```
#!/usr/bin/env python
# coding: utf-8

# In[1]:


import numpy as np
import matplotlib.pyplot as plt
from itertools import product
get_ipython().run_line_magic('matplotlib', 'inline')


# In[2]:


np.random.seed(1)
X = np.random.rand(8,2)
print(X)


# In[3]:


class Hierarchical():
    def distp(self,sample1,sample2):
        distance = []
        temp3 = list(product(range(len(sample1)), range(len(sample2))))
        for h in range(len(temp3)):
          i, j = temp3[h]
          distance.append(np.linalg.norm(np.array(sample1[i]) -
np.array(sample2[j])))
        return min(distance)

    def distc(self,samples):
        dist = np.zeros((len(samples),len(samples)))
        temp = list(product(range(dist.shape[0]), range(dist.shape[0])))
        for k in range(len(temp)):
          i, j = temp[k]
          if(i == j):
            dist[i,j] = 10**4
          else:
            dist[i,j] = float(self.distp(samples[i],samples[j]))
        return dist

    def distic(self,cl,sample):
        if sample[0]!='<class \'list\'>':
            sample = [sample]
        distance = []
        temp5 = list(product(range(len(cl)), range(len(sample))))

        for q in range(len(temp5)):
          i, j = temp5[q]
          distance.append(np.linalg.norm(np.array(cl[i])-np.array(sample[j])))
        return min(distance)

ci = [[i] for i in range(X.shape[0])]
cls     = [[list(X[i])] for i in range(X.shape[0])]

clslen = len(cls)
hclus  = Hierarchical()

while clslen > 1:
    distmat      = hclus.distc(cls)
    minIdex = np.where(distmat==distmat.min())[0]
```

```
    valueToAdd      = cls.pop(minIdex[1])
    cls[minIdex[0]].append(valueToAdd)
    print('Cluster Node 1 :' , ci[minIdex[0]])
    print('Cluster Node 2 :' , ci[minIdex[1]])
    ci[minIdex[0]].append(ci[minIdex[1]])
    ci[minIdex[0]] = [ci[minIdex[0]]]
    v = ci.pop(minIdex[1])
    clslen = len(cls)
    print('Current Clusters:',ci)
    print('\n')


# In[4]:


from scipy.cluster.hierarchy import dendrogram, linkage
from matplotlib import pyplot as plt
l = linkage(X, 'single')
fig = plt.figure()
dn = dendrogram(l)


# In[ ]:




# In[ ]:
```