IronPython: Bringing the dynamic world to the CLR

Carlos Alberto Cortez
calberto.cortez@gmail.com

FOSDEM, Brussels, Feb/2012

Overview

- Mono/CLR
- Python
- IronPython
  - And how it relates to CPython/Mono

Demo Time!

mono Cross platform, modern development framework

- Garbage collection
- JIT compilation
- Thread management
- Desktop and Web support
- Huge class library
- Multi language support

C# High level, multi paradigm, object oriented, evolving language

2.0 Generics/Iterators/Anon. methods

3.0 LINQ/Lambdas

4.0 Dynamic support

5.0 Asynchrous methods

So why use other programming languages, given that C# is getting better and better?

Paradigm

Static typing

Scripted

Functional

Metaprogramming

Low level

Logic

Dynamic

python General purpose, multi paradigm, clear-syntax focused, **dynamic** language

(or the python.org definition:
“Language that lets you work more quickly and integrate your systems more effectively”)

python paradigms

dynamic

object oriented

metaprogramming

scripted

functional

* extra extensions

python zen

Minimalist philosophy and readibility. Avoid the “There's more than one way to do it”

python goodies

New modules in Python/C/C++

Web development

Embedded in several applications

Less boilerplate than other lang.

Complete standard library

Introspection

python hello world

```
print “Hello World”
```

python hello file

```
with open ('spam.txt', 'w') as file:
    file.write ('Spam and eggs!')
```

python OSS

PyGtk/PyQt
Bazaar
BitTorrent
Mercurial
Ubuntu Software Center
YUM
Mailman
Twisted

IronPython Open source implementation of Python on top of CLR

Created by **Jim Hugunin**, who had previously created **Jython** (Python on top of Java), while trying to write an article called **“Why .NET is a terrible platform for dynamic languages”**

IronPython Highlights

Mantained by Microsoft until version 2.7
Released under Apache 2.0 licence
Entirely written in C#
Running on top of a dynamic platform

Same syntax as the standard implementation

Run python modules **Run IL assemblies**

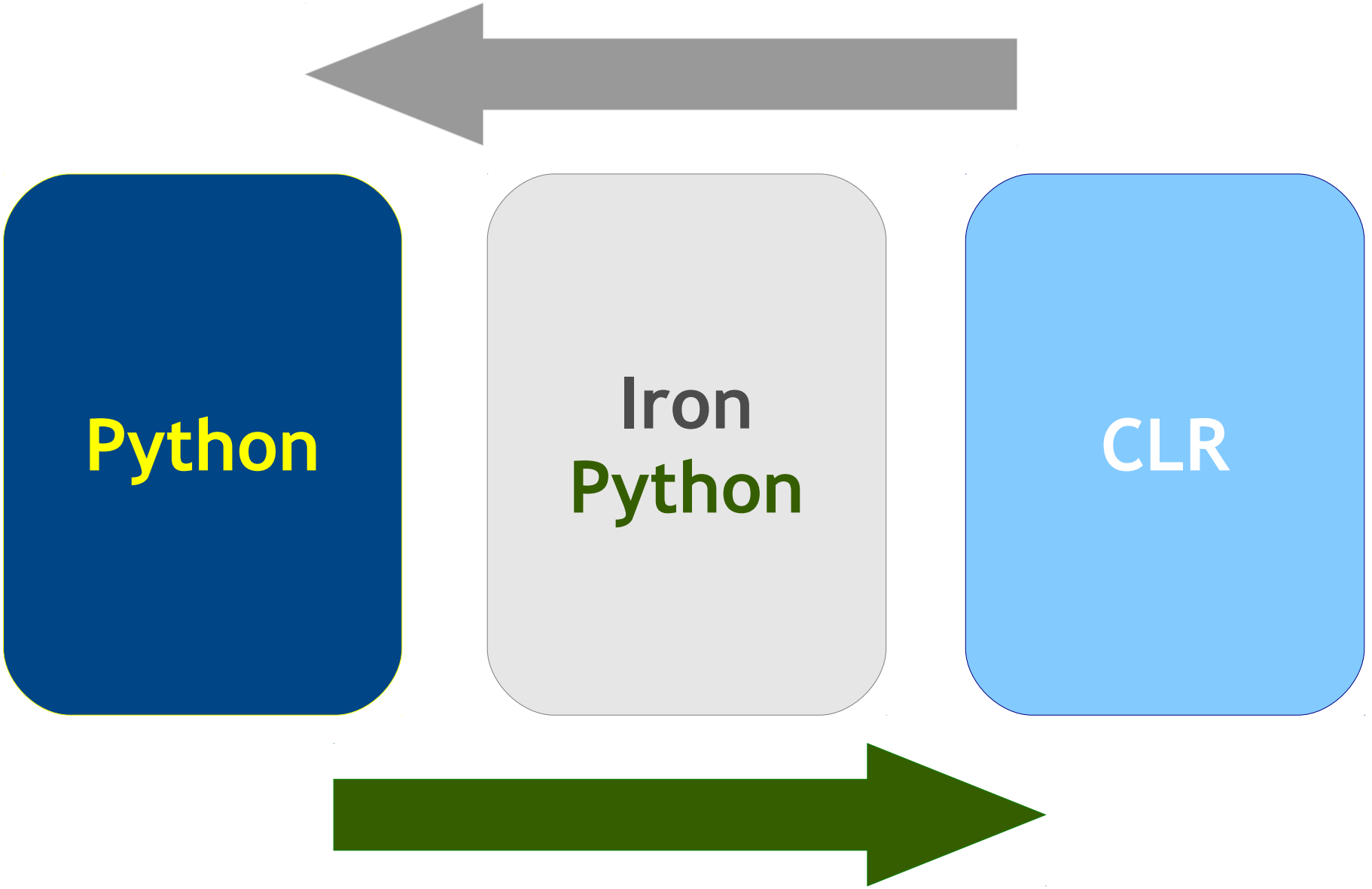
A script using the standard Python API

```
import socket

HOST = '127.0.0.1'
PORT = 50007
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind((HOST, PORT))
s.listen(1)

conn, addr = s.accept()
print 'Connected by', addr

data = conn.recv(1024)
conn.send(data)
conn.close()
```

A script using Gtk#

```
import clr
clr.AddReferenceByPartialName ("gtk-sharp")

from Gtk import *

class MainWindow (Window):

    def __init__ (self, Title):
        vbox = VBox ()
        self.Add (vbox)

        b = Button (Label = "Click me")
        vbox.PackStart (b, True, True, 0)

        vbox.ShowAll ()

    def OnDeleteEvent (*args):
        Application.Quit ()
```

IronPython Two worlds



DLR Dynamic Language Runtime

Set of services for dynamic languages to run and interop with the CLR

- No lexer/parser, but **expression trees**
- Dynamic type system
- Dynamic code generation
- **Interoperation with static typed langs.**
- Hosting API

DUCK TYPING

When I see a bird that *walks like a duck*, *swims like a duck*, And *quacks like a duck*, I call that bird a **duck**

DUCK TYPING

```
def DoQuack (duck):
    duck.Quack ()

class Duck (object):
    def Quack (self):
        “duck quacks!”

class AnotherBird (object):
    def Quack (self):
        “Not-a-bird quacks!”

DoQuack (Duck ())
DoQuack (AnotherBird ())
```

IronPython What for?

Embedded (Scripting) + **Unit Testing** + **Prototyping**

In my experience, it's very helpful to create models in a dynamic language, because there is a very low barrier to redesigning as you learn. You're able to quickly try out your ideas.

Guido Van Rossum

Hosting Simple expression

```
using System;
using IronPython.Hosting;

class MainClass
{
    public static void Main (string[] args)
    {
        var engine = Python.CreateEngine ();
        var source = engine.CreateScriptSourceFromString ("3.1416 * 2.0 - 13.8");
        double res = source.Execute<double> ();

        Console.WriteLine (res);
    }
}
```

Hosting Simple expression 2

```
class MainClass
{
    public static void Main (string[] args)
    {
        var engine = Python.CreateEngine ();
        var scope = engine.Runtime.CreateScope ();
        scope.SetVariable ("p", new Product () { Name = "MonoTester" });

        var source = engine.CreateScriptSourceFromString ("print p.Name");
        source.Execute (scope);
    }
}

public class Product
{
    public string Name { get; set; }
    public int Id { get; set; }
}
```

Hosting Dynamic

```
using System;
using IronPython.Hosting;

class MainClass
{
    public static void Main (string[] args)
    {
        var engine = Python.CreateEngine ();
        var runtime = engine.Runtime;
        dynamic pythonmod = runtime.UseFile ("/tmp/pysample.py");

        pythonmod.Simple ();
    }
}
```

```
def Simple ():
    print “hello from Python!”
```