16.4. Sorting a Dictionary

Previously, you have used a dictionary to accumulate counts, such as the frequencies of letters or words in a text. For example, the following code counts the frequencies of different numbers in the list.

Save & Run Original - 1 of 1 Show in CodeLens

```
L = ['E', 'F', 'B', 'A', 'D', 'I', 'I', 'C', 'B', 'A', 'D', 'D', 'E', 'D']

d = {}
for x in L:
    if x in d:
        d[x] = d[x] + 1
    else:
        d[x] = 1
for x in d.keys():
    print("{} appears {} times".format(x, d[x]))
```

```
E appears 2 times
F appears 1 times
B appears 2 times
A appears 2 times
D appears 4 times
I appears 2 times
C appears 1 times
```

Activity: 1 -- ActiveCode (ac18_4_1)

The dictionary's keys are not sorted in any particular order. In fact, you may get a different order of output than someone else running the same code. We can force the results to be displayed in some fixed ordering, by sorting the keys.

Save & Run Original - 1 of 1 Show in CodeLens

```
L = ['E', 'F', 'B', 'A', 'D', 'I', 'I', 'C', 'B', 'A', 'D', 'D', 'E', 'D']

d = {}
for x in L:
    if x in d:
        d[x] = d[x] + 1
    else:
        d[x] = 1
y = sorted(d.keys())
for k in y:
    print("{} appears {} times".format(k, d[k]))
```

```
A appears 2 times
B appears 2 times
C appears 1 times
D appears 4 times
E appears 2 times
F appears 1 times
I appears 2 times
```

Activity: 2 -- ActiveCode (ac18_4_2)

With a dictionary that's maintaining counts or some other kind of score, we might prefer to get the outputs sorted based on the count rather than based on the items. The standard way to do that in python is to sort based on a property of the key, in particular its value in the dictionary.

Here things get a little confusing because we have two different meaning of the word "key". One meaning is a key in a dictionary. The other meaning is the parameter name for the function that you pass into the sorted function.

Remember that the key function always takes as input one item from the sequence and returns a property of the item. In our case, the items to be sorted are the dictionary's keys, so each item is one key from the dictionary. To remind ourselves of that, we've named the parameter in tha lambda expression *k*. The property of key k that is supposed to be returned is its associated value in the dictionary. Hence, we have the lambda expression `lambda k: d[k]`.

Save & Run Original - 1 of 1 Show in CodeLens

```
L = ['E', 'F', 'B', 'A', 'D', 'I', 'I', 'C', 'B', 'A', 'D', 'D', 'E', 'D']

d = {}
for x in L:
    if x in d:
        d[x] = d[x] + 1
```

```
    else:
        d[x] = 1

y = sorted(d.keys(), key=lambda k: d[k], reverse=True)
for k in y:
    print("{} appears {} times".format(k, d[k]))

```

```
D appears 4 times
I appears 2 times
A appears 2 times
B appears 2 times
E appears 2 times
C appears 1 times
F appears 1 times
```

Activity: 3 -- ActiveCode (ac18_4_5)

Here's a version of that using a named function.

Save & Run | Original - 1 of 1 | Show in CodeLens

```
L = ['E', 'F', 'B', 'A', 'D', 'I', 'I', 'C', 'B', 'A', 'D', 'D', 'E', 'D']

d = {}
for x in L:
    if x in d:
        d[x] = d[x] + 1
    else:
        d[x] = 1

def g(k):
    return d[k]

y =(sorted(d.keys(), key=g, reverse=True))

# now loop through the keys
```

```
D appears 4 times
I appears 2 times
A appears 2 times
B appears 2 times
E appears 2 times
C appears 1 times
F appears 1 times
```

Activity: 4 -- ActiveCode (ac18_4_6)

Note

When we sort the keys, passing a function with `key=lambda x: d[x]` does not specify to sort the keys of a dictionary. The lists of keys are passed as the first parameter value in the invocation of sort. The key parameter provides a function that says *how* to sort them.

An experienced programmer would probably not even separate out the sorting step. And they might take advantage of the fact that when you pass a dictionary to something that is expecting a list, its the same as passing the list of keys.

Save & Run | Original - 1 of 1 | Show in CodeLens

```
L = ['E', 'F', 'B', 'A', 'D', 'I', 'I', 'C', 'B', 'A', 'D', 'D', 'E', 'D']

d = {}
for x in L:
    if x in d:
        d[x] = d[x] + 1
    else:
        d[x] = 1

# now loop through the sorted keys
for k in sorted(d, key=lambda k: d[k], reverse=True):
    print("{} appears {} times".format(k, d[k]))

```

```
D appears 4 times
I appears 2 times
A appears 2 times
B appears 2 times
E appears 2 times
C appears 1 times
F appears 1 times
```

Activity: 5 -- ActiveCode (ac18_4_7)

Eventually, you will be able to read code like that and immediately know what it's doing. For now, when you come across something confusing, like line 11, try breaking it down. The function `sorted` is invoked. Its

first parameter value is a dictionary, which really means the keys of the dictionary. The second parameter, the key function, decorates the dictionary key with a post-it note containing that key's value in dictionary d. The last parameter, True, says to sort in reverse order.

There is another way to sort dictionaries, by calling .items() to extract a sequence of (key, value) tuples, and then sorting that sequence of tuples. But it's better to learn the pythonic way of doing it, sorting the dictionary keys using a key function that takes one key as input and looks up the value in the dictionary.

Check Your Understanding

sort-4-1: Which of the following will sort the keys of d in ascending order of their values (i.e., from lowest to highest)?

```
L = [4, 5, 1, 0, 3, 8, 8, 2, 1, 0, 3, 3, 4, 3]

d = {}
for x in L:
    if x in d:
        d[x] = d[x] + 1
    else:
        d[x] = 1

def g(k, d):
    return d[k]

ks = d.keys()
```

- [ ] A. sorted(ks, key=g)
- [x] B. sorted(ks, key=lambda x: g(x, d))
- [x] C. sorted(ks, key=lambda x: d[x])

Check me | Compare me

✔ Correct.
B. The lambda function takes just one parameter, and calls g with two parameters.
C. The lambda function looks up the value of x in d.

Activity: 6 -- Multiple Choice (question18_4_1)

2. Sort the following dictionary based on the keys so that they are sorted a to z. Assign the resulting value to the variable `sorted_keys`.

Save & Run | 5/14/2021, 6:19:45 PM - 4 of 4 | Show in CodeLens

```
1
2 dictionary = {"Flowers": 10, 'Trees': 20, 'Chairs': 6, "Firepit": 1, 'Grill': 2, 'I
3 sorted_keys=sorted(dictionary)
4
```

Activity: 7 -- ActiveCode (ac18_4_8)

Result	Actual Value	Expected Value	Notes
Pass	['Cha...ees']	['Cha...ees']	Testing that sorted_keys has the correct value.

Expand Differences

You passed: 100.0% of the tests

3. Below, we have provided the dictionary `groceries`, whose keys are grocery items, and values are the number of each item that you need to buy at the store. Sort the dictionary's keys into alphabetical order, and save them as a list called `grocery_keys_sorted`.

Save & Run | 5/14/2021, 6:21:37 PM - 4 of 4 | Show in CodeLens

```
1
2 groceries = {'apples': 5, 'pasta': 3, 'carrots': 12, 'orange juice': 2, 'bananas':
3 grocery_keys_sorted=sorted(groceries)
4
```

Activity: 8 -- ActiveCode (ac18_4_9)

Result	Actual Value	Expected Value	Notes
Pass	['app...ach']	['app...ach']	Testing that grocery_keys_sorted was created correctly.

Expand Differences

You passed: 100.0% of the tests

4. Sort the following dictionary's keys based on the value from highest to lowest. Assign the resulting value to the variable `sorted_values`.

Save & Run 5/14/2021, 6:22:16 PM - 2 of 2 Show in CodeLens

```
1
2 dictionary = {"Flowers": 10, 'Trees': 20, 'Chairs': 6, "Firepit": 1, 'Grill': 2, 'I
3 sorted_values=sorted(dictionary,key=lambda k:dictionary[k],reverse=True)
4
5 for k in sorted_values:
6     print ("{} appears {} times".format(k,dictionary[k]))
7
8 print (sorted_values)
9
```

```
Trees appears 20 times
Lights appears 14 times
Flowers appears 10 times
Chairs appears 6 times
Grill appears 2 times
Firepit appears 1 times
['Trees', 'Lights', 'Flowers', 'Chairs', 'Grill', 'Firepit']
```

Activity: 9 -- ActiveCode (ac18_4_10)

Result	Actual Value	Expected Value	Notes
Pass	['Tre...pit']	['Tre...pit']	Testing that sorted_values has the correct value.

Expand Differences

You passed: 100.0% of the tests

You have attempted 10 of 9 activities on this page

16.3. Optional key parameter">
16.5. Breaking Ties: Second Sorting">
Completed. Well Done!
16.5. Breaking Ties: Second Sorting">Next Section - 16.5. Breaking Ties: Second Sorting

| Back to top