🐍 Python for Data Science – Q&A

Q1. Difference between list, tuple, set, and dictionary.
👉 List: ordered, mutable.
👉 Tuple: ordered, immutable.
👉 Set: unordered, unique elements.
👉 Dict: key–value pairs, fast lookup.

Q2. How do you handle missing values in a list?
👉 Use `None`, check with `if val is None`. Replace/remove using list comprehension or `filter()`.

Q3. What are Python decorators?
👉 Functions that modify behavior of another function without changing its code (e.g., `@staticmethod`, logging).

Q4. Explain shallow copy vs deep copy.
👉 Shallow copy copies references (nested objects still linked).
👉 Deep copy creates a full independent clone of all objects.

Q5. What is the difference between `is` and `==`?
👉 `is` checks identity (same memory object).
👉 `==` checks value equality.

Q6. How does Python manage memory?
👉 Through automatic garbage collection using reference counting + cyclic GC.

Q7. Explain list comprehension with an example.
👉 Compact way to create lists.
Example: `[x**2 for x in range(5)] → [0,1,4,9,16]`.

Q8. What is the difference between `@staticmethod` and `@classmethod`?
👉 `@staticmethod`: no access to class/instance.
👉 `@classmethod`: takes `cls`, can access/modify class variables.

Q9. What is pickling/unpickling in Python?
👉 Pickling = serialize object → byte stream.
👉 Unpickling = load byte stream back → object.

Q10. Difference between `append()` and `extend()`.
👉 `append()` adds a single item.
👉 `extend()` adds all elements from another iterable.

Q11. Explain generators and `yield`.
👉 Generators produce values lazily using `yield`, saving memory vs storing full list.

Q12. How are exceptions handled in Python?
👉 Using `try-except-finally`. Errors caught in `except`, cleanup in `finally`.

Q13. Difference between `range()` and `xrange()` (Python 2 vs 3).
👉 Python 2: `range()` returns list, `xrange()` returns generator-like object.
👉 Python 3: only `range()` exists (like `xrange`).

Q14. How is NumPy better than lists?
👉 Faster, memory-efficient, supports vectorized operations, and multi-dimensional arrays.

Q15. Explain broadcasting in NumPy.
👉 Automatic expansion of smaller arrays to match shape of larger arrays for elementwise ops.

Q16. How do you merge/join two Pandas DataFrames?
👉 Use `pd.merge(df1, df2, on='key')` or `df1.join(df2)`.

Q17. Difference between `loc[]` and `iloc[]`.
👉 `loc[]`: label-based indexing.
👉 `iloc[]`: position-based indexing.

Q18. How do you remove duplicates in a DataFrame?
👉 Use `df.drop_duplicates()`.

Q19. How to read large CSV files efficiently in Python?
👉 Use `chunksize` in `pd.read_csv()`, or Dask/Polars for big data.

Q20. What are lambda functions?
👉 Anonymous, inline functions. Example: `lambda x: x+1`.

Q21. Explain `map()`, `filter()`, `reduce()`.
👉 `map()` → apply function to iterable.
👉 `filter()` → keep elements passing condition.
👉 `reduce()` → cumulative function reduction.

Q22. How do you handle categorical data in Pandas?
👉 Use `pd.get_dummies()` for one-hot encoding or `astype('category')`.

Q23. Explain Python's Global Interpreter Lock (GIL).
👉 A mutex allowing only one thread to execute Python bytecode at a time (limits true parallelism).

Q24. How do you improve performance in Python code?
👉 Use vectorization (NumPy, Pandas), multiprocessing, Cython, or efficient data structures.

Q25. Write code to reverse a string and check if it's a palindrome.

```
s = "madam"
rev = s[::-1]
print("Palindrome" if s == rev else "Not palindrome")
```