🍰 Interview Cake

← course home (/table-of-contents)

In-Place Algorithm

An **in-place** algorithm operates *directly* on its input and *changes* it, instead of creating and returning a *new* object. This is sometimes called **destructive**, since the original input is "destroyed" when it's edited to create the new output.

Careful: "In-place" does *not* mean "without creating any additional variables!" Rather, it means "without creating a new copy of the input." In general, an in-place function will only create additional variables that are $O(1)$ space.

Here are two functions that do the same operation, except one is in-place and the other is out-of-place:

```python
def square_list_in_place(int_list):
    for index, element in enumerate(int_list):
        int_list[index] *= element

    # NOTE: We could make this function just return, since
    # we modify int_list in place.
    return int_list


def square_list_out_of_place(int_list):
    # We allocate a new list with the length of the input list
    squared_list = [None] * len(int_list)

    for index, element in enumerate(int_list):
        squared_list[index] = element ** 2

    return squared_list
```

Working in-place is a good way to save space. An in-place algorithm will generally have $O(1)$ space cost.

But be careful: an in-place algorithm can cause side effects. Your input is "destroyed" or "altered," which can affect code *outside* of your function. For example:

```python
original_list = [2, 3, 4, 5]
squared_list  = square_list_in_place(original_list)

print "squared: %s" % squared_list
# Prints: squared: [4, 9, 16, 25]

print "original list: %s" % original_list
# Prints: original list: [4, 9, 16, 25], confusingly!

# And if square_list_in_place() didn't return anything,
# which it could reasonably do, squared_list would be None!
```

Generally, out-of-place algorithms are considered safer because they avoid side effects. You should only use an in-place algorithm if you're very space constrained or you're *positive* you don't need the original input anymore, even for debugging.

← course home (/table-of-contents)

Next up: JavaScript Scope ➜ (/question/js-scope? section=javascript&course=fc1)

Want more coding interview help?

Check out **interviewcake.com** for more advice, guides, and practice questions.