

# Objectives

- Dictionaries

## Extra Credit Opportunity

Post in Canvas – up to 10 points  
extra credit



17TH NATIONAL SYMPOSIUM OF THEATER IN ACADEME PRESENTS

# BIOADAPTED

AN INTERACTIVE COMEDY EXPLORING THE  
FRIGHTENING FUTURE OF AI

PERFORMANCE CREATED/DIRECTED BY TJAŠA FERME

WITH:  
ARIANNE BANDA  
JUAN CARDENAS  
NASAYANO MCGRADY  
MELODY MUNITZ  
THAMMIE LAINE QUACH  
STEPHANIE WHITE

SPONSORS:  
THE OFFICE OF THE DEAN OF THE COLLEGE  
THE GLASGOW ENDOWMENT  
THE COMPUTER SCIENCE DEPARTMENT  
THE PHYSICS & ENGINEERING DEPARTMENT  
THE COGNITIVE & BEHAVIORAL SCIENCE DEPARTMENT

W&L

FREE AND OPEN TO THE PUBLIC

THURSDAY, MARCH 21, 2024  
7:00PM - 8:30PM  
STACKHOUSE THEATER

March 20, 2024

Sprenkle - CSCI111

2

# LOOKUP ALTERNATIVES

# List/String Lookup

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

# 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 look up my friend's number by her *name*.
  - Need 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: name; 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	Food type	Recipes
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
  - Variable name → value
  - Function name → function definition
  - ASCII value → character



# Example: A Textbook's Index

topicToPageNumber

Values

"integer"



20

"list"



60

Keys

"string"



35

"float"



20

Lots of empty space to add new entries

Keys are not in any order

Keys are unique; values are not necessarily unique

# 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?

# Creating Dictionaries in Python

Syntax:

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

```
empty = {}  
charToAscii = { 'a':97, 'b':98, ..., 'z':122 }
```

# 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

# Accessing Values Using Indexing

- Syntax:

`<dictionary>[<key>]`

- Examples:

```
charToAscii['z']
```

```
nameToPhoneNum['friendname']
```

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

# 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 `get` Method

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

```
charToAscii.get('z')  
nameToPhoneNum.get('friendname')
```
- If no mapping, **None** is returned instead of **KeyError**

# Accessing Values: Look Before You Leap

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

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

Know mapping exists  
before trying to access

- Or handle if get returns default

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



# Recall: Special Value **None**

- Special value we can use
  - E.g., Return value from function when there is an error
- If you execute 

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

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

# Example Using None as an Error

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

```
# example use
encLetter = encryptLetter(char, key)
if encLetter is None:
    print("Can't encrypt character", char, "in message: ")
```

# Inserting Key-Value Pairs

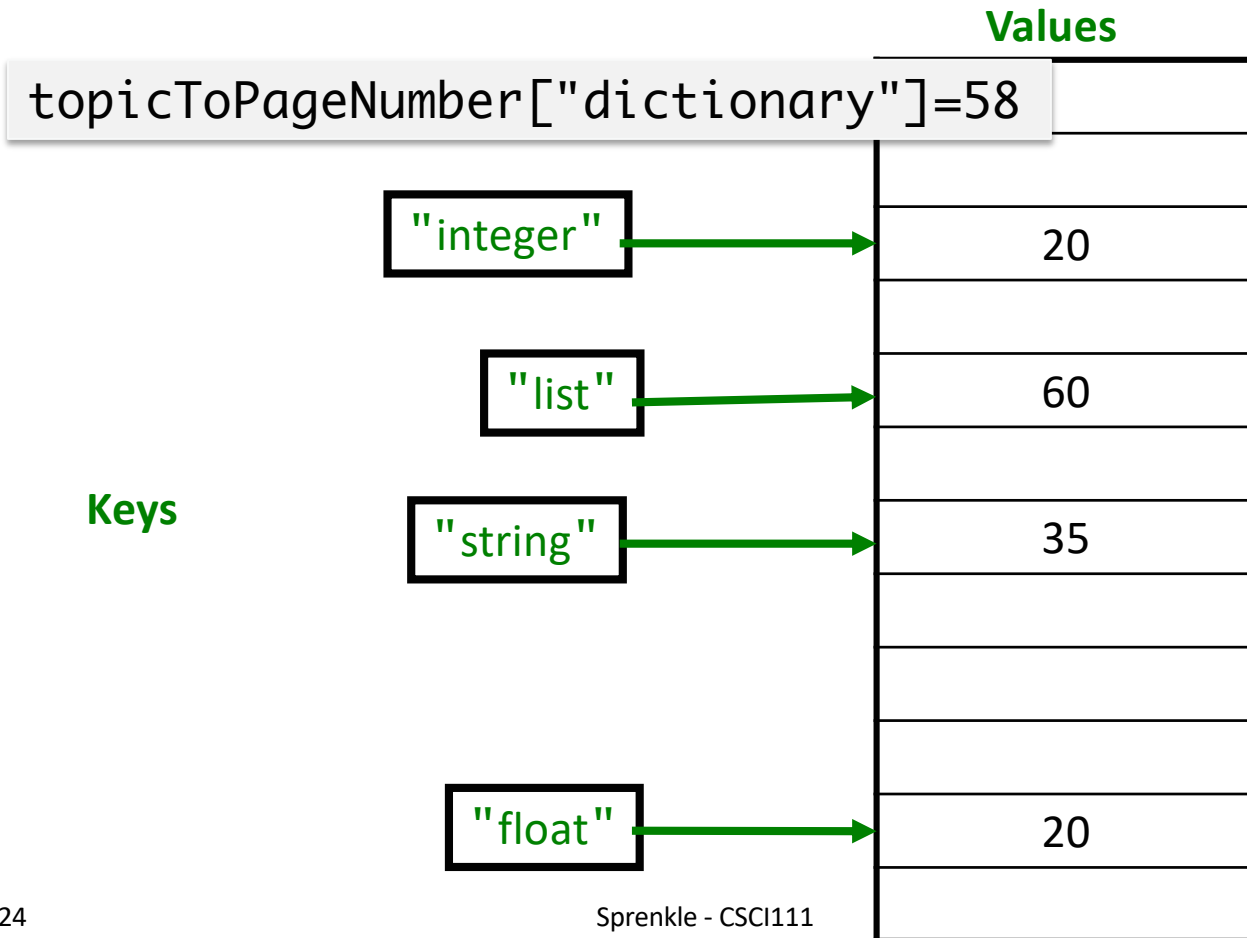
- Syntax:

`<dictionary>[<key>] = <value>`

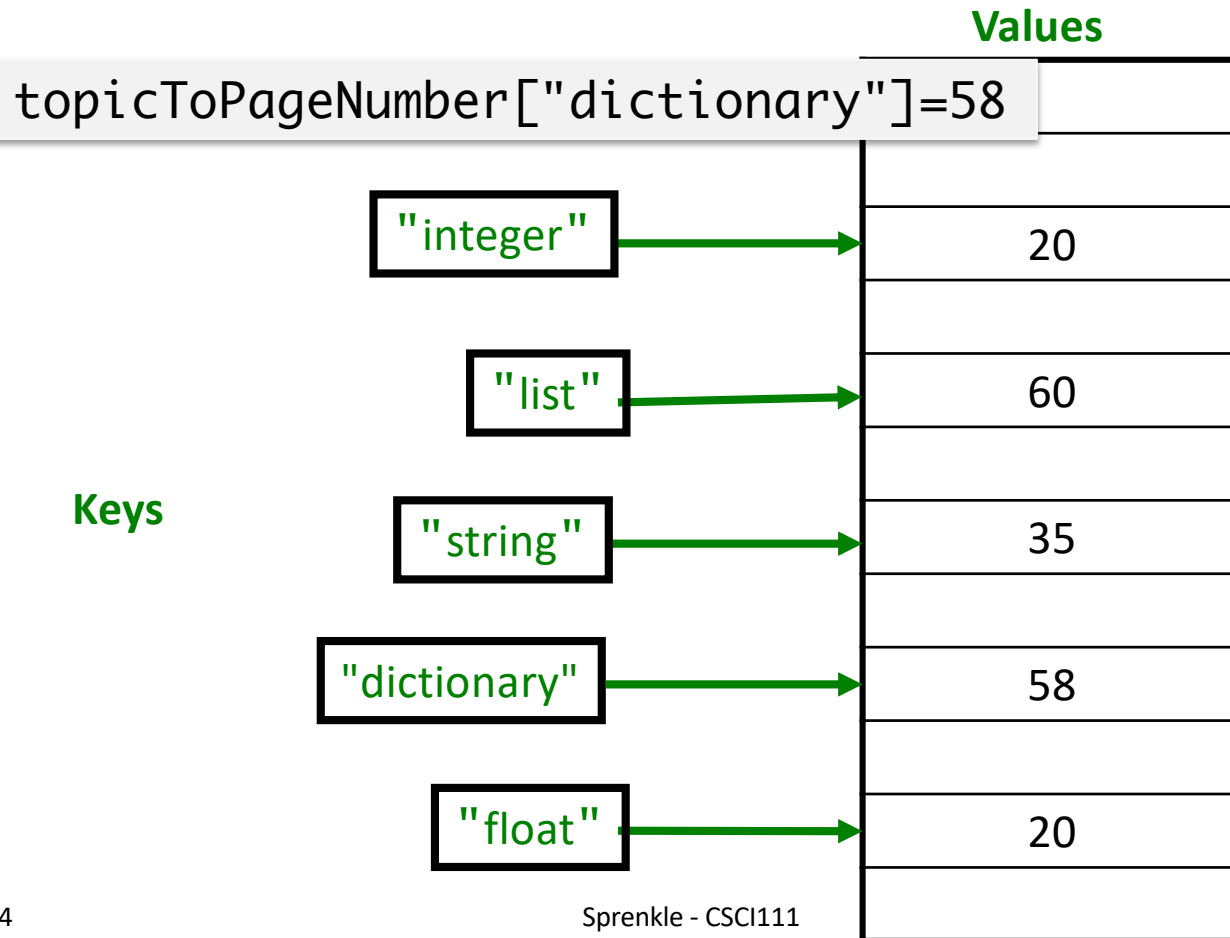
- `charToAscii['a'] = 97`

- Creates new mapping of 'a' → 97

# Textbook's Index: Before Insertion



# Textbook's Index: After Insertion



# Adding/Modifying Key-Value Pairs

- Syntax:

`<dictionary>[<key>] = <value>`

- Example:

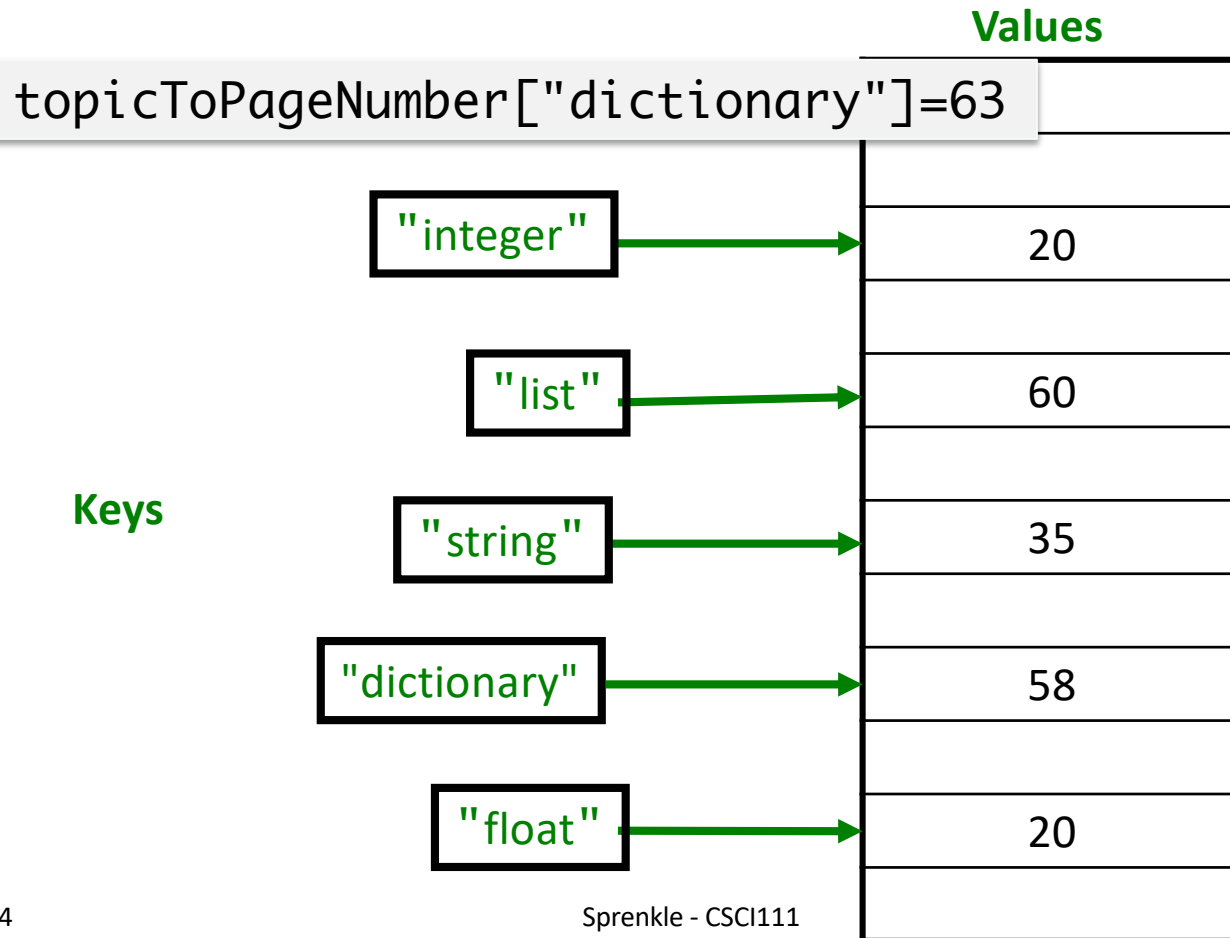
`nameToPhoneNum['registrar'] = 8455`

➤ Adds mapping for 'registrar' to 8455

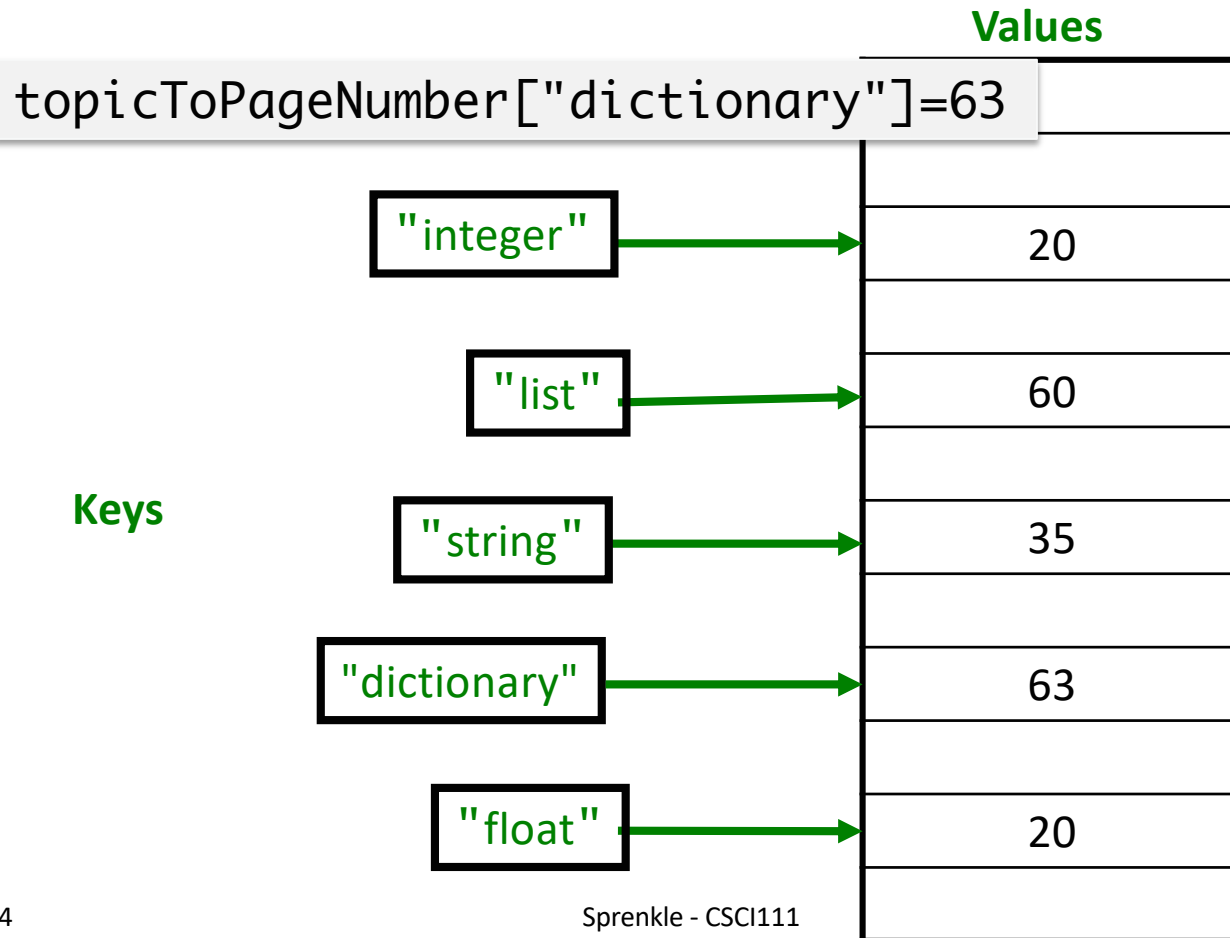
**OR**

➤ If mapping already existed, *modifies* old mapping to 8455

# Textbook's Index: Before Modification



# Textbook's Index: After Modification





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

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

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

# Using Dictionaries

`using_dictionary.py`

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

# Course Registration

- What does this mean under eligibility a course listing in Workday?

2 - reserved for REG:UG:CS=3JR R until 03/28/2024

# Representing Information

- Tag → User-friendly display
  - Ex: REG:UG:CS=3JR R → Junior or 3rd year
- Significance of date
  - 03/28/2024 →

# Problem

- Given a file (data/roster.dat) of the form

<firstname> <gradyear>

- Goal: quickly find the classyear of a particular student
  - Specifically, want to
    - Repeatedly prompt user for a first name of a student (given)
    - Display that student's graduation year

```
Whose class year? Bobby
Bobby is in the class of 2024
```

Example file:

```
Name1 2025
Name2 2026
Name3 2024
Name4 2026
Name5 2024
...
```

- Consider
  - How would we solve this before learning dictionaries?
  - How would we represent this information with dictionaries?
    - What is the key? What is the value?
  - If that dictionary existed, how would we implement the user input part?
  - How do we parse the file to create the dictionary?

# Towards a Solution

- Representing information in a dictionary
  - Key: Name
  - Value: Class year
- User interaction (given that dictionary)
  - Check if the name is in the dictionary.
  - If so, index their name in the dictionary to get the class year
  - If not, report an error

# Algorithm: Parse Data File

```
Name1 2025  
Name2 2026  
Name3 2024  
Name4 2026  
Name5 2024  
...
```

- 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/mappings in the dictionary)
- for testing only: Display dictionary, in sorted order
- Return dictionary

# Looking Ahead

- Lab 8: Due Friday
- No broader issue due this week