Lab 4 – Mapping using inheritance

Instructions:

Provide your solutions in a file named **lab4.py**. Make sure you run the doctest on your module

Problems:

Write a `Mapping` class that represents a 1-1 mapping between items. A 1-1 mapping is like a dictionary, but it has key1:key2 pairs (instead of key:value pairs). You can look up either key. For example, if m[1] is equal to 2, then m[2] is equal to 1. Your implementation MUST inherit from (subclass) dict. Basically, whenever you create key11:key2 pair, you also make the corresponding key2:key1 pair.

```
>>> m = Mapping()
>>> m[1]=2
>>> m
Mapping({1: 2, 2: 1})
>>> m[1]
2
>>> m[2]
1
>>> m[3]=4
>>> m
Mapping({1: 2, 2: 1, 3: 4, 4: 3})
>>> m[5]=5
>>> m
Mapping({1: 2, 2: 1, 3: 4, 4: 3, 5: 5})
>>> m[5]
5
>>> m.pop(1)
>>> m
Mapping({3: 4, 4: 3, 5: 5})
>>> m.pop(5)
>>> m
Mapping({3: 4, 4: 3})
>>> m = Mapping( {1:2,2:3,4:4})
>>> m
Mapping({2: 3, 3: 2, 4: 4})
>>>
```

Implementation notes: You MUST inherit from (subclass dict). You will get some functionality for free.

- Write the class first without worrying about self mapping keys (m[5]=5). Once you get that to work, worry about self mapping keys (notes below).
- `__repr__` - see test code for output
- `__getitem__` - get this for free! No need to implement.
- `pop()`
    - when a key is popped you, must also pop the key it maps to
    - For self mapping keys, you need to make sure that this does not create an error!!!!! Don't worry about this until you get everything else working.
- `__setitem__`
    - given key1:key2 make sure you also add key2:key1
    - before adding the key1:key2 and key2:key1 pairs, you should check whether either key1 or key2 exists already. If so, they should be popped before adding.
- `__init__` you should be able to construct
    - An empty Mapping
    - A Mapping constructed from a given dictionary, see test runs towards bottom.