Announcements: Prelim 1

- Rooms:
  - aa200 – jjm200 Baker Laboratory 200
  - jjm201 – sge200 Rockefeller 201
  - sge201 – zz200 Rockefeller 203
- covers material up through today
- no assert, try-except
- What to study: A1, A2, Labs 1-6, old exam questions:
  - Fall 2016, 2015, 2014 call-frame/diagram questions need to be converted to our notation.
- Prelim will probably be closer in style to Spring 2013-2014 than more recent exams

Prelim 1: Things that are not “fair game”

- Prelim 1 fall 2016: ignore 3b (too lecture-dependent)
- Prelim 1 spring 2016: ignore 1, 3, 6.
  - 4 is OK if you ignore the "if name == ..." line, and just assume all that stuff is script code to be run
- Prelim 1 fall 2015: ignore 4(a) – solutions have typos
  - 4(c) not fair game (asserts)
- Prelim 1 spring 2015: ignore 2(b), 3(b), 5
  - For 1(b), imagine that variable s contains some arbitrary, unknown string (we didn’t formally cover raw_input)
- Prelim 1 fall 2014: ignore 2(c), 4(a)
- Prelim 1 spring 2013: question 6: change cumunittest2 to cornelltest

More Announcements

- A2: due today. Solutions released Thursday.
- Lab 6: due in two weeks
  - Tuesday 3/14 labs: open office hours
  - Wednesday 3/15 labs: cancelled
- Thursday 3/9: optional in-class review session
- Tuesday 3/14: no lecture; office hours instead
  - Olin 155 during class times, Carpenter in between
- A3: released sometime after Prelim 1

The Map Function

- map(function, list)
  - Function has to have exactly 1 parameter
  - Otherwise, get an error
  - Returns a new list
  - Calls the function f once for each item

The Filter Function

- filter(Boolean_function, list)
  - Function must:
    - have exactly 1 parameter
    - return a Boolean
  - Returns a new list
  - Returns elements of list for which Boolean_function, returns True

For Loops

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

For Loops and Conditionals

```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
```

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
```

For Loops and Call Frames

```python
def add_one(thelist):
    """Adds 1 to every element
    Pre: thelist is all 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 = []
    for x in thelist:
        x = x + 1
        mycopy.append(x)
    return mycopy
```

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
```

Accumulator keeps result from being lost

WORKS!