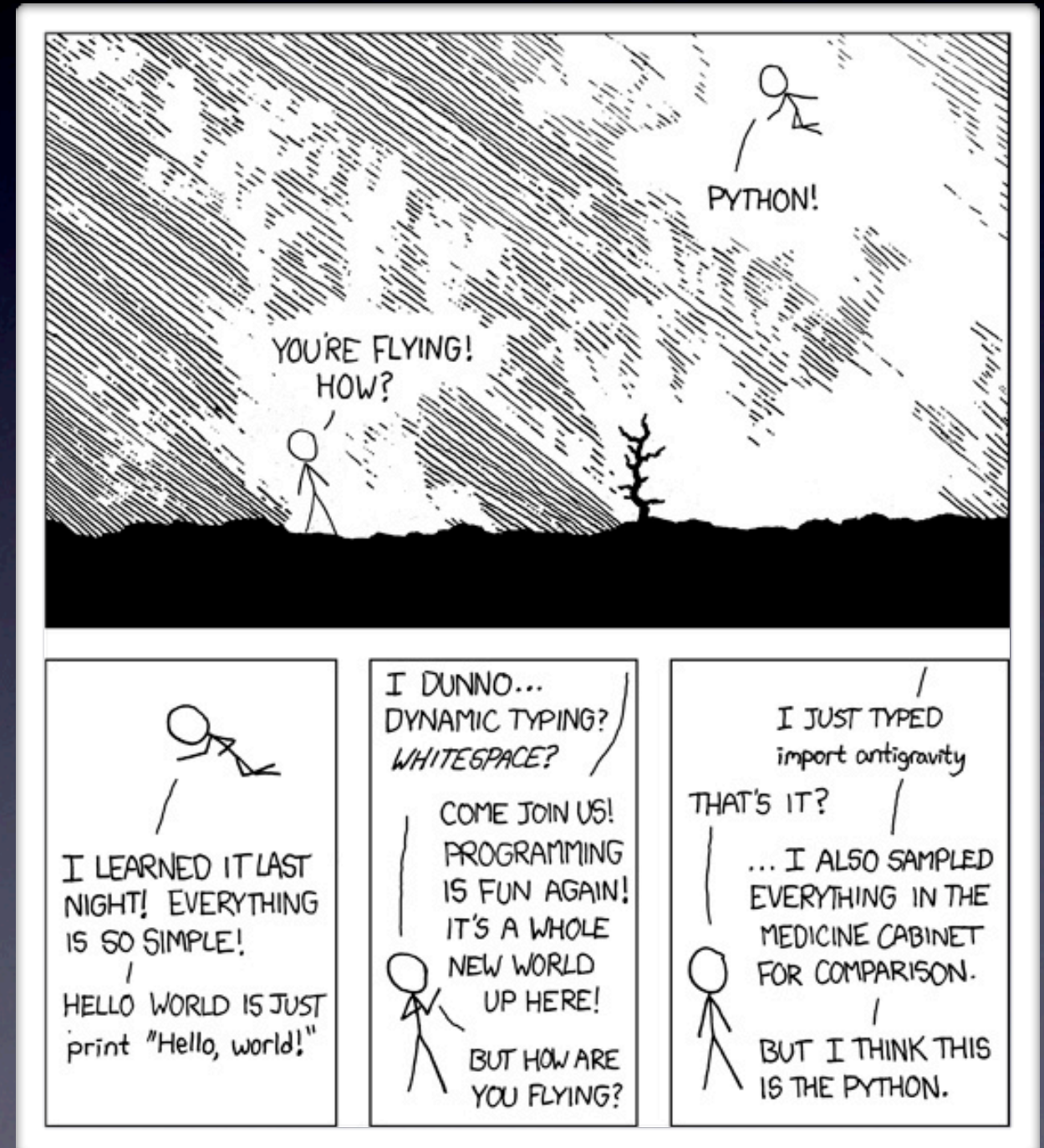
Overview

- Python background
 - History, popularity, current status
- A look at Python
 - Syntax
 - Features
 - Python for numerical computing
- Other Python features
- Python performance issues
- Pros/Cons

Background

Q: What is Python?

- Answer:
 - YAPL -- Yet Another Programming Language
- Fundamental features:
 - Interpreted
 - Dynamically typed
 - Automatic memory management
 - Departs from usual C-like syntax
 - Emphasises **productivity** and **readability**



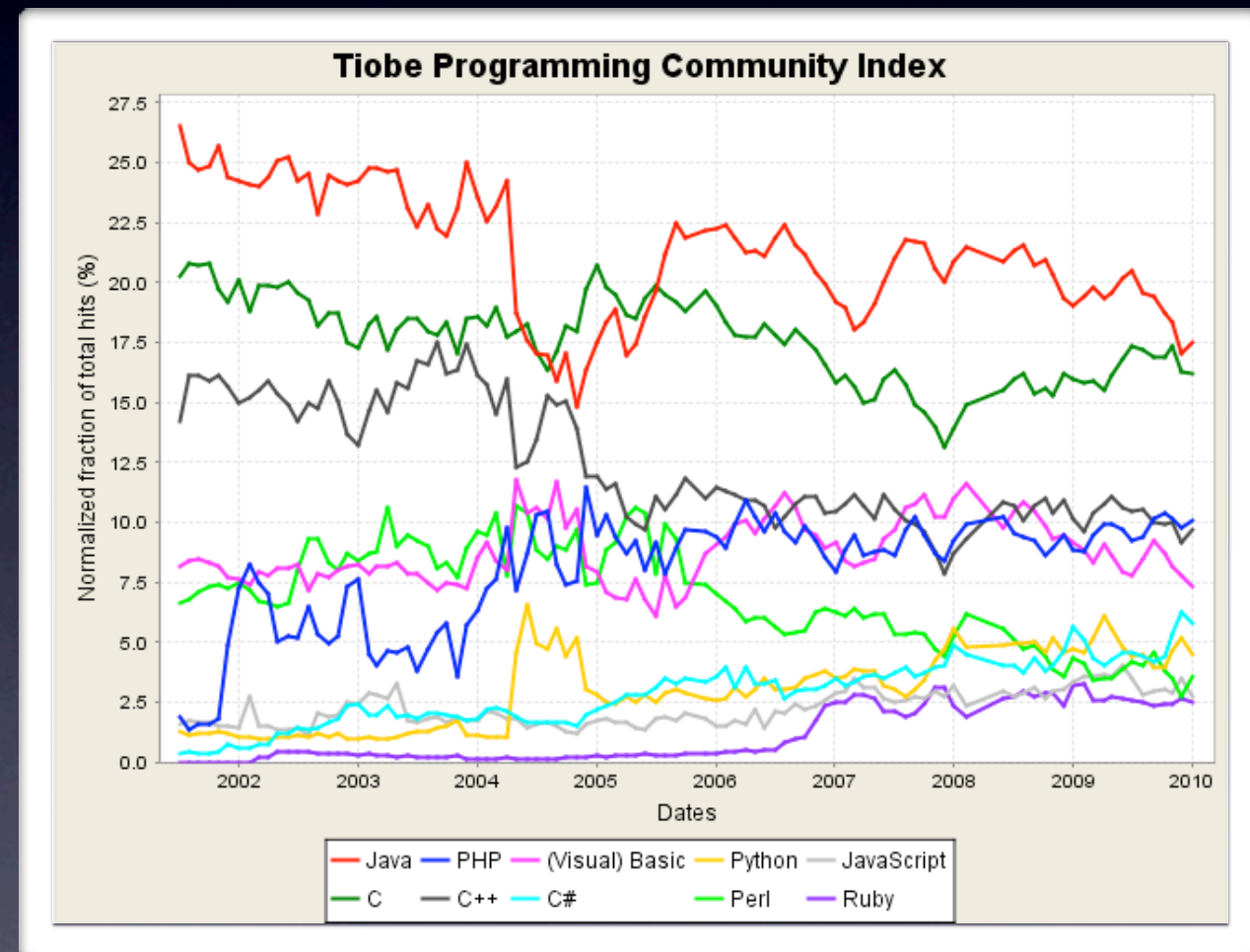
Some history...

- Conceived at the end of the 80s, initial release in 1990
 - (It's roughly the same age as Java!)
- Created by **Guido Van Rossum**
- A big fan of Monty Python's Flying Circus
- Now employed by Google
- “...I get to spend half my time on Python, no strings attached!”



Python these days...

- A very popular scripting language
- Extensively used at Google
- Google's AppEngine supports Java and Python
- Teaching programming



Python's Strengths

- Elegant, intuitive, easy to learn syntax
 - ...encourages **readability**
 - ...and **productivity**
- Integration with other languages
(Possibly the best language for this!)
- Powerful constructs
 - Do a lot with few symbols

A look at Python...

Running Python Programs



- The **Python interpreter** handles the work of running a **Python script**
- No explicit compilation
- Two ways of running Python code:
 - **From file:** Python reads code from a file (e.g. `myprog.py`) and executes it
 - **Command line interpreter:** The interpreter can be run as an interactive shell

Syntax Comparison

Java

```
public static double sumSquares( double[] nums )
{
    double total = 0.0;
    for( int i=0; i < nums.length; i++ )
    {
        double squared = nums[i] * nums[i];
        total += squared;
    }
    return total;
}
```

no for-each on Java primitives :-)

Python

```
def sum_squares( nums ):
    total = 0.0
    for val in nums:
        squared = val**2
        total += squared
    return total
```

└ to indicate tabs

- No curly braces
- No semicolons
- No arrays
- No typedefs

[Python Demo]

assignment	numeric operators mod, integer division, pow	dynamic typing	int, float, complex	bool
chained comparisons	strings 'x' "x" """x"""	sprintf syntax	collections: tuples, lists	collection indexing
functions, first-class objects	list comprehensions	lambda, map	dictionaries	loops
docstrings	introspection			

Numerical Computing in Python

- Mathematical packages
 - NumPy
Arrays, matrices, and matrix operators
 - SciPy
More functionality: optimisation, signal processing, stats, ...
 - Matplotlib

Sophisticated plotting tools (MATLAB-like)
- Performance
 - Very close to MATLAB (also interpreted)
 - With some tweaking, can get close to C++

Language	Notes	Time taken (sec)
Python	Pure Python: no SciPy or NumPy.	1500.0
Python	Pure Python with Psyco (a JIT compiler).	1138.0
Octave	<i>Estimate.</i>	60.0
Python	Python + NumPy: Using NumPy arrays instead of vanilla Python.	29.3
MATLAB	<i>Estimate.</i>	29.0
Python	Python + NumPy Blitz: NumPy operations auto-converted to C++ code.	9.5
Python	Python + NumPy Inline: Embed C++ in Python.	4.3
C++	Pure C++	2.16

[Python+Matplotlib Demo]

Scalability and Performance

(The bad news)

- Vanilla Python can never match the performance of compiled languages
- The default Python interpreter (CPython) can go wonky on multi-core systems without special care
- Be pragmatic about Python's scalability and performance
 - “Where performance is key, use Java or C++”

Things I Haven't Demonstrated...

- Language integration
 - R, AppleScript, C++, ...
- Classes
 - Supports multiple inheritance
 - No strict private vs. public enforcement
- Exception throwing and handling
- Web development
- Multi-threading, multi-processing
- Python3 vs Python2
- Object serialisation and storage
 - Really easy!

Conclusions

- Pros:
 - Quick to learn
 - Promotes good practices
 - Has great scientific support
 - Quick to write -- elegant, intuitive syntax
 - Integration with other languages
- Cons:
 - Performance, concurrency
 - No compile-time error checking
- Productivity vs. performance
- Interoperability vs. performance

Links

- Python is available for Windows, Linux, and OS X
 - <http://www.python.org/download/>
- There is also a 3 lecture introduction to Python on iTunesU
 - Look for:
Stanford “Programming Paradigms” Lectures
Jerry Cain
Lecture 24 onwards
- Zen of Python
 - <http://www.python.org/dev/peps/pep-0020/>
 - “Beautiful is better than ugly”
“Simple is better than complex”
...

Thanks for Listening



Any questions?