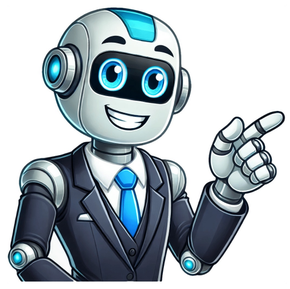
Click to prove you're human



# Elements of programming interviews python

Flipkart Internet Private Limited has published Elements of Programming Interviews in Python: The Insiders Guide. This comprehensive book covers everything needed to ace the next programming interview. The author, who initially thought they were proficient in Python, was pleasantly surprised by the book's ability to teach idiomatic code writing and introduce useful libraries. The book's content is divided into chapters that focus on problems similar to those used in Google interviews, making it a valuable resource for interview preparation. The solutions are provided with detailed explanations, and the authors also offer guidance on which concepts to prioritize during whiteboard interviews. While some readers may find the introduction to certain topics lacking, the book's structure and Github Judge feature are highly praised, providing a comprehensive approach to algorithmic problem-solving and preparing students for coding interviews."Cracking the Coding Interview" by EPI leaves CtCI behind in terms of systematizing learning. While CtCI covers a broader range of topics, EPI focuses more on algorithms and data structures, making it more systematic in its approach. The presentation style is also noteworthy, focusing on relevant topics that provide a clear path to solving problems. The book features multiple levels of problems, allowing learners to progress from easier, generally applicable issues to harder, more specialized ones.The landscape of programming interviews has become a highly competitive arena, driven by the demand for skilled software engineers in the tech industry. To succeed, candidates must be well-prepared to showcase their expertise and problem-solving abilities. Understanding that programming interviews typically involve both technical and non-technical evaluations is crucial. Candidates need to master coding proficiency and demonstrate cultural fit within the organization.The pressure to answer complex algorithmic questions under time constraints can be overwhelming. However, preparation should focus on understanding the underlying principles of algorithm design and optimization, as well as practicing coding problems. Non-technical aspects, such as behavioral competencies, are also essential to assess aligning with core values and team dynamics.To overcome interview anxiety, candidates may benefit from mock interviews, relaxation techniques, and insights from seasoned authors and practitioners. A well-rounded preparation strategy encompassing both technical mastery and interpersonal skills can significantly enhance a candidate's prospects."Elements of Programming Interviews in Python" is a valuable resource aimed at helping aspiring programmers excel in technical interviews. The book provides over 250 programming problems categorized by data structures and algorithms, with detailed solutions that illustrate the correct approach and underlying principles.Looking to master SQL queries, dynamic programming challenges, tree and graph manipulations, and more? This comprehensive guide provides clarity on common pitfalls and variations that interviewers might employ, helping readers build confidence and competence in tackling these coding challenges. With practical advice throughout the text, including effective study habits and problem-solving strategies, readers are encouraged to adopt a systematic approach when faced with coding challenges.Preparing for programming interviews can be daunting, especially in today's competitive job market. However, utilizing resources like "Elements of Programming Interviews in Python" can provide a structured approach to enhance your preparation. To get the most out of this book, it's essential to tailor your study sessions around its comprehensive content, covering both basic and advanced programming concepts. Starting with fundamental topics will help you build a solid foundation in core algorithms and data structures, which is crucial for tackling complex problems with confidence.Regular practice is vital, not only for reinforcing knowledge but also improving problem-solving speed and efficiency essential skills during timed interview scenarios. Dedicate time each week to focus solely on solving problems, starting with easier challenges before progressing to more difficult ones to build momentum and retain information better. Additionally, thoroughly review the explanations and solutions provided in the book to understand different approaches to problem-solving.Effective communication of thought processes is also critical, which can be achieved by understanding various problem-solving methods. Furthermore, having a strategy for negotiations following interviews is crucial, as articulating your value as a candidate significantly impacts the offers you receive.By incorporating consistent practice, a methodical approach to studying content, and understanding your worth as a candidate into your preparation, you'll be well-equipped for both technical and non-technical aspects of interviews.Practical guidance on mastering coding interview skills with a focus on Python is presented in this tutorial. It explores strategies for tackling common programming challenges through real-world examples, providing a comprehensive understanding of fundamental computer science concepts.When dealing with coding interviews, it's often helpful to familiarize yourself with certain built-in functions in Python. However, not every developer has the time or opportunity to do so beforehand. A common scenario presented during such interviews is having a list of elements and needing to iterate over them while accessing both indices and values. One classic example of this is the FizzBuzz problem, where you have to replace integers divisible by 3 with "fizz", those divisible by 5 with "buzz", and numbers divisible by both 3 and 5 with "fizzbuzz". A more elegant solution than using range() for this purpose involves employing enumerate(), which returns a counter (index) and the element value. For cases where starting from index 0 is not preferred, you can use the optional start parameter to set an offset. Some might argue against dropping map() and filter() in favor of list comprehensions due to their readability and functionality similarity, but Python's creator, Guido van Rossum, had valid reasons for wanting to remove them. List comprehensions are often easier to read and understand than their counterparts, making them a better choice in coding interviews, as they demonstrate familiarity with what's most common in Python. When it comes to debugging issues, instead of relying on print() statements, it's more efficient and professional to use a debugger for non-trivial bugs.Will let you develop quickly on the job. If using Python 3.7, can just call breakpoint() at desired location to drop into debugger. Calling breakpoint puts you into pdb, default Python debugger. On older versions, can import pdb explicitly like breakpoint(). Pdb is not as clean and requires more memory.There are other debuggers available but pdb is part of standard library so always available. Try out different debuggers before coding interviews to get used to workflow. Python has many ways to handle string formatting, it can be tricky to know what to use.In fact, we have two articles on string formatting in general and f-strings specifically. In coding interviews, suggested formatting approach is Python's f-strings. F-strings support string formatting mini-language and powerful string interpolation.F-strings allow adding variables or valid python expressions and evaluating them at runtime before adding to string. One risk is if outputting user-generated values, can introduce security risks. Template Strings may be safer option in such cases.Sorting is common in coding interviews and multiple ways to sort items. Unless interviewer wants you to implement own algorithm, use sorted(). Default sorts input in ascending order, reverse keyword argument sorts in descending order.Sorted() has optional key keyword argument that lets specify function called on every element prior to sorting. This allows custom sorting rules especially helpful for complex data types. By passing lambda function that returns each element's age, can easily sort list of dictionaries by single value.In coding interviews, picking right data structure is just as important as algorithms. Picking right data structure can have major impact on performance. Python has powerful and convenient functionality built into its standard data structure implementations which are incredibly useful in coding interviews.When dealing with duplicate elements in a dataset, new developers often resort to lists when sets would be more suitable, as they enforce uniqueness. Let's consider a function get_random_word() that returns a random word from a small set. To obtain 1000 unique words, you'd typically call this function repeatedly and store the results in a data structure containing every unique word. Here are three approaches: two suboptimal ones and one good approach.Bad Approach 1 involves storing values in a list, then converting it to a set. This approach isn't terrible but creates unnecessary lists and sets, which interviewers often notice.Bad Approach 2 avoids the conversion by storing values in a list without using other data structures. It tests for uniqueness by comparing new values with all elements currently in the list. As the number of words grows, the number of lookups increases quadratically, leading to poor time complexity (O(N)).Good Approach involves skipping lists altogether and using sets from the start. This approach allows near-constant-time checks whether a value is in the set or not, unlike lists which require linear-time lookups. The difference in lookup time means that the time complexity for adding to a set grows at a rate of O(N), which is better than the O(N) from the second approach.When using list comprehensions, be mindful of unnecessary memory usage. For instance, if you're asked to find the sum of the first 1000 perfect squares starting with 1, creating a list and then summing its values can lead to performance issues when dealing with larger datasets.Replacing brackets with parentheses can transform list comprehensions into generator expressions, which are ideal for retrieving data from sequences without needing access to all the data simultaneously. Generator expressions return generator objects that track their current state (e.g., i = 49) and calculate the next value only when asked for, allowing them to be used on large datasets due to memory efficiency.When it comes to dictionaries, Python offers elegant functionality to add, modify, or retrieve items with minimal code. The `.get()` method automatically handles key existence and returns either the value or a default, whereas `.setdefault()` sets a new value if the key doesn't exist while returning the current one. However, in some cases, explicitly checking for values is necessary, especially when using `.get()` without providing a default.In coding interviews, knowing how to effectively utilize Python's standard library can significantly enhance skills. This section will focus on useful functions like `.get()` and `.setdefault()`, which provide more efficient ways of working with dictionaries.Looking for an efficient way to store and look up student grades. One approach uses a dictionary, but iterating over each student can be tedious. A cleaner solution utilizes a defaultdict, which allows setting a default value if the key doesn't exist. This reduces code complexity by handling default values at the defaultdict level. string constants offer convenient ways to work with frequently referenced string values, including ascii_letters, digits, and punctuation. Using these constants simplifies code readability and usage. In contrast to real-life scenarios presented in coding interviews, actual projects rarely require generating all possible pairs of values. Instead, the focus is on developing efficient algorithms. The itertools library provides tools like permutations() and combinations(), which can be used to generate iterable sequences of input data. Permutations consider order, while combinations do not. These functions can significantly enhance code during interviews and beyond.Elements of Programming Interviews (EPI) offers a practical approach to computer science fundamentals through real-world programming interview questions. This tutorial is accompanied by a video course created by the Real Python team for deeper understanding. EPI prepares readers for contemporary software interviews, focusing on problems that stem from real-world applications and can be coded in a reasonable time. It emphasizes algorithmic techniques translation into the workplace.Tsung-Hsien Lee, a Staff Software Engineer at Toyota Research Institute, divides his time between teaching and conducting research on applied algorithms at the University of Texas at Austin. He holds multiple degrees, including a PhD from the University of California at Berkeley and an undergraduate degree from the Indian Institute of Technology at Kanpur. Lee's passion lies in designing and implementing algorithms, which he applies to various aspects of his life.Amit Prakash is co-founder and CTO of ThoughtSpot, a Silicon Valley startup. He previously worked as a Member of the Technical Staff at Google, where he focused on machine learning issues related to online advertising. Prior to that, he was part of Microsoft's web search team. Prakash holds a PhD from the University of Texas at Austin and an undergraduate degree from the Indian Institute of Technology at Kanpur.(Note: I used the "WRITE AS A NON-NATIVE ENGLISH SPEAKER (NNES)" method for this text)