


Data Structures: Lists and Tuples

CSE 231, Bill Punch


Data Structures and algorithms

- Part of the “science” in computer science is the design and use of data structures and algorithms
- As you go on in CS, you will learn more and more about these two areas

 Michigan State University CSE 231, Fall 2009 Intro to Lists 2


What is an algorithm

- an algorithm is a recipe, an approach to solving a problem
- an algorithm can be applied to a particular problem and solve it
- These are general problems such as “sorting” or “searching”

 Michigan State University CSE 231, Fall 2009 Intro to Lists 3

Data Structures


- Data structures are particular ways of storing data to be used typically by an algorithm
- Data structures are suited to solving certain problems, and they are often associated with algorithms.

 Michigan State University CSE 231, Fall 2009 Intro to Lists 4

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) that are designed for a particular task

 Michigan State University CSE 231, Fall 2009 Intro to Lists 5

Python built in data structures

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

 Michigan State University CSE 231, Fall 2009 Intro to Lists 6

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 (what kind)?

Michigan State University CSE 231, Fall 2009 Intro to Lists 7

Similarities with strings

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

Michigan State University CSE 231, Fall 2009 Intro to Lists 8

differences between lists and strings

- lists can contain a mixture of any python object, 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 ""

Michigan State University CSE 231, Fall 2009 Intro to Lists 9

Methods vs Functions

- a function is a small program (such as len) that takes some arguments, the stuff in the parenthesis, and returns some value
- a method is called in a special way, the "dot call". It is called in the context of an object (or a variable holding an object)

Michigan State University CSE 231, Fall 2009 Intro to Lists 10

Again, lists have methods

```
myList = ['a', 1, True]
myList.append('z')
```

arguments to the method

the object that we are calling the method with

the name of the method

Michigan State University CSE 231, Fall 2009 Intro to Lists 11

Some new things

- A list is mutable and can change:
 - myList[0]='a' #index assignment
 - myList.append(), myList.extend()
 - myList.pop()
 - myList.insert(), myList.remove()
 - myList.sort()
 - myList.reverse()

Michigan State University CSE 231, Fall 2009 Intro to Lists 12

Index assignment

- You can change the value at some index in a list

```
myLst = ['a', 1, True, 3.14159]
myLst[0]='abc'
print myLst
['abc', 1, True, 3.14159]
```



Michigan State University
CSE 231, Fall 2009

Intro to Lists

13

Slice assignment

- You can even change an entire slice

```
myLst = ['a', 1, True, 3.14159]
myLst[0:2]='xyz'
print myLst
['x', 'y', 'z', True, 3.14159]
```



Michigan State University
CSE 231, Fall 2009

Intro to Lists

14

More slice assignment

- The item on the RHS must be iterable (a sequence that you can “walk” through)
- If you use the “stride” part of a slice, the number of items on the LHS must match the number of items on the RHS
- Otherwise, the operation just substitutes.



Michigan State University
CSE 231, Fall 2009

Intro to Lists

15

Mutable operations

```
myList=[1,2,3,4]
myList.append(27)
print myList => [1,2,3,4,27]
```

Notice that there was NO RETURN VALUE from a mutable operation! The list was changed but nothing was returned



Michigan State University
CSE 231, Fall 2009

Intro to Lists

16

More mutables

```
myList = ['abc','def','ghi','jkl']
print myList.insert(2, 'xyz')
ERROR, insert changes the list but no
return value, nothing to print

print myList => ['abc','def','xyz','ghi','jkl']
```



Michigan State University
CSE 231, Fall 2009

Intro to Lists

17

calling modifier methods on a list

- Remember that calling any of those methods which modify the list, actually changes the values in the list
- Many do not return a value
- Strings are immutable because any changes are reflected in the return value, NOT the string itself



Michigan State University
CSE 231, Fall 2009

Intro to Lists

18

Example 12 simpleLists

for works as with strings

```
myList = [1,2,3,4]
for element in myList:
    print element, "\", element**2
```

prints out first 4 squares.

Note that the variable in the `for` statement (in this case `element`) is a name we can make up just like any other variable. That variable is assigned one element of the sequence each time through the loop



the range function

- a useful function that can be used in conjunction with the `for` iterator is the `range` function
- the `range` function generates a list integers that the `for` loop can iterate over



range is also half-open

- just like all the other range stuff we have seen, such as in slicing, the `range` function generates sequences that are half open
- That is, the call `range(a,b)`, generates integer values from `a` up to **but not including** `b`



for and range

```
for num in range(1,11):
    print num
```

prints the numbers 1-10.

`range(1,11)` generates `[1,2,3,4,5,6,7,8,9,10]`



Example 13 for statement

Example 14 perfect numbers

a perfect number

- numbers and their factors were mysterious to the Greeks and early mathematicians
- They were curious about the properties of numbers as they held some significance
- A perfect number is a number whose sum of factors (excluding the number) equals the number
- First perfect number is: 6 (1+2+3)



abundant, deficient

- abundant numbers summed to more than the number.
 - 12: $1+2+3+4+6 = 16$
- deficient numbers summed to less than the number.
 - 13: 1



design

- get the top of the range to check, max
- go through the numbers from 1-max
 - for each number, collect all the factors
 - once collected, sum up the factors
 - compare the sum and the number and respond accordingly
- summarize results



Tuples

Tuples

- Tuples are easy, they are simply immutable lists
- They are designated with (,)

```
myTuple = (1, 'a', 3.14, True)
```



The question is, Why?

- The real question is, why have an immutable list, a tuple, as a separate type?
- An immutable list gives you a data structure with some integrity, some permanent-ness if you will
- You know you cannot accidentally change one.



Michigan State University
CSE 231, Fall 2009

Intro to Lists

31

Lists and Tuple

- Everything that works with a list works with a tuple **except** methods that modify the tuple
- Thus indexing, slicing, len, print all work as expected
- However, none of the mutable methods work: append, extend, del



Michigan State University
CSE 231, Fall 2009

Intro to Lists

32

Commas make a tuple

- For tuples, you can think of a comma as the operator that makes a tuple, where the () simply acts as a grouping:

```
myTuple = 1,2 # creates (1,2)
myTuple = (1,) # creates (1)
myTuple = (1) # creates 1 not (1)
myTuple = 1, # creates (1)
```



Michigan State University
CSE 231, Fall 2009

Intro to Lists

33

Example 15 simpleTuples