ChatGPT - DSA

Competitive Programming Tricks Cheatshet for Python

Below is the **Competitive Programming Python Tricks Cheatsheet** — compact, practical, and focused on *speed, readability, and eliminating boilerplate*.
It includes **patterns, micro-optimizations, templates, and Python-specific hacks** used by top CP coders.

🏆 PYTHON COMPETITIVE PROGRAMMING TRICKS — ULTIMATE CHEATSHEET

⚡ 1. Fast Input / Output

```
python

import sys
input = sys.stdin.readline
```

Multiple ints

```
python

a, b, c = map(int, input().split())
```

Read list fast

```
python

arr = list(map(int, input().split()))
```

Fast output

```
python

sys.stdout.write(str(ans) + "\n")
```

🧮 2. Constants & Bounds

```python
INF = 10**18         # CP max int
NEG_INF = -10**18
MOD = 10**9 + 7      # common modulus
MOD2 = 998244353
```

🧠 3. Python Micro-Optimizations

✔️ Avoid repeated lookups

```python
append = res.append
for x in arr:
    append(x)
```

✔️ Use local variables inside loops
✔️ Use **`sum()`**, **`min()`**, **`max()`** → C optimized
✔️ Prefer list comprehension over loops

🛠️ 4. Sorting Tricks

Sort by multiple keys

```python
arr.sort(key=lambda x: (x[0], -x[1]))
```

Sort in descending

```python
arr.sort(reverse=True)
```

Custom comparator

```python
from functools import cmp_to_key
arr.sort(key=cmp_to_key(lambda a, b: -1 if a+b > b+a else 1))
```

🧱 5. Frequency & Counting

Fast counter

```python
from collections import Counter
cnt = Counter(arr)
```

Default dictionary

```python
from collections import defaultdict
d = defaultdict(int)
d[x] += 1
```

🚀 6. Two-Pointers / Sliding Window Template

```python
l = 0
for r in range(len(nums)):
    # expand window with nums[r]

    while condition_breaks:
        l += 1   # shrink window

    # process window [l, r]
```

🎯 7. Binary Search Templates

On sorted array

```python
import bisect

idx = bisect.bisect_left(arr, x)
idx = bisect.bisect_right(arr, x)
```

Manual binary search

```python
l, r = 0, n-1
while l <= r:
    mid = (l+r)//2
    if arr[mid] == target:
        ...
    elif arr[mid] < target:
        l = mid + 1
    else:
        r = mid - 1
```

Binary search answer (BS on function)

```python
def good(x): ...

l, r = low, high
while l < r:
    mid = (l+r)//2
    if good(mid):
        r = mid
    else:
        l = mid + 1
```

🧭 8. Graphs (Adjacency List Template)

```python
from collections import defaultdict, deque

g = defaultdict(list)
for u, v in edges:
    g[u].append(v)
    g[v].append(u)
```

BFS

```python
q = deque([start])
vis = set([start])

while q:
    x = q.popleft()
    for nei in g[x]:
        if nei not in vis:
            vis.add(nei)
            q.append(nei)
```

DFS (recursive)

```python
sys.setrecursionlimit(10**7)

def dfs(node, parent):
    for nei in g[node]:
        if nei != parent:
            dfs(nei, node)
```

🧩 9. Dijkstra (Fast Template)

```python
import heapq

def dijkstra(start):
    dist = [INF] * n
    dist[start] = 0
```

```
    pq = [(0, start)]

    while pq:
        d, node = heapq.heappop(pq)
        if d != dist[node]:
            continue
        for nxt, w in graph[node]:
            nd = d + w
            if nd < dist[nxt]:
                dist[nxt] = nd
                heapq.heappush(pq, (nd, nxt))

    return dist
```

🧵 10. Union-Find (DSU)

```
python


class DSU:
    def __init__(self, n):
        self.p = list(range(n))
        self.sz = [1]*n

    def find(self, x):
        while self.p[x] != x:
            self.p[x] = self.p[self.p[x]]
            x = self.p[x]
        return x

    def union(self, a, b):
        pa, pb = self.find(a), self.find(b)
        if pa == pb: return False
        if self.sz[pa] < self.sz[pb]:
            pa, pb = pb, pa
        self.p[pb] = pa
        self.sz[pa] += self.sz[pb]
        return True
```

🧮 11. Prefix Sum / Difference Array

Prefix sum

```
python


pref = [0]
for x in arr:
    pref.append(pref[-1] + x)

# sum of range [l, r]
res = pref[r+1] - pref[l]
```

Difference array for range updates

```
python


diff = [0]*(n+1)
diff[l] += val
diff[r+1] -= val
```

🧱 12. Matrix Traversals

```python
DIRS = [(1,0),(-1,0),(0,1),(0,-1)]

for r in range(m):
    for c in range(n):
        for dr, dc in DIRS:
            nr, nc = r+dr, c+dc
```

🧵 13. Bit Manipulation

Check bit

```python
x & (1 << k)
```

Set bit

```python
x |= 1 << k
```

Remove bit

```python
x &= ~(1 << k)
```

Count bits

```python
bin(x).count("1")
# or
x.bit_count()
```

📌 14. Heap Tricks

Max heap in Python

```python
import heapq
pq = []
heapq.heappush(pq, -x)
x = -heapq.heappop(pq)
```

Heapify

```python
heapq.heapify(arr)
```

🔥 15. Short & Powerful Python Idioms

Swap

```python
a, b = b, a
```

Read NxM matrix

```python
mat = [list(map(int, input().split())) for _ in range(n)]
```

Flatten list

```python
flat = [x for row in mat for x in row]
```

Check sorted

```python
arr == sorted(arr)
```

⚙️ 16. LRU Cache (DP Optimization)

```python
from functools import lru_cache

@lru_cache(None)
def dp(i, j):
    ...
```

🧠 17. Combinatorics

nCr fast

```python
import math
math.comb(n, r)
```

Modular inverse

```python
pow(a, MOD-2, MOD)
```

🧮 18. String Tricks

Reverse string

```python
s[::-1]
```

Check palindrome

```python
s == s[::-1]
```

Fast frequency of chars

```python
cnt = [0]*26
for ch in s:
    cnt[ord(ch)-97] += 1
```

🌀 19. Recursion to Iteration Conversion

To avoid TLE + recursion limits:

```python
stack = [start]
while stack:
    node = stack.pop()
    for nei in graph[node]:
        stack.append(nei)
```

🔥 20. 50 Essential Python One-Liners for CP

Here are best ones:

```python
# 1. clamp value
x = max(low, min(high, x))

# 2. rotate list
arr = arr[k:] + arr[:k]

# 3. get index or -1
idx = next((i for i,v in enumerate(arr) if v==x), -1)

# 4. remove duplicates but keep order
res = list(dict.fromkeys(arr))

# 5. chunk array
chunks = [arr[i:i+k] for i in range(0,len(arr),k)]

# 6. sort dict by value
sorted_d = sorted(d.items(), key=lambda x: x[1])

# 7. transpose matrix
zip(*mat)

# 8. multiple assignments
a, b, *rest = arr

# 9. top K largest
heapq.nlargest(k, arr)

#10. bottom K smallest
heapq.nsmallest(k, arr)
```

🎁 Want More?

I can also generate:

✅ **Python CP Boilerplate**
✅ **Templates for Graph, DP, Trees, BIT, Segment Tree**
✅ **500 Python Tricks for CP**
✅ **Full DSA Python Handbook**

Just tell me **"Give me advanced version"** or specify a topic.

Add more tricks

Perfect — here is **Round 2: MORE Competitive Programming Python Tricks**, including **advanced micro-optimizations, hidden Python behaviours, math/graph shortcuts, DP hacks, bit-level tricks, heap magic, and pro one-liners** used in high-level contests.

🏆 ADVANCED PYTHON COMPETITIVE PROGRAMMING TRICKS — PART 2

⚡ 1. Insanely Fast Input Parsing

Read entire input at once

```python
import sys
data = sys.stdin.read().strip().split()
it = iter(data)

def nxt():
    return next(it)
```

Then:

```python
n = int(nxt())
arr = [int(nxt()) for _ in range(n)]
```

This is **10x faster** on huge inputs.

⚡ 2. Avoid TLE by Pre-binding Builtins

```python
append = arr.append
range_ = range
for i in range_(n):
    append(i)
```

CP Python trick: **Local variables are faster than global lookups**.

⚙️ 3. Precompute Everything (CP Meta-Trick)

Good for repeated queries:

Precompute prefix, suffix, factorial, inverse, primes:

```python
fact = [1] * (n+1)
for i in range(1, n+1):
    fact[i] = fact[i-1] * i % MOD
```

Precompute smallest prime factors (SPF)

```python
spf = list(range(N))
for i in range(2, int(N**0.5)+1):
    if spf[i] == i:
        for j in range(i*i, N, i):
            spf[j] = i
```

🔥 4. Fast Power Tricks

Last digit trick

```python
pow(a, b, 10)
```

Check if number is power of 2

```python
x & (x-1) == 0
```

🧮 5. Prefix/Range Tricks

Range max/min with prefix & suffix

```python
pre = [0]*n
suf = [0]*n
pre[0] = arr[0]
suf[-1] = arr[-1]

for i in range(1,n): pre[i] = max(pre[i-1], arr[i])
for i in range(n-2,-1,-1): suf[i] = max(suf[i+1], arr[i])
```

Range updates with diff array (2D)

```python
diff = [[0]*(m+1) for _ in range(n+1)]
diff[x1][y1] += val
diff[x2+1][y1] -= val
diff[x1][y2+1] -= val
diff[x2+1][y2+1] += val
```

🧊 6. Tuple Trick — Immutable and Fast

Tuples compare lexicographically:

```
python

(min_cost, node)
```

Useful in PQ/Dijkstra.

🚀 7. Better Graph Traversals

Use array instead of set for visited

```
python

vis = [False]*n
```

BFS 0–1 (super fast)

```
python

from collections import deque
dq = deque()
dq.append((0,start))
dist[start] = 0

while dq:
    d,u = dq.popleft()
    for v,w in adj[u]:
        nd = d + w
        if nd < dist[v]:
            dist[v] = nd
            if w == 0:
                dq.appendleft((nd,v))
            else:
                dq.append((nd,v))
```

⚡ 8. Pythonic Sliding Window (Cleaner Version)

```
python

from collections import Counter
need = Counter(t)
missing = len(t)

l = 0
for r, ch in enumerate(s):
    if need[ch] > 0:
        missing -= 1
    need[ch] -= 1

    while missing == 0:
        # window l..r valid
        if need[s[l]] == 0:
            missing += 1
        need[s[l]] += 1
        l += 1
```

🔢 9. Binary Search on Floats

```python
l, r = 0.0, 1e9
for _ in range(80):         # enough precision
    mid = (l + r) / 2
    if good(mid):
        r = mid
    else:
        l = mid
```

🛠️ 10. Fenwick Tree (BIT) 10-line Template

```python
class BIT:
    def __init__(self,n):
        self.f=[0]*(n+1)
    def update(self,i,val):
        while i<len(self.f):
            self.f[i]+=val
            i+=i&-i
    def query(self,i):
        s=0
        while i>0:
            s+=self.f[i]
            i-=i&-i
        return s
```

🌳 11. Segment Tree Compact Implementation

```python
n = len(arr)
seg = [0]*4*n

def build(i,l,r):
    if l==r:
        seg[i] = arr[l]
    else:
        m=(l+r)//2
        build(2*i,l,m)
        build(2*i+1,m+1,r)
        seg[i] = seg[2*i] + seg[2*i+1]
```

🧠 12. Python Memoization Tricks

Memo with dictionary

```python
memo = {}
def solve(x):
```

```
    if x in memo: return memo[x]
    ...
```

Fast boolean cache

```
python

cache = [None]*(n+1)
```

🔥 13. Bitmask DP Tricks

Iterate over submasks

```
python

s = mask
while s:
    s = (s-1) & mask
```

Iterate all bits that are 1

```
python

i = mask
while i:
    b = i & -i
    idx = b.bit_length()-1
    i ^= b
```

💡 14. Matrix + Graph Compression

Convert grid to graph index fast

```
python

node = r*m + c
r = node // m
c = node % m
```

🚀 15. Heap Tricks

Replace:

```
python

heapq.heappushpop(h, x)
```

is faster than:

```
python

heapq.heappush(h,x); heapq.heappop(h)
```

Maintain top K efficiently

```
python

h = arr[:k]
heapq.heapify(h)
for x in arr[k:]:
    if x > h[0]:
        heapq.heapreplace(h, x)
```

🎯 16. Greedy Trick — Always Use Sorted + Heap

When dealing with intervals, scheduling, tasks:

```
python

events.sort()
import heapq
pq=[]
for start, end in events:
    heapq.heappush(pq, end)
    if len(pq) > k:
        heapq.heappop(pq)
```

🧮 17. Math Tricks

GCD & LCM

```
python

import math
math.gcd(a,b)
a*b // math.gcd(a,b)
```

Faster modulo addition

```
python

(x + y) % MOD
```

Fermat's Little Theorem

```
python

inv = pow(a, MOD-2, MOD)
```

---

18. DP Space Optimization

```python
prev = [0]*m
curr = [0]*m

for i in range(n):
    for j in range(m):
        curr[j] = ...
    prev, curr = curr, prev
```

---

19. Use `bisect` Like a Pro

Insert in sorted list

```python
import bisect
i = bisect.bisect_left(a,x)
a.insert(i,x)
```

Count numbers < x

```python
cnt = bisect.bisect_left(a, x)
```

Find number of elements in range [l, r]

```python
bisect.bisect_right(a,r) - bisect.bisect_left(a,l)
```

---

20. Super Useful Python One-Liners (Advanced)

```python
# 1. clamp
x = (x > hi) * hi + (x < lo) * lo + (lo <= x <= hi) * x

# 2. index of max
idx = max(range(len(arr)), key=arr.__getitem__)

# 3. arg sort
order = sorted(range(len(arr)), key=arr.__getitem__)

# 4. flatten nested lists
sum(list_of_lists, [])

# 5. cumulative OR
from itertools import accumulate
list(accumulate(arr, lambda a,b: a|b))
```

```
# 6. prefix minima
list(accumulate(arr, min))

# 7. if item exists else default
val = next((x for x in arr if cond(x)), default)

# 8. running max diff
mx = 0; mn = 10**18
for x in arr:
    mn = min(mn, x)
    mx = max(mx, x-mn)
```

Want 50 more, or tricks for specific categories?

I can generate:

🔥 **Bit-Manipulation Master List**
🔥 **Advanced Graph Tricks (Centroid, SCC, LCA, Binary Lifting)**
🔥 **DP Pattern Cheatsheet (80+ templates)**
🔥 **Python Optimization for 10x Speed**

Just tell me:
➡️ **"Give me more categories"** or
➡️ **"Give me 100 more tricks"**

more 50

🔥 **Here are 50 MORE Competitive Programming Python Tricks — Part 3**
These are advanced, high-value micro-optimizations, math hacks, graph shortcuts, DP/bitmask patterns, and Pythonic one-liners used by top CP players.

🏆 50 Advanced Python Tricks for Competitive Programming (Part 3)

1–10: FAST LIST / ARRAY TRICKS

1. Reverse in-place

```
python

arr.reverse()
```

2. Rotate array using deque

```python
from collections import deque
d = deque(arr)
d.rotate(k)
```

3. Delete element by index fast

```python
del arr[i]
```

4. Copy list fast

```python
b = arr[:]   # faster than list(arr)
```

5. Slice assignment

```python
arr[:] = [x*2 for x in arr]
```

6. Remove all occurrences

```python
arr = [x for x in arr if x != target]
```

7. Find all indices of a value

```python
idxs = [i for i,x in enumerate(arr) if x==v]
```

8. Count distinct in O(n)

```python
len(set(arr))
```

9. Compute differences

```python
diff = [arr[i+1] - arr[i] for i in range(n-1)]
```

10. Pair elements

```python
pairs = list(zip(arr, arr[1:]))
```

11–20: DICTIONARY / HASH TRICKS

11. Fast default dict for lists

```python
from collections import defaultdict
g = defaultdict(list)
```

12. Frequency map building

```python
freq = defaultdict(int)
for x in arr: freq[x]+=1
```

13. Invert dictionary (one-to-one)

```python
inv = {v:k for k,v in d.items()}
```

14. Merge two dicts

```python
d3 = {**d1, **d2}
```

15. Multi-key sort trick

```python
sorted(d.items(), key=lambda x: (x[1], x[0]))
```

16. Remove a key safely

```python
d.pop(key, None)
```

17. Dictionary comprehension

```python
sq = {x: x*x for x in arr}
```

18. Sort dict by keys efficiently

```python
sorted(d)
```

19. Count if exists else 0

```python
count = d.get(x, 0)
```

20. Maintain order & uniqueness

```python
res = list(dict.fromkeys(arr))
```

21–30: MATH & NUMBER THEORY

21. All divisors of n (O√n)

```python
div = []
for i in range(1, int(n**0.5)+1):
    if n%i==0:
        div.append(i)
        if i*i != n: div.append(n//i)
```

22. Integer sqrt fast

```python
import math
s = int(math.isqrt(n))
```

23. Check perfect square

```python
s = int(math.isqrt(n))
ok = s*s == n
```

24. Prime check (fast)

```python
def is_prime(n):
    if n < 2: return False
    if n % 2 == 0: return n == 2
    i = 3
    while i*i <= n:
        if n%i==0: return False
        i+=2
    return True
```

25. LCM for list

```python
lcm = arr[0]
for x in arr[1:]:
    lcm = lcm * x // math.gcd(lcm, x)
```

26. Modular add/sub

```python
x = (a + b) % MOD
y = (a - b) % MOD
```

27. Modular multiply

```python
(a * b) % MOD
```

28. Modular exponent shorthand

```python
pow(a, b, MOD)
```

29. Sum of digits

```python
sum(map(int, str(n)))
```

30. Fast min/max of 3

```python
m = min(a, b, c)
```

31–40: STRING / TEXT TRICKS

31. Reverse words

```python
" ".join(s.split()[::-1])
```

32. Remove spaces

```python
s = s.replace(" ", "")
```

33. Character frequency (fast)

```python
from collections import Counter
c = Counter(s)
```

34. First unique character

```python
next((ch for ch in s if s.count(ch)==1), None)
```

35. Check substring

```python
if sub in s:
```

36. String rotation check

```python
ok = t in (s+s)
```

37. Longest prefix-suffix (naive but short)

```python
[L for L in range(len(s)) if s[:L]==s[-L:]]
```

38. Build string with repetition

```python
s = "ab" * 5
```

39. Remove vowels

```python
"".join(c for c in s if c not in "aeiou")
```

40. Convert digits to int list

```python
lst = list(map(int, s))
```

41–50: GRAPH / DP / ADVANCED TRICKS

41. BFS level processing

```python
from collections import deque
q = deque([src])
level = 0
while q:
    for _ in range(len(q)):
        node = q.popleft()
        # process
    level += 1
```

42. Short BFS template

```python
for u in q: ...
```

43. Graph from edges using list

```python
g = [[] for _ in range(n)]
for u,v in edges:
    g[u].append(v)
```

44. Topological sort

```python
from collections import deque
q = deque()
```

```
for i in range(n):
    if indeg[i]==0:
        q.append(i)

res = []
while q:
    u = q.popleft()
    res.append(u)
    for v in g[u]:
        indeg[v]-=1
        if indeg[v]==0:
            q.append(v)
```

45. DP Table initialization trick

```
python

dp = [[0]*m for _ in range(n)]
```

46. DP default negative infinity

```
python

dp = [[-10**15]*m for _ in range(n)]
```

47. Knapsack optimization

```
python

for w in reversed(range(weight, MAX)):
```

48. Fast LIS using bisect

```
python

import bisect
lis = []
for x in arr:
    i = bisect.bisect_left(lis, x)
    if i == len(lis): lis.append(x)
    else: lis[i] = x
```

49. Floyd cycle detection (tortoise & hare)

```
python

slow, fast = f(x), f(f(x))
while slow != fast:
    slow = f(slow)
    fast = f(fast)
```

50. Count set bits 10× faster

```python
x.bit_count()
```

✅ Want 50 more?

I can generate:

🔥 **50 Graph Tricks**
🔥 **50 DP Patterns**
🔥 **50 Bit Manipulation Super Tricks**
🔥 **50 Python Micro-Optimizations**
🔥 **50 CP Templates (Segment Tree, LCA, SCC, FFT)**

Just tell me: **"More 50"** or specify a category.

more 50

🔥 **Here you go — 50 MORE Competitive Programming Python Tricks (Part 4)**
These focus on **algorithm templates, optimization hacks, math shortcuts, iterators, bitwise magic, greedy patterns, and advanced Python idioms** used by elite CP coders.

🏆 50 MORE PYTHON CP TRICKS — PART 4

1–10: LIST / ARRAY OPTIMIZATION TRICKS

1. Fast min/max index

```python
idx_min = min(range(len(arr)), key=arr.__getitem__)
idx_max = max(range(len(arr)), key=arr.__getitem__)
```

2. Remove duplicates but preserve last occurrence

```python
seen = set()
res = []
for x in reversed(arr):
    if x not in seen:
        seen.add(x)
        res.append(x)
res.reverse()
```

3. Build cumulative AND

```python
from itertools import accumulate
cand = list(accumulate(arr, lambda a,b: a & b))
```

4. Build cumulative OR

```python
cor = list(accumulate(arr, lambda a,b: a | b))
```

5. Build cumulative XOR

```python
cxor = list(accumulate(arr, lambda a,b: a ^ b))
```

6. Prefix maximum

```python
from itertools import accumulate
pmax = list(accumulate(arr, max))
```

7. Prefix minimum

```python
pmin = list(accumulate(arr, min))
```

8. Compute running average

```python
avg = [sum(arr[:i+1])/(i+1) for i in range(len(arr))]
```

9. Pairwise differences

```python
diff = [b-a for a,b in zip(arr, arr[1:])]
```

10. Pairwise sums

```python
sums = [a+b for a,b in zip(arr, arr[1:])]
```

11–20: STRING MANIPULATION TRICKS

11. Remove all digits

```
python

"".join(c for c in s if not c.isdigit())
```

12. Keep only digits

```
python

"".join(c for c in s if c.isdigit())
```

13. Sliding window word extraction

```
python

words = s.split()
```

14. Count substring occurrences

```
python

s.count(sub)
```

15. Fast palindrome check

```
python

s == s[::-1]
```

16. Replace multiple chars

```
python

tr = str.maketrans({"a":"x","b":"y"})
s2 = s.translate(tr)
```

17. String rotation (k steps)

```
python

rot = s[k:] + s[:k]
```

18. Find all starts of substring

```python
idxs = [i for i in range(len(s)) if s.startswith(sub, i)]
```

19. Remove non-alphabetic

```python
"".join(filter(str.isalpha, s))
```

20. Reverse every word

```python
" ".join(w[::-1] for w in s.split())
```

21–30: MATH & NUMBER THEORY ADVANCED

21. Fast Fibonacci using matrix exponentiation

```python
def fib(n):
    if n==0: return 0
    F = [[1,1],[1,0]]
    def mul(A,B):
        return [
            [(A[0][0]*B[0][0]+A[0][1]*B[1][0]), (A[0][0]*B[0][1]+A[0][1]*B[1][1])],
            [(A[1][0]*B[0][0]+A[1][1]*B[1][0]), (A[1][0]*B[0][1]+A[1][1]*B[1][1])]
        ]
    def matpow(M,n):
        if n==1: return M
        if n%2==0:
            H = matpow(M,n//2)
            return mul(H,H)
        return mul(M, matpow(M,n-1))
    return matpow(F,n)[0][1]
```

22. Euler Totient (φ) fast sieve

```python
phi = list(range(N+1))
for i in range(2,N+1):
    if phi[i]==i:
        for j in range(i, N+1, i):
            phi[j] -= phi[j]//i
```

23. Check coprime

```python
math.gcd(a,b) == 1
```

24. Modular divide

```python
(a * pow(b,MOD-2,MOD)) % MOD
```

25. Integer log base 2

```python
n.bit_length() - 1
```

26. Multiply safely under modulo

```python
(a % MOD) * (b % MOD) % MOD
```

27. Sum of first N numbers

```python
n*(n+1)//2
```

28. Sum of squares

```python
n*(n+1)*(2*n+1)//6
```

29. Power of 2 ceiling

```python
1 << ((x-1).bit_length())
```

30. Count trailing zeros in binary

```python
(x & -x).bit_length() - 1
```

31–40: BIT MANIPULATION PRO TRICKS

31. Get lowest set bit

```
python

lsb = x & -x
```

32. Remove lowest set bit

```
python

x &= x-1
```

33. Toggle bit

```
python

x ^= (1 << i)
```

34. Check bit

```
python

(x >> i) & 1
```

35. All subsets of mask

```
python

s = mask
while s:
    yield s
    s = (s-1) & mask
```

36. Combine bits from two masks

```
python

m = a | b
```

37. Intersection mask

```
python

m = a & b
```

38. Exclude bits

```
python

m = a & ~b
```

39. Highest bit index

```python
x.bit_length() - 1
```

40. Convert integer → bitstring

```python
bin(x)[2:]
```

41–50: GRAPH / DP / ALGO TRICKS

41. BFS for shortest path grid

```python
DIR = [(1,0),(-1,0),(0,1),(0,-1)]
```

42. Multi-source BFS

```python
from collections import deque
q = deque(starts)
```

43. BFS with visited array

```python
vis = [[False]*m for _ in range(n)]
```

44. DFS iterative

```python
stack=[src]
while stack:
    u = stack.pop()
```

45. DFS with manual recursion limit

```python
sys.setrecursionlimit(10**7)
```

46. DP with memo array

```python
dp = [-1]*n
```

47. 1D rolling DP optimization

```python
dp_new[j] = min(dp[j], dp[j-1]) + cost[j]
```

48. Interval scheduling pattern

```python
intervals.sort(key=lambda x: x[1])
```

49. Greedy sort by ratio

```python
arr.sort(key=lambda x: x[0]/x[1])
```

50. Sweep line template

```python
events = []
events.append((start, +1))
events.append((end, -1))
events.sort()
```

✅ Want 50 more?

I can generate sets like:

🔥 **50 Python Micro-optimizations**
🔥 **50 Tree & Graph Tricks**
🔥 **50 Dynamic Programming Patterns**
🔥 **50 Bit Manipulation Deep Tricks**
🔥 **50 Short Python One-Liners**
🔥 **50 Templates for All CP Problems**

Tell me **"more 50"**, or choose a category!

Lambda Functions usage and advanced tricks

Below is the **Ultimate Cheatsheet of Lambda Function Usage & Advanced Tricks in Python**, tailored for **competitive programming, DSA, and fast coding**.
Covers: sorting, advanced keys, functional tricks, closures, inline logic, currying, custom comparators, priority queues, BFS/DFS lambdas, and more.

🏆 PYTHON LAMBDA FUNCTIONS — USAGE + ADVANCED TRICKS

⚡ 1. Basic Lambda Format

```
python

f = lambda x: x * 2
```

Inline usage:

```
python

result = (lambda a, b: a + b)(3, 5)
```

⚡ 2. Sorting with Lambda (Most Common CP Use)

Sort by second element

```
python

arr.sort(key=lambda x: x[1])
```

Sort by multiple keys

```
python

arr.sort(key=lambda x: (x[0], -x[1]))
```

Sort tuples by absolute value

```
python

arr.sort(key=lambda x: abs(x))
```

Sort based on custom mapping

```python
order = {'A':1, 'B':2, 'C':3}
s.sort(key=lambda c: order[c])
```

⚡ 3. Lambda with `min` / `max`

Get max element by custom value

```python
best = max(arr, key=lambda x: x.score)
```

Get index of max

```python
idx = max(range(len(arr)), key=lambda i: arr[i])
```

⚡ 4. Lambda with `map`, `filter`, `reduce`

Map

```python
sq = list(map(lambda x: x*x, arr))
```

Filter

```python
evens = list(filter(lambda x: x%2==0, arr))
```

Reduce

```python
from functools import reduce
prod = reduce(lambda a,b: a*b, arr)
```

⚡ 5. Lambda for Conditional Expressions

Inline if-else

```python
f = lambda x: "even" if x%2==0 else "odd"
```

Piecewise functions

```python
f = lambda x: -1 if x<0 else (0 if x==0 else 1)
```

⚡ 6. Lambda in Priority Queues (heapq)

Python's **heapq** cannot use custom comparator, but you can encode priority:

Min-heap by value

```python
import heapq
pq = []
heapq.heappush(pq, (lambda x: -x)(5))  # simulate max heap
```

Priority tuple

```python
heapq.heappush(pq, (lambda x: (x.cost, x.time))(obj))
```

⚡ 7. Lambda in Dijkstra / BFS / Graph DP Keys

Sort graph edges by weight

```python
edges.sort(key=lambda x: x[2])
```

Sort adjacency by heuristic

```python
adj[u].sort(key=lambda v: h[v])
```

⚡ 8. Lambda for String Sorting

Sort by frequency

```python
s_sorted = sorted(s, key=lambda c: (-s.count(c), c))
```

(OK for short strings; avoid for long ones)

Case-insensitive sort

```python
words.sort(key=lambda w: w.lower())
```

⚡ 9. Lambda as Inline Named Functions (Hack)

Simulate "quick function definition" in CP:

Square

```python
sq = lambda x: x*x
```

Digit sum

```python
ds = lambda x: sum(map(int, str(x)))
```

Palindrome check

```python
pal = lambda s: s == s[::-1]
```

⚡ 10. Lambda Closures (store variables inside lambda)

```python
def power_mod(p):
    return lambda x: pow(x, p, 10**9+7)

cube_mod = power_mod(3)
cube_mod(5)  # 125 mod 1e9+7
```

⚡ 11. Lambda inside List Comprehension

```python

f = lambda x: x*x + 1
vals = [f(i) for i in range(10)]
```

⚡ 12. Lambda for Sorting Custom Structures

Sorting based on difference

```python

arr.sort(key=lambda x: x.profit - x.weight)
```

Sorting by ratio (Knapsack greedy)

```python

items.sort(key=lambda x: x.value / x.weight, reverse=True)
```

⚡ 13. Using `cmp_to_key` for REAL custom comparators (Advanced)

Python removed custom comparator support, but you can recreate it:

```python

from functools import cmp_to_key

def cmp(a, b):
    return -1 if str(a)+str(b) > str(b)+str(a) else 1

arr.sort(key=cmp_to_key(cmp))
```

Useful in:

- forming **largest number**
- custom tie-breakers
- stable ordering with complex rules

⚡ 14. Lambda to Avoid Recomputing (micro-optimization)

Binding repeated calculations into a lambda:

```python

mod = 10**9 + 7
mulmod = lambda a,b: (a*b) % mod
```

⚡ 15. Lambda with Default Arguments (freeze values)

Capture loop variable safely

```python
funcs = []
for i in range(5):
    funcs.append(lambda x, i=i: x+i)
```

Each lambda captures **its own i**.

⚡ 16. Lambda for Tuple Transformations

Flip tuple

```python
flip = lambda t: (t[1], t[0])
```

Add two tuples

```python
add = lambda a,b: (a[0]+b[0], a[1]+b[1])
```

⚡ 17. Lambda for Sorting by Digit Sum

```python
arr.sort(key=lambda x: sum(map(int, str(x))))
```

⚡ 18. Lambda for Quick Hashing (DSA trick)

Hash pair

```python
hp = lambda a,b: (a*1315423911 ^ b*2654435761) & ((1<<64)-1)
```

⚡ 19. Lambda in Binary Search (search on function)

```python
good = lambda x: x*x >= target
```

Used in:

- square root
- capacity problems
- optimization problems

⚡ 20. Lambda for Mathematical Transformations

Normalize vector

```python
norm = lambda x,y: (x/(x*x+y*y)**0.5, y/(x*x+y*y)**0.5)
```

Clamp value

```python
clamp = lambda x,l,h: max(l, min(h, x))
```

⚡ 21. Lambda for Merging Intervals

```python
intervals.sort(key=lambda x: x[0])
```

⚡ 22. Lambda as Inline Recursion (Y-combinator style)

(VERY advanced CP trick)

```python
import sys
sys.setrecursionlimit(10**7)

dfs = (lambda f:
          (lambda x,*args: f(f,x,*args))
       )(lambda f, node, par: [
            f(f, nei, node) for nei in g[node] if nei != par
       ])
```

Allows recursion **without defining a named function**.

⚡ 23. Lambda to Prebind Functions for Speed

```python
r = range
rr = lambda x: r(x)
```

Or:

```python
ap = arr.append
```

Used in CP to avoid repeated attribute lookups.

⚡ 24. Lambda for Modular Arithmetic Shortcuts

```python
addm = lambda a,b: (a+b) % MOD
subm = lambda a,b: (a-b) % MOD
mulm = lambda a,b: (a*b) % MOD
```

⚡ 25. Lambda in List Sorting Mixed Types

```python
arr.sort(key=lambda x: (isinstance(x,str), x))
```

⚡ 26. Lambda for Counting Property Over List

```python
count_pos = lambda arr: sum(1 for x in arr if x > 0)
```

⚡ 27. Lambda for BFS Node Priority

```python
get_priority = lambda node: dist[node] + heuristic[node]
```

⚡ 28. Lambda + itertools for powerful CP utilities

```python
from itertools import groupby
groups = [(k, len(list(v))) for k,v in groupby(arr)]
```

⚡ 29. Lambda for Partial Application (mini-currying)

```python
adder = lambda a: (lambda b: a+b)
```

⚡ 30. Lambda for Fast Inline DP transition

```python
trans = lambda a,b,c: min(a,b,c) + cost
```

🔥 WANT EVEN MORE?

I can give:

✅ **50 more advanced lambda tricks**
✅ **Lambdas in functional programming**
✅ **Lambda-heavy templates for sorting, graphs, DP**
✅ **Lambda-driven coding patterns for CP**

Just say **"more lambda tricks"**.

more 50

🔥 **Here are 50 MORE Advanced Lambda Tricks (Part 2)** — focusing on **sorting magic, functional programming, closures, combinators, DP shortcuts, graph tricks, math transformations, currying, decorators using lambdas, bitwise tricks, and CP-friendly patterns**.

This is the **deepest & most advanced lambda cheatsheet** you'll find for Python competitive programming.

🧠 50 MORE ADVANCED LAMBDA TRICKS — PART 2

⭐ 1–10: Advanced Sorting & Key Functions

1. Sort by custom boolean priority

```python
arr.sort(key=lambda x: (x<0, abs(x)))
```

2. Sort by string length then lexicographically

```python
arr.sort(key=lambda s: (len(s), s))
```

3. Sort by number of set bits

```python
arr.sort(key=lambda x: (x.bit_count(), x))
```

4. Sort by modulo

```python
arr.sort(key=lambda x: (x % k, x))
```

5. Sort by distance from a pivot

```python
pivot = 10
arr.sort(key=lambda x: abs(x - pivot))
```

6. Sort by digit sum

```python
arr.sort(key=lambda x: sum(map(int, str(x))))
```

7. Sort points by angle (polar ordering)

```python
import math
arr.sort(key=lambda p: math.atan2(p[1], p[0]))
```

8. Sort by last digit then by value

```python
arr.sort(key=lambda x: (x%10, x))
```

9. Sort list of dicts by multiple fields

```
python

arr.sort(key=lambda d: (d["age"], -d["score"]))
```

10. Sort strings by vowel count

```
python

vowels = set("aeiou")
arr.sort(key=lambda s: sum(c in vowels for c in s))
```

⭐ 11–20: Functional Programming Lambdas

11. Inline generator with lambda

```
python

gen = lambda n: (i*i for i in range(n))
```

12. Inline infinite generator

```
python

from itertools import count
inf = lambda start=0: count(start)
```

13. Lambda returning another lambda (currying)

```
python

add = lambda x: lambda y: x+y
```

14. Lambda pipeline

```
python

pipe = lambda f,g: lambda x: g(f(x))
```

15. Lambda to map over matrix

```
python

apply = lambda mat,f: [list(map(f,row)) for row in mat]
```

16. Function composition

```python
compose = lambda *fs: lambda x:
    (lambda v: [f(v) for f in fs][-1])(x)
```

17. Lambda that memoizes itself (closure trick)

```python
def memo():
    cache = {}
    return lambda f,x: cache.setdefault(x, f(x))
```

18. Lambda as switch-case

```python
switch = lambda x: {0:"zero", 1:"one", 2:"two"}.get(x, "other")
```

19. Lambda returning tuple of functions

```python
ops = lambda: (lambda a,b: a+b, lambda a,b: a*b)
plus, mul = ops()
```

20. Lambda filtering specific pattern

```python
filter_vowel = lambda lst: list(filter(lambda x: x[0] in "aeiou", lst))
```

⭐ 21–30: Math & Transform Lambdas

21. Euclidean distance

```python
dist = lambda a,b: ((a[0]-b[0])**2 + (a[1]-b[1])**2)**0.5
```

22. Manhattan distance

```python
man = lambda a,b: abs(a[0]-b[0]) + abs(a[1]-b[1])
```

23. Normalize vector

```python
norm = lambda x,y: (x/(x*x+y*y)**0.5, y/(x*x+y*y)**0.5)
```

24. Dot product

```python
dot = lambda a,b: sum(x*y for x,y in zip(a,b))
```

25. Clamp range

```python
clamp = lambda x,l,h: max(l, min(h, x))
```

26. Safe division

```python
sdiv = lambda a,b: a/b if b else float('inf')
```

27. Polynomial evaluation

```python
poly = lambda c,x: sum(ci * x**i for i,ci in enumerate(c))
```

28. Linear interpolation

```python
lerp = lambda a,b,t: a + (b-a)*t
```

29. Rotate 2D point

```python
rot = lambda p,th: (p[0]*math.cos(th)-p[1]*math.sin(th),
                    p[0]*math.sin(th)+p[1]*math.cos(th))
```

30. Check prime (fast)

```python
isprime = lambda n: n>1 and all(n%i for i in range(2, int(n**0.5)+1))
```

⭐ 31–40: Graph & DP Lambdas

31. Adjacency extractor

```
python

nbrs = lambda u, g: g[u]
```

32. Weighted graph relax

```
python

relax = lambda d,u,v,w: min(d[v], d[u]+w)
```

33. BFS condition lambda

```
python

valid = lambda r,c: 0<=r<n and 0<=c<m and not seen[r][c]
```

34. Grid direction iterator

```
python

D = [(1,0),(-1,0),(0,1),(0,-1)]
step = lambda r,c: ((r+dr, c+dc) for dr,dc in D)
```

35. DP combine lambda

```
python

comb = lambda a,b,c: min(a,b,c)
```

36. Knapsack choice

```
python

choose = lambda take,leave: max(take, leave)
```

37. Binary lifting parent selector

```
python

up2 = lambda u: parent[parent[u]]
```

38. Merge intervals lambda

```python
merge = lambda a,b: (min(a[0],b[0]), max(a[1],b[1]))
```

39. Check DAG node validity

```python
can = lambda u: indeg[u]==0
```

40. Dijkstra PQ element

```python
node_key = lambda u: (dist[u], u)
```

⭐ 41–50: Lambda Tricks, Hacks, Patterns

41. Inline max-of-three

```python
mx3 = lambda a,b,c: max(a,b,c)
```

42. Reverse mapping lambda

```python
revmap = lambda d: {v:k for k,v in d.items()}
```

43. Toggle boolean

```python
tog = lambda x: not x
```

44. Multi-key hash lambda

```python
hp = lambda *v: hash(v)
```

45. Range mapper

```python
mapr = lambda a,b: list(range(a,b+1))
```

46. Inline recursion (Y combinator short form)

```
python

Y = lambda f: (lambda x: f(lambda *args: x(x)(*args)))
              (lambda x: f(lambda *args: x(x)(*args)))
```

47. Inline DFS using lambda + stack

```
python

dfs = lambda start: (lambda s,vis: (
    (lambda _: None)(
        [s.append(v) for v in g[s.pop()] if v not in vis and not vis.add(v)]
    )
))( [start], set() )
```

48. Inline DP table initialization

```
python

dp = lambda n,m,val=0: [[val]*m for _ in range(n)]
```

49. Frequency counter lambda

```
python

freq = lambda arr: {x:arr.count(x) for x in set(arr)}
```

50. Conditional apply lambda

```
python

apply_if = lambda cond, f, x: f(x) if cond(x) else x
```

🔥 Want 50 MORE lambda tricks (Part 3)?

I can give:

✔️ **Lambda for bit manipulation**
✔️ **Lambda for segment trees / Fenwick trees**
✔️ **Lambda for greedy strategies**
✔️ **Lambda-based memoized recursion templates**
✔️ **Lambda-heavy CP one-liners**

Just reply: **"more lambda tricks"**.

🔥 **PYTHON ONE-LINER MASTERY — The Ultimate Competitive Programming & DSA Cheatsheet**
This is a **super-dense collection** of powerful, elegant, and highly practical **Python one-liners** covering **arrays, strings, DP, graphs, bits, math, heaps, sets, dicts, functional patterns, and CP shortcuts**.

Use this as your **master one-liner toolkit**.

🏆 PYTHON ONE-LINER MASTERY

⭐ 1. ARRAY & LIST ONE-LINERS

Create list of n zeros

```python
arr = [0]*n
```

List of 1..n

```python
arr = list(range(1, n+1))
```

Even numbers

```python
ev = [x for x in arr if x%2==0]
```

Flatten 2D list

```python
flat = [x for row in mat for x in row]
```

Reverse list

```python
rev = arr[::-1]
```

Prefix sum

```python
pref = [sum(arr[:i]) for i in range(len(arr)+1)]
```

Rotate list k steps

```python
rot = arr[k:] + arr[:k]
```

Unique elements (preserve order)

```python
uniq = list(dict.fromkeys(arr))
```

Pairwise sums

```python
pairs = [a+b for a,b in zip(arr, arr[1:])]
```

Differences

```python
diff = [arr[i+1]-arr[i] for i in range(len(arr)-1)]
```

⭐ 2. STRING ONE-LINERS

Reverse string

```python
rev = s[::-1]
```

Check palindrome

```python
is_pal = s == s[::-1]
```

Count vowels

```python
vc = sum(c in "aeiou" for c in s)
```

Remove vowels

```python
no_v = "".join(c for c in s if c not in "aeiou")
```

First non-repeating char

```python
first_unique = next((c for c in s if s.count(c)==1), None)
```

Sort characters by frequency

```python
res = "".join(sorted(s, key=lambda c: (-s.count(c), c)))
```

Check rotation

```python
is_rot = t in (s+s)
```

⭐ 3. DICTIONARY ONE-LINERS

Count frequencies

```python
freq = {x: arr.count(x) for x in set(arr)}
```

Reverse a dictionary

```python
rev = {v: k for k,v in d.items()}
```

Merge dictionaries

```python
merged = {**d1, **d2}
```

Sort dictionary by value

```python
sorted_items = sorted(d.items(), key=lambda x: x[1])
```

⭐ 4. SET ONE-LINERS

Remove duplicates

```python
uniq = list(set(arr))
```

Intersection

```python
inter = a & b
```

Union

```python
uni = a | b
```

Difference

```python
diff = a - b
```

⭐ 5. HEAP ONE-LINERS

Max heap push

```python
heapq.heappush(pq, -x)
```

Get top-k

```python
topk = heapq.nlargest(k, arr)
```

Heapify

```python
heapq.heapify(arr)
```

Replace min (faster push+pop)

```python
heapq.heapreplace(h, x)
```

⭐ 6. SORTING ONE-LINERS

Sort by multiple keys

```python
arr.sort(key=lambda x: (x[0], -x[1]))
```

Sort by digit sum

```python
arr.sort(key=lambda x: sum(map(int, str(x))))
```

Sort by absolute value

```python
arr.sort(key=lambda x: abs(x))
```

Sort by number of set bits

```python
arr.sort(key=lambda x: (x.bit_count(), x))
```

⭐ 7. BIT MANIPULATION ONE-LINERS

Count set bits

```python
bits = x.bit_count()
```

Check power of 2

```python
isp2 = x and not (x & (x-1))
```

Lowest set bit

```python
lsb = x & -x
```

Remove lowest set bit

```python
y = x & (x-1)
```

Toggle kth bit

```python
x ^= (1 << k)
```

⭐ 8. MATH ONE-LINERS

GCD & LCM

```python
g = math.gcd(a,b)
l = a*b//g
```

Sum of digits

```python
sd = sum(map(int,str(n)))
```

Prime check (sqrt)

```python
isprime = n>1 and all(n%i for i in range(2,int(n**0.5)+1))
```

Factor list

```python
fac = [i for i in range(1,n+1) if n%i==0]
```

Fast power mod

```python
pm = pow(a,b,MOD)
```

⭐ 9. DP ONE-LINERS

Initialize DP

```python
dp = [[0]*m for _ in range(n)]
```

Rolling DP array

```python
prev, curr = curr, prev
```

Top-down memo

```python
from functools import lru_cache
@lru_cache(None)
def dp(i,j): ...
```

min of 3

```python
ans = min(a,b,c)
```

⭐ 10. GRAPH ONE-LINERS

Build adjacency list

```python
g = [[] for _ in range(n)]
[g[u].append(v) or g[v].append(u) for u,v in edges]
```

BFS directions

```python
DIR = [(1,0),(-1,0),(0,1),(0,-1)]
```

Valid grid cell check

```python
valid = lambda r,c: 0<=r<n and 0<=c<m
```

Multi-source BFS

```python
from collections import deque
q = deque(starters)
```

⭐ 11. FUNCTIONAL ONE-LINERS

Map + filter

```python
res = list(map(f, filter(cond, arr)))
```

Compose two functions

```python
compose = lambda f,g: lambda x: g(f(x))
```

Reduce

```python
from functools import reduce
prod = reduce(lambda a,b: a*b, arr)
```

Inline recursion trick

```python
fact = (lambda f: lambda x: 1 if x<=1 else x*f(f)(x-1))(lambda f: lambda x: 1)
```

⭐ 12. MATRIX ONE-LINERS

Transpose

```python
t = list(zip(*mat))
```

Zero matrix

```python
mat = [[0]*m for _ in range(n)]
```

Identity matrix

```python
I = [[int(i==j) for j in range(n)] for i in range(n)]
```

Rotate matrix 90°

```python
rot = list(zip(*mat[::-1]))
```

⭐ 13. MISC CP POWER ONE-LINERS

Clamp

```python
clamp = lambda x,l,h: max(l,min(h,x))
```

Running max

```python
import itertools
rmax = list(itertools.accumulate(arr, max))
```

Running min

```python
rmin = list(itertools.accumulate(arr, min))
```

Chunk list

```python
chunks = [arr[i:i+k] for i in range(0,len(arr),k)]
```

Safe get from dict

```python
v = d.get(key, default)
```

Infinite generator

```python
from itertools import count
inf = count()
```

⭐ 14. EPIC ONE-LINERS FOR CP (RARE BUT GOLD)

Convert int → binary list

```python
bits = [int(c) for c in bin(x)[2:]]
```

Largest number from list of ints

```python
from functools import cmp_to_key
res = "".join(sorted(map(str, arr), key=cmp_to_key(lambda a,b: -1 if a+b>b+a else 1)))
```

LIS (O(n log n))

```python
import bisect
lis = []
[lis.append(x) if (i := bisect.bisect_left(lis,x))==len(lis) else lis.__setitem__(i,x) for x in
arr]
```

Fast prefix OR

```python
import itertools
pref_or = list(itertools.accumulate(arr, lambda a,b: a|b))
```

Fast prefix XOR

```python
pref_xor = list(itertools.accumulate(arr, lambda a,b: a^b))
```

🔥 Want 100 more one-liners across DP, graphs, math, strings, sets, bits, greedy, and interviews?

Just say **"more one-liners"**.

more oneliners

🔥 **Here are 100 MORE Python One-Liners (Mastery Part 2)** — focused on **DSA, competitive programming, graph theory, strings, math, bitwise, DP, greedy, I/O, and functional magic**. These are ultra-practical one-liners you can use in CP contests to write **clean, fast, expressive solutions**.

🏆 PYTHON ONE-LINER MASTERY — PART 2 (100 NEW ONELINERS)

⭐ 1–20: LIST & ARRAY ONE-LINERS

1. Sum of every sliding window of size k

```python
wins = [sum(arr[i:i+k]) for i in range(n-k+1)]
```

2. Running cumulative XOR

```python
cx = [arr[0]] + [arr[0]^arr[i] for i in range(1,n)]
```

3. Insert element at index i

```python
new = arr[:i] + [x] + arr[i:]
```

4. Remove element at index i

```python
new = arr[:i] + arr[i+1:]
```

5. Replace all negatives with 0

```python
new = [max(0,x) for x in arr]
```

6. Last k elements

```python
last_k = arr[-k:]
```

7. Count positives

```python
pos = sum(x>0 for x in arr)
```

8. Cumulative max diff

```python
best = max(arr[i]-min(arr[:i]) for i in range(len(arr)))
```

9. Convert flat → 2D matrix

```python
mat = [arr[i:i+m] for i in range(0,len(arr),m)]
```

10. First index of even number

```python
idx = next((i for i,x in enumerate(arr) if x%2==0), -1)
```

11. Replace None with 0

```python
clean = [0 if x is None else x for x in arr]
```

12. Delete duplicates but keep last

```python
clean = list(dict.fromkeys(reversed(arr)))[::-1]
```

13. Random element

```python
import random; r = random.choice(arr)
```

14. Reverse every second element

```python
rev2 = arr[::-2]
```

15. Prefix AND

```python
import itertools; pand = list(itertools.accumulate(arr, lambda a,b: a & b))
```

16. All numbers greater than avg

```python
above = [x for x in arr if x > sum(arr)/len(arr)]
```

17. Most frequent element

```python
mf = max(set(arr), key=arr.count)
```

18. Replace negatives with their abs

```python
ab = [abs(x) for x in arr]
```

19. List without repetition (orderless)

```python
u = list(set(arr))
```

20. Rotate left by k

```python
rot = arr[k:]+arr[:k]
```

⭐ 21–40: STRING ONE-LINERS

21. Remove digits

```python
nod = ''.join(c for c in s if not c.isdigit())
```

22. Extract digits only

```python
digits = ''.join(filter(str.isdigit, s))
```

23. All substrings (O(n²))

```python
subs = [s[i:j] for i in range(len(s)) for j in range(i+1,len(s)+1)]
```

24. Check isogram

```python
iso = len(s)==len(set(s))
```

25. Capitalize every word

```python
cap = ' '.join(w.capitalize() for w in s.split())
```

26. Most common char

```python
mc = max(set(s), key=s.count)
```

27. Longest word

```python
lw = max(s.split(), key=len)
```

28. Replace multiple chars

```python
tr = str.maketrans('aeiou','12345'); t = s.translate(tr)
```

29. Keep only alphabetic

```python
clean = ''.join(filter(str.isalpha, s))
```

30. Letter frequencies

```python
from collections import Counter; freq = Counter(s)
```

31. Unique chars in sorted order

```python
uniq = ''.join(sorted(set(s)))
```

32. Check anagram

```python
ana = sorted(s1)==sorted(s2)
```

33. Remove consecutive duplicates

```python
rcd = ''.join(c for i,c in enumerate(s) if i==0 or c!=s[i-1])
```

34. Reverse words

```python
revw = ' '.join(s.split()[::-1])
```

35. First uppercase char

```python
uc = next((c for c in s if c.isupper()), None)
```

36. Count uppercase

```python
countU = sum(c.isupper() for c in s)
```

37. Remove punctuation

```python
import re; clean = re.sub(r'\W','',s)
```

38. Duplicate characters

```python
dup = [c for c in s if s.count(c)>1]
```

39. Convert string to int list

```python
ints = list(map(int, s))
```

40. Character histogram

```python
hist = {c:s.count(c) for c in set(s)}
```

⭐ 41–60: DICT / SET / HEAP ONE-LINERS

41. Nested dictionary

```python
d = {i:{j:i*j for j in range(n)} for i in range(n)}
```

42. Frequency by comprehension

```python
freq = {x:arr.count(x) for x in set(arr)}
```

43. Argmax in dict

```python
best = max(d, key=d.get)
```

44. Argmin in dict

```python
worst = min(d, key=d.get)
```

45. Set symmetric difference

```python
sym = a ^ b
```

46. Find duplicates in list

```python
dups = {x for x in arr if arr.count(x)>1}
```

47. Sorted set

```python
ss = sorted(set(arr))
```

48. Heap push & pop (min-heap)

```python
import heapq; x = heapq.heappushpop(h, val)
```

49. Create max-heap

```python
h = [-x for x in arr]; heapq.heapify(h)
```

50. Merge lists using heap

```python
import heapq; merged = list(heapq.merge(a,b))
```

51. Get smallest k

```python
sm = heapq.nsmallest(k, arr)
```

52. Get largest k

```python
lg = heapq.nlargest(k, arr)
```

53. Min of dict values

```python
mn = min(d.values())
```

54. Max of dict values

```python
mx = max(d.values())
```

55. Delete key safely

```python
d.pop(key, None)
```

56. Invert mapping safely

```python
rev = {v:k for k,v in d.items()}
```

57. Remove a list of keys

```python
[d.pop(k,0) for k in keys]
```

58. Count #distinct

```python
distinct = len(set(arr))
```

59. Set of duplicates only

```python
dups = {x for x in set(arr) if arr.count(x)>1}
```

60. Dictionary comprehension with condition

```python
filt = {k:v for k,v in d.items() if v>0}
```

⭐ 61–80: MATH, BITS, NUMBER THEORY ONE-LINERS

61. Check power of 2

```python
p2 = x>0 and x&(x-1)==0
```

62. Count bits

```python
bits = x.bit_count()
```

63. Lowest set bit

```python
lsb = x & -x
```

64. Highest set bit index

```python
hi = x.bit_length()-1
```

65. Factor list

```python
fac = [i for i in range(1,n+1) if n%i==0]
```

66. Prime test

```python
isprime = n>1 and all(n%i for i in range(2,int(n**0.5)+1))
```

67. Fibonacci fast doubling (one-line interface)

```python
fib = lambda n: (lambda f,n: f(f,n))(lambda f,n: (1,0) if n==0 else (lambda a,b,c,d:
(a*a+b*b,a*b+b*d,b*b+d*d,a*b+b*d))(*f(f,n>>1)) if n&1 else (lambda a,b,c,d:
(a*a+b*b,a*b+b*d,b*b+d*d,a*b+b*d))( *f(f,n>>1) )[2:], n)[1]
```

(CP hack, use when needed)

68. Modular inv

```python
inv = pow(a,MOD-2,MOD)
```

69. Digit sum

```python
ds = sum(map(int,str(n)))
```

70. Manhattan distance

```python
man = lambda a,b: abs(a[0]-b[0])+abs(a[1]-b[1])
```

71. Euclidean distance

```python
dist = lambda a,b: ((a[0]-b[0])**2+(a[1]-b[1])**2)**0.5
```

72. Min of list via reduce

```python
from functools import reduce; mn = reduce(min, arr)
```

73. Max of list via reduce

```python
mx = reduce(max, arr)
```

74. Generate primes sieve

```python
prime=[True]*(n+1); [prime.__setitem__(j,False) for i in range(2,int(n**0.5)+1) if prime[i] for j in range(i*i,n+1,i)]
```

75. LCM of list

```python
import math; l=arr[0]; [l:=l*x//math.gcd(l,x) for x in arr[1:]]
```

76. Square test

```python
sq = int(n**0.5)**2==n
```

77. Next power of two

```python
np2 = 1<<((n-1).bit_length())
```

78. Fast exp modulo

```python
pm = pow(a,b,MOD)
```

79. Clamp

```python
cl = max(lo,min(hi,x))
```

80. Normalize vector

```python
norm = lambda x,y: (x/(x*x+y*y)**0.5, y/(x*x+y*y)**0.5)
```

⭐ 81–100: GRAPH, DP, GREEDY, RANDOM, I/O

81. Adjacency list from edges

```python
g=[[] for _ in range(n)]; [g[u].append(v) or g[v].append(u) for u,v in edges]
```

82. Grid neighbors

```python
N = lambda r,c: [(r+dr,c+dc) for dr,dc in [(1,0),(-1,0),(0,1),(0,-1)]]
```

83. BFS queue

```python
from collections import deque; q=deque([src])
```

84. Multi-source BFS queue

```python
q=deque(sources)
```

85. Validate grid cell

```python
valid = lambda r,c: 0<=r<n and 0<=c<m
```

86. Topological candidates

```python
zeros = [i for i in range(n) if indeg[i]==0]
```

87. Parent array initialization

```python
par = [-1]*n
```

88. DP init table

```python
dp = [[0]*m for _ in range(n)]
```

89. Rolling DP swap

```python
prev,cur = cur,prev
```

90. Max subarray (Kadane)

```python
best = max(cur := 0, *(cur:=max(x,cur+x) for x in arr))
```

91. Read space-separated ints

```python
arr = list(map(int,input().split()))
```

92. Read matrix

```python
mat = [list(map(int,input().split())) for _ in range(n)]
```

93. Fast input

```python
import sys; input=sys.stdin.readline
```

94. DFS recursive in one line

```python
dfs = lambda u: [dfs(v) for v in g[u] if v!=par[u]]
```

95. Binary search left

```python
import bisect; i = bisect.bisect_left(arr,x)
```

96. Binary search right

```python
i = bisect.bisect_right(arr,x)
```

97. Check sorted

```python
issorted = arr == sorted(arr)
```

98. Reverse sort

```python
arr.sort(reverse=True)
```

99. Custom sort comparator

```python
from functools import cmp_to_key; arr.sort(key=cmp_to_key(lambda a,b: (a>b)-(a<b)))
```

100. Infinite loop reading input

```python
for line in sys.stdin: ...
```

🔥 Want another 100 one-liners?

I can generate:

✅ **One-liners for Trees**
✅ **One-liners for Graph Algorithms**
✅ **One-liners for DP Patterns**
✅ **One-liners for Bit Manipulation**
✅ **One-liner math black-magic (FFT, matrix exp, combinatorics)**

Just say **"more oneliners"** or choose a category!