

10.5: MULTIPLE ASSIGNMENT WITH DICTIONARIES



Contributed by [Chuck Severance](#)
Clinical Associate Professor (School of Information) at [University of Michigan](#)

Combining `items`, tuple assignment, and `for`, you can see a nice code pattern for traversing the keys and values of a dictionary in a single loop:

CODE 10.5.1 (PYTHON):

```
%%python3
d = {'a':10, 'b':1, 'c':22}

for key, val in list(d.items()):
    print(val, key)
```

run

restart

This loop has two *iteration variables* because `items` returns a list of tuples and `key, val` is a tuple assignment that successively iterates through each of the key-value pairs in the dictionary.

For each iteration through the loop, both `key` and `value` are advanced to the next key-value pair in the dictionary (still in hash order).

The output of this loop is:

```
10 a
22 c
1 b
```

Again, it is in hash key order (i.e., no particular order).

If we combine these two techniques, we can print out the contents of a dictionary sorted by the *value* stored in each key-value pair.

To do this, we first make a list of tuples where each tuple is `(value, key)`. The `items` method would give us a list of `(key, value)` tuples, but this time we want to sort by value, not key. Once we have constructed the list with the value-key tuples, it is a simple matter to sort the list in reverse order and print out the new, sorted list.

```
>>> d = {'a':10, 'b':1, 'c':22}
>>> l = list()
>>> for key, val in d.items() :
...     l.append( (val, key) )
...
>>> l
[(10, 'a'), (22, 'c'), (1, 'b')]
>>> l.sort(reverse=True)
>>> l
[(22, 'c'), (10, 'a'), (1, 'b')]
>>>
```

By carefully constructing the list of tuples to have the value as the first element of each tuple, we can sort the list of tuples and get our dictionary contents sorted by value.