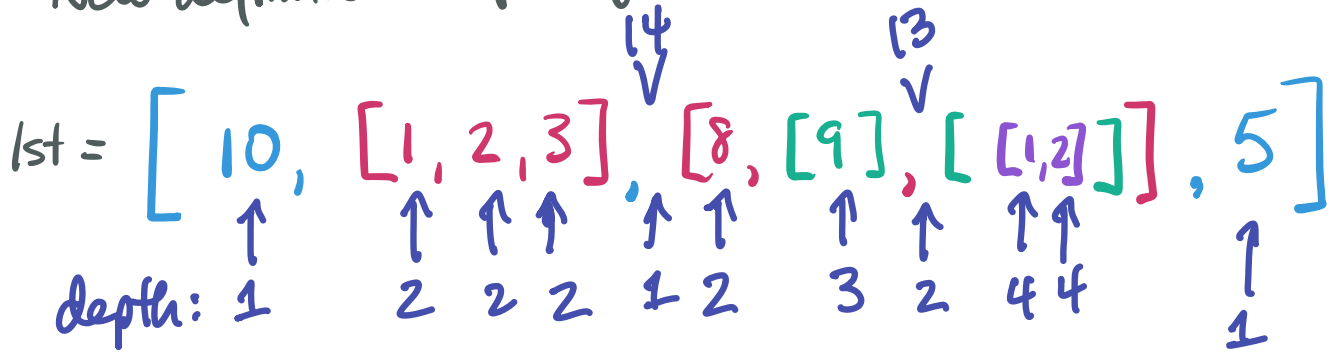


First-at-depth

New definition: depth of an object within a nested list.



lst = 68

Diagram illustrating the depth of an element in a list. The list is [68]. The depth of the element 68 is indicated by an arrow and the number 0 below it.

one & value we pass in needs to change from:

```
def first_at_depth(obj: Union[int, List], d: int) -> Optional[int]:  
    """Return the first (leftmost) item in <obj> at depth <d>.
```

Return None if there is no item at depth <d>.

Precondition: d >= 0.

1. Write doctests
2. Trace one big-ish example.
first-at-depth([10, [20, 21], [30, 40]], 2)

more sublists
here would
not matter!

first-at-depth ([10, [[20, 21]], [30, 40]], 2)

sublist	call	do / return correct result
→ 10	first-at-depth(10, 1)	None
→ [[20, 21]]	first-at-depth([[20, 21]], 1)	None
→ [30, 40]	first-at-depth([30, 40] 1)	30 stop!

Even if there were additional sublists in obj, there would be no reason to look at them. We've already found something at depth $d + \text{arity}$ that comes later cannot therefore be first!