Recursive functions

# xlist = [1, ["a", "b", ["x", "y"]]]

def flatten(lst):

flat\_lst = []

# We create an empty list we'll eventually fill with the flattened values

for item in lst:

if isinstance(item, list):

# We check if the item is a list and ...

flat\_lst.extend(flatten(item))

# ... add sublist's items to flat\_list by calling modularly recursive function

else:

flat\_lst.append(item)

# ... if item is not a list - directly append it to flat\_lst

return flat\_lst

The function flatten takes in a list as input and will output a flat (or one-dimensional) list. The function works as follows:

1. An empty list flat\_list is defined outside of the for loop.
2. A for loop iterates over each item in the list which was passed in, lst.
3. An if statement is used to check if item is a list itself.
4. If it is, flat\_list is extended using the result of calling flatten on the nested list.
5. In the else condition, meaning if the item is not a list, it's added directly to flat\_list.
6. At the end of the function, flat\_list is returned.
7. Once we get flat\_list, we can store all its values in a set by calling set() and passing in flat\_list as argument. That way, we can use set operations like union, intersection etc.
8. For example, if you wanted to add an element 'z' to xlist, you can just add it to flat\_list and create xset again.

So, in conclusion here, the function flatten is used for converting a multi-dimensional list into a flat list.