Software Engineering Online Course

Thank you very much for downloading **Software Engineering Online Course**. As you may know, people have look hundreds times for their chosen readings like this Software Engineering Online Course, but end up in malicious downloads.
Rather than reading a good book with a cup of coffee in the afternoon, instead they juggled with some malicious virus inside their computer.

Software Engineering Online Course is available in our digital library an online access to it is set as public so you can download it instantly.
Our books collection hosts in multiple countries, allowing you to get the most less latency time to download any of our books like this one.
Kindly say, the Software Engineering Online Course is universally compatible with any devices to read

Engineering Software as a Service Independently Published
Today, software engineers need to know not only how to program effectively but also how to develop proper engineering practices to make their codebase sustainable and healthy. This book emphasizes this difference between programming and software engineering. How can software engineers manage a living codebase that evolves and responds to changing requirements and demands over the length of its life? Based on their experience at Google, software engineers Titus Winters and Hyrum Wright, along with technical writer Tom Manshreck, present a candid and insightful look at how some of the world ' s leading practitioners construct and maintain software. This book covers Google ' s unique engineering culture, processes, and tools and how these aspects contribute to the effectiveness of an engineering organization. You ' ll explore three fundamental principles that software organizations should keep in mind when designing, architecting, writing, and maintaining code: How time affects the sustainability of software and how to make your code resilient over time How scale affects the viability of software practices within an engineering organization What trade-offs a typical engineer needs to make when evaluating design and development decisions

Artificial Intelligence with Python **Pearson Education**

Improve Your Creativity, Effectiveness, and Ultimately, Your Code In Modern Software Engineering, continuous delivery pioneer David Farley helps software professionals

think about their work more effectively, manage it more successfully, and genuinely improve the quality of their applications, their lives, and the lives of their colleagues. Writing for programmers, managers, and technical leads at all levels of experience, Farley illuminates durable principles at the heart of effective software development. He distills the discipline into two core exercises: learning and exploration and managing complexity. For each, he defines principles that can help you improve everything from your mindset to the quality of your code, and describes approaches proven to promote success. Farley's ideas and techniques cohere into a unified, scientific, and foundational approach to solving practical software development problems within realistic economic constraints. This general, durable, and pervasive approach to software engineering can help you solve problems you haven't encountered yet, using today's technologies and tomorrow's. It offers you deeper insight into what you do every day, helping you create better software, faster, with more pleasure and personal fulfillment. Clarify what you're trying to accomplish Choose your tools based on sensible criteria Organize work and systems to facilitate continuing incremental progress Evaluate your progress toward thriving systems, not just more "legacy code" Gain more value from experimentation and empiricism Stay in control as systems grow more complex Achieve rigor without too much rigidity Learn from history and experience

Distinguish "good" new software development ideas from "bad" ones Register your book for convenient access to downloads, updates, and/or corrections as they become available. See inside book for details.

IEEE Computer Society Real-World Software Engineering Problems

Pearson Education

This book seeks to provide an overall view of the nature of software engineering, focusing on real world practice and guiding students of software engineering to understand the benefits and drawbacks of various methods. The text follows the natural life cycle of software development, providing the reader with a comprehensive overview of the software development field. The text includes coverage of methods, tools, principles and guidelines. Case studies and examples are also included throughout the text, providing explicit guidelines for virtually every situation that a software engineer may encounter.Key Features:* Can be used by undergraduates and first year students of software engineering and development courses as well as professionals such as: Information Systems Managers, System Engineers, System Analysts, Software Project Managers, Software Engineers* Each chapter has a summary and exercisesSupplement:Instructor's guide and transparency masters: 0195111532

Optimized C++ Sofware Management

Training

Computer science graduates often find software engineering knowledge and skills are more in demand after they join the industry. However, given the lecture-based curriculum present in academia, it is not an easy undertaking to deliver industry-standard knowledge and skills in a software engineering classroom as such lectures hardly engage or convince students. Overcoming Challenges in Software Engineering Education: Delivering Non-Technical Knowledge and Skills combines recent advances and best practices to improve the curriculum of software engineering education. This book is an essential reference source for researchers and educators seeking to bridge the gap between industry expectations and what academia can provide in software engineering education.

OCA Java SE 7 Programmer I Certification Guide Packt Publishing Ltd

A surprisingly simple way for students to master any subject--based on one of the world's most popular online courses and the bestselling book A Mind for Numbers A Mind for Numbers and its wildly popular online companion course "Learning How to Learn" have empowered more than two million learners of all ages from around the world to master subjects that they once struggled with. Fans often wish they'd discovered these learning strategies earlier and ask how they can help their kids master these skills as well. Now in this new book for kids and teens, the authors reveal how to

make the most of time spent studying. We all have the tools to learn what might not seem to come naturally to us at first--the secret is to understand how the brain works so we can unlock its power. This book explains: • Why sometimes letting your mind wander is an important part of the learning process • How to avoid "rut think" in order to think outside the box • Why having a poor memory can be a good thing • The value of metaphors in developing understanding • A simple, yet powerful, way to stop procrastinating Filled with illustrations, application questions, and exercises, this book makes learning easy and fun.

Overcoming Challenges in Software Engineering Education: Delivering Non-Technical Knowledge and Skills Createspace Independent Pub

Focus on masters' level education in software engineering. Topics discussed include: software engineering principles, current software engineering curricula, experiences with ex- isting courses, and the future of software engineering edu- cation.

Deep Learning for Coders with fastai and PyTorch Springer

"The Fifth SEI Conference on Software Engineering was held in Pittsburgh, Pennsylvania, October 7-8, 1991. This annual conference is a forum for discussion of software engineering education and training among members of the academic, industry, and government communities. It is funded by the Education Program of the Software Engineering Institute, a federallyfunded research and development center of the U.S. Department of Defense. For the first time in

1991 it was held in conjunction with the Association for Computing Machinery and the IEEE Computer Society. Seven sessions addressed: software project courses, software engineering training in government and industry, curriculum issues, software engineering teaching styles, teaching design, topics inreal time and environments, and developing software engineering expertise."--PUBLISHER'S WEBSITE.

System Design Interview - An Insider's Guide IGI Global

Software Engineering: Principles and Practices (SEPP) is intended for use by college or university juniors, seniors, or graduate students who are enrolled in a general one-semester course or two-semester sequence of courses in software engineering and who are majoring in software engineering, computer science, applied computer science, computer information systems, business information systems, information technology, or any other area in which software development is the focus. It is assumed that these students have taken at least two computer programming courses.Because of its sequencing, hierarchical structure, and broad coverage of the system development life cycle (SDLC), SEPP may also be appropriate for use in an introductory survey course in a full-fledged software engineering curriculum. In such a course, the instructor can choose the topics to be covered as well as the depth in which those topics are treated in an effort to provide freshmen or sophomore software engineering students

with a preview of the concepts they will encounter later in the curriculum.

Making Embedded Systems Simon and Schuster

The system design interview is considered to be the most complex and most difficult technical job interview by many. Those questions are intimidating, but don't worry. It's just that nobody has taken the time to prepare you systematically. We take the time. We go slow. We draw lots of diagrams and use lots of examples. You'll learn step-by-step, one question at a time.Don't miss out.What's inside?- An insider's take on what interviewers really look for and why.- A 4-step framework for solving any system design interview question.- 16 real system design interview questions with detailed solutions.- 188 diagrams to visually explain how different systems work.

Continuous Delivery IGI Global

In today’s fast and competitive world, a program’s performance is just as important to customers as the features it provides. This practical guide teaches developers performance-tuning principles that enable optimization in C++. You’ll learn how to make code that already embodies best practices of C++ design run faster and consume fewer resources on any computer—whether it’s a watch, phone, workstation, supercomputer, or globe-spanning network of servers. Author Kurt Guntheroth provides several running examples that demonstrate how to apply these principles incrementally to improve existing code so it meets customer requirements for responsiveness and throughput. The advice in this book will prove itself the first time you hear a colleague exclaim,

"Wow, that was fast. Who fixed something?" Locate performance hot spots using the profiler and software timers Learn to perform repeatable experiments to measure performance of code changes Optimize use of dynamically allocated variables Improve performance of hot loops and functions Speed up string handling functions Recognize efficient algorithms and optimization patterns Learn the strengths—and weaknesses—of C++ container classes View searching and sorting through an optimizer's eye Make efficient use of C++ streaming I/O functions Use C++ thread-based concurrency features effectively

Software Engineering Simon and Schuster

An introductory course on Software Engineering remains one of the hardest subjects to teach largely because of the wide range of topics the area enc- passes. I have believed for some time that we often tend to teach too many concepts and topics in an introductory course resulting in shallow knowledge and little insight on application of these concepts. And Software Engineering is ?nally about application of concepts to e?ciently engineer good software solutions. Goals I believe that an introductory course on Software Engineering should focus on imparting to students the knowledge and skills that are needed to successfully execute a commercial project of a few person-months e?ort while employing proper practices and techniques. It is worth pointing out that a vast majority of the projects executed in the industry today fall in this scope—executed by a small team over a few months. I also believe that by carefully selecting the concepts and topics, we can, in the course of a semester,

achieve this. This is the motivation of this book. The goal of this book is to introduce to the students a limited number of concepts and practices which will achieve the following two objectives: – Teach the student the skills needed to execute a smallish commercial project.

Introduction to Medical Software

Springer Science & Business Media

AUDIENCE Software Engineering: Principles and Practices (SEPP) is intended for use by college or university juniors, seniors, or graduate students who are enrolled in a general one-semester course or two-semester sequence of courses in software engineering and who are majoring in computer science, applied computer science, computer information systems, business information systems, information technology, or any other area in which software development is the focus. It is assumed that these students have taken at least two computer programming courses as well as any additional computing courses required in the first two years of their major. SEPP may also be appropriate for use in an introductory survey course in a full-fledged software engineering curriculum. In such a course, the instructor can choose the topics to be covered as well as the depth in which those topics are treated in an effort to provide freshmen or sophomore software engineering students with a preview of the concepts they will encounter later in their curriculum. SWEBOK CONTENT SEPP covers or touches on most of the topics

listed in the Software Engineering Body of Knowledge (SWEBOK) Guide V3. This guide contains a comprehensive description of the knowledge required of a professional software engineer after four years of experience and is viewed by the IEEE as the authoritative source of software engineering knowledge. In addition, the Guide was used to inform the contents of the Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science and the Software Engineering 2013 Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering, both of which were developed by a joint task force of the IEEE Computer Society (IEEE-CS) and the Association for Computing Machinery (ACM). FEATURES * The beginning of each chapter includes a relevant and thought-provoking quote that can be used by the instructor to pique the interests of his or her students and generate some initial discussion about the topic at hand. * The beginning of each chapter also includes a big question of the form: What is...? The answer to this question is then answered in the following paragraph. This paragraph provides students with both a succinct definition of the term and a context into which the chapter's concepts can be placed. * Since a large amount of information can be represented in a relatively small space using a table, and since a picture is worth a thousand words, the text includes over 230 tables and figures.

* In many places in the text, talking points are displayed as bulleted lists instead of being buried in the narrative. * A significant proportion of the examples in the text are drawn from the real-life experiences of the author's own software development practice that began in 1987. * Every effort has been made to present concepts clearly and logically, utilize consistent language and terminology across all chapters and topics, and articulate concepts fully yet concisely. * Specialized, trendy, and/or arcane language that is inaccessible to the average software development student is either clearly defined or replaced in favor of clear and generalizable terminology. * Although references to the original works that contain the formulas discussed in the text are provided, these formulas have been transformed into a predictable and uniform mathematical notation. * The introductory chapters and the chapters that cover the umbrella activities and tasks of the SDLC include projects that require students to apply something they have learned in the chapters. INSTRUCTOR SUPPLEMENTS * Lecture/Discussion Outlines * PowerPoint Presentations * Test Banks * Real-World Case Studies STUDENT SUPPLEMENTS * Form Templates * Videos

Software Evolution and Maintenance Springer Science & Business Media

Build real-world Artificial Intelligence applications with Python to intelligently interact with the world around you About This Book Step into the amazing world of intelligent apps using this comprehensive guide Enter the world of Artificial Intelligence,

explore it, and create your own applications Work through simple yet insightful examples that will get you up and running with Artificial Intelligence in no time Who This Book Is For This book is for Python developers who want to build real-world Artificial Intelligence applications. This book is friendly to Python beginners, but being familiar with Python would be useful to play around with the code. It will also be useful for experienced Python programmers who are looking to use Artificial Intelligence techniques in their existing technology stacks. What You Will Learn Realize different classification and regression techniques Understand the concept of clustering and how to use it to automatically segment data See how to build an intelligent recommender system Understand logic programming and how to use it Build automatic speech recognition systems Understand the basics of heuristic search and genetic programming Develop games using Artificial Intelligence Learn how reinforcement learning works Discover how to build intelligent applications centered on images, text, and time series data See how to use deep learning algorithms and build applications based on it In Detail Artificial Intelligence is becoming increasingly relevant in the modern world where everything is driven by technology and data. It is used extensively across many fields such as search engines, image recognition, robotics, finance, and so on. We will explore various real-world scenarios in this book and you'll learn about various algorithms that can be used to build Artificial Intelligence applications. During the course of this book, you will find out how to make informed decisions about what algorithms to use in a given context. Starting from the basics of Artificial Intelligence, you will learn how to develop various building blocks using different data mining techniques. You will see how to implement different algorithms to get the best possible results, and will understand how to apply them to real-world scenarios. If you want to add an intelligence

layer to any application that's based on images, text, stock market, or some other form of data, this exciting book on Artificial Intelligence will definitely be your guide! Style and approach This highly practical book will show you how to implement Artificial Intelligence. The book provides multiple examples enabling you to create smart applications to meet the needs of your organization. In every chapter, we explain an algorithm, implement it, and then build a smart application.

Issues in Software Engineering Education

Jones & Bartlett Learning

The first course in software engineering is the most critical. Education must start from an understanding of the heart of software development, from familiar ground that is common to all software development endeavors. This book is an in-depth introduction to software engineering that uses a systematic, universal kernel to teach the essential elements of all software engineering methods. This kernel, Essence, is a vocabulary for defining methods and practices. Essence was envisioned and originally created by Ivar Jacobson and his colleagues, developed by Software Engineering Method and Theory (SEMAT) and approved by The Object Management Group (OMG) as a standard in 2014. Essence is a practice-independent framework for thinking and reasoning about the practices we have and the practices we need. Essence establishes a shared and standard understanding of what is at the heart of software development. Essence is agnostic to any particular method, lifecycle independent, programming language independent, concise, scalable, extensible, and formally specified. Essence frees the practices from their method

prisons. The first part of the book describes Essence, the essential elements to work with, the essential things to do and the essential competencies you need when developing software. The other three parts describe more and more advanced use cases of Essence. Using real but manageable examples, it covers the fundamentals of Essence and the innovative use of serious games to support software engineering. It also explains how current practices such as user stories, use cases, Scrum, and micro-services can be described using Essence, and illustrates how their activities can be represented using the Essence notions of cards and checklists. The fourth part of the book offers a vision how Essence can be scaled to support large, complex systems engineering. Essence is supported by an ecosystem developed and maintained by a community of experienced people worldwide. From this ecosystem, professors and students can select what they need and create their own way of working, thus learning how to create ONE way of working that matches the particular situation and needs.

Multimedia Software Engineering O'Reilly Media

Winner of the 2011 Jolt Excellence Award! Getting software released to users is often a painful, risky, and time-consuming process. This groundbreaking new book sets out the principles and technical practices that enable rapid, incremental delivery of high quality, valuable new functionality to users. Through automation of the build, deployment, and testing process, and improved collaboration between developers, testers, and operations, delivery teams can get changes released in a

matter of hours— sometimes even minutes–no matter what the size of a project or the complexity of its code base. Jez Humble and David Farley begin by presenting the foundations of a rapid, reliable, low-risk delivery process. Next, they introduce the "deployment pipeline," an automated process for managing all changes, from check-in to release. Finally, they discuss the "ecosystem" needed to support continuous delivery, from infrastructure, data and configuration management to governance. The authors introduce state-of-the-art techniques, including automated infrastructure management and data migration, and the use of virtualization. For each, they review key issues, identify best practices, and demonstrate how to mitigate risks. Coverage includes • Automating all facets of building, integrating, testing, and deploying software • Implementing deployment pipelines at team and organizational levels • Improving collaboration between developers, testers, and operations • Developing features incrementally on large and distributed teams • Implementing an effective configuration management strategy • Automating acceptance testing, from analysis to implementation • Testing capacity and other non-functional requirements • Implementing continuous deployment and zero-downtime releases • Managing infrastructure, data, components and dependencies • Navigating risk management, compliance, and auditing Whether you're a developer, systems administrator, tester, or manager, this book will help your organization move from idea to release faster than ever—so you can deliver value to your business rapidly and reliably.

Software Engineering Education Addison-Wesley Professional

AUDIENCE Software Engineering: Principles and Practices (SEPP) is intended for use by college or university juniors, seniors, or graduate students who are enrolled in a general one-semester course or two-semester sequence of courses in software engineering and who are majoring in computer science, applied computer science, computer information systems, business information systems, information technology, or any other area in which software development is the focus. It is assumed that these students have taken at least two computer programming courses as well as any additional computing courses required in the first two years of their major. SEPP may also be appropriate for use in an introductory survey course in a full-fledged software engineering curriculum. In such a course, the instructor can choose the topics to be covered as well as the depth in which those topics are treated in an effort to provide freshmen or sophomore software engineering students with a preview of the concepts they will encounter later in their curriculum. SWEBOK CONTENT SEPP covers or touches on most of the topics listed in the Software Engineering Body of Knowledge (SWEBOK) Guide V3. This guide contains a comprehensive description of the knowledge required of a professional software engineer after four years of experience and is viewed by the IEEE as the authoritative source of software engineering knowledge. In addition, the Guide was used to inform the contents of the Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science and the Software Engineering 2013 Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering, both of which were developed by a joint task force of the IEEE Computer Society (IEEE-CS) and the Association for Computing Machinery (ACM). FEATURES * The beginning of each chapter includes a relevant

and thought-provoking quote that can be used by the instructor to pique the interests of his or her students and generate some initial discussion about the topic at hand. * The beginning of each chapter also includes a big question of the form: What is...? The answer to this question is then answered in the following paragraph. This paragraph provides students with both a succinct definition of the term and a context into which the chapter's concepts can be placed. * Since a large amount of information can be represented in a relatively small space using a table, and since a picture is worth a thousand words, the text includes over 230 tables and figures. * In many places in the text, talking points are displayed as bulleted lists instead of being buried in the narrative. * A significant proportion of the examples in the text are drawn from the real-life experiences of the author's own software development practice that began in 1987. * Every effort has been made to present concepts clearly and logically, utilize consistent language and terminology across all chapters and topics, and articulate concepts fully yet concisely. * Specialized, trendy, and/or arcane language that is inaccessible to the average software development student is either clearly defined or replaced in favor of clear and generalizable terminology. * Although references to the original works that contain the formulas discussed in the text are provided, these formulas have been transformed into a predictable and uniform mathematical notation. * The introductory chapters and the chapters that cover the umbrella activities and tasks of the SDLC include projects that require students to apply something they have learned in the chapters. INSTRUCTOR SUPPLEMENTS * Lecture/Discussion Outlines * PowerPoint Presentations * Test Banks * Real-World Case Studies STUDENT SUPPLEMENTS * Form Templates * Videos

Become an Effective Software Engineering Manager Mit Press

Widely considered one of the best practical

guides to programming, Steve McConnell's original CODE COMPLETE has been helping developers write better software for more than a decade. Now this classic book has been fully updated and revised with leading-edge practices—and hundreds of new code samples—illustrating the art and science of software construction. Capturing the body of knowledge available from research, academia, and everyday commercial practice, McConnell synthesizes the most effective techniques and must-know principles into clear, pragmatic guidance. No matter what your experience level, development environment, or project size, this book will inform and stimulate your thinking—and help you build the highest quality code. Discover the timeless techniques and strategies that help you: Design for minimum complexity and maximum creativity Reap the benefits of collaborative development Apply defensive programming techniques to reduce and flush out errors Exploit opportunities to refactor—or evolve—code, and do it safely Use construction practices that are right-weight for your project Debug problems quickly and effectively Resolve critical construction issues early and correctly Build quality into the beginning, middle, and end of your project

Learning How to Learn O'Reilly Media

Computer Architecture/Software Engineering

Software Engineering Fundamentals CRC Press

Summary This book is a comprehensive guide to the 1Z0-803 exam. You'll explore a wide range of important Java topics as you systematically learn how to pass the certification exam. Each chapter starts with a list of the exam objectives covered in that

chapter. You'll find sample questions and exercises designed to reinforce key concepts and to prepare you for what you'll see in the real exam, along with numerous tips, notes, and visual aids throughout the book. About This Book To earn the OCA Java SE 7 Programmer Certification, you need to know your Java inside and out, and to pass the exam it's good to understand the test itself. This book cracks open the questions, exercises, and expectations you'll face on the OCA exam so you'll be ready and confident on test day. OCA Java SE 7 Programmer I Certification Guide is a comprehensive guide to the 1Z0-803 exam. You'll explore important Java topics as you systematically learn what is required. Each chapter starts with a list of exam objectives, followed by sample questions and exercises designed to reinforce key concepts. It provides multiple ways to digest important techniques and concepts, including analogies, diagrams, flowcharts, and lots of well-commented code. Written for developers with a working knowledge of Java who want to earn the OCA Java SE 7 Programmer I Certification. Purchase of the print book includes a free eBook in PDF, Kindle, and ePub formats from Manning Publications. What's Inside Covers all exam topics Hands-on coding exercises How to avoid built-in traps and pitfalls About the Author Mala Gupta has been training programmers to pass Java certification exams since 2006. She holds OCA Java SE7 Programmer I, SCWCD, and SCJP certifications. Table of Contents

Introduction Java basics Working with Java data types Methods and encapsulation String, StringBuilder, Arrays, and ArrayList Flow control Working with inheritance Exception handling Full mock exam

Testing Computer Software "O'Reilly Media, Inc."

Interested in developing embedded systems? Since they donâ??t tolerate inefficiency, these systems require a disciplined approach to programming. This easy-to-read guide helps you cultivate a host of good development practices, based on classic software design patterns and new patterns unique to embedded programming. Learn how to build system architecture for processors, not operating systems, and discover specific techniques for dealing with hardware difficulties and manufacturing requirements. Written by an expert whoâ??s created embedded systems ranging from urban surveillance and DNA scanners to childrenâ??s toys, this book is ideal for intermediate and experienced programmers, no matter what platform you use. Optimize your system to reduce cost and increase performance Develop an architecture that makes your software robust in resource-constrained environments Explore sensors, motors, and other I/O devices Do more with less: reduce RAM consumption, code space, processor cycles, and power consumption Learn how to update embedded code directly in the processor Discover how to implement complex mathematics on small processors Understand what interviewers look for when you apply for an embedded systems job "Making Embedded Systems is the book for a C programmer who wants to enter the fun (and lucrative) world of

embedded systems. Itâ??s very well writtenâ??entertaining, evenâ??and filled with clear illustrations." â??Jack Ganssle, author and embedded system expert.