

# PYTHON PROGRAMMING

## BASIC LEVEL

1. What are mutable and immutable data types in Python? Give examples.
2. What is the difference between List and Tuple?
3. What is the difference between Dictionary and Set?
4. What is the difference between is and == operator?
5. How does pass by value and pass by reference work in Python?
6. Explain the difference between shallow copy and deep copy.
7. When and how do you use \*args and \*\*kwargs?
8. What is list comprehension? Give an example.
9. Give an example of dictionary comprehension.
10. What is the range() function explain its operation ?
11. How can you reverse a string? Show 3 methods.
12. What are the methods to check for a substring in a string?
13. Explain the use of split(), join(), and strip() functions.
14. What are the different ways of string formatting? (%, .format(), f-strings)
15. What does it mean that strings are immutable?
16. What are the different methods to add/remove elements in a list?
17. How do you check if a key exists in a dictionary?
18. What are the different ways to sort a list?
19. What is the difference between append() and extend()?
20. What is the difference between remove(), pop(), and del?
21. When do you use break, continue, and pass statements?
22. How can you write if-elif-else as a ternary operator?
23. What is the difference between for loop and while loop?
24. What is the performance impact of nested loops?
25. What is the use of the enumerate() function?
26. How do you define default parameters in a function?

27. What is a lambda function? Give an example.
28. Explain map(), filter(), and reduce() functions with examples.
29. What is a recursive function? Give an example for factorial.
30. What is the difference between global and local variables?

## INTERMEDIATE LEVEL

31. How can you convert a list to a dictionary?
32. How do you merge two lists to create a dictionary?
33. How can you sort a dictionary by keys or values?
34. How do you access a nested dictionary?
35. What is the use of defaultdict?
36. When do you use the Counter class?
37. Explain set operations (union, intersection, difference).
38. What is tuple unpacking?
39. Explain the use of zip() function with an example.
40. What are the methods to remove duplicates from a list?
41. What are the different modes for reading/writing files?
42. Why do we use the with statement in file handling?
43. How do you read/write CSV files?
44. How do you handle JSON files in Python?
45. How can you read a large file efficiently?
46. How does the try-except block work?
47. What is the use of the finally block?
48. How can you handle multiple exceptions together?
49. How do you create a custom exception?
50. What is the use of the raise statement?
51. What is the difference between Class and Object?
52. What is the use of the \_\_init\_\_ method?
53. What is the difference between instance variable and class variable?

54. Give an example of inheritance.
55. What is method overriding?
56. What is the use of the self keyword?
57. What is the difference between `__str__` and `__repr__`?
58. Explain encapsulation, inheritance, and polymorphism.
59. What is the difference between class method and static method?
60. What is the use of the `@property` decorator?
61. What is a generator? Explain the `yield` keyword.
62. What is the difference between iterator and iterable?
63. What is a decorator? Give a simple example.
64. What is a context manager?
65. What does `__name__ == "__main__"` mean?

## **ADVANCED LEVEL**

66. What is the difference between NumPy array and Python list?
67. What are the different methods to create a NumPy array?
68. How do you perform array slicing and indexing?
69. What is broadcasting in NumPy?
70. Explain the use of `np.array()`, `np.zeros()`, `np.ones()`, `np.arange()`.
71. How do you reshape an array?
72. What is the use of `np.where()` function?
73. How do you handle missing values (`NaN`) in an array?
74. What are the methods to concatenate two arrays?
75. What are universal functions (`ufuncs`)?
76. What is the difference between Pandas DataFrame and Series?
77. How do you load a CSV file in Pandas?
78. How do you view the first 5 rows of a DataFrame?
79. How do you handle missing values? (`fillna()`, `dropna()`)
80. How do you select and filter columns?

81. What is the use of the `groupby()` function?
82. How do you merge/join two DataFrames?
83. What is the difference between `apply()`, `map()`, and `applymap()`?
84. How do you sort a DataFrame?
85. How do you create a pivot table?

## DATA SCIENCE SPECIFIC

86. How do you remove duplicate rows from a DataFrame?
87. How do you change the data type of a column?
88. How do you perform operations on string columns?
89. How do you parse and manipulate DateTime columns?
90. How do you convert categorical data to numerical?
91. How do you create a simple line plot in Matplotlib?
92. How do you create multiple subplots?
93. How is Seaborn different from Matplotlib?
94. How do you visualize a correlation matrix as a heatmap?
95. What is the difference between bar chart and histogram?
96. Why is list comprehension faster than a loop?
97. What are the tips for efficiently processing large datasets?
98. What practices should be followed for memory optimization?
99. How do you debug code in Python? (`pdb` module)
100. What steps should be taken to make code production-ready?

## PYTHON CORE CONCEPTS

101. How does memory management work in Python?
102. What is garbage collection? How does it work?
103. What is reference counting in Python?
104. What is the use of the `id()` function?
105. Why is everything an object in Python?

106. What is small integer caching? (-5 to 256)
107. What is string interning?
108. What is the use of `sys.getsizeof()` function?
109. What is a memory leak and how do you avoid it in Python?
110. What is the use of `__slots__` in a class?
111. How do you count character frequency in a string?
112. How do you replace multiple spaces with a single space?
113. What are the different methods to capitalize a string?
114. Explain use cases for `startswith()` and `endswith()`.
115. How do you use regular expressions (regex) in Python?
116. What is the difference between `re.match()`, `re.search()`, and `re.findall()`?
117. How do you convert a string to bytes and bytes to string?
118. How do you handle Unicode strings?
119. How do you create multi-line strings?
120. When do you use raw strings (`r'''`)?
121. How do you flatten a list? (nested list to single list)
122. How do you find the nth largest element in a list?
123. How do you merge two sorted lists?
124. How do you remove consecutive duplicates from a list?
125. How do you divide a list into chunks?

1. Give an example of dictionary comprehension with a condition.
2. How do you flatten a nested dictionary?
3. How do you find a key from a value in a dictionary?
4. What are the methods to merge multiple dictionaries?
5. What is the advantage of the get() method over the [] operator?
6. How do you invert a dictionary (keys->values, values->keys)?
7. What is the use of ChainMap?
8. How do you sort a dictionary in different ways?
9. What is the difference between OrderedDict and normal dict?
10. How do you use if-else in dictionary comprehension?
11. What is a frozen set? How is it different from a normal set?
12. How do you find the symmetric difference of two sets?
13. What are all the methods to add/remove elements in a set?
14. What is the benefit of creating a set from a list?
15. Explain issubset(), issuperset(), and isdisjoint().
16. What is a named tuple? How do you create it?
17. Is there any trick to modify an element in a tuple?
18. Explain tuple unpacking with \* operator.
19. How do you create a single element tuple?
20. How do you sort a tuple?
21. What is deque? How is it better than a list?
22. What is a practical use case of namedtuple?
23. What is the use case of OrderedDict in Python 3.7+?
24. Give a real-world example of ChainMap.
25. What is the use of UserDict, UserList, and UserString?

## FUNCTIONAL PROGRAMMING

1. What is a closure in Python? Give an example.
2. What does first-class functions mean?
3. What are higher-order functions?
4. What are partial functions? Explain functools.partial.
5. What is the difference between @staticmethod and @classmethod decorators?
6. How do you apply multiple decorators to a function?
7. How do you create a decorator with arguments?
8. What is the use of the @wraps decorator?
9. What are function annotations (type hints)?
10. What is the difference between \* and / parameters (Python 3.8+)?
11. How do you create a custom iterator? Explain \_\_iter\_\_ and \_\_next\_\_.
12. What is the difference between generator expression and list comprehension?
13. What is the use of the yield from statement?
14. How do you convert a generator to a list?
15. How do you create an infinite generator?
16. What are the important functions in the itertools module?
17. What is the use of itertools.chain()
18. Explain itertools.cycle() and itertools.repeat().

19. What is the difference between `itertools.combinations()` and `itertools.permutations()`?
20. What are the memory benefits of generators?

## OBJECT-ORIENTED ADVANCED

### Magic Methods (Dunder Methods)

1. What is the difference between `__new__` and `__init__`?
2. What is the use of the `__call__` method?
3. What is the use of `__len__`, `__getitem__`, and `__setitem__`?
4. Explain operator overloading methods like `__add__`, `__sub__`.
5. How are `__enter__` and `__exit__` methods used in context managers?
6. When is the `__del__` destructor method called?
7. What is the use of the `__hash__` method?
8. Explain comparison methods `__eq__`, `__lt__`, `__gt__`.
9. What is the difference between `__dict__` and `__slots__`?
10. What is the use of `__name__`, `__doc__`, and `__module__` attributes?
11. What is MRO (Method Resolution Order) in multiple inheritance?
12. What is the use of the `super()` function?
13. What is an abstract base class? Explain the ABC module.
14. What is the use of the `@abstractmethod` decorator?
15. What is duck typing in Python?

## ADVANCED TOPICS

1. What is the difference between multithreading and multiprocessing?
2. What is GIL (Global Interpreter Lock)?
3. How do you use the threading module basically?
4. When should you use the multiprocessing module?
5. What is asyncio? Explain `async/await`.
6. What is the use of the `assert` statement?
7. What is exception chaining (`raise ... from`)?
8. How do you design a custom exception hierarchy?
9. What is the use of the `warnings` module?
10. How do you create a context manager using `contextlib`?
11. What is the use of the `__init__.py` file?
12. What is the difference between relative and absolute imports?
13. What is a virtual environment? Why do we use it?
14. What is the difference between pip and conda?
15. How do you generate and use a `requirements.txt` file?