

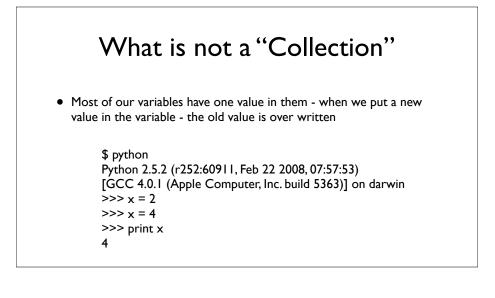
A List is a kind of Collection



- A collection allows us to put many values in a single "variable"
- A collection is nice because we can carry all many values around in one convenient package.

friends = ['Joseph', 'Glenn', 'Sally']

carryon = ['socks', 'shirt', 'perfume']



List Constants

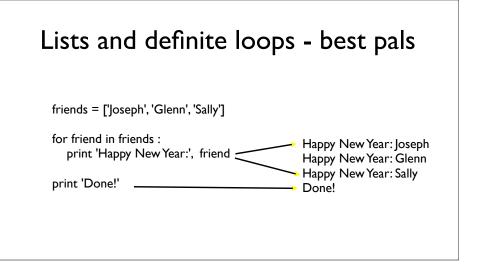
- List constants are surrounded by square brakets and the elements in the list are separated by commas.
- A list element can be any Python object even another list
- A list can be empty

>>> print [1, 24, 76]
[1, 24, 76]
>>> print ['red', 'yellow', 'blue']
['red', 'yellow', 'blue']
>>> print ['red', 24, 98.6]
['red', 24, 98.59999999999999999999999]
>>> print [1, [5, 6], 7]
[1, [5, 6], 7]
>>> print []
[]

We already use lists!

Г

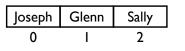
	5
for i in [5, 4, 3, 2, 1] :	4
print i	3
	2
print 'Blastoff!'	I
•	Blastoff!





Looking Inside Lists

• Just like strings, we can get at any single element in a list using an index specified in square brackets



>>> friends = ['Joseph', 'Glenn', 'Sally']
>>> print friends[1]
Glenn
>>>

Lists are Mutable

- Strings are "immutable" we cannot change the contents of a string - we must make a new string to make any change
- Lists are "mutable" we can change an element of a list using the index operator

>>> fruit = 'Banana'
>>> fruit[0] = 'b'
Traceback
TypeError: 'str' object does not
support item assignment
>>> x = fruit.lower()
>>> print x
bannna
>>> lotto = [2, 14, 26, 41, 63]
>>> print lotto
[2, 14, 26, 41, 63]
>>> lotto[2] = 28
>>> print lotto
[2, 14, 28, 41, 63]

How Long is a List?

- The len() function takes a list as a parameter and returns the number of *elements* in the list
- Actually len() tells us the number of elements of *any* set or sequence (i.e. such as a string...)

>>> greet = 'Hello Bob'
>>> print len(greet)
9
>>> x = [1, 2, 'joe', 99]
>>> print len(x)
4
>>>

Using the range function

- The range function returns a list of numbers that range from zero to one less than the parameter
- We can construct an index loop using for and an integer iterator

>>> print range(4)
[0, 1, 2, 3]
>>> friends = ['Joseph', 'Glenn', 'Sally']
>>> print len(friends)
3

>>> print range(len(friends)) [0, 1, 2] >>>

A tale of two loops...

friends = ['Joseph', 'Glenn', 'Sally']

for friend in friends : print 'Happy New Year:', friend

for i in range(len(friends)) :
 friend = friends[i]
 print 'Happy New Year:', friend

>>> friends = ['Joseph', 'Glenn', 'Sally']
>>> print len(friends)
3
>>> print range(len(friends))
[0, 1, 2]
>>>

Happy New Year: Joseph Happy New Year: Glenn Happy New Year: Sally

Concatenating lis	sts using +
• We can create a new list by adding two exsiting lists together	>>> a = [1, 2, 3] >>> b = [4, 5, 6] >>> c = a + b >>> print c [1, 2, 3, 4, 5, 6] >>> print a [1, 2, 3]

Lists can be sliced using :

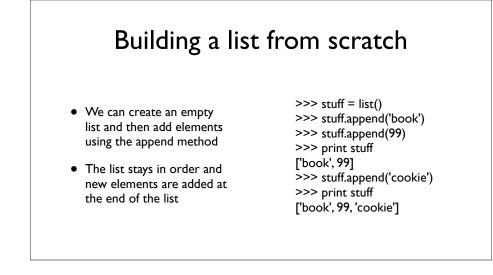
>>> t = [9,41,12,3,74,15] >>> t[1:3] [41,12] >>> t[:4] [9,41,12,3] >>> t[3:] [3,74,15] >>> t[:] [9,41,12,3,74,15]

Remember: Just like in strings, the second number is "up to but not including"

List Methods

>>> x = list()
>>> type(x)
<type 'list'>
>>> dir(x)
['append', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse', 'sort']
>>>

http://docs.python.org/tutorial/datastructures.html



Is Something in a List?

- Python provides two operators that let you check if an item is in a list
- These are logical operators that return True or False
- They do not modify the list

>>> some = [1, 9, 21, 10, 16] >>> 9 in some True >>> 15 in some False >>> 20 not in some True >>>

A List is an Ordered Sequence

- A list can hold many items and keeps those items in the order until we do something to change the order
- A list can be sorted (i.e. change its order)
- The sort method (unlike in strings) means "sort yourself"

>>> friends = ['Joseph', 'Glenn', 'Sally']
>>> friends.sort()
>>> print friends
['Glenn', 'Joseph', 'Sally']
>>> print friends[1]
Joseph
>>>

Built in Functions and Lists

- There are a number of functions built into Python that take lists as parameters
- Remember the loops we built? These are much simpler

>>> nums = [3, 41, 12, 9, 74, 15]
>>> print len(nums)
6
>>> print max(nums)
74
>>> print min(nums)
3
>>> print sum(nums)
154
>>> print sum(nums)/len(nums)
25

total = 0
count = 0
while True :
 inp = raw_input('Enter a number: ')
 if inp == 'done' : break
 value = float(inp)
 total = total + value
 count = count + 1

average = total / count print 'Average:', average

Averaging with a list

Enter a number: 3 Enter a number: 9 Enter a number: 5 Enter a number: done Average: 5.66666666667

numlist = list() while True : inp = raw_input('Enter a number: ') if inp == 'done' : break value = float(inp) numlist.append(value) average = sum(numlist) / len(numlist)

print 'Average:', average

http://docs.python.org/lib/built-in-funcs.html

Best Friends: Strings and Lists

>>> abc = 'With three words'
>>> stuff = abc.split()
>>> print stuff
['With', 'three', 'words']
>>> print len(stuff)
3
>>> print stuff[0]
With

>>> print stuff ['With', 'three', 'words'] >>> for w in stuff : ... print w ... With three words

>>>

Split breaks a string into parts produces a list of strings. We think of these as words. We can access a particular word or loop through all the words.

<pre>>>> line = 'A lot >>> etc = line.split()</pre>	of spaces'
<pre>>>> print etc ['A', 'lot', 'of', 'spaces'] >>></pre>	
<pre>>>> line = 'first;second;third' >>> thing = line.split() >>> print thing ['first;second;third'] >>> print len(thing)</pre>	When you do not specify a delimiter, multiple spaces are treated like "one" delimiter.
<pre>1 >>> thing = line.split(';') >>> print thing ['first', 'second', 'third'] >>> print len(thing) 3 >>></pre>	You can specify what delimiter character to use in the splitting.

From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008

fhand = open('mbox-short.txt') for line in fhand:	Sat
line = line.rstrip()	Fri Fri
<pre>if not line.startswith('From ') : continue words = line.split()</pre>	Fri
print words[2]	•••

>>> line = 'From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008'
>>> words = line.split()
>>> print words
['From', 'stephen.marquard@uct.ac.za', 'Sat', 'Jan', '5', '09:14:16', '2008']
>>>

The Double Split Pattern • Sometimes we split a line one way and then grab one of the pieces of the line and split that piece again From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008 words = line.split() email = words[1] pieces = email.split('@') ['stephen.marquard', 'uct.ac.za'] print pieces[1] 'uct.ac.za'

The Do	ouble Split Pattern
• Sometimes we split a line one way and then grab one of the pieces of the line and split that piece again	
From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008	
words = line.split() email = words[1] pieces = email.split('@') print pieces[1]	stephen.marquard@uct.ac.za
	['stephen.marquard', 'uct.ac.za']
	'uct.ac.za'

The Double Split Pattern

• Sometimes we split a line one way and then grab one of the pieces of the line and split that piece again

From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008

words = line.split() email = words[1] pieces = email.split('@') print pieces[1]

stephen.marquard@uct.ac.za

['stephen.marguard', 'uct.ac.za']

The Double Split Pattern

• Sometimes we split a line one way and then grab one of the pieces of the line and split that piece again

From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008

words = line.split() email = words[1] pieces = email.split('@') print pieces[1]

stephen.marquard@uct.ac.za

['stephen.marguard', 'uct.ac.za']

'uct.ac.za'

List Summary • Concept of a collection • List methods: append, remove • Lists and definite loops • Sorting lists Indexing and lookup • Splitting strings into lists of

- List mutability
- Functions: len, min, max, sum
- Slicing lists

- words
- Using split to parse strings