Objectives

- Continuing with dictionaries

Review: Dictionaries

- What is a dictionary in Python?
- What is the syntax for creating a new dictionary?
- How do we access a key's value from a dictionary? (2 ways)
  - What happens if there is no mapping for that key?
- How do we create a key → value mapping in a dictionary?
- How can we iterate through a dictionary?
- What other operations can we perform on dictionaries?
- Review the solution to the problem where we looked up student's class years

Review: Creating Dictionaries in Python

Syntax:

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

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

Review: 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

Review: Dictionary Methods

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

Review: Accessing Values Using Indexing

- Syntax:

  `<dictionary>[<key>]`

- Examples:

```
charToAscii['z']

nameToPhoneNum['friendname']
```

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

Review: 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

Review: 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 2027
…
```

- Consider
  - How would we solve this before learning dictionaries?
  - How would we solve this 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?

years_dictionary.py

Algorithm: Parse Data File

```
Name1 2027
Name2 2026
Name3 2024
Name4 2026
Name5 2025
…
```

- 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

Problem

Example file:

```
Name1 2027
Name2 2026
Name3 2024
Name4 2026
Name5 2025
...
```

- Given a file of the form
  - <firstname> <classyear>
- Goal: Report the *number* of students in each graduation year, ex:

```
2024 1
2025 6
2026 8
2027 7
```

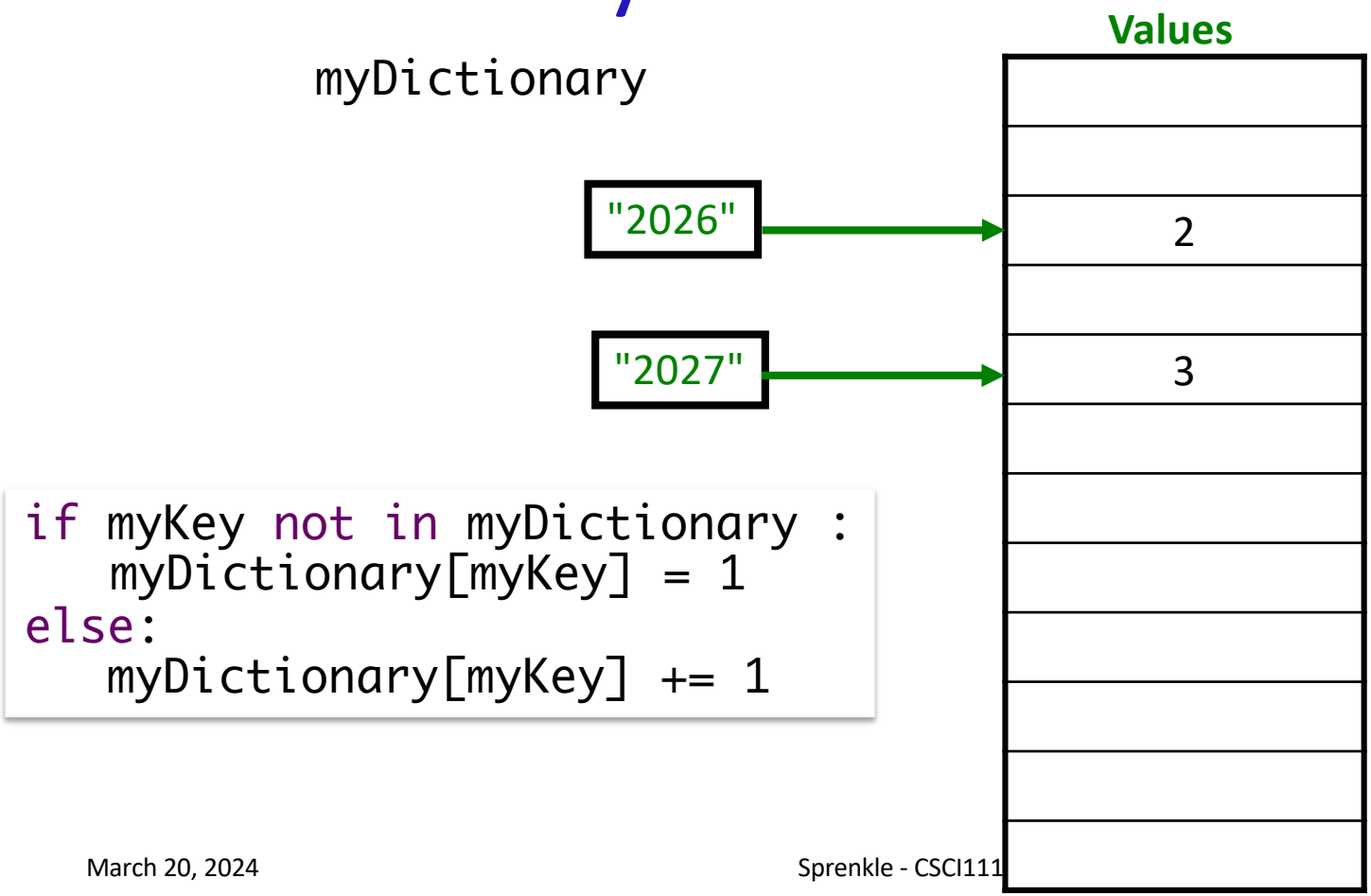
- Problem-solving Approach:
  - How should we model the data?
  - Pretend you are the computer, how would you solve this problem?

Equivalent Solutions:
A Dictionary of Accumulators

```
if myKey not in myDictionary :
    myDictionary[myKey] = 1
else:
    count = myDictionary[myKey] + 1
    myDictionary[myKey] = count
```

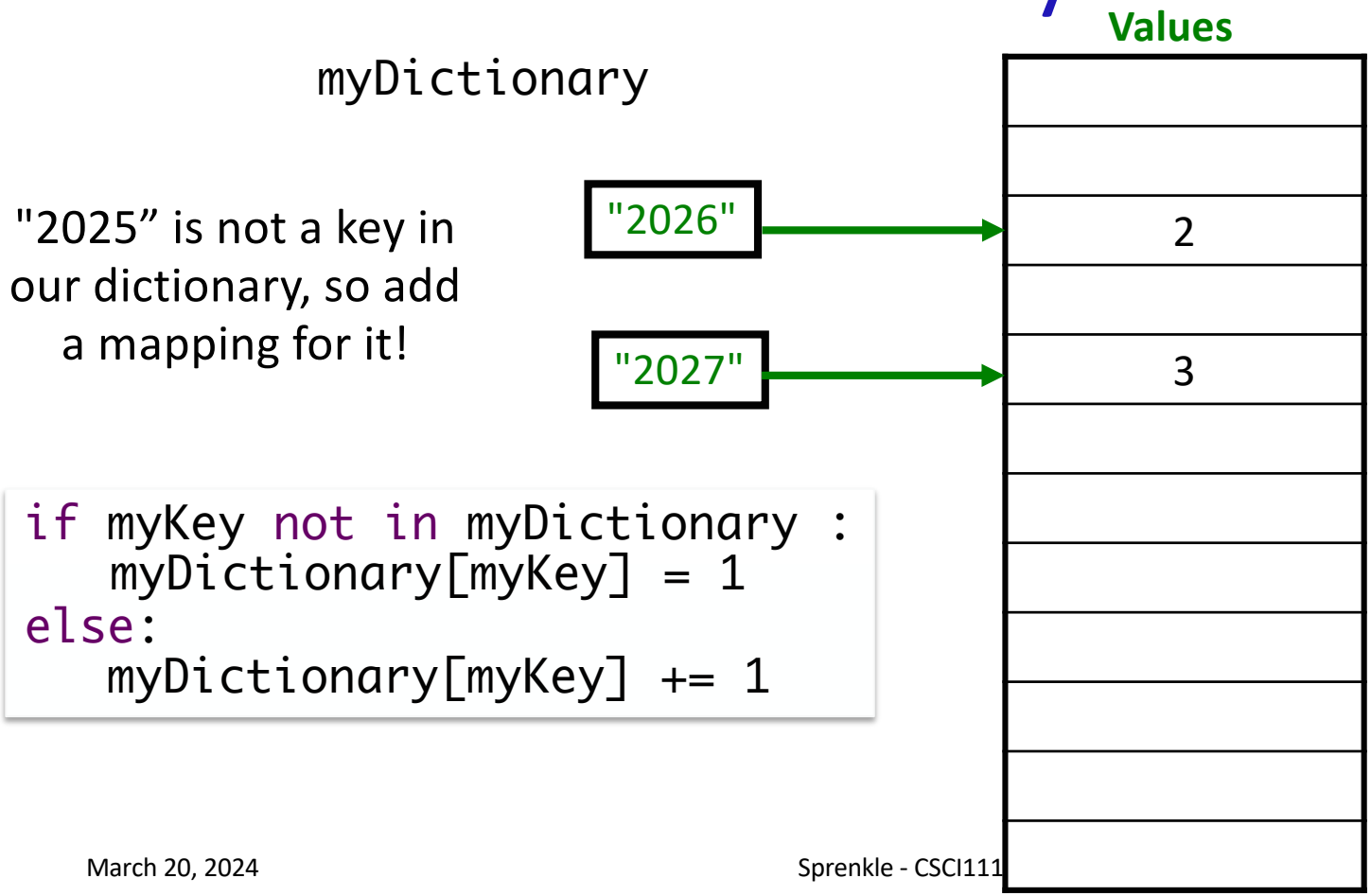
```
if myKey not in myDictionary :
    myDictionary[myKey] = 1
else:
    myDictionary[myKey] += 1
```

Our Dictionary



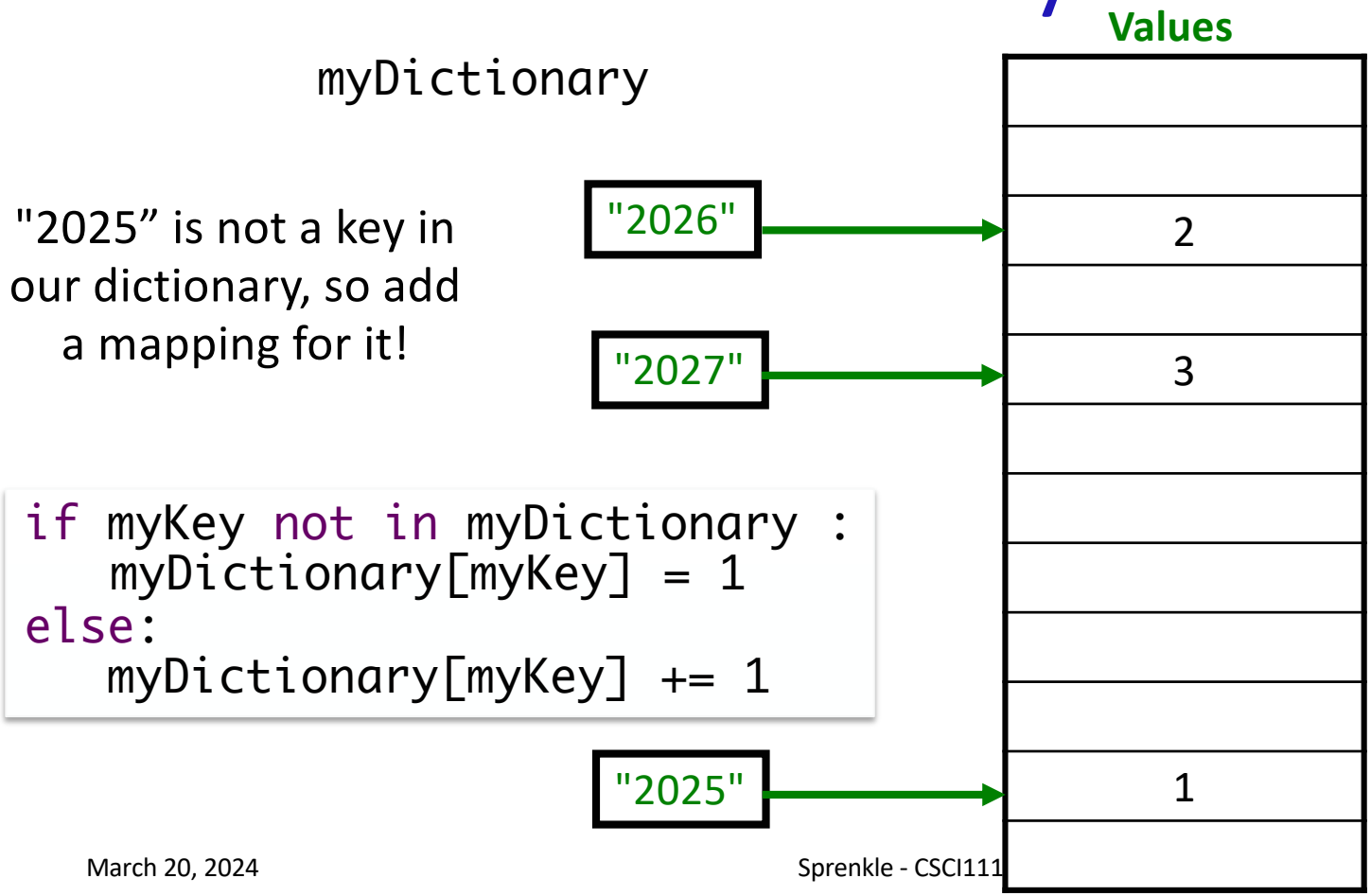

```
if myKey not in myDictionary :
    myDictionary[myKey] = 1
else:
    myDictionary[myKey] += 1
```

Scenario 1: A new class year




```
if myKey not in myDictionary :
    myDictionary[myKey] = 1
else:
    myDictionary[myKey] += 1
```

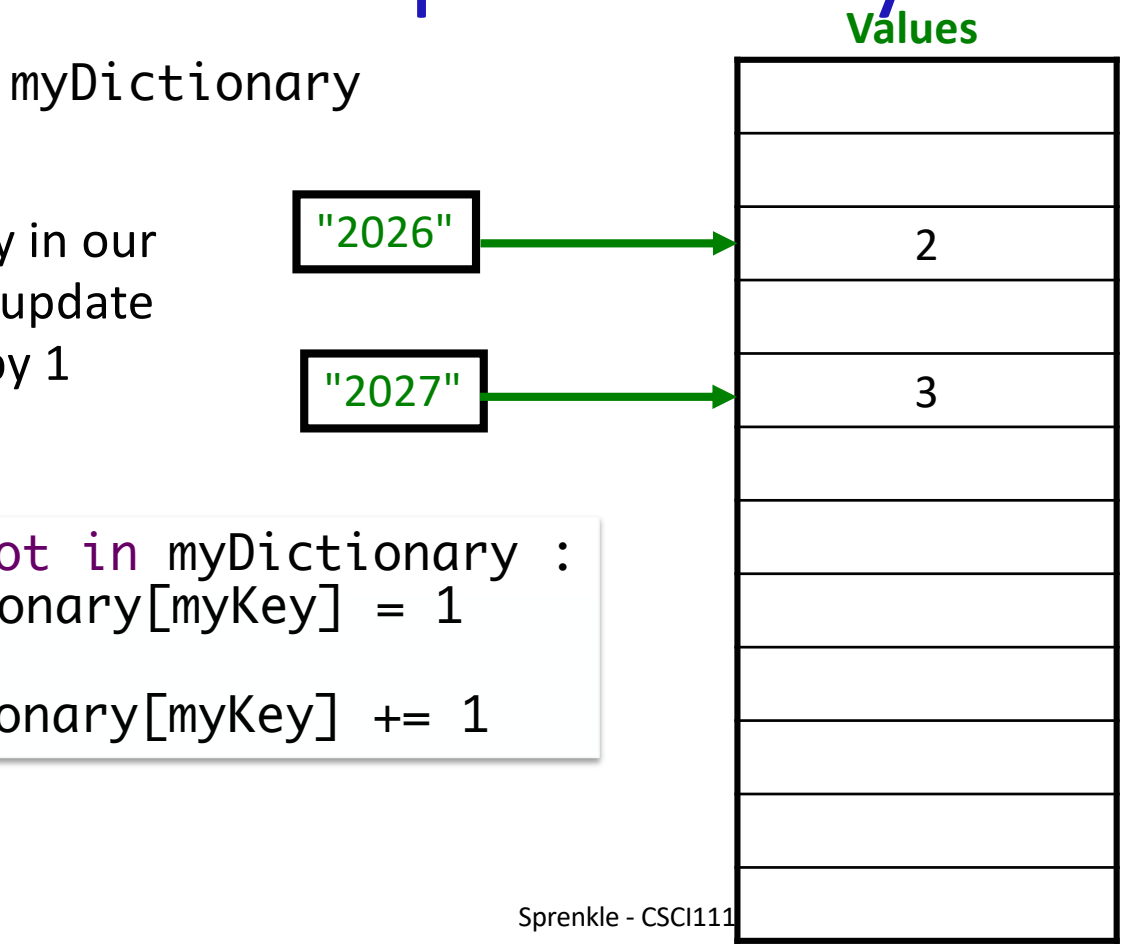
Scenario 1: A new class year




```
if myKey not in myDictionary :
    myDictionary[myKey] = 1
else:
    myDictionary[myKey] += 1
```

Scenario 2: A repeat class year

"2026" is a key in our dictionary, so update its count by 1

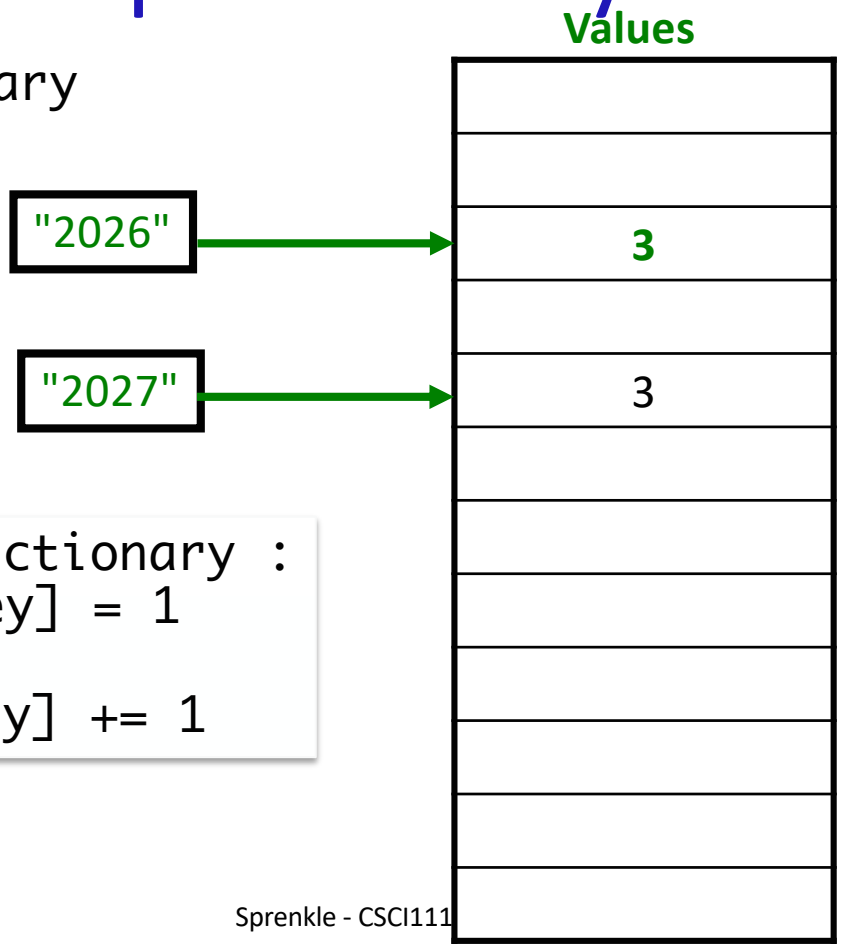



```
if myKey not in myDictionary :
    myDictionary[myKey] = 1
else:
    myDictionary[myKey] += 1
```

Scenario 2: A repeat class year

myDictionary

"2026" is a key in our dictionary, so update its count by 1




```
if myKey not in myDictionary :
    myDictionary[myKey] = 1
else:
    myDictionary[myKey] += 1
```

Discussion: Comparing Lists and Dictionaries

- What are their structures? Properties?
- How are they similar?
- How are they different?
- When do you use one or the other?

Lists vs. Dictionaries

Lists	Dictionaries
integer positions (0, …) to any type of value	Map immutable keys (int, float, string) to any type of value
Ordered	Unordered
Slower to find a value (`in` or `find`)	Fast to find a value (use key)
Fast to print in order	Slower to print in order (by key)
Only as big as you make it	Takes up a lot of space (so can add elements in the middle)

Why Dictionaries?

- An alternative 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 looking through a list if a lot of elements	• Requires a lot of space, • Requires unique, immutable keys

Looking Ahead

- Pre Lab 9
  - Dictionaries, Classes (Monday)
  - Fewer exercises, fewer opportunities to confirm your understanding
- Wednesday – Sprenkle away all day
  - Tuesday: some exam review during lab
  - Wednesday: time to work on lab and review for exam
- Friday – Exam 2
  - Preparation document online