```
#!/usr/bin/env python
"""
coverage -- explore the use of the 'coverage' measure for predicting which
            combination of interst operators should work best
"""
import os, sys, math, numpy

def load_locations (fn):
    fd = open (fn)
    lines = fd.readlines()
    fd.close()
    nf = int (lines[1])
    locs = numpy.zeros ((nf, 2))
    for f in range (0, nf):
        v = lines[f+2].split()
        # Each line contains x, y, and stuff we're not interested in.
        locs[f,1] = float (v[0])
        locs[f,0] = float (v[1])
    return locs

def coverage (locs):
    nlocs, loclen = locs.shape
    if loclen != 2:
        raise ValueError, "I don't have two values for the location!"
    dsum = 0.0
    npaths = 0
    ncoin = 0
    for i in range (0, nlocs):
        y1 = locs[i,0]
        x1 = locs[i,1]
        for j in range (i+1, nlocs):
            y2 = locs[j,0]
            x2 = locs[j,1]
            d = math.sqrt ((y2 - y1)**2 + (x2 - x1)**2)
            if d == 0.0:
                ncoin += 1
            else:
                dsum += 1.0 / d
                npaths += 1
    return npaths / dsum, npaths, ncoin

def mean_coverage1 (opname, imsets):
    csum = 0.0
    nfiles = 0
    for imset in imsets:
        for fno in range (1,7):
            fn = 'locations/%s_%s%d.txt' % (opname, imset, fno)
            locs = load_locations (fn)
            c, np, nc = coverage (locs)
            # print '  ', fn, 'gave', np, 'paths and', nc, 'coincident points.'
            csum += c
            nfiles += 1
    return csum / nfiles

def sort_by_value (d):
    '''Return the keys of dictionary d sorted by their values'''
    items = d.items()
    backitems = [[v[1],v[0]] for v in items]
    backitems.sort()
    return [backitems[i][1] for i in range(0,len(backitems))]

#-----------------------------------------------------------------------------
# Main program
#-----------------------------------------------------------------------------
opnames = ['ebr', 'haraff', 'harlap', 'hesaff', 'heslap', 'ibr', 'mser',
           'salient', 'sfop', 'sift', 'surf']
imsets = ['bark', 'bikes', 'boat', 'graf', 'leuv', 'trees', 'ubc', 'wall']
results = {}
```



```


# Calculate the coverage for each individual operator.
for op in opnames:
    mc = mean_coverage1 (op, imsets)
    print op, mc
    results[op] = mc

# Output what we've calculated in descending order of coverage.
ds = sort_by_value (results)
for k in reversed (ds):
    print k, results[k]

#-----------------------------------------------------------------------------
# End of coverage
#-----------------------------------------------------------------------------
```