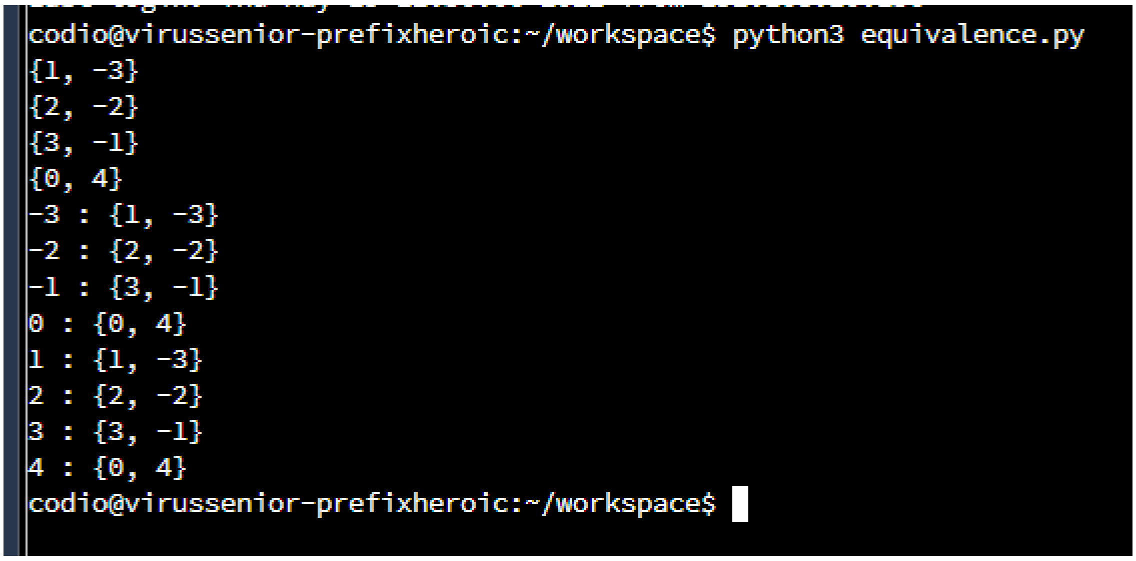
Equivalence Testing in Python

```
# CODE SOURCE: https://stackoverflow.com/questions/38924421/is-there-a-
standard-way-to-partition-an-interable-into-equivalence-classes-
given/38924631#38924631

def equivalence_partition(iterable, relation):
    """Partitions a set of objects into equivalence classes

    Args:
        iterable: collection of objects to be partitioned
        relation: equivalence relation. I.e. relation(o1,o2) evaluates
to True
            if and only if o1 and o2 are equivalent

    Returns: classes, partitions
        classes: A sequence of sets. Each one is an equivalence class
        partitions: A dictionary mapping objects to equivalence classes
    """
    classes = []
    partitions = {}
    for o in iterable:  # for each object
        # find the class it is in
        found = False
        for c in classes:
            if relation(next(iter(c)), o):  # is it equivalent to this
class?
                c.add(o)
                partitions[o] = c
                found = True
                break
        if not found:  # it is in a new class
            classes.append(set([o]))
            partitions[o] = classes[-1]
    return classes, partitions


def equivalence_enumeration(iterable, relation):
    """Partitions a set of objects into equivalence classes

    Same as equivalence_partition() but also numbers the classes.

    Args:
        iterable: collection of objects to be partitioned
        relation: equivalence relation. I.e. relation(o1,o2) evaluates
to True
            if and only if o1 and o2 are equivalent

    Returns: classes, partitions, ids
        classes: A sequence of sets. Each one is an equivalence class
        partitions: A dictionary mapping objects to equivalence classes
        ids: A dictionary mapping objects to the indices of their
equivalence classes
    """
    classes, partitions = equivalence_partition(iterable, relation)
    ids = {}
    for i, c in enumerate(classes):
        for o in c:
            ids[o] = i
    return classes, partitions, ids


def check_equivalence_partition(classes, partitions, relation):
    """Checks that a partition is consistent under the relationship"""
    for o, c in partitions.items():
        for _c in classes:
            assert (o in _c) ^ (not _c is c)
    for c1 in classes:
        for o1 in c1:
            for c2 in classes:
                for o2 in c2:
                    assert (c1 is c2) ^ (not relation(o1, o2))


def test_equivalence_partition():
    relation = lambda x, y: (x - y) % 4 == 0
    classes, partitions = equivalence_partition(
        range(-3, 5),
        relation
    )
    check_equivalence_partition(classes, partitions, relation)
    for c in classes: print(c)
    for o, c in partitions.items(): print(o, ':', c)


if __name__ == '__main__':
    test_equivalence_partition()
```

Output



```
codio@virussenior-prefixheroic:~/workspace$ python3 equivalence.py
{1, -3}
{2, -2}
{3, -1}
{0, 4}
-3 : {1, -3}
-2 : {2, -2}
-1 : {3, -1}
0 : {0, 4}
1 : {1, -3}
2 : {2, -2}
3 : {3, -1}
4 : {0, 4}
codio@virussenior-prefixheroic:~/workspace$
```