Lecture 13

For-Loops
Announcements for This Lecture

Reading

- Today: Chapters 8, 10
- Thursday: Chapter 11

Assignments/Lab

- A3 is due on Tomorrow
  - Survey is now posted
  - Will be graded before exam
- A4 after exam and break
  - Longer time to do this one
  - Covers this lecture and next
- No lab next week
  - Current lab due in two weeks
  - But fair game on exam

- Prelim, 10/11 5:15 OR 7:30
  - Material up to Tuesday
  - Study guide is posted
  - Times/rooms by last name
- Review next Wednesday
  - Room/Time are TBA

10/4/18

For Loops
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

```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
    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:
  - 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:

```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 the 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
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

    for x in thelist:
        result = result + x

    return result

• loop sequence: thelist
• loop variable: x
• body: result = result + x
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
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

DOS NOT WORK!
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

def add_one(seq):
    add_one([5, 4, 7, 5])
def add_one(thelist):
    """Adds 1 to every elt
    Pre: thelist is all numb."""
    for x in thelist:
        x = x + 1

add_one(seq):

10/4/18
```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):
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 |
| 1 |
| 2 |

5
4
7
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.
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

add_one(seq):
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)
  ```

- 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] = thelist[x] + 1
  ```

Try this in Python Tutor to see what happens.
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] = thelist[x]+1
```

Try this in Python Tutor to see what happens
This is the Motivation for Iterables

- **Iterables** are objects
  - Contain data like a list
  - But cannot slice them
- Have list-like properties
  - Can use them in a for-loop
  - Can convert them to lists
  - `mylist = list(myiterable)`
- **Example**: Files
  - Use `open()` to create object
  - Makes iterable for reading
Iterables, Lists, and For-Loops

```python
>>> file = open('sample.txt')

>>> list(file)
['This is line 1
', 'This is line 2
']

>>> file = open('sample.txt')

>>> for line in file:
...    print(line)
This is line one
This is line two
```

```
print adds \n in addition to one from file
```
The Range Iterable

- **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

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

Accumulator
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.
   - A4: 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
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.
   - **A4**: 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
   - CUAVU team, vehicle keeps moving until reached its goal

10/4/18

For Loops

**Cannot do this yet**
Impossible w/ Python for