

Python For Everyone, Enhanced eText

Cay S. Horstmann; Rance D. Necaise

Expand | Collapse

8. Sets and Dictionaries357

8.1 Sets358

8.1.1 Creating and Using Sets358

Self Check359

8.1.2 Adding and Removing Elements359

Self Check360

8.1.3 Subsets360

Self Check361

8.1.4 Set Union, Intersection, and Difference361

Self Check364

Worked Example 8.1 Counting Unique Words364

Programming Tip 8.1 Use Python Sets, Not Lists, for Efficient Set Operations366

Special Topic 8.1 Hashing367

Computing & Society 8.1 Standardization368

8.2 Dictionaries368

8.3 Complex Structures378

Chapter Summary391

Interactive Review and Practice392

End-of-Chapter ExercisesEX8-1

9. Objects and Classes393

10. Inheritance443

```
for character in cast :
    print(character)
```

Note that the order in which the elements of the set are visited depends on how they are stored internally. For example, the loop above displays the following:


The cast of characters includes:  
Gumbys  
Spiny  
Luigi

Note that the order of the elements in the output is different from the order in which the set was created. (See [Special Topic 8.1](#) for more information on the ordering used by sets.)

The fact that sets do not retain the initial ordering is not a problem when working with sets. In fact, the lack of an ordering makes it possible to implement set operations very efficiently.

However, you usually want to display the elements in sorted order. Use the `sorted` function, which returns a list (not a set) of the elements in sorted order. The following loop prints the cast in sorted order:

```
for character in sorted(cast) :
    print(character)
```

**SELF CHECK**

1. Which of the following statements does *not* describe a set?

☐ A set is a container that stores a collection of values.

☐ A set stores unique values.

☒ The elements of a set are stored in linear order.

☐ The elements of a set cannot be accessed by position.

One correct, 0 errors, 100%

2. Rearrange the following lines of code to produce a function that prints the even values contained in a set of integers. The set is supplied as an argument to the function. Not all lines are used in the solution.

GOOD JOB! ✓

```
def printEvenValues(values) :
    for element in values :
        if element % 2 == 0 :
            print(element)
```

```
for i in range(len(values)) :
    print(element[i])
if element.iseven() :
```

3 correct, 0 errors, 100%, 45 seconds

Start over

Back to Page

358 / 554

Python For Everyone, Enhanced  
eText  
Cay S. Horstmann; Rance D. Necaise

Expand | Collapse

8. Sets and Dictionaries357

8.1 Sets358

8.1.1 Creating and Using Sets358

Self Check359

8.1.2 Adding and Removing Elements359

Self Check360

8.1.3 Subsets360

Self Check361

8.1.4 Set Union, Intersection, and Difference361

Self Check364

Worked Example 8.1 Counting Unique Words364

Programming Tip 8.1 Use Python Sets, Not Lists, for Efficient Set Operations366

Special Topic 8.1 Hashing367

Computing & Society 8.1 Standardization368

8.2 Dictionaries368

8.3 Complex Structures378

Chapter Summary391

Interactive Review and Practice392

End-of-Chapter ExercisesEX8-1

9. Objects and Classes393

10. Inheritance443

11. Recursion489

12. Sorting and Searching525

AppendicesA-1

3. How many elements are contained in the set `courses` after the following code segment is executed?

```
courses = {"cs101", "math101", "hist201"}
courses.add("lit112")
courses.add("math101")
courses.add("bio132")
```

☐ 3

☐ 4

☒ 5

☐ 6

Item "math101" is added by the second add operation, but because it is already in the set, it is not added again. Sets cannot contain duplicates.

One correct, 0 errors, 100%

4. For each of the following code segments, indicate what is printed or enter error if an exception is raised.

GOOD JOB!

Statements	Result	Explanation
<pre>names = {"John", "Susie", "Robert"} names.add("Amy") names.add("Joe") names.discard("amy") print(len(names))</pre>	5	The discard method does not raise an exception when the element being removed is not in the set.
<pre>names = {"John", "Susie", "Robert"} names.add("Sam") names.remove("amy") print(len(names))</pre>	error	But the remove method does.
<pre>names = {"John", "Susie", "Robert"} names.add("Arthur") names.remove("Robert") names.clear() names.add("Luigi") print(len(names))</pre>	1	The clear method removes all elements from the set.
<pre>names = {"John", "Susie", "Robert"} names.discard("Robert") names.add("June") names.add("amy") names.remove("John") names.add("Robert") if "AMY" in names :     print(len(names)) else :     print(0)</pre>	0	Although the set contains two spellings of the name Amy, neither of the two are spelled with all capital letters.

4 correct, 0 errors, 100%, 65 seconds

Start over

Back to Page

359 / 554

8.1.2

8.1.2 - Adding and Removing Elements (3, 4)

Python For Everyone, Enhanced eText  
Cay S. Horstmann; Rance D. Necaise

Expand | Collapse

8. Sets and Dictionaries357

8.1 Sets358

8.1.1 Creating and Using Sets358

Self Check359

8.1.2 Adding and Removing Elements359

Self Check360

**8.1.3 Subsets360**

Self Check361

8.1.4 Set Union, Intersection, and Difference361

Self Check364

Worked Example 8.1 Counting Unique Words364

Programming Tip 8.1 Use Python Sets, Not Lists, for Efficient Set Operations366

Special Topic 8.1 Hashing367

Computing & Society 8.1 Standardization368

8.2 Dictionaries368

8.3 Complex Structures378

Chapter Summary391

Interactive Review and Practice392

End-of-Chapter ExercisesEXB-1

9. Objects and Classes393

10. Inheritance443

11. Recursion489

12. Sorting and Searching525

AppendicesA-1

```
print("At least one of the colors in the Italian flag does not.")
```

You can also test for equality between two sets using the `==` and `!=` operators. Two sets are equal if and only if they have exactly the same elements.

```
french = { "Red", "White", "Blue" }
if british == french :
    print("The British and French flags use the same colors.")
```

SELF CHECK

5. Consider the following code segment. What is true about these sets?

```
names = {"Jane", "Joe", "Amy", "Lisa"}
names1 = {"Joe", "Amy", "Lisa"}
names2 = {"Jane", "Joe"}
```

☒

 names2 is a subset of names.

☐

 names2 is not a subset of names.

☐

 names1 is not a subset of names.

☐

 names1 is a subset of names2.

One correct, 0 errors, 100%

6. Consider the following set definitions and indicate the result of each statement.

```
basket1 = {"apples", "oranges", "grapes"}
basket2 = {"blueberries", "pears", "apples", "oranges", "lemons"}
basket3 = {"bananas", "lemons", "grapes", "apples"}
basket4 = {"grapes", "apples"}
```

☒

True

False

 basket1.issubset(basket2)

The set basket1 contains an element, "grapes", that is not contained in the set basket2.

☒

True

False

 basket4.issubset(basket1)

All of the elements in basket4 are also in basket1.

☒

True

False

 basket4.issubset(basket3)

All of the elements in basket4 are also in basket3.

☒

True

False

 len(basket3) == len(basket1)

The number of elements in the two sets are not the same.

☒

True

False

 basket4 == basket1

For two sets to be equal, they must contain the same number of elements and the same exact elements.

☒

True

False

 basket1 != basket3

The two sets do not contain the same number of elements.

6 correct, 0 errors, 100%

Back to Page

360 / 554

Python For Everyone, Enhanced eText

Cay S. Horstmann; Rance D. Necaise

Expand | Collapse

End-of-Chapter Exercises EA-1

8. Sets and Dictionaries 357

8.1 Sets 358

8.1.1 Creating and Using Sets 358

Self Check 359

8.1.2 Adding and Removing Elements 359

Self Check 360

8.1.3 Subsets 360

Self Check 361

8.1.4 Set Union, Intersection, and Difference 361

Self Check 364

Worked Example 8.1 Counting Unique Words 364

Programming Tip 8.1 Use Python Sets, Not Lists, for Efficient Set Operations 366

Special Topic 8.1 Hashing 367

Computing & Society 8.1 Standardization 368

8.2 Dictionaries 368

8.3 Complex Structures 378

Chapter Summary 391

Interactive Review and Practice 392

End-of-Chapter Exercises EX-1

9. Objects and Classes 393

10. Iterators 394

VIII  
xii  
yer  
zealand

SELF CHECK

7. What elements are contained in the set `fruit` after the following code segment is executed?

```
basket1 = {"blueberries", "pears", "apples", "oranges", "lemons"}  
basket2 = {"bananas", "lemons", "grapes", "apples"}  
fruit = basket1.union(basket2)
```

☐ {"blueberries", "pears", "oranges"}  
☐ {"lemons", "apples"}  
☒ {"bananas", "lemons", "grapes", "apples", "blueberries", "oranges", "pears"}  
☐ {"blueberries", "pears", "apples", "oranges", "lemons"}

One correct, 0 errors, 100%

8. Test your understanding of set operations with the activity below. For each task, enter the name of the method or function that must be used to complete the task.

GOOD JOB! ✓

Task	Answer	Example Use
Determine the number of elements contained in a set.	<code>len</code>	<code>size = len(myset)</code>
Create an empty set.	<code>set</code>	<code>myset = set()</code>
Given two sets, <code>setA</code> and <code>setB</code> , determine if all of the elements of <code>setA</code> are contained in <code>setB</code> .	<code>issubset</code>	<code>if setA.issubset(setB) :</code>
Remove all of the elements of a set.	<code>clear</code>	<code>myset.clear()</code>
Add a new element to an existing set.	<code>add</code>	<code>myset.add(400)</code>
Remove an element from a set, not knowing if the element is actually in the set.	<code>discard</code>	<code>myset.discard(500)</code>
Given two sets, <code>setA</code> and <code>setB</code> , create a new set that contains all of the unique elements in both <code>setA</code> and <code>setB</code> .	<code>union</code>	<code>newset = setA.union(setB)</code>
Given two sets, <code>setA</code> and <code>setB</code> , create a new set that contains only those elements that are in <code>setA</code> but not in <code>setB</code> .	<code>difference</code>	<code>newset = setA.difference(setB)</code>
Given two sets, <code>setA</code> and <code>setB</code> , create a new set that contains only those elements that are in both <code>setA</code> and <code>setB</code> .	<code>intersection</code>	<code>newset = setA.intersection(setB)</code>

9 correct, 0 errors, 100%, 67 seconds


Start over

Back to Page

361 / 554

8.1.4

8.1.4 - Set Union, Intersection, and Difference (7, 8)



Python For Everyone, Enhanced eText


Cay S. Horstmann; Rance D. Necaise

Expand | Collapse

- 8.2 Dictionaries 368
  - Syntax 8.1 Set and Dictionary Literals 369
  - 8.2.1 Creating Dictionaries 369
  - Self Check 370**
  - 8.2.2 Accessing Dictionary Values 370
  - Self Check 370
  - 8.2.3 Adding and Modifying Items 370
  - Self Check 370
  - 8.2.4 Removing Items 371
  - Self Check 371
  - 8.2.5 Traversing a Dictionary 372
  - Self Check 374
  - Special Topic 8.2 Iterating over Dictionary Items 374
  - Special Topic 8.3 Storing Data Records 375
  - Worked Example 8.2 Translating Text Messages 375
- 8.3 Complex Structures 378
  - Chapter Summary 391
  - Interactive Review and Practice 392
  - End-of-Chapter Exercises EX8-1
- 9. Objects and Classes 393
- 10. Inheritance 443

contacts = { "Fred": 7235591, "Mary": 3841212, "Bob": 3841212, "Sarah": 2213278 }


Each key/value pair is separated by a colon. You enclose the key/value pairs in braces, just as you would when forming a set. When the braces contain key/value pairs, they denote a dictionary, not a set. The only ambiguous case is an empty {}. By convention, it denotes an empty dictionary, not an empty set.



**Figure 8** A Dictionary with Four Entries

You can create a duplicate copy of a dictionary using the dict function:

```
oldContacts = dict(contacts)
```



SELF CHECK

1. Which of the following statements does *not* describe a dictionary?

- ☐ A dictionary is a container that stores associations between keys and values.
- ☐ The keys in a dictionary are unique.
- ☒ The entries in a dictionary can be accessed by position.
- ☐ The entries of a dictionary can be accessed by key.

One correct, 0 errors, 100%

2. Complete the following function, which creates and initializes a dictionary that maps the English words "one" through "five" to the numbers 1 through 5.

```
numbers.py
1
2 ##
3 # Creates and initializes a dictionary that contains elements which maps
4 # the English words "one" through "five" to the numbers 1 through 5.
5 # @return a dictionary containing the indicated mappings
6 #
7 def createDict():
8     names = { "one": 1, "two": 2, "three": 3, "four": 4, "five": 5 }
9     return names
10
```

[CodeCheck](#) [Reset](#)

**Testers**


**Running Tester.py**

pass

five = 5, four = 4, one = 1, three = 3, two = 2,  
Expected: five = 5, four = 4, one = 1, three = 3, two = 2,


**Score**

1/1



Back to Page

370 / 554



Python For Everyone, Enhanced eText

Cay S. Horstmann; Rance D. Necaise

Expand | Collapse

8.2.1 Creating Dictionaries369

Self Check370

8.2.2 Accessing Dictionary Values370

**Self Check370**

8.2.3 Adding and Modifying Items370

Self Check370

8.2.4 Removing Items371

Self Check371

8.2.5 Traversing a Dictionary372

Self Check374

Special Topic 8.2 Iterating over Dictionary Items374

Special Topic 8.3 Storing Data Records375

Worked Example 8.2 Translating Text Messages375

8.3 Complex Structures378

Chapter Summary391

Interactive Review and Practice392

End-of-Chapter ExercisesEX8-1

9. Objects and Classes393

10. Inheritance443


11. Recursion489

12. Sorting and Searching525

```
if "John" in contacts :
    print("John's number is", contacts["John"])
else :
    print("John is not in my contact list.")
```

Often, you want to use a default value if a key is not present. For example, if there is no number for Fred, you want to dial the directory assistance number instead. Instead of using the `in` operator, you can simply call the `get` method and pass the key and a default value. The default value is returned if there is no matching key.

```
number = contacts.get("Fred", 411)
print("Dial " + number)
```



3. Given the following dictionary definition

```
favoriteFoods = {"Peg": "burgers", "Cy": "hotdogs", "Bob": "apple pie"}
```

which of the following statements prints Peg's favorite food?

☐ `print("Peg's favorite food is: ", favoriteFoods.get(1))`

☐ `print("Peg's favorite food is: ", favoriteFoods)`

☒ `print("Peg's favorite food is: ", favoriteFoods["Peg"])`

☐ `print("Peg's favorite food is: ", favoriteFoods[1])`

One correct, 0 errors, 100%

4. Consider the following dictionary definition:

```
abbrv = {"OH": "Ohio", "CA": "California", "ME": "Maine", "TX": "Texas"}
```


Indicate the result of each of the following code segments. Enter error if an exception is raised.

GOOD JOB! ✓

Code Segment	Result	Explanation
<code>print(abbrv["ME"])</code>	Maine	The key "ME" maps to the state Maine.
<code>print(abbrv["VA"])</code>	error	An exception is raised when using the subscript operator if the key is not in the dictionary.
<code>print(abbrv.get("TV", "Invalid"))</code>	Invalid	The <code>get</code> method, which can also be used to access a value, does not raise an exception if the key does not exist; it returns the second argument instead.
<code>print(abbrv["Tx"])</code>	error	When strings are used as dictionary keys, they must match exactly.

4 correct, 0 errors, 100%, 31 seconds

Start over



[Back to Page](#)

< 370 / 554 >

Python For Everyone, Enhanced eText

Cay S. Horstmann; Rance D. Necaise

Expand | Collapse

8.2 Dictionaries368

Syntax 8.1 Set and Dictionary Literals369

8.2.1 Creating Dictionaries369

Self Check370

8.2.2 Accessing Dictionary Values370

Self Check370

**8.2.3 Adding and Modifying Items370**

Self Check370

8.2.4 Removing Items371

Self Check371

8.2.5 Traversing a Dictionary372

Self Check374

Special Topic 8.2 Iterating over Dictionary Items374

Special Topic 8.3 Storing Data Records375

Worked Example 8.2 Translating Text Messages375

8.3 Complex Structures378

Chapter Summary391

Interactive Review and Practice392

End-of-Chapter ExercisesEX8-1

9. Objects and Classes393

10. Inheritance443

Figure 9 Adding and Modifying Dictionary Entries

Sometimes you may not know which items will be contained in the dictionary when it's created. You can create an empty dictionary like this:

```
favoriteColors = {}
```

and add new items as needed:

```
favoriteColors["Juliet"] = "Blue"
favoriteColors["Adam"] = "Red"
favoriteColors["Eve"] = "Blue"
favoriteColors["Romeo"] = "Green"
```

SELF CHECK

5. How many entries are contained in the dictionary courses after the following code segment is executed?

```
courses = {"cs101": "Intro to Computer Science"}
courses["math101"] = "Discrete Mathematics"
courses["cs101"] = "Intro to Programming"
courses["hist201"] = "World History"
```

☐ 1

☐ 2

☒ 3

☐ 4

One correct, 0 errors, 100%

6. Trace through the following program that reads a collection of keys and values from the user and adds them to a dictionary:

GOOD JOB! ✓

```
data = {}
key = int(input("Enter key (0 to quit): "))
while key > 0:
    value = input("Enter value: ")
    data[key] = value
    size = len(data)
    key = int(input("Enter key (0 to quit): "))
print(data[3])
```

key	value	size	Output
3	A	1	
12	X	2	
4	X	3	
3	Y	3	
7	Q	4	
0			Y

12 correct, 0 errors, 100%, 67 seconds

Start over

Back to Page

370 / 554

8.2.3

8.2.3 - Adding and Modifying Items (5, 6)



Python For Everyone, Enhanced eText

Cay S. Horstmann; Rance D. Necaise

Expand | Collapse

8.2 Dictionaries368

Syntax 8.1 Set and Dictionary Literals369

8.2.1 Creating Dictionaries369

Self Check370

8.2.2 Accessing Dictionary Values370

Self Check370

8.2.3 Adding and Modifying Items370

Self Check370

**8.2.4 Removing Items371**

Self Check371

8.2.5 Traversing a Dictionary372

Self Check374

Special Topic 8.2 Iterating over Dictionary Items374

Special Topic 8.3 Storing Data Records375

Worked Example 8.2 Translating Text Messages375

8.3 Complex Structures378

Chapter Summary391

Interactive Review and Practice392

End-of-Chapter ExercisesEX8-1

9. Objects and Classes393

10. Inheritance443

After `contacts.pop("Fred")`

contacts =

"Sarah"2213278

"Bob"3841212

"John"2228182

"Mary"3841212

**Figure 10** Removing a Dictionary Entry

If the key is not in the dictionary, the pop method raises a `KeyError` exception. To prevent the exception from being raised, you can test for the key in the dictionary:

```
if "Fred" in contacts :
    contacts.pop("Fred")
```

SELF CHECK

7. Given the dictionary `courses` that stores course numbers and course titles for the courses taught at a college, which of the following correctly removes the entry for `bio111` without possibly causing an exception?

☐ `courses.pop("bio111")`

☐ `courses.remove("bio111")`

☒

if "bio111" in courses :  
    `courses.pop("bio111")`

☐

if "Cellular Biology" in courses :  
    `courses.remove("bio111")`

One correct, 0 errors, 100%

8. How many entries are contained in the `periodicTable` dictionary after the following code segment is executed?

```
periodicTable = {"H": "Hydrogen", "O": "Oxygen", "N": "Nickel"}
periodicTable["He"] = "Helium"
periodicTable["Au"] = "Gold"
periodicTable["N"] = "Nitrogen"
if "Gold" in periodicTable :
    periodicTable["Cu"] = "Copper"
periodicTable.pop("O")
```

☒ 4

The `in` operator tests for inclusion of a key in the dictionary. The statement `if "Gold" in periodicTable` returns `False` because "Gold" is not a key, it is a value associated with the key "Au".

☐ 5

☐ 6

☐ 7

One correct, 0 errors, 100%

Back to Page

371 / 554

8.2.4

8.2.4 - Removing Items (7, 8)



Python For Everyone, Enhanced  
eText  
Cay S. Horstmann; Rance D. Necaise

Expand | Collapse

8.2 Dictionaries368

Syntax 8.1 Set and Dictionary Literals369

8.2.1 Creating Dictionaries369

Self Check370

8.2.2 Accessing Dictionary Values370

Self Check370

8.2.3 Adding and Modifying Items370

Self Check370

8.2.4 Removing Items371

Self Check371

**8.2.5 Traversing a Dictionary372**

Self Check374

Special Topic 8.2 Iterating over Dictionary  
Items374

Special Topic 8.3 Storing Data Records375

Worked Example 8.2 Translating Text  
Messages375

8.3 Complex Structures378

Chapter Summary391

Interactive Review and Practice392

End-of-Chapter ExercisesEX8-1

9. Objects and Classes393

10. Inheritance443

11. Recursion489

12. Sorting and Searching525

AppendicesA-1

d.values()

Returns a sequence containing all values in the dictionary.

SELF CHECK

• 9. Given the following dictionary definition

```
favoriteFoods = {"Peg": "burgers", "Cy": "hotdogs", "Bob": "apple pie"}
```

Which code segment correctly prints the dictionary, both the key and value, in alphabetical order by the person's name?

☐ print(favoriteFoods)

☒ for name in sorted(favoriteFoods):  
print(name, favoriteFoods[name])

☐ for name in (favoriteFoods):  
print(name, favoriteFoods[name])

☐ for name in sorted(favoriteFoods):  
print(favoriteFoods[name])

One correct, 0 errors, 100%

•• 10. Complete the following function that builds and returns a list of the values contained in the given dictionary. Do not use the values dictionary method.

```
buildlist.py
1 # Builds and returns a list of the values contained in a dictionary.
2 # @param data a dictionary containing key/value pairs
3 # @return a list containing the values from the dictionary
4 #
5 def buildList(data):
6     list = []
7     for key in data:
8         list.append(data[key])
9     return list
```

CodeCheck

Reset

Testers

Running Tester.py

pass

Fred, John, Mary, Sarah,  
Expected: Fred, John, Mary, Sarah,

Running Tester2.py

pass


apple pie, burgers, chocolate, hotdogs,  
Expected: apple pie, burgers, chocolate, hotdogs,

Score

2/2

Back to Page

372 / 554



Python For Everyone, Enhanced eText  
Cay S. Horstmann; Rance D. Necaise

Expand | Collapse

Syntax 8.1 Set and Dictionary Literals369

8.2.1 Creating Dictionaries369

Self Check370

8.2.2 Accessing Dictionary Values370

Self Check370

8.2.3 Adding and Modifying Items370

Self Check370

8.2.4 Removing Items371

Self Check371

8.2.5 Traversing a Dictionary372

Self Check374

Special Topic 8.2 Iterating over Dictionary Items374

Special Topic 8.3 Storing Data Records375

Worked Example 8.2 Translating Text Messages375

8.3 Complex Structures378

8.3.1 A Dictionary of Sets378

8.3.2 A Dictionary of Lists381

Self Check383


Special Topic 8.4 User Modules383

Worked Example 8.3 Graphics: Pie Charts384

Toolbox 8.1 Harvesting JSON Data from the Web388

Chapter Summary391

Interactive Review and Practice392



1. Consider the following code segment:

```
pets = {}
pets["Snowball"] = [77, 85, 90]
pets["Spike"] = [65, 90, 75]
```

The complex structure `pets` would best be characterized as:

☐ A dictionary of dictionaries

☒ A dictionary of lists

☐ A dictionary of sets

☐ A dictionary of strings

One correct, 0 errors, 100%

2. Suppose you need to design a program in which you create a phone book that includes the name, home phone number, and cell phone number for all of your friends. What type of container or structure would be the best choice for this problem?

☒ a dictionary of dictionaries

☐ a dictionary

☐ a dictionary of lists

☐ a dictionary of sets

One correct, 0 errors, 100%

3. Indicate whether each of the following statements is true or false.

☒ True ☐ False

A dictionary of sets can be used in the `icecreamsales.py` program of Section 8.3.2 to store sales data for the ice cream flavors.

The sales data must be saved in the container in the same order in which it is added so that the data for each store will be in the same position in each list. The elements in a set are not stored in the order in which they are added.

☒ True ☐ False

An exception is raised if we try to create a set of lists.


An exception is raised because a set can only contain elements that are hashable. A list is not hashable (see Special Topic 8.1).

☒ True ☐ False

Suppose the owner of the ice cream stores in Section 8.3.2 wants to find out which flavors sell well in each store. Specifically, for each store, which flavors had sales of at least \$8,000? A list of sets with one set per store would be required to answer this question.

Each set would contain those flavors of ice cream for which there were sales of at least \$8,000.

3 correct, 0 errors, 100%

 Back to Page

< 381 / 554 >

Python For Everyone, Enhanced eText

Cay S. Horstmann; Rance D. Necaise

Expand

Collapse

Self Check

370

8.2.3 Adding and Modifying Items

370

Self Check

370

8.2.4 Removing Items

371

Self Check

371

8.2.5 Traversing a Dictionary

372

Self Check

374

Special Topic 8.2 Iterating over Dictionary Items

374

Special Topic 8.3 Storing Data Records

375

Worked Example 8.2 Translating Text Messages

375

8.3 Complex Structures

378

8.3.1 A Dictionary of Sets

378

8.3.2 A Dictionary of Lists

381

Self Check

383

Special Topic 8.4 User Modules

383

Worked Example 8.3 Graphics: Pie Charts

384

Toolbox 8.1 Harvesting JSON Data from the Web

388

The sales data must be saved in the container in the same order in which it is added so that the data for each store will be in the same position in each list. The elements in a set are not stored in the order in which they are added.

True

False

An exception is raised if we try to create a set of lists.

An exception is raised because a set can only contain elements that are hashable. A list is not hashable (see Special Topic 8.1).

True

False

Suppose the owner of the ice cream stores in Section 8.3.2 wants to find out which flavors sell well in each store. Specifically, for each store, which flavors had sales of at least \$8,000? A list of sets with one set per store would be required to answer this question.

Each set would contain those flavors of ice cream for which there were sales of at least \$8,000.

3 correct, 0 errors, 100%

4. Your task is to design and implement a function that can be used with the book index created by the `buildindex.py` program at the end of Section 8.3.1. The function, `findMostFrequentWord`, should take the index listing as an argument, which is represented by a dictionary of sets, and return the word that occurs on the most pages within the book. If multiple words occur on the most number of pages, then return any one of the words.

mostfrequent.py

```
1 ## Finds the word that occurs on the most number of pages in a book index.
2 # @param entries a dictionary containing the entries of the index.
3 # @return the word that occurs most frequently or the empty string if
4 # the dictionary is empty.
5
6 def findMostFrequentWord(entries):
7     mostFrequent = ""
8     wordCounter = 0
9     for key in entries:
10         if len(entries[key]) > wordCounter:
11             wordCounter = len(entries[key])
12             mostFrequent = key
13     return mostFrequent
```

CodeCheck

Reset

Calling with Arguments

	Name	Arguments	Actual	Expected
pass	findMostFrequentWord	{"example": {7, 10}, "index": {7}, "program": {7, 11, 15}, "type": {6, 8}, "set": {20}}	program	program
pass	findMostFrequentWord	{"blue": {1, 12, 15}, "red": {5, 12, 20, 25}, "green": {15, 25}}	red	red

Score

2/2

Back to Page

381 / 554

8.3.1/2\_

8.3.1/2\_ - A Dictionary of Sets/Lists (4)