CS 1110

Prelim 1 Review
Spring 2014

Slides by Walker White, Lillian Lee, Steve Marschner
Announcements

Prelim is tonight!!!
7:30–9:00 PM
Baker Laboratory 200

Prelim study tips
Understand A1 and A2; understand labs 1–6
Re-work examples from lecture
Practice old prelims (for style, not content!)
Work some problems from the book
Practice coding on codingbat.com
Exam Info

• Prelim 1: 7:30–9:00PM, Tuesday, March 10
  ▪ Location: Baker Laboratory 200

• To help you study:
  ▪ Old prelims, review slides are online
  ▪ Solutions to Assignment 2 are online

• Arrive early! Helps reduce stress

• Grades will be released as soon as practical
  ▪ CMS will let you know
  ▪ Hopefully by drop deadline
What is on the Exam?

• Five Topics (+2pts for name, NetID, lab):
  ▪ String manipulation (A1, Lab 2)
  ▪ Call frames and the call stack (A2)
  ▪ Functions on mutable objects (Lab 3&5)
  ▪ Testing and debugging (A1, Lab 3)
  ▪ Operations on lists, simple iteration (Labs 5&6)

• Tip: always re-read the specs of functions after you finish a problem!
def make_netid(name, n):
    """]Returns a netid for name with suffix n

Netid is either two letters and a number (if the student has no
middle name) or three letters and a number (if the student has
a middle name). Letters in netid are lowercase.

Example: make_netid('Walker McMillan White', 2) is 'wmw2'
Example: make_netid('Walker White', 4) is 'ww4'

Precondition: name is a string either with format '<first-name>
<last-name>' or '<first-name> <middle-name> <last-name>';
names are separated by spaces. n > 0 is an int."""
## Useful String Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>s.index(s1)</td>
<td>Returns first position of s1 in s; error if not there</td>
</tr>
<tr>
<td>s.count(s1)</td>
<td>Returns number of occurrences of s1 in s</td>
</tr>
<tr>
<td>s.lower()</td>
<td>Returns copy of s with all letters lower case</td>
</tr>
<tr>
<td>s.upper()</td>
<td>Returns copy of s with all letters upper case</td>
</tr>
<tr>
<td>s.strip()</td>
<td>Returns copy of s with whitespace removed</td>
</tr>
</tbody>
</table>

- We will give you any methods you need
- But you must know how to slice strings!
def make_netid(name,n):

    """Returns a netid for name with suffix n."""

    name = name.lower()  # switch to lower case
    fpos = name.find(' ')  # find first space
    first = name[:fpos]
    last = name[fpos+1:]
    mpos = last.find(' ')  # see if there is another space

    if mpos == -1:
        return first[0]+last[0]+`n`  # remember, n is not a string
    else:
        middle = last[:mpos]
        last = last[mpos+1:]
        return first[0]+middle[0]+last[0]+`n`
Frames Example

- Given functions to right
  - Function fname() is not important for problem
  - Use the numbers given
- Execute the call: `lname_first('John Doe')`
- Draw entire call stack when helper function `lname` completes line 1
  - Draw nothing else

```python
def lname_first(s):
    
    """Precondition: s in the form <first-name> <last-name>""

    1 first = fname(s)
    2 last = lname(s)
    3 return last + ',' + first

def lname(s):
    
    """Prec: see last_name_first""

    1 end = s.find(' ')
    2 return s[end+1:]
Call Stack Example: `lname_first('John Doe')`

```python
def lname_first(s):
    """Precondition: s in the form
    <first-name> <last-name>""
    first = fname(s)
    last = lname(s)
    return last + ',' + first

def lname(s):
    """Prec: see last_name_first""
    end = s.find(' ')  # Highlighted line
    return s[end+1:]
```

Prelim 1 Review
Call Stack Example: \texttt{lname_first('John Doe')}

\begin{tabular}{|c|c|}
\hline
\texttt{lname_first:} & \times \texttt{2} \\
\hline
\texttt{s} & \texttt{'John Doe'} \\
\hline
\texttt{first} & \texttt{'John'} \\
\hline
\texttt{last} & \phantom{123} \\
\hline
\end{tabular}

\begin{tabular}{|c|c|}
\hline
\texttt{lname:} & \times \texttt{2} \\
\hline
\texttt{s} & \texttt{'John Doe'} \\
\hline
\texttt{end} & \texttt{4} \\
\hline
\end{tabular}

\begin{align*}
\texttt{def} \ & \texttt{lname_first(s):} \\
& \text{Omitting this is okay.} \\
& \text{Line 2 is not complete.} \\
& \text{\texttt{Prec: see last_name_first}} \\
& \text{Line 1 is \textbf{complete}.} \\
& \text{Counter is next line.} \\
\end{align*}

\begin{align*}
& \texttt{first = \texttt{fname(s)}} \\
& \texttt{last = \texttt{lname(s)}} \\
& \texttt{return last + first} \\
& \texttt{end = \texttt{s.find(' ')}} \\
& \texttt{return s[end+1:]} \\
\end{align*}
Example with a Mutable Object

```python
def shift(p):
    """Shift coords left
    Precondition: p a point"
    temp = p.x
    p.x = p.y
    p.y = p.z
    p.z = temp

>>> p = Point(1.0,2.0,3.0)
>>> shift(p)
```

Function Call
Example with a Mutable Object

```python
def shift(p):
    """Shift coords left
    Precondition: p a point""
    temp = p.x
    p.x = p.y
    p.y = p.z
    p.z = temp
```

- May get a function on a mutable object
  ```
  >>> p = Point(1.0,2.0,3.0)
  >>> shift(p)
  ```
- You are not expected to come up w/ the “folder”
  - Will provide it for you
  - You just track changes
def f(x, y):
c = 3*x
y[0] = b + c + y[1]
c = 0
def g(a, b):
f(2, a)
return b + a[0]
b = 4
c = 5
x = g([b, c], 7)
Objects: example with colors

- Type: RGB in module colormodel
  - Constructor call: colormodel.RGB(r,g,b)
    --- assuming prior line import colormodel, and r, g, b are ints in interval 0..255

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Invariant</th>
</tr>
</thead>
<tbody>
<tr>
<td>red</td>
<td>int, within range 0..255</td>
</tr>
<tr>
<td>green</td>
<td>int, within range 0..255</td>
</tr>
<tr>
<td>blue</td>
<td>int, within range 0..255</td>
</tr>
</tbody>
</table>
Function that Modifies Object

def increase10(rgb):
    """Increase each attribute by 10% (up to 255)
    Precondition: rgb an RGB object"
    pass  # implement me
# store in t the value of rgb's red attribute

# Which of these is correct? What do the others do?

t = colormodel.RED

t = rgb.red()

t = rgb.r

t = rgb.red

t = colormodel.rgb.red
Sample step – answer in bold

# store in t the value of rgb's red attribute
# Which of these is correct? What do the others do?

\[ t = \text{colormodel.RED} \quad \# \text{refers to something in colormodel} \]

\[ t = \text{rgb.red()} \quad \# \text{call to function "in" rgb} \]

\[ t = \text{rgb.r} \quad \# \text{attribute r of rgb, but there's no such attribute} \]

\[ t = \text{rgb.red} \quad \# \text{<obj name>.<attr name> is the way to access} \]

\[ t = \text{colormodel.rgb.red} \quad \# \text{refers to something in rgb in \#colormodel} \]
Should increase10 have return statement?
Should increase10 have return statement?

No; the spec doesn't say so.
Function that Modifies Object

```python
def increase10(rgb):
    """Increase each attribute by 10% (up to 255)"""
    red = rgb.red  # puts red attribute value in local var
    red = 1.1*red  # increase by 10%
    red = int(round(red))  # convert to closest int
    rgb.red = min(255,red)  # cannot go over 255
    # Do the others in one line
    rgb.green = min(255,int(round(1.1*rgb.green)))
    rgb.blue = min(255,int(round(1.1*rgb.blue)))
```

Procedure: no return
def multcap(x):
    """Returns: min of nearest int to x*1.1 and 255.
    Precond: x a number"""
    return min(int(round(x*1.1)), 255)

def increase10(rgb):
    """Increase each attribute by 10% (up to 255)"""
    # alternate solution with massive map
    alist = map(multcap, [rgb.red, rgb.green, rgb.blue])
    rgb.red = alist[0]
    rgb.green = alist[1]
    rgb.blue = alist[2]

    Procedure:
    no return
Code up a test case for increase10 (assume in module reviewp1)

testcolor = colormodel.RGB(10,100,255)
reviewp1.increase10(testcolor)
cunittest2.assert_equals(colormodel.RGB(11,110,255),
    testcolor)

Why not this?
cunittest2.assert_equals(colormodel.RGB(11,110,255),
    reviewp1.increase10(testcolor))

No return value to compare against.
Lab 6 review
def first(x):
    1. print 'Starting first.'
    2. second(x)
    3. print 'Ending first'

def second(x):
    1. print 'Starting second.'
    2. if third(x):
        3. pass
    4. else:
        5. print 'Caught False at second'
        6. print 'Ending second'

def third(x):
    1. print 'Starting third.'
    2. print 'Ending third.'
    3. return x < 1

What is the output of first(2)?
def first(x):
  1. print 'Starting first.'
  2. second(x)
  3. print 'Ending first'

def second(x):
  1. print 'Starting second.'
  2. if third(x):
  3.   pass
  4. else:
  5.   print 'Caught False at second'
  6.   print 'Ending second'

def third(x):
  1. print 'Starting third.'
  2. print 'Ending third.'
  3. return x < 1

What is the output of first(2)?

'Starting first.'
'Starting second.'
'Starting third.'
'Caught False at second'
'Ending second'
'Ending first'
def first(x):
    1. print 'Starting first.'
    2. second(x)
    3. print 'Ending first'

def second(x):
    1. print 'Starting second.'
    2. if third(x):
        3. pass
    4. else:
        5. print 'Caught False at second'
    6. print 'Ending second'

def third(x):
    1. print 'Starting third.'
    2. print 'Ending third.'
    3. return x < 1

What is the output of first(0)?
Tracing Control Flow

```python
def first(x):
    1. print 'Starting first.'
    2. second(x)
    3. print 'Ending first'

def second(x):
    1. print 'Starting second.'
    2. if third(x):
       3. pass
    4. else:
       5. print 'Caught False at second'
    6. print 'Ending second'

def third(x):
    1. print 'Starting third.'
    2. print 'Ending third.'
    3. return x < 1
```

What is the output of `first(0)`?

'Starting first.'
'Starting second.'
'Starting third.'
'Ending third'
'Ending second'
'Ending first'
• Type: Length in module ell
  ▪ Constructor call: ell.Length(ft,in)
    --- assuming prior line import ell and ft and in are ints, given:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Invariant</th>
</tr>
</thead>
<tbody>
<tr>
<td>feet</td>
<td>int, non-negative, = 12 in</td>
</tr>
<tr>
<td>inches</td>
<td>int, within range 0..11 inclusive</td>
</tr>
</tbody>
</table>

**def** difference(len1,len2):

"""Returns: Difference between len1 and len2
Result is returned in inches
Precondition: len1 and len2 are length objects
len1 is longer than len2"""

pass # implement me
def difference(len1, len2):

    """Returns: Difference between len1 and len2
    Result is returned in inches
    Precondition: len1 and len2 are length objects
    len1 is longer than len2""

    feetdif = (len1.feet - len2.feet) * 12
    inchdif = len1.inches - len2.inches  # may be negative

    return feetdif + inchdif
def pigify(w):
    """Returns: copy of w converted to Pig Latin
    'y' is a vowel if it is not the first letter
    If word begins with a vowel, append 'hay'
    If word starts with 'q', assume followed by 'u';
    move 'qu' to the end, and append 'ay'
    If word begins with a consonant, move all
    consonants up to first vowel (or to end of w, if
    none) to end and add 'ay'
    
    Precondition: w contains only (lowercase) letters, and at least one letter"""
Picking Test Cases

```python
def pigify(w):
    """Returns: copy of w converted to Pig Latin""
    ...
```

• Test Cases (Determined by the rules):
  - yield => ieldyay (y as consonant)
  - byline => ylinebay (y as vowel)
  - are => arehay (Starts with vowel)
  - quiet => ietquay (Starts with qu)
  - ship => ipshay (Starts with consonant(s))
  - bzzz => bzzzay (All consonants)
Looking for inspiration?
"""What most schools don't teach: Learn about a new "superpower" that isn't being taught in 90% of US schools."""

https://www.youtube.com/watch?feature=player_embedded&v=nKJlu9yen5nc