

IronPython IN ACTION

Michael J. Foord
Christian Muirhead
FOREWORD BY JIM HUGUNIN



contents

foreword xvii
preface xx
acknowledgments xxii
about this book xxiii

PART 1 GETTING STARTED WITH IRONPYTHON 1

1 A new language for .NET 3

1.1 An introduction to IronPython 5

What is IronPython? 6 ▪ *A brief history of IronPython 9*
IronPython for Python programmers 11 ▪ *IronPython for .NET
programmers 13*

1.2 Python on the CLR 15

Dynamic languages on .NET and the DLR 15 ▪ *Silverlight: a new
CLR 18* ▪ *The Python programming language 20* ▪ *Multiple
programming paradigms 22*

1.3 Live objects on the console: the interactive interpreter 23

Using the interactive interpreter 23 ▪ *The .NET framework:
assemblies, namespaces, and references 25* ▪ *Live objects and the
interactive interpreter 25* ▪ *Object introspection with dir and help 27*

1.4 Summary 28

2 *Introduction to Python* 29

- 2.1 An overview of Python 31
Python datatypes 32 ▪ Names, objects, and references 40 ▪ Mutable and immutable objects 41
- 2.2 Python: basic constructs 41
*Statements and expressions 42 ▪ Conditionals and loops 43
 Functions 44 ▪ Built-in functions 45 ▪ Classes 47*
- 2.3 Additional Python features 50
*Exception handling 50 ▪ Closures and scoping rules 52 ▪ List comprehensions 54 ▪ Modules, packages, and importing 55
 Docstrings 58 ▪ The Python standard library 58*
- 2.4 Summary 61

3 *.NET objects and IronPython* 62

- 3.1 Introducing .NET 63
Translating MSDN documentation into IronPython 63 ▪ The Form class 65
- 3.2 Structures, enumerations, and collections: .NET types 67
Methods and properties inherited from Control 67 ▪ Adding a Label to the Form: ControlCollection 68 ▪ Configuring the Label: the Color structure 70 ▪ The FormBorderStyle enumeration 71 ▪ Hello World with Form and Label 72
- 3.3 Handling events 73
Delegates and the MouseEventArgs 74 ▪ Event handlers in IronPython 75
- 3.4 Subclassing .NET types 77
- 3.5 Summary 78

PART 2 CORE DEVELOPMENT TECHNIQUES 79

4 *Writing an application and design patterns with IronPython* 81

- 4.1 Data modeling and duck typing 82
Python and protocols 82 ▪ Duck typing in action 83
- 4.2 Model-View-Controller in IronPython 84
Introducing the running example 85 ▪ The view layer: creating a user interface 86 ▪ A data model 88 ▪ A controller class 89

- 4.3 The command pattern 91
 - The SaveFileDialog 92* ▪ *Writing files: the .NET and Python ways 93*
 - Handling exceptions and the system message box 95* ▪ *The SaveCommand 98* ▪ *The SaveAsCommand 100*
- 4.4 Integrating commands with our running example 100
 - Menu classes and lambda 101* ▪ *.NET classes: ToolBar and images 103*
 - Bringing the GUI to life 105*
- 4.5 Summary 108

5 *First-class functions in action with XML* 110

- 5.1 First-class functions 111
 - Higher order functions 111* ▪ *Python decorators 113* ▪ *A null-argument-checking decorator 113*
- 5.2 Representing documents with XML 114
 - The .NET XmlWriter 116* ▪ *A DocumentWriter Class 118* ▪ *An alternative with an inner function 120*
- 5.3 Reading XML 121
 - XMLReader 121* ▪ *An IronPython XmlDocumentReader 123*
- 5.4 Handler functions for MultiDoc XML 126
- 5.5 The Open command 129
- 5.6 Summary 132

6 *Properties, dialogs, and Visual Studio* 133

- 6.1 Document observers 134
 - Python properties 134* ▪ *Adding the OpenCommand 138*
- 6.2 More with TabPages: dialogs and Visual Studio 139
 - Remove pages: OK and Cancel dialog box 139* ▪ *Rename pages: a modal dialog 143* ▪ *Visual Studio Express and IronPython 148*
 - Adding pages: code reuse in action 151* ▪ *Wiring the commands to the view 152*
- 6.3 Object serializing with BinaryFormatter 154
- 6.4 Summary 156

7 *Agile testing: where dynamic typing shines* 157

- 7.1 The unittest module 158
 - Creating a TestCase 159* ▪ *setUp and tearDown 162* ▪ *Test suites with multiple modules 163*

- 7.2 Testing with mocks 166
 - Mock objects 166* ▪ *Modifying live objects: the art of the monkey patch 169*
 - Mocks and dependency injection 173*
- 7.3 Functional testing 175
 - Interacting with the GUI thread 176* ▪ *An AsyncExecutor for asynchronous interactions 178* ▪ *The functional test: making MultiDoc dance 179*
- 7.4 Summary 182

8 **Metaprogramming, protocols, and more 183**

- 8.1 Protocols instead of interfaces 184
 - A myriad of magic methods 184* ▪ *Operator overloading 187*
 - Iteration 191* ▪ *Generators 192* ▪ *Equality and inequality 193*
- 8.2 Dynamic attribute access 195
 - Attribute access with built-in functions 196* ▪ *Attribute access through magic methods 197* ▪ *Proxying attribute access 198*
- 8.3 Metaprogramming 199
 - Introduction to metaclasses 200* ▪ *Uses of metaclasses 201* ▪ *A profiling metaclass 202*
- 8.4 IronPython and the CLR 205
 - .NET arrays 205* ▪ *Overloaded methods 208* ▪ *out, ref, params, and pointer parameters 208* ▪ *Value types 210* ▪ *Interfaces 211*
 - Attributes 212* ▪ *Static compilation of IronPython code 213*
- 8.5 Summary 214

PART 3 IRONPYTHON AND ADVANCED .NET..... 215

9 **WPF and IronPython 217**

- 9.1 Hello World with WPF and IronPython 220
 - WPF from code 221* ▪ *Hello World from XAML 223*
- 9.2 WPF in action 226
 - Layout with the Grid 227* ▪ *The WPF ComboBox and CheckBox 229*
 - The Image control 231* ▪ *The Expander 232* ▪ *The ScrollViewer 233*
 - The TextBlock: a lightweight document control 234* ▪ *The XamlWriter 236*
- 9.3 XPS documents and flow content 236
 - FlowDocument viewer classes 238* ▪ *Flow document markup 239*
 - Document XAML and object tree processing 240*
- 9.4 Summary 243

- 10 Windows system administration with IronPython 244**
- 10.1 System administration with Python 245
 - Simple scripts 245* ▪ *Shell scripting with IronPython 246*
 - 10.2 WMI and the System.Management assembly 251
 - System.Management 251* ▪ *Connecting to remote computers 255*
 - 10.3 PowerShell and IronPython 260
 - Using PowerShell from IronPython 260* ▪ *Using IronPython from PowerShell 264*
 - 10.4 Summary 271
- 11 IronPython and ASP.NET 273**
- 11.1 Introducing ASP.NET 274
 - Web controls 274* ▪ *Pages and user controls 275* ▪ *Rendering, server code, and the page lifecycle 275*
 - 11.2 Adding IronPython to ASP.NET 276
 - Writing a first application 277* ▪ *Handling an event 279*
 - 11.3 ASP.NET infrastructure 280
 - The App_Script folder 280* ▪ *The Global.py file 281* ▪ *The Web.config file 282*
 - 11.4 A web-based MultiDoc Viewer 282
 - Page structure 283* ▪ *Code-behind 285*
 - 11.5 Editing MultiDocs 287
 - Swapping controls 288* ▪ *Handling view state 289* ▪ *Additional events 292*
 - 11.6 Converting the Editor into a user control 294
 - View state again 295* ▪ *Adding parameters 296*
 - 11.7 Summary 298
- 12 Databases and web services 299**
- 12.1 Relational databases and ADO.NET 300
 - Trying it out using PostgreSQL 301* ▪ *Connecting to the database 303*
Executing commands 304 ▪ *Setting parameters 305* ▪ *Querying the database 306*
Reading multirow results 307 ▪ *Using transactions 309*
DataAdapters and DataSets 311
 - 12.2 Web services 313
 - Using a simple web service 314* ▪ *Using SOAP services from IronPython 317* ▪ *REST services in IronPython 319*
 - 12.3 Summary 328

- 13 Silverlight: IronPython in the browser 329**
- 13.1 Introduction to Silverlight 330
 - Dynamic Silverlight 332* ▪ *Your Python application 334*
 - Silverlight controls 335* ▪ *Packaging a Silverlight application 339*
 - 13.2 A Silverlight Twitter client 341
 - Cross-domain policies 341* ▪ *Debugging Silverlight applications 343*
 - The user interface 344* ▪ *Accessing network resources 346* ▪ *Threads and dispatching onto the UI thread 349* ▪ *IsolatedStorage in the browser 351*
 - 13.3 Videos and the browser DOM 353
 - The MediaElement video player 353* ▪ *Accessing the browser DOM 354*
 - 13.4 Summary 356

PART 4 REACHING OUT WITH IRONPYTHON 357

- 14 Extending IronPython with C#/VB.NET 359**
- 14.1 Writing a class library for IronPython 360
 - Working with Visual Studio or MonoDevelop 361* ▪ *Python objects from class libraries 362* ▪ *Calling unmanaged code with the P/Invoke attribute 366* ▪ *Methods with attributes through subclassing 370*
 - 14.2 Creating dynamic (and Pythonic) objects from C#/VB.NET 374
 - Providing dynamic attribute access 374* ▪ *Python magic methods 378*
 - APIs with keyword and multiple arguments 378*
 - 14.3 Compiling and using assemblies at runtime 382
 - 14.4 Summary 385
- 15 Embedding the IronPython engine 386**
- 15.1 Creating a custom executable 387
 - The IronPython engine 387* ▪ *Executing a Python file 389*
 - 15.2 IronPython as a scripting engine 393
 - Setting and fetching variables from a scope 394* ▪ *Providing modules and assemblies for the engine 398* ▪ *Python code as an embedded resource 400*
 - 15.3 Python plugins for .NET applications 402
 - A plugin class and registry 403* ▪ *Autodiscovery of user plugins 404*
 - Diverting standard output 406* ▪ *Calling the user plugins 407*

15.4	Using DLR objects from other .NET languages	409
	<i>Expressions, functions, and Python types</i>	409
	<i>Dynamic operations with ObjectOperations</i>	412
	<i>The built-in Python functions and modules</i>	414
	<i>The future of interacting with dynamic objects</i>	417
15.5	Summary	418
<i>appendix A</i>	<i>A whirlwind tour of C#</i>	419
<i>appendix B</i>	<i>Python magic methods</i>	433
<i>appendix C</i>	<i>For more information</i>	445
	<i>index</i>	449