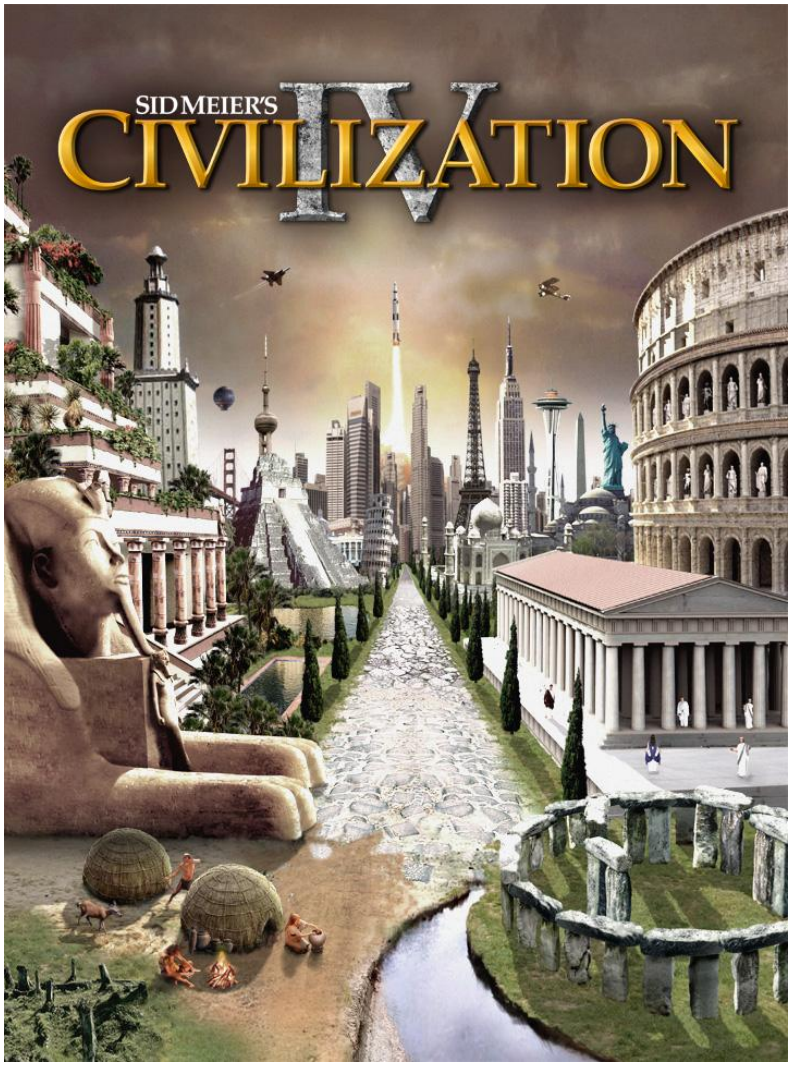
Zen of the DLR

Platforms, Balance and Working Code

Jim Hugunin
DLR Architect

"Python plays a key role in our production pipeline. Without it a project the size of Star Wars: Episode II would have been very difficult to pull off. From crowd rendering to batch processing to compositing, Python binds all things together,"

- Tommy Burnette, Senior Technical Director, Industrial Light & Magic.



"Python, like many good technologies, soon spreads virally throughout your development team and finds its way into all sorts of applications and tools…Python scripts are used in many areas of the game."

- Mustafa Thamer, Civilization IV development team

"Python has been an important part of Google since the beginning, and remains so as the system grows and evolves."

– Peter Norvig, director of search quality at Google

Too much of my world lived in C!

C:\WINDOWS\system32\cmd.exe

```
Compiling the project. Please wait...
During the process of compilation,
it is normal to have compiler/linker issue warnings.
ladder.c: In function `SERV_EXECUTE_PLC_LOGIC_BLOCK':
ladder.c:11393: internal compiler error: Segmentation fault
Please submit a full bug report,
with preprocessed source if appropriate.
Send email to arm-gcc@codesourcery.com for instructions.
```

Standard ECMA-335

Common Language Infrastructure (CLI)

- This International Standard defines the Common Language Infrastructure (CLI) in which applications written in multiple high-level languages can be executed in different system environments without the need to rewrite those applications to take into consideration the unique characteristics of those environments.

ActiveState's report on Python for .NET

- "The speed of the current system is so low as to render the current implementation useless for anything beyond demonstration purposes."

Is the MS CLI actually that "common"?

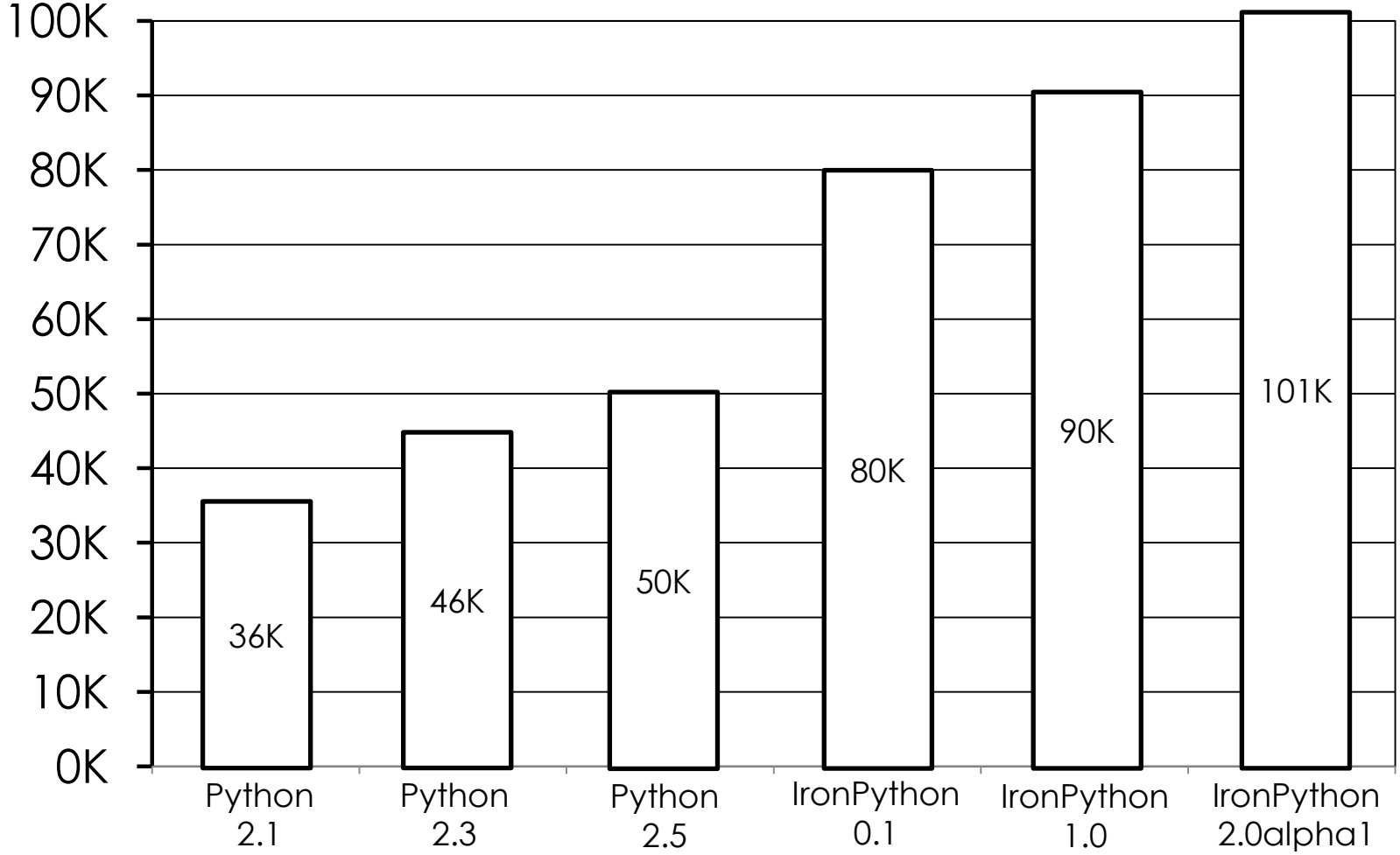
- “The CLI and CLR are it seems designed only for statically typed languages, so in their current form it is extremely difficult if not impossible to write a usable .NET compiler for dynamically-typed languages like Smalltalk, LISP, Dylan, Perl, Python, etc - you can get them working, but they run several orders of magnitude more slowly than traditional implementations, and their degree of interoperabilty with languages like C# is annoyingly limited." – Richard Bayarri Bartual – Delphi Forums

Jon Udell in Infoworld

- “The CLI is, by design, not friendly to dynamic languages. Prototypes were built, but ran way too slowly."
  - Aug. 2003

- How did Microsoft screw up so badly that .NET is a worse platform for dynamic languages than the JVM?

Standard Pystone Benchmark



New Comments

- “IronPython: .NET *is* a good platform for dynamic languages” – GameDev.Net, March 2004
- “Before IronPython, the common wisdom was that it was difficult to make dynamic languages perform well on the CLR.” – Edd Dumbill, July 2004
- “There was a meme floating around, a few years ago, that the CLR is inherently unfriendly to dynamic languages. As one of the transmitters of that meme, I'm delighted to be proved wrong.” – Jon Udell, InfoWorld, July 2004

Lessons Learned

- Performance of applications is easily confused with quality of platform
- Compilers are still hard

Sharing is good

- Shared bytecode intermediate language
- Just in time and ahead of time compilers
- Highly tuned garbage collector
- Reflection and dynamic loading support
- Generic type system
- Tool integration
  - Debugging
  - Profiling
- …

Which languages?

JavaScript

VisualBasic

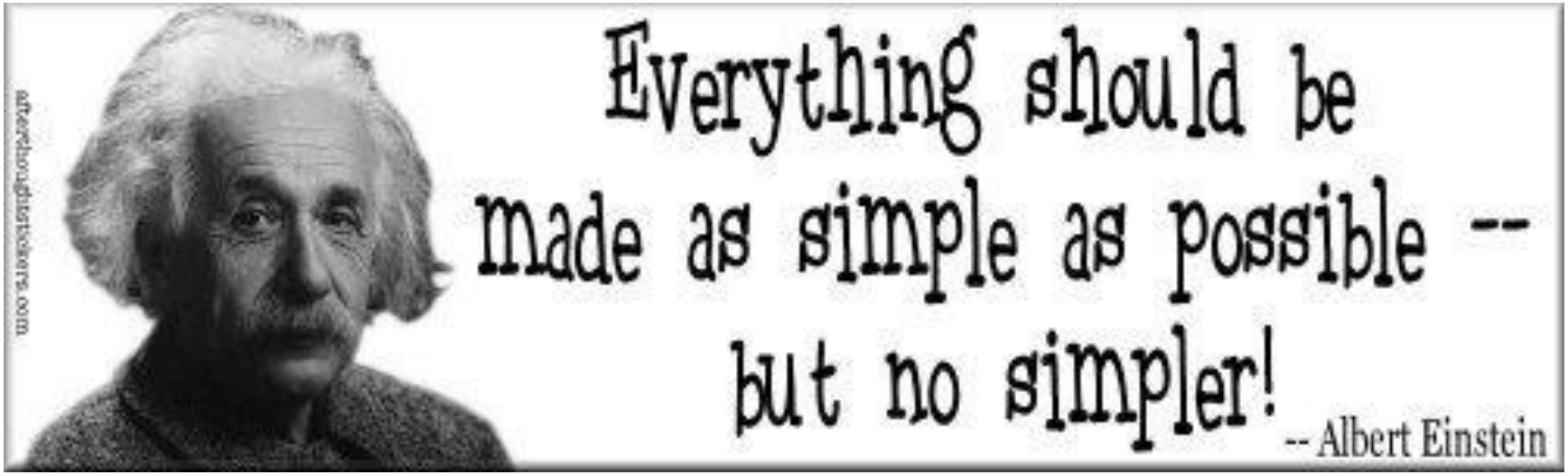
A wrist friendly language for the CLI

An Excuse to Build a Platform

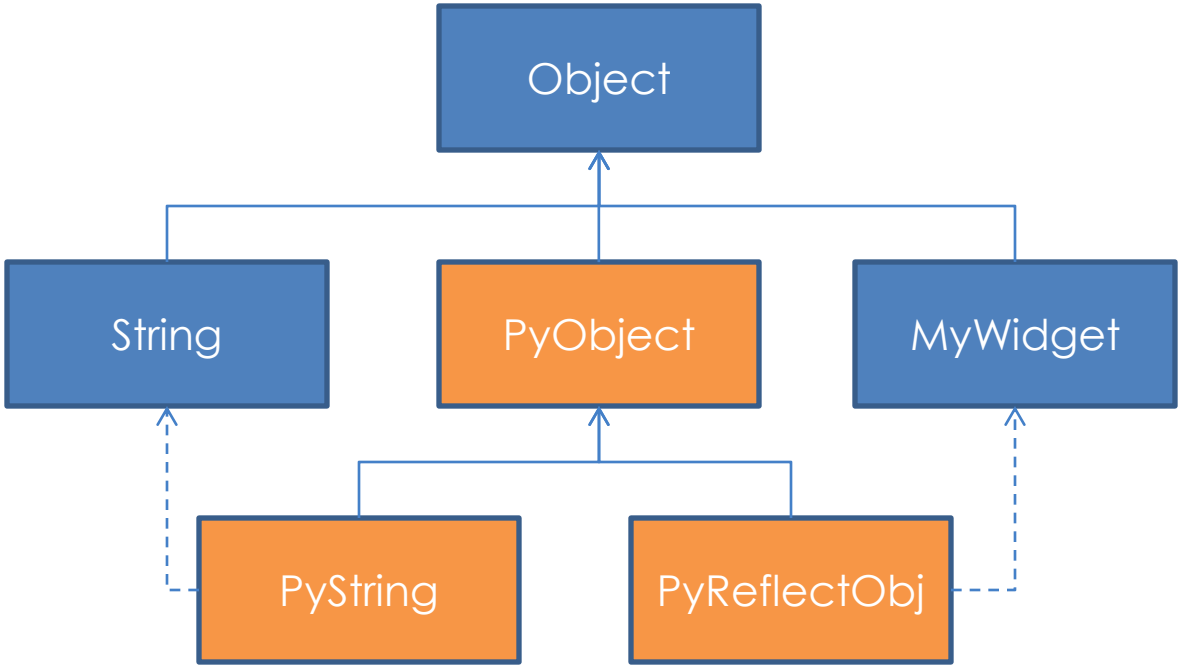
Dynamic
Language
Runtime

Pieces of the DLR

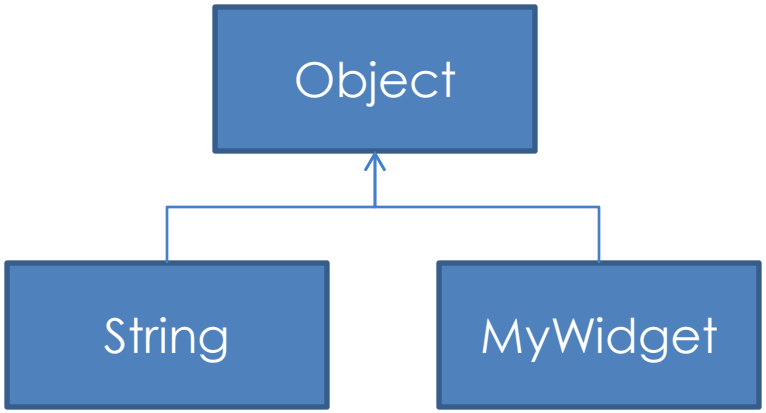
- Dynamic Type System
- Hosting APIs
- Compiler Helpers


Everything should be
made as simple as possible --
but no simpler!
-- Albert Einstein

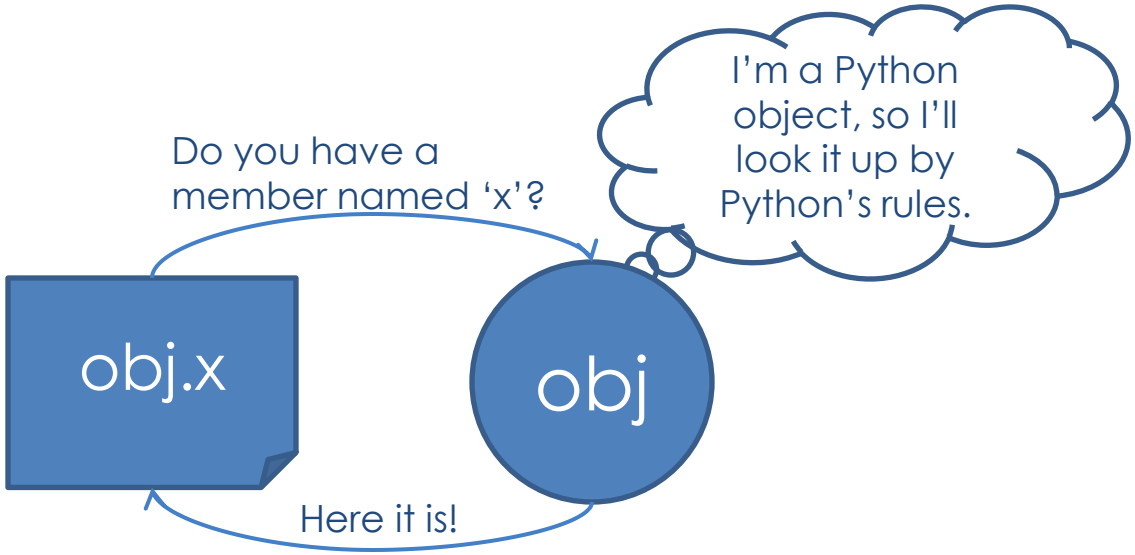
The One True Object



The One True Object



Messages

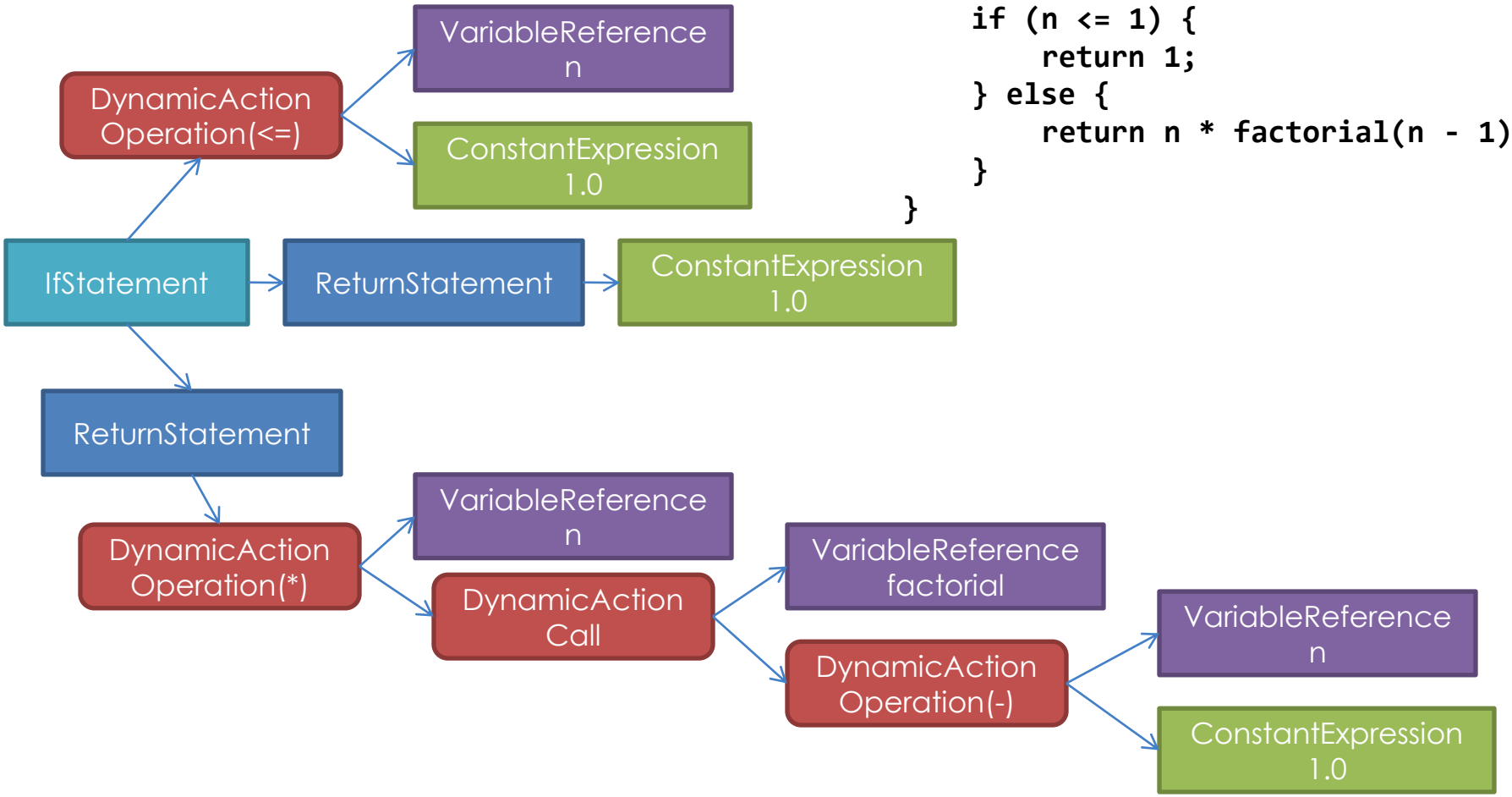

I'm a Python object, so I'll look it up by Python's rules.
Do you have a member named 'x'?
obj.x
obj
Here it is!

Basic Messages

- [Get | Set | Delete]Member(name, case-sensitivity)
  - gets, sets or deletes a named member on an object
- [Call | CreateInstance](arguments and modifiers)
  - standard call or 'new' call to an object with arguments
  - modifiers include
    - parameter names
    - support for expanding argument list or keyword dictionary
    - argument representing an implicit this
- SimpleOperation(OperationKind enumeration)
  - catch-all for simple common operations – with no parameters
  - Add, Subtract, Multiply, Divide, …
  - GreaterThan, LessThan, Equal, …
  - GetItem/Indexing, …
- Convert(Type)
  - converts an object to a given static type if possible

Trees

```
function factorial(n) {
    if (n <= 1) {
        return 1;
    } else {
        return n * factorial(n - 1);
    }
}
```



What’s the hard part?

2 + 2

x = 2147483647

x + 1

2147483648L (N-bit)

2147483648.0

OverflowException

-2147483648

2147483648L (64-bit)

?

```
x = 1
x / 3
```

0

0.333333333333

⅓

RangeError

?

Being Python…

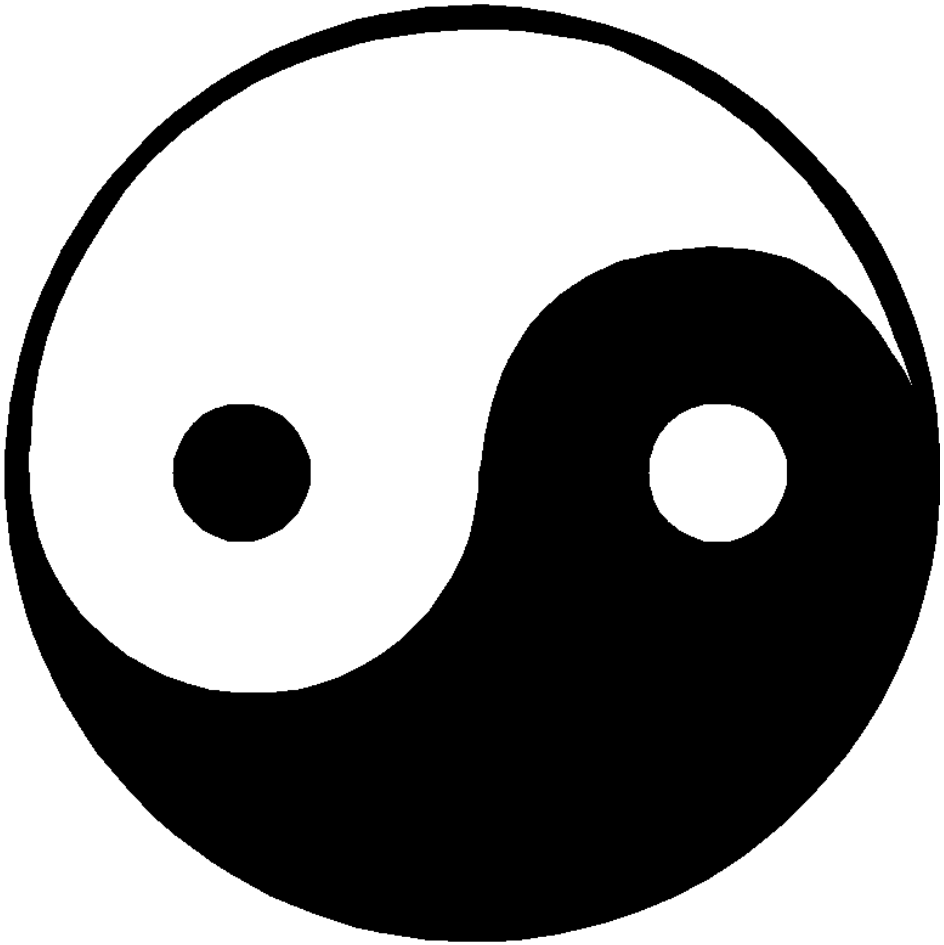
- Language
  - Concrete compatibility – regression test suite
- Libraries
  - Python-based libraries should just work
  - C-based libraries need a porting job
    - More and more over time
- Community
  - Microsoft joined the PSF as corporate member
  - Active mailing list with users helping each other
  - FePy sourceforge project
  - On going dialog with core Python developers

True Python Implementation

- Good but far from perfect
- Run 85 out of 295 regression tests
- Why are tests missing?
  - Missing C-based extension modules
    - Many are platform specific, even SGI specific
  - Bugs in IronPython
  - Bugs in Test Suite

Compatibility – the o's

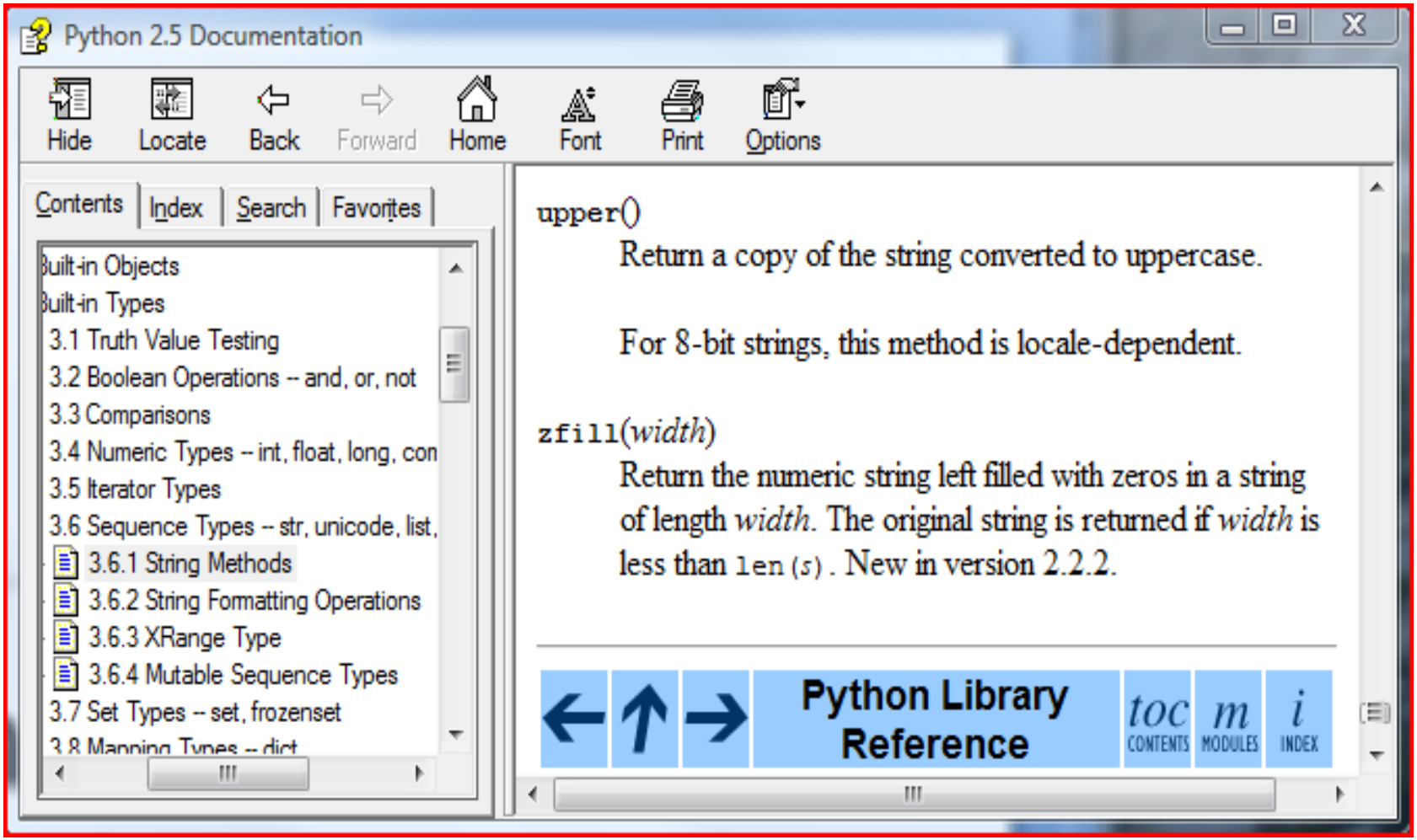
- missing modules – openpty, ossaudiodev
- operator.concat(13,29)
  - New test added between Python-2.4 and Python-2.4.3
  - CPython throws a TypeError
  - IronPython returns 42
  - Discussion with core Python team
- type(str) is type
  - Fixed now in 2.0alpha1
- test depends on sorting order of string keys in dictionary
  - d = {'a', 'b', 'c'}
  - d.keys() == ['b', 'c', 'a'] vs. ['a', 'b', 'c'] vs. ['c', 'b', 'a']
  - This is a bad test
- Partial module - os
  - os.tmpnam, os.tmpfile, os.stat
  - Could fix this test by removing these functions from os
  - Would that be the right thing?



A seemingly simple question

```
>>> s = “python and .net working together"
>>> s.upper()
```

Python docs for string



MSDN docs for string

String.ToUpper Method () - Microsoft Visual Studio 2005 Documentation - Micro...

File Edit View Tools Window Help

Back How Do I Search Index Contents

Contents

Filtered by:

(unfiltered)

ToLowerInvariant I
ToString Method
ToUpper Method
ToUpperInvariant I

Co... In... He...

String.ToUpper Method ()

URL: ms-help://MS.VSCC.v80/MS.MSDN.v80/MS.NETDEVFX.v20.en/cp

.NET Framework Class Library

String.ToUpper Method ()

See Also Example

Collapse All Language Filter: C#

Returns a copy of this String converted to uppercase, using the casing rules of the current culture.

Ready

A Seemingly Simple Answer

```
>>> s = “python and .net working together"
>>> s.upper()
'PYTHON AND .NET WORKING TOGETHER‘
>>> s.ToUpper()
'PYTHON AND .NET WORKING TOGETHER'
```

This is just C#'s extension methods

- Something fun to do with C# 3.0

```
using IronPython.Runtime.Types;
…
public static void Main() {
    string s = "python and c# working together";
    Console.WriteLine(s.title());
}
>> Python And C# Working Together
```

Python Community Feedback

```
>>> s = “python and .net working together"
>>> s.upper()
'PYTHON AND .NET WORKING TOGETHER‘
>>> s.ToUpper()
Traceback (most recent call last):
  File , line 0, in input##308
AttributeError: 'str' object has no attribute 'ToUpper'
```

Who is right?

- .NET developer
  - Should call ToUpper() method which is on System.String
  - Must do this to be .NET experience compatible!
- Python developer
  - Should throw AttributeError – no 'ToUpper' on strings
  - Must do this to be Python compatible!

Can we please everyone?

- Python can select behavior per module
- Can't break existing modules
- Same lexical scoping as extension methods

```
>>> 1/2
0
>>> from __future__ import division
>>> 1/2
0.5
```

Can we please everyone?

```
>>> s = “python and .net working together"
>>> s.upper()
'PYTHON AND .NET WORKING TOGETHER‘
>>> s.ToUpper()
Traceback (most recent call last):
  File , line 0, in input##308
AttributeError: 'str' object has no attribute 'ToUpper‘
>>> import clr
>>> s.ToUpper()
'PYTHON AND .NET WORKING TOGETHER'
```

Each community is different

- IronRuby adds ruby-style names
  - `s.to_upper and s.ToUpper`
  - `Choice is good!`
- JavaScript currently adds JS-style names
  - `s.toUpper() and s.ToUpper()`
  - No final call yet!
- VB blithely ignores this nonsense
  - `s.ToUpper == s.toupper == s.TOUPPER == …`
  - `Why does anyone worry about casing?`

Two Exceptional Worlds

```
>>> import System
>>> System.Math.DivRem(5,2)
(2, 1)
>>> System.Math.DivRem(5,0)
Traceback (most recent call last):
  File , line 0, in input##158
  File mscorlib, line unknown, in DivRem
ZeroDivisionError: Attempted to divide by zero.
```

- But the docs say DivRem throws DivideByZeroException?

C#

```
public static int DivRem (
    int a,
    int b,
    out int result
)
```

Parameters

a

The System.Int32 that contains the dividend.

b

The **System.Int32** that contains the divisor.

result

The **System.Int32** that receives the remainder.

Return Value

The **System.Int32** containing the quotient of the specified numbers.

Exceptions

Exception type	Condition
DivideByZeroException	**b** is zero.

Example

Two Exceptional Worlds: You get what you asked for

```
>>> try:
...     System.Math.DivRem(5,0)
... except ZeroDivisionError, e:
...     print e, type(e)
...
Attempted to divide by zero. <type 'instance'>
```

- Explicitly catching the Python exception
- You get the expected exception
- This code will also work for ‘1/0’
- Uses old-style classes for better compat

Two Exceptional Worlds: You get what you asked for

```
>>> try:
...     System.Math.DivRem(5,0)
... except System.DivideByZeroException, e:
...     print type(e)
...
<type 'DivideByZeroException'>
```

- Explicitly catching the .NET exception
- You get the expected exception
- This code will also work for ‘1/0’

Two Exceptional Worlds: You get what you asked for

```
>>> try:
...     System.Math.DivRem(5,0)
... except System.Exception, e:
...     print type(e)
...
<type 'DivideByZeroException'>
```

- Also works for catching supertype

Balance Again

- You need to understand your language
  - And community!

It’s all about the platform

Microsoft ASP.net

Windows Vista™

Microsoft .net™

Microsoft Silverlight™

…

Microsoft Exchange Server 2007

Microsoft Robotics Studio

xna Game Studio Express
your world. your game.

The Bleeding Edge

- We’re not really ready for you today
- But that’s not necessarily a bad thing…

Timing of Small Changes


The best way to predict the future is to invent it
. Alan Kay

Code

Questions?

http://codeplex.com/ironpython

dlr@microsoft.com