A Functional Start To Computing With Python Chapman Hallcrc Textbooks In Computing

Right here, we have countless book **A Functional Start To Computing With Python Chapman Hallcrc Textbooks In Computing** and collections to check out. We additionally manage to pay for variant types and furthermore type of the books to browse. The up to standard book, fiction, history, novel, scientific research, as without difficulty as various extra sorts of books are readily open here.

As this A Functional Start To Computing With Python Chapman Hallcrc Textbooks In Computing, it ends stirring inborn one of the favored ebook A Functional Start To Computing With Python Chapman Hallcrc Textbooks In Computing collections that we have. This is why you remain in the best website to look the unbelievable book to have.

FARMER DANIKA

Trends in Functional Programming Volume 6 Cambridge University Press
Discovering Computer Science: Interdisciplinary Problems, Principles, and Python Programming introduces computational problem solving as a vehicle of discovery in a wide variety of disciplines. With a principles-oriented introduction to computational thinking, the text provides a broader and deeper introduction to computer science than typical introductory programming books. Organized around interdisciplinary problem domains, rather than programming language features, each chapter guides students through increasingly sophisticated algorithmic and programming techniques. The author uses a spiral approach to introduce Python language features in increasingly complex contexts as the book progresses. The text places programming in the context of fundamental computer science principles, such as abstraction, efficiency, and algorithmic techniques, and offers overviews of fundamental topics that are traditionally put off until later courses. The book includes thirty well-developed independent projects that encourage students to explore questions across disciplinary boundaries. Each is motivated by a problem that students can investigate by developing algorithms and implementing them as Python programs. The book's accompanying website — http://discoverCS.denison.edu — includes sample code and data files, pointers for further exploration, errata, and links to Python language references. Containing over 600 homework exercises and over 300 integrated reflection questions, this textbook is appropriate for a first computer science course for computer science majors, an introductory scientific computing course or, at a slower pace, any introductory computer science course.
Structure and Interpretation of Computer Programs CRC Press
Agda is an advanced programming language based on Type Theory. Agda's type system is expressive enough to support full functional verification of programs, in two styles. In external verification, we write pure functional programs and then write proofs of properties about them. The proofs are separate external artifacts, typically using structural induction. In internal verification, we specify properties of programs through rich types for the programs themselves. This often necessitates including proofs inside code, to show the type checker that the specified properties hold. The power to prove properties of programs in these two styles is a profound addition to the practice of programming, giving programmers the power to guarantee the absence of bugs, and thus improve the quality of software more than previously possible. Verified Functional Programming in Agda is the first book to provide a systematic exposition of external and internal verification in Agda, suitable for undergraduate students of Computer Science. No familiarity with functional programming or computer-checked proofs is presupposed. The book begins with an introduction to functional programming through familiar examples like booleans, natural numbers, and lists, and techniques for external verification. Internal verification is considered through the examples of vectors, binary search trees, and Braun trees. More advanced material on type-level computation, explicit reasoning about termination, and normalization by evaluation is also included. The book also includes a medium-sized case study on Huffman encoding and decoding.
Learn Functional Programming with Elixir "O'Reilly Media, Inc."
Summary Functional Programming in Scala is a serious tutorial for programmers looking to learn FP and apply it to the everyday business of coding. The book guides readers from basic techniques to advanced topics in a logical, concise, and clear progression. In it, you'll find concrete examples and exercises that open up the world of functional programming. Purchase of the print book includes a free eBook in PDF, Kindle, and ePub formats from Manning Publications. About the Technology Functional programming (FP) is a style of software development emphasizing functions that don't depend on program state. Functional code is easier to test and reuse, simpler to parallelize, and less prone to bugs than other code. Scala is an emerging JVM language that offers strong support for FP. Its familiar syntax and transparent interoperability with Java make Scala a great place to start learning FP. About the Book Functional Programming in Scala is a serious tutorial for programmers looking to learn FP and apply it to their everyday work. The book guides readers from basic techniques to advanced topics in a logical, concise, and clear progression. In it, you'll find concrete examples and exercises that open up the world of functional programming. This book assumes no prior experience with functional programming. Some prior exposure to Scala or Java is helpful. What's Inside Functional programming concepts The whys and hows of FP How to write multicore programs Exercises and checks for understanding About the Authors Paul Chiusano and Rúnar Bjarnason are recognized experts in functional programming with Scala and are core contributors to the Scalaz library. Table of Contents PART 1 INTRODUCTION TO FUNCTIONAL PROGRAMMING What is functional programming? Getting started with functional programming in Scala Functional data structures Handling errors without exceptions Strictness and laziness Purely functional state PART 2 FUNCTIONAL DESIGN AND COMBINATOR LIBRARIES Purely functional parallelism Property-based testing Parser combinators PART 3 COMMON STRUCTURES IN FUNCTIONAL DESIGN Monoids Monads Applicative and traversable functors PART 4 EFFECTS AND I/O External effects and I/O Local effects and mutable state Stream processing and incremental I/O
Real-World Functional Programming Apress
This is Volume 7 of Trends in Functional Programming (TFP). It contains a refereed selection of the papers that were presented at TFP 2006: the Seventh Symposium on Trends in Functional Programming. which took place in Nottingham, 19-21 April, 2006. TFP is an international forum for researchers from all functional programming communities spanning the entire width of topics in the field. Its goal is to provide a broad view of current and future trends in functional programming in a lively and friendly setting, thus promoting new research directions related to the field of functional programming and the relationship between functional programming and other fields of computer science. True to the spirit of TFP, the selection of papers in this volume covers a wide range of topics, including dependently typed programming, generic programming, purely functional data structures, function synthesis, declarative debugging, implementation of functional programming languages, and memory management. A particular emerging trend is that of dependently typed programming, reflected by a number of papers in the present selection and by the co-location of TFP and Types 2006.
Composing Software CRC Press
A student introduction to the design of algorithms for problem solving. Written from a functional programming perspective, the text should appeal to anyone studying algorithms. Included are end-of-chapter exercises and bibliographic references.
Explorations in Computing CRC Press
All software design is composition: the act of breaking complex problems down into smaller problems and composing those solutions. Most developers have a limited understanding of compositional techniques. It's time for that to change.In "Composing Software", Eric Elliott shares the fundamentals of composition, including both function composition and object composition, and explores them in the context of JavaScript. The book covers the foundations of both functional programming and object oriented programming to help the reader better understand how to build and structure complex applications using simple building blocks.You'll learn: Functional programmingObject compositionHow to work with composite data structuresClosuresHigher order functionsFunctors (e.g., array.map)Monads (e.g., promises)TransducersLensesAll of this in the context of JavaScript, the most used programming language in the world. But the learning doesn't stop at JavaScript. You'll be able to apply these lessons to any language. This book is about the timeless principles of software composition and its lessons will outlast the hot languages and frameworks of today. Unlike most programming books, this one may still be relevant 20 years from now.This book began life as a popular blog post series that attracted hundreds of thousands of readers and influenced the way software is built at many high growth tech startups and fortune 500 companies
Functional Programming in C++ CRC Press
This book introduces Miranda at a level appropriate for professionals with little or no prior experience in programming. The emphasis is on the process of crafting programs, solving problems, and avoiding common errors. Using a large number of running examples and case studies, the book encourages the design of well structured, reusable software together with proofs of correctness. A tear-out card enables readers to acquire a Miranda compiler from Research Software Ltd. at a substantial discount off the published list price.
Functional C Simon and Schuster
The classic guide to how computers work, updated with new chapters and interactive graphics "For me, Code was a revelation. It was the first book about programming that spoke to me. It started with a story, and it built up, layer by layer, analogy by analogy, until I understood not just the Code, but the System. Code is a book that is as much about Systems Thinking and abstractions as it is about code and programming. Code teaches us how many unseen layers there are between the computer systems that we as users look at every day and the magical silicon rocks that we infused with lightning and taught to think." - Scott Hanselman, Partner Program Director, Microsoft, and host of Hanselminutes Computers are everywhere, most obviously in our laptops and smartphones, but also our cars, televisions, microwave ovens, alarm clocks, robot vacuum cleaners, and other smart appliances. Have you ever wondered what goes on inside these devices to make our lives easier but occasionally more infuriating? For more than 20 years, readers have delighted in Charles Petzold's illuminating story of the secret inner life of computers, and now he has revised it for this new age of computing. Cleverly illustrated and easy to understand, this is the book that cracks the mystery. You'll discover what flashlights, black cats, seesaws, and the ride of Paul Revere can teach you about computing, and how human ingenuity and our compulsion to communicate have shaped every electronic device we use. This new expanded edition explores more deeply the bit-by-bit and gate-by-gate construction of the heart of every smart device, the central processing unit that combines the simplest of basic operations to perform the most complex of feats. Petzold's companion website, CodeHiddenLanguage.com, uses animated graphics of key circuits in the book to make computers even easier to comprehend. In addition to substantially revised and updated content, new chapters include: Chapter 18: Let's Build a Clock! Chapter 21: The Arithmetic Logic Unit Chapter 22: Registers and Busses Chapter 23: CPU Control Signals Chapter 24: Jumps, Loops, and Calls Chapter 28: The World Brain From the simple ticking of clocks to the worldwide hum of the internet, Code reveals the essence of the digital revolution.
Programming Elixir 1. 6 Pragmatic Bookshelf
Create succinct and expressive implementations with functional programming in Python Key Features Learn how to choose between imperative and functional approaches based on expressiveness, clarity, and performance Get familiar with complex concepts such as monads, concurrency, and immutability Apply functional Python to common Exploratory Data Analysis (EDA) programming problems Book Description If you’re a Python developer who wants to discover how to take the power of functional programming (FP) and bring it into your own programs, then this book is essential for you, even if you know next to nothing about the paradigm. Starting with a general overview of functional concepts, you’ll explore common functional features such as first-class and higher-order functions, pure functions, and more. You’ll see how these are accomplished in Python 3.6 to give you the core foundations you’ll build upon. After that, you’ll discover common functional optimizations for Python to help your apps reach even higher speeds. You’ll learn FP concepts such as lazy evaluation using Python’s generator functions and expressions. Moving forward, you’ll learn to design and implement decorators to create composite functions. You'll also

explore data preparation techniques and data exploration in depth, and see how the Python standard library fits the functional programming model. Finally, to top off your journey into the world of functional Python, you'll at look at the PyMonad project and some larger examples to put everything into perspective. What you will learn Use Python's generator functions and generator expressions to work with collections in a non-strict (or lazy) manner Utilize Python library modules including itertools, functools, multiprocessing, and concurrent features to ensure efficient functional programs Use Python strings with object-oriented suffix notation and prefix notation Avoid stateful classes with families of tuples Design and implement decorators to create composite functions Use functions such as max(), min(), map(), filter(), and sorted() Write higher-order functions Who this book is for This book is for Python developers who would like to perform Functional programming with Python. Python Programming knowledge is assumed.

Functional Programming in R "O'Reilly Media, Inc."

Racket is a descendant of Lisp, a programming language renowned for its elegance, power, and challenging learning curve. But while Racket retains the functional goodness of Lisp, it was designed with beginning programmers in mind. Realm of Racket is your introduction to the Racket language. In Realm of Racket, you'll learn to program by creating increasingly complex games. Your journey begins with the Guess My Number game and coverage of some basic Racket etiquette. Next you'll dig into syntax and semantics, lists, structures, and conditionals, and learn to work with recursion and the GUI as you build the Robot Snake game. After that it's on to lambda and mutant structs (and an Orc Battle), and fancy loops and the Dice of Doom. Finally, you'll explore laziness, AI, distributed games, and the Hungry Henry game. As you progress through the games, chapter checkpoints and challenges help reinforce what you've learned. Offbeat comics keep things fun along the way. As you travel through the Racket realm, you'll: –Master the quirks of Racket's syntax and semantics –Learn to write concise and elegant functional programs –Create a graphical user interface using the 2htdp/image library –Create a server to handle true multiplayer games Realm of Racket is a lighthearted guide to some serious programming. Read it to see why Racketeers have so much fun!

How to Design Programs, second edition Addison Wesley Longman

A Functional Start to Computing with Python enables students to quickly learn computing without having to use loops, variables, and object abstractions at the start. Requiring no prior programming experience, the book draws on Python's flexible data types and operations as well as its capacity for defining new functions. Along with the specifics of Python, the text covers important concepts of computing, including software engineering motivation, algorithms behind syntax rules, advanced functional programming ideas, and, briefly, finite state machines. Taking a student-friendly, interactive approach to teach computing, the book addresses more difficult concepts and abstractions later in the text. The author presents ample explanations of data types, operators, and expressions. He also describes comprehensions—the powerful specifications of lists and dictionaries—before introducing loops and variables. This approach helps students better understand assignment syntax and iteration by giving them a mental model of sophisticated data first. Web Resource The book's supplementary website at http://functionalfirstpython.com/ provides many ancillaries, including: Interactive flashcards on Python language elements Links to extra support for each chapter Unit testing and programming exercises An interactive Python stepper tool Chapter-by-chapter points Material for lectures

Grokking Functional Programming Packt Publishing Ltd

Summary Functional Programming in C++ teaches developers the practical side of functional programming and the tools that C++ provides to develop software in the functional style. This in-depth guide is full of useful diagrams that help you understand FP concepts and begin to think functionally. Purchase of the print book includes a free eBook in PDF, Kindle, and ePub formats from Manning Publications. About the Technology Well-written code is easier to test and reuse, simpler to parallelize, and less error prone. Mastering the functional style of programming can help you tackle the demands of modern apps and will lead to simpler expression of complex program logic, graceful error handling, and elegant concurrency. C++ supports FP with templates, lambdas, and other core language features, along with many parts of the STL. About the Book Functional Programming in C++ helps you unleash the functional side of your brain, as you gain a powerful new perspective on C++ coding. You'll discover dozens of examples, diagrams, and illustrations that break down the functional concepts you can apply in C++, including lazy evaluation, function objects and invokables, algebraic data types, and more. As you read, you'll match FP techniques with practical scenarios where they offer the most benefit. What's inside Writing safer code with no performance penalties Explicitly handling errors through the type system Extending C++ with new control structures Composing tasks with DSLs About the Reader Written for developers with two or more years of experience coding in C++. About the Author Ivan Čukić is a core developer at KDE and has been coding in C++ since 1998. He teaches modern C++ and functional programming at the Faculty of Mathematics at the University of Belgrade. Table of Contents Introduction to functional programming Getting started with functional programming Function objects Creating new functions from the old ones Purity: Avoiding mutable state Lazy evaluation Ranges Functional data structures Algebraic data types and pattern matching Monads Template metaprogramming Functional design for concurrent systems Testing and debugging

A Functional Start to Computing with Python MIT Press

Master functions and discover how to write functional programs in R. In this concise book, you'll make your functions pure by avoiding side-effects; you'll write functions that manipulate other functions, and you'll construct complex functions using simpler functions as building blocks. In Functional Programming in R, you'll see how we can replace loops, which can have side-effects, with recursive functions that can more easily avoid them. In addition, the book covers why you shouldn't use recursion when loops are more efficient and how you can get the best of both worlds. Functional programming is a style of programming, like object-oriented programming, but one that focuses on data transformations and calculations rather than objects and state. Where in object-oriented programming you model your programs by describing which states an object can be in and how methods will reveal or modify that state, in functional programming you model programs by describing how functions translate input data to output data. Functions themselves are considered to be data you can manipulate and much of the strength of functional programming comes from manipulating functions; that is, building more complex functions by combining simpler functions. What You'll Learn Write functions in R including infix operators and replacement functions Create higher order functions Pass functions to other functions and start using functions as data you can manipulate Use Filer, Map and Reduce functions to express the intent behind code clearly and safely Build new functions from existing functions without necessarily writing any new functions, using point-free programming Create functions that carry data along with them Who This Book Is For Those with at least some experience with programming in R.

Code Intellect Books

This fast-moving tutorial introduces you to OCaml, an industrial-strength programming language designed for expressiveness, safety, and speed. Through the book's many examples, you'll quickly learn how OCaml stands out as a tool for writing fast, succinct, and readable systems code. Real World OCaml takes you through the concepts of the language at a brisk pace, and then helps you explore the tools and techniques that make OCaml an effective and practical tool. In the book's third section, you'll delve deep into the details of the compiler toolchain and OCaml's simple and efficient runtime system. Learn the foundations of the language, such as higher-order functions, algebraic data types, and modules Explore advanced features such as functors, first-class modules, and objects Leverage Core, a comprehensive general-purpose standard library for OCaml Design effective and reusable libraries, making the most of OCaml's approach to abstraction and modularity Tackle practical programming problems from command-line parsing to asynchronous network programming Examine profiling and interactive debugging techniques with tools such as GNU gdb

Computer Programming for Absolute Beginners John Wiley & Sons

Summary Functional Programming in C# teaches you to apply functional thinking to real-world problems using the C# language. The book, with its many practical examples, is written for proficient C# programmers with no prior FP experience. It will give you an awesome new perspective. Purchase of the print book includes a free eBook in PDF, Kindle, and ePub formats from Manning Publications. About the Technology Functional programming changes the way you think about code. For C# developers, FP techniques can greatly improve state management, concurrency, event handling, and long-term code maintenance. And C# offers the flexibility that allows you to benefit fully from the application of functional techniques. This book gives you the awesome power of a new perspective. About the Book Functional Programming in C# teaches you to apply functional thinking to real-world problems using the C# language. You'll start by learning the principles of functional programming and the language features that allow you to program functionally. As you explore the many practical examples, you'll learn the power of function composition, data flow programming, immutable data structures, and monadic composition with LINQ. What's Inside Write readable, team-friendly code Master async and data streams Radically improve error handling Event sourcing and other FP patterns About the Reader Written for proficient C# programmers with no prior FP experience. About the Author Enrico Buonanno studied computer science at Columbia University and has 15 years of experience as a developer, architect, and trainer. Table of Contents PART 1 - CORE CONCEPTS Introducing functional programming Why function purity matters Designing function signatures and types Patterns in functional programming Designing programs with function composition PART 2 - BECOMING FUNCTIONAL Functional error handling Structuring an application with functions Working effectively with multi-argument functions Thinking about data functionally Event sourcing: a functional approach to persistence PART 3 - ADVANCED TECHNIQUES Lazy computations, continuations, and the beauty of monadic composition Stateful programs and stateful computations Working with asynchronous computations Data streams and the Reactive Extensions An introduction to message-passing concurrency

Introduction to Scientific Programming with Python Pragmatic Bookshelf

This is a one-chapter book that gives a brief overview of fundamental computer concepts like hardware, software, security, and networking. The goal of the Exploring series is to move students beyond the point-and-click, to understanding the why and how behind each skill.

Functional Programming Courier Corporation

The notion that "thinking about computing is one of the most exciting things the human mind can do" sets both The Little Schemer (formerly known as The Little LISPer) and its new companion volume, The Seasoned Schemer, apart from other books on LISP. The authors' enthusiasm for their subject is compelling as they present abstract concepts in a humorous and easy-to-grasp fashion. Together, these books will open new doors of thought to anyone who wants to find out what computing is really about. The Little Schemer introduces computing as an extension of arithmetic and algebra; things that everyone studies in grade school and high school. It introduces programs as recursive functions and briefly discusses the limits of what computers can do. The authors use the programming language Scheme, and interesting foods to illustrate these abstract ideas. The Seasoned Schemer informs the reader about additional dimensions of computing: functions as values, change of state, and exceptional cases. The Little LISPer has been a popular introduction to LISP for many years. It had appeared in French and Japanese. The Little Schemer and The Seasoned Schemer are worthy successors and will prove equally popular as textbooks for Scheme courses as well as companion texts for any complete introductory course in Computer Science.

Discovering Computer Science Addison Wesley

Smooth, powerful, and small, Elixir is an excellent language for learning functional programming, and with this hands-on introduction, you'll discover just how powerful Elixir can be. Authors Simon St. Laurent and J. David Eisenberg show you how Elixir combines the robust functional programming of Erlang with an approach that looks more like Ruby, and includes powerful macro features for metaprogramming. Updated to cover Elixir 1.4, the second edition of this practical book helps you write simple Elixir programs by teaching one skill at a time. Once you pick up pattern matching, process-oriented programming, and other concepts, you'll understand why Elixir makes it easier to build concurrent and resilient programs that scale up and down with ease. Get comfortable with IEx, Elixir's command line interface Learn Elixir's basic structures by working with numbers Discover atoms, pattern matching, and guards: the foundations of your program structure Delve into the heart of Elixir processing with recursion, strings, lists, and higher-order functions Create Elixir processes and send messages among them Store and manipulate structured data with Erlang Term Storage and the Mnesia database Build resilient applications with the Open Telecom Platform

Discovering Computer Science Addison-Wesley Professional

A new version of the classic and widely used text adapted for the JavaScript programming language. Since the publication of its first edition in 1984 and its second edition in 1996, Structure and Interpretation of Computer Programs (SICP) has influenced computer science curricula around the world. Widely adopted as a textbook, the book has its origins in a popular entry-level computer science course taught by Harold Abelson and Gerald Jay Sussman at MIT. SICP introduces the reader to central ideas of computation by establishing a series of mental models for computation. Earlier editions used the programming language Scheme in their program examples. This new version of the second edition has been adapted for JavaScript. The first three chapters of SICP cover programming concepts that are common to all modern high-level programming languages. Chapters four and five, which used Scheme to formulate language processors for Scheme, required significant revision. Chapter four offers new material, in particular an introduction to the notion of

program parsing. The evaluator and compiler in chapter five introduce a subtle stack discipline to support return statements (a prominent feature of statement-oriented languages) without sacrificing tail recursion. The JavaScript programs included in the book run in any implementation of the language that complies with the ECMAScript 2020 specification, using the JavaScript package sicp provided by the MIT Press website.

Functional Programming CRC Press

Functional programming languages like F#, Erlang, and Scala are attractingattention as an efficient way to handle the new requirements for programmingmulti-processor and high-availability applications. Microsoft's new F# is a truefunctional language and C# uses functional language features for LINQ andother recent advances. Real-World Functional Programming is a unique tutorial that explores thefunctional programming model through the F# and C# languages. The clearlypresented ideas and examples teach readers how functional programming differsfrom other approaches. It explains how ideas look in F#-a functionallanguage-as well as how they can be successfully used to solve programmingproblems in C#. Readers build on what they know about .NET and learn wherea functional approach makes the most sense and how to apply it effectively inthose cases. The reader should have a good working knowledge of C#. No prior exposure toF# or functional programming is required. Purchase of the print book comes with an offer of a free PDF, ePub, and Kindle eBook from Manning. Also available is all code from the book.