

Python for Earth Scientists

Andrew Walker

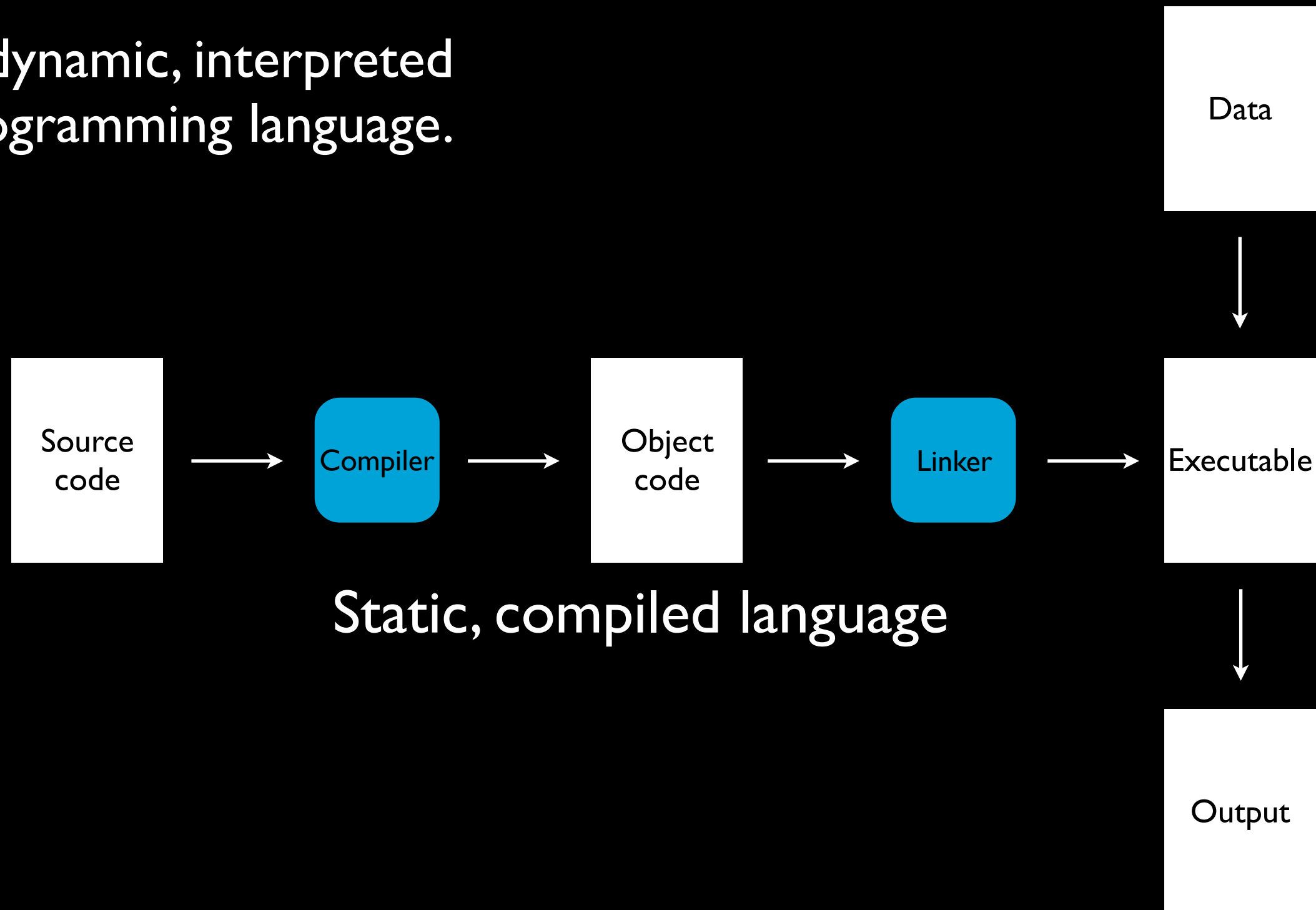
andrew.walker@bris.ac.uk

Python is:

A dynamic, interpreted
programming language.

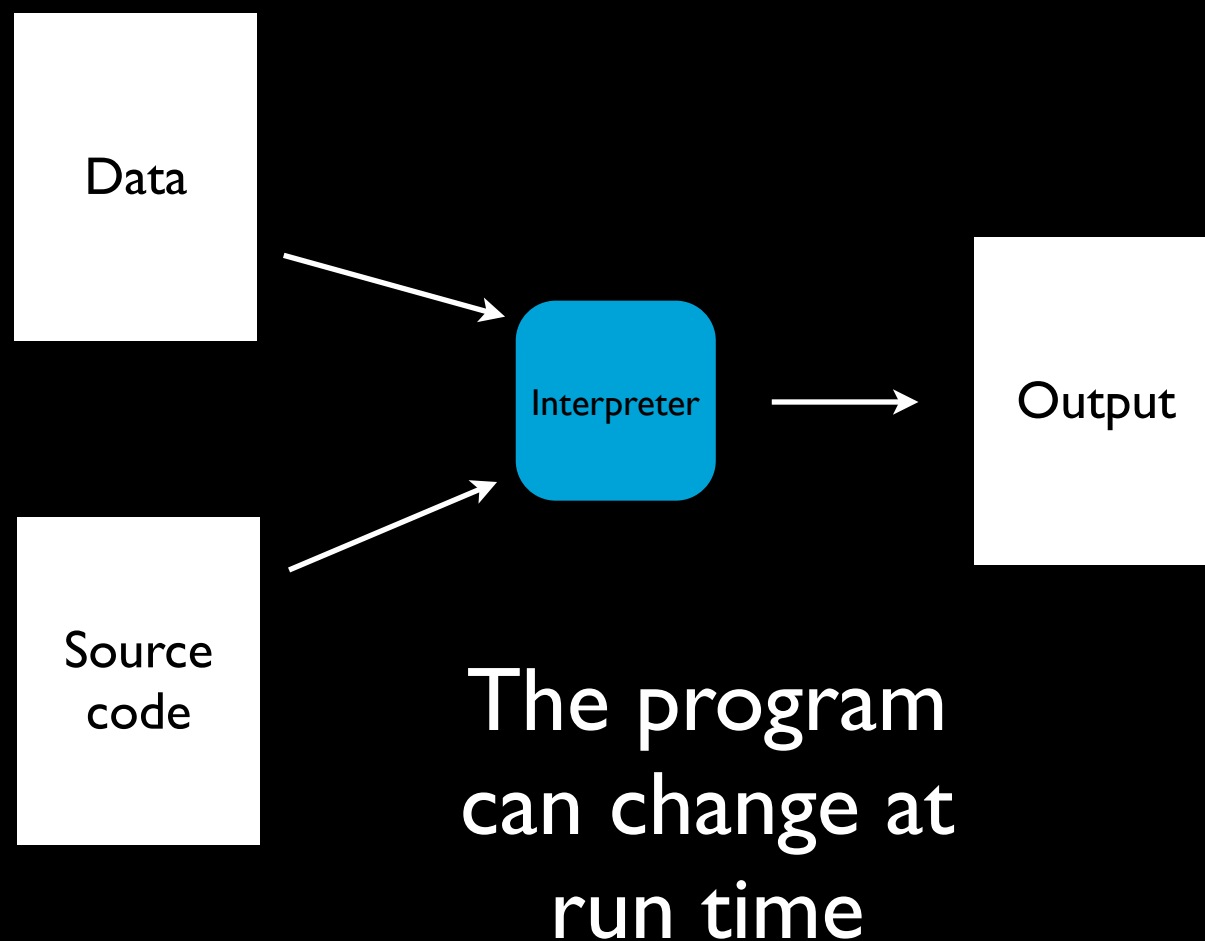
Python is:

A dynamic, interpreted programming language.



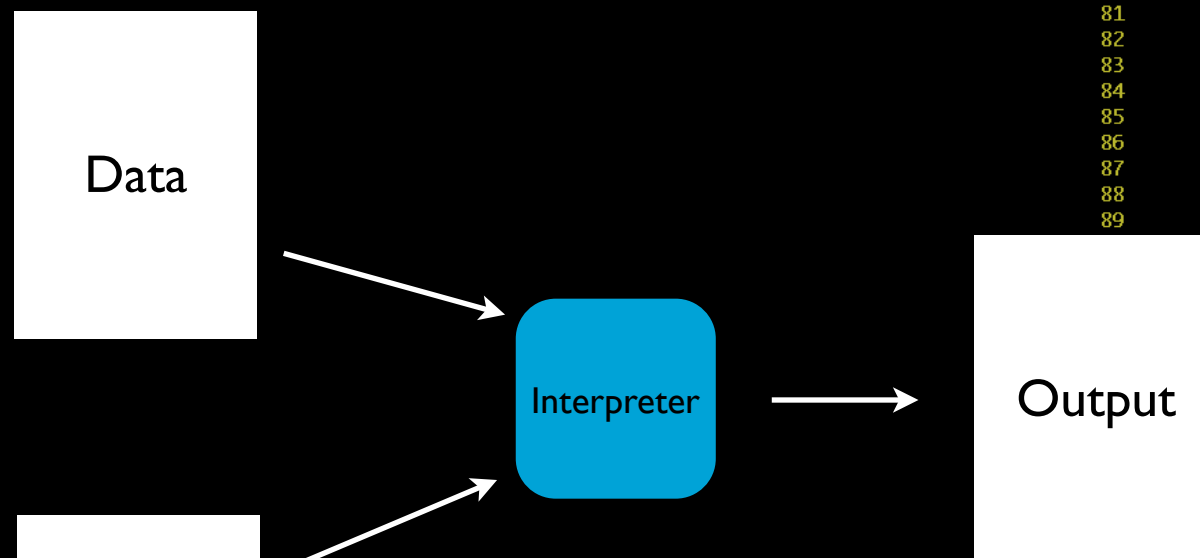
Python is:

A dynamic, interpreted programming language.



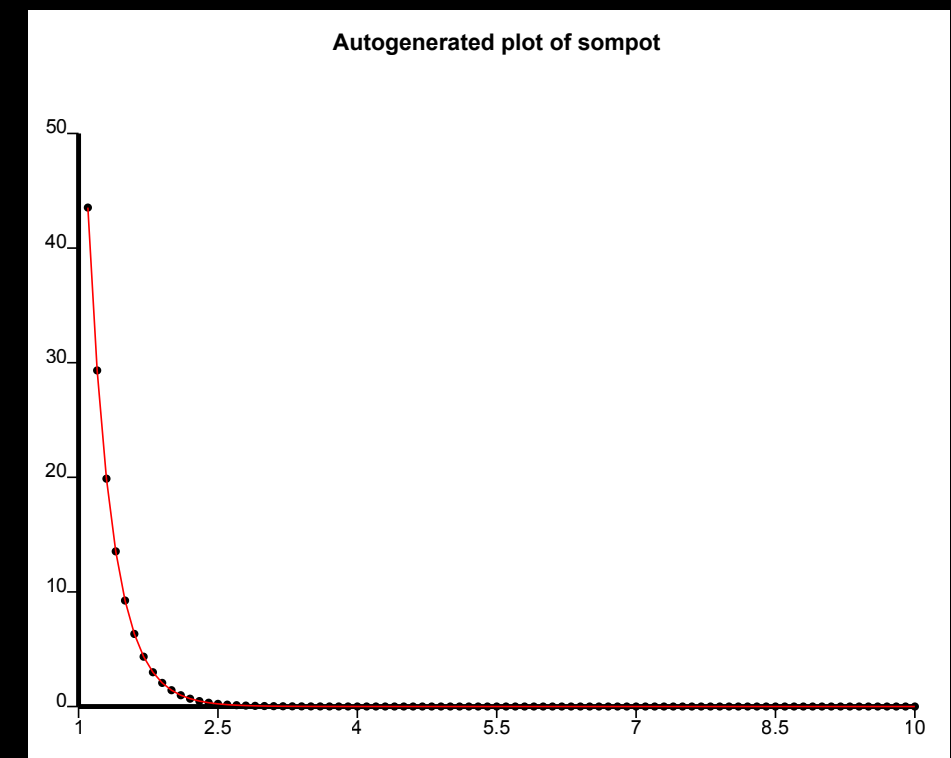
Python is:

A dynamic, interpreted programming language.



The program
can change at
run time

```
50 <parameter name="a" dictRef="gulp:buckingham.a">
51   <scalar units="gulpunits:eV">2.409505000000e3</scalar>
52 </parameter>
53 <parameter name="roh" dictRef="gulp:buckingham.roh">
54   <scalar units="gulpunits:Ang^-1">2.649000000000e-1</scalar>
55 </parameter>
56 <parameter name="c" dictRef="gulp:buckingham.c">
57 <!-- Modified from 0 to 10 to show how it changes graph -->
58   <scalar units="gulpunits:eV^-6">-10.000000000000e0</scalar>
59 </parameter>
60 <!-- The chunk of stuff within the expression will be the same for all
61 instances of this type of potential - so this can be included with
62 Xinclude. The idea is to provide enough info (with a dictRef) so that
63 e.g. Fortran codes can use the potential if they are not network enabled) -->
64 <expression dictRef="mypotentialDict:buckingham">
65   <scalar units="eV"/>
66   <m:math>
67     <m:lambda>
68       <m:bvar><m:ci>R</m:ci></m:bvar>
69       <m:apply>
70         <m:minus/>
71         <m:ci>
72           <m:apply>
73             <m:times/>
74             <m:ci>a</m:ci>
75             <m:ci>
76               <m:apply>
77                 <m:exp/>
78                 <m:ci>
79                   <m:apply>
80                     <m:divide/>
81                     <m:ci>
82                       <m:apply>
83                         <m:minus/>
84                         <m:ci>R</m:ci>
85                       </m:apply>
86                     </m:ci>
87                     <m:ci>roh</m:ci>
88                   </m:apply>
89                 </m:ci>
90               </m:apply>
91             </m:ci>
92           </m:apply>
93         </m:ci>
94       </m:apply>
95     </m:math>
96   </expression>
```



Python is:

A dynamic, interpreted
programming language...

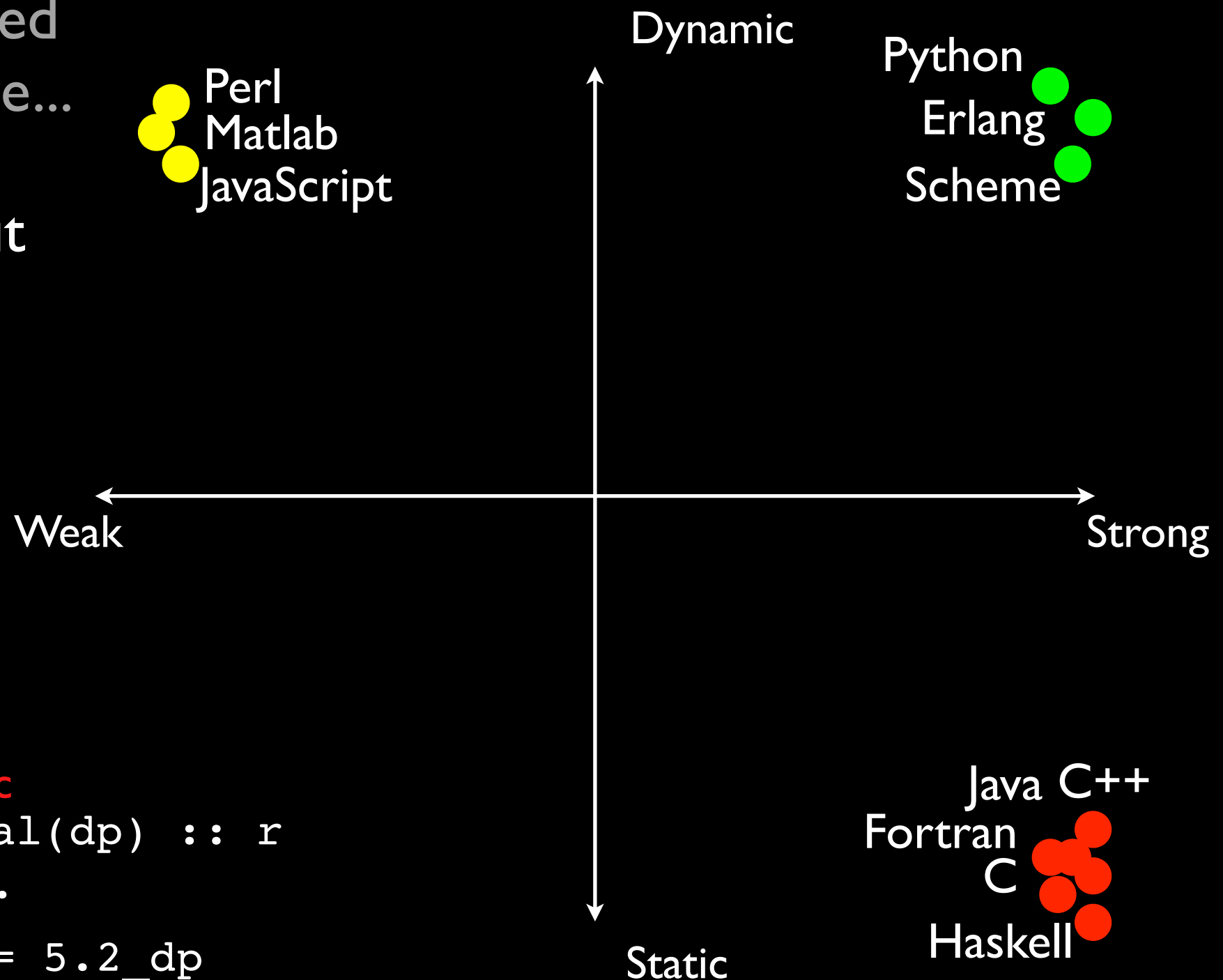
... strongly typed but
dynamically typed.

Python is:

A dynamic, interpreted programming language...

...strongly typed but dynamically typed.

Some typing styles



Weak

```
x = 1 + "2.0"
```

Dynamic

```
a = 5
```

```
...
```

```
a = "cat"
```

Static

```
real(dp) :: r
```

```
...
```

```
r = 5.2_dp
```

Python is:

A dynamic, interpreted
programming language...

... strongly typed but
dynamically typed ...

... imperative and object
oriented.

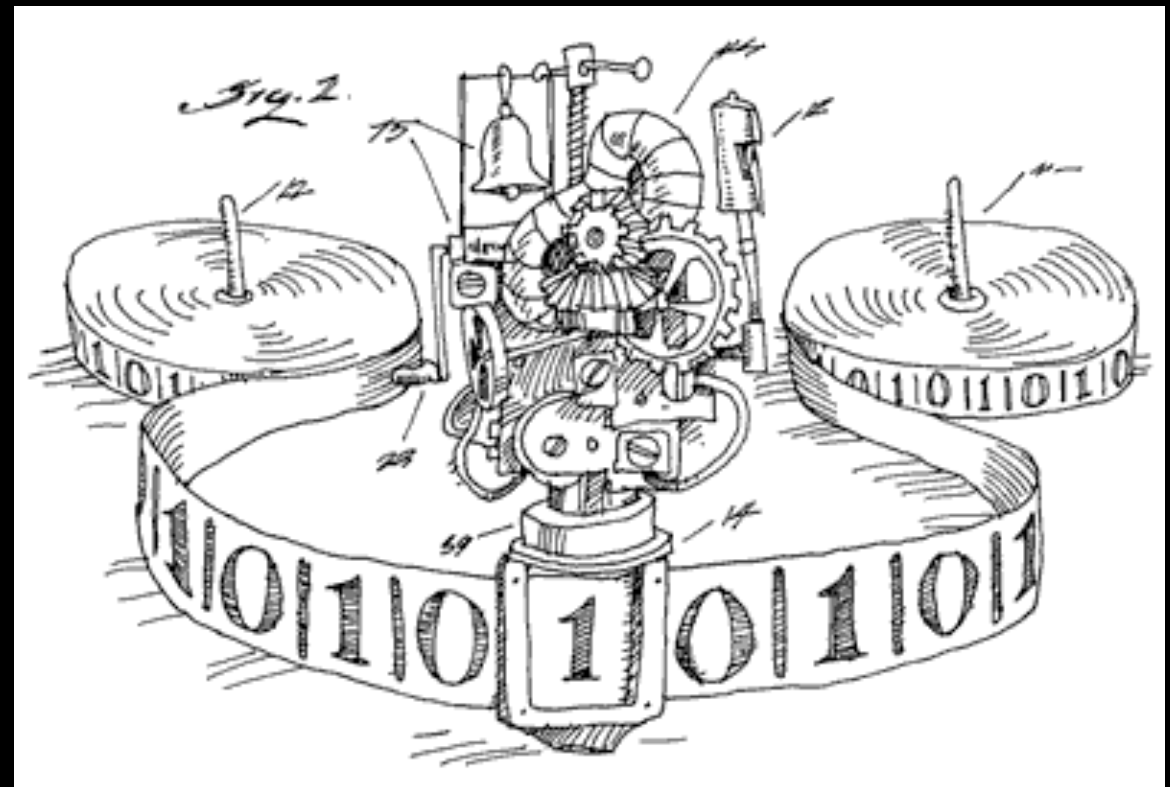
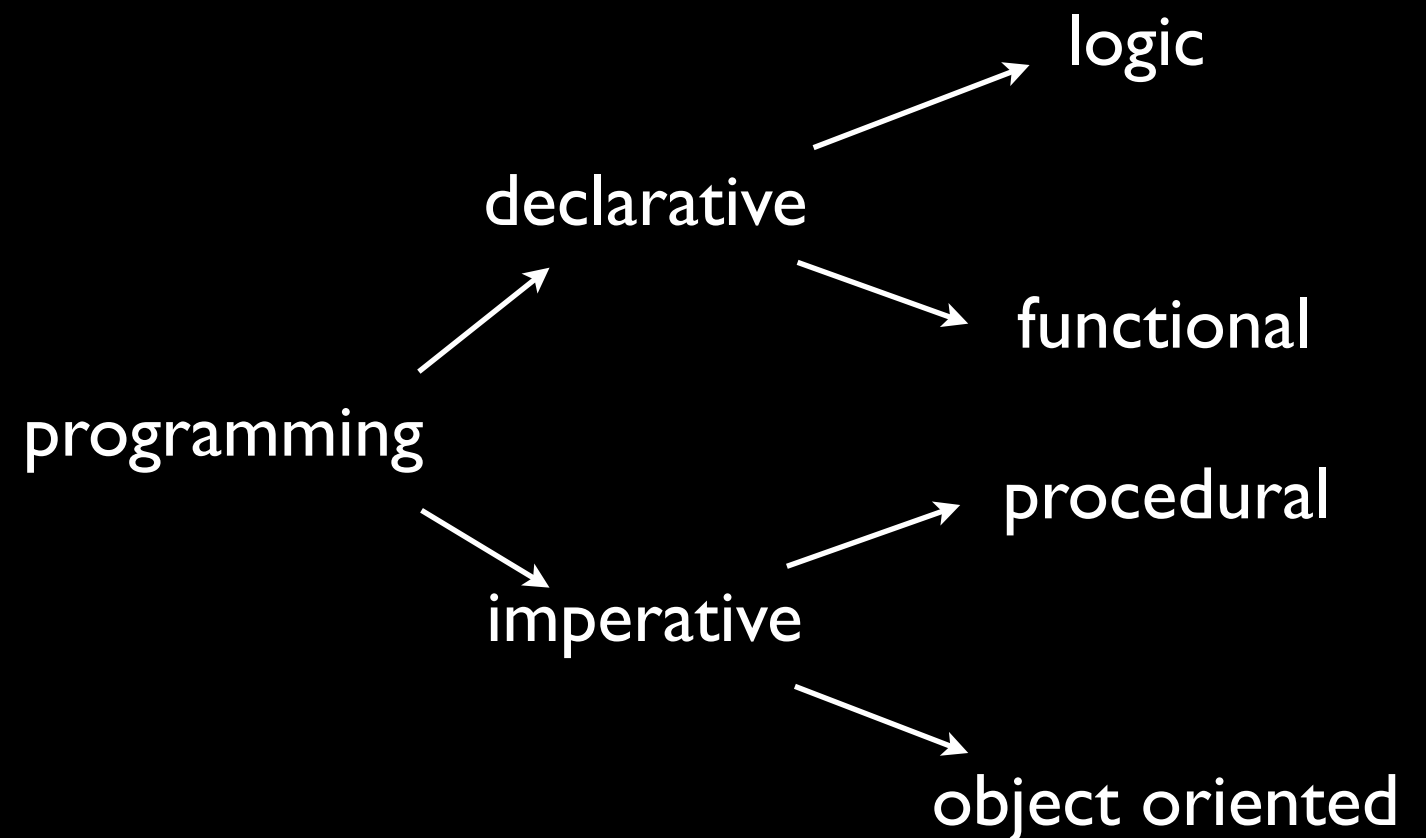
Python is:

A dynamic, interpreted programming language...

... strongly typed but dynamically typed ...

... imperative and object oriented.

(Can look procedural)



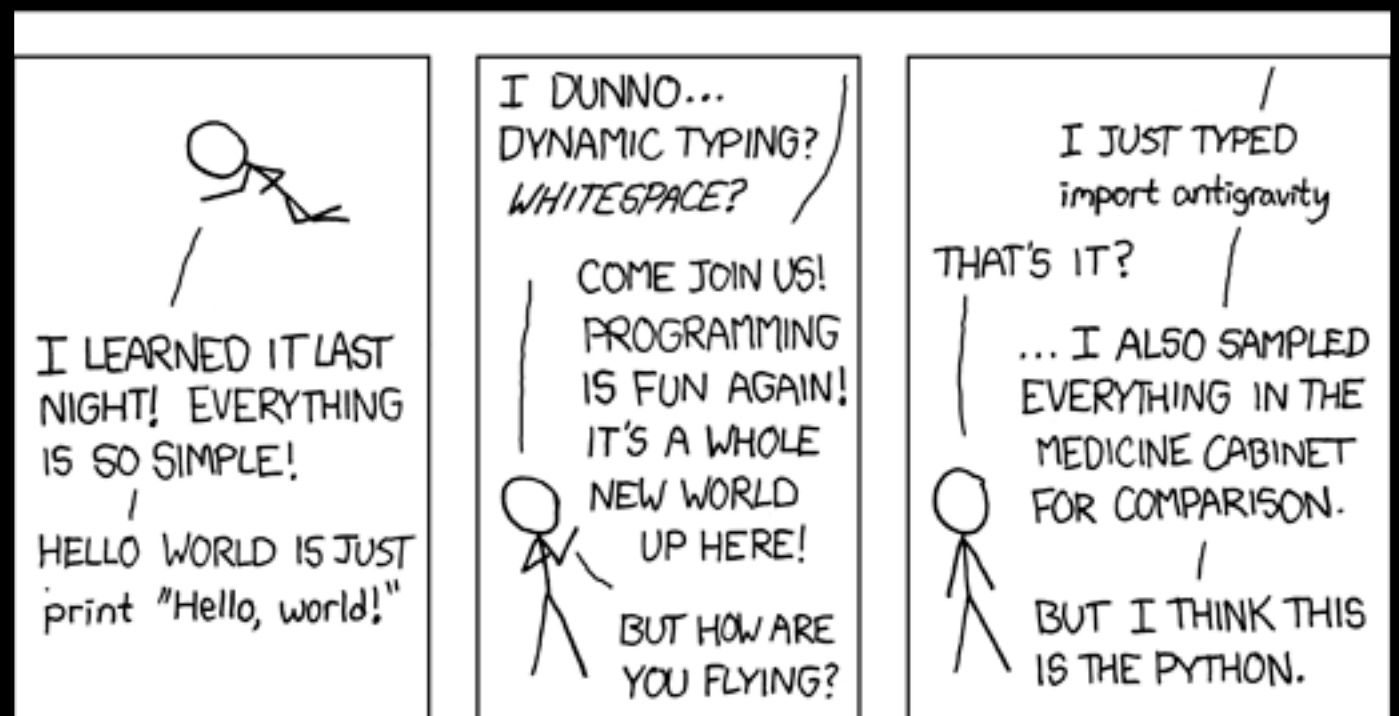
Python is:

A dynamic, interpreted
programming language...

... strongly typed but
dynamically typed ...

... imperative and object
oriented ...

... fun to learn and use.



<http://xkcd.com/353/>

Portable

Good as 'glue'

Open

Multi-paradigm

Modern

Testable

Designed



Opinionated

Comes with
batteries included

Evolving

Encourages rapid
(agile) development

Free

Today

- Now. I'll talk for ~30 mins on Python's syntax and landscape
- 2 pm. Practical 1: First steps
- 3 pm. I'll say something about types, modules, scripts and the standard library
- 4 pm. Practical 2: Scripts and modules

Thursday

- 1 pm. I'll explain object oriented programming
- 2 pm. Practical 3: Object oriented programming
- 3 pm. I'll introduce SciPy, NumPy and Matplotlib
- 4 pm. Practical 4: NumPy, SciPy and Matplotlib

Syntax

```
print "Hello, world!"
```

Starting with this is compulsory

```
a = 1 # an integer
b = 2.7 # a float
c = "Hello, world" # a string
print a
print b
print c
x1 = 6 # This is OK
1x = 7 # This is an error
```

Variable assignment, comments and three types

Syntax

```
a = 1
b = 10
c = a + b # What is c?
(4 < 5) # True
(4 > 5) # False
((4 < 5) or (4 > 5)) # True
```

Some mathematical and logical operators.

Syntax

```
if sky == 'blue':  
    birdsong = True  
elif sky == 'black':  
    birdsong = False  
else:  
    pass    #do nothing
```

If, elif, else. Note indentation. No case statement.

Syntax

```
a = 0
while (a < 10):
    print a
    a = a + 1
```

While loop

```
for a in range(10):
    print a
```

Iteration

Syntax

```
def add_ten (value):  
    value = value + 10  
    return value
```

Function definition

```
b = add_ten(5)  
print b
```

Function use

Landscape

```
#!/usr/bin/env python
"""
List files in the working
directory

"""
import os

def main():
    print os.getcwd()
    for fn in os.listdir('.'):
        print fn

if __name__ == '__main__':
    main()
```



The interpreter



Language parser and implementation



Host
computer &
operating
system:
access to
disks,
network,
users etc.

Landscape

```
#!/usr/bin/env python
"""
List files in the working
directory

"""
import os
def main():
    print os.getcwd()
    for fn in os.listdir('.'):
        print fn

if __name__ == '__main__':
    main()
```



The interpreter



Language parser and implementation



Host
computer &
operating
system:
access to
disks,
network,
users etc.



User
module

Third
party
module

Standard
library
module

Extension
module
(compiled)



The interpreter



Language parser and implementation

CPython

C

Reference implementation.
Runs on windows, unix, mac etc.
Extension modules are .so/.dll

Jython

Java

Targets Java virtual machine.
Can load Java libraries as extension modules

IronPython

C#

Targets .NET framework. Can integrate with silverlight, visual basic etc.

PyPy

Python

Compiler and implementation.
Includes JIT compiler and can be very fast

Standard library

~300 modules designed to be shipped with interpreter

- Mathematical functions
- Advanced string processing
- Many data types and specialised algorithms
- Threading and multiprocessing
- Access to OS facilities
- Networking: sockets, http, email, ftp
- Data storage, database access
- Compression, encryption, hashing
- File formats, XML
- Graphics


Package index

16725 free to use packages (collections of modules) in a searchable index

Browse

Topic :: Scientific/Engineering [unselect]

Not Logged In

[Login](#)
[Register](#)
[Lost Login?](#)
[Use OpenID](#) 

Packages

There are 1041 packages in the categories you have selected. Please select more categories, or [show all](#) matching packages now.

Topic

[Adaptive Technologies](#) (3) [Artistic Software](#) (10) [Communications](#) (13) [Database](#) (33) [Desktop Environment](#) (6)
[Documentation](#) (4) [Education](#) (43) [Games/Entertainment](#) (14) [Home Automation](#) (4) [Internet](#) (118) [Multimedia](#) (61)
[Office/Business](#) (15) [Other/Nonlisted Topic](#) (3) [Religion](#) (2) [Security](#) (9) [Sociology](#) (4) [Software Development](#) (497)
[System](#) (50) [Terminals](#) (1) [Text Editors](#) (3) [Text Processing](#) (53) [Utilities](#) (89)

Environment

[Console](#) (370) [MacOS X](#) (21) [No Input/Output \(Daemon\)](#) (9) [Other Environment](#) (29) [Plugins](#) (7) [Web Environment](#) (90)
[Win32 \(MS Windows\)](#) (37) [X11 Applications](#) (49)

Framework

[Django](#) (11) [Paste](#) (21) [Plone](#) (43) [Pylons](#) (5) [Twisted](#) (1) [Zope2](#) (3) [Zope3](#) (7)

Development Status

[1 - Planning](#) (12) [2 - Pre-Alpha](#) (42) [3 - Alpha](#) (279) [4 - Beta](#) (389) [5 - Production/Stable](#) (223) [6 - Mature](#) (12)
[7 - Inactive](#) (3)

Intended Audience

[Developers](#) (570) [Education](#) (107) [End Users/Desktop](#) (131) [Financial and Insurance Industry](#) (16) [Healthcare Industry](#) (31)
[Information Technology](#) (58) [Legal Industry](#) (1) [Manufacturing](#) (11) [Other Audience](#) (36) [Religion](#) (3)
[Science/Research](#) (755) [System Administrators](#) (26) [Telecommunications Industry](#) (8)

Distribution

The interpreter

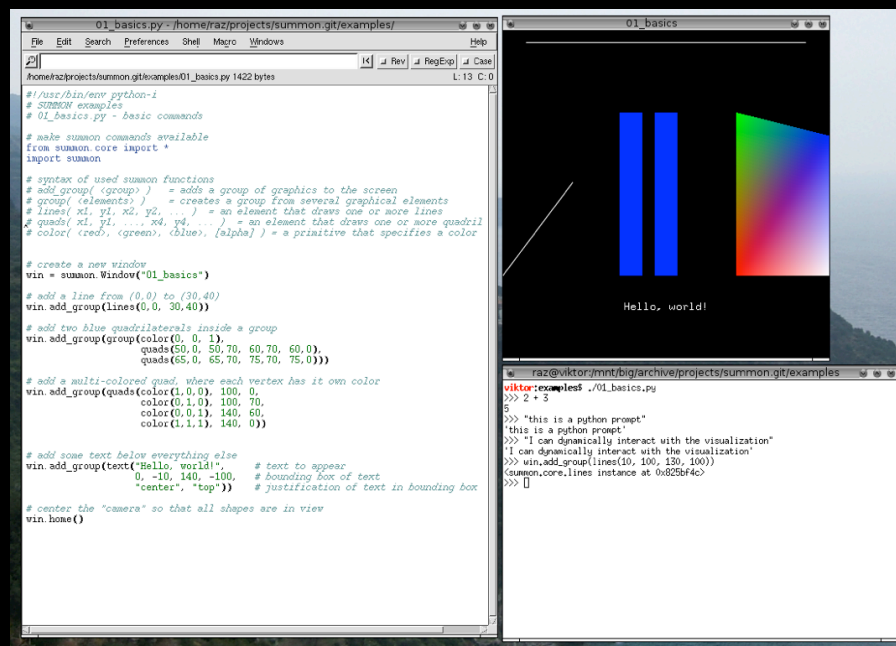


Language parser and implementation

Standard
library
modules

Third
party
modules

Module
management
and upgrade
tool



Integrated development environment

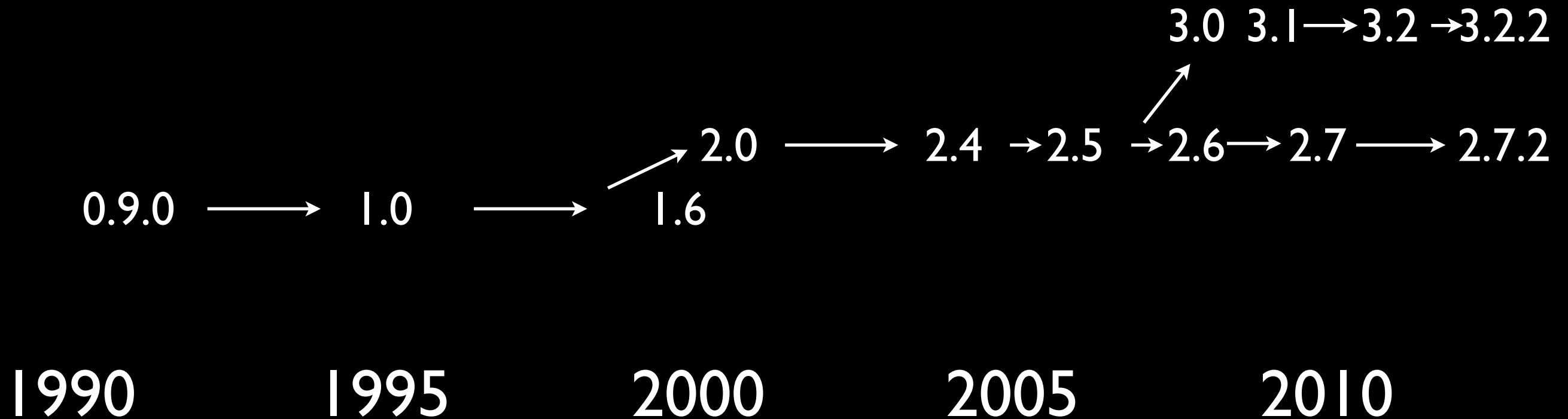
Interpreter ± collection
of modules ± other stuff
= Python distribution

Some distributions

- Official CPython distribution
- Distribution with OS: Mac, Linux
- Commercial: ActivePython, Enthought
- Specialised: stackless, embedded

Versioning

In principle language version is separate from interpreter version and versioning of modules. However, CPython and Python stay in lock-step



Development

Interpreter developed in a similar way to other large pieces of open source code (distributed version control, bug tracker, etc.)

Language developed by the adoption of Python Enhancement Proposals (PEPs) - open process but ultimately decided on by “BDFL”

(Python Language Moratorium until late 2012)

<http://www.python.org/dev/peps/>
<http://docs.python.org/devguide/#>

<http://docs.python.org/>

<http://www.python.org/>

Enthought Python Distribution

Commercial python distribution aimed at scientists. Free for academics. What we will use because it is easy to install and has all the modules we need.