A

ACE THE PROGRAMMING INTERVIEW

160 QUESTIONS AND ANSWERS FOR SUCCESS

EDWARD GUINESS

WILEY

Table of Contents

Chapter 1: Hiring Programmers: The Inside Story

- Reasons They Recruit
- Talking to Managers
- Preparing Your CV
- Using Job Sites
- Recruitment Agencies
- Searching for Jobs Yourself
- Emerging Alternatives

Chapter 2: Handling the Phone Interview with Confidence

- Knowing What to Expect

Chapter 3: In-Person Interviews

- Preparing for the Interview
- The Most Important Thing
- The Second Most Important Thing
- Communicating Effectively

Chapter 4: Negotiating a Job Offer

- Understanding the Market
- Doing the Numbers
- The Role of the Recruiting Agent
- Start as You Mean to Go On
- Evaluating a Contract
- What to Do If Things Go Wrong
- Summary of Negotiating Tips

Chapter 5: Programming Fundamentals

Understanding Binary, Octal, Hexadecimal
Understanding Data Structures
Sorting
Working with Recursion
Modeling with Object-Oriented Programming
Thinking Like a Functional Programmer
Understanding SQL
Full-Stack Web Development
Deciphering Regular Expressions
Recognizing Hard Problems
QUESTIONS
ANSWERS

Chapter 6: Code Quality
Writing Clear Code
Writing Expressive Code
Measuring Efficiency and Performance
Understanding What “Modular” Means
Understanding the SOLID principles
Avoiding Code Duplication
QUESTIONS
ANSWERS

Chapter 7: The Usual Suspects
Concurrent Programming
Relational Databases
Pointers
Design Issues
Bad Habits

QUESTIONS
ANSWERS

Chapter 8: Quirks and Idioms

Binary Fractions and Floating Point Numbers
QUESTIONS
JavaScript
QUESTIONS
C#
QUESTIONS
Java
QUESTIONS
Perl
QUESTIONS
Ruby
QUESTIONS
Transact-SQL
QUESTIONS
ANSWERS

Chapter 9: Testing—Not Just for Testers

Unit Tests
Test-Driven Development
Writing Good Unit Tests
Testing Slow Things
Unit Testing Frameworks
Mock Objects
QUESTIONS
ANSWERS

Chapter 10: The Right Tools
- Exploring Visual Studio
- QUESTIONS
- Exploiting Command-Line Tools
- QUESTIONS
- Understanding PowerShell
- QUESTIONS
- Troubleshooting with Utilities from Sysinternals
- QUESTIONS
- Managing Source Code
- QUESTIONS
- QUESTIONS
- QUESTIONS
- ANSWERS

Chapter 11: Notorious Interview Questions
- Estimating on the Spot
- QUESTIONS
- Solving Puzzles and Brain-Teasers
- QUESTIONS
- Solving Probability Problems
- QUESTIONS
- Coping with Concurrency
- QUESTIONS
- Doing Tricks with Bits
- QUESTIONS
- Devising Recursive Algorithms
- QUESTIONS
- Understanding Logic Gates

QUESTIONS
Writing Code to…Prove You Can Code
QUESTIONS
Answers

Chapter 12: Programming Wisdom
QUESTIONS
ANSWERS

Appendix A: Preparing Your Cheat Sheets
General and Behavioral
Programming, General
Programming Concepts
Work History
Questions to Ask, If Given the Opportunity

Introduction
Code for this Book
How This Book is Organized

Chapter 1

Hiring Programmers: The Inside Story

When I was a young boy, making new friends seemed easy. I had grown up with the surreal humor of Monty Python and my usual approach to a potential new friend would be something like *"I am a knight who says Ekki-Ekki-Ekki-PTANG*!" I thought it was hilarious. I made a few great friends (one is still a friend 30 years later), but I also had a lot of misses. Actually, mostly I had misses, nearly all the time. Sometimes my approach would generate open hostility. I couldn't understand it.

What my young self didn't realize was that enthusiasm for the absurd was not a universal constant. Not all kids my age had seen Monty Python, and even if they had, not everyone loved and appreciated it like I did. I was seeing the world through my own Python-shaped glasses, and I did not appreciate the diversity of other childhood experiences. Not everyone was on the same wavelength.

As naïve as this might seem, many hiring managers make the same basic mistake when interviewing. Perhaps they suppose that because they have a lot of hard-won experience in a certain area then *of course* everyone with experience in that area will see things the same way. Further, they might assume that their thought process will be similar. At the interview the hiring manager might abruptly open the conversation with the equivalent of my Python-inspired icebreaker:

"*Nice to meet you*; *now, could you please describe a situation where it would be inappropriate to normalize a set of database tables into Boyce-Codd normal form*."

It might be that you love Monty Python and, for you, meeting an interviewer who quotes from Python might seem like a dream come true. If that is the case, then I wish you all the best; just be careful how you respond when the interviewer asks you to "*Please, walk this way*."

For the rest of us, establishing a level of communication that is easy and familiar will take some preparation and effort.

It might have occurred to you that I'm hinting at the concept of rapport. *Rapport* is an excellent word; it describes feelings of harmony, of sympathy, and of being on the same wavelength. I hesitate to use this word only because it has been somewhat hijacked in recent times and now comes loaded with connotations of insincerity, like the fake grin of a novice salesman.

But for an interview to be really effective, to have the ideal degree of communication, and to put yourself on common ground with the interviewer, you really do need to work on establishing a rapport.

One of the simplest and most effective ways to start building rapport is to try to see things from the interviewer's perspective. If you understand the motivations of the interviewer, establishing a common ground and adapting your responses appropriately

becomes much easier. You can quickly home in on what the interviewer is looking for, in both a positive and negative sense.

In this chapter, I will take a thorough look at the process of finding a programming job including:

- What motivates a hiring manager, and how to tailor your approach appropriately
- How to prepare a CV that will get you to an interview
- How to use job sites
- Understanding recruitment agencies; how they work and how to work effectively with them
- How to find jobs without a recruitment agency (it can be done!)

Let's start by taking a look at some of the most common reasons why a company might want to hire a programmer.

Reasons They Recruit

Without exception, a company hiring a programmer will have a reason for hiring. If you know what that reason is and understand the motivation for it, then you can optimize your approach accordingly.

Planned expansion

A common scenario—one that will become increasingly common as the major world economies resume growing—is for companies to make medium- to long-term plans for expansion that require them to take on more programmers in accordance with their plans for business growth.

The interviewer's motivation and approach

Because the role is part of longer term plans, the interviewer is unlikely to feel a great sense of urgency. Well-prepared interviewers will have a job profile, perhaps also a person profile, and be comparing candidates to these. With time on their side, they are less likely to compromise their pre-determined requirements, and although they might not realistically expect to perfectly match every aspect, they will probably be less open to considering candidates who deviate by any significant degree.

Your approach

Your approach should be to highlight areas where your skill and experience matches well with the advertised job profile. That is the easy part.

What about skills that aren't a good fit? For example, suppose you apply for the role of a .NET programmer and during the interview it becomes apparent that the interviewer has an expectation that the ideal candidate will have experience of a certain component library, which happens to be one you've not used. In areas where your background is not such a good fit, you have three basic options.

Play it down

Your first option is to downplay the perceived gap in skills—including the option of substituting other experience as being of equivalent value. If you decide to play it down, you might say:

"*It's been my experience that it never takes long to learn the basics of a new component library, since, as programmers, we face an endless supply of new components and frameworks both open source and from the major vendors*."

You might also comment that learning is part of the job:

"*One thing I really like about programming is the experience of learning new technologies and platforms. It's part of the ongoing attraction of the job*."

The biggest risk in taking this approach is that you might appear evasive, so be wary of overdoing it. After all, it is unlikely the interviewer chose the requirement at random, and if you push too hard at downplaying the requirement you might inadvertently maneuver yourself into an argument. Be sure to avoid that.

Take it on the chin

Your second option is to take it on the chin, so to speak. Take ownership of the "development opportunity" and perhaps talk briefly about how you have acquired

other, similar skills or experiences.

If you decide to take it on the chin, simply agree with the interviewer's observation and at the same time show your enthusiasm for learning something new:

"*I don't have experience of that particular technology but I would really enjoy learning it*."

If the interviewer persists, you might feel it appropriate to ask how other developers in the team might learn new skills:

"*Could you describe how the developers in your team generally learn new skills*?"

Each example of learning given by the interviewer is an opportunity to show how, as a part of the team, you would benefit from the same approach and so acquire the necessary skill.

Understand the requirement

Your third option is to explore the interviewer's motivation, to gently probe the motives underpinning the requirement.

Exploring the underpinning motivation of the requirement gives you the best chance of looking good, but to take this approach you must have established a reasonable rapport; otherwise, you risk appearing argumentative. The basic idea is that you explore the requirement looking to show that you understand and can meaningfully address the underlying requirement despite lacking a specific skill or certain experience.

For instance, you might lack experience of a particular IoC container, let's say Microsoft Unity. You might ask the reason for using that particular implementation of IoC:

"*I know there are a few good reasons for using an inversion of control pattern*; *could you describe how the Microsoft Unity framework helps with the kind of work you do here*?"

If the answer is to encourage loose-coupling of components, then this gives you an opening to talk about your understanding of dependency injection principles. If the answer is to encourage the writing of code that is structured in a way that better supports unit testing, then you can talk about your experience of refactoring legacy code to add unit tests, or you could talk about your understanding of dependency injection without using an IoC container.

If the interviewer asks a pointed question, or simply makes a challenging statement regarding the relevance of your experience in a certain area, your response should be respectful but equally forthright. Often the case is that an interviewer with a blunt approach is looking for a similarly direct approach in the interviewee.

Challenge: "*I don't see any experience of Microsoft Unity in your CV. We use that, so your experience doesn't match our job spec*."

Response: "*Is that a deal-breaker*?"

The principle at work here is *mirroring* the behavior of the interviewer (refer to the section on establishing rapport in Chapter 3 for a discussion of this powerful technique).

Whatever approach you take, keep in mind that you should not dwell on any

particular mismatch. In particular, keep your comments brief and to the point. The more you talk about it, the more prominence it will have in the interviewer's memory of the interview when they reflect afterward. There's not much you can do if the interviewer appears to be stuck on an apparent mismatch—just keep your comments brief and resist the temptation to ramble on the topic.

Specific projects

When a company spots an opportunity in the market, it might scramble to put together a development team focused on delivering a solution to capitalize on the opportunity. There could be pressure to be first to market, or perhaps the commercial opportunity is constrained to a limited window of time.

The interviewer's motivation and approach

The interviewer wants to know that you can work under some pressure, and that you are someone who finishes what you start. Sometimes that pressure to hire can relax the strictness with which the interviewer will match your experience against the details of the job specification, although of course you can't assume that will be the case.

Your approach

Be aware that although you need to demonstrate how your skills and experience are a good match for the job as advertised, the interviewer also probably has the needs of a specific project in mind. The company might have adopted the job specification to suit the project, but more often than not interviewers reuse a standard job description and look for the “extras” at the interview.

Your enthusiasm and ability to adapt might count for more than usual. Showing an ability to quickly grasp key aspects of the project gives you an advantage over applicants who stick to the published job specification.

Many consulting and service-oriented companies routinely put new software development teams together to respond to customer demand, so the chances are higher that this type of company will recruit with a specific project in mind. If you aren't sure of a company's motivation for hiring, there's absolutely no harm in asking directly:

“*Could I ask why you are recruiting*? *Is it for a specific project*?”

Replacing someone

A third common reason for hiring a programmer is simply to replace someone who is leaving or who has left the company.

The interviewer's motivation and approach

The interviewer will have similar motivation as in the planned expansion scenario. They will probably be under some time pressure, but not as much as if they were recruiting for a specific project.

An interesting aspect of this situation is that they will probably have experience of a previous person filling this role, for better or for worse, and will therefore have a list (written or not) of things they want to ensure the next person will and won't bring to the role. For example, perhaps the previous person was a stickler for detail, and this might, therefore, be high on the list of personal characteristics the interviewer wants to confirm in you.

Your approach

Obviously, unless you're very lucky, you're unlikely to gather insight into what the interviewer liked and disliked about the person you hope to replace prior to the interview. What you can do at the interview is ask about unique challenges of the role, things for which you might to need to be on guard, and so on:

"*Could you tell me a bit about the challenges of this role that might make it different from the usual programming job*?"

Experienced interviewers are unlikely to volunteer much information about the previous person, but they might give you clues along the lines of how certain attributes are important; for instance, "the ability to get along with the team." If you ask the right kind of question, you might get vital clues about the things you need to highlight with regard to your experience and ability:

"*Can I ask whether there have been issues about how the team has been working together that make this ability particularly important*?"

Talking to Managers

It is a story often told. A capable and bright programmer impresses everyone and is promoted to team leader or manager. The newly promoted manager takes on the responsibility of hiring new programmers and uses his awesome interpersonal skills to hire more bright programmers.

The problem is that this is almost never how it happens. Many otherwise excellent programmers simply don't have awesome interpersonal skills, and sometimes these are the people who will be running the interview and interviewing you.

Is that a good thing, or is it bad? It depends. It could mean you suffer a terrible interview experience—in which case, count yourself lucky you won't be working for a poor communicator—or it could be a huge advantage. Think of it like this: In every human relationship, having things in common helps, and the more you have in common the easier breaking the ice and enjoying a conversation will be.

Now, as one programmer talking to another programmer (even if the manager isn't currently working as a programmer) what might you have in common? The answer is “lots.” Have you ever spent hours or days tracking down a difficult bug? Sure you have—and almost certainly the manager has, too. What do you like about a particular programming language? The manager might feel the same. Do you regularly visit a website for programmers? The manager might, too. Do you have a favorite XKCD cartoon? Do you visit http://thedailywtf.com? What is your favorite programming book? Have you ever used the word “nullable” in conversation with a non-techie? What annoys you about the IDE you use? There are lots of topics you can discuss together.

Tech talk—don't hold back

It might become apparent during the interview that the interviewer is not as technical as you might have assumed. He might have a background in project management or product marketing, for example. How should you react? Don't make the mistake of thinking you need to “dumb it down” for the interview. The problem with “dumbing it down” is that you put yourself at a disadvantage when the interviewer compares your responses to another candidate. He might not understand everything you say, but his impression of your response will be colored by the language you use, and if you talk purely in metaphors (for example) then you risk giving the impression that you don't actually know the subject as well as the other candidate who talks about specific language or framework features using their proper names.

If a non-technical interviewer asks you a technical question then he expects a technical answer. The case might be that he has a “cheat sheet” of his own, a list of questions and answers, for example.

Don't hold back—answer the question fully and as you would if talking to a technical interviewer. If the non-technical interviewer wants you to explain something in non-technical terms, then you could use a metaphor (see the next section “Using metaphors”). As a rule, keep your answers grounded in reality using real names and proper terms for things.

Using metaphors

Sometimes a non-technical interviewer wants to assess how well you can explain a technical subject to someone who is non-technical. In this case, a metaphor is your best bet.

For example, suppose you are asked to explain the concept of an IP address in non-technical terms. Here is the definition from Wikipedia:

> **An Internet Protocol address (IP address) is a numerical label assigned to each device (e.g., computer, printer) participating in a computer network that uses the Internet Protocol for communication.**

The first metaphor that might occur to you is that an IP address is much like a mailing address. A mailing address is used to deliver mail, and an IP address is used to deliver packets of information, so the metaphor seems to work. On the other hand, mailing addresses vary widely in format and local conventions, whereas an IP address conforms to a strict set of rules that are consistently and globally applied. Perhaps a better metaphor would be latitude and longitude?

While metaphors might conveniently help a non-technical person relate to an aspect of a technical subject, they are almost by definition an imperfect representation. They have limits, and you should never cling to an imperfect metaphor after reaching those limits. For example, if you use latitude and longitude as a metaphor for an IP address, presumably you don't mean to imply that IP addresses are assigned purely due to the physical location of the computer or device.

Preparing Your CV

A good CV (also known as a resumé) gets you past the filters of recruitment agencies and human resource (HR) departments. A good CV will mean your application is not rejected out of hand, and it could give you an opportunity to talk to someone about the job role. A CV by itself is never going to win you the job. In other words, no hiring manager ever decides to hire someone purely on the strength of her CV alone.

Include relevant keywords but keep them in context

Knowing that, in most cases, non-technical agents will filter your CV has one very important implication—you need to be sure to include keywords that are relevant to the job and in particular to the job advertisement. The technical hiring manager might well understand that working on the ECMAScript specification means you have excellent knowledge of JavaScript, but the typical recruiter will not see the connection unless you spell it out and include the keyword “JavaScript” in your CV. Of course, you also want to avoid the appearance of insincerely stuffing your CV full of keywords, so ensure that any list of keywords is presented in proper context.

Write as well as you can

Poor writing puts you at a severe disadvantage. What you write must be clear and concise. You must proofread (or have a friend proofread) and ruthlessly rewrite or delete anything that isn't clear or relevant.

When I am unsure how to write something, I find that pretending to tell it to a friend is helpful. If you have no friends, tell it to your hand, and then write down the words you used. It doesn't matter what they are at first because the next step is to revise those words into shape. Showing a bit of personality is good, but don't ramble. Also keep in mind that what might seem funny to you while writing your CV might not necessarily look so clever to everyone who reads it. A much better strategy is to let your personality show at the interview after you've established a rapport. If in doubt, always err on the side of plain and simple writing.

Some managers don't care so much about spelling and grammar, but others (like me) care a great deal. I think my distaste stems from years of reviewing poor code that is often full of spelling errors. Using the spelling checker in your word processor or browser doesn't take a lot of extra effort. Don't ignore the red squiggly lines!

Most people find that writing well takes a bit of effort and discipline, so be realistic and allow yourself plenty of time to write and revise.