

## Objectives

- Dictionaries

## Review Lab 8

- What went well?
- What didn't go well?
  - What does that mean you should work on/prepare for better?

# LOOKUP ALTERNATIVES

Nov 8, 2017

Sprenkle - CSCI111

3

## List/String Lookup

- How do we “lookup” a value in a list?
- Answer:
  - By its index/position
- Requires:
  - Knowing the index where a value is

Nov 8, 2017

Sprenkle - CSCI111

4

## Alternative Lookup

- Alternative: look up something by its key
  - Example: When I lookup my friend's phone number in my contacts, I don't know that the number is at position X in my contacts. I can look up my friend's number by her *name*.
  - Have a fast way to figure out "given this key, what is the value associated with it?"
- This type of data structure is known as a **dictionary** in Python
  - Maps a **key** to a **value**
  - Contacts' key: "Friend's name", value: phone number

Nov 8, 2017

Sprenkle - CSCI111

5

## Examples of Dictionaries

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

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

Nov 8, 2017

Sprenkle - CSCI111

6

## Examples of Dictionaries

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

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

Nov 8, 2017

Sprenkle - CSCI111

7

## Examples of Dictionaries

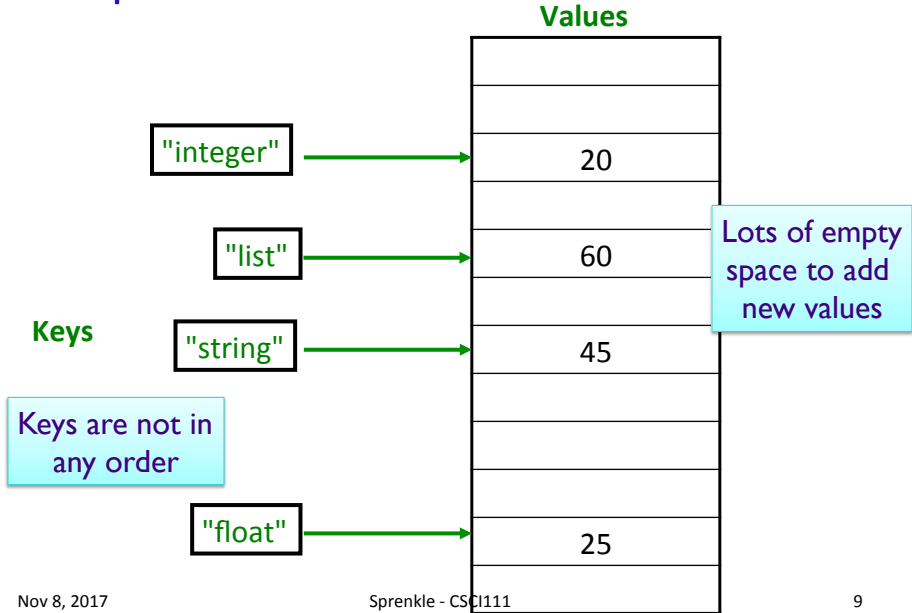
- Real-world:
  - Dictionary
  - Textbook's index
  - Cookbook
  - URL (Uniform Resource Locator)
- Examples from class
  - Variable name → value
  - Function name → function definition
  - ASCII value → character

Nov 8, 2017

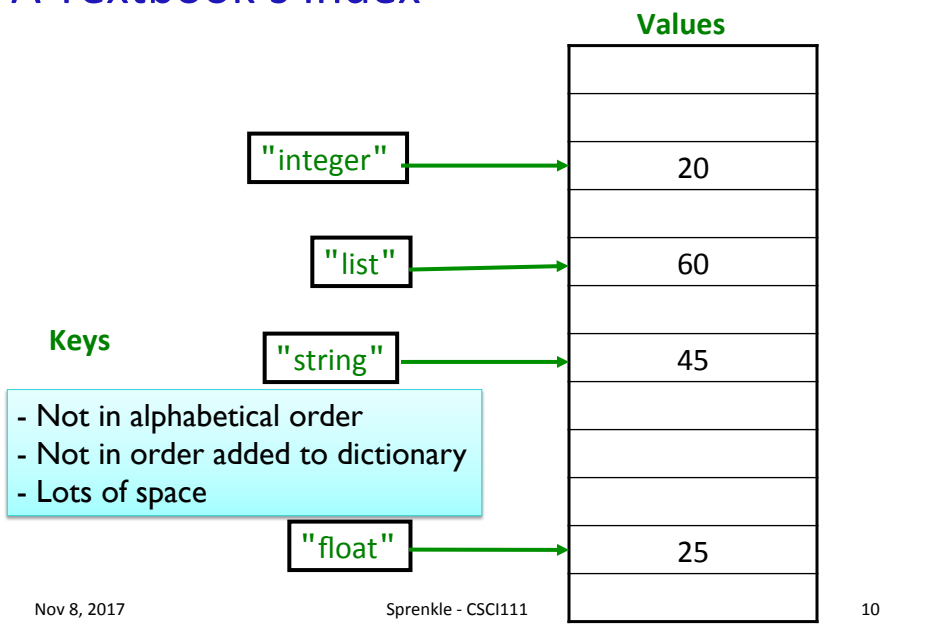
Sprenkle - CSCI111

8

## Example: A Textbook's Index



## A Textbook's Index



## 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 given key?

Nov 8, 2017

Sprenkle - CSCI111

11

## Creating Dictionaries in Python

Syntax:

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

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

Nov 8, 2017

Sprenkle - CSCI111

12

## Dictionary Operations

Indexing	<code>&lt;dict&gt;[&lt;key&gt;]</code>
Length (# of keys)	<code>len(&lt;dict&gt;)</code>
Iteration	<code>for &lt;key&gt; in &lt;dict&gt;:</code>
Membership	<code>&lt;key&gt; in &lt;dict&gt;</code>
Deletion	<code>del &lt;dict&gt;[&lt;key&gt;]</code>

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

## Dictionary Methods

Method Name	Functionality
<code>&lt;dict&gt;.clear()</code>	Remove all items from dictionary
<code>&lt;dict&gt;.keys()</code>	Returns a copy of dictionary's keys (a set-like object)
<code>&lt;dict&gt;.values()</code>	Returns a copy of dictionary's values (a set-like object)
<code>&lt;dict&gt;.get(x, default)</code>	Returns <code>&lt;dict&gt;[x]</code> if <code>x</code> is a key; Otherwise, returns <code>None</code> (or default value)

## Accessing Values Using Keys

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

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

```
contacts['friendname']
```

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

## Accessing Values 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('friendname')
```

- If no mapping,  
**None** is returned instead of **KeyError**



## Accessing Values

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

```
if 'friend' in contacts:  
    number = contacts['friend']
```

Know mapping exists  
before trying to access

- Or handle if `get` returns default

```
number = contacts.get('friend')  
if number is None:  
    # do something ...
```

No phone number exists

Nov 8, 2017

Sprenkle - CSCI111

17

## Recall: 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

Nov 8, 2017

Sprenkle - CSCI111

18

## 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)
```

Nov 8, 2017

Sprenkle - CSCI111

19

## Inserting Key-Value Pairs

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

`ascii_dictionary.py`

Nov 8, 2017

Sprenkle - CSCI111

20

## Textbook's Index

```
bookindex["dictionary"]=58
```

Keys

"integer"



20

"list"



60

"string"



45

"float"



25

Values

Nov 8, 2017

Sprenkle - CSCI111

21

## Textbook's Index

```
bookindex["dictionary"]=58
```

Keys

"integer"



20

"list"



60

"string"



45

"dictionary"



58

"float"



25

Values

Nov 8, 2017

Sprenkle - CSCI111

22

## Adding/Modifying Key-Value Pairs

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

Nov 8, 2017

Sprenkle - CSCI111

23

## Methods `keys()` and `values()`

- Don't actually return a `list` object
- But can be used similarly to a list
- If you want to make them into a list:

```
keys = list(mydict.keys())
```

Nov 8, 2017

Sprenkle - CSCI111

24

## Using Dictionaries

`using_dictionary.py`

- Demonstrate lots of operations, methods, etc. in using dictionaries

Nov 8, 2017

Sprenkle - CSCI111

25

## Problem

`years_dictionary.py`

- Part 1:
  - Given a file of the form
    - <lastname> <classyear>
  - Goal: I want to quickly find out what a student's class is
    - 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?
- Part 2:
  - Prompt user for the last name of the student
  - Display the student's graduation year

Nov 8, 2017

Sprenkle - CSCI111 **Part 3: Repeat Part 2**

26

## Algorithm to Problem

- Create an empty dictionary
- Read in the file line by line
  - Split the line
  - From the split, get the last name and the year
  - Add a mapping of the last name to the year in the dictionary
    - (accumulate the data in the dictionary)
- Process the data in the dictionary, e.g.,
  - Display it, in sorted order
  - Get user input to get answers

Nov 8, 2017

Sprenkle - CSCI111

27

## Looking Ahead

- Exam on Friday

Nov 8, 2017

Sprenkle - CSCI111

28