

## Objectives

- More on Lists
  - Methods
  - Using in functions
- Dictionaries

Oct 22, 2007

Sprenkle - CS111

1

## Other Sequences of Data

- 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: January, February, March, ...
  - Shopping list
- Can represent this data as a **list** in Python
  - Similar to **arrays** in other languages

Oct 22, 2007

Sprenkle - CS111

2

## Benefits of Lists

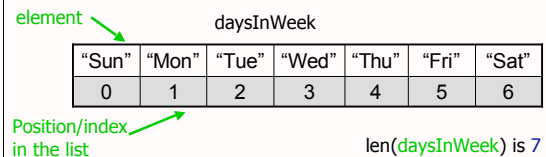
- Group related items together
  - Name with a single variable name instead of creating many separate variables
    - E.g., sunday = "Sun"
  - List has an **order**
- Convenient for dealing with large amounts of data
  - Example: could keep all the temperature data in a list if needed to reuse later
- Functions and methods for handling, manipulating lists

Oct 22, 2007

Sprenkle - CS111

3

## Lists: A Closer Look



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

Oct 22, 2007

Sprenkle - CS111

4

## List Operations

Concatenation	<code>&lt;seq&gt; + &lt;seq&gt;</code>
Repetition	<code>&lt;seq&gt; * &lt;int_expr&gt;</code>
Indexing	<code>&lt;seq&gt;[&lt;int_expr&gt;]</code>
Length	<code>len(&lt;seq&gt;)</code>
Slicing	<code>&lt;seq&gt;[:]</code>
Iteration	<code>for &lt;var&gt; in &lt;seq&gt;:</code>
Membership	<code>&lt;expr&gt; in &lt;seq&gt;</code>

Similar to operations for strings

Oct 22, 2007

Sprenkle - CS111

5

## Iterating through a List

- Read as
    - For every element in the list ...
      - An item in the list
      - list object
  - Equivalent to
- ```
for x in xrange(len(list)):
    print list[x]
```
- Iterates through positions in list

Oct 22, 2007

Sprenkle - CS111

daysOfWeek.py

6

## 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 not student in enrolledstudents:  
    print student, "is not enrolled"
```
- **Problem:** If have a list **attendingstudents**, check if each attending student is an enrolled student

Oct 22, 2007

Sprenkle - CS111

7

## List Methods

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

Oct 22, 2007

Sprenkle - CS111

8

## Fibonacci Sequence

- Print the first 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
```

Oct 22, 2007

Sprenkle - CS111

fibs.py

9

## Fibonacci Sequence

- Print the first 15 Fibonacci numbers
    - $F_0 = F_1 = 1$
    - $F_n = F_{n-1} + F_{n-2}$
- Similar to xrange  
Can use same parameters
- ```
fibs = range(15)    # creates a list of size 15  
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
```

Oct 22, 2007

Sprenkle - CS111

fibs2.py

10

## 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

Oct 22, 2007

Sprenkle - CS111

11

## Lists vs. Strings

- Strings are **immutable**
    - Can't be mutated?
    - Er, can't be modified/changed
  - Lists can be changed
    - **Mutable!**
- ```
groceryList=["milk", "eggs", "bread", "Doritos", "OJ", "sugar"]  
groceryList[0] = "skim milk"  
groceryList[3] = "popcorn"  
  
groceryList is now ["skim milk", "eggs", "bread", "popcorn",  
"OJ", "sugar"]
```

Oct 22, 2007

Sprenkle - CS111

12

## Copies of Lists

- What does the following code output?

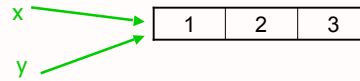
```
x = [1, 2, 3]
y = x
y[0] = -1
print x
print y
```

Oct 22, 2007

Sprenkle - CS111

13

## List Identifiers are Pointers



- `y` is not a copy of `x`
  - Points to what `x` points to
- How to make a copy of `y`?
  - `y = x + []`
  - OR: `y = []`
    - `y.extend(x)`

Oct 22, 2007

Sprenkle - CS111

14

## Lists as Parameters to Functions

- If modify a list passed as a parameter into a function, the list is modified outside the function
  - Lists are **not** passed-by-value
  - Different from immutable types (e.g., numbers, strings)
- Parameter is actually a **pointer** to the list in memory

Oct 22, 2007

Sprenkle - CS111

15

## Problem: Sort a list of 3 numbers, in descending order

- How with list methods?
- Can we do this using only three comparisons?

```
# order list such that list[0] >= list[1] >= list[2]
def descendSort3Nums( list3 ):
```

Called as:

```
list = ...
descendSort3Nums(list)
```

Oct 22, 2007

Sprenkle - CS111

[descendSort.py](#)

16

## How Does **in** Work for Lists?

- Example: `x` in groceries, where groceries is a list
- For each element in list, checks if element equals (`==`) `x`
- In the worst case, how many elements does **in** have to check?
  - How could we improve the search?

Oct 22, 2007

Sprenkle - CS111

17

## Search Improvements

- Sort the List
  - How helps search?

Oct 22, 2007

Sprenkle - CS111

[Hi-Low game](#)

18

## Faster Lookups

- What if I wanted to know the Registrar's phone number?
  - Would I search through an alphabetized list of phone numbers?
  - No, I would look up the Registrar and find the phone number **associated** with the Registrar
- This type of data structure is known as a **dictionary** in Python
  - Maps a **key** to a **value**

Oct 22, 2007

Sprenkle - CS111

19

## Examples of Dictionaries

- What are the keys and values for
  - Dictionary
  - Textbook's index
  - Cookbook
  - URL (Uniform Resource Locator)
  - Any other things we've done/used in class?
    - IN CLASS ANSWERS: variable names --> values, function names --> function definitions, ascii value --> character, character --> ascii value

Oct 22, 2007

Sprenkle - CS111

20

## Examples of Dictionaries

- What are the keys and values for
  - Dictionary
  - Textbook's index
  - Cookbook
  - URL (Uniform Resource Locator)
  - Any other things we've done/used in class?
- Keys are not necessarily alphabetized
- Mappings are from *one* key to one *or more* values
  - Keys are **unique**, Values are not necessarily unique

Oct 22, 2007

Sprenkle - CS111

21

## Creating Dictionaries in Python

Syntax:

```
{<key>:<value>, ..., <key>:<value>}
```

```
empty = {}
```

```
ascii = { 'a':97, 'b':98, 'c':99, ..., 'z':122 }
```

Oct 22, 2007

Sprenkle - CS111

22

## Accessing Values using Keys

- Syntax:  
`<dictionary>[<key>]`
- Examples:  
`ascii['z']`  
`directory['registrar']`
- **KeyError** if key is not in dictionary

Oct 22, 2007

Sprenkle - CS111

23

## Inserting Key-Value Pairs

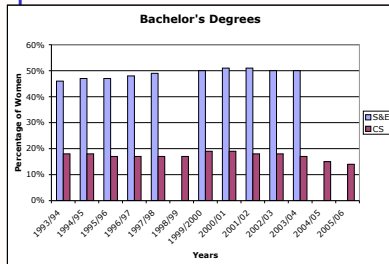
- Syntax:  
`<dictionary>[<key>] = <value>`
- `ascii['a'] = 97`

Oct 22, 2007

Sprenkle - CS111 `ascii_dictionary.py`

24

## Broader Issues: Diversity in Computer Science



- Science & Engineering: near 50%

Oct 22, 2007

Sprenkle - CS111

25