

Objectives

- Wrap up files
- Lists

Mar 9, 2011

Sprengle - CSCI111

1

Review: Files

- How do you open a file in Python?
- Whenever you open a file (i.e., construct a file), what should you always remember to do?
- What data types are read/written in files by default?
- How do we handle numeric data?

Mar 9, 2011

Sprengle - CSCI111

2

Reviewing Solved Problems

Mar 9, 2011

Sprengle - CSCI111

3

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, on the same line.
 - 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?

`writeSumReport.py`

Mar 9, 2011

Sprengle - CSCI111

4

Recall

- Focus for most of remainder of semester is **data types** and what we can **do** with that data

Mar 9, 2011

Sprengle - CSCI111

5

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 9, 2011

Sprengle - CSCI111

6

Lists: A Sequence of Data Elements

element		daysInWeek					
"Sun"	"Mon"	"Tue"	"Wed"	"Thu"	"Fri"	"Sat"	
0	1	2	3	4	5	6	

Position in the list

len(daysInWeek) is 7

- Elements in lists can be *any* data type

What does this look similar to, in structure?

Mar 9, 2011

Sprengle - CSCI111

7

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-functions.ppt>, <18-files.ppt>, <20-lists.ppt>]
- Lists can contain >1 type

Mar 9, 2011

Sprengle - CSCI111

8

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 9, 2011

Sprengle - CSCI111

9

List Operations

Similar to operations for strings

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

Mar 9, 2011

Sprengle - CSCI111

10

Lists: A Sequence of Data Elements

element		daysInWeek					
"Sun"	"Mon"	"Tue"	"Wed"	"Thu"	"Fri"	"Sat"	
0	1	2	3	4	5	6	

Position in the list

len(daysInWeek) is 7

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

Mar 9, 2011

Sprengle - CSCI111

11

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 9, 2011

Sprengle - CSCI111

daysOfWeek.py

12

## Practice

- Get the *list* of weekend days from the days of the week list
  - `daysInWeek=["Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"]`

Mar 9, 2011

Sprenkle - CSCI111

13

## 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:]` ← Gives back a *list*
  - or
  - `weekend = [daysInWeek[0]] + [daysInWeek[-1]]` ← Gives back an element of list, which is a *str*

Mar 9, 2011

Sprenkle - CSCI111

14

## 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 9, 2011

Sprenkle - CSCI111

15

## 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 9, 2011

Sprenkle - CSCI111

16

## Fibonacci Sequence

- Goal: Solve using *list*
- $F_0 = F_1 = 1$
- $F_n = F_{n-1} + F_{n-2}$
- Example sequence: 1, 1, 2, 3, 5, 8, 13, 21, ...

Mar 9, 2011

Sprenkle - CSCI111

17

## Fibonacci Sequence

- Create a list of the 1st 15 Fibonacci numbers
  - $F_0 = F_1 = 1; F_n = F_{n-1} + F_{n-2}$  Grow list as we go

```
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 9, 2011

Sprenkle - CSCI111

fibs.py

18

## Fibonacci Sequence

- Create a list of the 1st 15 Fibonacci numbers

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

- Create list
- Update values

Similar to xrange,  
Call similarly

```
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
```

Mar 9, 2011

Sprengle - CSCI111

fibs2.py

19

## range vs xrange

- **range**: creates a *list*

➤ Use when you want a list

`for x in range(10000):`

Won't be able to use this list outside of for loop because not named

➤ Loop goes through each element in the *list*

- list has 10,000 integers from 0 to 9,999

- **xrange**: creates an *iterator*

➤ More efficient to use in **for** loops when you want a counter (not a list)

`for x in xrange(10000):`

➤ Generates 10,000 numbers, one by one

Mar 9, 2011

Sprengle - CSCI111

20

## Lists vs. Arrays

- Briefly, lists are similar to arrays in other languages
  - More similar to *Vectors* in C++ and *ArrayLists* in Java
- Typically, arrays have **static** lengths
  - Can't insert and remove elements from arrays so that the length of the array changes
  - Need to make the array as big as you'll think you'll need

Mar 9, 2011

Sprengle - CSCI111

21

## Lists vs. Strings

- Strings are **immutable**
  - Can't be mutated?
  - Er, can't be modified/changed
- Lists are **mutable**
  - Can be changed
  - Changes how we call/use methods

```
groceryList=["milk", "eggs", "bread", "Doritos", "0J", \ "sugar"]
groceryList[0] = "skim milk"
groceryList[3] = "popcorn"
groceryList is now ["skim milk", "eggs", "bread", \ "popcorn", "0J", "sugar"]
```

Mar 9, 2011

Sprengle - CSCI111

22

## Practice

- list = [7,8,9]
- string = "789"
- list[1]
- string[1]
- string.upper()
- list.reverse()
- string
- list
- string = string.upper()
- list = list.reverse()
- string
- list

Mar 9, 2011

Sprengle - CSCI111

23

## Special Value: None

- Special value we can use
  - E.g., Return value from function when there is an error
- Similar to **null** in Java
- If you execute

```
list = list.sort()
print list
```

➤ Prints None because `list.sort()` does **not** return anything

Mar 9, 2011

Sprengle - CSCI111

24

## Practice: Wheel of Fortune

- Allow user to choose between several categories of puzzles
  - Each category is represented by a different file, e.g., oscars.txt, grammy\_noms.txt, famous\_pairs.txt
- How to model/implement this in Python?
  - How to represent data?

Mar 9, 2011

Sprenkle - CSCI111

25

## Practice: Wheel of Fortune

- Modify to keep track of previous guesses
  - If user made that guess before, print message
- What are the data types of the data we're modeling?

Mar 9, 2011

Sprenkle - CSCI111

26

## Practice: Wheel of Fortune

- Model the wheel
  - Money
  - Bankruptcy, lose a turn, free spin
- Simulate spinning the wheel

Mar 9, 2011

Sprenkle - CSCI111

27

## Assignments

- Lab 7 due Friday
- Broader Issue: Digital Humanities due Friday

Mar 9, 2011

Sprenkle - CSCI111

28

## Practice: Wheel of Fortune

- Read in all puzzles from a file, then randomly select from those puzzles
- Modify: don't allow repeats

Mar 9, 2011

Sprenkle - CSCI111

29

## Exam

- Focus on material after first exam
- **More** focus on reading and understanding code
- When writing code, don't need comments or constants unless
  - explicitly asked or
  - it helps you or it helps me understand what you're trying to do
- Reminders:
  - Concise (but complete!) answers
  - Budget time to complete writing code

Mar 9, 2011

Sprenkle - CSCI111

30