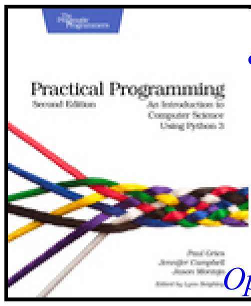
Mutability and Aliasing



- Chapter 8.5 Aliasing: What's in a Name?

Optional reading

Mutability

We say that lists are *mutable*: they can be modified. All the other types we have seen so far (`str`, `int`, `float` and `bool`) are *immutable*: they cannot be modified.

Here are several examples of lists being modified:

```
>>> classes = ['chem', 'bio', 'cs', 'eng']
>>>
>>> # Elements can be added:
>>> classes.append('math')
>>> classes
['chem', 'bio', 'cs', 'eng', 'math']
>>>
>>> # Elements can be replaced:
>>> classes[1] = 'soc'
>>> classes
['chem', 'soc', 'cs', 'eng', 'math']
>>>
>>> # Elements can be removed:
>>> classes.pop()
'math'
>>> classes
['chem', 'soc', 'cs', 'eng']
```

Aliasing

Consider the following code:

```
>>> lst1 = [11, 12, 13, 14, 15, 16, 17]
>>> lst2 = lst1
>>> lst1[-1] = 18
>>> lst2
[11, 12, 13, 14, 15, 16, 18]
```

After the second statement executes, `lst1` and `lst2` both refer to the same list. When two variables refer to the same objects, they are *aliases*. If that list is modified, both of `lst1` and `lst2` will see the change.

Jennifer Campbell • Paul Gries
University of Toronto