8.3: TRAVERSING A LIST

Contributed by Chuck Severance
Clinical Associate Professor (School of Information) at University of Michigan

The most common way to traverse the elements of a list is with a `for` loop. The syntax is the same as for strings:

CODE 8.3.1 (PYTHON):

```
%%python3

cheeses = ['feta','gorgonzola','brie','mozzarella']

for cheese in cheeses:
    print(cheese)
```

run restart

This works well if you only need to read the elements of the list. But if you want to write or update the elements, you need the indices. A common way to do that is to combine the functions `range` and `len` :

CODE 8.3.1 (PYTHON):

```
%%python3
numbers = [96, 23, 40, 76, 7]

for i in range(len(numbers)):
    numbers[i] = numbers[i] * 2
    print(numbers[i])
```

run restart

This loop traverses the list and updates each element. `len` returns the number of elements in the list. `range` returns a list of indices from 0 to $n - 1$, where n is the length of the list. Each time through the loop, `i` gets the index of the next element. The assignment statement in the body uses `i` to read the old value of the element and to assign the new value.

A `for` loop over an empty list never executes the body:

```
for x in empty:
    print('This never happens.')
```

Although a list can contain another list, the nested list still counts as a single element. The length of this list is four:

```
['spam', 1, ['Brie', 'Roquefort', 'Pol le Veq'], [1, 2, 3]]
```