🔥 If You Can Answer These Questions, You Won't Fail Any Python Interview!

🧱 BASIC: Laying the Foundation

1. What is Python, and what are its key features?
2. What are Python's built-in data types?
3. How do you declare and use variables in Python?
4. What is the difference between a list, tuple, and set?
5. How do you iterate over a list in Python?
6. What are Python's conditional statements, and how are they used?
7. How does Python handle memory management?
8. What is the use of the `len()` function?
9. What is the difference between `is` and `==` in Python?
10. How do you handle exceptions in Python?
11. What are Python functions, and how do you define them?
12. What is the difference between `args` and `*kwargs`?
13. How is Python's `for` loop different from other programming languages?
14. What is the purpose of the `range()` function?
15. How do you import and use modules in Python?
16. What are Python decorators, and how do they work?
17. How do you reverse a string in Python?
18. How do you check if an element exists in a list?
19. What is a lambda function? Provide an example.

🧩 INTERMEDIATE: Keep Practicing

1. What is the difference between shallow copy and deep copy in Python?
2. What are Python comprehensions, and how are they used?
3. How does Python's garbage collection work?
4. What is Python's Global Interpreter Lock (GIL)?
5. What is the difference between mutable and immutable objects?
6. How do you use Python's `zip()` function?
7. What is the difference between `@staticmethod` and `@classmethod`?
8. How do you merge two dictionaries in Python?
9. What is the difference between `sort()` and `sorted()`?
10. How do you handle file operations in Python?
11. What are Python's iterators and generators?
12. How do you use the `with` statement in Python?
13. What is Python's `itertools` module, and when would you use it?
14. What is the difference between positional and keyword arguments?
15. How do you perform matrix operations in Python?
16. What are Python's metaclasses, and how are they used?
17. How do you perform unit testing in Python?
18. How is Python's `os` module used?
19. What are Python's `argsort()` and `argmax()` functions?
20. How do you optimize code performance in Python?

🚀 ADVANCED: Taking Your Skills to the Next Level

1. How does Python's multiprocessing differ from threading?
2. How do you implement a custom Python metaclass?
3. How do you implement memoization in Python?
4. What is Python's `asyncio`, and how does it handle concurrency?
5. How do you profile Python code to identify performance bottlenecks?
6. How do you handle circular imports in Python projects?
7. What are Python’s weak references, and when would you use them?
8. How do you implement a binary search algorithm in Python?
9. How does Python's `dataclasses` module work, and when should you use it?
10. What are Python's context managers, and how do you create a custom one?