Chapter 23: groupby()

Parameter	Details
iterable	Any python iterable
key	Function(criteria) on which to group the iterable

In Python, the `itertools.groupby()` method allows developers to group values of an iterable class based on a specified property into another iterable set of values.

Section 23.1: Example 4

In this example we see what happens when we use different types of iterable.

```
things = [("animal", "bear"), ("animal", "duck"), ("plant", "cactus"), ("vehicle", "harley"), \
          ("vehicle", "speed boat"), ("vehicle", "school bus")]
dic = {}
f = lambda x: x[0]
for key, group in groupby(sorted(things, key=f), f):
    dic[key] = list(group)
dic
```

Results in

```
{'animal': [('animal', 'bear'), ('animal', 'duck')],
 'plant': [('plant', 'cactus')],
 'vehicle': [('vehicle', 'harley'),
  ('vehicle', 'speed boat'),
  ('vehicle', 'school bus')]}
```

This example below is essentially the same as the one above it. The only difference is that I have changed all the tuples to lists.

```
things = [["animal", "bear"], ["animal", "duck"], ["vehicle", "harley"], ["plant", "cactus"], \
          ["vehicle", "speed boat"], ["vehicle", "school bus"]]
dic = {}
f = lambda x: x[0]
for key, group in groupby(sorted(things, key=f), f):
    dic[key] = list(group)
dic
```

Results

```
{'animal': [['animal', 'bear'], ['animal', 'duck']],
 'plant': [['plant', 'cactus']],
 'vehicle': [['vehicle', 'harley'],
  ['vehicle', 'speed boat'],
  ['vehicle', 'school bus']]}
```

Section 23.2: Example 2

This example illustrates how the default key is chosen if we do not specify any

```
c = groupby(['goat', 'dog', 'cow', 1, 1, 2, 3, 11, 10, ('persons', 'man', 'woman')])
dic = {}
for k, v in c:
```

```
    dic[k] = list(v)
dic
```

Results in

```
{1: [1, 1],
 2: [2],
 3: [3],
 ('persons', 'man', 'woman'): [('persons', 'man', 'woman')],
 'cow': ['cow'],
 'dog': ['dog'],
 10: [10],
 11: [11],
 'goat': ['goat']}
```

Notice here that the tuple as a whole counts as one key in this list

Section 23.3: Example 3

Notice in this example that mulato and camel don't show up in our result. Only the last element with the specified key shows up. The last result for c actually wipes out two previous results. But watch the new version where I have the data sorted first on same key.

```
list_things = ['goat', 'dog', 'donkey', 'mulato', 'cow', 'cat', ('persons', 'man', 'woman'), \
               'wombat', 'mongoose', 'malloo', 'camel']
c = groupby(list_things, key=lambda x: x[0])
dic = {}
for k, v in c:
    dic[k] = list(v)
dic
```

Results in

```
{'c': ['camel'],
 'd': ['dog', 'donkey'],
 'g': ['goat'],
 'm': ['mongoose', 'malloo'],
 'persons': [('persons', 'man', 'woman')],
 'w': ['wombat']}
```

Sorted Version

```
list_things = ['goat', 'dog', 'donkey', 'mulato', 'cow', 'cat', ('persons', 'man', 'woman'), \
               'wombat', 'mongoose', 'malloo', 'camel']
sorted_list = sorted(list_things, key = lambda x: x[0])
print(sorted_list)
print()
c = groupby(sorted_list, key=lambda x: x[0])
dic = {}
for k, v in c:
    dic[k] = list(v)
dic
```

Results in

```
['cow', 'cat', 'camel', 'dog', 'donkey', 'goat', 'mulato', 'mongoose', 'malloo', ('persons', 'man',
'woman'), 'wombat']
```

```
{'c': ['cow', 'cat', 'camel'],
 'd': ['dog', 'donkey'],
 'g': ['goat'],
 'm': ['mulato', 'mongoose', 'malloo'],
 'persons': [('persons', 'man', 'woman')],
 'w': ['wombat']}
```