

Objectives

- More on files
- Introduction to Lists

Mar 3, 2008

Sprenkle - CS111

1

Files

- Conceptually, a file is a **sequence** of data stored in memory
- To use a file in a Python script, create an object of type **file**
 - **file** is a data type **constructor** - "constructs" a file object
 - `<varname> = file(<filename>, <mode>)`
 - `<filename>` : string
 - `<mode>` : string, either "r" for read or "w" for write
 - Example: `dataFile = file("years.dat", "r")`

Mar 3, 2008

Sprenkle - CS111

2

Reading from a File

- Examples of reading from a file using file methods
 - Show file: `data/years.dat` Typically use .dat or .txt file extension for these types of data/text files
- `file_read.py` (using `read()`)
 - How is what Python printed different than the file's content?
 - How to fix?
- `file_read2.py` (using `readline()`)

Mar 3, 2008

Sprenkle - CS111

3

Reading from a File

- Recall that a file is a **sequence** of data
- Can use a **for** loop to iterate through a file
 - A line (of type **string**) from the file
 - file object
 - `for line in dataFile:`
`print line`
 - Read as: for each line in the file, do something

Mar 3, 2008

Sprenkle - CS111

`file_read3.py`

4

Updated Wheel of Fortune

- Look at code that uses files...

Mar 3, 2008

Sprenkle - CS111

5

Problem: Searching a File

- We want to search a file for some term. We want to know *which lines* of the file contain that term and a *count* of the number of lines that contained that term

Mar 3, 2008

Sprenkle - CS111

`file_search.py`

6

Problem: Searching a File

- This time, we want to ignore all lines that begin with “#” (a.k.a., the line is a comment)
 - Assume comments are at the beginning of the line
 - Why would we have comments in a data file?
 - data/years2.dat
 - How can we revise the previous solution to do this?

Mar 3, 2008

Sprengle - CS111

file_search2.py

7

Problem: Ignore Comments

- Assumed that comments were at the beginning of the line
- Ignore everything after the ‘#’
 - Or look for the term in everything before the ‘#’

Mar 3, 2008

Sprengle - CS111

file_search3.py

8

Writing to a File

- Create a file object in write mode:
 - Example: myFile = file(“years.txt”, “w”)
- Example: create a file from user input
 - file_write.py
 - What happens if execute the program again with different user input?

Mar 3, 2008

Sprengle - CS111

9

Handling Numeric Data

- We have been dealing with reading and writing strings so far
- What do we need to do to read **numbers** from a file?
- How can we write numbers to a file?

Mar 3, 2008

Sprengle - CS111

10

Handling Numeric Data

- We have been dealing with reading and writing strings so far
- What do we need to do to read **numbers** from a file?
 - Cast as a numeric type, e.g., int or float
- How can we write numbers to a file?
 - Cast number as a string

Mar 3, 2008

Sprengle - CS111

11

Problem: Temperature Data

- Given: data file that contains the daily high temperatures for the last year for one location
 - Data file contains one temperature per line
 - Example: data/florida.dat
- Problem: What is the average high temperature (to 2 decimal places) for the location?
- Rule of Thumb: Always look at the data file before you try to process it

Mar 3, 2008

Sprengle - CS111

12

Problem: Create a Summary Report

- Given: a file containing students names and their years (freshman, sophomore, junior, or senior) for this class
- Problem: create a report (in a file) that says the year and how many students from that year are in this class.
 - Again, we want to ignore comments in the file
- Do we need to start this program from scratch? Have code we can use or repackage?

`writeSumFile.py`

Mar 3, 2008

Sprengle - CS111

13

Other Sequences of Data

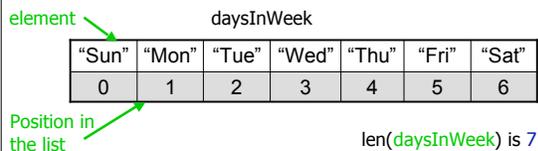
- Sequences so far ...
 - String: sequence of characters
 - Files: sequence of data (lines) in a file
- We commonly group a sequence of data together and refer to them by one name
 - Days of the week: Sunday, Monday, Tuesday, ...
 - Months of the year: Jan, Feb, Mar, ...
 - Shopping list
- Can represent this data as a **list** in Python
 - Similar to **arrays** in other languages

Mar 3, 2008

Sprengle - CS111

14

Lists: A Sequence of Data Elements



- Elements in lists can be *any* data type

Mar 3, 2008

Sprengle - CS111

15

Example Lists in Python

- List of strings:
 - `daysInWeek=["Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"]`
- List of floats
 - `highTemps=[60.4, 70.2, 63.8, 55.7, 54.2]`
- List of file objects
 - `recentFiles=[<17-modules.ppt>, <18-files.ppt>, <19-lists.ppt>]`
- Can contain >1 type

file objects

Mar 3, 2008

Sprengle - CS111

16

Benefits of Lists

- Group related items together
 - Instead of creating separate variables
 - `sunday = "Sun"`
 - `monday = "Mon"`
- Convenient for dealing with large amounts of data
 - Example: could keep all the temperature data in a list if needed to reuse later
- Functions and methods for handling, manipulating lists

Mar 3, 2008

Sprengle - CS111

17

List Operations

Similar to operations for strings

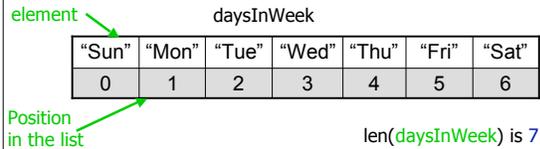
Concatenation	<code><seq> + <seq></code>
Repetition	<code><seq> * <int-expr></code>
Indexing	<code><seq>[<int-expr>]</code>
Length	<code>len(<seq>)</code>
Slicing	<code><seq>[:]</code>
Iteration	<code>for <var> in <seq>:</code>
Membership	<code><expr> in <seq></code>

Mar 3, 2008

Sprengle - CS111

18

Lists: A Sequence of Data Elements



- `<listname>[<int_expr>]`
 - Similar to accessing characters in a string
 - `daysInWeek[-1]` is "Sat"
 - `daysInWeek[0]` is "Sun"

Mar 3, 2008

Sprenkle - CS111

19

Iterating through a List

- Read as
 - For every element in the list ...
- ```

An item in the list list object
 for item in list:
 print item
Iterates through items in list

```
- Equivalent to
 

```

for x in xrange(len(list)):
 print list[x]
Iterates through positions in list

```

Mar 3, 2008

Sprenkle - CS111

daysOfWeek.py

20

## Practice

- Get the list of weekend days from the days of the week list

Mar 3, 2008

Sprenkle - CS111

21

## Practice

- Get the list of weekend days from the days of the week list
  - `daysInWeek=["Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"]`
  - `weekend = daysInWeek[:1] + daysInWeek[-1:]`
  - or
  - `weekend = [ daysInWeek[0]] + [daysInWeek[-1]]`

Mar 3, 2008

Sprenkle - CS111

22

## Membership

- Check if a list contains an element
- Example problem
  - `enrolledstudents` is a list of students who are enrolled in the class
  - Want to check if a student who attends the class is enrolled in the class

```

if student not in enrolledstudents:
 print student, "is not enrolled"

```
- **Problem:** If have a list `attendingstudents`, check if each attending student is an enrolled student

Mar 3, 2008

Sprenkle - CS111

23

## List Methods

| Method Name                            | Functionality                                                                                |
|----------------------------------------|----------------------------------------------------------------------------------------------|
| <code>&lt;list&gt;.append(x)</code>    | Add element <i>x</i> to the end                                                              |
| <code>&lt;list&gt;.sort()</code>       | Sort the list                                                                                |
| <code>&lt;list&gt;.reverse()</code>    | Reverse the list                                                                             |
| <code>&lt;list&gt;.index(x)</code>     | Returns the index of the first occurrence of <i>x</i> , Error if <i>x</i> is not in the list |
| <code>&lt;list&gt;.insert(i, x)</code> | Insert <i>x</i> into list at index <i>i</i>                                                  |
| <code>&lt;list&gt;.count(x)</code>     | Returns the number of occurrences of <i>x</i> in list                                        |
| <code>&lt;list&gt;.remove(x)</code>    | Deletes the first occurrence of <i>x</i> in list                                             |
| <code>&lt;list&gt;.pop(i)</code>       | Deletes the <i>i</i> th element of the list and returns its value                            |

Note: methods do **not** return a copy of the list ...

Mar 3, 2008

Sprenkle - CS111

24

## Fibonacci Sequence

- Create a list of the 1st 15 Fibonacci numbers

➤  $F_0 = F_1 = 1; F_n = F_{n-1} + F_{n-2}$

```
fibs = [] # create an empty list
fibs.append(1) # append the first two Fib numbers
fibs.append(1)
for x in xrange(2,16): # compute the next 13 nums
 newfib = fibs[x-1]+fibs[x-2]
 fibs.append(newfib)

print fibs # print out the list
```

Mar 3, 2008

Sprengle - CS111

fibs.py

25

## Fibonacci Sequence

- Create a list of the 1st 15 Fibonacci numbers

➤  $F_0 = F_1 = 1; F_n = F_{n-1} + F_{n-2}$

```
fibs = range(15) # creates a list of size 15,
 # containing nums 0 to 14
fibs[0] = 1
fibs[1] = 1
for x in xrange(2,15):
 newfib = fibs[x-1]+fibs[x-2]
 fibs[x] = newfib

for num in fibs: # print each num on sep line
 print num
```

Similar to xrange.  
Call similarly

Mar 3, 2008

Sprengle - CS111

fibs2.py

26

## Broader Issue

- Environmental Monitoring with Sensor Networks
  - ZebraNet
  - Volcano monitoring

Mar 3, 2008

Sprengle - CS111

27