CSC108H Lecture 14

Dan Zingaro

October 12, 2012

Lists

A list is a sequence, like a string. However, compared to strings:

- List elements can be of any type; string elements are single characters
- Lists are mutable; strings are immutable
- Lists can be heterogeneous (elements within the same list can have different types)
- Lists can be nested in other lists
- Some methods are available to both lists and strings, but many differ

List Functions

- `len`: length of a list (i.e. number of elements)
- `min`, `max`: minimum or maximum of list
- `sum`: sum of the elements in the list
- (Except for `sum`, these work on strings, too.)

ConcepTest

```
lst = ['abc', 'def', 'ghi']
lst[1] = 'wxyz'
print(len(lst))
```

What is the output of this code?

- A. 3
- B. 9
- C. 10
- D. 4
- E. No output; there is an error in the second line

List Methods

- Use `dir(list)` to get a list of list methods
- Use `help(list.x)` for help on method name `x`

- `L.append(element)`: add `element` to the end of `L`
- `L.extend(lst)`: add all elements of `lst` to the end of `L`
- `L.insert(index, element)`: insert `element` at index `index` of `L`, pushing later elements forward
- `L.pop()`: remove and return the element at the end of `L`
- `L.pop(index)`: remove and return the element at index `index` of `L`
- `L.remove(element)`: remove first occurrence of `element` from `L`
- `L.sort()`: sort the elements of `L`

ConcepTest

What is the value of a after this code runs?

```
a = [2, 4, 6, 8]
a.remove(4)
a.pop(2)
```

- A. `[2, 4]`
- B. `[6, 8]`
- C. `[2, 6]`
- D. `[2, 8]`
- E. Nothing; the code produces an error

ConcepTest

What is the value of a after this code runs?

```
a = [2, 4, 6, 8]
a.pop(2)
a.remove(4)
```

- A. `[2, 4]`
- B. `[6, 8]`
- C. `[2, 6]`
- D. `[2, 8]`
- E. Nothing; the code produces an error

Looping over Lists

The syntax for looping through all elements in a list is the same as for looping through a string:

```
for element in lst:
  <do something, using element>
```

- The loop variable in the example above is `element`
- `element` "steps through" each element in the list `lst`

Example: Length of Strings

- Write a function that takes a list of strings, and prints out the length of each string in the list
- e.g. if the list is `['abc', 'q', '']`, the output would be as follows

```
3
1
0
```

Strings Example: Word Count

`'this is  a sample  string'`

Write a function that takes a string and returns the number of words in that string. Words may be separated by multiple spaces! The string is guaranteed not to start or end with a space. Do not use `split`.