Prelim 1 Review
Spring 2017
Exam Info

- Prelim 1: 7:30–9:00PM, Tuesday, March 14th
  - Baker Lab 200, Rockefeller Hall 201, 203
  - No Electronics, No Notes, Closed book.
  - Bring your Cornell ID
  - Put your Name & NetId on Each Page!!!
What is on the Exam?

• String slicing functions (A1, Lab 3)
• Booleans & Conditionals (Lab 1, Lab 5)
• Testing and debugging (A1, Lab 3)
• Object and Memory Diagramming (A2)
• Working with Objects (Lab 5)
• Lists and For-Loops (Lab 6)
• Terminology

Not a Complete list, but the major Highlights…
What is on the Exam?

- String slicing functions (A1, Lab 3)
- Booleans & Conditionals (Lab 1, Lab 5)
- Testing and debugging (A1, Lab 3)
- Object and Memory Diagramming (A2)
- Working with Objects (Lab 5)
- Lists and For-Loops (Lab 6)
- Terminology
What are Objects?

• An object is like a folder; It contains other variables (Attributes) with values
• Extends the built in Types in Python
• It has a unique ID that identifies it
  ▪ Cannot ever change
  ▪ Has no meaning; only identifies
• Classes provide a “Template”
Working with Objects

• 3 Major things we’ll ask you to do with objects:
  ▪ Access Attributes of an object
  ▪ Create a new object
  ▪ Modify an existing object (objects are mutable)
Example

• Class: Length
  ▪ Constructor function: Length(ft,in)
  ▪ Remember constructor is just a function that gives us back a mutable object of that type
  ▪ Attributes:

<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</td>
</tr>
</tbody>
</table>
def area(len1,len2):
    
    """Returns: Area of a rectangle (float) with sides
    len1 and len2 in square feet

    Parameter len1: the first length
    Parameter len2: the second length
    Precondition: len1, len2 length objects"
    
    pass # implement me
```python
def area(len1, len2):
    """Returns: Area of a rectangle (float) with sides len1 and len2 in square feet

    Parameter len1: the first length
    Parameter len2: the second length
    Precondition: len1, len2 length objects"

    len1_ft = len1.feet + len1.inches/12.0
    len2_ft = len2.feet + len2.inches/12.0
    return len1_ft * len2_ft
```
def area(len1, len2):

    """Returns: Area of a rectangle (float) with sides len1 and len2 in square feet

Parameter len1: the first length
Parameter len2: the second length
Precondition: len1, len2 length objects"

    len1_ft = len1.feet + len1.inches/12.0
    len2_ft = len2.feet + len2.inches/12.0
    return len1_ft * len2_ft

Why divide by 12.0, not 12?
Let’s Diagram this!

```python
# Spec

def area(len1, len2):
    len1_ft = len1.feet + len1.inches / 12.0
    len2_ft = len2.feet + len2.inches / 12.0
    return len1_ft * len2_ft

a1 = Length(1, 6)
a2 = Length(2, 0)
rect_area = area(a1, a2)
```

3/12/17 Prelim 1 Review
def difference(len1, len2):

    """Returns: A length object that is the Difference between len1 and len2
    Parameter len1: the first length
    Precondition: len1 is a length object longer than len2
    Parameter len2: the second length
    Precondition: len2 is a length object shorter than len1"

    pass  # implement me
def difference(len1, len2):
    """spec""
    new_feet = len1.feet - len2.feet
    new_inches = len1.inches - len2.inches
    if new_inches < 0:
        new_feet = new_feet - 1
        new_inches = new_inches + 12
    return Length(new_feet, new_inches)
def difference2(len1, len2):
    """Modifies len1 by subtracting len2 from it

Parameter len1: the first length
Precondition: len1 is a length object longer than len2

Parameter len2: the second length
Precondition: len2 is a length object shorter than len1"

pass # implement me
def difference2(len1, len2):
    """spec""
    new_feet = len1.feet - len2.feet
    new_inches = len1.inches - len2.inches
    if new_inches < 0:
        new_feet = new_feet - 1
        new_inches = new_inches + 12
    len1.feet = new_feet
    len1.inches = new_inches
For Loops

• Syntax:
  
  for item in list:

  <do something>

• Range Function:

  ▪ range(n) returns a list [0, 1, 2, …. n-2, n-1]
  ▪ This list has n elements
  ▪ MUST use for modifying a list, so you can get the indices
Useful List Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>x.index(a)</td>
<td>Returns first position of a in x; error if not there</td>
</tr>
<tr>
<td>x.append(a)</td>
<td>Modify x to add element a to the end</td>
</tr>
<tr>
<td>x.insert(a,k)</td>
<td>Modify x to put a at position k (and move rest to right)</td>
</tr>
<tr>
<td>x.remove(a)</td>
<td>Modify x to remove first occurrence of a</td>
</tr>
<tr>
<td>x.sort()</td>
<td>Modify x so that elements are in sorted order</td>
</tr>
</tbody>
</table>

- We will give you any methods you need.
  - Note: No x.find(a) for lists!
  - But you must know how to slice lists!
def replace(thelist,a,b):
    """Returns: COPY of thelist with all occurrences of a replaced by b
    Example: replace([1,2,3,1], 1, 4) = [4,2,3,4].
    Parameter thelist: list to copy
    Precondition: thelist is a list of ints
    Parameter a: the value to remove
    Precondition: a is an int
    Parameter b: the value to insert
    Precondition: b is an int """
    return [] # Stub return. IMPLEMENT ME
def replace(thelist, a, b):
    
    """**Returns**: COPY of thelist with all occurrences of a replaced by b

    **Example**: replace([1,2,3,1], 1, 4) = [4,2,3,4].""

    result = [] # Accumulator
    for x in thelist:
        if x == a:
            result.append(b)
        else:
            result.append(x)
    return result
def replace(thelist, a, b):
    """Returns: COPY of thelist with all occurrences of a replaced by b
    Example: replace([1,2,3,1], 1, 4) = [4,2,3,4]."""
    result = []  # Accumulator
    for i in range(len(thelist)):
        if thelist[i] == a:
            result.append(b)
        else:
            result.append(thelist[i])
    return result
def replace(thelist, a, b):
    """Returns: COPY of thelist with all occurrences of a replaced by b
    Example: replace([1,2,3,1], 1, 4) = [4,2,3,4]."""
    result = []  # Accumulator
    for i in range(len(thelist)):
        if thelist[i] == a:
            result.append(b)
        else:
            result.append(thelist[i])
    return result

How would you write this function if it was to modify thelist instead?
Good Luck!