Let’s solve the given question with stack.

Since we know that every [ comes with at least one integer before it, meaning:

int[char] = int \* char

Let’s examine further implementations to this problem to make sure we have better algorithm.

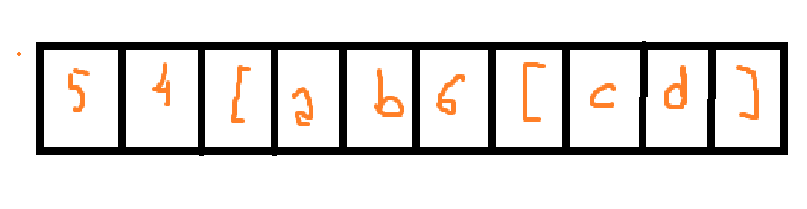
k[k[k[]]] or might be k[k[] k[] k[]] where k can be more than one digits.

In this question what makes us think of the subproblems is the [] meaning every ] has to have a [.

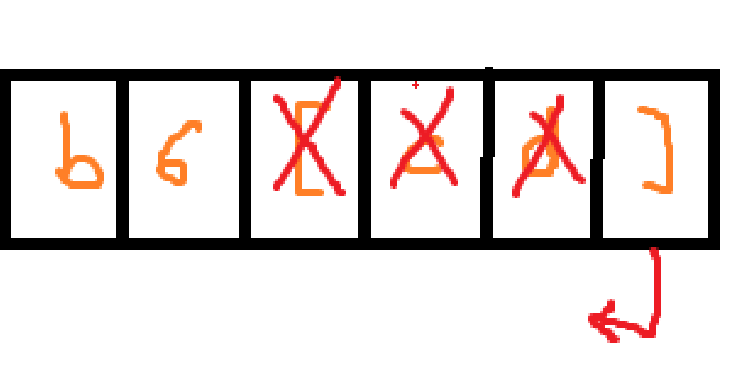
Let take:

54[ab6[cd]] for example.

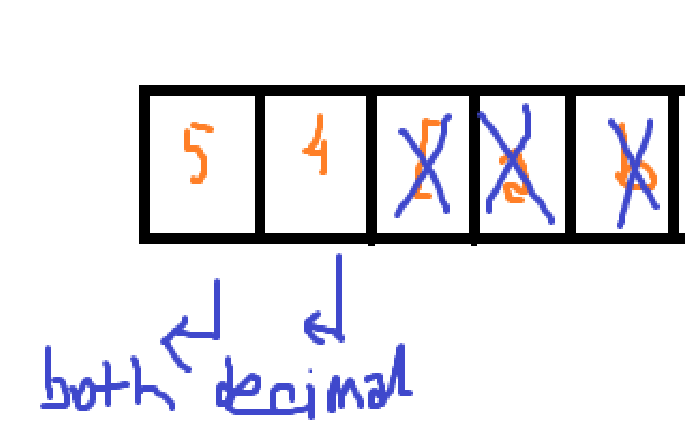
Let’s add every character into our stack. The main point here is we DO recognize the [ but we do not perform anything on it yet. Meaning we should keep adding it up until we do not see ].



After we see the ] we do not add it to the stack but what we do is, we start popping each of the characters until we see a [ which guarantees a number behind it.



Since 6.isdigit() and b is not an numeric value we can simply pop 6 (.pop() also returns the value.) to multiply the values inside: cd



After this operation is done we are to continue onward with the string given. And 54[ab6[cd]], ] doesn’t have to be ] but we are lucky in this case.

We start popping the list where we left off.

Then the very same thing happens. And in order to maintain such order like:

54[ abcdcdcdcdcd] we need to add our newly found substring to this very stack.