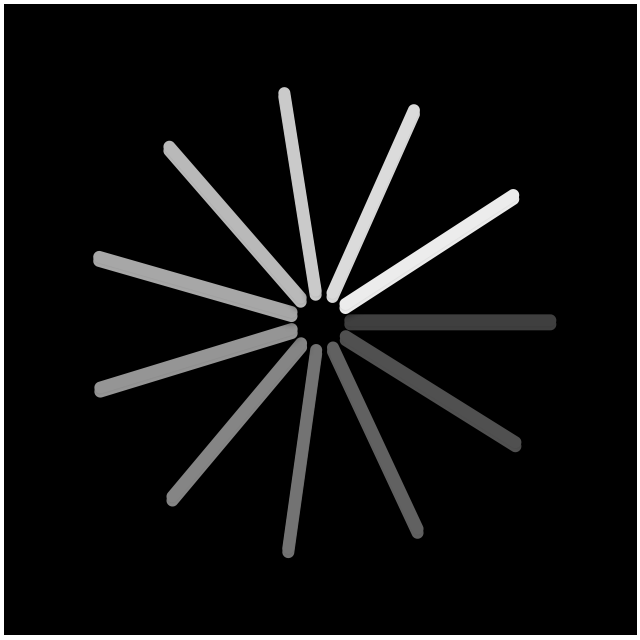
Dictionaries

Let's begin by watching the video "Dictionaries" (4:32).



The second major Python data structure is the dictionary. Dictionaries differ from lists in that you can access items in a dictionary by a key rather than a position. The thing that is most important to notice right now is that the get item and set item operations on a dictionary are $O(1)$. Another important dictionary operation is the contains operation. Checking to see whether a key is in the dictionary or not is also $O(1)$. The efficiency of all dictionary operations is summarized in the table. One important side note on dictionary performance is that the efficiency we provide in the table are for average performance. In some rare cases the contains, get item, and set item operations can degenerate into $O(n)$ performance but we will get into that when we talk about the different ways that a dictionary could be implemented.

Efficiency of Dictionary Operations

Operation	**Big-O Efficiency**
copy	O(n)
get item	O(1)
set item	O(1)
delete item	O(1)
contains (in)	O(1)
iteration	O(n)

Contains (in) Operation - Lists vs. Dictionaries

For our last performance experiment we will compare the performance of the contains operation between lists and dictionaries. In the process we will confirm that the contains operator for lists is $O(n)$ and the contains operator for dictionaries is $O(1)$. The experiment we will use to compare the two is simple. We'll make a list with a range of numbers in it. Then we will pick numbers at random and check to see if the numbers are in the list. If our performance tables are correct the bigger the list the longer it should take to determine if any one number is contained in the list.

We will repeat the same experiment for a dictionary that contains numbers as the keys. In this experiment we should see that determining whether or not a number is in the dictionary is not only much faster, but the time it takes to check should remain constant even as the dictionary grows larger.

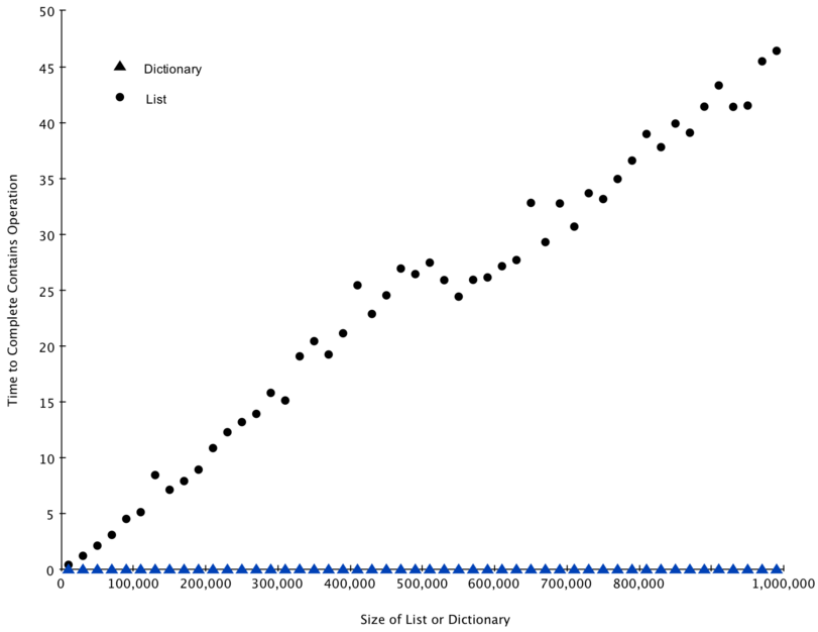
Listing 6 implements this comparison. Notice that we are performing exactly the same operation, `number in container`. The difference is that on line 7 `x` is a list, and on line 9 `x` is a dictionary.

Lists and Dictionaries Comparison

Listing 6

```
1
2  import timeit
   import random
3
3  for i in range(10000,1000001
   ,20000):
4      t = timeit.Timer("rando
   m.randrange(%d) in x"%i,
5                       "from _
   _main__ import random,x")
6      x = list(range(i))
       lst_time = t.timeit(numb
7  er=1000)
       x = {j:None for j in ran
8  ge(i)}
       d_time = t.timeit(number
9  =1000)
1      print("%d,%10.3f,%10.3f"
0  % (i, lst_time, d_time))
1
1
```

Our chart below summarizes the results of running Listing 6. You can see that the dictionary is consistently faster. For the smallest list size of 10,000 elements a dictionary is 89.4 times faster than a list. For the largest list size of 990,000 elements the dictionary is 11,603 times faster! You can also see that the time it takes for the contains operator on the list grows linearly with the size of the list. This verifies the assertion that the contains operator on a list is $O(n)$. It can also be seen that the time for the contains operator on a dictionary is constant even as the dictionary size grows. In fact for a dictionary size of 10,000 the contains operation took 0.004 milliseconds and for the dictionary size of 990,000 it also took 0.004 milliseconds.



Source: Problem Solving and Algorithms in Python (http://interactivepython.org/runestone/static/pythonds/index.html#) from Bradley Miller on www.interactivepython.org (http://interactivepython.org/runestone/static/pythonds/index.html#).