

Given a list of numbers, write a function that only returns a list of positive numbers.

Problem Statement

Input: a list of numbers

Output: a list of positive numbers from original list.

Process: checks in a list if a number is > 0 . and add it in a new list (if > 0).

Cases

$l_1 = [3, -4.5, 7, 2.1]$

$> 0 \rightarrow \text{newlist} = [3] \quad \times \quad [3, 7] \quad [3, 7, 2.1]$

$l_1 = []$

$\rightarrow \text{newlist} = []$

$l_1 = [3, 7, 2.1]$

$> 0 \rightarrow \text{newlist} = [3] \quad [3, 7] \quad [3, 7, 2.1]$

Design If a list is empty, should return an empty list.
Otherwise, check if the first element is > 0 .
If yes, add it to a new list.
If not, do nothing.
Then, move to the next element and repeat the previous process until we reach the end of the list. Finally, return the new list.

Implementation

```
def pos_numbers(lst):  
    if len(lst) == 0:  
        return []  
    else:  
        newlst = []  
        for k in range(len(lst)):  
            if lst[k] > 0:  
                newlst = newlst + [lst[k]]  
        return newlst
```

Diagram annotations:
- A red arrow points from the text $+ [lst[k]]$ to the $[lst[k]]$ part of the line `newlst = newlst + [lst[k]]`.
- A red circle is drawn around $[lst[k]]$ in the same line.
- A red arrow points from the circle down to the text $+ int$.
- A red arrow points from the `newlst` in the line `newlst = newlst + [lst[k]]` down to the text `list`.

Test

It does not work because newlst is a list and `lst[k]` is an integer \Rightarrow concatenation not possible

How to fix it? \rightarrow Go back to Implementation and replace `lst[k]` by `[lst[k]]`.

Then, all the tests pass (if the input is an empty list or a list of numbers).