Manipulating Lists
What is a List?

- A linearly ordered set (indexed) of values.
- It can store values of different types.
- Its values can be changed.

```python
>>> myList = ['INF510', 10, 0.1, 'Jeremy']
>>> print myList[0]
INF510
>>> x = myList[1:-1]
>>> print x
['INF510', 10, 0.1]
```
What is a List?

- List are surrounded by square brackets 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.

```python
>>> myList = ['INF510', 10, 0.1, 'Jeremy']
>>> anotherList = ['x', 'y', 'z']
>>> emptyList = []
>>> myList[1] = anotherList
>>> print myList
['INF510', ['x', 'y', 'z'], 0.1, 'Jeremy']
```
Built-In Functions for List

- There are a number of functions built into Python that take lists as parameters

```python
>>> 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
```
Sorting

• Lists can be Sorted

```python
>>> friends = [ 'Joseph', 'Glenn', 'Sally' ]
>>> friends.sort()
>>> print friends
['Glenn', 'Joseph', 'Sally']
>>> print friends[1]
Joseph
```
Strings and Lists: The split operation

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

```python
>>> abc = 'With three words'
>>> stuff = abc.split()
>>> print stuff
['With', 'three', 'words']
>>> print len(stuff)
3
>>> print stuff[0]
With
```
The split Function Delimiter

- When you do not specify a delimiter, multiple spaces are treated like one delimiter
- You can specify the delimiter character in the splitting

```python
>>> line = 'A lot of spaces'
>>> etc = line.split()
>>> print etc
['A', 'lot', 'of', 'spaces']
>>> line = 'first;second;third'
>>> thing = line.split()
>>> print thing
['first;second;third']
>>> print len(thing)
1
>>> thing = line.split(';')
>>> print thing
['first', 'second', 'third']
>>> print len(thing)
3
```
Splitting Twice

• Sometimes we split a line once
• … then use a section and split it again

```python
line = 'From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008'
words = line.split()
email = words[1]
pieces = email.split('@')
host = pieces[1]

stephen.marquard@uct.ac.za
['stephen.marquard', 'uct.ac.za']
'uct.ac.za'
```
List Membership

• 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
>>>
The Range Function and Lists

• 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

```python
>>> print range(4)
[0, 1, 2, 3]
>>> friends = ['Joseph', 'Glenn', 'Sally']
>>> print len(friends)
3
>>> print range(len(friends))
[0, 1, 2]
>>>```
Looping Through a List

• Two different ways
  – Use the in operator
  – Use the range to index all the elements of the list

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

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
```
List Slicing

• Same as with Strings...

• Also, just like in strings, the second number is “up to but not including”

```python
>>> 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]
```
List Concatenation

• We can create a new list by adding two existing lists together

```python
>>> a = [1, 2, 3]
>>> b = [4, 5, 6]
>>> c = a + b
>>> print c
[1, 2, 3, 4, 5, 6]
>>> print a
[1, 2, 3]
```
List Methods

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