CompSci 101

Introduction to Computer Science

<table>
<thead>
<tr>
<th>pop</th>
<th>none</th>
<th>hybrid</th>
</tr>
</thead>
<tbody>
<tr>
<td>pop</td>
<td>position</td>
<td>hybrid</td>
</tr>
<tr>
<td>sort</td>
<td>none</td>
<td>mutator</td>
</tr>
</tbody>
</table>

February 19, 2015

Prof. Rodger
Announcements

• Class Feb 17 was snowed out
• RQ10 and reading up later today
• APT 4 is due Tuesday
• There is lab this week
• No assignment out yet
• Do not discuss Exam1 with anyone until it is handed back next week
Georges Harik and Noam Shazeer created the underlying data that led to AdSense

Harik and Shazeer spent years analyzing data on webpages, trying to understand clusters of words and how they worked together. The data they gather wound up being used by Google for its AdSense product, which analyzed webpages for words, and then stuck ads on them.
Creating a list

• Given a list of numbers, create a second list of every number squared.

```python
nums = [8, 3, 5, 4, 1]
sqnums = []
for v in nums:
    sqnums.append(v*v)
print(sqnums)
```

```
[64, 9, 25, 16, 1]
```
More on List operations

• See list operations on next page

• Mutator vs hybrid vs return
  – Mutator changes the list (no return value)
  – Hybrid changes list and returns value
  – Return – returns value, no change to list
<table>
<thead>
<tr>
<th>Method</th>
<th>Parameters</th>
<th>Result</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>append</td>
<td>item</td>
<td>mutator</td>
<td>Adds a new item to the end of a list</td>
</tr>
<tr>
<td>insert</td>
<td>position, item</td>
<td>mutator</td>
<td>Inserts a new item at the position given</td>
</tr>
<tr>
<td>pop</td>
<td>none</td>
<td>hybrid</td>
<td>Removes and returns the last item</td>
</tr>
<tr>
<td>pop</td>
<td>position</td>
<td>hybrid</td>
<td>Removes and returns the item at position</td>
</tr>
<tr>
<td>sort</td>
<td>none</td>
<td>mutator</td>
<td>Modifies a list to be sorted</td>
</tr>
<tr>
<td>reverse</td>
<td>none</td>
<td>mutator</td>
<td>Modifies a list to be in reverse order</td>
</tr>
<tr>
<td>index</td>
<td>item</td>
<td>return idx</td>
<td>Returns the position of first occurrence of item</td>
</tr>
<tr>
<td>count</td>
<td>item</td>
<td>return ct</td>
<td>Returns the number of occurrences of item</td>
</tr>
<tr>
<td>remove</td>
<td>item</td>
<td>mutator</td>
<td>Removes the first occurrence of item</td>
</tr>
</tbody>
</table>
Problem

• Remove all negative numbers from list

• Two ways
  1) return a new list with all negative numbers removed
  2) Modify a list to remove negative numbers
def removeNegatives(numberlist):
    # return a new list without negatives
    answer = []
    for num in numberlist:
        if num >= 0:
            answer.append(num)
    return answer

somenums = [3, -1, 8, -5, -2, 6, 7]
nonegs = removeNegatives(somenums)
def removeNegatives2(numberlist):
    # remove the negative numbers
    # from the list
    for x in range(len(numberlist)):
        value = numberlist[x]
        if value < 0:
            numberlist.pop(x)

somenums = [3, -1, 8, -5, -2, 6, 7]
removeNegatives2(somenums)
def removeNegatives3(numberlist):
    # remove the negative numbers
    # from the list
    pos = 0;
    while (pos < len(numberlist)):
        value = numberlist[pos]
        if value < 0:
            numberlist.pop(pos)
        pos = pos + 1

somenums = [3, -1, 8, -5, -2, 6, 7]
removeNegatives3(somenums)
Richard Stallman

- MacArthur Fellowship (Genious grant)
- ACM Grace Murray Hopper award
- Started GNU – Free Software Foundation (1983)
  - GNU Compiler Collection
  - GNU Emacs
List Comprehension

• Take advantage of patterns, make a new list based on per element calculations of another list

• Format:

  [<expression with variable> for <variable> in <old list>]

• Example:

  nums = [8, 3, 5, 4, 1]
  sqnums = [v*v for v in nums]
Examples of List Comprehensions

[v for v in nums]
[2 for v in nums]
[v*2 for v in nums]
Creating a list with just the even numbers

nums = [8, 3, 5, 4, 1]
evennums = []
for v in nums:
    if v % 2 == 0:
        evennums.append(v)
print evennums

[8, 4]
List Comprehension with Filtering

- Create list and use “if” to filter out elements to the list
- Format:
  - \[<\text{expression with variable}> \text{ for } <\text{variable}> \text{ in } <\text{old list}> \text{ if } <\text{filter with variable}> \ ]

- Example: \[\text{nums} = [8, 3, 5, 4, 1]\]
  
  \[\text{evenums} = [v \text{ for } v \text{ in } \text{nums} \text{ if } v \% 2 == 0]\]
• What is the list for the following:
  1) \([j+1 \text{ for } j \text{ in } \text{range}(20) \text{ if } (j\%3) == 0]\]
  2) \([i*2 \text{ for } i \text{ in } [j+1 \text{ for } j \text{ in } \text{range}(20) \text{ if } (j\%3) == 0] \text{ if } i*i > 19]\]