Recursion:

① Function description.

② Base Case.

③ divide.
combine.

Nested List Ex. [[1,2],3] → [1,2,3].

```
def gather_all(obj):
    if not isinstance(obj, list):
        return [obj]
    else:
        acc = []
        for item in obj:
            acc += gather_all(item)
        return acc
```

[□, △, ☆].

```
def gather_all_even(obj):
    if not isinstance(obj, list):
        if obj % 2 == 0:
            return [obj]
        else: return []
    else:
        acc = []
        for item in obj:
            acc += gather_all_even(item)
        return acc.
```

```
def count_lists(obj):
    if not isinstance(obj, list):
        return 0
    else:
        acc = 0
        for item in obj:
            acc += count_lists(item)
        return acc + 1
```

[[[1,2], [3]]]

4 lists

[□, ☆, △]

[1, [2, 3], 4]

[[1,2], [3,4]]

```
def count_at_depth(obj, n):
    if n == 0:
        return 1 if not isinstance(obj, list)
                 else 0
    if not isinstance(obj, list):
        return 0
    else:
        acc = 0
        for item in obj:
            acc += count_at_depth(item, n-1)
        return acc
```

[□, ☆, △]

```
def gather_by_depth(obj):
    """
    >>> gbd(5)
    {0: [5]}
    >>> gbd([[1, 2], 3])
    {2: [1, 2], 1: [3]}
    """
    if not isinstance(obj, list):
        return {0: [obj]}.

    else:
        acc = {}.
        for item in obj:
            d = gather_by_depth(item)
            for depth in d:
                if depth + 1 not in acc:
                    acc[depth + 1] = d[depth]
                else:
                    acc[depth + 1] += d[depth]

    return acc
```

[□, ☆, ◁]

[□, △, ☆]

d: {1: [A, B, C], 2: [D, F]}

acc: {2: [A, B], 3: [D, F]}

```
def add_one(obj) -> None:
    if isinstance(obj, list):
        for i in range(len(obj)):
            if not isinstance(obj[i], list):
                obj[i] += 1
            else:
                add_one(obj[i])
```

[5, 2, [3]] → [6, 3, [4]]

[0, ✱, 0]

```
def all_permutation(s):
    if len(s) <= 1:
        return [s]
    else:
        acc = []
        for perm in all_perm(s[1:]):
            for i in range(len(perm) + 1):
                temp = perm[:i] + s[0] + perm[i:]
                acc.append(temp)
        return acc
```

'123'

['123' '312'
'213' '321'
'231' '132']

['23', '32']

'' '1' '23'

'2' '1' '3'

'32' '1' ''

[5, [2], [3, [2, 6]]] [5, □, ○, ✱]

index 0 : not a list.

index 1 ~ ∞ , always a list.

```
L=['A', ['B'], ['C', ['D','E']]]
number_list(L, 6)
[6, [7], [8, [9, 10]]]
return 11
```

```
def number_list(obj, n)
    if !isinstance(obj, list):
        if (n(obj) == 1:
            obj[0] = n
            return n+1
        else:
            obj[0] = n.
            n += 1
            for item in obj[1:]:
                n = number_list(item, n)

    return n
```

[5, □, ✱, △]

```
def number_list(obj, n):
    # handle index 0.
    obj[0] = n
    n += 1

    # handle sublists.
    for item in obj[1:]:
        n = number_list(item, n).

    return n.
```