12. Dictionaries

12.1. Dictionaries

12.2. Dictionary Operations

12.3. Dictionary Methods

12.4. Aliasing and Copying

12.6. Glossary

12.7. Exercises

12.1. Dictionaries

All of the compound data types we have studied in detail so far — strings, lists, and tuples — are sequential collections. This means that the items in the collection are ordered from left to right and they use integers as indices to access the values they contain.

Dictionaries are a different kind of collection. They are Python’s built-in mapping type. A map is an unordered, associative collection. The association, or mapping, is from a key, which can be any immutable type, to a value, which can be any Python data object.

As an example, we will create a dictionary to translate English words into Spanish. For this dictionary, the keys are strings and the values will also be strings.

One way to create a dictionary is to start with the empty dictionary and add key-value pairs. The empty dictionary is denoted {}

1 eng2sp = {}

2 eng2sp['one'] = 'uno'

3 eng2sp['two'] = 'dos'

4 eng2sp['three'] = 'tres'

The first assignment creates an empty dictionary named eng2sp. The other assignments add new key-value pairs to the dictionary. The left hand side gives the dictionary and the key being associated. The right hand side gives the value being associated with that key. We can print the current value of the dictionary in the usual way. The key-value pairs of the dictionary are separated by commas. Each pair contains a key and a value separated by a colon.

The order of the pairs may not be what you expected. Python uses complex algorithms, designed for very fast access, to determine where the key-value pairs are stored in a dictionary. For our purposes we can think of this ordering as unpredictable.

Another way to create a dictionary is to provide a list of key-value pairs using the same syntax as the previous output.

1 eng2sp = {'three': 'tres', 'one': 'uno', 'two': 'dos'}

2 print(eng2sp)

It doesn’t matter what order we write the pairs. The values in a dictionary are accessed with keys, not with indices, so there is no need to care about ordering.

Here is how we use a key to look up the corresponding value.

1 eng2sp = {'three': 'tres', 'one': 'uno', 'two': 'dos'}

2

3 value = eng2sp['two']

4 print(value)

The key 'two' yields the value 'dos'.

Check your understanding

dict-1-5: A dictionary is an unordered collection of key-value pairs.

A. False

B. True

dict-1-6: What is printed by the following statements?

mydict = {"cat":12, "dog":6, "elephant":23}

print(mydict["dog"])

A. 12

B. 6

C. 23

D. Error, you cannot use the index operator with a dictionary.

12.2. Dictionary Operations

The del statement removes a key-value pair from a dictionary. For example, the following dictionary contains the names of various fruits and the number of each fruit in stock. If someone buys all of the pears, we can remove the entry from the dictionary.

1 inventory = {'apples': 430, 'bananas': 312, 'oranges': 525, 'pears': 217}

2

3 del inventory['pears']

Dictionaries are also mutable. As we’ve seen before with lists, this means that the dictionary can be modified by referencing an association on the left hand side of the assignment statement. In the previous example, instead of deleting the entry for pears, we could have set the inventory to 0.

1 inventory = {'apples': 430, 'bananas': 312, 'oranges': 525, 'pears': 217}

2

3 inventory['pears'] = 0

Similarily, a new shipment of 200 bananas arriving could be handled like this.

1 inventory = {'apples': 430, 'bananas': 312, 'oranges': 525, 'pears': 217}

2 inventory['bananas'] = inventory['bananas'] + 200

3

4

5 numItems = len(inventory)

Notice that there are now 512 bananas—the dictionary has been modified. Note also that the len function also works on dictionaries. It returns the number of key-value pairs:

Check your understanding

dict-2-4: What is printed by the following statements?

mydict = {"cat":12, "dog":6, "elephant":23}

mydict["mouse"] = mydict["cat"] + mydict["dog"]

print(mydict["mouse"])

A. 12

B. 0

C. 18

D. Error, there is no entry with mouse as the key.

12.3. Dictionary Methods

Dictionaries have a number of useful built-in methods. The following table provides a summary and more details can be found in the Python Documentation.

Method

Parameters

Description

keys

none

Returns a view of the keys in the dictionary

values

none

Returns a view of the values in the dictionary

items

none

Returns a view of the key-value pairs in the dictionary

get

key

Returns the value associated with key; None otherwise

get

key,alt

Returns the value associated with key; alt otherwise

The keys method returns what Python 3 calls a view of its underlying keys. We can iterate over the view or turn the view into a list by using the list conversion function.

1 inventory = {'apples': 430, 'bananas': 312, 'oranges': 525, 'pears': 217}

2

​3 for akey in inventory.keys(): # the order in which we get the keys is not defined

4 print("Got key", akey, "which maps to value", inventory[akey])

5

​6 ks = list(inventory.keys())

7 print(ks)

8

It is so common to iterate over the keys in a dictionary that you can omit the keys method call in the for loop — iterating over a dictionary implicitly iterates over its keys.

1 inventory = {'apples': 430, 'bananas': 312, 'oranges': 525, 'pears': 217}

2

​ 3 for k in inventory:

4 print("Got key", k)

5

​

As we saw earlier with strings and lists, dictionary methods use dot notation, which specifies the name of the method to the right of the dot and the name of the object on which to apply the method immediately to the left of the dot. The empty parentheses in the case of keys indicate that this method takes no parameters.

The values and items methods are similar to keys. They return view objects which can be turned into lists or iterated over directly. Note that the items are shown as tuples containing the key and the associated value.

1 inventory = {'apples': 430, 'bananas': 312, 'oranges': 525, 'pears': 217}

2

​3 print(list(inventory.values()))

4 print(list(inventory.items()))

5

​6f or (k,v) in inventory.items():

7 print("Got", k, "that maps to", v)

8

​9 for k in inventory:

10 print("Got", k, "that maps to", inventory[k])

11

Note that tuples are often useful for getting both the key and the value at the same time while you are looping. The two loops do the same thing.

The in and not in operators can test if a key is in the dictionary:

1 inventory = {'apples': 430, 'bananas': 312, 'oranges': 525, 'pears': 217}

2 print('apples' in inventory)

3 print('cherries' in inventory)

4

​5 if 'bananas' in inventory:

6 print(inventory['bananas'])

7 else:

8 print("We have no bananas")

9

​

This operator can be very useful since looking up a non-existent key in a dictionary causes a runtime error.

The get method allows us to access the value associated with a key, similar to the [ ] operator. The important difference is that get will not cause a runtime error if the key is not present. It will instead return None. There exists a variation of get that allows a second parameter that serves as an alternative return value in the case where the key is not present. This can be seen in the final example below. In this case, since “cherries” is not a key, return 0 (instead of None).

1 inventory = {'apples': 430, 'bananas': 312, 'oranges': 525, 'pears': 217}

2

​3 print(inventory.get("apples"))

4 print(inventory.get("cherries"))

5

​6 print(inventory.get("cherries", 0))

7

​

Check your understanding

dict-3-7: What is printed by the following statements?

mydict = {"cat":12, "dog":6, "elephant":23, "bear":20}

keylist = list(mydict.keys())

keylist.sort()

print(keylist[3])

A. cat

B. dog

C. elephant

D. bear

dict-3-8: What is printed by the following statements?

mydict = {"cat":12, "dog":6, "elephant":23, "bear":20}

answer = mydict.get("cat") // mydict.get("dog")

print(answer)

A. 2

B. 0.5

C. bear

D. Error, divide is not a valid operation on dictionaries.

dict-3-9: What is printed by the following statements?

mydict = {"cat":12, "dog":6, "elephant":23, "bear":20}

print("dog" in mydict)

A. True

B. False

dict-3-10: What is printed by the following statements?

mydict = {"cat":12, "dog":6, "elephant":23, "bear":20}

print(23 in mydict)

A. True

B. False

dict-3-11: What is printed by the following statements?

total = 0

mydict = {"cat":12, "dog":6, "elephant":23, "bear":20}

for akey in mydict:

if len(akey) > 3:

total = total + mydict[akey]

print(total)

A. 18

B. 43

C. 0

D. 61

12.4. Aliasing and Copying

Because dictionaries are mutable, you need to be aware of aliasing (as we saw with lists). Whenever two variables refer to the same dictionary object, changes to one affect the other. For example, opposites is a dictionary that contains pairs of opposites.

1 opposites = {'up': 'down', 'right': 'wrong', 'true': 'false'}

2 alias = opposites

3

​4 print(alias is opposites)

5

​6 alias['right'] = 'left'

7 print(opposites['right'])

8

​

As you can see from the is operator, alias and opposites refer to the same object.

If you want to modify a dictionary and keep a copy of the original, use the dictionary copy method. Since acopy is a copy of the dictionary, changes to it will not effect the original.

acopy = opposites.copy()

acopy['right'] = 'left' # does not change opposites

Check your understanding

dict-4-2: What is printed by the following statements?

mydict = {"cat":12, "dog":6, "elephant":23, "bear":20}

yourdict = mydict

yourdict["elephant"] = 999

print(mydict["elephant"])

A. 23

B. None

C. 999

D. Error, there are two different keys named elephant.

12.6. Glossary

dictionary

A collection of key-value pairs that maps from keys to values. The keys can be any immutable type, and the values can be any type.

key

A data item that is mapped to a value in a dictionary. Keys are used to look up values in a dictionary.

key-value pair

One of the pairs of items in a dictionary. Values are looked up in a dictionary by key.

mapping type

A mapping type is a data type comprised of a collection of keys and associated values. Python’s only built-in mapping type is the dictionary. Dictionaries implement the associative array abstract data type.

12.7. Exercises

Write a program that allows the user to enter a string. It then prints a table of the letters of the alphabet in alphabetical order which occur in the string together with the number of times each letter occurs. Case should be ignored. A sample run of the program might look this this:

Please enter a sentence: ThiS is String with Upper and lower case Letters.

a 2

c 1

d 1

e 5

g 1

h 2

i 4

l 2

n 2

o 1

p 2

r 4

s 5

t 5

u 1

w 2

$

Give the Python interpreter’s response to each of the following from a continuous interpreter session:

>>> d = {'apples': 15, 'bananas': 35, 'grapes': 12}

>>> d['banana']

>>> d['oranges'] = 20

>>> len(d)

>>> 'grapes' in d

>>> d['pears']

>>> d.get('pears', 0)

>>> fruits = d.keys()

>>> fruits.sort()

>>> print(fruits)

>>> del d['apples']

>>> 'apples' in d

Be sure you understand why you get each result. Then apply what you have learned to fill in the body of the function below, and add code for the tests indicated:

1 def add\_fruit(inventory, fruit, quantity=0):

2 pass

3

​4 # make these tests work...

5 new\_inventory = {}

6 add\_fruit(new\_inventory, 'strawberries', 10)

7 # test that 'strawberries' in new\_inventory

8 # test that new\_inventory['strawberries'] is 10

9 add\_fruit(new\_inventory, 'strawberries', 25)

10# test that new\_inve ntory['strawberries'] is now 35)

11

​

Write a program called alice\_words.py that creates a text file named alice\_words.txt containing an alphabetical listing of all the words, and the number of times each occurs, in the text version of Alice’s Adventures in Wonderland. (You can obtain a free plain text version of the book, along with many others, from http://www.gutenberg.org.)

How many times does the word, alice, occur in the book? I

What is the longest word in Alice in Wonderland? How many characters does it have?

Write a function named translator that takes a parameter containing a sentence in English (no punctuation and all words in lowercase) and returns that sentence translated to Pirate.

For example, the sentence “hello there students” should be translated to “avast there swabbies”.