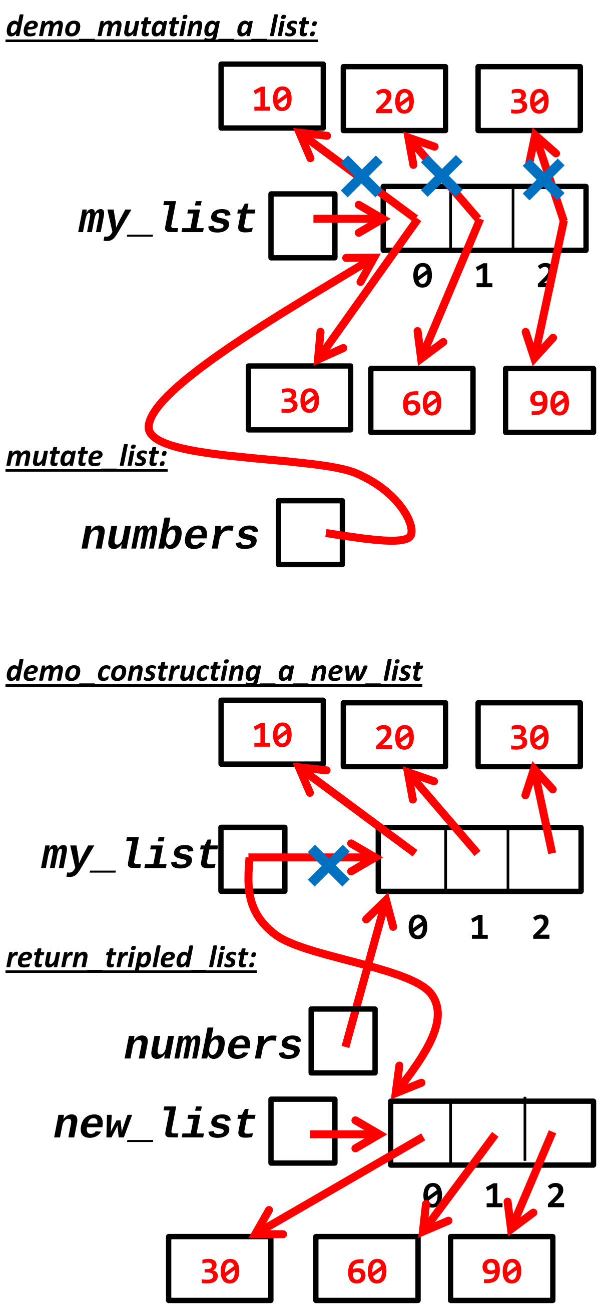
8. We have seen that it is simply not possible for a function to change the arrow in the ***caller*** that corresponds to one of the function's arguments. But many objects can be ***mutated***, which means that ***the object's value (not the variable's reference) changes.***

   To see this, draw a Box and Pointer diagram that shows what happens when *main* (below) executes. Also show the output that is printed. Do NOT show boxes for the loop variables ***k*** and ***number***, since that would clutter the diagram.

```
def main():
    demo_mutating_a_list()
    demo_constructing_a_new_list()

def demo_mutating_a_list():
    my_list = [10, 20, 30]
    mutate_list(my_list)
    print('A.', my_list)

def mutate_list(numbers):
    for k in range(len(numbers)):
        numbers[k] = numbers[k] * 3

def demo_constructing_a_new_list():
    my_list = [10, 20, 30]
    my_list = return_tripled_list(my_list)
    print('B.', my_list)

def return_tripled_list(numbers):
    new_list = []
    for number in numbers:
        new_list.append(number * 3)

    return new_list
```

Box and Pointer diagram:



Output:

A. [30, 60, 90]

B. [30, 60, 90]

mutate_list and ***return_tripled_list*** both end up with a tripled list. ***Which one uses less storage?*** [✓ *mutate_list*] ***return_tripled_list*** (circle your choice)