SAMS
Programming A/B

Week 4 Lecture – Lists
July 24, 2017

Mark Stehlik
Quiz…

• Lots of zeroes
• Indentation errors
• Returns inside loops that didn't mean to be
• Recap…
  – Run your code before submission; not at very end!
  – Print your function/variable values
• All grades should now be visible to you in autolab
Lists

• Similar to strings, but different
• String – an immutable sequence of characters
• List – a mutable sequence of data values
Representing Lists in Python

We will use a list to represent a collection of data values.

scores = [78, 93, 80, 68, 100, 94, 85]

colors = ['red', 'green', 'blue']

mixed = ['purple', 100, 90.5]

A list is an ordered sequence of values and may contain values of any data type.

In Python lists may be heterogeneous (may contain items of different data types).
Lists

• More examples:
  – Empty list
    • \( a = [ ] \)
    • \( a = \text{list}() \)
  – A way to create multiple duplicate elements
    • \( a2 = [ " " ] \ast 9 \quad \# \text{ how is this different from } s = " " \ast 9?? \)
    • \( arr = [0] \ast 5 \)
Some List Operations

```python
>>> names = ["Al", "Jane", "Jill", "Mark"]

>>> Al in names
error ... Al is not defined

>>> "Al" in names
True

>>> names + names

>>> names
["Al", "Jane", "Jill", "Mark"]

+ does not alter the original list
```
Some List Operations (continued)

>>> a = ["A", "B", "C"]

>>> a += a

Accessing the elements of a list

• Indexing
  
a = [2, 4, 6, 8, 10, 12]
  print(a[0], a[3], a[6])  # a[6] is an index error
  print(a[-1], a[-2])

• Valid indexes (as with strings) are
  -len .. 0 .. len-1

• Slicing, too
  a[1:3] -> [4, 6]
  a[2:] -> [6, 8, 10, 12]
List Functions

• Like strings, lists have a length
  • print(len(a))

• But also other functions
  • max, min, list, sum
  • arr = list(range(10)) produces
    [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
  • list("Mark") produces what?
List membership

• How to test to see if something is in the list

```python
def search(target):
    for i in range(len(list)):
        if (list[i] == target):
            return True
    return False
```

• Or use "in/not in"

```python
def search(target):
    return target in list
```
List membership

• Another way to loop over a list:

```python
def search(target):
    for value in list:  # no index here, just each value from first to last
        if (value == target):
            return True
    return False
```
List functions (and two methods)...

<table>
<thead>
<tr>
<th>Operation</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>x in s</td>
<td>True if an item of s is equal to x, else False</td>
</tr>
<tr>
<td>x not in s</td>
<td>False if an item of s is equal to x, else True</td>
</tr>
<tr>
<td>s + t</td>
<td>the concatenation of s and t</td>
</tr>
<tr>
<td>s * n, n * s</td>
<td>n shallow copies of s concatenated</td>
</tr>
<tr>
<td>s[i]</td>
<td>i\textsuperscript{th} item of s, origin 0</td>
</tr>
<tr>
<td>s[i:j]</td>
<td>slice of s from i to j</td>
</tr>
<tr>
<td>s[i:j:k]</td>
<td>slice of s from i to j with step k</td>
</tr>
<tr>
<td>len(s)</td>
<td>length of s</td>
</tr>
<tr>
<td>min(s)</td>
<td>smallest item of s</td>
</tr>
<tr>
<td>max(s)</td>
<td>largest item of s</td>
</tr>
<tr>
<td>s.index(i)</td>
<td>index of the first occurrence of i in s</td>
</tr>
<tr>
<td>s.count(i)</td>
<td>total number of occurrences of i in s</td>
</tr>
</tbody>
</table>
Lists are mutable!

• Unlike strings, you can alter the contents of a list

    \[
        a = [2, 4, 6, 8, 10, 12] \\
        a[0] = 1 \\
        a[3] = 17
    \]

• You can even alter segments of the list (slices)

    \[
        a[1:3] = [3, 5, 7, 9] \rightarrow [2, 3, 5, 7, 9, 8, 10, 12] \\
    \]

    # Note: the new segment doesn't have to be the same length!
Lists are mutable!

• Another example (what does this do?)
  
  ```python
  for i in range (len(a)):
    a[i] = i
  print(a)
  ```

• Yet another example: replace the elements of a with the first n odd numbers, e.g.
  
  ```python
  a = [2, 4, 6, 8, 10, 12]
  and I want a to be [1, 3, 5, 7, 9, 11]
  write the code to change the values of a…
  (in above loop, replace a[i] = i with a[i] = a[i] – 1)
  ```
Lists aliases...

- Create a list
  
  \[ a = [1, 2, 3] \]

- Assign it to another variable
  
  \[ b = a \]

- The second variable references the same list (b said to be an alias for a)

  \[
  \begin{align*}
  & \text{print}(b) \\
  & b[2] = 17 \\
  & \text{print}(a, b)
  \end{align*}
  \]
Lists aliases and functions…

• You're not going to like this, but function parameters are aliases as well (unlike simple types)
  def double(a):
      for i in range(len(a)):
          a[i] = 2 * a[i]
  a = [1, 2, 3]
  double(a)
  print(a)

• So changes to a list parameter are seen outside the function
List methods (some alter the list)...

<table>
<thead>
<tr>
<th>Operation</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>s[i] = x</code></td>
<td>item $i$ of $s$ is replaced by $x$</td>
</tr>
<tr>
<td><code>s[i:j] = t</code></td>
<td>slice of $s$ from $i$ to $j$ is replaced by the contents of the iterable $t$</td>
</tr>
<tr>
<td><code>del s[i:j]</code></td>
<td>same as $s[i:j] = []$</td>
</tr>
<tr>
<td><code>s[i:j:k] = t</code></td>
<td>the elements of $s[i:j:k]$ are replaced by those of $t$</td>
</tr>
<tr>
<td><code>del s[i:j:k]</code></td>
<td>removes the elements of $s[i:j:k]$ from the list</td>
</tr>
<tr>
<td><code>s.append(x)</code></td>
<td>same as $s[len(s):len(s)] = [x]$</td>
</tr>
<tr>
<td><code>s.extend(x)</code></td>
<td>same as $s[len(s):len(s)] = x`</td>
</tr>
<tr>
<td><code>s.count(x)</code></td>
<td>return number of $i$'s for which $s[i] == x$</td>
</tr>
<tr>
<td><code>s.index(x[, i[, j]])</code></td>
<td>return smallest $k$ such that $s[k] == x$ and $i &lt;= k &lt; j$</td>
</tr>
<tr>
<td><code>s.insert(i, x)</code></td>
<td>same as $s[i:i] = [x]$</td>
</tr>
<tr>
<td><code>s.pop([i])</code></td>
<td>same as $x = s[i]$; del s[i]; return $x$</td>
</tr>
<tr>
<td><code>s.remove(x)</code></td>
<td>same as del $s[s.index(x)]$</td>
</tr>
<tr>
<td><code>s.reverse()</code></td>
<td>reverses the items of $s$ in place</td>
</tr>
<tr>
<td><code>s.sort([key[, reverse]])</code></td>
<td>sort the items of $s$ in place</td>
</tr>
</tbody>
</table>