Python `.join()` — 20 Interview Questions and Answers

A Light■Theme Illustrated PDF with real-world examples, explanations, and common pitfalls of Python’s `str.join()` method.

■ Introduction

The `.join()` method concatenates elements of an iterable (like a list or tuple) into a single string with a chosen separator. It’s faster and cleaner than loops for building strings.

Q1. Purpose of `.join()`

```
words = ["Python", "is", "fun"]
print(" ".join(words))  # Python is fun
```

Q2. Join non-string items

```
nums = [1, 2, 3]
print(" ".join(map(str, nums)))  # 1 2 3
```

Q3. Join with commas

```
names = ["Alice", "Bob", "Charlie"]
print(", ".join(names))  # Alice, Bob, Charlie
```

Q4. Join tuple

```
t = ("2025", "10", "25")
print("-".join(t))  # 2025-10-25
```

Q5. Empty list

```
print(",".join([]))  # ''
```

Q6. Join dict keys

```
data = {"a": 1, "b": 2}
print("-".join(data))  # a-b
```

Q7. Join dict values

```
data = {"a": "apple", "b": "banana"}
print(", ".join(data.values()))  # apple, banana
```

Q8. Join string chars

```
s = "HELLO"
print(".".join(s))  # H.E.L.L.O
```

Q9. Join with newline

```
lines = ["Line1", "Line2"]
print("\n".join(lines))
```

Q10. Join folder path

```
folders = ["home", "user", "docs"]
print("/".join(folders))  # home/user/docs
```

Q11. Uppercase join

```
words = ["python", "rocks"]
print(" ".join(w.upper() for w in words))
```

Q12. Join unique

```
items = ["a","b","a"]
print(", ".join(set(items)))
```

Q13. Join with zeros

```
nums = [5, 12, 45]
print("-".join(f"{n:02}" for n in nums))  # 05-12-45
```

Q14. Flatten and join

```
nested = [["A", "B"], ["C", "D"]]
flat = [x for s in nested for x in s]
print("".join(flat))  # ABCD
```

Q15. Join generator

```
print(" ".join(str(i**2) for i in range(5)))
```

Q16. Empty vs space

```
chars = ["A", "B"]
print("".join(chars))  # AB
print(" ".join(chars))  # A B
```

Q17. Join binary

```
nums = [3,7,10]
print(" | ".join(bin(n)[2:] for n in nums))
```

Q18. Reverse join

```
names = ["Tom", "Jerry"]
print(" -> ".join(reversed(names)))
```

Q19. Join CSV

```
row = ["ID","Name","Age"]
print(",".join(row))
```

Q20. Trick — list.join()

```
" ".join(["a","b"])  # ■
["a","b"].join(" ")  # ■ AttributeError
```

■ Summary & Interview Takeaways

• `.join()` combines iterables into a single string efficiently.

• Works only on strings; convert numbers first.

• Faster than loops for concatenation.

• Common in CSV creation, paths, and sentence building.

• Inverse of `split()`.