Python Mastery Self-Assessment Questions

Python Fundamentals & Core Concepts

1. What are Python's main data types? (Give examples.)
2. Difference between list, tuple, set, and dict.
3. Why are Python strings immutable?
4. What is the difference between is and ==?
5. What's the difference between shallow copy and deep copy? (copy.copy vs copy.deepcopy)
6. How does Python handle memory management and garbage collection?
7. What are Python's built-in data structures?
8. Explain slicing in Python with an example.
9. What is the difference between append() and extend() for lists?
10. How do you merge two dictionaries in Python (multiple ways)?
11. What's the difference between str() and repr()?
12. Why do Python functions return None by default if no return statement is given?
13. Explain Python's range() function and how it differs from xrange() in Python 2.
14. What happens if you use a mutable object as a dictionary key?
15. Why is Python considered dynamically typed?
16. How does Python handle variable scope? (Explain LEGB rule: Local, Enclosing, Global, Built-in)
17. What's the difference between global and nonlocal?
18. Explain what a Python __doc__ string is.
19. How do you swap two variables in Python without using a temporary variable?
20. What's the difference between del, remove(), pop(), and clear() for collections?

Functions & Modules

21. Explain the difference between positional, keyword, default, and variable-length arguments.
22. What are *args and **kwargs used for?
23. What's the difference between a function and a method?
24. Why are default arguments in Python evaluated only once?
25. Explain higher-order functions with an example.
26. What's a closure in Python?
27. What is recursion? Show an example.

28. Explain what decorators are and when to use them.
29. How do you implement a custom decorator that accepts arguments?
30. What's the difference between a Python script and a Python module?
31. How do relative imports differ from absolute imports in Python?
32. What's the difference between import X, from X import Y, and import X as Z?
33. What happens if you import a module multiple times in Python?
34. What is __name__ == '__main__' used for?
35. How do Python namespaces work when importing modules?

Object-Oriented Programming

36. Difference between a class and an object.
37. What are __init__, __new__, and __del__?
38. What's the difference between instance variables and class variables?
39. Explain method resolution order (MRO) in multiple inheritance.
40. How does Python handle method overriding?
41. What is polymorphism?
42. Explain encapsulation in Python.
43. What are abstract base classes (ABC) in Python?
44. Difference between @staticmethod and @classmethod.
45. How do you make a class iterable?
46. How do you implement operator overloading in Python? (Example: overloading + operator.)
47. What are mixins in Python?
48. Explain duck typing in Python.
49. What is the difference between __str__ and __repr__?
50. How do dataclasses differ from regular classes?

Advanced Python Features

51. What are Python generators and how do they differ from regular functions?
52. Explain the yield keyword.
53. Difference between generator expressions and list comprehensions.
54. What are coroutines in Python?
55. Explain async/await in Python.
56. How is concurrency different from parallelism in Python?

57. What is the Global Interpreter Lock (GIL)?
58. How does multiprocessing differ from threading?
59. What is asyncio and when would you use it?
60. What is memoization and how can you implement it in Python?
61. Explain functools.lru_cache.
62. What is monkey patching? Is it good practice?
63. Explain metaclasses in Python.
64. How do you dynamically create classes at runtime?
65. What's the difference between __slots__ and normal attributes in classes?
66. How do you implement a singleton pattern in Python?
67. What are descriptors in Python?
68. What is the difference between deepcopy and pickling?
69. What are weak references and why use them?
70. How does Python's garbage collector handle circular references?

Files, I/O, and Exceptions

71. Difference between text mode and binary mode in file handling.
72. How do you safely read a file in Python? (with open)
73. Explain context managers (with keyword).
74. How do you create a custom context manager?
75. What are Python's built-in exception classes?
76. Difference between try/except/finally and try/except/else.
77. How do you raise custom exceptions?
78. How does Python's traceback system work?
79. What's the difference between os and pathlib for file handling?
80. How do you check if a file exists in Python?

Data Science / Libraries / Ecosystem

81. Difference between NumPy arrays and Python lists.
82. Why is NumPy faster than lists?
83. Explain broadcasting in NumPy.
84. What's the difference between pandas.DataFrame and pandas.Series?
85. How does groupby work in pandas?

86. What's the difference between apply(), map(), and applymap() in pandas?

87. Explain the difference between shallow and deep copies in pandas.

88. What's the difference between .loc[] and .iloc[]?

89. How do you handle missing data in pandas?

90. What's the role of vectorization in pandas and NumPy?

Python & Software Engineering Practices

91. What is PEP 8 and why is it important?

92. Explain type hints in Python.

93. Difference between static typing and dynamic typing.

94. What's the difference between unit tests, integration tests, and functional tests?

95. How do you use unittest or pytest in Python?

96. What are Python virtual environments and why use them?

97. Difference between venv, virtualenv, and conda.

98. What's the difference between a Python package and a module?

99. What is dependency injection and why is it useful in Python?

100. How do you profile Python code for performance bottlenecks?

101. What tools can you use for debugging Python applications? (pdb, ipdb, breakpoint())

102. How do you write Python code that is memory-efficient?

System-Level & Expert Questions

103. How does Python's memory allocation differ for small integers vs large objects?

104. How does CPython store dictionary keys internally?

105. Explain hash collisions and how Python handles them.

106. What's the difference between list.sort() and sorted()?

107. How does Python implement list slicing internally (time complexity)?

108. What's the complexity of searching in a set vs a list?

109. What's the difference between shallow copies in lists and assignment =?

110. How does Python handle immutability for tuples with mutable objects inside?

111. What's the difference between Python's CPython, PyPy, and Jython?

112. What happens when you run python -O?

113. How do Python decorators work under the hood?

114. Explain how Python bytecode is executed by the interpreter.

115. How does Python optimize string interning?

116. How do __hash__ and __eq__ affect dictionary/set behavior?

117. How does Python handle exceptions at the bytecode level?

118. How does Python's garbage collector detect unreachable cycles?

119. What is the difference between weakref.WeakValueDictionary and a normal dict?

120. What are some limitations of Python due to the GIL, and how do you work around them?