11.8. Accumulating the Best Key

Now what if we want to find the *key* associated with the maximum value? It would be nice to just find the maximum value as above, and then look up the key associated with it, but dictionaries don't work that way. You can look up the value associated with a key, but not the key associated with a value. (The reason for that is there may be more than one key that has the same value).

The trick is to have the accumulator keep track of the best key so far instead of the best value so far. For simplicity, let's assume that there are at least two keys in the dictionary. Then, similar to our first version of computing the max of a list, we can initialize the best-key-so-far to be the first key, and loop through the keys, replacing the best-so-far whenever we find a better one.

In the exercise below, we have provided skeleton code. See if you can fill it in. An answer is provided, but you'll learn more if you try to write it yourself first.

Question Answer

Save & Run Original - 1 of 1 Show in CodeLens

```
d = {'a': 194, 'b': 54, 'c':34, 'd': 44, 'e': 312, 'full':31}

ks = d.keys()
best_key_so_far = list(ks)[0]  # Have to turn ks into a real
for k in ks:
    if d[k] > d[best_key_so_far]:
        best_key_so_far = k

print("key " + best_key_so_far + " has the highest value, " +
```

```
key e has the highest value, 312
```

Activity: 1 -- ActiveCode (answer10_7_1)

Check your Understanding

1. Create a dictionary called `d` that keeps track of all the characters in the string `placement` and notes how many times each character was seen. Then, find the key with the lowest value in this dictionary and assign that key to `min_value`.

Save & Run 5/14/2021, 9:19:10 AM - 2 of 2 Show in CodeLens

```
placement = "Beaches are cool places to visit in spring however the Mackinaw Bridge
d = {}

for c in placement:
    if c not in d:
        d[c] = 0
    d[c] = d[c] + 1

keys = list(d.keys())
min_value = keys[0]

for key in keys:
    if d[key] < d[min_value]:
        min_value = key
```

Activity: 2 -- ActiveCode (ac10_7_2)

Result	Actual Value	Expected Value	Notes	
Pass	[' ',... 'x']	[' ',... 'x']	Testing the keys were created correctly	Expand Differences
Pass	[1, 2..., 27]	[1, 2..., 27]	Testing the values were created correctly	Expand Differences
Pass	'x'	'x'	Testing that min_value has been correctly assigned	

You passed: 100.0% of the tests

5. Create a dictionary called `lett_d` that keeps track of all of the characters in the string `product` and notes how many times each character was seen. Then, find the key with the highest value in this dictionary and assign that key to `max_value`.

Save & Run 5/14/2021, 9:21:18 AM - 2 of 2 Show in CodeLens

```
product = "iphone and android phones"

lett_d = {}

for c in product:
    if c not in lett_d:
        lett_d[c] = 0
    lett_d[c] = lett_d[c] + 1
keys = list(lett_d.keys())

max_value = keys[0]
for key in keys:
    if lett_d[key] > lett_d[max_value]:
        max_value = key
```

Activity: 3 -- ActiveCode (ac10_7_3)

Result	Actual Value	Expected Value	Notes
Pass	[(' '..., 1)]	[(' '..., 1)]	Testing that lett_d has been created correctly.
Pass	'n'	'n'	Testing that max_value has been correctly assigned

Expand Differences

You passed: 100.0% of the tests

You have attempted 4 of 4 activities on this page

11.7. Accumulating Results From a Dictionary">
11.9. When to use a dictionary">
ccumulating Results From a Dictionary">
Completed. Well Done!
11.9. When to use a dictionary">Next Section - 11.9. When to use a dictionary

| Back to top