

Objectives

- Dictionaries

Mar 11, 2009

Sprenkle - CS111

1

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 11, 2009

Sprenkle - CS111

2

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

Mar 11, 2009

Sprenkle - CS111

3

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 11, 2009

Sprenkle - CS111

4

Examples of Dictionaries

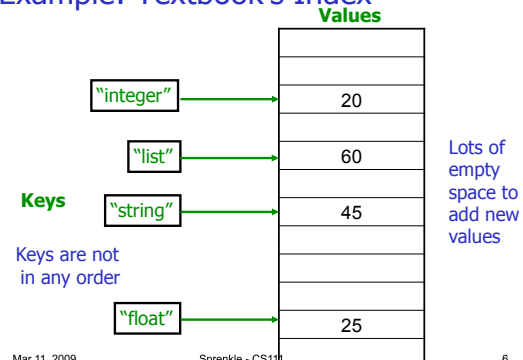
- Real-world:
 - Dictionary
 - Textbook's index
 - Cookbook
 - URL (Uniform Resource Locator)
- Examples from class
 - Function name --> function definition
 - Variable name --> value
 - ASCII value --> character

Mar 11, 2009

Sprenkle - CS111

5

Example: Textbook's Index



Mar 11, 2009

Sprenkle - CS111

6

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 11, 2009

Sprenkle - CS111

7

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 (<i>much</i> faster than lists if a lot of elements)	Requires a lot of space, unique keys

Mar 11, 2009

Sprenkle - CS111

8

Creating Dictionaries in Python

Syntax:

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

```
empty = {}
```

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

Mar 11, 2009

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 11, 2009

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 <i>None</i> (or default value)

Mar 11, 2009

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 11, 2009

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 11, 2009

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 11, 2009

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 11, 2009

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

Mar 11, 2009

Sprenkle - CS111

16

Inserting Key-Value Pairs

- Syntax:

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

Mar 11, 2009

Sprenkle - CS111

ascii_dictionary.py

17

Textbook's Index

- `bookindex["dictionary"] = 58`

		Values
Keys	"integer"	20
	"list"	60
	"string"	45
	"float"	25

Mar 11, 2009

Sprenkle - CS111

18

19

20

21

22

23

24

Analyzing years_dictionary2.py

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

Mar 11, 2009

Sprenkle - CS111

25

This Week

- Lab 8 due Friday
- Broader Issue: Digital Humanities

Mar 11, 2009

Sprenkle - CS111

26