Lists

Data Structures

- ***Data structures*** are particular ways of storing data to make some operations easier or more efficient.
  - That is, they are tuned for certain tasks.
- Data structures are suited to solving certain problems.

Kinds of Data Structures

Roughly two kinds of data structures:

- Built-in data structures - data structures that are so common as to be provided by default.
- User-defined data structures - (classes in object oriented programming) designed for a particular task.

Python Built-in Data Structures

- Python comes with a general set of built-in data structures:
  - strings
  - lists
  - tuples
  - dictionaries
  - sets
  - and others...

- Today we are going to look at ***lists***

The Python List Data Structure

- A list is very simple - it is just an ordered sequence of items.
- You have seen such a sequence before in a string. A string is just a particular kind of list.

Example with the split method

- Recall how we used the split method:

```
>>>myStr = “john, m, doe, 42, 50000”
>>>first, middle, last, age, income = myStr.split(“,”)
```

- The split string method returned the string split up into exactly 4 pieces

Example with the split method

- But what if we didn't have exactly 4 pieces?

```
>>>myStr = "john, m, doe"
>>>first, middle, last, age, income = myStr.split(",")
```

- What will happen?

Example with the split method

- We need to do something more general when we want to break up a string into smaller pieces
  - And we don't know the exact number of pieces!

Using split to give us a list

- We can use the split method to give us a list of an unknown amount of pieces

```
>>>myStr = “I like to eat cookies”
>>>pieces = myStr.split(“ “)
```

- What is in pieces?

Using split to give us a list

- We can use the split method to give us a list of an unknown amount of pieces

```
>>>myStr = “I like to eat cookies.”
>>>pieces = myStr.split(“ “)
```

- What is in pieces?

```
>>>pieces
[‘I’, ‘like’, ‘to’, ‘eat’, ‘cookies’]
```

Using split to give us a list

- We can use the split method to give us a list of an unknown amount of pieces

>>>myStr = “I like to eat cookies. Don’t judge me.”

- What if there’s more than one space in between the words?

Using split to give us a list

- We can use the split method to give us a list of an unknown amount of pieces

>>>myStr = “I like to eat cookies. Don’t judge me.”

>>>pieces = myStr.split() #no argument

[‘I’, ‘like’, ‘to’, ‘eat’, ‘cookies.’, “Don’t”, ‘judge’ ‘me.’]

Using split to give us a list

- The split method gives me an ordered sequence of items!
  - Is an iterable data structure
- How can I get each piece out of a list?
  - Use a for loop!

Creating a list

- Like all data structures, lists have a *constructor function*. It takes an iterable data structure and adds each item to the list.
- It also has a shortcut: the use of square brackets [ ] to indicate the items to be added to the list.

Creating a list

>>> aList = list('abc')

- This creates the list that looks like this:
  - ['a', 'b', 'c']

>>>newList = [1, 3.14159, 'a', True]

- We can also create a list with this shortcut
- This list contains a lot of data structures of different types – what are they?

Similarities with Strings

- concatenate/+ (but only of lists)
- repeat/*
- indexing (the [ ] operator)
- slicing ([:])
- membership (the in operator)
- len (the length operator)

Operators

[1, 2, 3] + [4] ⇒ [1, 2, 3, 4]

[1, 2, 3] * 2 ⇒ [1, 2, 3, 1, 2, 3]

1 in [1, 2, 3] ⇒ True

[1, 2, 3] < [1, 2, 4] ⇒ True

Compare index to index, the first difference determines the result.

Differences Between Lists and Strings

- Lists can contain a mixture of any python object (even other lists); strings can only hold characters.
  - 1,”bill”,1.2345, True
- Lists are **mutable**; their values can be changed while strings are immutable.
- Lists are designated with [ ], with elements separated by commas; strings use “”.

```
myList = [1, 'a', 3.14159, True]
```

myList

1	'a'	3.14159	True	
0	1	2	3	Index Forward
−4	−3	−2	−1	Index Backward

```
myList[1] → 'a'

myList[:3] → [1, 'a', 3.14159]
```

FIGURE 6.1 The structure of a list.

Adding to lists

- We can use the list method append() to add on to the end of a list

```
>>>myList = [‘I’, ‘like’]
>>>myList.append(‘cookies’)
>>>myList
[‘I’, ‘like’, ‘cookies’]
```

List Functions

- len(lst): Number of elements in list (top level).
  len([1, [1, 2], 3]) ⇒ 3
- min(lst): Minimum element in the list. If list of lists, looks at first element of each list.
- max(lst): Max element in the list.
- sum(lst): Sum the elements, numeric only.