**heapq –** 0 indexed min heap implementation in python.

* heapq.**heappushpop(heap, item)** - push item on the heap, then pop and return the smallest item from the heap.
* heapq.**heapify(x)** - transform list x into a heap, in-place, in linear time.
* heapq.**heapreplace(heap, item)** - pop and return the smallest item from the heap, and also push the new item.
* heapq.**nlargest(*n*, *iterable*, *key=None*)** - return a list with the *n* largest elements from the dataset defined by *iterable*.
* heapq.**nsmallest(*n*, *iterable*, *key=None*)** - return a list with the *n* smallest elements from the dataset defined by *iterable*.

**Counter**(*[iterable])* - a dict subclass for counting hashable objects.

* **most\_common**(*[n]*) - return a list of the n most common elements and their counts from the most common to the least. If n is omitted or None, it returns all elements in the counter.
* **total**() - compute the sum of the counts.
* **elements**() - return an iterator over elements repeating each as many times as its count.
* **subtract**([*iterable-or-mapping*]) - elements are subtracted from an *iterable* or from another *mapping* (or counter).

**deque**(*[iterable[, maxlen]]*) - double-ended queue, thread-safe, memory efficient appends and pops from either side of the deque with approximately the same O(1)

* **append(x)** - add x to the right side of the deque.
* **appendleft(x)** - add x to the left side of the deque.
* **pop()** - remove and return an element from the right side of the deque.
* **popleft()** - remove and return an element from the left side of the deque.

**Stack** – LIFO – last in first out - Both insertion and deletion at the top

* stack = [5]
* stack.**append**(6) # i.e, *stack.push(6)*, add item at the end of list,
* stack.**pop**() # remove the most recently added item.

**Queue** - FIFO - first in first out - insertion happens at the front but deletion happens at the rear of queue.

* queue = deque([2, 3])
* queue.**append**(5) – *i.e, enqueue()* - add an item at the rear of the queue
* queue.**popleft**() – *i.e., dequeue()* – remove an item for the front of the queue.

**Sort**

* items = [11, 4, 7, 2, 9, 3]; *sorted(items))* or *items.sort(reverse=True)*
* A = [[10, 8], [90, 2], [45, 6]]; sorted(A, key=lambda x: x[1]))
* A = [[2, 'Dog'], [0, 'Bird'], [7, 'Cat']]
* sorted(A, key=lambda x: x[0], reverse=True))

**Loop**

* items = [8, 4, 7, 2, 9, 3]

*for* ***idx****,* ***val*** *in enumerate(items):* OR  *for* ***i*** *in range(3, len(items)):*

* *grid2D[n x n] = [[0] \* n for \_ in range(n)]*