Dynamic Languages on .NET with the DLR

Jim Hugunin
DLR Architect

Microsoft's .NET Framework and the Common Language Infrastructure (CLI)

- One runtime engine for many languages
  - Shared bytecode intermediate language
  - Just in time and ahead of time compilers
  - One highly tuned garbage collector
  - Reflection and dynamic loading support
  - Debugger and profiler integration
  - …
- Many major languages in production use today
  - Microsoft: C#, VB.Net, Managed C++, J#, JScript.Net
  - Others: Eiffel, COBOL, Fortran, RPG and Delphi
- Enables deep integration between languages
  - Language choice is flexible – best tool for the job
  - Frameworks build value from larger ecosystem

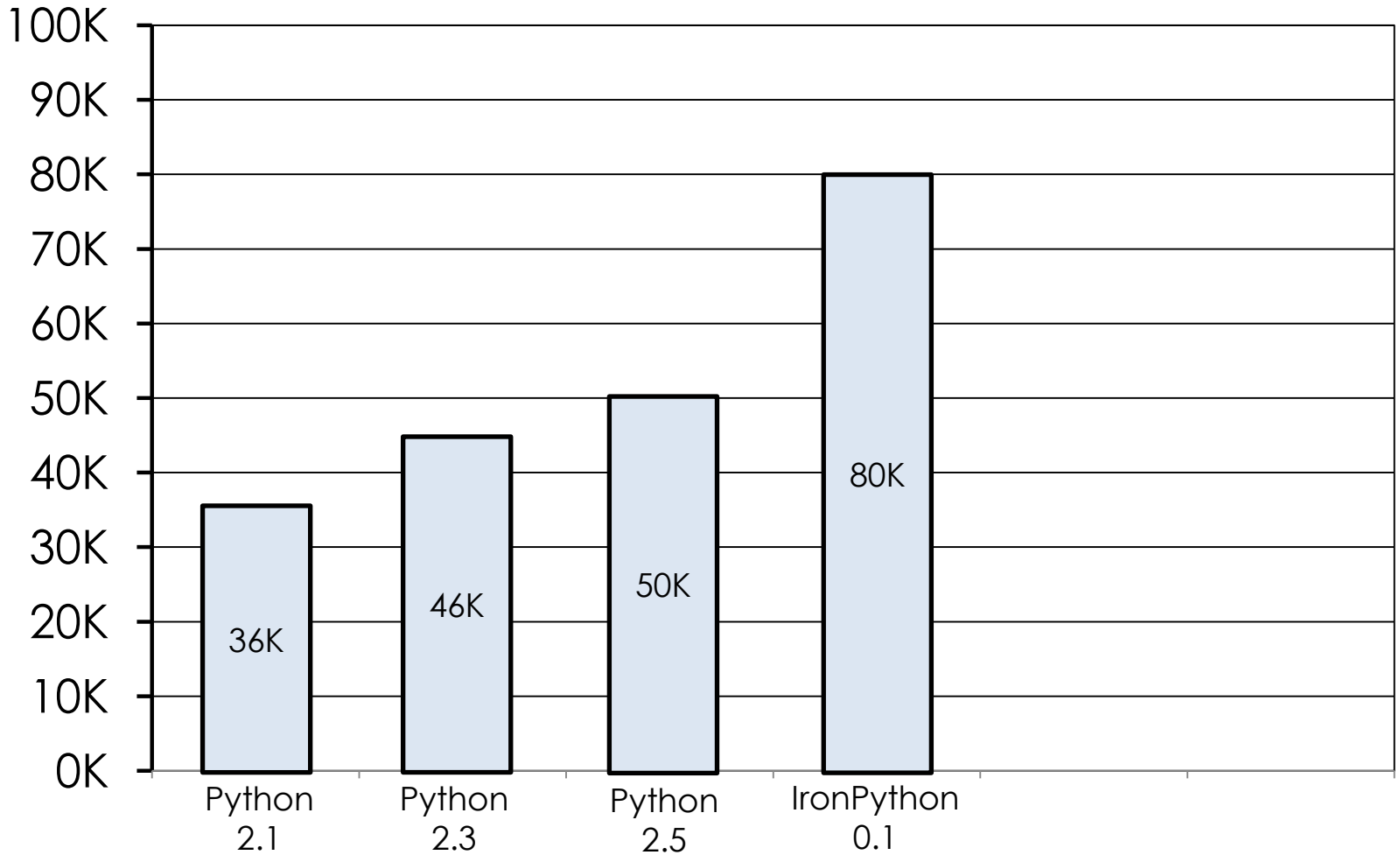
Inspiration

"*The speed of the current system is so low as to render the current implementation useless for anything beyond demonstration purposes.*" – ActiveState's report on Python for .NET

"*The CLI is, by design, not friendly to dynamic languages. Prototypes were built, but ran way too slowly.*" – Jon Udell, InfoWorld, Aug. 2003

- How could Microsoft have screwed up so badly that the CLR is far worse than the JVM for dynamic languages?
  - Jython shows that dynamic languages can run well on the JVM
- I decided to write a short pithy paper called, "Why .NET is a terrible platform for dynamic languages"

Standard Pystone Benchmark



New Comments

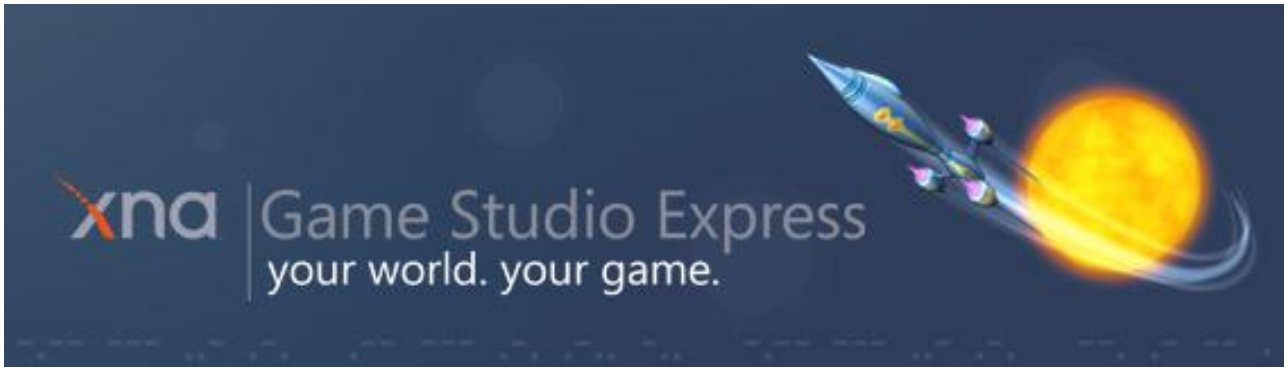
- "IronPython: .NET *is* a good platform for dynamic languages" – GameDev.Net, March 2004
- "Before IronPython, the common wisdom was that it was difficult to make dynamic languages perform well on the CLR." – Edd Dumbill, July 2004
- "There was a meme floating around, a few years ago, that the CLR is 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

Observations

- It's easy to blame the platform for the performance of an application.
- Building compilers is still too hard.

IronPython's dual goals

- True Python Implementation
  - Interactive and dynamic experience
  - Existing programmer knowledge and code
  - Rich set of libraries
  - Run existing regression tests and code
- Seamless integration with .NET
  - Consume .NET libraries and run inside .NET hosts
  - Interoperate with other .NET languages
  - Exploit .NET infrastructure
    - Visual Studio, debugger, profiler, JIT, GC, …
    - Let other people do our work



- A framework for building games
  - Clearly not the world's first…
- Focus is on C# development
  - This is a dramatic step away from C++
- What if I don't want to use C#?

Extending Python in C

```
#include <Python.h>

typedef struct {
    PyObject_HEAD
    PyObject *first; /* first name */
    PyObject *last;  /* last name */
    int number;
} Noddy;

static PyTypeObject noddy_NoddyType = {
    PyObject_HEAD_INIT(NULL)
    0,                         /*ob_size*/
    "noddy.Noddy",             /*tp_name*/
    sizeof(noddy_NoddyObject), /*tp_basicsize*/
    0,                         /*tp_itemsize*/
    0,                         /*tp_dealloc*/
    0,                         /*tp_print*/
    0,                         /*tp_getattr*/
    0,                         /*tp_setattr*/
    0,                         /*tp_compare*/
    0,                         /*tp_repr*/
    0,                         /*tp_as_number*/
    0,                         /*tp_as_sequence*/
    0,                         /*tp_as_mapping*/
    0,                         /*tp_hash */
    0,                         /*tp_call*/
    0,                         /*tp_str*/
    0,                         /*tp_getattro*/
    0,                         /*tp_setattro*/
    0,                         /*tp_as_buffer*/
    Py_TPFLAGS_DEFAULT,        /*tp_flags*/
    "Noddy objects",           /* tp_doc */
};
static PyMethodDef noddy_methods[] = {
    {NULL}  /* Sentinel */
};

#ifndef PyMODINIT_FUNC                  /* declarations for DLL import/export */
#define PyMODINIT_FUNC void
#endif
PyMODINIT_FUNC
initnoddy(void)
{
    PyObject* m;

    noddy_NoddyType.tp_new = PyType_GenericNew;
    if (PyType_Ready(&noddy_NoddyType) < 0)
        return;

    m = Py_InitModule3("noddy", noddy_methods,
                       "Example module that creates an extension type.");

    Py_INCREF(&noddy_NoddyType);
    PyModule_AddObject(m, "Noddy", (PyObject *)&noddy_NoddyType);
}
```

```
static void
Noddy_dealloc(Noddy* self)
{
    Py_XDECREF(self->first);
    Py_XDECREF(self->last);
    self->ob_type->tp_free((PyObject*)self);
}

static PyObject *
Noddy_new(PyTypeObject *type, PyObject *args, PyObject *kwds)
{
    Noddy *self;

    self = (Noddy *)type->tp_alloc(type, 0);
    if (self != NULL) {
        self->first = PyString_FromString("");
        if (self->first == NULL)
          {
            Py_DECREF(self);
            return NULL;
          }

        self->last = PyString_FromString("");
        if (self->last == NULL)
          {
            Py_DECREF(self);
            return NULL;
          }

        self->number = 0;
    }

    return (PyObject *)self;
}
static int
Noddy_init(Noddy *self, PyObject *args, PyObject *kwds)
{
    PyObject *first=NULL, *last=NULL, *tmp;

    static char *kwlist[] = {"first", "last", NULL};

    if (! PyArg_ParseTupleAndKeywords(args, kwds, "|OO", kwlist,
                                      &first, &last))
        return -1;

    if (first) {
        tmp = self->first;
        Py_INCREF(first);
        self->first = first;
        Py_XDECREF(tmp);
    }

    if (last) {
        tmp = self->last;
        Py_INCREF(last);
        self->last = last;
        Py_XDECREF(tmp);
    }

    return 0;
}


static PyMemberDef Noddy_members[] = {
    {"first", T_OBJECT_EX, offsetof(Noddy, first), 0,
     "first name"},
    {"last", T_OBJECT_EX, offsetof(Noddy, last), 0,
     "last name"},
    {NULL}  /* Sentinel */
};
```

Extending Python in C#

```
namespace noddy {
    public class Noddy {
        public string first, last;
        public Noddy(string first, string last) {
            this.first = first;
            this.last = last;
        }
    }
}
```

Memory Management On Thin Ice

```
void bug(PyObject *list) {
    PyObject *item = PyList_GetItem(list, 0);

    PyList_SetItem(list, 1, PyInt_FromLong(0L));

    PyObject_Print(item, stdout, 0); /*  BUG!  */
}
```

Memory Management On Thin Ice

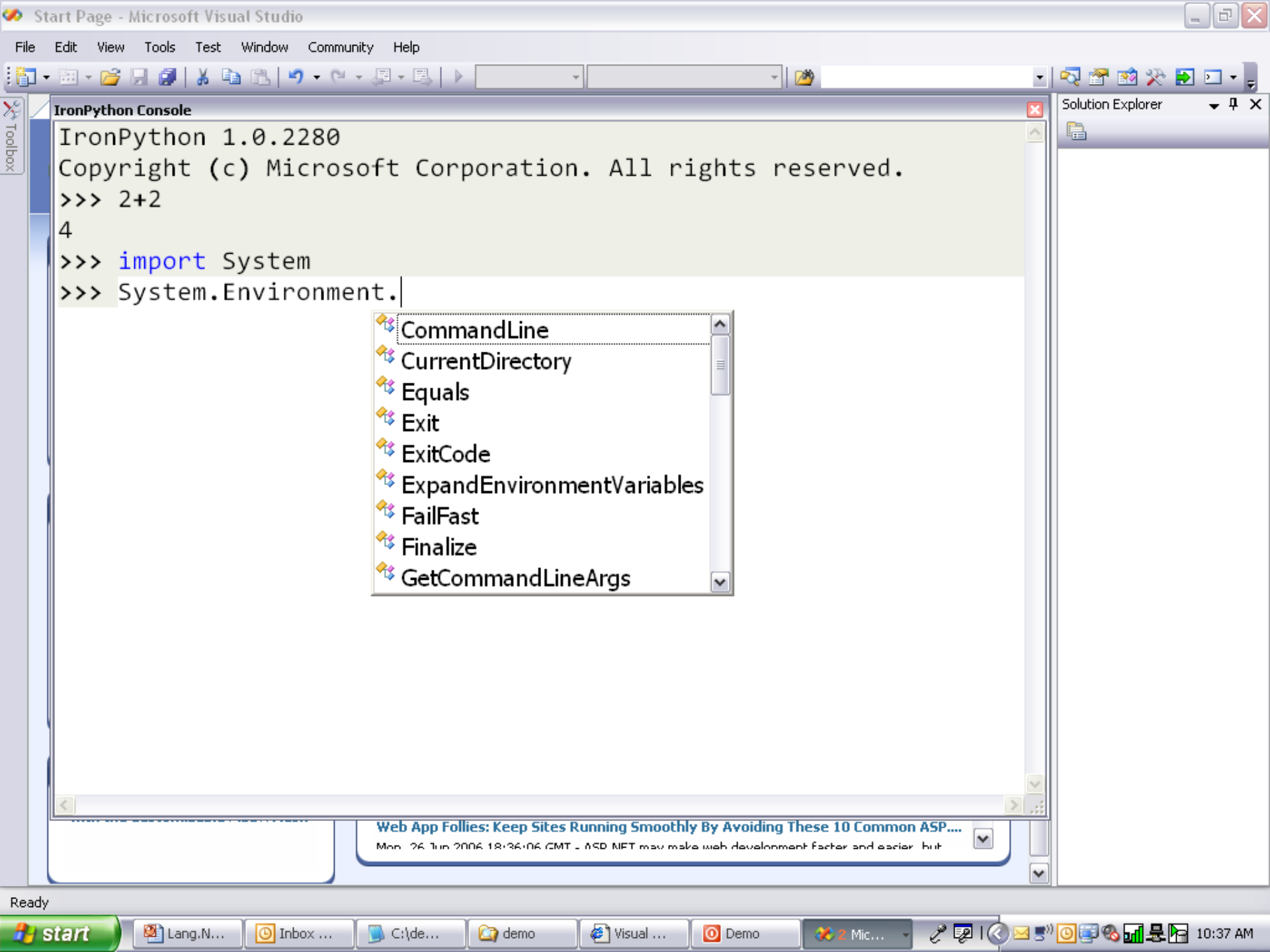
```
void bug(PyObject *list) {
    PyObject *item = PyList_GetItem(list, 0);
    Py_INCREF(item);
    PyList_SetItem(list, 1, PyInt_FromLong(0L));
    Py_DECREF(item);
    PyObject_Print(item, stdout, 0); /* FIXED! */
}
```

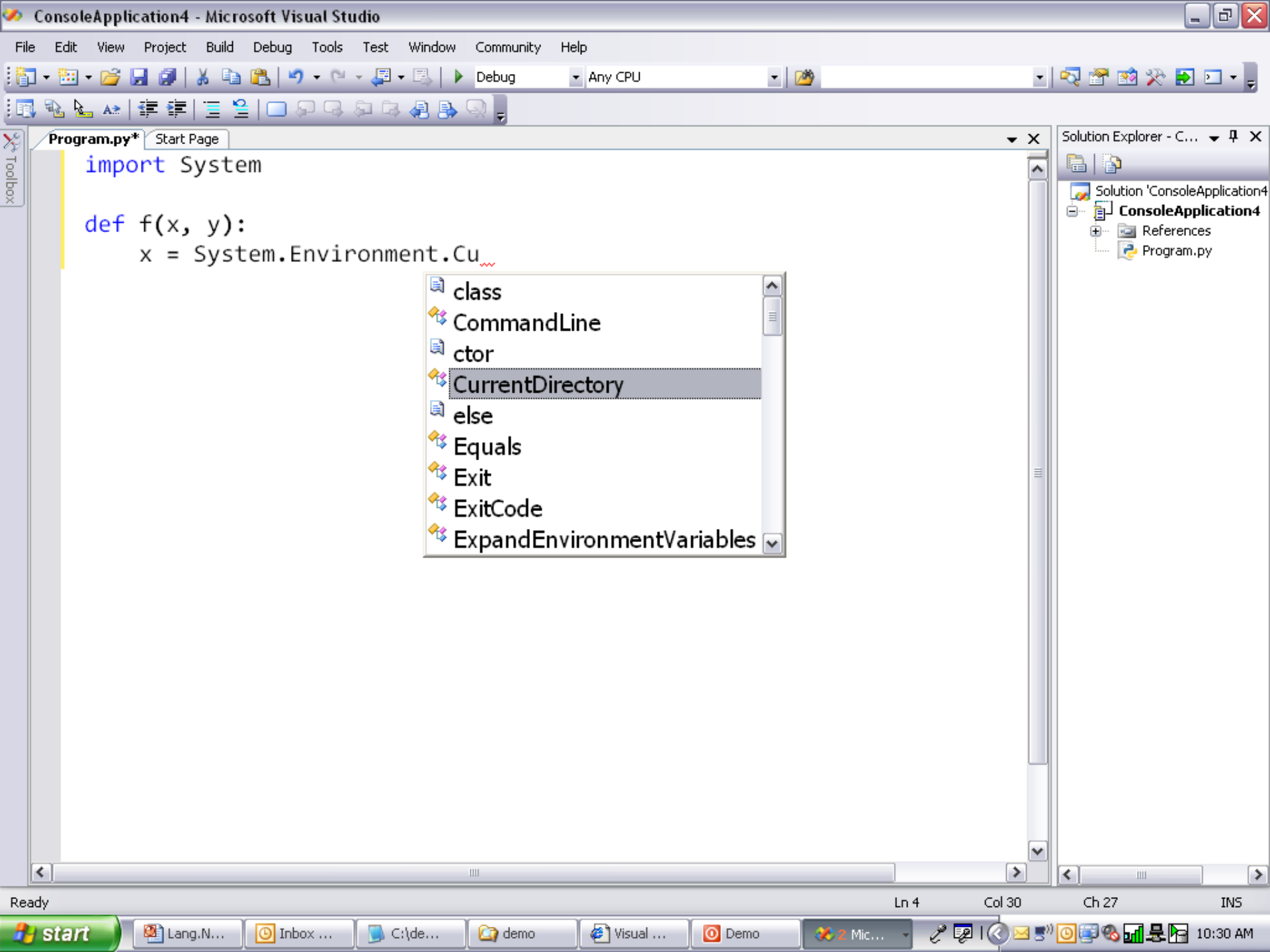
Memory Management On Solid Ground

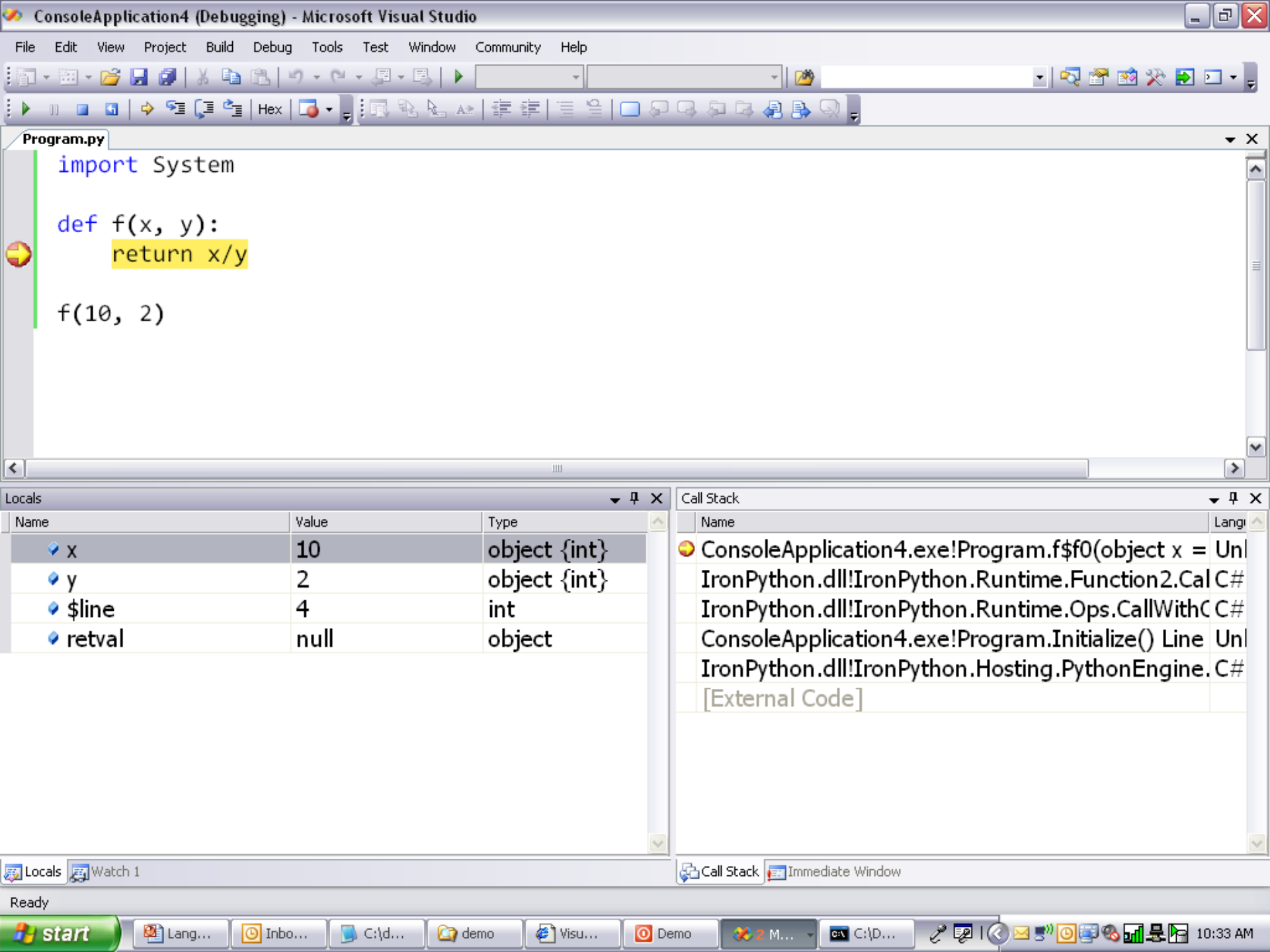
```
public static void easy(IList list) {
    object item = list[0];

    list[1] = 0;

    Console.WriteLine(item);
}
```

Visual Studio Integration

- VS SDK language integration sample
  - IronPython team working with VS SDK team
  - Implementation is 100% in C#
    - SDK handles all required C++ or COM code
  - Full source code included w/ VS SDK


Start Page - Microsoft Visual Studio
File Edit View Tools Test Window Community Help
IronPython Console
IronPython 1.0.2280
Copyright (c) Microsoft Corporation. All rights reserved.
>>> 2+2
4
>>> import System
>>> System.Environment.
CommandLine
CurrentDirectory
Equals
Exit
ExitCode
ExpandEnvironmentVariables
FailFast
Finalize
GetCommandLineArgs
Solution Explorer
Toolbox
Web App Follies: Keep Sites Running Smoothly By Avoiding These 10 Common ASP....
Ready
start
10:37 AM


ConsoleApplication4 - Microsoft Visual Studio
File Edit View Project Build Debug Tools Test Window Community Help
Debug
Any CPU
Program.py*
Start Page
import System
def f(x, y):
x = System.Environment.Cu
class
CommandLine
ctor
CurrentDirectory
else
Equals
Exit
ExitCode
ExpandEnvironmentVariables
Solution Explorer - C...
Solution 'ConsoleApplication4
ConsoleApplication4
References
Program.py
Toolbox
Ready
Ln 4
Col 30
Ch 27
INS
start
10:30 AM


ConsoleApplication4 (Debugging) - Microsoft Visual Studio
File Edit View Project Build Debug Tools Test Window Community Help
Hex
Program.py
import System
def f(x, y):
return x/y
f(10, 2)
Locals
Name Value Type
x 10 object {int}
y 2 object {int}
$line 4 int
retval null object
Locals Watch 1
Call Stack
Name Lang
ConsoleApplication4.exe!Program.f$f0(object x = Unl
IronPython.dll!IronPython.Runtime.Function2.Cal C#
IronPython.dll!IronPython.Runtime.Ops.CallWithC C#
ConsoleApplication4.exe!Program.Initialize() Line Unl
IronPython.dll!IronPython.Hosting.PythonEngine. C#
[External Code]
Call Stack Immediate Window
Ready
start
10:33 AM

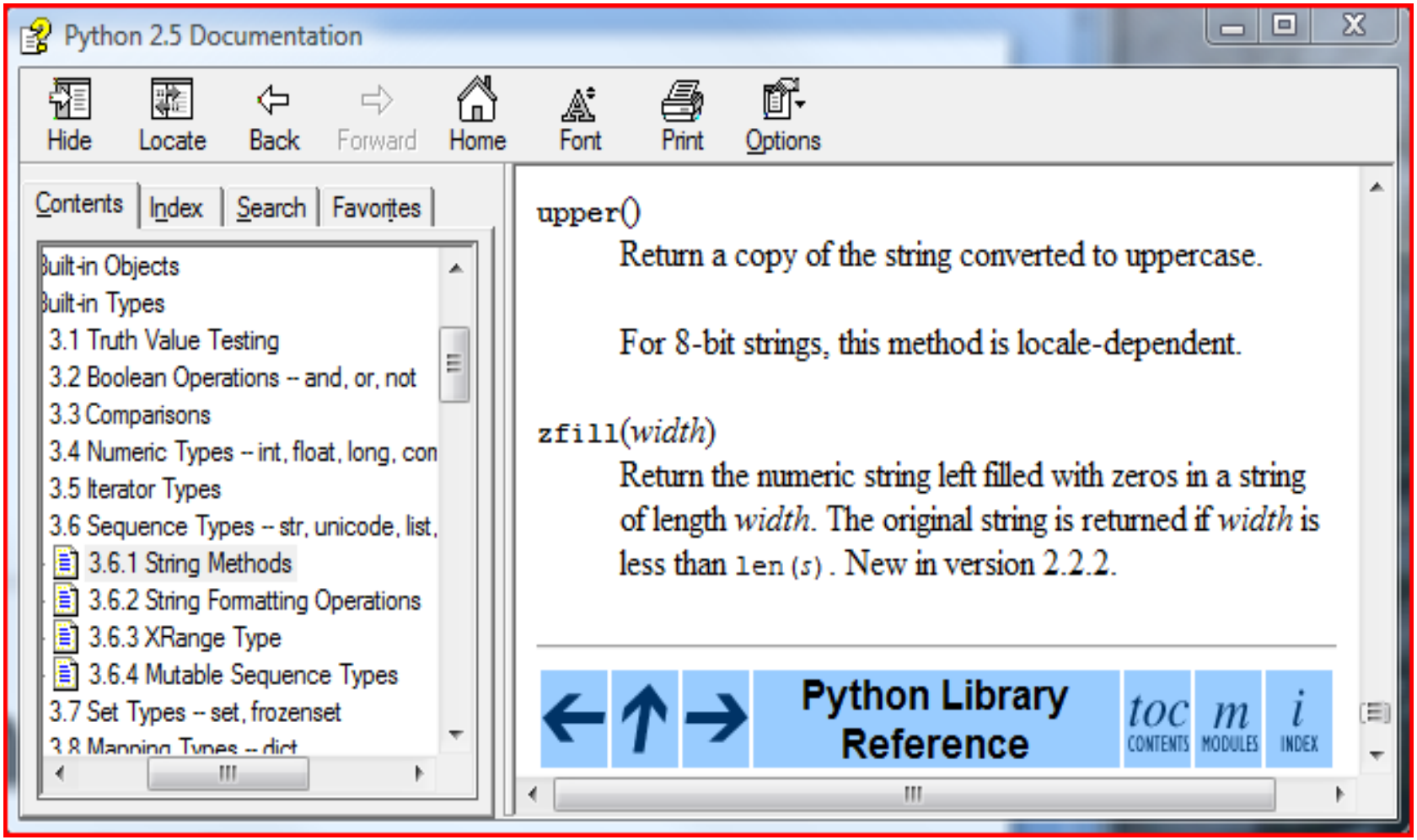
IronPython's dual goals

- True Python Implementation
  - Interactive and dynamic experience
  - Existing programmer knowledge and code
  - Rich set of libraries
  - Run existing regression tests and code
- Seamless integration with .NET
  - Consume .NET libraries and run inside .NET hosts
  - Interoperate with other .NET languages
  - Exploit .NET infrastructure
    - Visual Studio, debugger, profiler, JIT, GC, …
    - Let other people do our work

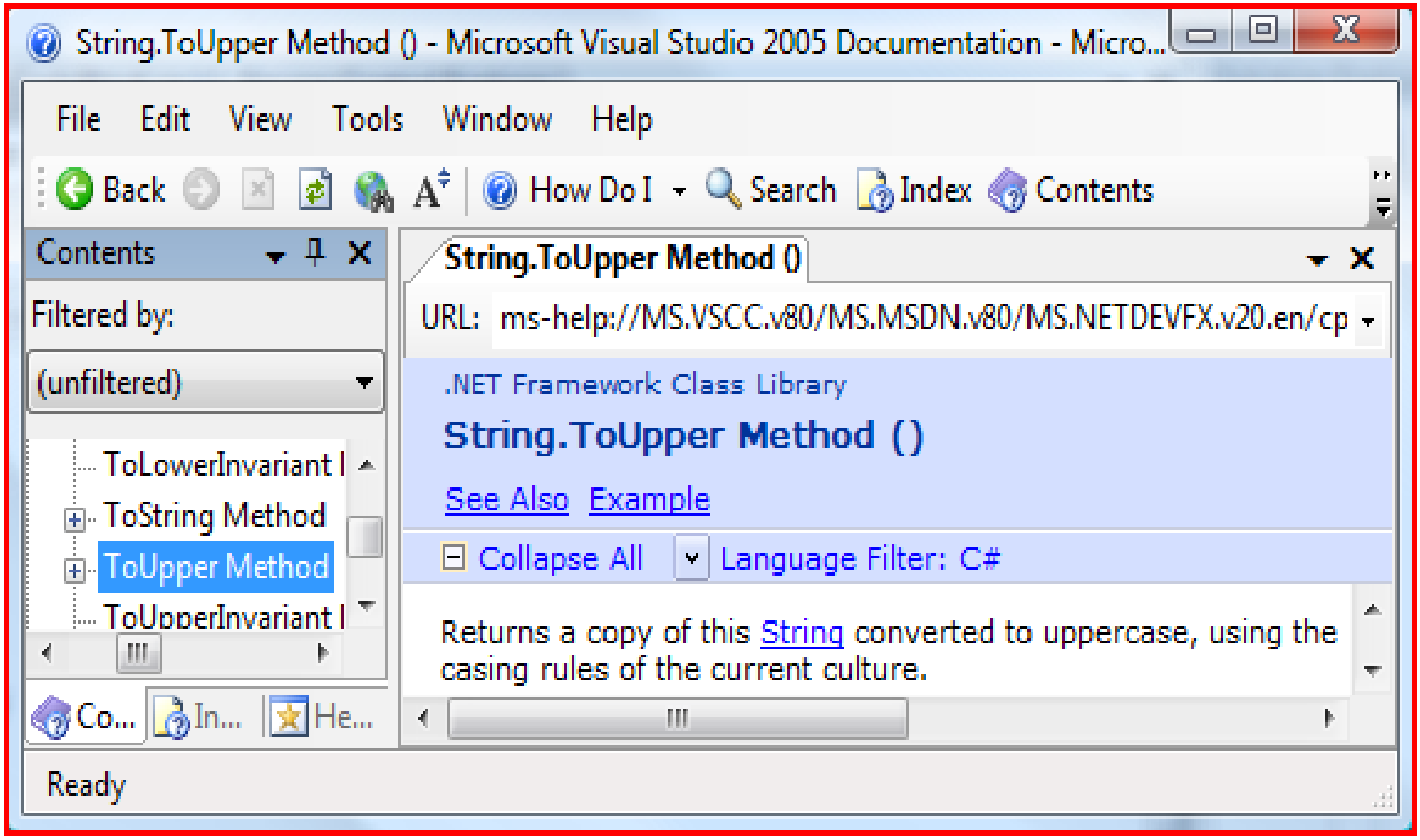
Being Python

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

Python docs for string



MSDN docs for string



A Seemingly Simple Answer

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

C#-3.0 Extension Methods

```
namespace IronPython.Runtime {
    public static class PythonStringExtensions {
        …
        public static bool isspace(string this) {
            if (this.Length == 0) return false;
            for (int i = this.Length - 1; i >= 0; i--) {
                if (!Char.IsWhiteSpace(this, i)) return false;
            }
            return true;
        }
        …
        public static string upper(this string self) {
             return self.ToUpper();
        }
        …
    }
}
```

C#-3.0 Extension Methods

```
using System;


namespace MyProject {
    public static class Program {
        public static void Main() {
            string s = “python from C#”;
            Console.WriteLine(s.upper());
        }
    }
}
```

Error:
No method upper defined on String

C#-3.0 Extension Methods

```
using System;
using IronPython.Runtime;

namespace MyProject {
    public static class Program {
        public static void Main() {
            string s = “python from C#”;
            Console.WriteLine(s.upper());
        }
    }
}
```

prints:
PYTHON FROM C#

Everyone is happy

```
>>> 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'
```

IronPython-1.0

- Released September 5, 2006
- http://codeplex.com/ironpython
- Open Source License
- Active user community

- IronPython-1.1 released April 17, 2007

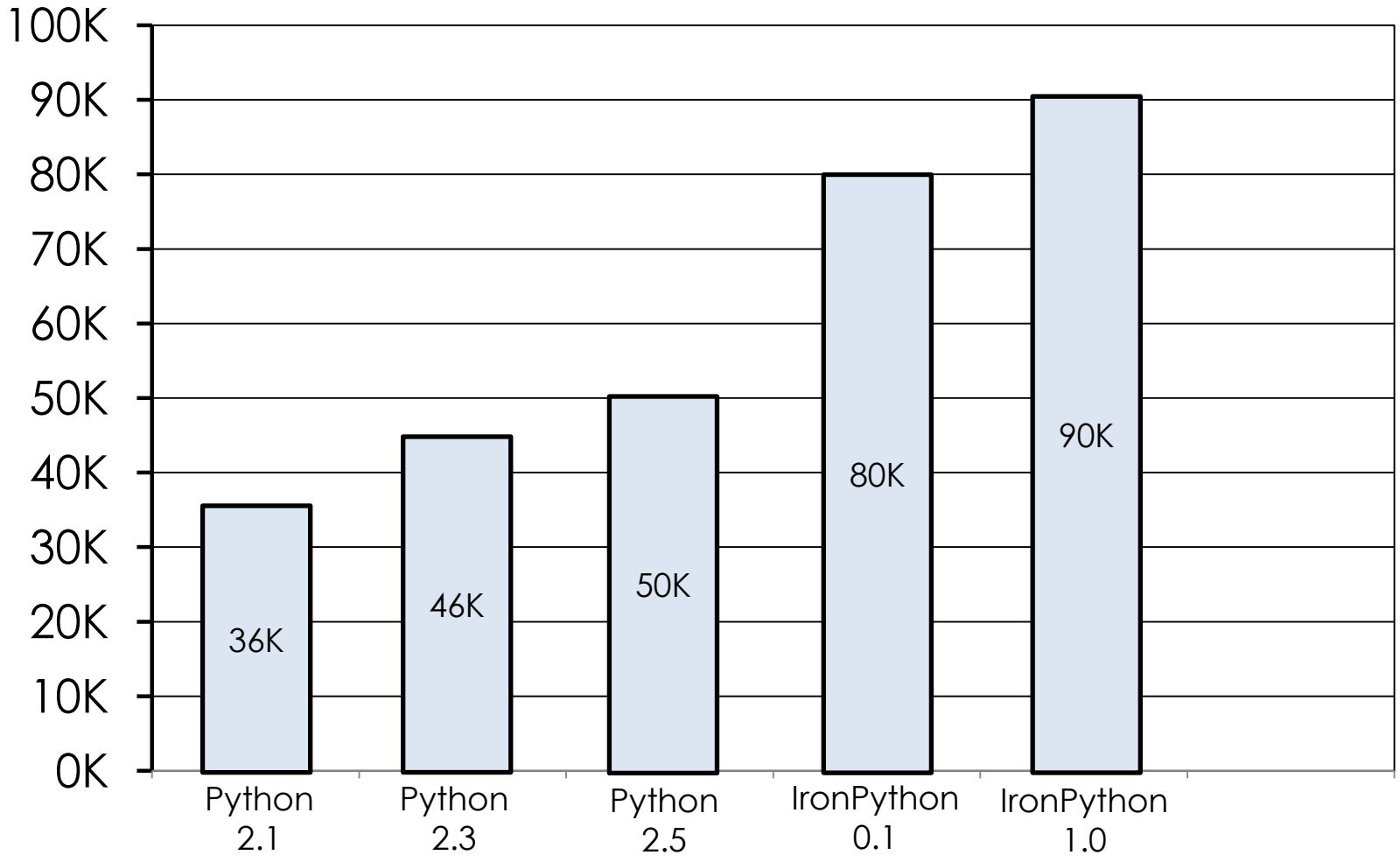
The CLR is a Good Platform

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

CLR 2.0 Made It Even Better

- DynamicMethods
  - Code generation that can be collected normally
  - Versatile tool for language implementers
- Delegate performance
  - Delegates are lightweight and type-safe function pointers
  - Performance improvement to important feature
- Generics embedded in the runtime
  - Full dynamic reflection support
  - Major new feature added in multi-language friendly way
- General platform performance work
  - IronPython leverages any platform improvements

Standard Pystone Benchmark



Why DLR?


JavaScript
php
Lua
python
APC
perl
VisualBasic
macromedia
COLDFUSIONMX7
A wrist friendly language for the CLI

Building a DLR Language

- Implement tokenizer and parser
- Translate your AST to DLR Trees
- Implement your custom types and customizations to existing .NET types

- Tuning
  - Refine and optimize your runtime libraries
  - Refine and optimize your dynamic types

Can we please everyone?

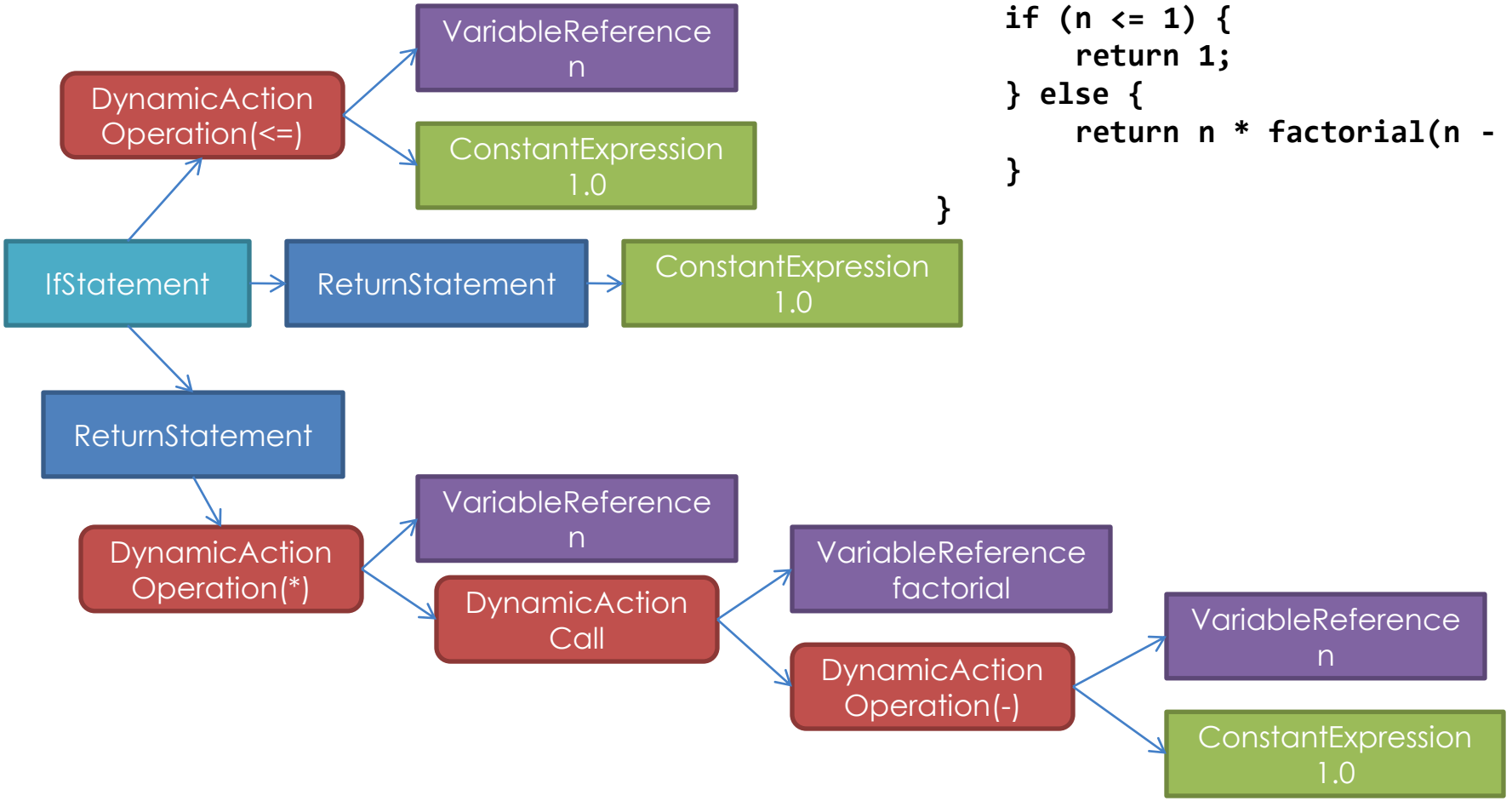
```
>>> s = “python and .net working together"
>>> s.upper()
'PYTHON AND .NET WORKING TOGETHER‘
>>> s.ToUpper()
Traceback (most recent call last):
  File , line 0, in input##308
AttributeError: 'str' object has no attribute 'ToUpper‘
>>> import clr
>>> s.ToUpper()
'PYTHON AND .NET WORKING TOGETHER'
```

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()
  - Not final yet – hard to identify JS "style"!
- VB blithely ignores this nonsense
  - s.ToUpper == s.toupper == s.TOUPPER == …
  - Why does anyone worry about casing?

DLR Trees
What vs. How

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



Compiling Factorial – to IL IronPython-0.9

```
def factorial(n):
    if n <= 1: return 1
    return n * factorial(n-1)
```

0 LOAD_FAST 0 (n)	ldarg.0
3 LOAD_CONST 1 (1)	ldsfld object __main__::c0PST04000002
6 COMPARE_OP 2 (<=)	call object Ops::LessThanEqual(object,object)
9 JUMP_IF_FALSE 8 (to 20)	call bool IronPython...Ops::IsTrue(object) brfalse IL_0020
12 POP_TOP	
13 LOAD_CONST 1 (1)	ldsfld object __main__::c0PST04000002
16 RETURN_VALUE	ret

Compiling Factorial – to x86

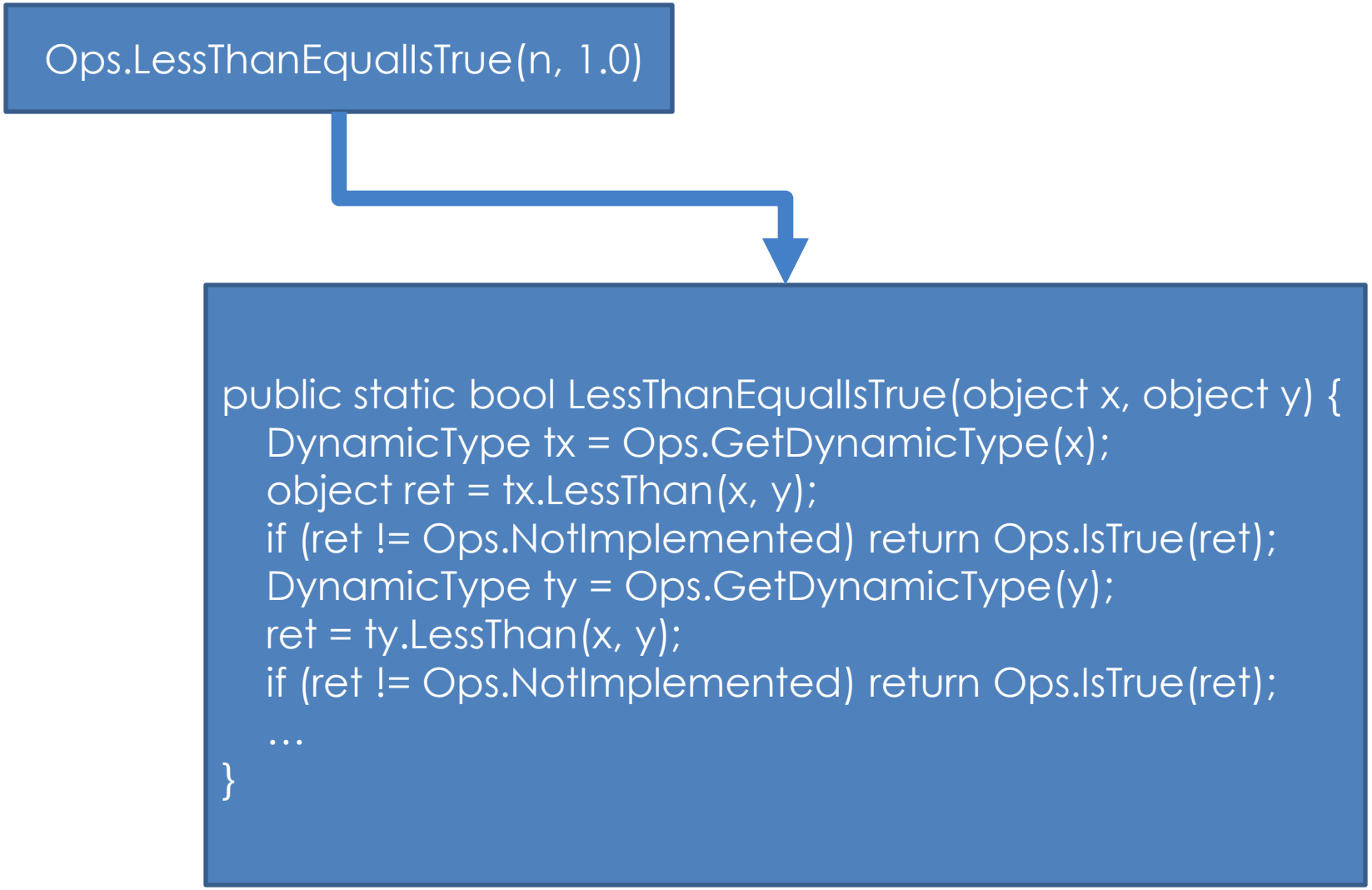
0 LOAD_FAST 0 (n)	0000001b mov edx,dword ptr ds:[01B054E4h]
3 LOAD_CONST 1 (1)	00000021 mov ecx,esi
6 COMPARE_OP 2 (<=)	00000023 call dword ptr ds:[036E3184h]
9 JUMP_IF_FALSE 8 (to 20)	00000029 mov edi,eax 0000002b mov ecx,edi 0000002d call dword ptr ds:[036E3084h] 00000033 mov edi,eax 00000035 test edi,edi 00000037 je 00000043
12 POP_TOP	
13 LOAD_CONST 1 (1)	00000039 mov eax,dword ptr ds:[01B054E4h]
16 RETURN_VALUE	<pop 4 registers and ret>

Compiling Factorial – to IL IronPython-1.0

```
def factorial(n):
    if n <= 1: return 1
    return n * factorial(n-1)
```

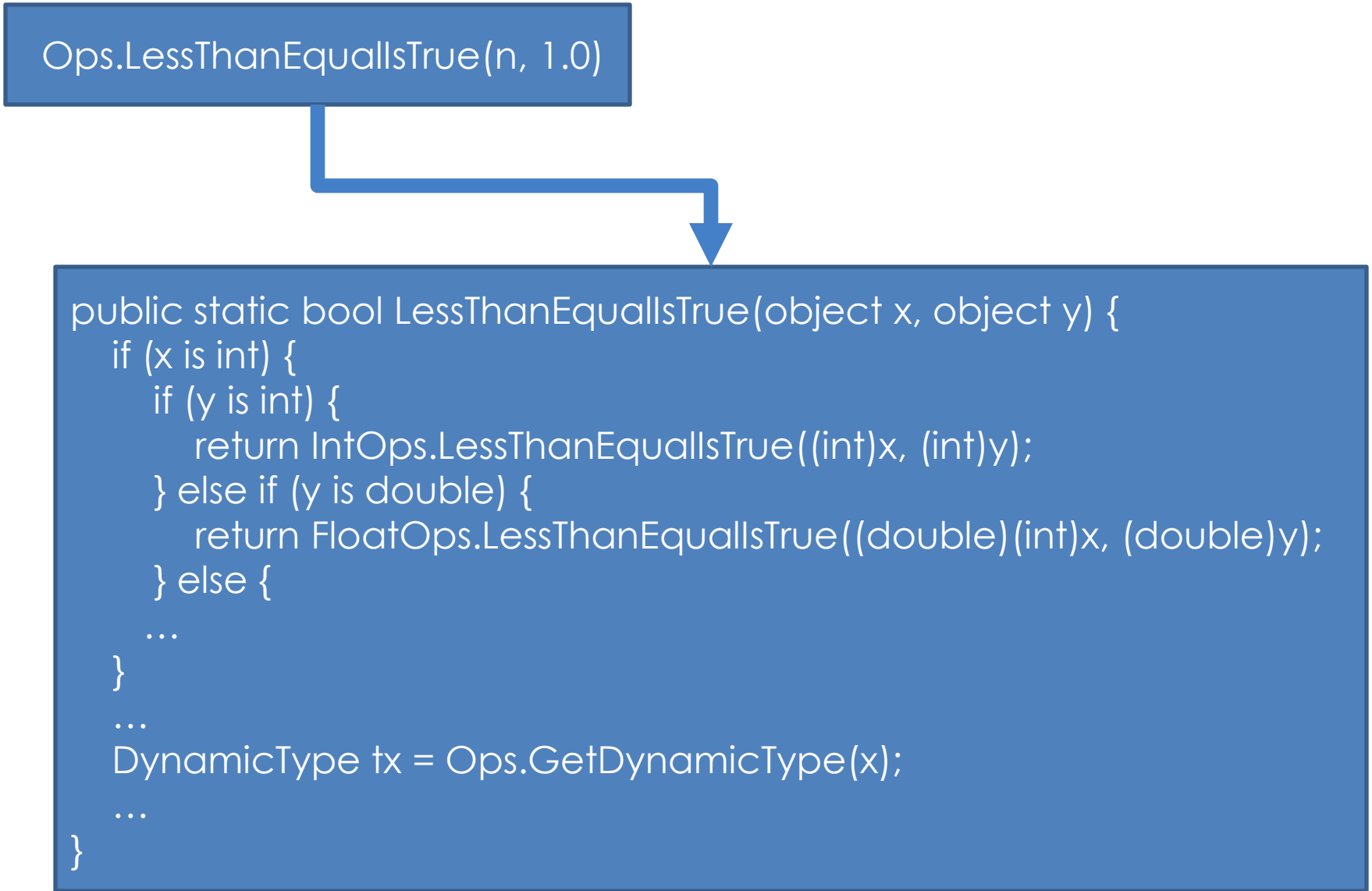
0 LOAD_FAST 0 (n)	ldarg.0
3 LOAD_CONST 1 (1)	ldsfld object __main__::c0PST04000002
6 COMPARE_OP 2 (<=)	call bool Ops::LessThanEqualIsTrue(object,object)
9 JUMP_IF_FALSE 8 (to 20)	brfalse <dest>
12 POP_TOP	
13 LOAD_CONST 1 (1)	ldsfld object __main__::c0PST04000002
16 RETURN_VALUE	ret

Dynamic Actions



Dynamic Actions with Fast Paths

Ops.LessThanEqualIsTrue(n, 1.0)



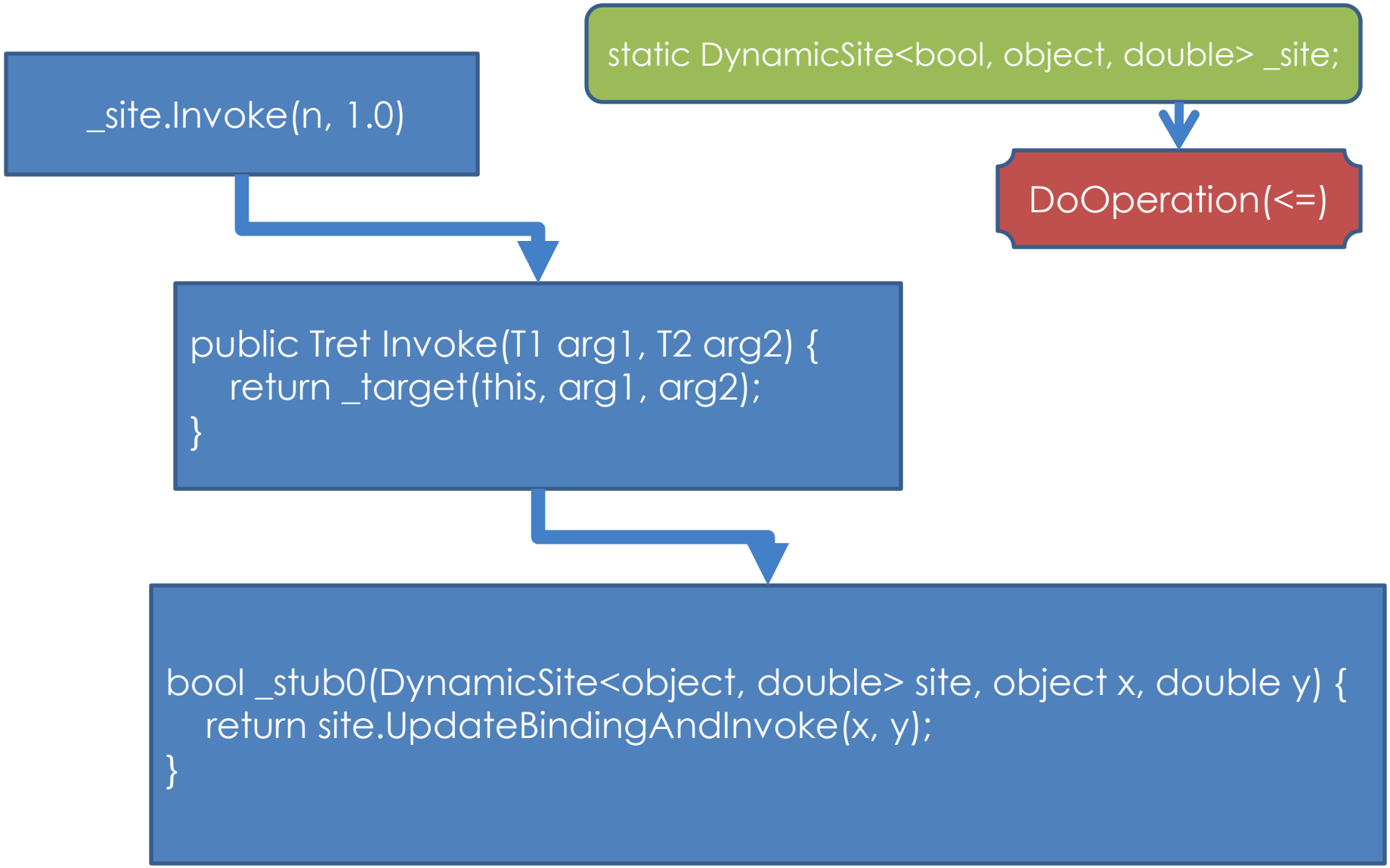
```
public static bool LessThanEqualIsTrue(object x, object y) {
    if (x is int) {
        if (y is int) {
            return IntOps.LessThanEqualIsTrue((int)x, (int)y);
        } else if (y is double) {
            return FloatOps.LessThanEqualIsTrue((double)(int)x, (double)y);
        } else {
        …
    }
    …
    DynamicType tx = Ops.GetDynamicType(x);
    …
}
```

Compiling Factorial to IL IronPython w/ DLR

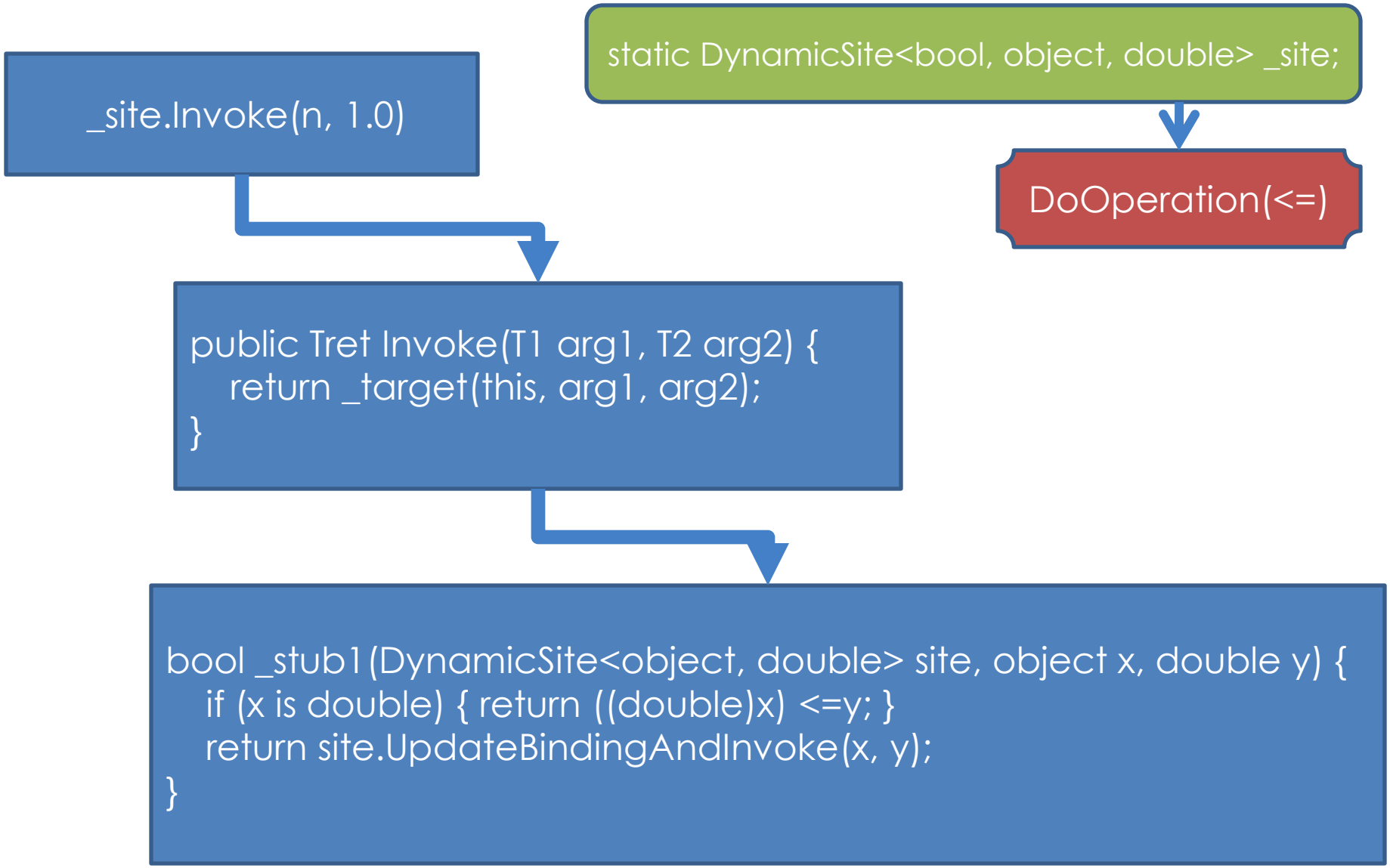
```
def factorial(n):
    if n <= 1: return 1
    return n * factorial(n-1)
```

0 LOAD_FAST 0 (n)	*ldsfld class FastDynamicSite`3<object,int32,bool>* ldarg.0
3 LOAD_CONST 1 (1)	ldc.i4.1
6 COMPARE_OP 2 (==)	call FastDynamicSite`3<object,int32,bool>::Invoke(!0, !1)
9 JUMP_IF_FALSE 8 (to 20)	brfalse <dest>
12 POP_TOP	
13 LOAD_CONST 1 (1)	IL_0015: ldsfld object __main__::c0PST04000002
16 RETURN_VALUE	IL_001a: ret

Dynamic Sites



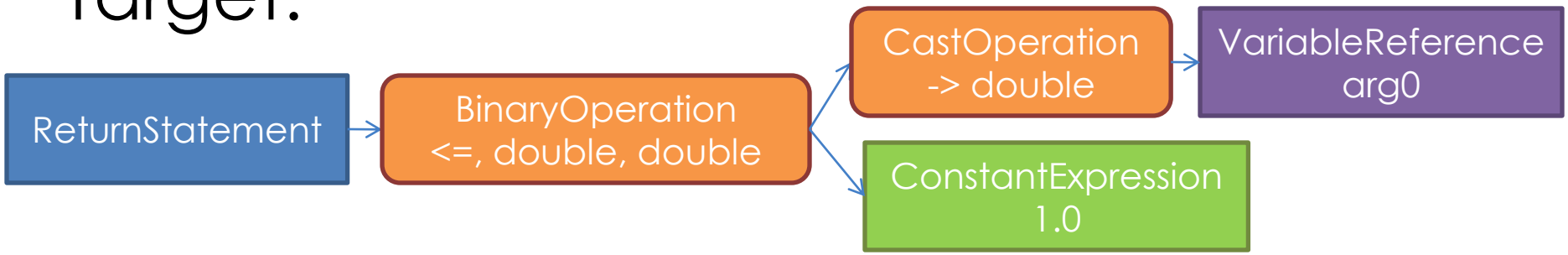
Dynamic Sites



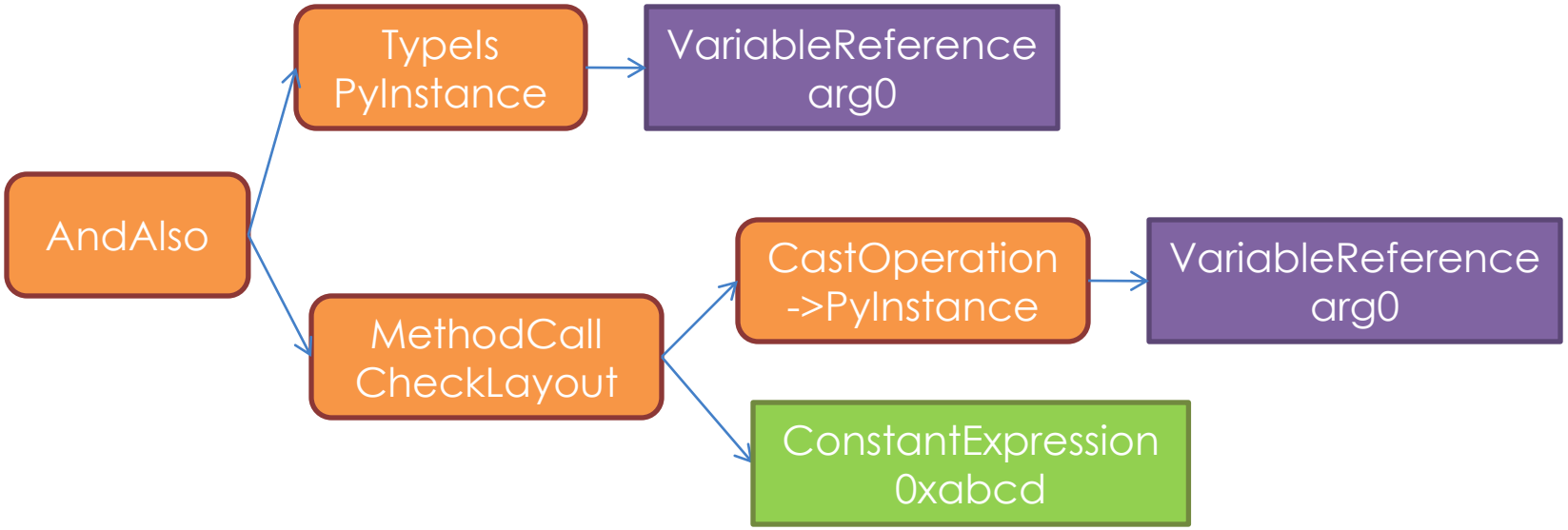
Rules

Test:

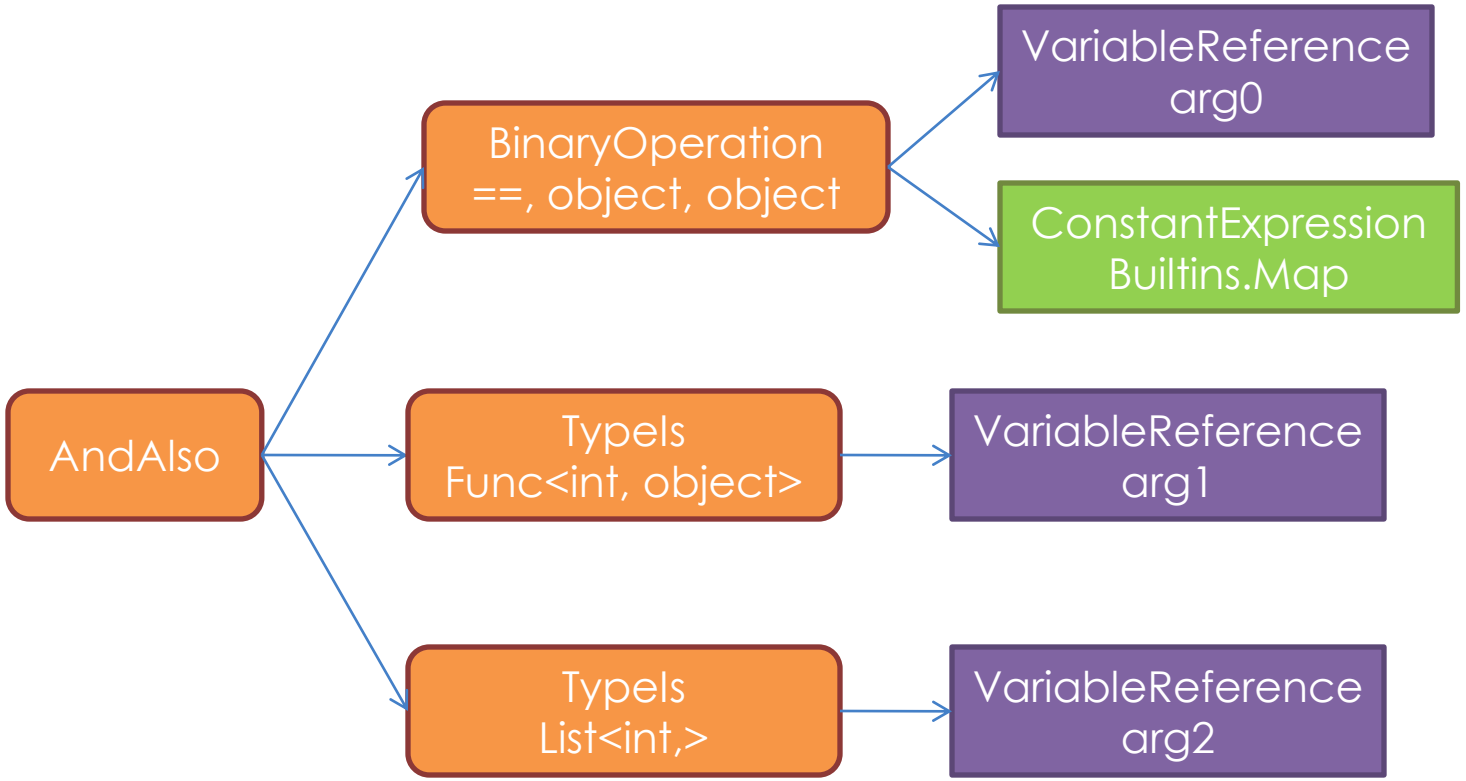
Target:



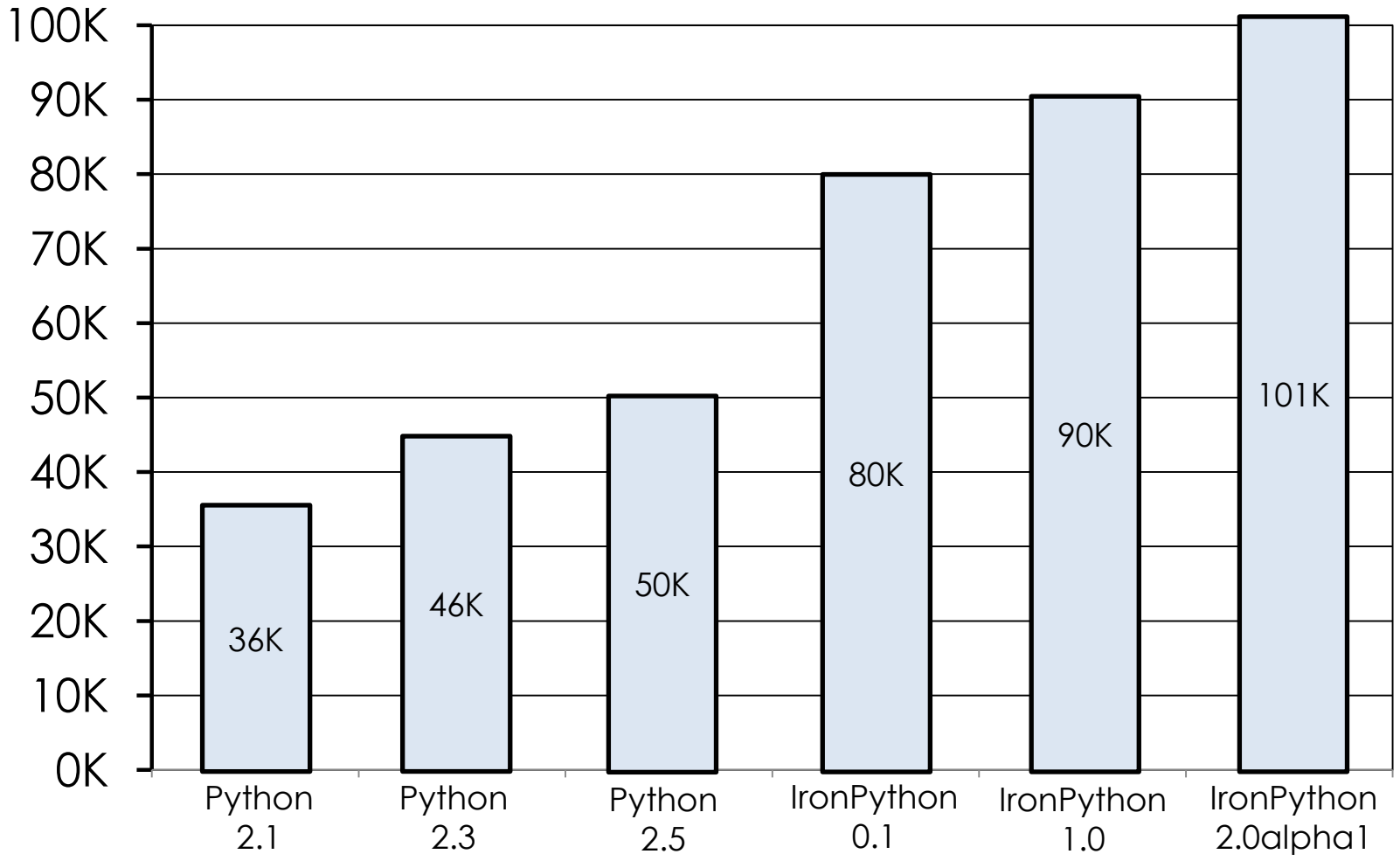
Language-Specific Tests



Experimental Optimizations



Standard Pystone Benchmark



Microsoft ASP.net

Windows Vista™

• • •

Microsoft .net

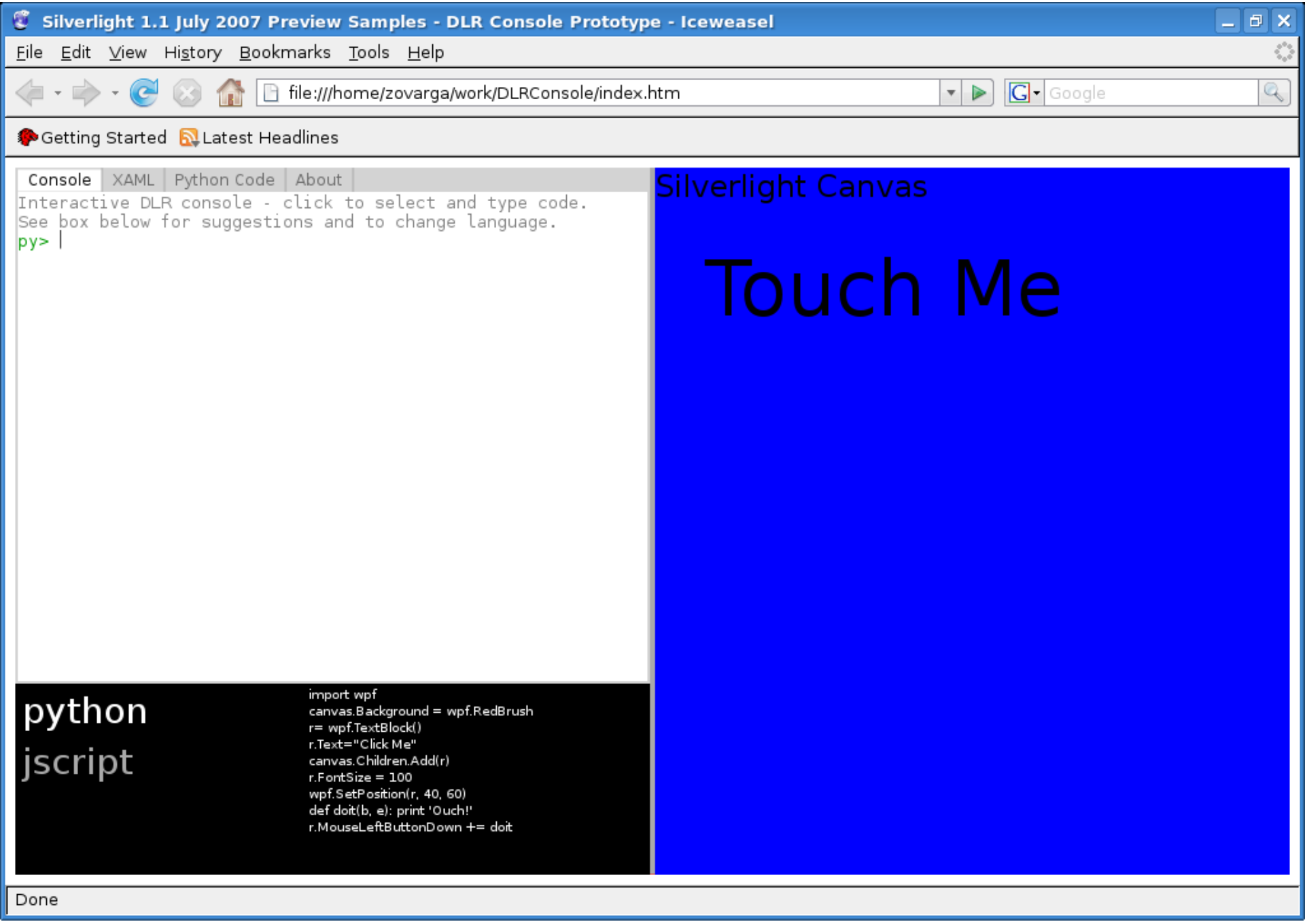
Microsoft Silverlight™

Microsoft Exchange Server 2007

Microsoft Robotics Studio

xna Game Studio Express
your world. your game.

My Language in the Browser


Silverlight 1.1 July 2007 Preview Samples - DLR Console Prototype - Iceweasel
File Edit View History Bookmarks Tools Help
file:///home/zovarga/work/DLRConsole/index.htm
Google
Getting Started
Latest Headlines
Console
XAML
Python Code
About
Interactive DLR console - click to select and type code.
See box below for suggestions and to change language.
py>
python
jscript
import wpf
canvas.Background = wpf.RedBrush
r= wpf.TextBlock()
r.Text="Click Me"
canvas.Children.Add(r)
r.FontSize = 100
wpf.SetPosition(r, 40, 60)
def doit(b, e): print 'Ouch!'
r.MouseLeftButtonDown += doit
Silverlight Canvas
Touch Me
Done

Why DLR?

- Reduce the engineering barriers
  - Let someone else do that for you
- Encourage sharing of libraries
  - Don't keep reinventing the wheel

- Focus on what's unique to you

Questions?

Lang.NET Conference
January 28-30, 2008
http://langnetsymposium.com
Microsoft Campus - Redmond, WA

http://codeplex.com/ironpython
dlr@microsoft.com