# Python `count()` Method — 30 Interview Questions & Answers

Covers String, List, Tuple, and Collection-based count() usage — from basics to complex interview scenarios.

## 1■■ What does str.count() do?

```
"banana".count("na")          # 2
"banana".count("na", 0, 4)    # 1
```

## 2■■ Is str.count() case-sensitive?

```
"Python python".count("Python")      # 1
"Python python".lower().count("python")  # 2
```

## 3■■ Does str.count() handle overlapping matches?

```
s, sub = "aaaa", "aa"
sum(1 for i in range(len(s)) if s.startswith(sub, i))  # 3
s.count("aa")  # 2
```

## 4■■ start and end parameters usage

```
"banana".count("a", 1, -1)   # 2
```

## 5■■ Empty substring behavior

```
"abc".count("")       # 4
"abc".count("", 1, 3) # 3
```

## 6■■ Unicode support

```
"café".count("é")  # 1
```

## 7■■ Count words via separators

```
s = "a b  c"
spaces = s.count(" ")
len(s.split())  # 3
```

## 8■■ Count multiple substrings

```
s = "error warn error info warn"
targets = ["error", "warn"]
{t: s.count(t) for t in targets}
```

## 9■■ Count in bytes/bytearray

```
b = b"\x00\x01\x00\x02"
b.count(b"\x00")      # 2
bytearray(b).count(0)  # 2
```

## ■ Count single characters

```
"xxxy".count("x")  # 3
```

## 11■■ Frequency validation

```
p = "a1b2c"
sum(ch.isdigit() for ch in p) >= 2
```

## 12■■ Count lines via newlines

```
text = "a\nb\nc"
text.count("\n") + 1   # 3
```

## 13■■ list.count() basic usage

```
[1,2,2,3].count(2)  # 2
```

## 14■■ True, 1, 1.0 equivalence

```
[True, 1, 1.0, "1"].count(1)  # 3
```

## 15■■ NaN counting edge case

```
import math
vals = [float('nan'), float('nan')]
sum(math.isnan(v) for v in vals)  # 2
```

## 16■■ Count sublists or tuples

```
L = [[1,2],[1,2],[2,3]]
L.count([1,2])  # 2
```

## 17■■ Performance comparison

```
names = ["a","b","a"]
names.count("a")
from collections import Counter
Counter(names)["a"]
```

## 18■■ tuple.count()

```
(1,2,2,3).count(2)  # 2
```

## 19■■ deque.count()

```
from collections import deque
dq = deque([1,2,2,3])
dq.count(2)  # 2
```

## 20■■ array.count()

```
from array import array
a = array('i', [1,2,2,3])
a.count(2)  # 2
```

## 21■■ Custom object count

```
class Box:
    def __init__(self,v): self.v=v
    def __eq__(self,o): return isinstance(o,Box) and self.v==o.v
L=[Box(1),Box(1),Box(2)]
L.count(Box(1))  # 2
```

## 22■■ Counting None

```
[None, 0, None].count(None)  # 2
```

## 23■■ Count multiple efficiently

```
from collections import Counter
L=[1,1,2,3,3,3]
C=Counter(L)
C[1],C[2],C[3]  # (2,1,3)
```

## 24■■ count() vs Counter

```
# count(): single query
# Counter: many queries, top-K, set ops
```

## 25■■ Predicate counting

```
nums = [1,2,3,4,5,6]
sum(n%2==0 for n in nums)  # 3
```

## 26■■ Count bytes patterns

```
b=b"\x00\xff\x00"
b.count(0)         # 2
b.count(b"\x00\xff")  # 1
```

## 27■■ Validate minimum occurrences

```
s="Error: x, error: y, ERROR: z"
s.lower().count("error")>=3
```

## 28■■ Performance vs regex

```
"log log ERROR log".count("ERROR")  # faster
```

## 29■■ Multi-delimiter words

```
import re
text="a,b; c\td"
len(re.split(r"[,\s;]+", text))  # 4
```

## 30■■ Detect exactly-once patterns

```
s="BEGIN ... END"
s.count("BEGIN")==1 and s.count("END")==1
```

## ■ Summary

This document covers string and collection-based count() usage with 30 progressive examples and answers — a practical interview-ready reference.