Exercise

```
def ave_positives(my_list):
    """Returns: average (float) of the positive values in my_list
    my_list: a list of numbers with at least one positive value"""
```

- Be goal oriented → ***can work backwards***
- ***Name a variable*** for any value that you need but don't have yet
- Break down a problem!
  - ... ***break into parts***
  - ... ***solve simpler version first***
- Remember loop/accumulation pattern

Exercise (Example Solution)

```
def ave_positives(my_list):
    """Returns: average (float) of the positive values in my_list
    my_list: a list of numbers with at least one positive value"""
    result= 0
    count= 0
    for x in my_list:
        if x > 0:
            result= result + x
            count= count + 1
    ave= result/count
    return ave
```

- Be goal oriented → ***can work backwards***
- ***Name a variable*** for any value that you need but don't have yet
- Break down a problem!
  - ... ***break into parts***
  - ... ***solve simpler version first***
- Remember loop/accumulation pattern

For-Loop Mistake #1 (Q)

Modifying the loop variable (here: x).

```
def add_one(the_list):
    """Adds 1 to every element in the list
    Precondition: the_list is a list of all numbers
    (either floats or ints)"""
    for x in the_list:
        x = x+1

a = [5, 4, 7]
add_one(a)
print(a)
```

What gets printed?

A: [5, 4, 7]
B: [5, 4, 7, 5, 4, 7]
C: [6, 5, 8]
D: **Error**
E: I don't know

For-Loop Mistake #1 (A)

Modifying the loop variable (here: x).

```
def add_one(the_list):
    """Adds 1 to every element in the list
    Precondition: the_list is a list of all numbers
    (either floats or ints)"""
    for x in the_list:
        x = x+1

a = [5, 4, 7]
add_one(a)
print(a)
```

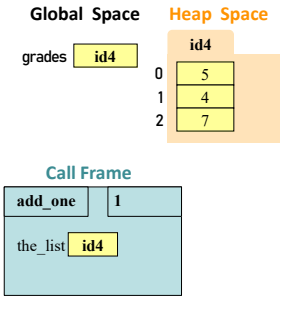
Actually it does not do this!

What gets printed?

A: [5, 4, 7] **CORRECT**
B: [5, 4, 7, 5, 4, 7]
C: [6, 5, 8]
D: **Error**
E: I don't know

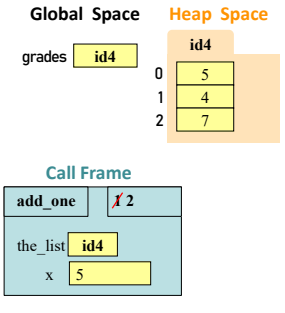
Modifying the Loop Variable (1)

```
def add_one(the_list):
    """Adds 1 to every elt
    Pre: the_list is all numb."""
1   for x in the_list:
2       x = x+1

grades = [5,4,7]
add_one(grades)
```

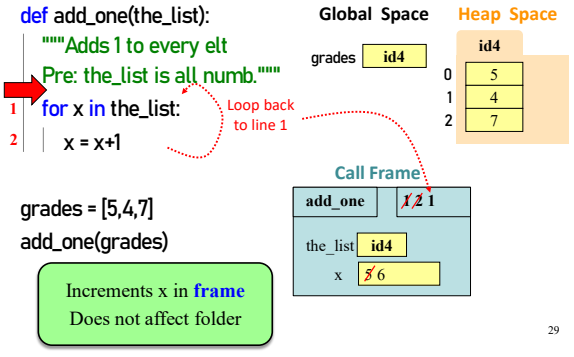


Modifying the Loop Variable (2)

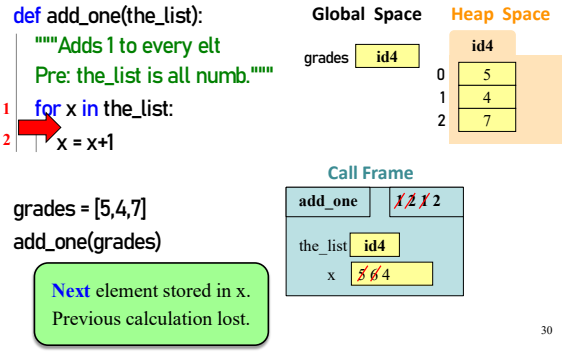
```
def add_one(the_list):
    """Adds 1 to every elt
    Pre: the_list is all numb."""
1   for x in the_list:
2       x = x+1

grades = [5,4,7]
add_one(grades)
```



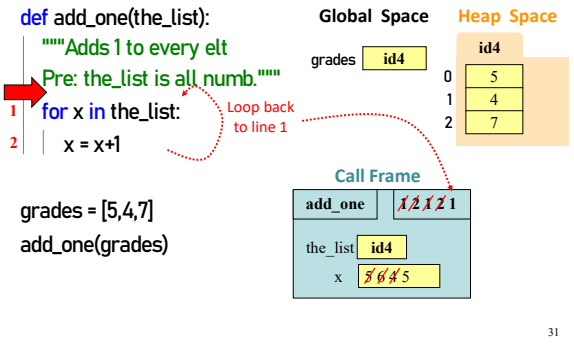
Modifying the Loop Variable (3)



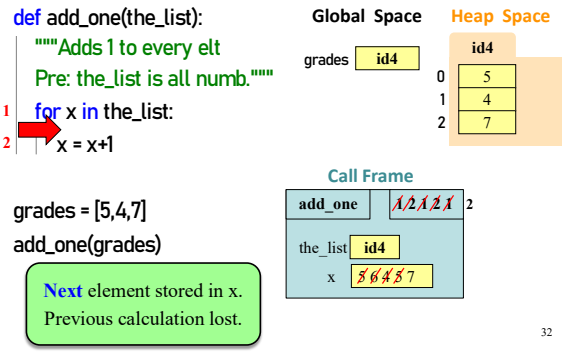
Modifying the Loop Variable (4)



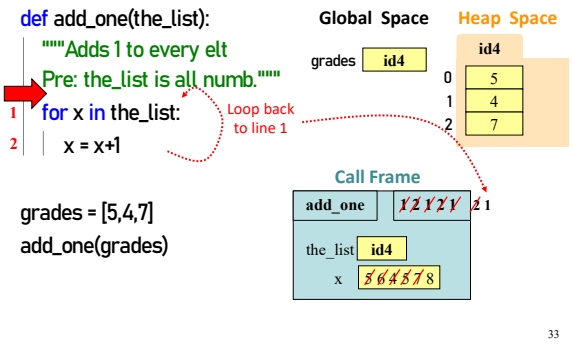
Modifying the Loop Variable (5)



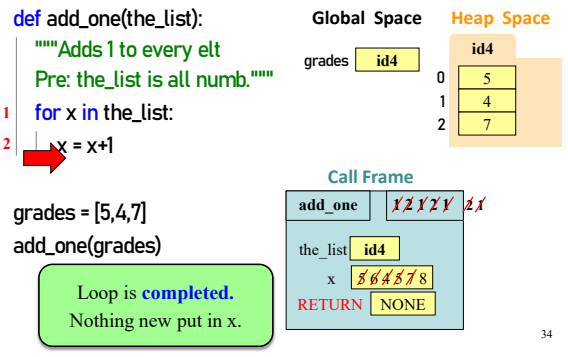
Modifying the Loop Variable (6)



Modifying the Loop Variable (7)



Modifying the Loop Variable (8)



Modifying the Loop Variable (9)

```
def add_one(the_list):
    """Adds 1 to every elt
    Pre: the_list is all numb."""
1   for x in the_list:
2       x = x+1
```

Global Space: grades → id4

Heap Space: id4 → 0: 5, 1: 4, 2: 7

Call Frame: add_one; the_list → id4; x → 5 6 4 5 7 8; N → NONE

```
grades = [5,4,7]
add_one(grades)
```

No lasting changes.
What did we accomplish? ☹

For-Loop Mistake #2 (Q)

Modifying the loop sequence as you walk through it.

What gets printed?

```
b = [1, 2, 3]
for a in b:
    b.append(a)
print(b)
```

A: never prints b
B: [1, 2, 3, 1, 2, 3]
C: [1, 2, 3]
D: I do not know

For-Loop Mistake #2 (A)

Modifying the loop sequence as you walk through it.

What gets printed?

```
b = [1, 2, 3]
for a in b:
    b.append(a)
print b
```

INFINITE LOOP!

A: never prints b **CORRECT***
B: [1, 2, 3, 1, 2, 3]
C: [1, 2, 3]
D: I do not know

*** Runs out of memory eventually, then probably throws an error.**