Objectives

- Dictionaries

Review

- How can I get a subset of a list?
  - E.g., I want elements 2 through 5 of my list
- How can I iterate through a list?
  - Especially important in Deal or No Deal
- Advice on Deal or No Deal: keep it simple
  - Then do formatting, etc.

Get out handout from last time

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?

Faster Lookups

- 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**
  - Phone book's key: "Registrar", value: phone number

Examples of Dictionaries

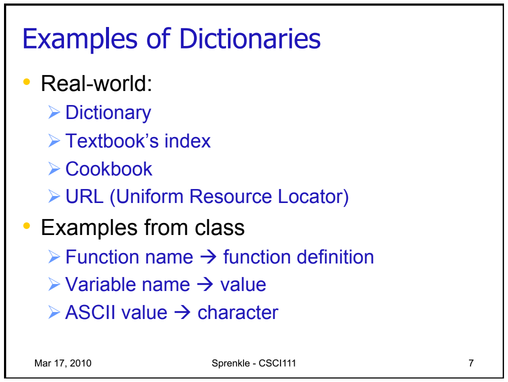
Dictionary	Keys	Values
Dictionary		
Textbook's index		
Cookbook		
URL (Uniform Resource Locator)		

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

Examples of Dictionaries

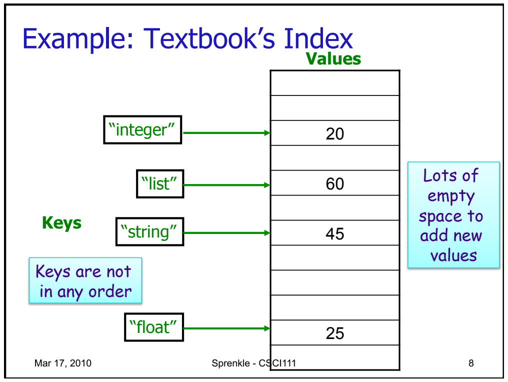
Dictionary	Keys	Values
Dictionary	Word	Definition
Textbook's index	Keyword	Page number
Cookbook	Recipe Name	Recipe
URL (Uniform Resource Locator)	URL	Web page

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



Examples of Dictionaries

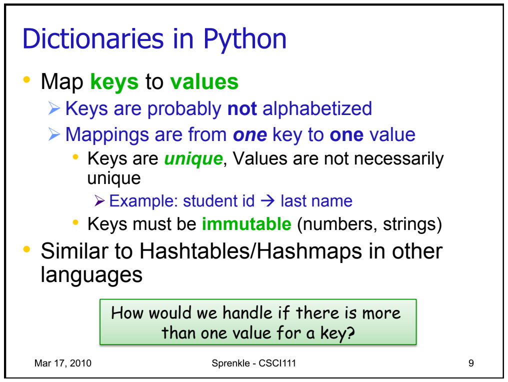
- Real-world:
  - Dictionary
  - Textbook's index
  - Cookbook
  - URL (Uniform Resource Locator)
- Examples from class
  - Function name → function definition
  - Variable name → value
  - ASCII value → character



Example: Textbook's Index

Lots of empty space to add new values

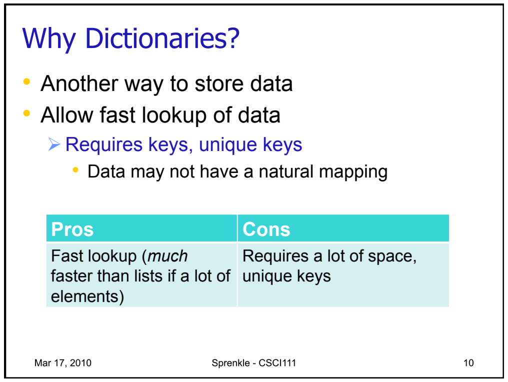
Keys are not in any order



Dictionaries in Python

- Map **keys** to **values**
  - Keys are probably **not** alphabetized
  - Mappings are from ***one*** key to **one** value
    - 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

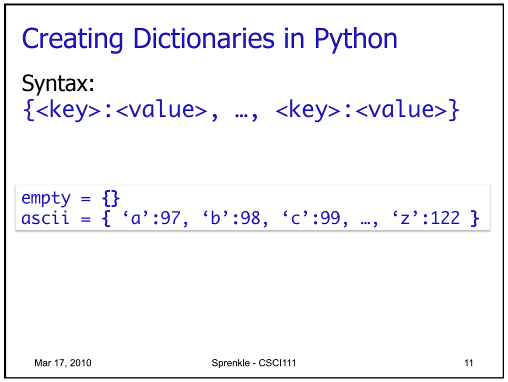
How would we handle if there is more than one value for a key?



Why Dictionaries?

- Another way to store data
- Allow fast lookup of data
  - Requires keys, unique keys
    - Data may not have a natural mapping

Pros	Cons
Fast lookup (*much* faster than lists if a lot of elements)	Requires a lot of space, unique keys

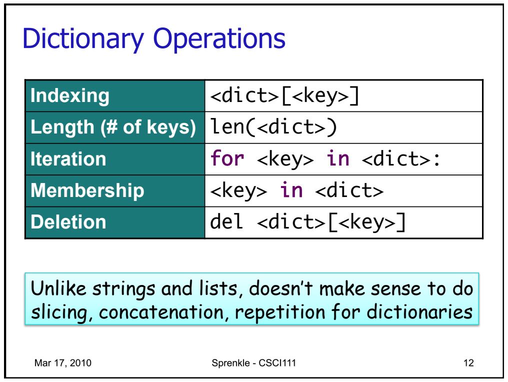


Creating Dictionaries in Python

Syntax:

`{<key>:<value>, …, <key>:<value>}`

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



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

Dictionary Methods

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

Accessing Values using Keys

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

```
ascii['z']

directory['registrar']
```

- `KeyError` if key is not in dictionary
  - Runtime error; exits program

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`

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 …
```

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

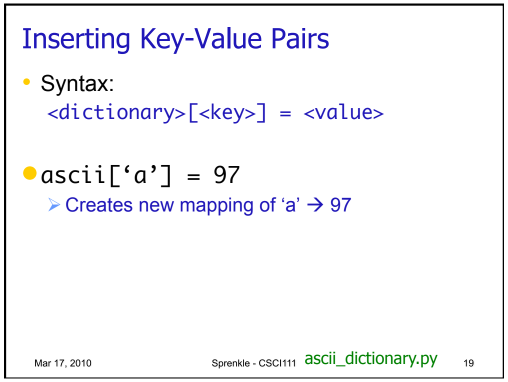
```
list = list.sort()
print list
```

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

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
encLetter = translateLetter(char, key)
if encLetter is None:
     print "Error in message: ", char
```

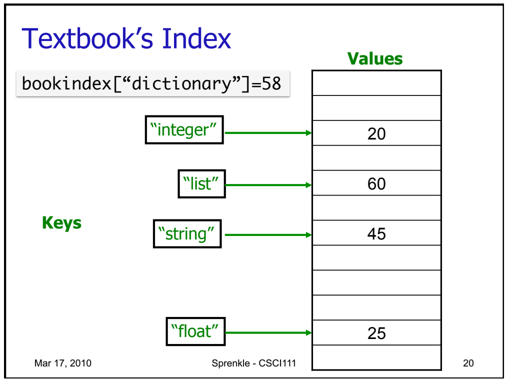


Inserting Key-Value Pairs

- Syntax:
  `<dictionary>[<key>] = <value>`

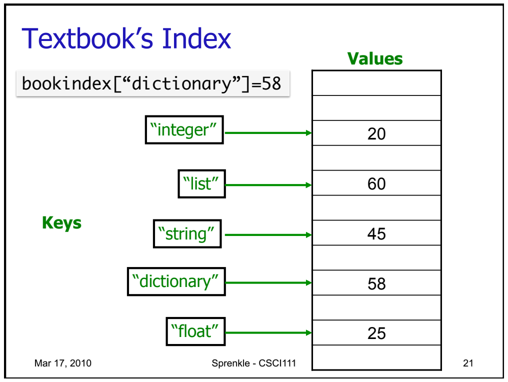
- `ascii['a'] = 97`
  - Creates new mapping of 'a' → 97

ascii_dictionary.py



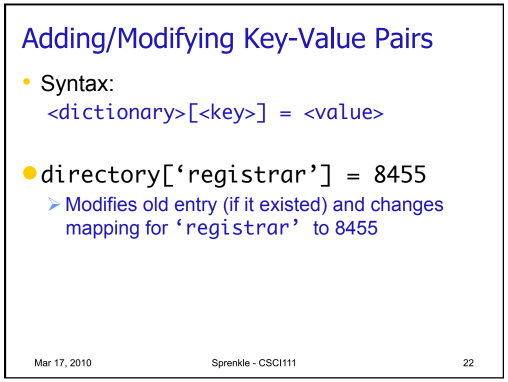
Textbook's Index

`bookindex["dictionary"]=58`



Textbook's Index

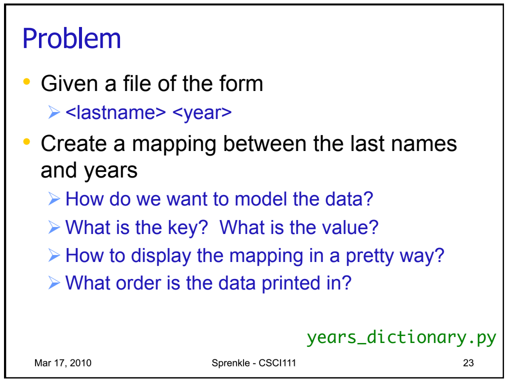
`bookindex["dictionary"]=58`



Adding/Modifying Key-Value Pairs

- Syntax:
  `<dictionary>[<key>] = <value>`

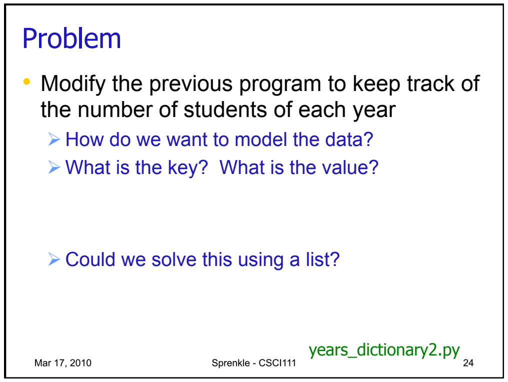
- `directory['registrar'] = 8455`
  - Modifies old entry (if it existed) and changes mapping for `'registrar'` to 8455



Problem

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

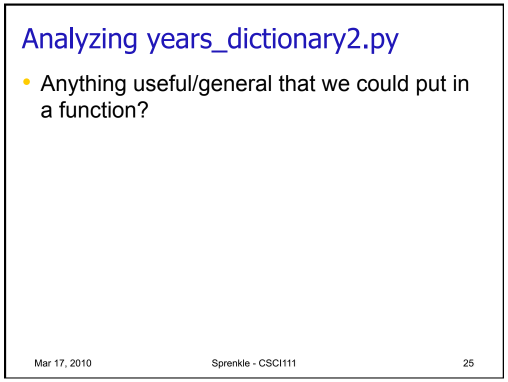
`years_dictionary.py`



Problem

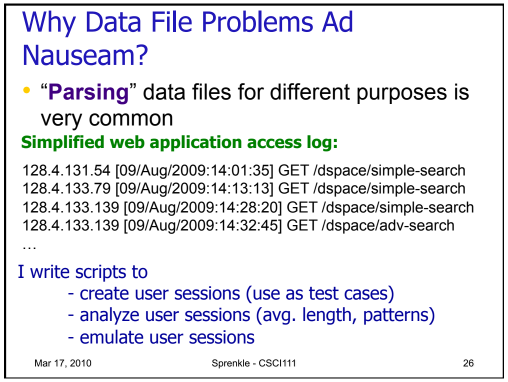
- Modify the previous program to keep track of the number of students of each year
  - How do we want to model the data?
  - What is the key? What is the value?
  - Could we solve this using a list?

years_dictionary2.py



Analyzing years_dictionary2.py

- Anything useful/general that we could put in a function?



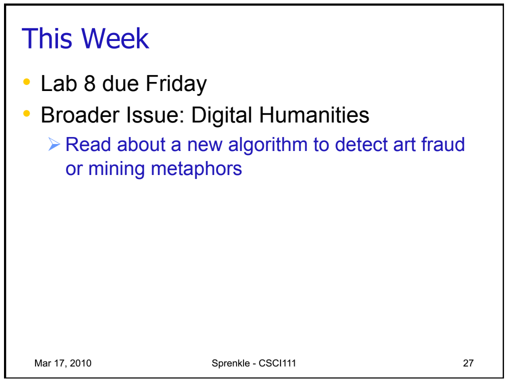
Why Data File Problems Ad Nauseam?

- "**Parsing**" data files for different purposes is very common

Simplified web application access log:

```
128.4.131.54 [09/Aug/2009:14:01:35] GET /dspace/simple-search
128.4.133.79 [09/Aug/2009:14:13:13] GET /dspace/simple-search
128.4.133.139 [09/Aug/2009:14:28:20] GET /dspace/simple-search
128.4.133.139 [09/Aug/2009:14:32:45] GET /dspace/adv-search
…
```

I write scripts to
- create user sessions (use as test cases)
- analyze user sessions (avg. length, patterns)
- emulate user sessions



This Week

- Lab 8 due Friday
- Broader Issue: Digital Humanities
  - Read about a new algorithm to detect art fraud or mining metaphors