1. Python Basics (Most Asked)

Q1. Why is Python called an interpreted language?

Python code is executed line-by-line by the Python interpreter. Internally it compiles source code into bytecode (`.pyc`) which is executed by the Python Virtual Machine (PVM).

Q2. Difference between list and tuple

Feature	List	Tuple
Mutability	Mutable	Immutable
Speed	Slower	Faster
Syntax	`[]`	`()`
Use-case	dynamic data	fixed data

Q3. What are Python's key features?

- Easy syntax
- Large standard library
- Cross-platform
- Supports OOP + Functional programming
- Dynamic typing
- Huge ecosystem for ML, web, automation

---

2. Data Types + Memory Concepts

Q4. Mutable vs Immutable types

Immutable: int, float, str, tuple
Mutable: list, dict, set

Example

```
a = 10 b = a a = 20 # b still 10 because integers are immutable
```

Q5. Shallow copy vs Deep copy

- **Shallow copy:** copies object reference (nested objects shared)
- **Deep copy:** recursively copies everything

```
import copy a = [[1,2],[3,4]] b = copy.copy(a)       # shallow c = copy.deepcopy(a)   # deep
```

---

3. Functions & Arguments

Q6. What is *args and **kwargs?

- `*args`: variable positional arguments

- **kwargs: variable keyword arguments

```
def func(*args, **kwargs): print(args) print(kwargs)
```

Q7. Default argument gotcha

Default arguments are evaluated once at function definition time.

❌ Wrong:

```
def add(x, arr=[]): arr.append(x) return arr
```

✅ Correct:

```
def add(x, arr=None): if arr is None: arr = [] arr.append(x) return arr
```

---

4. OOP in Python (Very Important)

Q8. What is `self`?

`self` refers to the current object instance. It is passed automatically when calling methods.

Q9. Class method vs Static method

```
class A:    @classmethod def cm(cls): return cls    @staticmethod def sm(): return "static"
```

- `@classmethod` uses `cls` and can modify class state
- `@staticmethod` is utility function; no access to class/instance state

Q10. Inheritance types

- Single
- Multiple
- Multilevel
- Hierarchical
- Hybrid

---

5. Exception Handling

Q11. Try-Except-Else-Finally

```
try: x = 10/2 except ZeroDivisionError: print("error") else: print("success") finally: print("always runs")
```

---

6. Python Collections (High Frequency)

Q12. Dictionary internal working

Python dict is implemented using **hash table**.

- Keys must be hashable (immutable)
- Average lookup: **O(1)**

Q13. set vs list

- set removes duplicates
- set membership is faster O(1)

---

7. Comprehensions

List comprehension

```
squares = [x*x for x in range(10)]
```

Dict comprehension

```
d = {x: x*x for x in range(5)}
```

---

8. Iterators & Generators (Asked in Product Companies)

Q14. Iterator vs Generator

- Iterator: implements `__iter__()` and `__next__()`
- Generator: uses `yield`, lazy evaluation

```
def gen(): for i in range(3): yield i
```

Benefits:

- Memory efficient
- Works for large datasets

9. Decorators (Very Important)

Q15. What is a decorator?

A function that modifies another function without changing its code.

`def deco(fn): def wrapper(): print("before") fn() print("after") return wrapper @deco def hello(): print("hi")`

---

10. Multithreading vs Multiprocessing

Q16. What is GIL?

Global Interpreter Lock allows only one thread to execute Python bytecode at a time.

So:

- Threading helps in **I/O bound tasks**
- Multiprocessing helps in **CPU bound tasks**

---

11. Common Coding Interview Patterns (Must Know)

Pattern 1: Two Pointers

Used in:

- sorted arrays
- pair sum

```
def pair_sum(arr, target): i, j = 0, len(arr)-1 while i < j: s = arr[i] + arr[j] if s == target: return True elif s < target: i += 1 else: j -= 1 return False
```

Pattern 2: Sliding Window

Used in:

- max sum subarray
- longest substring

```
def max_sum_k(arr, k): window = sum(arr[:k]) ans = window for i in range(k, len(arr)): window += arr[i] - arr[i-k] ans = max(ans, window) return ans
```

Pattern 3: HashMap frequency

Used in:

- duplicates

- anagrams

`from collections import Counter Counter("aabbbcc")`

---

12. Time Complexity Cheat Sheet

Operation	Complexity
list append	O(1)
list insert at beginning	O(n)
dict lookup	O(1) avg
sorting	O(n log n)
set membership	O(1)

---

13. Most Common Interview Questions List (Rapid Fire)

- Explain `__init__`
- Explain `__str__` vs `__repr__`
- What is `lambda`?
-

What is `map`, `filter`, `reduce`?

- What is slicing?
- What is `with` statement?
- What are context managers?
- Difference between `is` and `==`
- How Python manages memory?
- How to optimize Python code?

---

14. Mini Project / Practical Questions

Q: Read a file and count word frequency

`from collections import Counter with open("file.txt", "r") as f: words = f.read().split() freq = Counter(words) print(freq.most_common(10))`

Q: Remove duplicates from list while preserving order

```
def unique(arr): seen = set() res = [] for x in arr: if x not in seen: res.append(x) seen.add(x) return res
```