Lecture 8

Loops
Announcements for Today

Assignment 1

• We are still grading it
  • Will finish Sat at noon
• 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 next Friday
  • 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
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
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

# 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

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

The for-loop:

for x in seq:
  print x
For Loops: Processing Sequences

The for-loop:

```python
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  For 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)"
    result = 0
    for x in thelist:
        result = result + x
    return result
```

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

10/6/15 For 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)"
    result = 0
    for x in thelist:
        result = result + x
    return result
```

10/6/15

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

add_one(seq):

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):
```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
```

```
add_one

thelist

x

1

id4

6

```

```
seq

id4

0

1

2

id4

5

4

7

```

Increments x in **frame**

Does not affect folder
def add_one(thelist):
    
    """Adds 1 to every elt
      Pre: thelist is all numb."""

    for x in thelist:
        x = x+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 back to line 1

seq id4
0 5
1 4
2 7

thelist id4
x 5
For Loops and Call Frames

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

add_one(seq):

1  for x in thelist:
2      x = x + 1

seq  add_one
     id4
1  id4
   5
2  4
3  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

For Loops and Call Frames

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
```

```
seq
```

```
add_one
thelist
x
id4
```

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

add_one(seq):

No changes to folder
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
total = 0
# add the squares of ints
# in range 2..200 to total
total = total + 2*2
total = total + 3*3
...
total = total + 200*200

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

The for-loop:

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

• 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
   - CUAUV team, vehicle keeps moving until reached its goal

For Loops
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 $x$ in sequence:
| process $x$

for $x$ in range($n$):
| do next thing

Cannot do this yet
Impossible w/ Python for
Beyond Sequences: The **while-loop**

```python
while <condition>:
    statement 1
    ...
    statement n
```

- **repetend or body**

- **Relationship to for-loop**
  - Broader notion of “still stuff to do”
  - Must explicitly ensure condition becomes false
  - *You* explicitly manage what changes per iteration

11/10/15
While-Loops
print 'Before while'
count = 0
i = 0
while i < 3:
    print 'Start loop ' + str(i)
    count = count + i
    i = i + 1
    print 'End loop'
print 'After while'

Output:
Before while
Start loop 0
End loop
Start loop 1
End loop
Start loop 2
End loop
After while
**while Versus for**

# process range b..c-1

```python
for k in range(b, c):
    process k
```

** Must remember to increment **

```python
# process range b..c
for k in range(b, c+1):
    process k
```

# process range b..c-1

```python
k = b
while k < c:
    process k
    k = k + 1
```

# process range b..c

```python
k = b
while k <= c:
    process k
    k = k + 1
```
Cases to Use **while**

Great for when you must **modify** the loop variable

```python
# Remove all 3's from list t
i = 0
while i < len(t):
    # no 3's in t[0..i-1]
    if t[i] == 3:
        del t[i]
    else:
        i = i + 1
```

```python
# Remove all 3's from list t
while 3 in t:
    t.remove(3)
```
Cases to Use **while**

Great for when you must **modify** the loop variable

```python
# Remove all 3's from list t
i = 0
while i < len(t):
    # no 3's in t[0..i-1]
    if t[i] == 3:
        del t[i]
    else:
        i += 1

# Remove all 3's from list t
while 3 in t:
    t.remove(3)
```

The stopping condition is not a numerical counter this time. Simplifies code a lot.

Stopping point keeps changing.
Cases to Use while

- Want square root of $c$
  - Make poly $f(x) = x^2 - c$
  - Want root of the poly ($x$ such that $f(x)$ is 0)

- Use **Newton’s Method**
  - $x_0 = \text{GUESS } (c/2??)$
  - $x_{n+1} = x_n - f(x_n)/f'(x_n)$
    - $= x_n - (x_nx_n - c)/(2x_n)$
    - $= x_n - x_n/2 + c/2x_n$
    - $= x_n/2 + c/2x_n$
  - Stop when $x_n$ good enough

```python
def sqrt(c):
    """Return: square root of $c$
    Uses Newton’s method
    Pre: $c \geq 0$ (int or float)"
    x = c/2
    # Check for convergence
    while abs(x*x - c) > 1e-6:
        # Get $x_{n+1}$ from $x_n$
        x = x / 2 + c / (2*x)
    return x
```