Python - OOPS - Interview Questions-Answers

Top 50 Python Interview Questions (Beginner → Intermediate)

1. What is Python?

Python is a high-level, interpreted, and dynamically typed programming language known for its readability and simplicity.

2. What are Python's key features?

Easy syntax, interpreted, dynamically typed, object-oriented, portable, and has vast libraries.

3. What is PEP 8?

PEP 8 is a style guide for writing clean, readable, and consistent Python code.

4. What are Python data types?

int, float, str, list, tuple, dict, set, bool, complex.

5. What is the difference between mutable and immutable types?

Mutable objects (list, dict, set) can change after creation; immutable ones (tuple, str, int) cannot.

6. What is the difference between list and tuple?

Lists are mutable; tuples are immutable.

7. What is a dictionary in Python?

A key-value data structure that allows fast lookups via keys.

8. What are sets in Python?

Unordered collections of unique elements.

9. What are Python namespaces?

Namespaces store variable names and their corresponding objects (e.g., local, global, built-in).

10. What are Python modules and packages?

Modules are .py files; packages are directories with `__init__.py` that group modules.

11. How do you import a module?

`import module_name` or `from module_name import function_name` .

12. What is `__init__.py` ?

It marks a directory as a Python package and can execute initialization code.

13. What is the difference between `is` and `==` ?

`is` checks object identity; `==` checks value equality.

14. What is a function in Python?

A reusable block of code defined using `def` that performs a specific task.

15. What are *args and **kwargs?

`*args` handles variable numbers of positional arguments; `**kwargs` handles variable keyword arguments.

16. What is lambda in Python?

An anonymous, single-line function defined with `lambda` keyword.

17. What are decorators?

Functions that modify the behavior of other functions without changing their code.

18. What are generators?

Functions that yield values one at a time using `yield` , saving memory.

19. What is the difference between iterator and iterable?

Iterables can be looped; iterators produce elements one by one using `next()` .

20. What are list comprehensions?

Concise syntax for creating lists: `[x**2 for x in range(5)]` .

21. How to handle exceptions?

Using `try` , `except` , `else` , and `finally` blocks.

22. What is the purpose of `with` statement?

Simplifies resource management (e.g., file handling) by automatically closing resources.

23. What is Python’s GIL?

Global Interpreter Lock ensures only one thread executes Python bytecode at a time.

24. Difference between shallow copy and deep copy?

Shallow copy copies references; deep copy copies objects recursively.

25. How to manage memory in Python?

Python uses automatic garbage collection to reclaim unused memory.

26. What is the difference between `@staticmethod` and `@classmethod`?

`@staticmethod` doesn't access class or instance; `@classmethod` accesses class but not instance data.

27. What are docstrings?

String literals used to document modules, functions, classes, or methods.

28. What is `__str__` vs `__repr__`?

`__str__` is for user-friendly output; `__repr__` is for developer/debug output.

29. What is inheritance in Python?

Mechanism to derive a new class (child) from an existing one (parent).

30. What is multiple inheritance?

A class can inherit from more than one parent class.

31. What is polymorphism?

Same interface, different implementation (e.g., method overriding).

32. What is encapsulation?

Restricting access to class attributes using private/protected variables.

33. What are Python's built-in data structures?

List, Tuple, Set, Dictionary.

34. What is slicing in Python?

Extracting a portion of a sequence using `[start:end:step]`.

35. How is memory managed in Python?

Through reference counting and a cyclic garbage collector.

36. Difference between `remove()`, `pop()`, and `del`?

`remove()` deletes by value, `pop()` by index, `del` deletes object or index.

37. What are Python's built-in functions?

Examples: `len()`, `type()`, `id()`, `sum()`, `min()`, `max()`, `sorted()`.

38. What are global, local, and nonlocal keywords?

They define variable scope—global (module), local (function), nonlocal (outer enclosing function).

39. What is recursion?

A function calling itself to solve smaller subproblems.

40. What is a virtual environment?

An isolated Python environment for managing dependencies.

41. What are Python's file modes?

'r' (read), 'w' (write), 'a' (append), 'b' (binary), '+' (read/write).

42. How to read a file line by line?

Using `for line in open('file.txt'):` or `with open(...) as f:` loop.

43. How to merge two dictionaries?

`dict1.update(dict2)` or `{**dict1, **dict2}`.

44. What is `zip()` used for?

Combines multiple iterables into tuples element-wise.

45. What is `enumerate()` used for?

Adds index to iterable items during iteration.

46. What is `map()`, `filter()`, and `reduce()`?

`map()` applies a function, `filter()` filters elements, `reduce()` accumulates values.

47. What is difference between deep and shallow copy?

Deep copy copies nested objects; shallow only top-level references.

48. How to reverse a list?

Using `list.reverse()` or `list[::-1]`.

49. What are Python modules for data science?

NumPy, Pandas, Matplotlib, Scikit-learn, TensorFlow, PyTorch.

50. What are Python best practices?

Follow PEP8, use virtual environments, write modular code, handle exceptions, and test

51. `append()` vs `extend()`

`append()` adds a single element to the list; `extend()` adds elements from another iterable.

52. `copy()` vs `deepcopy()`

`copy()` makes a shallow copy (references nested objects); `deepcopy()` duplicates all nested data.

53. `sort()` vs `sorted()`

`sort()` modifies the list in place; `sorted()` returns a new sorted list.

54. `count()` vs `len()`

`count(x)` returns how many times `x` appears; `len()` returns total number of items.

55. `any()` vs `all()`

`any()` returns True if at least one element is True; `all()` returns True if all elements are True.

56. `get()` vs direct key access in dict

`dict.get(key)` returns `None` (or default) if key missing; `dict[key]` raises `KeyError`.

57. `isnumeric()` vs `isdigit()` vs `isdecimal()`

All check for numbers — `isdecimal()` for base-10 digits, `isdigit()` for digits, `isnumeric()` for numeric chars like fractions.

58. `startswith()` vs `find()`

`startswith()` checks if string begins with substring; `find()` gives position of substring anywhere.

59. `replace()` vs `translate()`

`replace()` substitutes substrings; `translate()` replaces multiple characters using a mapping table.

60. split() vs rsplit()

`split()` breaks from left; `rsplit()` breaks from right (useful with `maxsplit`).

61. isinstance() vs type()

`isinstance(obj, cls)` checks inheritance; `type(obj)` checks exact class.

62. input() vs sys.stdin.readline()

`input()` strips newline automatically; `sys.stdin.readline()` keeps newline character.

63. eval() vs exec()

`eval()` runs single expressions and returns result; `exec()` executes code blocks and returns None.

64. enumerate() vs range(len())

`enumerate()` gives both index and value directly; `range(len())` gives only indices.

65. tuple() vs list()

`tuple()` creates an immutable sequence; `list()` creates a mutable one.

66. filter() vs list comprehension

Both filter items; `filter()` uses a function, list comprehension is more readable and flexible.

67. map() vs list comprehension

Both transform items; `map()` returns iterator using a function, list comprehension allows inline expressions.

68. dir() vs help()

`dir()` lists attributes and methods; `help()` gives detailed documentation.

69. del vs clear()

`del` deletes variable or item; `clear()` empties container but keeps it existing.

70. @property vs normal method

71. Write a program to find if a number is even or odd.

```
n = int(input("Enter number: "))
if n % 2 == 0:
    print("Even")
else:
    print("Odd")
```

72. Find the largest of three numbers.

```
a, b, c = map(int, input("Enter three numbers: ").split())
print("Largest:", max(a, b, c))
```

73. Check if a number is prime.

```
n = int(input("Enter number: "))
if n > 1:
    for i in range(2, int(n**0.5)+1):
        if n % i == 0:
            print("Not Prime")
            break
    else:
        print("Prime")
```

74. Reverse a string.

```
s = input("Enter string: ")
print("Reversed:", s[::-1])
```

75. Check if a string is palindrome.

```
s = input("Enter string: ")
print("Palindrome" if s == s[::-1] else "Not Palindrome")
```

76. Count vowels in a string.

```
s = input("Enter string: ").lower()
count = sum(1 for ch in s if ch in 'aeiou')
print("Vowels:", count)
```

77. Find factorial of a number.

```
n = int(input("Enter number: "))
fact = 1
for i in range(1, n+1):
    fact *= i
print("Factorial:", fact)
```

78. Print Fibonacci sequence up to n terms.

```
n = int(input("Enter terms: "))
a, b = 0, 1
for _ in range(n):
    print(a, end=" ")
    a, b = b, a+b
```

79. Find sum of digits of a number.

```
n = input("Enter number: ")
print("Sum:", sum(int(d) for d in n))
```

80. Find the largest element in a list.

```
lst = list(map(int, input("Enter list: ").split()))
print("Largest:", max(lst))
```

81. Find second largest element in list.

```
lst = sorted(set(map(int, input("Enter numbers: ").split())))
print("Second Largest:", lst[-2])
```

82. Count frequency of elements in list.

```
lst = input("Enter items: ").split()
freq = {i: lst.count(i) for i in set(lst)}
print(freq)
```

83. Remove duplicates from list.

```
lst = input("Enter items: ").split()
print("Unique:", list(set(lst)))
```

84. Check if element exists in list.

```
lst = input("Enter list: ").split()
x = input("Enter element: ")
print("Found" if x in lst else "Not Found")
```

85. Swap two numbers.

```
a, b = map(int, input("Enter two numbers: ").split())
a, b = b, a
print("After Swap:", a, b)
```

86. Find minimum and maximum in list.

```
lst = list(map(int, input("Enter list: ").split()))
print("Min:", min(lst), "Max:", max(lst))
```

87. Find sum of all elements in list.

```
lst = list(map(int, input("Enter list: ").split()))
print("Sum:", sum(lst))
```

88. Check if number is Armstrong.

```
n = input("Enter number: ")
s = sum(int(d)**len(n) for d in n)
print("Armstrong" if s == int(n) else "Not Armstrong")
```

89. Count uppercase, lowercase, digits in a string.

```
s = input("Enter string: ")
u = sum(c.isupper() for c in s)
l = sum(c.islower() for c in s)
d = sum(c.isdigit() for c in s)
print("Upper:", u, "Lower:", l, "Digits:", d)
```

90. Reverse words in a sentence.

```
s = input("Enter sentence: ")
print(" ".join(s.split()[::-1]))
```

91. Find common elements between two lists.

```
a = input("List A: ").split()
b = input("List B: ").split()
print("Common:", list(set(a) & set(b)))
```

92. Count words in a string.

```
s = input("Enter sentence: ")
print("Word count:", len(s.split()))
```

93. Check if string is anagram.

```
a, b = input("Enter two strings: ").split()
print("Anagram" if sorted(a) == sorted(b) else "Not Anagram")
```

94. Print multiplication table.

```
n = int(input("Enter number: "))
for i in range(1, 11):
    print(f"{n} x {i} = {n*i}")
```

95. Convert Celsius to Fahrenheit.

```
c = float(input("Enter Celsius: "))
print("Fahrenheit:", (c * 9/5) + 32)
```

96. Find largest word in a sentence.

```
s = input("Enter sentence: ").split()
print("Longest word:", max(s, key=len))
```

97. Find numbers divisible by 3 and 5 in a range.

```
n = int(input("Enter limit: "))
print([i for i in range(1, n+1) if i%3==0 and i%5==0])
```

98. Count occurrences of a character.

```
s = input("Enter string: ")
ch = input("Enter character: ")
print("Count:", s.count(ch))
```

99. Merge two dictionaries.

```
d1 = {'a':1, 'b':2}
d2 = {'c':3, 'd':4}
d1.update(d2)
print(d1)
```

100. Find index of element in list.

```
lst = input("Enter list: ").split()
```