Lecture 8

For-Loops
Announcements for Today

Assignment 1

• We are still grading it
  ▪ Will finish Sat or Sun
• If you are close…
  ▪ Will get feedback in CMS
  ▪ Fix your assignment
• If you are very wrong…
  ▪ You got an e-mail
  ▪ Set up tutoring session
• FINISH THE SURVEY

Assignment 2

• Posted Monday
  ▪ We need one more lecture
  ▪ Due after Fall Break
  ▪ Do not start until A1 done

Reading

• Read Chapters 8, 10
• Focus on sections we skipped
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
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

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**: $\text{sum}([1,2,3])$ vs. $\text{sum}([4,5,6,7,8,9,10])$

• 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

# Print contents of seq
x = seq[0]
print x
x = seq[1]
print x
...
x = seq[len(seq)-1]
print x

• Remember:
  ▪ We cannot program
  ...

The for-loop:

```python
for x in seq:
    print x
```

• Key Concepts
  ▪ loop sequence: `seq`
  ▪ loop variable: `x`
  ▪ body: `print x`
  ▪ Also called `repetend`
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
```

10/6/15
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

```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
```

Accumulator variable

- loop sequence: thelist
- loop variable: x
- body: result = result + x
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
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
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:
        # procedure; no return
        x = x + 1

    # procedure; no return
```

10/6/15 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):
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
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):
```

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

add_one(seq):

Loop back to line 1
```

```
seq
0
1
2
```

```
add_one
thelist
x
```

```
5
4
7
```

```
id4
```

```
id4
```

```
id4
```

```
id4
```

```
id4
```

10/6/15 For Loops
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):
```

```
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):

Loop is **completed**.
Nothing new put in x.
def add_one(thelist):
    """Adds 1 to every elt
    Pre: thelist is all numb."""
    for x in thelist:
        x = x + 1

def add_one(seq):
    for x in seq:
        x = x + 1

seq = [5, 4, 7]
add_one(seq)

No changes to folder
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
```

Accumulator keeps result from being lost
For Loops: Processing Ranges of Integers

total = 0

# add the squares of ints # in range 2..200 to total

```python
for x in range(2, 201):
    total = total + x * x
```

• For each x in the range 2..200, add x*x to total

The range function:

- `range(x)`: List of ints 0 to x-1
- `range(a, b)`: List of ints a to b-1
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
Important Concept in CS: Doing Things Repeatedly

1. Process each item in a sequence
   - Compute aggregate statistics for a dataset, such as the mean, median, standard deviation, etc.
   - Send everyone in a Facebook group an appointment time

2. Perform \( n \) trials or get \( n \) samples.
   - Draw a triangle six times to make a hexagon
   - Run a protein-folding simulation for \( 10^6 \) time steps

3. Do something an unknown number of times
   - CUAVUUV team, vehicle keeps moving until reached its goal
Important Concept in CS: Doing Things Repeatedly

1. Process each item in a sequence
   - Compute aggregate statistics for a dataset, such as the mean, median, standard deviation, etc.
   - Send everyone in a Facebook group an appointment time

2. Perform $n$ trials or get $n$ samples.
   - Draw a triangle six times to make a hexagon
   - Run a protein-folding simulation for $10^6$ time steps

3. Do something an unknown number of times
   - CUAUV team, vehicle keeps moving until reached its goal

For Loops

```python
for x in sequence:
  process x
```

```python
for x in range(n):
  do next thing
```

Cannot do this yet
Impossible w/ Python for