```
#Program for composition on Fuzzy and Crips relations

import numpy as np

def maxMin(x, y):
        z = []
        for x1 in x:
            for y1 in y.T:
                z.append(max(np.minimum(x1, y1)))
        return np.array(z).reshape((x.shape[0], y.shape[1]))



# Max-Product Composition given by Rosenfeld

def maxProduct(x, y):
        z = []
        for x1 in x:
            for y1 in y.T:
                z.append(max(np.multiply(x1, y1)))
        return np.array(z).reshape((x.shape[0], y.shape[1]))


# 3 arrays for the example

r1 = np.array([[1, 0,  .7], [.3, .2, 0], [0,  .5,1]])
r2 = np.array([[.6,  .6, 0], [0, .6, .1], [0,  .1,0]])
r3 = np.array([[1,  0,  .7], [0, 1, 0], [.7, 0, 1]])

print ("R1oR2 => Max-Min :\n" + str(maxMin(r1, r2)) + "\n")
print ("R1oR2 => Max-Product :\n" + str(maxProduct(r1, r2)) + "\n\n")

print ("R1oR3 => Max-Min :\n" + str(maxMin(r1, r3)) + "\n")
print ("R1oR3 => Max-Product :\n" + str(maxProduct(r1, r3)) + "\n\n")

print ("R1oR2oR3 => Max-Min :\n" + str(maxMin(r1, maxMin(r2, r3))) + "\n")
print ("R1oR2oR3 => Max-Product :\n" + str(maxProduct(r1, maxProduct(r2, r3))) + "\n\n")
```

```
R1oR2 => Max-Min :
[[0.6 0.6 0. ]
 [0.3 0.3 0.1]
 [0.  0.5 0.1]]

R1oR2 => Max-Product :
[[0.6  0.6  0.  ]
 [0.18 0.18 0.02]
 [0.   0.3  0.05]]


R1oR3 => Max-Min :
[[1.  0.  0.7]
 [0.3 0.2 0.3]
 [0.7 0.5 1. ]]

R1oR3 => Max-Product :
[[1.   0.   0.7 ]
 [0.3  0.2  0.21]
 [0.7  0.5  1.  ]]


R1oR2oR3 => Max-Min :
[[0.6 0.6 0.6]
 [0.3 0.3 0.3]
 [0.1 0.5 0.1]]

R1oR2oR3 => Max-Product :
[[0.6   0.6   0.42 ]
 [0.18  0.18  0.126]
 [0.035 0.3   0.05 ]]
```