Software Engineering Without Degree

Code Complete
Ruby on Rails Tutorial
How to Become a Successful Programmer Without a Degree
Foot in the Door
Become a Software Engineer in 6 Months
Research Software Engineering with Python
Software Engineering Essentials, Volume I
Strategize: Product Strategy and Product Roadmap Practices for the Digital Age
The Hitchhiker's Guide to Coding
How to Become a Software Engineer
Ask a Manager
Software Engineer
Lean Software Systems Engineering for Developers
Beginning Software Engineering
Software Engineering Education
The New Hacker's Dictionary, third edition
Optimized C++
Software Engineering
Guide to the Software Engineering Body of Knowledge (Swebok(r))
Software Engineering for Internet Applications
Software Engineering 2004
Essentials of Software Engineering
The 7 Second CV
Software Engineering Essentials
The Productive Programmer
Developer Hegemony
Software Engineering as a Career
The Complete Software Developer's Career Guide
The Self-Taught Programmer
Fundamentals of Software Engineering
Software Engineering at Google
How the Internet Became Commercial
Software Engineering
Augmented
Software Engineering Measurement
The Self-Taught Computer Scientist
Engineer Your Software!
Artificial Intelligence and Software Engineering

VAUGHAN REGINA

Code Complete CRC Press
Graduate to the next level of your software development career, learning the tools you need to successfully manage the complexity of modern software systems. Whether you are a developer at a small software company, or one of many developers at a large enterprise, your success directly correlates to the ability of your development team to rapidly respond to change. What makes this task challenging in today's world, is that the technical challenges we as developers strive to overcome are becoming increasingly more complex. We have to consider many more options when it comes to things like requirements, solution hosting, support, pace of change, and generally with less time and warning. A good developer knows that it is critical to manage every aspect of software development from soup to nuts, and understands that when details and decisions are left to chance, outcomes can be negatively impacted. Poor planning can result in increased errors, substandard quality, budget and schedule overruns, and result in the ultimate business failure, dissatisfied customers, and stakeholders. This book will help you put on the lenses of a software engineer. You will come away with an understanding of how to view the entire spectrum of the software development process, learn valuable concepts, and apply these principles through meaningful examples, case studies, and source code. What You Will Learn Move beyond being a programmer to being a professional software engineer Spend more time doing software development; minimize time spent dealing with ineffective or inadequate processes Reduce errors in judgment and provide predictable outcomes, while still maintaining agility and responsiveness using Lean and Agile practices Know the steps you can take to ensure a shared understanding among stakeholders Discover tools to validate user experience early and often to minimize costly re-work Develop software designs and architectures that enable long-term business agility Implement patterns and processes that result in "falling into the pit of success" instead of into the "pit of failure" Adopt processes and patterns that will result in pervasive "institutionalized" quality Understand the necessity of redefining the essential role of technical leadership to ensure team maturity and growth This book is for software developers and team leaders who have struggled to implement design and development best practices due to lack of team resources, in-depth knowledge, or experience, and want a book designed to provide the confidence and foundational skills needed to achieve success. Doug Durham is CEO of Don't Panic Labs, a firm that helps companies innovate through the design and development of software technologies. He is also the co-founder of Nebraska Global (parent company of Don't Panic Labs), a pioneer in the startup landscape in Nebraska. Doug has more than three decades of software engineering and development experience in aerospace and defense, healthcare, manufacturing, ecommerce, consumer web applications, and Internet network services. He is passionate about the process of solving problems through software, and the application of sound engineering principles and patterns to these efforts. Doug has taught at the University of Nebraska-Lincoln Raikes School of Computer Science, and serves on a College of Engineering advisory board. He often speaks at industry conferences on the topic of software engineering, and is a frequent guest lecturer at the University of Nebraska-Lincoln. Chad Michel is Lead Software Architect for Don't Panic Labs and has more than 20 years of software development and engineering experience. He helps clients solve problems through innovative software solutions. He graduated from the University of Nebraska-Lincoln with a bachelor's degree in computer engineering and a master's degree in computer science. Chad has worked for several companies in Lincoln, where he helped build a practice management application for lawyers, developed key features for an ecommerce application, and helped wrangle a content delivery network into a stable platform. He often speaks at technical meetups hosted by Don't Panic Labs, as well as at many other conferences and technical groups. He regularly contributes to the Don't Panic Labs blog. Chad is a fourth-degree black belt in Tae Kwon Do. .
Ruby on Rails Tutorial Independently Published
Computer Architecture/Software Engineering
How to Become a Successful Programmer Without a Degree "O'Reilly Media, Inc."
Are you looking to change careers? Ever think of software development?Software Developers are in high demand and can earn a high income. Believe it or not, these jobs also don't require a college degree. Many people are able to become self-taught software engineers and developers everyday. Learn how to take those steps with this guide. With this short book, you can learn how to take the first steps. It will teach you strategies for how to choose a language to learn, how and where to learn a programming language. The Hitchhiker's Guide to Coding will also help you understand how to gain some experience as a new coder. You can discover strategies for finding and applying to jobs and finally, what to expect when on the job. All of this is brought to you by an author with over 16 years of experience in software engineering.Download your copy today and embark on your journey to an exciting career in programming!
Foot in the Door Addison-Wesley Professional
Starting a career as a software engineer without a computer science degree is a long and difficult journey, Hasan Armstrong discovered this whilst attempting to switch from a career in healthcare to software engineering. He now works as a software engineer and incorporates all the lessons he has learnt in this book. This book will provide a roadmap to getting a job as a software engineer without a computer science degree, as well as providing solutions to the obstacles you may face along the way, like learning new programming languages, handling interview questions, negotiating job offers and much more. Through his youtube channel, Hasan has helped several thousands of people learn to code. What you will learn in this book? How to determine if a job as a software engineer is even for you? Should you become a front-end, backend or full stack software engineer? Mindsets and habits of software engineers who seek excellence. Programming topics you will need to learn and practice before you can start applying for software engineering roles. Practices to stay healthy, avoid burnout syndrome and remain happy and fulfilled as a self-taught software engineer. Increase the likelihood of landing a software engineering role, by creating a personal brand, a CV that stands out and finding companies you want to work for. Mindsets and habits of exceptional software

engineers Interviewer asks "What kind of salary do you expect for this role?" - How should you reply? You've started working as a software engineer. How can you climb the career ladder? The dark side of working as a software engineer. How should you handle workplace politics, mental health issues and technical debt? We are keen to help you land a software engineering role and help you progress in that role. So if you want to know if software engineering is for you, in the process of learning to code or applying for software engineering roles this book is worth purchasing. **Buy the paperback version of this book, and get the kindle version absolutely FREE**

Become a Software Engineer in 6 Months Robinson

Software development is hard, but creating good software is even harder, especially if your main job is something other than developing software. Engineer Your Software! opens the world of software engineering, weaving engineering techniques and measurement into software development activities. Focusing on architecture and design, Engineer Your Software! claims that no matter how you write software, design and engineering matter and can be applied at any point in the process. Engineer Your Software! provides advice, patterns, design criteria, measures, and techniques that will help you get it right the first time. Engineer Your Software! also provides solutions to many vexing issues that developers run into time and time again. Developed over 40 years of creating large software applications, these lessons are sprinkled with real-world examples from actual software projects. Along the way, the author describes common design principles and design patterns that can make life a lot easier for anyone tasked with writing anything from a simple script to the largest enterprise-scale systems.

Research Software Engineering with Python CRC Press

Discover the foundations of software engineering with this easy and intuitive guide In the newly updated second edition of Beginning Software Engineering, expert programmer and tech educator Rod Stephens delivers an instructive and intuitive introduction to the fundamentals of software engineering. In the book, you'll learn to create well-constructed software applications that meet the needs of users while developing the practical, hands-on skills needed to build robust, efficient, and reliable software. The author skips the unnecessary jargon and sticks to simple and straightforward English to help you understand the concepts and ideas discussed within. He also offers you real-world tested methods you can apply to any programming language. You'll also get: Practical tips for preparing for programming job interviews, which often include questions about software engineering practices A no-nonsense guide to requirements gathering, system modeling, design, implementation, testing, and debugging Brand-new coverage of user interface design, algorithms, and programming language choices Beginning Software Engineering doesn't assume any experience with programming, development, or management. It's plentiful figures and graphics help to explain the foundational concepts and every chapter offers several case examples, Try It Out, and How It Works explanatory sections. For anyone interested in a new career in software development, or simply curious about the software engineering process, Beginning Software Engineering, Second Edition is the handbook you've been waiting for.

Sofware Management Training

The Internet and smartphone are just the latest in a 250-year- long cycle of disruption that has continuously changed the way we live, the way we work and the way we interact. The coming Augmented Age, however, promises a level of disruption, behavioural shifts and changes that are unparalleled. While consumers today are camping outside of an Apple store waiting to be one of the first to score a new Apple Watch or iPhone, the next generation of wearables will be able to predict if we're likely to have a heart attack and recommend a course of action. We watch news of Google's self-driving cars, but don't likely realise this means progressive cities will have to ban human drivers in the next decade because us humans are too risky. Following on from the Industrial or machine age, the space age and the digital age, the Augmented Age will be based on four key disruptive themes—Artificial Intelligence, Experience Design, Smart Infrastructure, and HealthTech. Historically the previous 'ages' bought significant disruption and changes, but on a net basis jobs were created, wealth was enhanced, and the health and security of society improved. What will the Augmented Age bring? Will robots take our jobs, and AI's subsume us as inferior intelligences, or will this usher in a new age of abundance? Augmented is a book on future history, but more than that, it is a story about how you will live your life in a world that will change more in the next 20 years than it has in the last 250 years. Are you ready to adapt? Because if history proves anything, you don't have much of a choice.

Software Engineering Essentials, Volume I Random House

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

Strategize: Product Strategy and Product Roadmap Practices for the Digital Age Springer

The product of many years of practical experience and research in the software measurement business, this technical reference helps you select what metrics to collect, how to convert measurement data to management information, and provides the statistics necessary to perform these conversions. The author explains how to manage software development

The Hitchhiker's Guide to Coding Simple Programmer, LLC

Software Engineering as a CareerIndependently Published

How to Become a Software Engineer BlogIntoBook.com

As the Digital Age continues to flourish, electronic devices¿found in everything from refrigerators to wristwatches¿are a constant, if not unavoidable, part of everyday life. None of these devices can work without software, and therefore, without the help of software engineering professionals. As the

demand for computing devices continues to grow, so too does the demand for software engineers. What the job entails, what it pays, and future prospects are discussed along with insights from industry insiders.

Ask a Manager Pearson Education

From the creator of the popular website Ask a Manager and New York's work-advice columnist comes a witty, practical guide to 200 difficult professional conversations—featuring all-new advice! There's a reason Alison Green has been called "the Dear Abby of the work world." Ten years as a workplace-advice columnist have taught her that people avoid awkward conversations in the office because they simply don't know what to say. Thankfully, Green does—and in this incredibly helpful book, she tackles the tough discussions you may need to have during your career. You'll learn what to say when • coworkers push their work on you—then take credit for it • you accidentally trash-talk someone in an email then hit "reply all" • you're being micromanaged—or not being managed at all • you catch a colleague in a lie • your boss seems unhappy with your work • your cubemate's loud speakerphone is making you homicidal • you got drunk at the holiday party Praise for Ask a Manager "A must-read for anyone who works . . . [Alison Green's] advice boils down to the idea that you should be professional (even when others are not) and that communicating in a straightforward manner with candor and kindness will get you far, no matter where you work."—Booklist (starred review) "The author's friendly, warm, no-nonsense writing is a pleasure to read, and her advice can be widely applied to relationships in all areas of readers' lives. Ideal for anyone new to the job market or new to management, or anyone hoping to improve their work experience."—Library Journal (starred review) "I am a huge fan of Alison Green's Ask a Manager column. This book is even better. It teaches us how to deal with many of the most vexing big and little problems in our workplaces—and to do so with grace, confidence, and a sense of humor."—Robert Sutton, Stanford professor and author of The No Asshole Rule and The Asshole Survival Guide "Ask a Manager is the ultimate playbook for navigating the traditional workforce in a diplomatic but firm way."—Erin Lowry, author of Broke Millennial: Stop Scraping By and Get Your Financial Life Together

Software Engineer Routledge

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

Lean Software Systems Engineering for Developers Independently Published

Create a winning game plan for your digital products with Strategize: Product Strategy and Product Roadmap Practices for the Digital Age, 2nd edition. Using a wide range of proven techniques and tools, product management expert Roman Pichler explains how to create a winning product strategy and actionable roadmap. Comprehensive and insightful, the book will enable you to make the right strategic decisions in today's dynamic digital age. If you work as a product manager, Scrum product owner, product portfolio manager, head of product, or product coach, then this book is for you. What you will learn: * Create an inspiring vision for your product. * Develop a product strategy that maximises the chances of launching a winning product. * Successfully adapt the strategy across the product life cycle to achieve sustained product success. * Measure the value your product creates using the right key performance indicators (KPIs). * Build an actionable outcome-based product roadmap that aligns stakeholders and directs the product backlog. * Regularly review the product strategy and roadmap and keep them up-to-date. Written in an engaging and easily accessible style, Strategize offers practical advice and valuable examples so that you can apply the practices directly to your products. This second, revised, and extended edition offers new concepts, more tools, and additional tips and examples. Praise for Strategize: "Strategize offers a comprehensive approach to product strategy using the latest practices geared specifically to digital products. Not just theory, the book is chock-full of real-world examples, making it easier to apply the principles to your company and products. Strategize is essential reading for everyone in charge of products: product executives, product managers, and product owners." Steve Johnson, Founder at Under10 Consulting. "Whether you are new to product management or an experienced practitioner, Strategize is a must read. You are guaranteed to get new ideas about how to develop or improve your product strategy and how to execute it successfully. It's an essential addition to every product manager's reading list." Marc Abraham, Senior Group Product Manager at Intercom.

Beginning Software Engineering John Wiley & Sons

The follow-up to Cory Althoff's bestselling The Self-Taught Programmer, which inspired hundreds of thousands of professionals to learn to program outside of school! Fresh out of college and with just a year of self-study behind him, Cory Althoff was offered a dream first job as a software engineer for a well-known tech company, but he quickly found himself overwhelmed by the amount of things he needed to know, but hadn't learned yet. This experience combined with his personal journey learning to program inspired his widely praised guide, The Self-Taught Programmer. Now Cory's back with another guide for the self-taught community of learners focusing on the foundations of computer science. The Self-Taught Computer Scientist introduces beginner and self-taught programmers to computer science fundamentals that are essential for success in programming and software engineering fields. Computer science is a massive subject that could cover an entire lifetime of learning. This book does not aim to cover everything you would learn about if you went to school to get a computer science degree. Instead, Cory's goal is to give you an introduction to some of the most important concepts in computer science that apply to a programming career. With a focus on data structures and algorithms, The Self-Taught Computer Scientist helps you fill gaps in your knowledge, prepare for a technical interview, feel knowledgeable and confident on the job, and ultimately, become a better programmer. Learn different algorithms including linear and binary

search and test your knowledge with feedback loops Understand what a data structure is and study arrays, linked lists, stacks, queues, hash tables, binary trees, binary heaps, and graphs Prepare for technical interviews and feel comfortable working with more experienced colleagues Discover additional resources and tools to expand your skillset and continue your learning journey It's as simple as this: You have to study computer science if you want to become a successful programmer, and if you don't understand computer science, you won't get hired. Ready for a career in programming, coding, or software engineering and willing to embrace an "always be learning" mindset? The Self-Taught Computer Scientist is for you.

Software Engineering Education Jones & Bartlett Learning

This new edition of the hacker's own phenomenally successful lexicon includes more than 100 new entries and updates or revises 200 more. This new edition of the hacker's own phenomenally successful lexicon includes more than 100 new entries and updates or revises 200 more. Historically and etymologically richer than its predecessor, it supplies additional background on existing entries and clarifies the murky origins of several important jargon terms (overturning a few long-standing folk etymologies) while still retaining its high giggle value. Sample definition hacker n. [originally, someone who makes furniture with an axe] 1. A person who enjoys exploring the details of programmable systems and how to stretch their capabilities, as opposed to most users, who prefer to learn only the minimum necessary. 2. One who programs enthusiastically (even obsessively) or who enjoys programming rather than just theorizing about programming. 3. A person capable of appreciating {hack value}. 4. A person who is good at programming quickly. 5. An expert at a particular program, or one who frequently does work using it or on it; as in `a UNIX hacker'. (Definitions 1 through 5 are correlated, and people who fit them congregate.) 6. An expert or enthusiast of any kind. One might be an astronomy hacker, for example. 7. One who enjoys the intellectual challenge of creatively overcoming or circumventing limitations. 8. [deprecated] A malicious meddler who tries to discover sensitive information by poking around. Hence `password hacker', `network hacker'. The correct term is {cracker}. The term 'hacker' also tends to connote membership in the global community defined by the net (see {network, the} and {Internet address}). It also implies that the person described is seen to subscribe to some version of the hacker ethic (see {hacker ethic, the}). It is better to be described as a hacker by others than to describe oneself that way. Hackers consider themselves something of an elite (a meritocracy based on ability), though one to which new members are gladly welcome. There is thus a certain ego satisfaction to be had in identifying yourself as a hacker (but if you claim to be one and are not, you'll quickly be labeled {bogus}). See also {wannabee}.

The New Hacker's Dictionary, third edition Nicholas Brealey International

SE 2004 provides guidance on what should constitute an undergraduate software engineering education. This report takes into account much of the work that has been done in software engineering education over the last quarter of a century. This volume represents the first such effort by the ACM and the IEEE-CS to develop curriculum guidelines for software engineering.

Optimized C++ "O'Reilly Media, Inc."

How To Teach Yourself Coding And Land The Tech Job Of Your Dreams Without A Degree The tech industry is booming. Hundreds of thousands of new tech jobs appear every year. The average salary in tech is almost two times higher than the national average salary in the USA. Tech companies are ranked among the world's best employers. Besides, the tech industry is wonderfully diverse and you're sure to find a job that matches your personal strengths. You can master Python and work on machine learning algorithms. You can become a QA tester... or maybe your dream job is all about improving a search engine. Does this sound good enough for you? But what if you've missed your opportunity to get a degree in programming or engineering? Can you still get your foot in the door and join the ranks of IT professionals if you're a self-taught programmer without formal qualifications? Nadya Primak's answer is YES. And she knows what she's talking about because she's been through it herself: she studied Visual Arts, but now she's a software engineer with a 6-figure salary. If you want to be like her, check out her book. It's a practical guide to learning coding skills, job hunting in tech, interview preparation and much more! Here's what she'll teach you: How to leverage your existing knowledge and skills to start a career in tech How to choose good coding courses that match your goals and learning style How to find the courage to break away with your boring job How to survive your interview for your first tech job... and make it the start of a new exciting career! Finally, how to land your dream job in tech! You don't have to follow in Nadya's footsteps exactly - maybe your path towards your dream tech job will be different. But her experience is sure to teach you a lot! Take your first step towards your dream career. Scroll up, click on "Buy Now with 1-Click", and Get Your Copy Now!

Software Engineering Educreation Publishing

After completing this self-contained course on server-based Internet applications software that grew out of an MIT course, students who start with only the knowledge of how to write and debug a computer program will have learned how to build sophisticated Web-based applications.

Guide to the Software Engineering Body of Knowledge (Swebok(r)) Morgan & Claypool Publishers

This is the eBook of the printed book and may not include any media, website access codes, or print supplements that may come packaged with the bound book. Used by sites as varied as Twitter, GitHub, Disney, and Airbnb, Ruby on Rails is one of the most popular frameworks for developing web applications, but it can be challenging to learn and use. Whether you’re new to web development or new only to Rails, Ruby on Rails™ Tutorial, Fourth Edition, is the solution. Best-selling author and leading Rails developer Michael Hartl teaches Rails by guiding you through the development of three example applications of increasing sophistication. The tutorial’s examples focus on the general principles of web development needed for virtually any kind of website. The updates to this edition include full compatibility with Rails 5, a division of the largest chapters into more manageable units, and a huge number of new exercises interspersed in each chapter for maximum reinforcement of the material. This indispensable guide provides integrated tutorials not only for Rails, but also for the essential Ruby, HTML, CSS, and SQL skills you need when developing web applications. Hartl explains how each new technique solves a real-world problem, and then he demonstrates it with bite-sized code that’s simple enough to understand, yet novel enough to be useful. Whatever your previous web development experience, this book will guide you to true Rails mastery. This book will help you Install and set up your Rails development environment, including pre-installed integrated development environment (IDE) in the cloud Go beyond generated code to truly understand how to build Rails applications from scratch Learn testing and test-driven development (TDD) Effectively

use the Model-View-Controller (MVC) pattern Structure applications using the REST architecture Build static pages and transform them into dynamic ones Master the Ruby programming skills all Rails developers need Create high-quality site layouts and data models Implement registration and authentication systems, including validation and secure passwords Update, display, and delete users

Upload images in production using a cloud storage service Implement account activation and password reset, including sending email with Rails Add social features and microblogging, including an introduction to Ajax Record version changes with Git and create a secure remote repository at Bitbucket Deploy your applications early and often with Heroku