Python - Lists

A List is a kind of Collection

-- A collection allows many values in a single "variable"

-- A collection is nice because many values can be carried around in one convenient package

> friends = ['Joseph', 'Glenn', 'Sally'] carryon = ['socks', 'shirt', 'perfume']

What is not a "Collection"

-- Most of the variables discussed have one value in them - when a new value is assigned to a variable - the old value is over written

> >>> <u>×</u> = 2 >>> x = 4 >>> print x 4

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, 98.599999999999999 >>> print [1, [5, 6], 7] [1, [5, 6], 7] >>> print []

We already use lists!

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

5

4 3

3
2

Blastoff!

Lists and definite loops - best pals

friends = ['Joseph', 'Glenn', 'Sally'] for friend in friends : print 'Happy New Year:', friend print 'Done!'

Happy New Year: Joseph Happy New Year: Glenn Happy New Year: Sally Done!



Looking Inside Lists

-- Just like strings, any single element in a list can be acquired using an index specified in square brackets



>>> print friends[1] Glenn

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

Lists are Mutable

-- Strings are "immutable" *cannot* change the contents of a string - 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 >> x = fruit.lower() >>> print x banana >>> lotto[2] = 28 >>> print lotto [2, 14, 28, 41, 63]

- TypeError: 'str' object does not support item assignment
- >>> lotto = [2, 14, 26, 41, 63] >>> print lotto[2, 14, 26, 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() determines the number of elements of *any* set or sequence (i.e. such as a string...)

9 >>> print len(x) 4 >>>

>>> greet = 'Hello Bob' >>> print len(greet)

>>> **x** = [1, 2, 'joe', 99]

Using the range function

-- The range function returns a list of numbers, which range from zero to one less than the parameter

-- Construct an index loop using for and an integer iterator

>>> print range(4) [0, 1, 2, 3] >>> print len(friends) 3 [0, 1, 2] >>>

- >>> friends = ['Joseph', 'Glenn', 'Sall
- >>> print range(len(friends))

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 >>> print len(friends) 3 [0, 1, 2]>>>

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



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

>>> print range(len(friends))

Concatenating lists using +

-- Create a new list by adding two existing lists together

- >>> print c

- [1, 2, 3]

>>> a = [1, 2, 3] >>> **b** = [4, 5, 6] >>> c = a + b [1, 2, 3, 4, 5, 6] >>> print a

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']

>>>

Building a list from scratch

-- Create an empty list and add elements using the append method

-- The list stays in order and new elements are added at the end of the list >>> stuff = list()
>>> stuff.append('book')
>>> stuff.append(99)
>>> print stuff
['book', 99]
>>> stuff.append('cookie')
>>> print stuff
['book', 99, 'cookie']

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

>>> 9 in some True >>> 15 in some False >>> 20 not in some True >>>

>>> some = [1, 9, 21, 10, 16]

A List is an Ordered Sequence

-- A list can hold many items and keep 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', 'Sall >>> 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

>>> print len(nums) 6 >>> print max(nums) 74 >>> print min(nums) 3 >>> print sum(nums) 154 25

- >>> nums = [3, 41, 12, 9, 74, 15]
- >> print sum(nums)/len(nums)

```
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
```

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

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

Best Friends: Strings and Lists

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

Split breaks a string into parts produces a list of strings. Access a particular word or loop through all the words.

>>> line = 'A lot
>>> etc = line.split()
>>> print etc
['A', 'lot',

>>>

- >>> line = 'first;second;third'
 >>> thing = line split()
- >>> thing = line.split()
- >>> print thing
- ['first;second;third']
- >>> print len(thing)

1

>>> thing = line.split(';')
>>> print thing
['first', 'second', 'third']
\$>> print len(thing)
>>>

of spaces'

'of', 'spaces']

When you do not specify a delimiter, multiple spaces are treated like "one" delimiter.

You can specify what delimiter character to use in the splitting.