99 Programming Skills Interview Questions to Ask

Questions

1. Can you explain what a variable is and how you use it in programming, like explaining it to a child?

2. What are the basic data types you know, and why do we need different types?

3. Explain what a loop is and give an example of when you would use one.

4. What's the difference between '==' and '=' in a programming language you are familiar with?

5. Can you describe what a function is and why functions are useful?

6. What is an array, and how is it useful for storing data?

7. Explain the concept of 'if/else' statements. When would you use them?

8. What is a string in programming, and what are some common operations you can perform on strings?

9. Describe what you know about comments in code and why they are important.

10. What does debugging mean? What are some techniques you use to debug code?

11. Explain the difference between a compiled and an interpreted language.

12. What is object-oriented programming? Can you give a simple example?

13. Describe what version control is and why it is important for collaborative coding.

14. What are some common coding errors you've encountered, and how did you fix them?

15. Have you used any APIs? What was your experience?

16. Can you explain what a class is and how objects are created from it?

17. What is inheritance, and how does it promote code reusability?

18. Explain what polymorphism is with a real-world analogy.

19. Describe what a data structure is, and name a few examples.

20. What is the difference between a stack and a queue?

21. What are some common sorting algorithms, and how do they work?

22. What is a database? Have you worked with any? What types?

23. What's the difference between client-side and server-side programming?

24. Explain the concept of scope in programming.

25. What does it mean for code to be 'readable,' and why is readability important?

26. How would you approach solving a new programming problem you've never seen before?

27. Have you ever contributed to an open-source project? Describe your experience.

28. Explain what a linked list is.

29. What is recursion? Can you explain with an example?

30. Explain the difference between processes and threads, and when would you choose one over the other?

31. Describe the concept of recursion. Can you provide an example of a problem that is best solved using recursion?

32. What are design patterns, and why are they useful in software development? Give examples.

33. Explain the SOLID principles of object-oriented design. How do these principles contribute to maintainable code?

34. Describe the difference between SQL and NoSQL databases. What are the trade-offs of each?

35. What is the purpose of version control systems like Git? Explain the common Git workflow.

36. Explain the concept of caching. What are different caching strategies, and when would you use each?

37. Describe how you would approach debugging a complex software issue. What tools or techniques would you use?

38. What are unit tests? Why are they important, and how do you write effective unit tests?

39. Explain the concept of API (Application Programming Interface). How do APIs enable communication between different systems?

40. What is Big O notation, and how is it used to analyze the performance of algorithms?

41. Describe common data structures like arrays, linked lists, trees, and graphs. What are their respective use cases?

42. Explain the concept of concurrency and parallelism. How can you achieve concurrency in your programming language of choice?

43. What are the benefits of using a framework (e.g., React, Angular, Django, Spring)? What are potential drawbacks?

44. Describe the concept of dependency injection. How does it improve code testability and maintainability?

45. Explain the difference between authentication and authorization. How are they typically implemented in web applications?

46. What are some common security vulnerabilities in web applications (e.g., XSS, SQL injection)? How can you prevent them?

47. Describe the concept of code refactoring. When and why should you refactor code?

48. Explain the concept of microservices architecture. What are the advantages and disadvantages compared to a monolithic architecture?

49. How would you design a simple RESTful API? What considerations would you take into account?

50. Explain the concept of dependency injection and its benefits.

51. How would you implement a thread-safe singleton pattern?

52. Describe the differences between optimistic and pessimistic locking.

53. What are the advantages and disadvantages of microservices architecture?

54. Explain the concept of event sourcing.

55. How would you design a rate limiter?

56. Describe the CAP theorem and its implications.

57. What are the trade-offs between strong and eventual consistency?

58. Explain the concept of CQRS (Command Query Responsibility Segregation).

59. How would you implement a distributed cache?

60. Describe the different types of NoSQL databases and their use cases.

61. What are the advantages and disadvantages of using a message queue?

62. Explain the concept of idempotency in API design.

63. How would you handle transactions in a distributed system?

64. Describe the different types of design patterns (e.g., creational, structural, behavioral).

65. Explain the concept of domain-driven design (DDD).

66. How would you optimize a slow-performing database query?

67. Describe the different types of caching strategies (e.g., write-through, write-back).

68. What are the security considerations when designing a web application?

69. Explain the concept of OAuth 2.0.

70. How would you implement a secure authentication and authorization system?

71. Describe the different types of testing (e.g., unit, integration, end-to-end).

72. How would you implement continuous integration and continuous delivery (CI/CD)?

73. Explain the concept of infrastructure as code (IaC).

74. How would you monitor and debug a production application?

75. Describe the different types of logging strategies.

76. What are the key considerations when scaling an application?

77. Explain the concept of containerization (e.g., Docker).

78. How would you manage configuration in a distributed system?

79. How do you optimize code for both speed and memory usage, especially when dealing with large datasets?

80. Explain the concept of 'bytecode' and its role in programming language execution.

81. Describe a time you had to debug a complex memory leak. What tools and techniques did you use?

82. Design a system for handling a high volume of concurrent requests. Discuss trade-offs between different architectural patterns.

83. How would you implement a custom garbage collector? What are the challenges?

84. Explain the CAP theorem and how it applies to distributed systems. Give examples.

85. Describe the differences between symmetric and asymmetric encryption. When would you use each?

86. How do you prevent race conditions in a multi-threaded environment? Explain with example.

87. Explain the concept of reflection in programming. What are its use cases and drawbacks?

88. How do you approach testing code that interacts heavily with external APIs or services?

89. Describe the design patterns you find most useful in your work and why.

90. How would you design a real-time recommendation system?

91. What are some advanced techniques for optimizing database queries?

92. Explain the concept of 'code injection' and how to prevent it.

93. How would you implement a fault-tolerant system?

94. Describe the differences between microservices and a monolithic architecture.

95. Explain the concept of 'zero-downtime deployment'. How do you achieve it?

96. How do you stay up-to-date with the latest trends and technologies in programming?

97. Describe a time you had to learn a new programming language or framework quickly. What was your strategy?