Seminar 4: Activity

Question:

Run equivalence.py in your chosen Jupyter Notebook workspace - Testing with Python - which is an implementation of equivalence partitioning. This test partitions integers [-3,5] into equivalence classes based on *lambda x, y: (x-y)%4 == 0*.

In the output, you should be able to see how a set of objects to be partitioned are considered, and a function evaluates if the two objects are equivalent before printing the result.

test_equivalence_partition() produces the following output:

set([1, -3]) set([2, -2]) set([3, -1]) set([0, 4]) 0 : set([0, 4]) 1 : set([1, -3]) 2 : set([2, -2]) 3 : set([3, -1]) 4 : set([0, 4]) -2 : set([2, -2]) -3 : set([1, -3]) -1 : set([3, -1])

You should carry out further investigations on the code and experiment with it.

Remember to record your results, ideas and team discussions in your e-portfolio.

equivalence.py code:

```
# 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()
```

Answer Response:

The equivalence partitioning test ran successfully and produced the following results:

1. Equivalence Classes:
   - Set 1: {-3, 1}
   - Set 2: {-2, 2}
   - Set 3: {-1, 3}
   - Set 4: {0, 4}
2. Partitions:
   - -3 belongs to the set {-3, 1}
   - -2 belongs to the set {-2, 2}
   - -1 belongs to the set {-1, 3}
   - 0 belongs to the set {0, 4}
   - 1 belongs to the set {-3, 1}
   - 2 belongs to the set {-2, 2}
   - 3 belongs to the set {-1, 3}
   - 4 belongs to the set {0, 4}

The partitioning is done based on the equivalence relation (x - y) % 4 == 0, which groups numbers with the same remainder when divided by 4.