

## Objectives

- Exam next Friday
- Introduction to Dictionaries
- Broader Issue

Mar 7, 2008

Sprenkle - CS111

1

## Sexual Assault Statement

- [http://bloggery.wlu.edu/ec/2008/03/statement\\_on\\_sexual\\_assault.html](http://bloggery.wlu.edu/ec/2008/03/statement_on_sexual_assault.html)

Mar 7, 2008

Sprenkle - CS111

2

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

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

Mar 7, 2008

Sprenkle - CS111

3

## 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**

Mar 7, 2008

Sprenkle - CS111

4

## Examples of Dictionaries

Dictionary	Keys	Values
Dictionary		
Textbook's index		
Cookbook		
URL (Uniform Resource Locator)		

- Any other things we've done/used in class?

Mar 7, 2008

Sprenkle - CS111

5

## Examples of Dictionaries

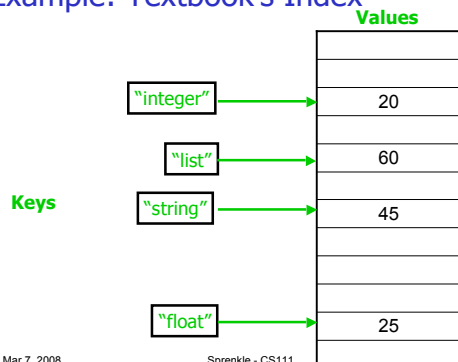
- Real-world:
  - Dictionary
  - Textbook's index
  - Cookbook
  - URL (Uniform Resource Locator)
- Examples from class
  - Function name --> function definition
  - Variable name --> value
  - ASCII value --> character

Mar 7, 2008

Sprenkle - CS111

6

## Example: Textbook's Index



Mar 7, 2008

Sprenkle - CS111

7

## Dictionaries in Python

- Map **keys** to **values**
  - Keys are probably not alphabetized
  - Mappings are from *one* key to *one or more* values
    - Keys are **unique**, Values are not necessarily unique
      - Example: student id → last name
    - Keys must be **immutable** (numbers, strings)
- Similar to Hashtables/Hashmaps in other languages

How would we handle if there is more than one value for a key?

Mar 7, 2008

Sprenkle - CS111

8

## Creating Dictionaries in Python

Syntax:

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

```
empty = {}
```

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

Mar 7, 2008

Sprenkle - CS111

9

## Dictionary Operations

Indexing	<dict>[<key>]
Length (# of keys)	len(<dict>)
Iteration	for <key> in <dict>:
Membership	<key> in <dict>
Deletion	del <dict>[<key>]

Unlike strings and lists, doesn't make sense to do slicing, concatenation, repetition for dictionaries

Mar 7, 2008

Sprenkle - CS111

10

## Dictionary Methods

Method Name	Functionality
<dict>.clear()	Remove all items from dictionary
<dict>.keys()	Returns a copy of dictionary's <i>list</i> of keys
<dict>.values()	Returns a copy of dictionary's <i>list</i> of values
<dict>.get(x, default)	Returns <dict>[x] if x is a key; Otherwise, returns <b>None</b> (or default value)

Mar 7, 2008

Sprenkle - CS111

11

## Accessing Values using Keys

- Syntax:
 

```
<dictionary>[<key>]
```
- Examples:
 

```
ascii['z']
```

```
directory['registrar']
```
- **KeyError** if key is not in dictionary
  - Runtime error; exits program

Mar 7, 2008

Sprenkle - CS111

12

## Alternatively, Using `get` method

- `<dict>.get(x[,default])`
  - Returns `<dict>[x]` if `x` is a key; Otherwise, returns `None` (or default value)

```
ascii.get('z')

directory.get('registrar')
```
- If no mapping, get **None** back instead of **KeyError**

Mar 7, 2008

Sprenkle - CS111

13

## Accessing Values Using Keys

- Typically, you will check if dictionary has a key before trying to access the key

```
if 'z' in ascii:
    value = ascii['z']
```

Know mapping exists  
before trying to access

- Or handle if get default back

```
val = ascii.get('z')
if val is None:
    # do something ...
```

Mar 7, 2008

Sprenkle - CS111

14

## 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 7, 2008

Sprenkle - CS111

15

## Example Using **None**

```
# returns the lowercase letter translated by the key.
# If letter is not a lowercase letter, returns None
def translateLetter( letter, key ):
    if letter < 'a' or letter > 'z':
        return None
    # As usual ...
```

```
# example use
enclLetter = translateLetter(char, key)
if enclLetter is None:
    print "Error in message: ", char
```

Mar 7, 2008

Sprenkle - CS111

16

## Inserting Key-Value Pairs

- Syntax:

```
<dictionary>[<key>] = <value>
```
- `ascii['a'] = 97`
  - Creates new mapping of 'a' --> 97

Mar 7, 2008

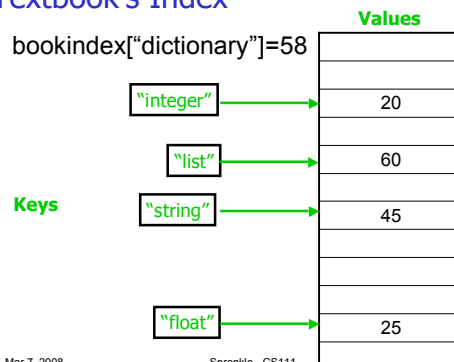
Sprenkle - CS111

[ascii\\_dictionary.py](#)

17

## Textbook's Index

- `bookindex["dictionary"] = 58`



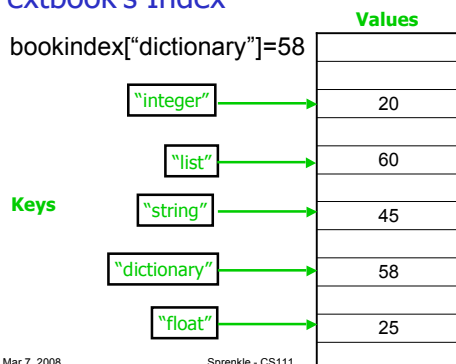
Mar 7, 2008

Sprenkle - CS111

18

## Textbook's Index

- `bookindex["dictionary"] = 58`



Mar 7, 2008

Sprenkle - CS111

19

## Adding/Modifying Key-Value Pairs

- **Syntax:**  
`<dictionary>[<key>] = <value>`
- `directory['registrar'] = 8455`
  - Modifies old entry (if it existed) and changes mapping for 'registrar' to 8455

Mar 7, 2008

Sprenkle - CS111

20

## Example Problem

- Given a file of the form
  - > <lastname> <year>
- Create a mapping between the last names and years

Mar 7, 2008

Sprenkle - CS111

years\_dictionary.py

21

## Example Problem

- Given a file of the form
  - <lastname> <year>
- Create and display a mapping between the last names and years
  - How to display the mapping in a pretty way?
  - What order is the data printed in?

Mar 7, 2008

Sprenkle - CS111

1 years\_dictionary.py

22

## Example Problem

- Modify the previous program to keep track of the number of students of each year

➤ Could we solve this using a list?

Mar 7, 2008

Sprenkle - CS111

years\_dictionary2.py

23

## Broader Issues: Environmental Monitoring

- Interdisciplinary projects involving sensor networks
  - Important new-ish CS research area
- Disclaimer:
  - Not a seismologist or a biologist
- Groups
  - Volcano: Joe, Stuart, Joa, Ty
  - Zebra: Andrew, Clay, Colin, Nay
  - Zebra: Lucy, Arturo, Vasil, Greg
  - Zebra: Julie, Alex, Dave

Mar 7, 2008

Sprenkle - CS111

24

## Discussion

- What are the CS challenges to both projects?
  - Any challenges only applicable to one project?
- How does the environment impact the CS research problems/solutions?
- How did the researchers address these challenges?
  - How would **you** address the challenges?

Mar 7, 2008

Sprenkle - CS111

25

## Overview of Challenges: Efficiency

- Some programmers thought that efficiency didn't matter anymore
  - GB of memory, terabytes of storage on machines
- Now: small and embedded devices
  - Need to be efficient!
- Energy in battery powered nodes
- Amount of data stored (when to delete?)
- When, amount of data transferred

Mar 7, 2008

Sprenkle - CS111

26

## Overview of Challenges: Reliability

- Data delivery
  - Missing data
  - Connectivity (good signal?)
  - Duplicate data (different sources?)
  - Dead sensor nodes
  - Calibration of data (time synchronization)
- Nodes
  - Withstand extreme weather, conditions
  - Battery life
- Robustness: recover from software failure/malfunction or bad data

Mar 7, 2008

Sprenkle - CS111

27

## Overview of Challenges

- Testing
  - Accurately simulate conditions (which will vary widely over long periods of time)
- Different goals from domain scientists
  - CS: push boundaries of sensor networks
    - Example: Improve reliability of data to 95%
    - Seismologists: need 100% reliable data

Mar 7, 2008

Sprenkle - CS111

28

## Overview of Solutions: Efficiency

- Energy in battery powered nodes
  - Solar-powered batteries
  - Only transmit if new data
- Amount of data stored (when to delete?)
  - Notify all when data gets to base station
- When, amount of data transferred
  - ZebraNet: only transmit if new data
    - Only transmit if zebra gives data to base
  - Volcano: only when "interesting" data

Mar 7, 2008

Sprenkle - CS111

29

## Overview of Solutions: Reliability

- Data delivery
  - Redundancy of data -- verify/validate it is correct
  - Only send to zebras with history of reporting back to base station
- Nodes
  - Weather proofing
  - Batteries: solar-panels to recharge

Mar 7, 2008

Sprenkle - CS111

30

## Overview of Solutions: Testing

- Novel simulations!
- Emulate environment/scenarios on computer
- Emulate zebras with horses
- Push software to make sure it “recovers” appropriately from errors or bad information