

Dictionaries

Chapter 7

This chapter covers

- Defining a dictionary
- Using dictionary operations
- Determining what can be used as a key
- Creating sparse matrices
- Using dictionaries as caches
- Trusting the efficiency of dictionaries

What is a dictionary?

- Dictionaries access values by means of integers, strings, or other Python objects called keys, which indicate where in the dictionary a given value is found.
- Both lists and dictionaries can store objects of any type.
- Values stored in a dictionary are not implicitly ordered relative to one another because dictionary keys aren't just numbers.

```
1  # dictionaries
2  x = {1: "one", 2: "two"}
3  print(x)
```

Dictionaries

- A dictionary is a way of mapping from one set of arbitrary objects to an associated but equally arbitrary set of objects.

```
20 english_to_french = {}  
21 english_to_french['red'] = 'rouge'  
22 english_to_french['blue'] = 'bleu'  
23 english_to_french['green'] = 'vert'  
24 print("red is", english_to_french['red'])
```

Other dictionary operations

Dictionary operation	Explanation	Example
<code>{}</code>	Creates an empty dictionary	<code>x = {}</code>
<code>len</code>	Returns the number of entries in a dictionary	<code>len(x)</code>
<code>keys</code>	Returns a view of all keys in a dictionary	<code>x.keys()</code>
<code>values</code>	Returns a view of all values in a dictionary	<code>x.values()</code>
<code>items</code>	Returns a view of all items in a dictionary	<code>x.items()</code>
<code>del</code>	Removes an entry from a dictionary	<code>del(x[key])</code>
<code>in</code>	Tests whether a key exists in a dictionary	<code>'y' in x</code>
<code>get</code>	Returns the value of a key or a configurable default	<code>x.get('y', None)</code>
<code>copy</code>	Makes a shallow copy of a dictionary	<code>y = x.copy()</code>



QUICK CHECK

- Assume that you have a dictionary `x = {'a':1, 'b':2, 'c':3, 'd':4}` and a dictionary `y = {'a':6, 'e':5, 'f':6}`. What would be the contents of `x` after the following snippets of code have executed?:

```
del x['d']
```

```
z = x.setdefault('g', 7)
```

```
x.update(y)
```

Word counting

```
26  # word counting
27  sample_string = "To be or not to be"
28  occurrences = {}
29  ✓ for word in sample_string.split():
30      |     occurrences[word] = occurrences.get(word, 0) + 1
31
32  ✓ for word in occurrences:
33      |     print("The word", word, "occurs", occurrences[word], "times in the string")
```

What can be used as a key?

Python type	Immutable?	Hashtable?	Dictionary key?
int	Yes	Yes	Yes
float	Yes	Yes	Yes
boolean	Yes	Yes	Yes
complex	Yes	Yes	Yes
str	Yes	Yes	Yes
bytes	Yes	Yes	Yes
bytearray	No	No	No
list	No	No	No
tuple	Yes	Sometimes	Sometimes
set	No	No	No

Sparse matrices

- In mathematical terms, a matrix is a two-dimensional grid of numbers, usually written in textbooks as a grid with square brackets on each side.
- A fairly standard way to represent such a matrix is by means of a list of lists.
- To implement sparse matrices by using dictionaries with tuple indices.

```
36 matrix = [[3, 0, -2, 11], [0, 9, 0, 0], [0, 7, 0, 0], [0, 0, 0, -5]]
37
38 matrix = {(0, 0): 3, (0, 2): -2, (0, 3): 11, (1, 1): 9, (2, 1): 7, (3, 3): -5}
```

Efficiency of dictionaries

- The truth is that the Python dictionary implementation is quite fast.
- Many of the internal language features rely on dictionaries, and a lot of work has gone into making them efficient.
- Because all of Python's data structures are heavily optimized, you shouldn't spend much time worrying about which is faster or more efficient.
- If the problem can be solved more easily and cleanly by using a dictionary than by using a list, do it that way, and consider alternatives only if it's clear that dictionaries are causing an unacceptable slowdown.

Lab

Using Dictionaries

Summary

- Dictionaries are powerful data structures, used for many purposes even within Python itself.
- Dictionary keys must be immutable, but any immutable object can be a dictionary key.
- Using keys means accessing collections of data more directly and with less code than many other solutions.