

Lists

Ruth Anderson

CSE 140

University of Washington

What is a list?

- A list is an ordered sequence of values

3	1	4	4	5	9
---	---	---	---	---	---

"Four"	"score"	"and"	"seven"	"years"
--------	---------	-------	---------	---------

- What operations should a list support efficiently and conveniently?
 - Creation
 - Querying
 - Modification

List creation

```
a = [ 3, 1, 2*2, 1, 10/2, 10-1 ]
```

3	1	4	1	5	9
---	---	---	---	---	---

```
b = [ 5, 3, 'hi' ]
```

```
c = [ 4, 'a', a ]
```

List querying

- Extracting part of the list:
 - Single element: `mylist[index]`
 - Sublist (“slicing”): `mylist[startidx : endidx]`
- Find/lookup in a list
 - `elt in mylist`
 - Evaluates to a boolean value
 - `mylist.index(x)`
 - Return the int index in the list of the first item whose value is x. It is an error if there is no such item.
 - `list.count(x)`
 - Return the number of times x appears in the list.

List mutation

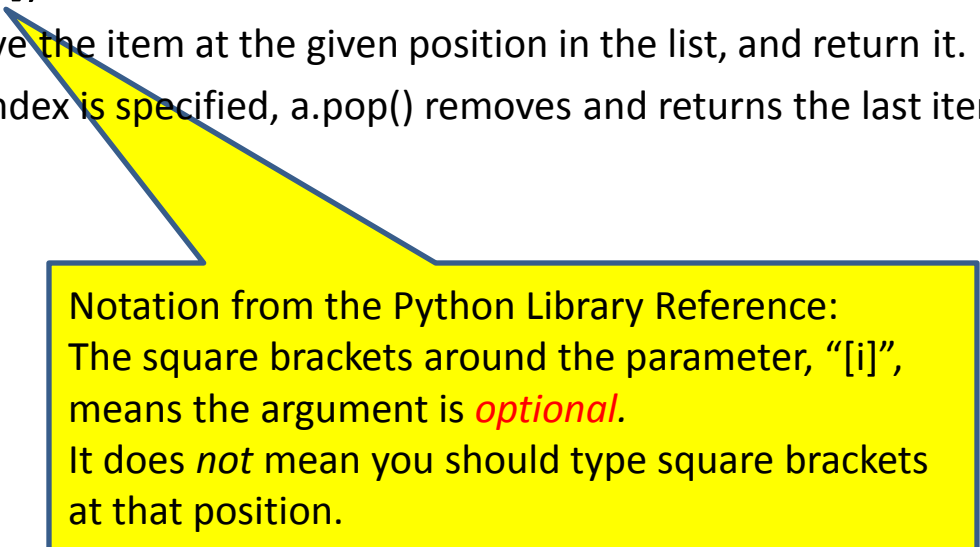
- Insertion
- Removal
- Replacement
- Rearrangement

List insertion

- `mylist.append(x)`
 - Extend the list by inserting `x` at the end
- `mylist.extend(L)`
 - Extend the list by appending all the items in the argument list
- `mylist.insert(i, x)`
 - Insert an item before the a given position.
 - `a.insert(0, x)` inserts at the front of the list
 - `a.insert(len(a), x)` is equivalent to `a.append(x)`

List removal

- `list.remove(x)`
 - Remove the first item from the list whose value is `x`
 - It is an error if there is no such item
- `list.pop([i])`
 - Remove the item at the given position in the list, and return it.
 - If no index is specified, `a.pop()` removes and returns the last item in the list.



Notation from the Python Library Reference:
The square brackets around the parameter, “[i]”,
means the argument is *optional*.
It does *not* mean you should type square brackets
at that position.

List replacement

- `mylist[index] = newvalue`
- `mylist[start : end] = newsublist`
 - Can change the length of the list
 - `mylist[start : end] = []` removes multiple elements
 - `a[len(a):] = L` is equivalent to `a.extend(L)`

List rearrangement

- `list.sort()`
 - Sort the items of the list, in place.
 - “in place” means by modifying the original list, not by creating a new list.
- `list.reverse()`
 - Reverse the elements of the list, in place.

How to evaluate a list expression

There are two new forms of expression:

- `[a, b, c, d]` list **creation**
 - To evaluate:
 - evaluate each element to a value, from left to right
 - make a list of the values
 - The elements can be arbitrary values, including lists
 - `["a", 3, 3.14*r*r, fahr_to_cent(-40), [3+4, 5*6]]`

Same tokens “[]”
with two *distinct*
meanings

List
expression

- `a[b]` list **indexing** or dereferencing

Index
expression

- To evaluate:
 - evaluate the list expression to a value
 - evaluate the index expression to a value
 - if the list value is not a list, execution terminates with an error
 - if the element is not in range (not a valid index), execution terminates with an error
 - the value is the given element of the list value (counting from **zero**)

List expression examples

What does this mean (or is it an error)?

```
["four", "score", "and", "seven", "years"][2]
```

```
["four", "score", "and", "seven", "years"][0,2,3]
```

```
["four", "score", "and", "seven", "years"][[0,2,3]]
```

```
["four", "score", "and", "seven", "years"][[0,2,3][1]]
```

Exercise: list lookup

```
def index(somelist, value):  
    """Return the position of the first  
    occurrence of the element value in the  
    list somelist.  
    Return None if value does not appear in  
    somelist."""
```

Examples:

```
gettysburg = ["four", "score", "and", "seven",  
             "years", "ago"]
```

```
index(gettysburg, "and") => 2
```

```
index(gettysburg, "years") => 4
```

Fact: `mylist[index(mylist, x)] == x`

Exercise: list lookup

```
def index(somelist, value):  
    """Return the position of the first  
    occurrence of the element value in the  
    list somelist.  
    Return None if value does not appear in  
    somelist."""  
    i = 0  
    for c in somelist:  
        if c == value:  
            return i  
        i = i + 1  
    return None
```

Exercise: convert units

```
ctemps = [-40, 0, 20, 37, 100]
# Goal: set ftemps to [-40, 32, 68, 98.6, 212]
# Assume a function celsius_to_fahrenheit exists

ftemps = []
for c in ctemps:
    f = celsius_to_fahrenheit(c)
    ftemps.append(f)
```

List slicing

`mylist[startindex : endindex]` evaluates to a **sublist** of the original list

- `mylist[index]` evaluates to an **element** of the original list
- Arguments are like those to the **range** function
 - `mylist[start : end : step]`
 - start index is inclusive, end index is exclusive
 - *All 3 indices are optional*
- Can assign to a slice: `mylist[s : e] = yourlist`

List slicing examples

```
test_list = ['e0', 'e1', 'e2', 'e3', 'e4', 'e5', 'e6']
```

From e2 to the end of the list:

```
test_list[2:]
```

From beginning up to (but not including) e5:

```
test_list[:5]
```

Last element:

```
test_list[-1]
```

Last four elements:

```
test_list[-4:]
```

Everything except last three elements:

```
test_list[:-3]
```

Reverse the list:

```
test_list[::-1]
```

Get a copy of the whole list:

```
test_list[:]
```