1. Record Nr. UNINA9910813993503321

Autore	Kak Avinash C
Titolo	Scripting with objects : a comparative presentation of object-oriented scripting with Perl and Python / / Avinash C. Kak
Pubbl/distr/stampa	Hoboken, N.J., : Wiley-Interscience, c2008
ISBN	0470255781 9780470255780
Edizione	[1st ed.]
Descrizione fisica	xxxiv, 1279 p. : ill
Disciplina	005.1/17
Soggetti	Object-oriented programming (Computer science) Perl (Computer program language) Python (Computer program language) Scripting languages (Computer science)
Lingua di pubblicazione	Inglese
Formato	Materiale a stampa
Livello bibliografico	Monografia
Nota di bibliografia	Includes bibliographical references and index.
Nota di contenuto	Intro -- Scripting with Objects: A Comparative Presentation of Object-Oriented Scripting With Perl and Python -- Contents -- Preface -- Acknowledgments -- 1 Multilanguage View of Application Development and OO Scripting -- 1.1 Scripting Languages Versus Systems Programming Languages -- 1.2 Organization of this Book -- 1.3 Credits and Suggestions for Further Reading -- 2 Perl - A Review of the Basics -- 2.1 Scalar Values in Perl -- 2.1.1 Numbers -- 2.1.2 Strings -- 2.2 Perl's Variables: Scalars, Arrays, and Hashes -- 2.2.1 Scalar -- 2.2.2 Array -- 2.2.3 Hash -- 2.3 Lexical Scope, Lexical Variables, and Global Variables -- 2.3.1 Lexical Variables -- 2.3.2 Package Variables -- 2.4 Displaying Arrays -- 2.5 Displaying Hashes -- 2.6 Terminal and File I/O -- 2.6.1 Terminal I/O -- 2.6.2 File I/O -- 2.6.2.1 I/O for Text Files -- 2.6.2.2 I/O for Binary Files -- 2.7 Functions, Subroutines, and Functions Used as Operators -- 2.7.1 Using a Function as an Operator -- 2.7.2 User-Defined Functions -- 2.7.3 Passing Arguments to Functions -- 2.7.4 Functions Can be Called with Keyword Arguments -- 2.7.5 Default Values for Function Arguments -- 2.8 What Is Returned by Evaluation Depends on Context -- 2.9 Conditional Evaluation and Loop Control Structures -- 2.9.1 Controlling an Outer

Loop from an Inner Loop -- 2.9.2 When Is a Conditional True or False? -- 2.9.3 Complex Conditionals -- 2.10 Functions Supplied with Here-Doc Arguments -- 2.11 Modules and Packages in Perl -- 2.11.1 Creating Your Own Module -- 2.11.2 Importing Names from a Module -- 2.11.3 "Nesting" of Modules -- 2.11.4 The Autoloading Feature -- 2.11.5 Package Constructors and Destructors -- 2.12 Temporarily Localizing a Global Variable -- 2.13 Typeglobs for Global Names -- 2.13.1 Creating Global Variables by Direct Assignments to Typeglob Slots -- 2.14 The eval Operator.
2.15 grep() and map() Functions -- 2.16 Interacting with the Directory Structure -- 2.16.1 Directory Handles -- 2.16.2 File Tests -- 2.16.3 Taking Advantage of Shell's Globbing -- 2.16.4 Scanning a Directory Tree -- 2.17 Launching Processes -- 2.17.1 Launching a Child Process with system() -- 2.17.2 Launching a Child Process with Backticks -- 2.17.3 exec( ) for Transferring Control to a Process -- 2.17.4 Launching a Child Process with fork() -- 2.17.5 open( ) for Interprocess Communications -- 2.18 Sending and Trapping Signals -- 2.19 Credits and Suggestions for Further Reading -- 2.20 Homework -- 3 Python - A Review of the Basics -- 3.1 Language Model: Perl versus Python -- 3.2 Numbers -- 3.3 Python Containers: Sequences -- 3.3.1 Strings -- 3.3.2 Tuples -- 3.3.3 Lists -- 3.3.4 Xrange Sequences -- 3.4 Python Containers: Dictionaries -- 3.5 Built-in Types as Classes -- 3.5.1 String Type as a Class -- 3.5.2 Numeric Types as Classes -- 3.6 Subclassing the Built-in Types -- 3.6.1 Subclassing the String Type -- 3.6.2 Subclassing the Integer Type -- 3.7 Terminal and File I/O -- 3.7.1 Terminal I/O -- 3.7.2 File I/O -- 3.7.2.1 I/O for Text Files -- 3.7.2.2 I/O for Binary Files -- 3.8 User-defined Functions -- 3.8.1 A Function Is an Object -- 3.8.2 The Object Returned by a Function Call -- 3.8.3 Default Arguments for Function Parameters -- 3.8.4 Functions Can Be Called with Arbitrary Number of Arguments -- 3.8.5 Functions Can Be Called with Keyword Arguments -- 3.8.6 Anonymous Functions with Lambda Expressions -- 3.8.7 Closures -- 3.9 Control Structures -- 3.9.1 When Is a Conditional True or False? -- 3.9.2 Complex Conditionals -- 3.10 Modules in Python -- 3.10.1 Importing a Module -- 3.10.2 Importing Specific Names from a Module -- 3.11 Scoping Rules, Namespaces, and Name Resolution -- 3.11.1 Nested Namespaces -- 3.11.2 Name Resolution for Imported Modules.
3.11.3 What about the Names Imported with from...import Syntax? -- 3.11.4 Deeply Nested Namespaces and the global Declaration -- 3.11.5 Python Is Lexically Scoped -- 3.12 The eval() Function -- 3.13 map() and filterQ Functions -- 3.14 Interacting with the Directory Structure -- 3.14.1 File Tests -- 3.14.2 Taking Advantage of Shell's Globbing -- 3.14.3 Scanning a Directory Tree -- 3.15 Launching Processes -- 3.15.1 Launching a Child Process with os.system( ) -- 3.15.2 os.exec Functions for Launching External Commands -- 3.15.3 Launching a Child Process with os.fork( ) -- 3.15.4 os.popen( ) for Interprocess Communication -- 3.16 Sending and Trapping Signals -- 3.17 Credits and Suggestions for Further Reading -- 3.18 Homework -- 4 Regular Expressions for String Processing -- 4.1 What is an Input String? -- 4.2 Simple Substring Search -- 4.3 What is Meant by a Match between a Regex and an Input String? -- 4.4 Regex Matching at Line and Word Boundaries -- 4.5 Character Classes for Regex Matching -- 4.6 Specifying Alternatives in a Regex -- 4.7 Subexpression of a Regex -- 4.8 Extracting Substrings from an Input String -- 4.8.1 Other Uses of Parentheses in Regular Expressions -- 4.9 Abbreviated Notation for Character Classes -- 4.10 Quantifier Metacharacters -- 4.10.1 Greediness of Quantifiers -- 4.10.2 The Sometimes Unintended Consequence of Greedy Quantifiers -- 4.10.3 Nongreedy Quantifiers --

4.10.4 Perl and Python Example Scripts with Quantifiers -- 4.11 Match Modifiers -- 4.11.1 Case-Insensitive Matching -- 4.11.2 Going Global -- 4.11.3 Input Strings Consisting of Multiple Lines -- 4.11.4 Multiline Regular Expressions -- 4.11.5 Other Match Modifiers -- 4.12 Splitting Strings -- 4.12.1 Joining Strings -- 4.13 Regexes for Search and Replace Operations -- 4.14 Credits and Suggestions for Further Reading -- 4.15 Homework -- 5 References in Perl.
5.1 Referencing and Dereferencing Operators (Summary) -- 5.2 Referencing and Dereferencing a Scalar -- 5.3 Referencing and Dereferencing a Named Array -- 5.4 Referencing and Dereferencing an Anonymous Array -- 5.5 Referencing and Dereferencing a Named Hash -- 5.6 Referencing and Dereferencing an Anonymous Hash -- 5.7 Referencing and Dereferencing A Named Subroutine -- 5.8 Referencing and Dereferencing An Anonymous Subroutine -- 5.9 Subroutines Returning References to Subroutines -- 5.10 Closures -- 5.11 Enforcing Privacy in Modules -- 5.12 References to Typeglobs -- 5.13 The ref() Function -- 5.14 Symbolic References -- 5.14.1 Symbolic References to Subroutines -- 5.15 Credits and Suggestions for Further Reading -- 5.16 Homework -- 6 The Notion of a Class in Perl -- 6.1 Defining a Class in Perl -- 6.1.1 Blessing an Object into a Package -- 6.1.2 Providing a Class with a Constructor -- 6.1.3 Data Hiding and Data Access Issues -- 6.1.4 Packaging a Class into a Module -- 6.2 Constructors Can Be Called with Keyword Arguments -- 6.3 Default Values for Instance Variables -- 6.4 Instance Object Destruction -- 6.4.1 Destructors and the Problem of Circular References -- 6.5 Controlling the Interaction between DESTROY( ) and AUTOLOAD( ) -- 6.6 Class Data and Methods -- 6.7 Reblessing Objects -- 6.8 Operator Overloading and Class Customization -- 6.9 Credits and Suggestions for Further Reading -- 6.10 Homework -- 7 The Notion of a Class in Python -- 7.1 Defining a Class in Python -- 7.1.1 Constructors and System-Supplied Attributes -- 7.1.2 Class Definition: The Syntax -- 7.2 New-Style Versus Classic Classes in Python -- 7.3 Defining Methods -- 7.3.1 A Method Can Be Defined Outside a Class -- 7.3.2 Bound and Unbound Methods -- 7.3.3 Using __getattr__( ) as a Catch-All for Nonexistent Methods -- 7.3.4 __getattr__( ) versus __getattribute__( ).
7.4 Destruction of Instance Objects -- 7.5 Encapsulation Issues for Classes -- 7.6 Defining Class Variables, Static Methods, and Class Methods -- 7.6.1 An Instance Variable Hides a Class Variable of the Same Name -- 7.7 Private Data Attributes and Methods -- 7.8 Defining a Class with Slots -- 7.9 Descriptor Classes in Python -- 7.10 Operator Overloading and Class Customization -- 7.11 Credits and Suggestions for Further Reading -- 7.12 Homework -- 8 Inheritance and Polymorphism in Perl -- 8.1 Inheritance in Mainstream OO -- 8.2 Inheritance and Polymorphism in Perl: Comparison with Mainstream OO Languages -- 8.3 The ISA Array for Specifying the Parents of a Class -- 8.4 An Example of Class Derivation in Perl -- 8.5 A Small Demonstration of Polymorphism in Perl OO -- 8.6 How a Derived-Class Method Calls on a Base-Class Method -- 8.7 The UNIVERSAL Class -- 8.7.1 Adding Functionality to the UNIVERSAL class -- 8.8 How a Method is Searched For in a Class Hierarchy -- 8.9 Inherited Methods Behave As If Locally Defined -- 8.10 Destruction of Derived-Class Instances -- 8.11 Diamond Inheritance -- 8.12 On the Inheritability of a Class -- 8.13 Local Variables and Subroutines in Derived Classes -- 8.14 Operator Overloading and Inheritance -- 8.15 Credits and Suggestions for Further Reading -- 8.16 Homework -- 9 Inheritance and Polymorphism in Python -- 9.1 Extending a Class in Python -- 9.2 Extending a Base-Class Method in a Single-Inheritance Chain -- 9.3 A Simple Demonstration of Polymorphism in Python OO -- 9.4

Destruction of Derived-Class Instances in Single-Inheritance Chains -- 9.5 The Root Class object -- 9.6 Subclassing from the Built-In Types -- 9.6.1 Subclassing the Built-In dict -- 9.6.2 Subclassing the Built-In list -- 9.6.3 Subclassing the Built-In tuple -- 9.7 On Overriding __new__( ) and __init__( ) -- 9.8 Multiple Inheritance.
9.8.1 Method Resolution Order for Classic Classes.

Sommario/riassunto

Object-Oriented scripting with Perl and Python Scripting languages are becoming increasingly important for software development. These higher-level languages, with their built-in easy-to-use data structures are convenient for programmers to use as "glue" languages for assembling multi-language applications and for quick prototyping of software architectures. Scripting languages are also used extensively in Web-based applications. Based on the same overall philosophy that made Programming with Objects such a wide success, Scripting with Objects takes a novel dual-language approach to learning advanced scripting with Perl and Python, the dominant languages of the genre. This method of comparing basic syntax and writing application-level scripts is designed to give readers a more comprehensive and expansive perspective on the subject. Beginning with an overview of the importance of scripting languages-and how they differ from mainstream systems programming languages-the book explores: Regular expressions for string processing The notion of a class in Perl and Python Inheritance and polymorphism in Perl and Python Handling exceptions Abstract classes and methods in Perl and Python Weak references for memory management Scripting for graphical user interfaces Multithreaded scripting Scripting for network programming Interacting with databases Processing XML with Perl and Python This book serves as an excellent textbook for a one-semester undergraduate course on advanced scripting in which the students have some prior experience using Perl and Python, or for a two-semester course for students who will be experiencing scripting for the first time. Scripting with Objects is also an ideal resource for industry professionals who are making the transition from Perl to Python, or vice versa.