Example: Summing the Elements of a List

def sum(thelist):
    """Returns: the sum of all elements in thelist
    Precondition: thelist is a list of all numbers (either floats or ints)"
    result = 0
    result = result + thelist[0]
    result = result + thelist[1]
    ... return result

There is a problem here

Working with Sequences

- Sequences are potentially unbounded
  - Number of elements inside them is not fixed
  - Functions must handle sequences of different lengths
  - Example: sum([1,2,3]) vs. sum([4,5,6,7,8,9,10])
- Cannot process with fixed number of lines
  - Each line of code can handle at most one element
  - What if # of elements > # of lines of code?
- We need a new control structure

For Loops: Processing Sequences

- Key Concepts
  - loop sequence: seq
  - loop variable: x
  - body: print(x)
- Remember:
  - Cannot program ...

For Loops: Processing Sequences

To execute the for-loop:
1. Check if there is a “next” element of loop sequence
2. If not, terminate execution
3. Otherwise, put the element in the loop variable
4. Execute all of the body
5. Repeat as long as 1 is true

Example: Summing the Elements of a List

def sum(thelist):
    """Returns: the sum of all elements in thelist
    Precondition: thelist is a list of all numbers (either floats or ints)"
    result = 0
    for x in thelist:
        result = result + x
    return result

For Loops and Conditionals

def num_ints(thelist):
    """Returns: the number of ints in thelist
    Precondition: thelist is a list of any mix of types"
    result = 0
    for x in the list:
        if type(x) == int:
            result = result + 1
    return result

Body
Modifying the Contents of a List

```python
def add_one(thelist):
    """(Procedure) Adds 1 to every element in the list
    Precondition: thelist is a list of all numbers
    (either floats or ints)""
    for x in thelist:
        x = x+1
    # procedure; no return
```

On The Other Hand

```python
def copy_add_one(thelist):
    """Returns: copy with 1 added to every element
    Precondition: thelist is a list of all numbers
    (either floats or ints)""
    mycopy = []
    # accumulator
    for x in thelist:
        x = x+1
    mycopy.append(x)
    # add to end of accumulator
    return mycopy
```

How Can We Modify A List?

- Never modify loop var!
- This is an infinite loop:
  ```python
  for x in thelist:
      thelist.append(1)
  ```

```
Try this in Python Tutor
to see what happens
```

This is the Motivation for Iterators

- Iterators are objects
  - Contain data like a list
  - But cannot slice them
- Access data with `next()`
  - Function to get next value
  - Keeps going until end
  - Get an error if go to far
- Can convert back & forth
  ```python
  myiter = iter(mylist)
  mylist = list(myiter)
  ```

The Range Iterator

- `range(x)`
  - Creates an iterator
  - Stores [0,1,...,x-1]
  - But not a list!
  - But try `list(range(x))`
- `range(a,b)`
  - Stores [a,...,b-1]
- `range(a,b,n)`
  - Stores [a,a+n,...,b-1]
- Very versatile tool
- Great for processing ints

```
Accumulator
```

<table>
<thead>
<tr>
<th>seq</th>
<th>id1</th>
<th>id1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>id2</td>
<td>id2</td>
</tr>
</tbody>
</table>
```

Modifying the Contents of a List

```python
def add_one(thelist):
    """(Procedure) Adds 1 to every element in the list
    Precondition: thelist is a list of all numbers
    (either floats or ints)""
    size = len(thelist)
    for k in range(size):
        thelist[k] = thelist[k]+1
    # procedure; no return
```

```
WORKS!
```