5. heapq (Priority queue / min Heap)

A heap is a special tree-based data structure where:

The smallest element is always at the top (in a min heap).

Python's heapq module implements a min heap by default.

```
import heapq
```

← without this import, heap functions won't work.

- Creating a Heap

Method 1: Convert list → heap

```
arr = [3, 1, 5]
heapq.heapify(arr)

print(arr)
```

O(n)

o/p: [1, 3, 5] # smallest element comes to front

Imp: Heap is not fully sorted. Only the smallest element is guaranteed at index 0.

- Core Operations

(i) Insert into heap — O(logn)

heapq.heappush(arr, 2)

⇒ Adds the element while maintaining heap property.

Example:

```
arr = [1, 3, 5]
heapq.heappush(arr, 2)

print(arr)
```

o/p: [1, 2, 5, 3]

(ii) Remove smallest element

heapq.heappop(arr) — O(logn)

⇒ Removes and returns minimum element.

Example:

```
arr = [1, 2, 5, 3]
x = heapq.heappop(arr)

print(x)          #1
```

Very Important: Python has Min Heap only.

Max Heap Trick (Interview favourite)

To simulate max heap in python:
⇒ Store negative values

- Insert in max heap
  heapq.heappush(arr, -x)

- Remove from max heap
  max_val = -heapq.heappop(arr)

Largest positive → smallest negative
10 → -10
5 → -5

Heap treats -10 as smallest, which corresponds to largest original value.

Example:

```
import heapq
arr = []

heapq.heappush(arr, -10)
heapq.heappush(arr, -5)
heapq.heappush(arr, -20)

print(-heapq.heappop(arr))
```

o/p: 20 (largest value)