Loops
Example: Summing the Elements of a List

```python
def sum(thelist):
    """Returns: the sum of all elements in thelist
    Precondition: thelist is a list of all numbers (either floats or ints)"
    pass  # Stub to be implemented
```

Remember our approach: Outline first; then implement
def sum(thelist):
    """Returns: the sum of all elements in thelist
    Precondition: thelist is a list of all numbers (either floats or ints)"
    
    # Create a variable to hold result (start at 0)
    # Add each list element to variable
    # Return the variable
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]
    ...       There is a problem here
    return result
# Print contents of seq

```python
x = seq[0]
print(x)
x = seq[1]
print(x)
...
x = seq[len(seq)-1]
print(x)
```

- **Remember:**
  - Cannot program …

- **Key Concepts**
  - **loop sequence:** `seq`
  - **loop variable:** `x`
  - **body:** `print(x)`
  - Also called **repetend**

## The for-loop:

```python
for x in seq:
    print(x)
```
For Loops: Processing Sequences

The for-loop:

for x in seq:
    print(x)

• loop sequence: seq
• loop variable: x
• body: print(x)

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

```python
def sum(thelist):
    """Returns: the sum of all elements in thelist
    Precondition: thelist is a list of all numbers (either floats or ints)"
    # Create a variable to hold result (start at 0)
    # Add each list element to variable
    # Return the variable
```

Example: Summing the Elements of a List

```python
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
```

- **loop sequence:** `thelist`
- **loop variable:** `x`
- **body:** `result = result + x`
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
```

- **loop sequence:** `thelist`
- **loop variable:** `x`
- **body:** `result = result + x`

Accumulator variable
def num_ints(thelist):
    
    """Returns: the number of ints in thelist
    Precondition: thelist is a list of any mix of types"
    
    # Create a variable to hold result (start at 0)
    # for each element in the list...
    # check if it is an int
    # add 1 if it is
    
    # Return the variable
```python
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 thelist:
        if type(x) == int:
            result = result + 1
    return result
```

For Loops and Conditionals
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
```

9/28/18

For Loops
def add_one(thelist):
    """Adds 1 to every elt
    Pre: thelist is all numb."""
    for x in thelist:
        x = x + 1

add_one(seq):

seq
id4
0
5
1
4
2
7

add_one
thelist
id4
1
def add_one(thelist):
    """Adds 1 to every elt
    Pre: thelist is all numb."""
    for x in thelist:
        x = x + 1

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

add_one(seq):

Increments x in frame
Does not affect folder

Loop back to line 1
def add_one(thelist):
    """Adds 1 to every elt
    Pre: thelist is all numb."""
    for x in thelist:
    x = x+1

add_one(seq):

seq id4
0 5
1 4
2 7

Next element stored in x.
Previous calculation lost.
def add_one(thelist):
    """Adds 1 to every elt
    Pre: thelist is all numb."""
    for x in thelist:
        x = x + 1

add_one(seq):

Loop back to line 1
def add_one(thelist):
    """Adds 1 to every elt
    Pre: thelist is all numb."""
    for x in thelist:
        x = x + 1

add_one(seq):

Next element stored in x.
Previous calculation lost.
For Loops and Call Frames

```python
def add_one(thelist):
    '''Adds 1 to every elt
    Pre: thelist is all numb.'''
    for x in thelist:
        x = x + 1
```

```python
add_one(seq):
```
For Loops and Call Frames

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

add_one(seq):
```

Loop is `completed`. Nothing new put in x.
For Loops and Call Frames

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

9/28/18

For Loops
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

- Need a second sequence
- How about the *positions*?

  ```python
  thelist = [5, 2, 7, 1]
  thepos = [0, 1, 2, 3]
  
  for x in thepos:
      thelist[x] = x+1
  ```
How Can We Modify A List?

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

- Need a second sequence
- How about the *positions*?

```python
thelist = [5, 2, 7, 1]
thepos = [0, 1, 2, 3]

for x in thepos:
    thelist[x] = x+1
```

Try this in Python Tutor to see what happens.
The Range Function

- **range(x)**
  - Creates an iterable
  - 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

```plaintext
total = 0  # add the squares of ints
# in range 2..200 to total
for x in range(2,201):
    total = total + x*x
```

Accumulator