15.–18. 09. 2008
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

IronPython und die Dynamic Language Runtime (DLR)

Softwareentwicklung mit Python auf der Microsoft Windows Plattform

Klaus Rohe (klrohe@microsoft.com)

Microsoft Deutschland GmbH

Einige Anmerkungen

- Kein Python Tutorial:
  - Es werden einige Unterschiede von Python zu C# / Java dargestellt
- Die Eigenschaften, Beispiele und Demos beziehen sich auf die Version IronPython 2.0 Beta 4. Die endgültig freigegebene Version von IronPython 2.0 kann andere Eigenschaften besitzen und sich anders verhalten.

Agenda

- Python
- IronPython & die Dynamic Language Runtime
- Benutzung von .NET Bibliotheken in IronPython
  - ADO.NET
  - WPF
- Einbetten von IronPython in Applikationen
- Interoperabilität
  - IronPython & Cpython
  - Interoperabilität CPython - .NET

Historie von Python

- Anfang der 1990er Jahre von Guido van Rossum am Zentrum für Mathematik und Informatik Amsterdam als Nachfolger für die Lehrsprache **ABC** entwickelt
  - Wurde ursprünglich für das verteilte Betriebssystem **Amoeba** entwickelt: http://www.cs.vu.nl/pub/amoeba/

- Der Name der Sprache bezieht sich auf die britische Komiker Gruppe ***Monty Python***

Charakterisierung von Python (1)

- Python-Programm wird interpretiert
- Unterstützt mehrere Programmierparadigmen:
  - Prozedural
  - Objektorientiert
  - Funktional
- Klare, einfache Syntax
- Portabel, verfügbar auf allen Plattformen mit C-Compiler
  - Windows, Mac OS, UNIX, LINUX, …
- Leicht erweiterbar mit C/C++ Bibliotheken
  - Z. B. Bibliothek für Win32 und COM

Charakterisierung von Python (2)

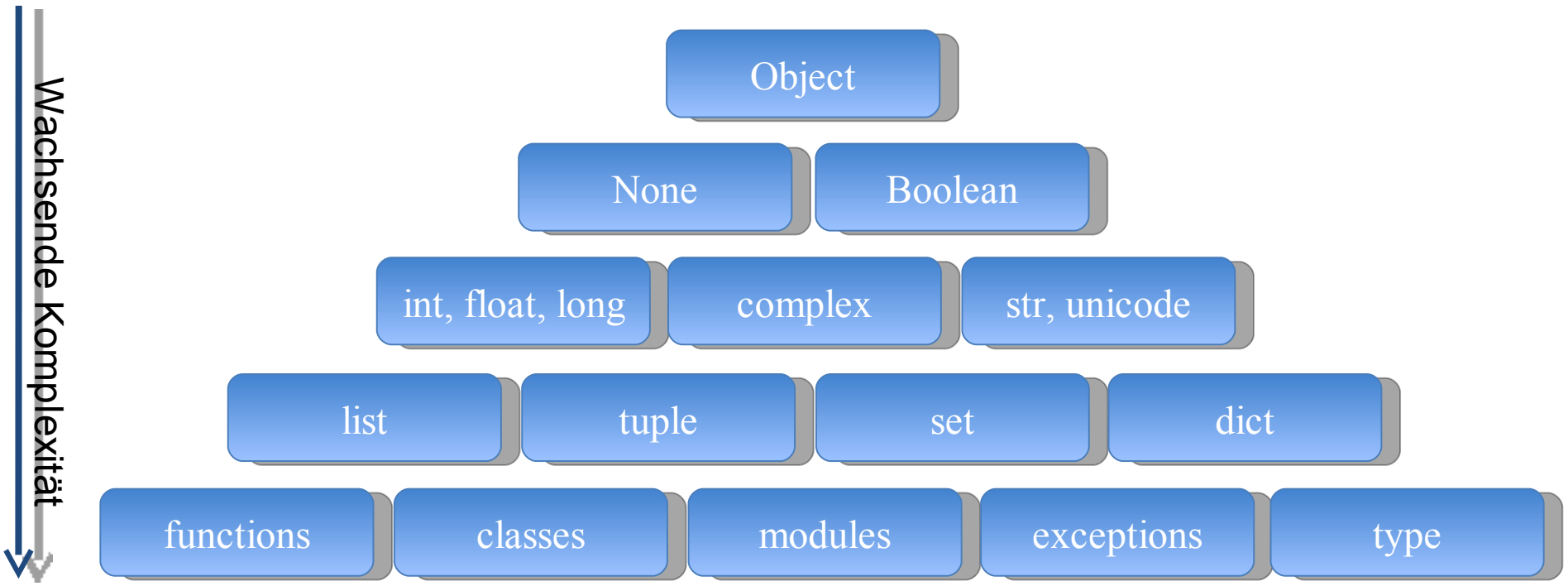
- Python wird als dynamische Programmiersprache bezeichnet:

> - **Dynamic programming language** is a term used broadly in computer science to describe a class of high level programming languages that execute at runtime many common behaviors that other languages might perform during compilation, if at all. These behaviors could include extension of the program, by adding new code, by extending objects and definitions, or by modifying the type system, all during program execution. These behaviors can be emulated in nearly any language of sufficient complexity, but dynamic languages provide direct tools to make use of them.
> - Dynamic languages and dynamic typing are not identical concepts, and a dynamic language need not be dynamically typed, though many dynamic languages *are* dynamically typed.

http://en.wikipedia.org/wiki/Dynamic_programming_language

- Python besitzt dynamische Typprüfung

Python Datentypen



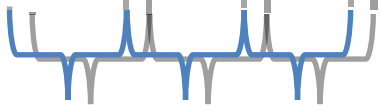
Vordefinierte Datentypen in Python

Typ	Beispiele
str	‘hello world’, “Python”
unicode	u’a unicode string’
int	-3, 3, 4711
long	662473726694923701112012 8L, 7L
float	0.0, -3.1, 2e100, 2.76e-10
complex	8j, 2 + 3j
list	[], [1, 2, 3, 4, 5], [1.0, “xxx”, 3]
tuple	(), (1,), (1, 2, 3)
dict	{}, {'a': 12, 'b': 'xxx'}
set	set(), set([1, 2, 3, 4])
NoneType	None
bool	True, False

Eigenarten der Python-Syntax

- Python erzwingt eine sehr lesbare Formatierung der Programme.
- Ohne „Tabs“ Syntaxfehler!

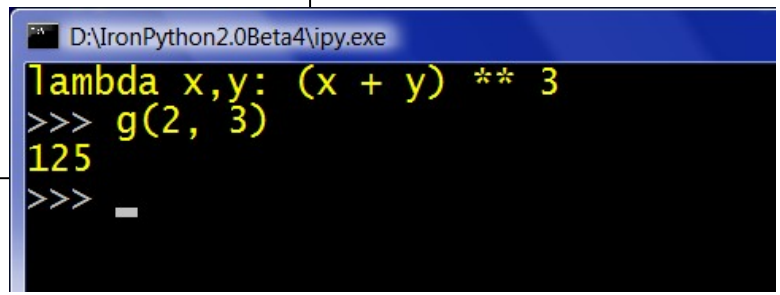
```
1  def bubblesort(ar):
2      l = len(ar) - 1
3
4      for i in range(l):
5          for j in range(l-i):
6              if ar[j] > ar[j+1]:
7                  temp = ar[j]
8                  ar[j] = ar[j+1]
9                  ar[j+1] = temp
10
11     return len(ar)
```



„Tabs“

Python: funktional und dynamisch

```
# Generierung von Funktionen aus lambda-Ausdrücken

def funcgen(varlist, expr):
    lambdastr = 'lambda '

    i = 0
    for v in varlist:
        if i == 0:
            lambdastr = lambdastr + v
        else:
            lambdastr = lambdastr + ',' + v
        i = i  + 1

    lambdastr = lambdastr + ': ' + expr
    print lambdastr
    return eval(lambdastr)

g = funcgen(['x', 'y'], '(x + y) ** 3')
```



Gründe für Python

- Warum Python?
  - Einfach zu lernen und zu benutzen
    - => *steile Lernkurve*
  - Erlaubt schnelle Entwicklung und Prototyping
    - => *geringe Entwicklungszeit*
  - Viel Frameworks und Klassenbibliotheken
    - Web-Anwendungen (Django, …)
    - Technisch-wissenschaftliche Anwendungen (PIL, Scipy, Sympy, …)
    - Skriptsprache für Anwendungen wie: GIMP, Blender, ArcGIS, …

Python Implementierungen

- Das original Python wird auch als CPython bezeichnet (in C implementiert)
- Jython eine Implementierung von Python in Java für die Java JVM
  - http://www.jython.org/Project/contributors.html
- PyPy ist eine Implementierung von Python in Python
  - http://codespeak.net/pypy/dist/pypy/doc/home.html
- **IronPython** Implementierung von Python in C# für .NET

Etwas Statistik

TIOBE Programming Community Index for September 2008

Position Sep 2008	Position Sep 2007	Delta in Position	Programming Language	Ratings Sep 2008	Delta Sep 2007	Status
1	1	=	Java	20.715%	-0.99%	A
2	2	=	C	15.379%	+0.47%	A
3	5	↑↑	C++	10.716%	+0.78%	A
4	3	↓	(Visual) Basic	10.490%	-0.26%	A
5	4	↓	PHP	9.243%	-0.96%	A
6	8	↑↑	Python	5.012%	+1.99%	A
7	6	↓	Perl	4.841%	-0.58%	A
8	7	↓	C#	4.334%	+0.75%	A
9	9	=	JavaScript	3.130%	+0.41%	A
10	14	↑↑↑↑	Delphi	3.055%	+1.83%	A
11	10	↓	Ruby	2.762%	+0.70%	A
12	13	↑	D	1.265%	-0.11%	A
13	11	↓↓	PL/SQL	0.700%	-1.16%	A--
14	12	↓↓	SAS	0.640%	-0.76%	B
15	23	↑↑↑↑↑↑↑↑	ActionScript	0.472%	+0.07%	B
16	16	=	Lisp/Scheme	0.419%	-0.21%	B
17	18	↑	Lua	0.415%	-0.16%	B
18	22	↑↑↑↑	Pascal	0.400%	-0.03%	B
19	-	↑↑↑↑↑↑↑↑↑↑	PowerShell	0.384%	0.00%	B
20	17	↓↓↓	COBOL	0.360%	-0.27%	B

Category	Ratings September 2008	Delta September 2007
Object-Oriented Languages	57.6%	+4.7%
Procedural Languages	39.9%	-3.4%
Functional Languages	1.9%	-0.2%
Logical Languages	0.7%	-1.1%

Category	Ratings September 2008	Delta September 2007
Statically Typed Languages	59.1%	+2.0%
Dynamically Typed Languages	40.9%	-2.0%

- A mainstream languages
- B < 0.7%

http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

Agenda

- Python
- IronPython & die Dynamic Language Runtime
- Benutzung von .NET Bibliotheken in IronPython
  - ADO.NET
  - WPF
- Einbetten von IronPython in Applikationen
- Interoperabilität
  - IronPython & Cpython
  - Interoperabilität CPython - .NET

IronPython: ein Rückblick

ActiveState
Dynamic Tools for Dynamic Languages

How could Microsoft have screwed up so badly that the CLR is far worse
VM for
ages?
ses.

Let me check this out. I will write a short paper -

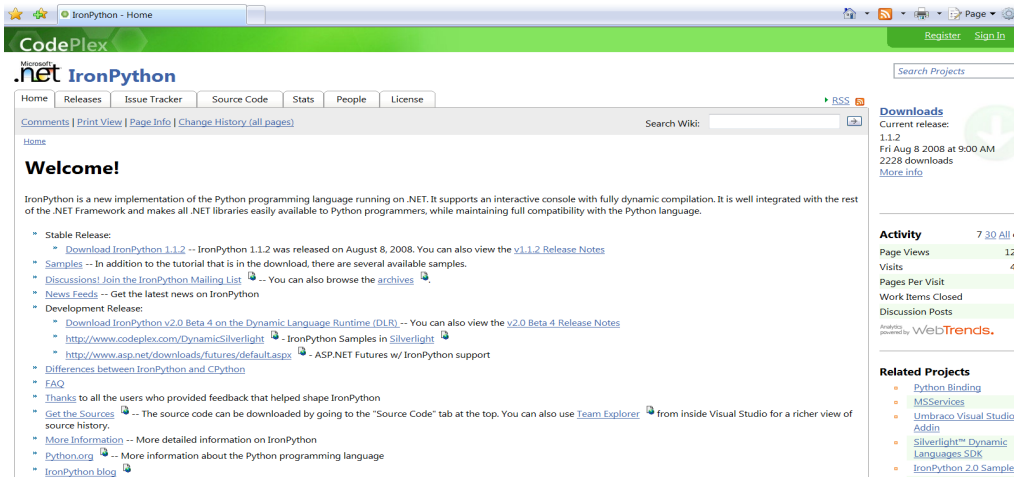
John Udell

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

Jim Hugunin

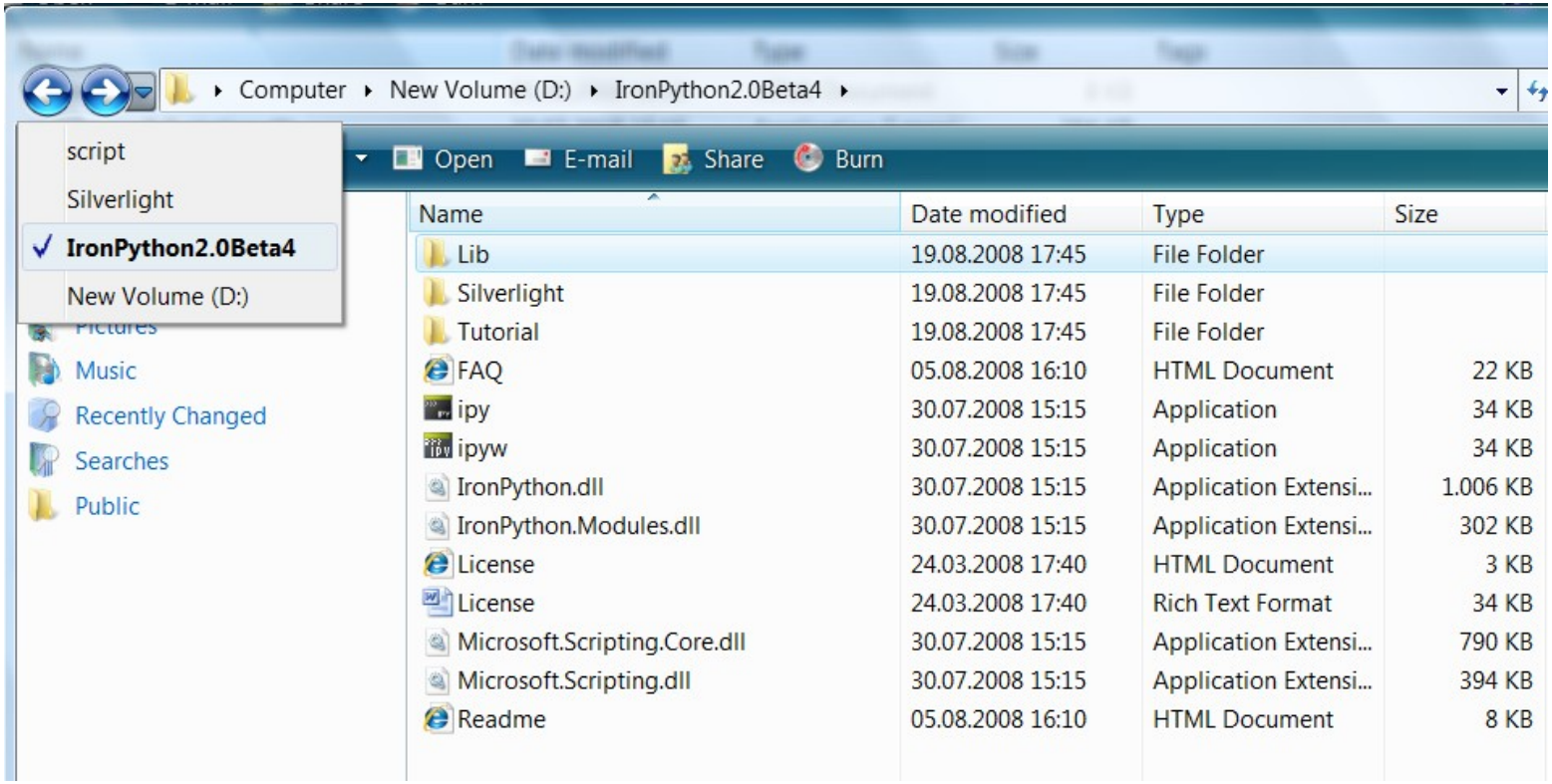
IronPython Versionen

- IronPython 1.x
  - Aktuelle Version: IronPython 1.1.2
- IronPython 2.0
  - IronPython 2.0 ist mit der Dynamic Language Runtime (DLR) implementiert
  - Aktuelle Version: IronPython 2.0 Beta 4
- http://www.codeplex.com/IronPython



IronPython (1)

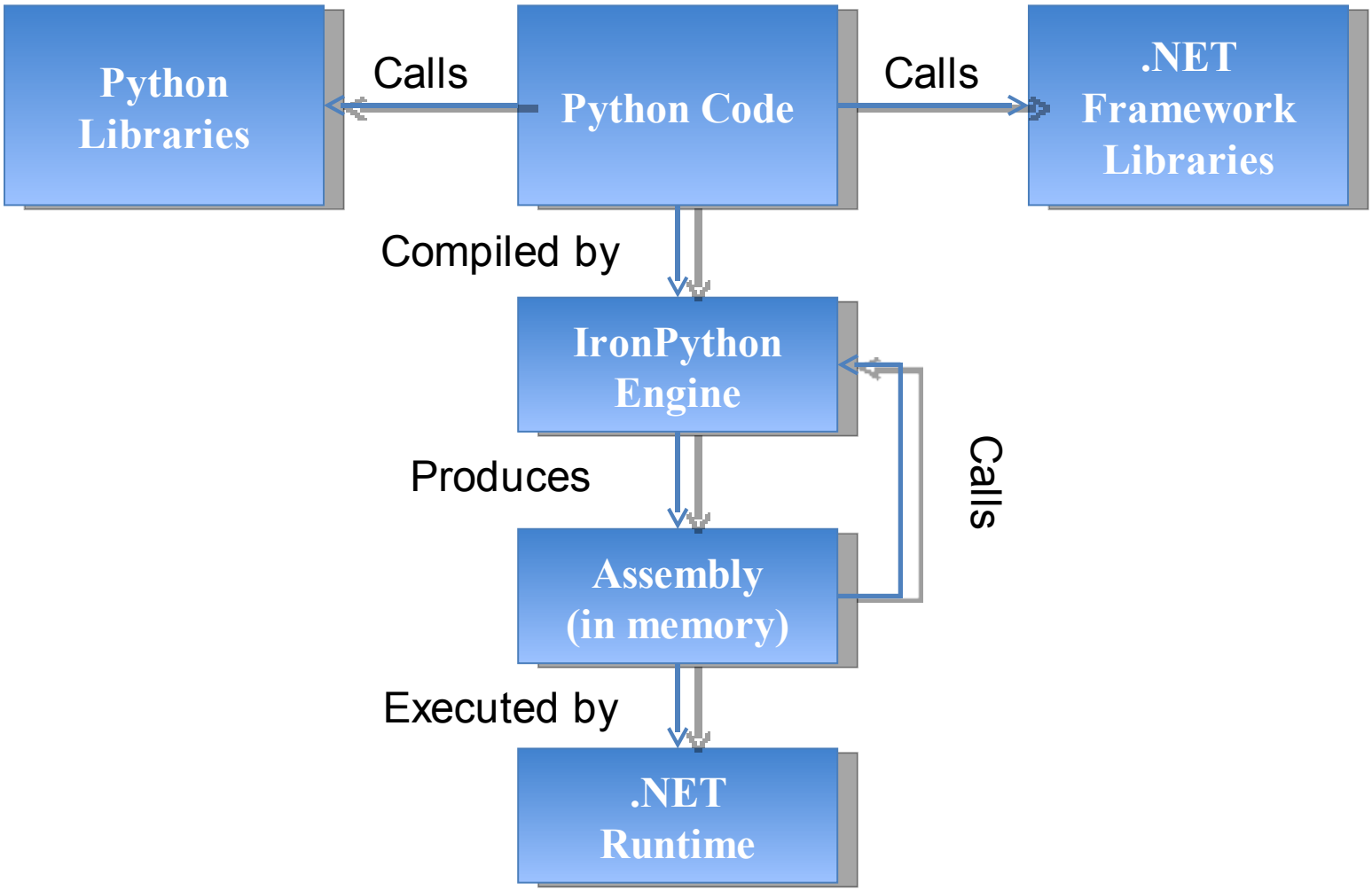
- IronPython ist als zip-File, msi-Installationspaket oder als Quellcode erhältlich.
- Verzeichnisstruktur IronPython 2.0:



IronPython (2)

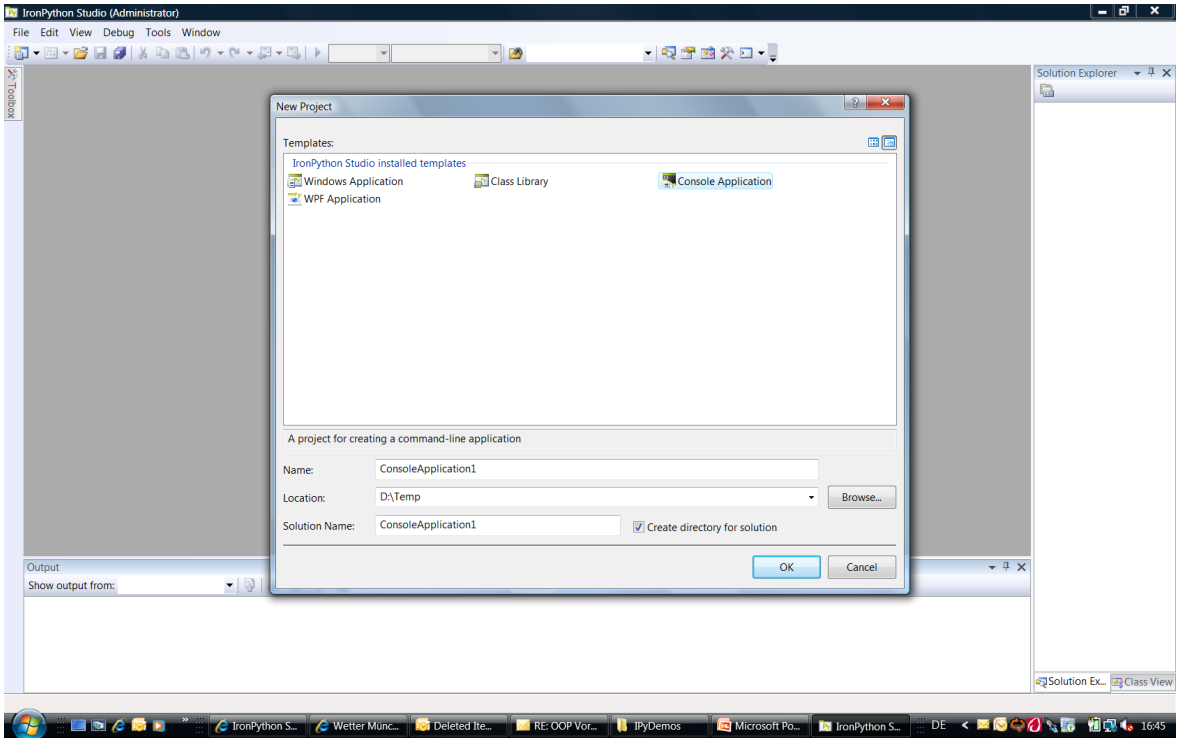
- ipy.exe:
  - Ausführen von IronPython-Skripts mit interaktiver Console
- ipyw.exe:
  - Ausführen von IronPython-Skripts ohne interaktive Console
- Umgebungsvariable:
  - IRONPYTHONPATH:
    - Suchpfad für Python-Module
  - IRONPYTHONSTARTUP:
    - Suchpfad für Startup-Module

IronPython Ausführungsmodell



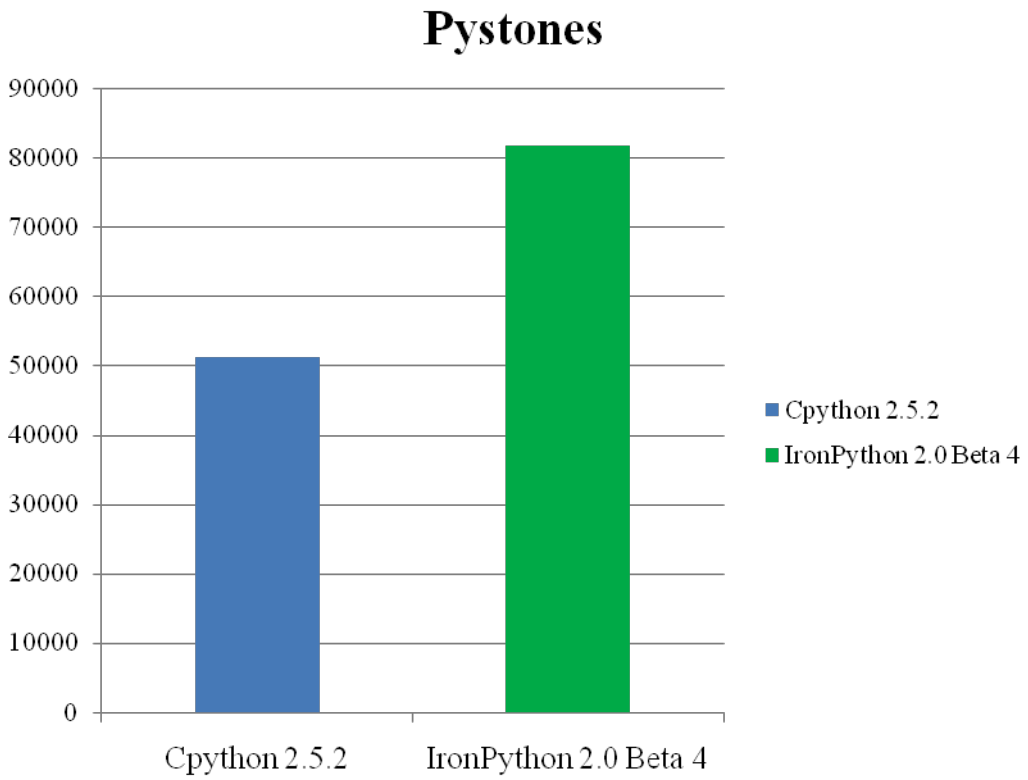
Entwicklungsumgebung: IronPython Studio

- Download: http://www.codeplex.com/IronPythonStudio
- Mit dem VS 2008 SDK implementiert



IronPython performance: Pystones/second

- Venerable benchmark
  - Dhrystone in Python
  - In Lib/test/pystone.py
  - Only benchmark that ships with Python
- Simple but non-trivial
  - ~200 lines of code
  - Most basic Python ops
  - Very little OO
- 32-bit Vista Enterprise Service Pack 1.0
  - .NET 3.5 SP1 (2.0.50727.3053)
  - AMD x64 X2 4200+4.00 GB RAM
  - IronPython 2.0 Beta 4



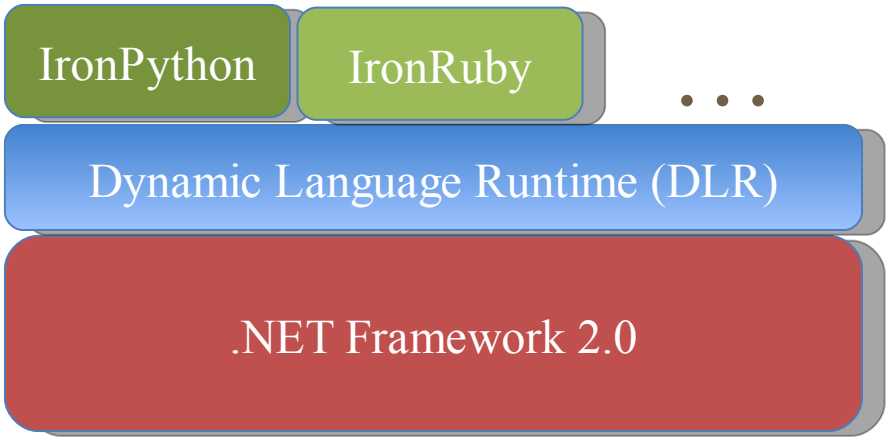
	Pystones
Cpython 2.5.2	51275,5
IronPython 2.0 Beta 4	81788,3
Ipy / Cpy =	1,595

Demo: IronPython

- Commandozeile:
  - IronPython Console: ipy.exe
  - IronPython Skriptr ausführen: ipyw.exe
- IronPython Studio

Dynamic Language Runtime (1)

- Plattform zur Implementierung von dynamischen Sprachen auf .NET
  - Basiert auf Erfahrungen von Jim Hugunin‘s bei der Implementierung von IronPython 1.0 auf .NET
- Die DLR ist ein Aufsatz auf dem .NET Framework und wurde in C# implementiert.



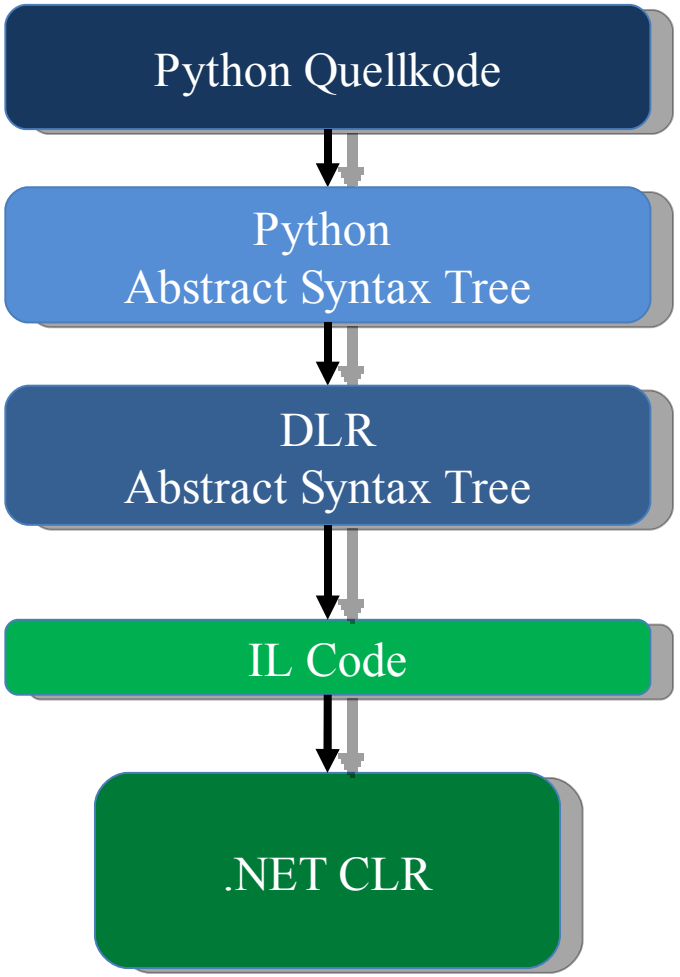
Dynamic Language Runtime (2)

- Nutzt die Dienste und Eigenschaften des .NET Frameworks:
  - Garbage collector
  - Just-in-time Compiler (JIT)
  - .NET Bibliotheken
  - Werkzeuge

- Vereinfachte Beschreibung der DLR:
  - Gegeben ist ein abstrakter Syntaxbaum (AST). Die DLR führt die Operationen und Konstrukte des Syntaxbaums aus, unter Berücksichtigung der Operationen und Regeln, welche für die Sprache definiert sind. Basisoperationen werden auf CLR Default-Operationen zurückgeführt (Integer-Operationen, …)

Dynamic Language Runtime (3)

- Python Quellkode wird in einen „Python“-AST transformiert
- „Python“-AST wird in DLR-AST transformiert
- DLR-AST wird in IL (Intermediate Language) übersetzt, mittels CODEDOM, und von der CLR ausgeführt.



Zusammenfassung DLR

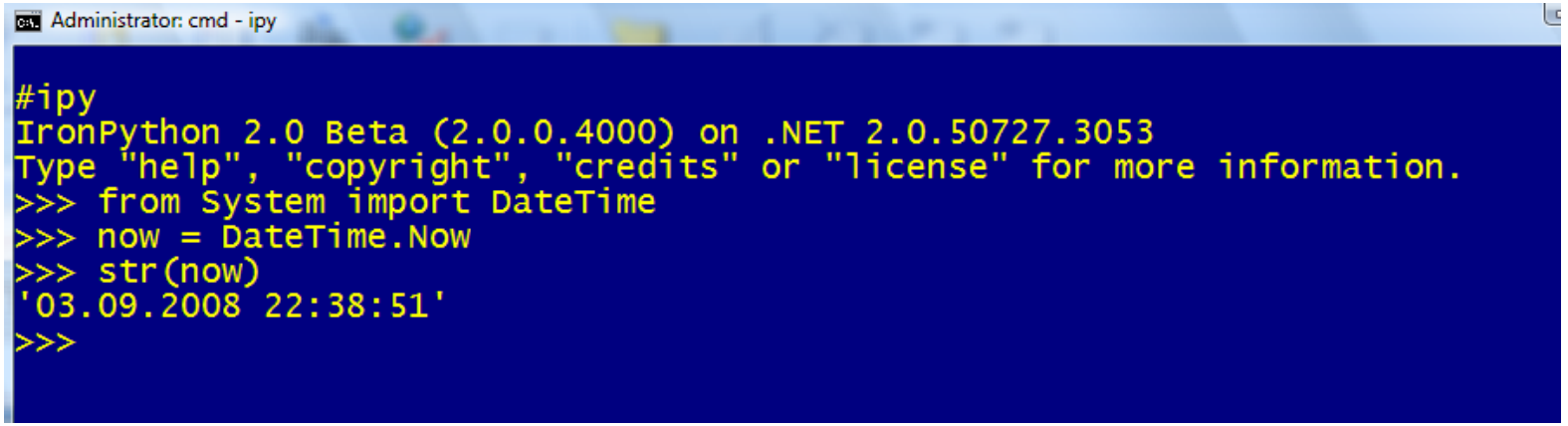
- Die Dynamic System Runtime (DLR) ist die Grundlage für die effiziente und performante Implementierung dynamischer Sprachen mit .NET
- Die DLR bietet Werkzeuge:
  - Zum eigentlichen Implementieren der Sprache, wie Erzeugen von Abstrakten Syntaxbäumen und effiziente Codegenerierung für die .NET CLR
    - Erzeugung von „Sprach unabhängigen“ ASTs
  - Einbetten der Sprache in eigenen Applikationen (Skriptconsolen, …)
- Die Implementierung der DLR befindet sich in den Assemblies:
  - `Microsoft.Scripting.dll`
  - `Microsoft.Scripting.Core.dll`

Agenda

- Python
- IronPython & die Dynamic Language Runtime
- Benutzung von .NET Bibliotheken in IronPython
  - ADO.NET
  - WPF
- Einbetten von IronPython in Applikationen
- Interoperabilität
  - IronPython & Cpython
  - Interoperabilität CPython - .NET

.NET Bibliotheken in IronPython (1)

- In IronPython können sämtliche Klassenbibliotheken des .NET Frameworks benutzt werden.
- In IronPython können die .NET Namensräume von geladenen Assemblies wie Python-Pakete behandelt werden



.NET Bibliotheken in IronPython (2)

- .NET Assemblies werden wie folgt bekannt gemacht:

```
import clr
clr.AddReference("System.Data")
```

- IRONPYTHONPATH muss die Verzeichnisse enthalten, in denen sich die referenzierten Assemblies befinden
- ***clr*** ist ein in IronPython eingebauter Modul, der Methoden zum Referenzieren von .NET Assemblies bereitstellt.
- clr.References gibt eine Liste aller existierenden Referenzen aus.

Datenbankzugriff mit ADO.NET in IronPython

```
import clr
clr.AddReference("System.Data")

from System.Data import *
from System.Data.SqlClient import *

constr = 'Integrated Security=SSPI;Initial Catalog=acronymdb;'
constr = constr + 'Data Source=KLROHE04\SQLEXPRESS;Workstation ID=KLROHE04'

con = SqlConnection(constr)
cmdstr = 'SELECT acrn, meaning FROM acronym WHERE acrn LIKE @SEARCHSTR ORDER BY 1'
selcmd = SqlCommand(cmdstr, con)
selcmd.Prepare()
selcmd.Parameters.Clear()
selcmd.Parameters.AddWithValue('@SEARCHSTR', 'A%')

con.Open()
rd = selcmd.ExecuteReader()

i = 0
while rd.Read():
     i = i + 1
     print rd.GetString(0) + '\t' + rd.GetString(1)

print str(i) + ' records retrieved'
```

Windows Presentation Foundation (WPF) & IronPython

```
# Reference the WPF assemblies
import clr
clr.AddReferenceByPartialName("PresentationCore")
clr.AddReferenceByPartialName("PresentationFramework")
import System.Windows

# Initialization Constants
Window = System.Windows.Window
Application = System.Windows.Application
Button = System.Windows.Controls.Button
StackPanel = System.Windows.Controls.StackPanel
Label = System.Windows.Controls.Label
Thickness = System.Windows.Thickness

# Create window
my_window = Window()
my_window.Title = 'Welcome to IronPython'
# Create StackPanel to Layout UI elements
my_stack = StackPanel()
my_stack.Margin = Thickness(15)
my_window.Content = my_stack
```

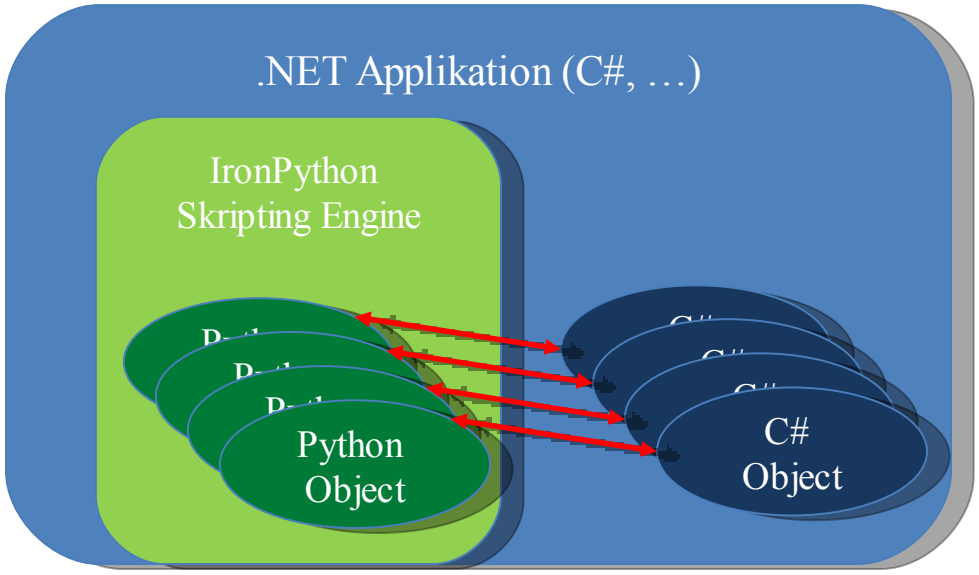
```
# Create Button and add a Button Click event handler
my_button = Button()
my_button.Content = 'Push Me'
my_button.FontSize = 24


def clicker(sender, args):
   # Create new label
   my_message = Label()
   my_message.FontSize = 48
   my_message.Content = 'Welcome to IronPython!'
   # Add label into stack panel of controls
   my_stack.Children.Add (my_message)

my_button.Click += clicker
my_stack.Children.Add (my_button)

# Run application
my_app = Application()
my_app.Run(my_window)
```

Agenda

- Python
- IronPython & die Dynamic Language Runtime
- Benutzung von .NET Bibliotheken in IronPython
  - ADO.NET
  - WPF
- Einbetten von IronPython in Applikationen
- Interoperabilität
  - IronPython & Cpython
  - Interoperabilität CPython - .NET

Einbetten von IronPython in Appliaktionen

- Warum IronPython einbetten:
  - Ausführbares Programm um eine IronPython Applikation bauen
  - Interessanter:
    - Applikation mit Skript-Fähigkeit ausstatten
    - Applikation durch IronPython erweiterbar machen



Agenda

- Python
- IronPython & die Dynamic Language Runtime
- Benutzung von .NET Bibliotheken in IronPython
  - ADO.NET
  - WPF
- Einbetten von IronPython in Applikationen
- Interoperabilität
  - IronPython & Cpython
  - Interoperabilität CPython - .NET

IronPython & CPython

- Kann man die für CPython existierenden umfangreichen Softwarebibliotheken und Frameworks auch in IronPython nutzen?
- Antwort: Im Prinzip ja, aber …
- Bibliotheken/Frameworks, welche nur mit Python implementiert wurden, sollten mit IronPython funktionieren.
  - Beispiel: Django, Framework zur Web-Entwicklung, vergleichbar mit Rails.
  - PyCon 2008: Django funktioniert mit IronPython und MS SqlServer: http://www.infoq.com/news/2008/03/django-and-ironpython
- Schwieriger: Python Bibliotheken/Frameworks, die C/C++ Erweiterungen nutzen.
  - Das sind sehr viele!!

IronPython & CPython Frameworks mit C/C++ Erweiterungen

- CPython Frameworks mit C/C++ Erweiterungen können nicht direkt in IronPython benutzt werden

- IronClad Projekt:
  - http://www.resolversystems.com/documentation/index.php/Ironclad
  - „Ironclad is a project to let IronPython users use CPython C Extensions by acting as the glue between the IronPython interpreter and these extensions."

- Andere Lösungen:
  - Funktionalitäten über Web-Services verfügbar machen

  - ***Funktionalitäten über CORBA verfügbar machen:***
    - ***OmniOrb für Cpython***
    - ***IIOP.NET für Kommunikation von IronPython (.NET) mit dem CORBA Server***

Beipiel: Funktionalitäten von NumPy und SymPy in IronPython nutzen

- NumPy : http://www.scipy.org/NumPy
  - Mächtiges CPython Framework für Numerik, Lineare Algebra usw.
- SymPy: http://wiki.sympy.org/wiki/Main_Page
  - CPython Framework für Computeralgebra (CAS)
- Es sollen folgende Funktionalitäten für IronPython verfügbar sein:
  - Symbolisches Differenzieren, Integrieren, Reihenentwicklung einer Funktion
  - Num. Auswertung einer Funktion für ein Intervall mit def. Schrittweite

Lösung mit OmniOrb und IIOP.NET

- OmniOrb: Open Source CORBA Implementierung für Cpython
- IronPython nutzt IIOP.NET zur Kommunikation mit dem Server:
- IIOP.NET: Open Source Bibliothek zur Kommunikation mit CORBA oder J2EE Servern über IIOP
- Kommandozeilen-Werkzeug `IDLToCLSCompiler` zur Erzeugung eines .NET Assemblies, welches die IIOP Kommunikation kapselt

Details der Beispiellösung

```
// CAS.idl
module CAS
{
      typedef sequence<double> DoubleList;

      struct xyresult
      {
            DoubleList x;
            DoubleList y;
      };

      interface CASService
      {
            string derivative(in string fexpr, in string varname, in long order);

            string integral(in string fexpr, in string varname);
            string expand2series(in string fexpr, in string varname, in long order);

            xyresult evaluate(in string fexpr, in double start, in double end, in
double step);
      };
};
```

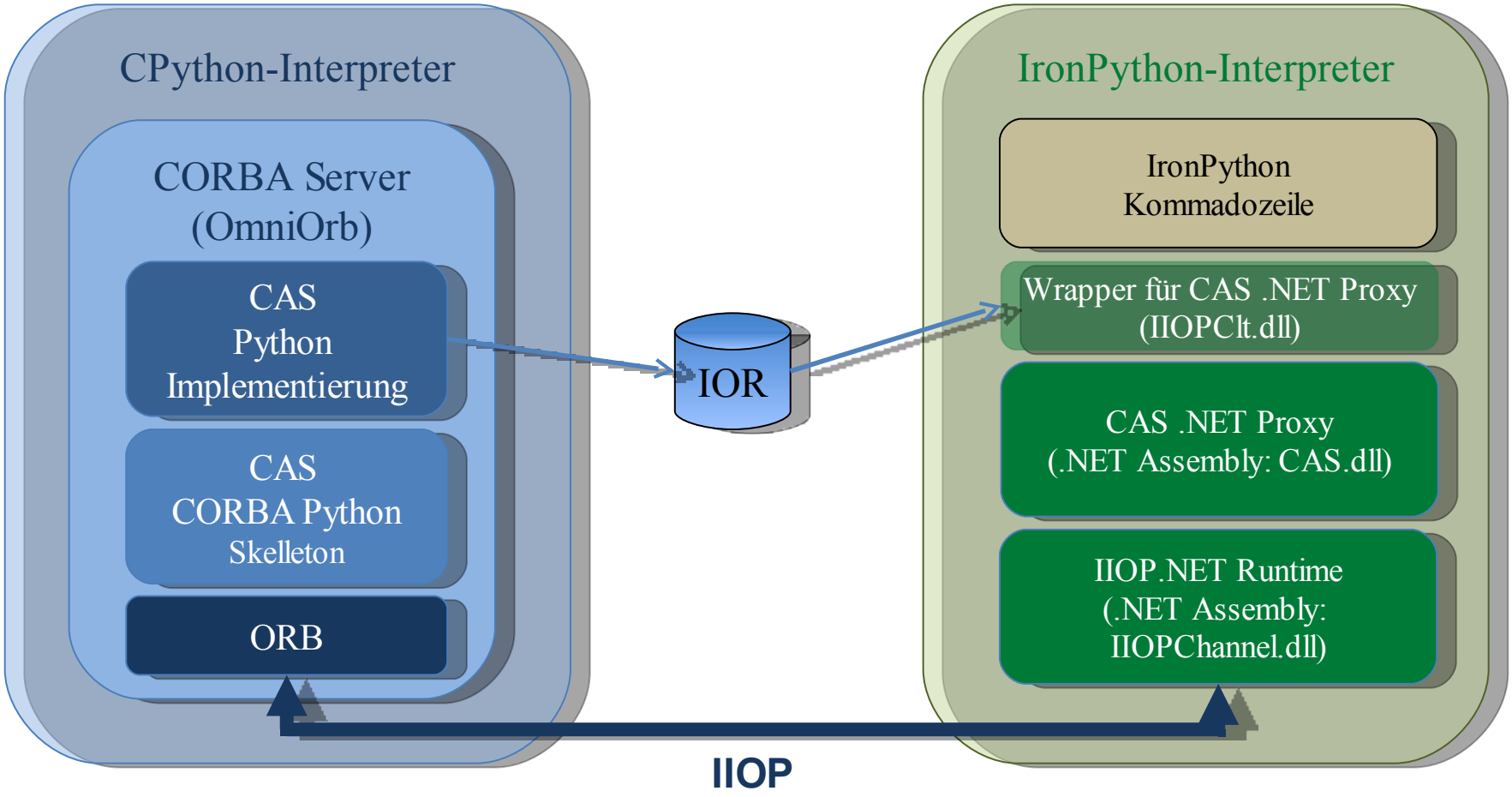
OmniOrb IDL-Compiler:
omniidl -bpython CAS.idl

IIOP.NET IDL-Compiler:
IDLToCLSCompiler CAS CAS.idl

CAS
CORBA Python
Skelleton

CAS .NET Proxy
für IIOP Kommunikation
mit dem CORBA Server

Architekturschaubild des Beispiels



The Python for Windows extensions

- Python Win32 Extensions
  - http://starship.python.net/crew/mhammond
  - Download: https://sourceforge.net/projects/pywin32/
- PyWin32 ist ein Python-Paket, welches es ermöglicht von Python-Skripts Win32- und COM-Schnittstellen zu nutzen
- Beispiel:
  - Interaktion eines Python-Skripts mit Excel, Word oder Outlook über die Office COM-Schnittstelle
  - Jede Komponente mit COM-Schnittstelle

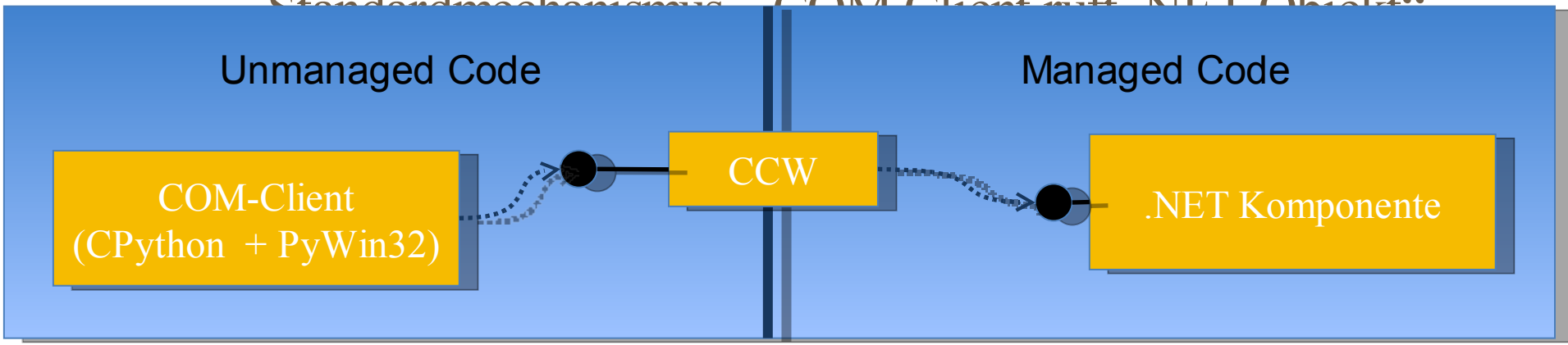
Beispiel: Python-Skript steuert Excel

```
from Tkinter import Tk
from time import sleep
from tkMessageBox import showwarning
import win32com.client as win32

warn = lambda app: showwarning(app, 'Exit?,)
RANGE = range(3, 8)

def excel():
    app = 'Excel'
    xl = win32.gencache.EnsureDispatch('%s.Application'
% app)
    ss = xl.Workbooks.Add()
    sh = ss.ActiveSheet
    xl.Visible = True
    sleep(1)

    sh.Cells(1,1).Value = 'Python-to-%s Demo' % app
    sleep(1)
    for i in RANGE:
        sh.Cells(i,1).Value = 'Line %d' % i
        sleep(1)
    sh.Cells(i+2,1).Value = „Das war‘s dann!"

    warn(app)
    ss.Close(False)
    xl.Application.Quit()

if __name__=='__main__':
    Tk().withdraw()
    excel()
```

Zugriff mit PyWin32 auf .NET Komponenten

- PyWin32 COM-Schnittstelle kann man benutzen, um .NET Komponenten in CPython aufzurufen

- .NET Komponente muss mit einer COM-Schnittstelle dekoriert werden:
  - COM Callable Wrapper (CCW), .NET Standardmechanismus „COM Client ruft .NET Objekt“



Rezept zum Erzeugen eines CCWs

- Visual Studio Projekt vom Type "Class Library,, anlegen
- In der Datei AssemblyInfo.cs [assembly: ComVisible(true)] setzen
- “Visual Studio 2008 Command Prompt” starten
- Strong-Name-Tool zum Generieren eines Schlüssels:
  - **sn –k snk.key**
- In den Projekteigenschaften klick “signing” und check “Sign the assembly” und Strong-Name-Key-File eingeben, hier: “snk.key”
- Assembly im GAC (Global Assembly Cache) installieren: **gacutil /i TestLib.dll**
- **tblexp TestLib.dll** Type-Library erzeugen
- **regasm TestLib.dll** Type-Library registrieren
- **oleview** Werkzeug zum Auflisten aller registrierten COM-Objekte

Demo: Zugriff von Cpython auf .NET Assembly

- Visual Studio 2008 C# Library Projekt

- CPython 2.5.2 (IDLE)

Zusammenfassung

- IronPython ist eine effiziente Implementierung von Python auf dem .NET Framework.
- Die Effizienz wird durch die DLR erreicht, einer Erweiterung für das .NET Framework 2.0 (und höher), welche die Implementierung dynamischer Sprache auf .NET vereinfacht.
- Mit IronPython können sämtliche Klassenbibliotheken des .NET Framework genutzt werden.
- DieInteroperabilität mit von IronPython mit CPython-Bibliotheken ist möglich …
- CPython kann über das Zusatzmodul PyWin32 auch .NET Komponenten nutzen

Infos (1)

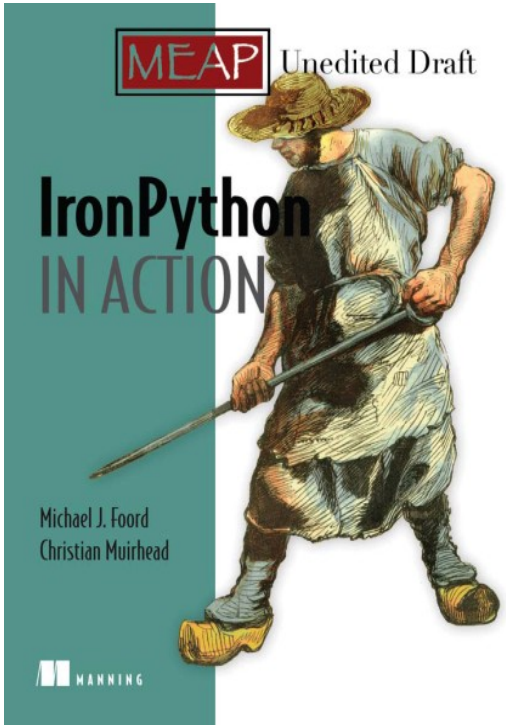
- Python: http://www.python.org/
- IronPython auf Codeplex: http://www.codeplex.com/IronPython
- IronPython, MS SQL, and PEP 249:
  http://blogs.msdn.com/dinoviehland/archive/2008/03/17/ironpython-ms-sql-a
- Dynamic Languages on .NET:
  http://www.voidspace.org.uk/ironpython/ACCU2008
- Embedding IronPython:
  http://www.voidspace.org.uk/ironpython/dlr_hosting.shtml#introduction
- IronPython Roadmap:
  http://devhawk.net/2008/07/17/IronPython+Post+20+Roadmap.aspx

Infos (2)

- IIOP.NET: http://iiop-net.sourceforge.net/
  - Thomas Haug, Es geht auch ohne Seife, Alternativen zu SOAP basierter Kommunikation, Kaffeklatsch das Magazin rund um Software-entwicklung, 03/2008, Seite 8 - 14
- OmniOrb: http://omniorb.sourceforge.net/
  - CORBA explained simply: http://www.ciaranmchale.com/corba-explained-simply/
- Sympy: http://code.google.com/p/sympy/

- Scipy, Numpy: http://www.scipy.org/

Infos (3)



IronPython in Action,
M. J. Foord and C.Muirhead
MEAP Release: September 2007
Softbound print: January 2009 (est.)
450 pages
ISBN: 1-933988-33-9

http://www.manning.com/foord/

.NET und Visual Studio - weitere Angebote

MSDN – Das Microsoft Developer Network

- Nachrichten & Informationen für Entwickler rund um Microsoft Technologien
- www.msdn-online.de

Xtopia

- 17.-18. November 2008 in Berlin
- Die Microsoft-Konferenz für Business, Web Technology, Design & UX.
- Programm und Anmeldung unter: www.xtopia-konferenz.de

Microsoft Technical Summit 2008

- 19.-21. November 2008 in Berlin
- Microsoft-Technologien, -Produkte und -Services von heute und morgen!
- Programm und Anmeldung unter: www.technical-summit.de

Visual Studio Team System Information Day

- Regelmäßige ganztägige Informationsveranstaltung von Microsoft
- Praxisnahe Demos & viel Raum für Diskussionen
- Details & Anmeldung: www.event-team.com/events/visualstudio

Vielen Dank!

Klaus Rohe (klrohe@microsoft.com)

Microsoft Deutschland GmbH