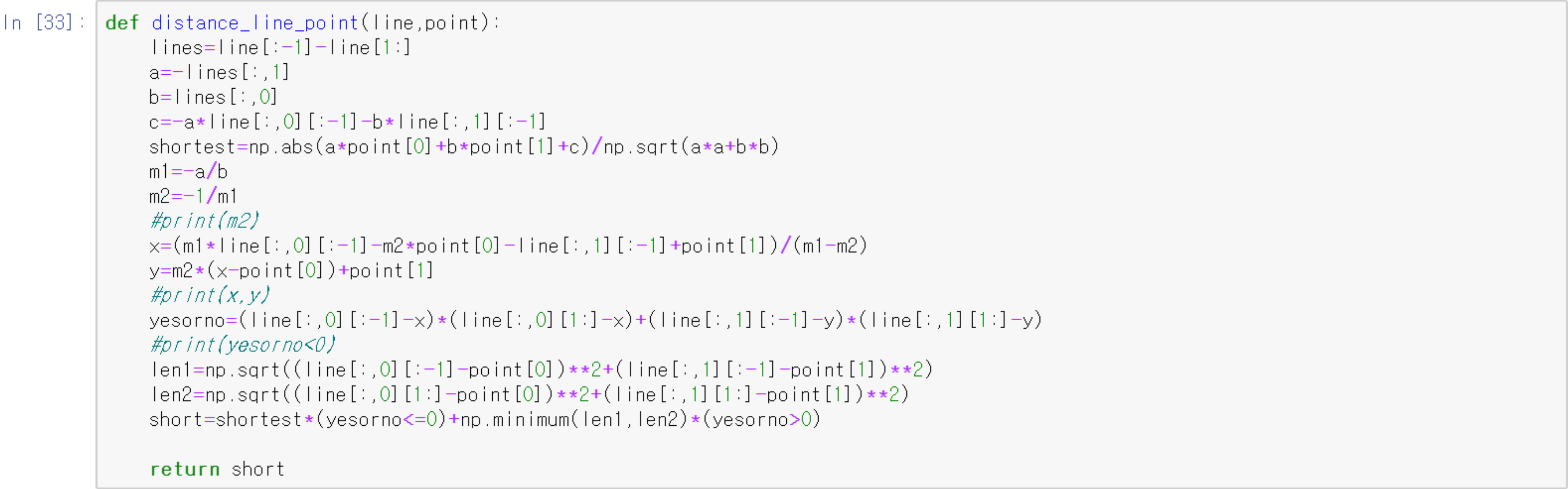
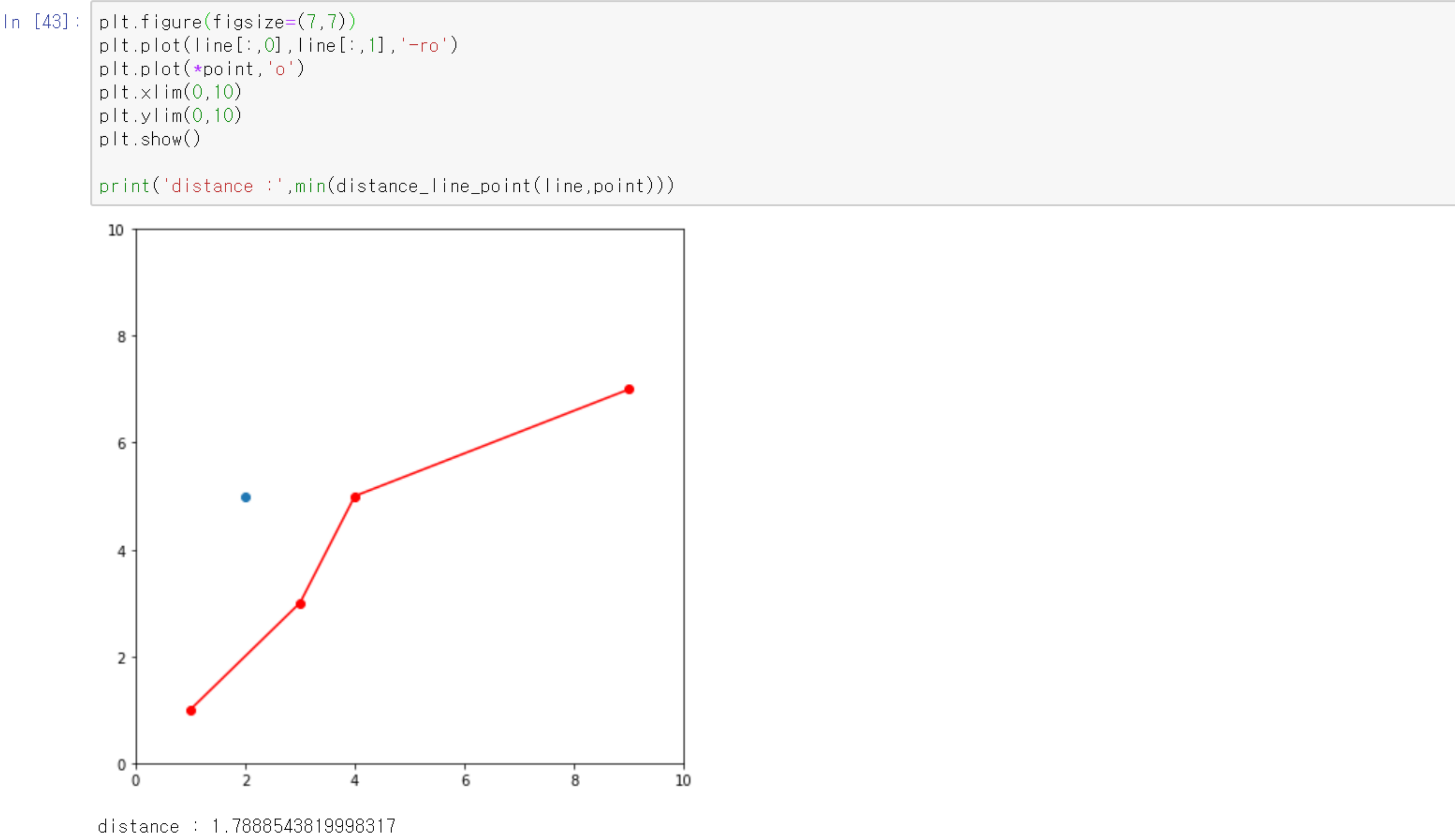


In [33]:

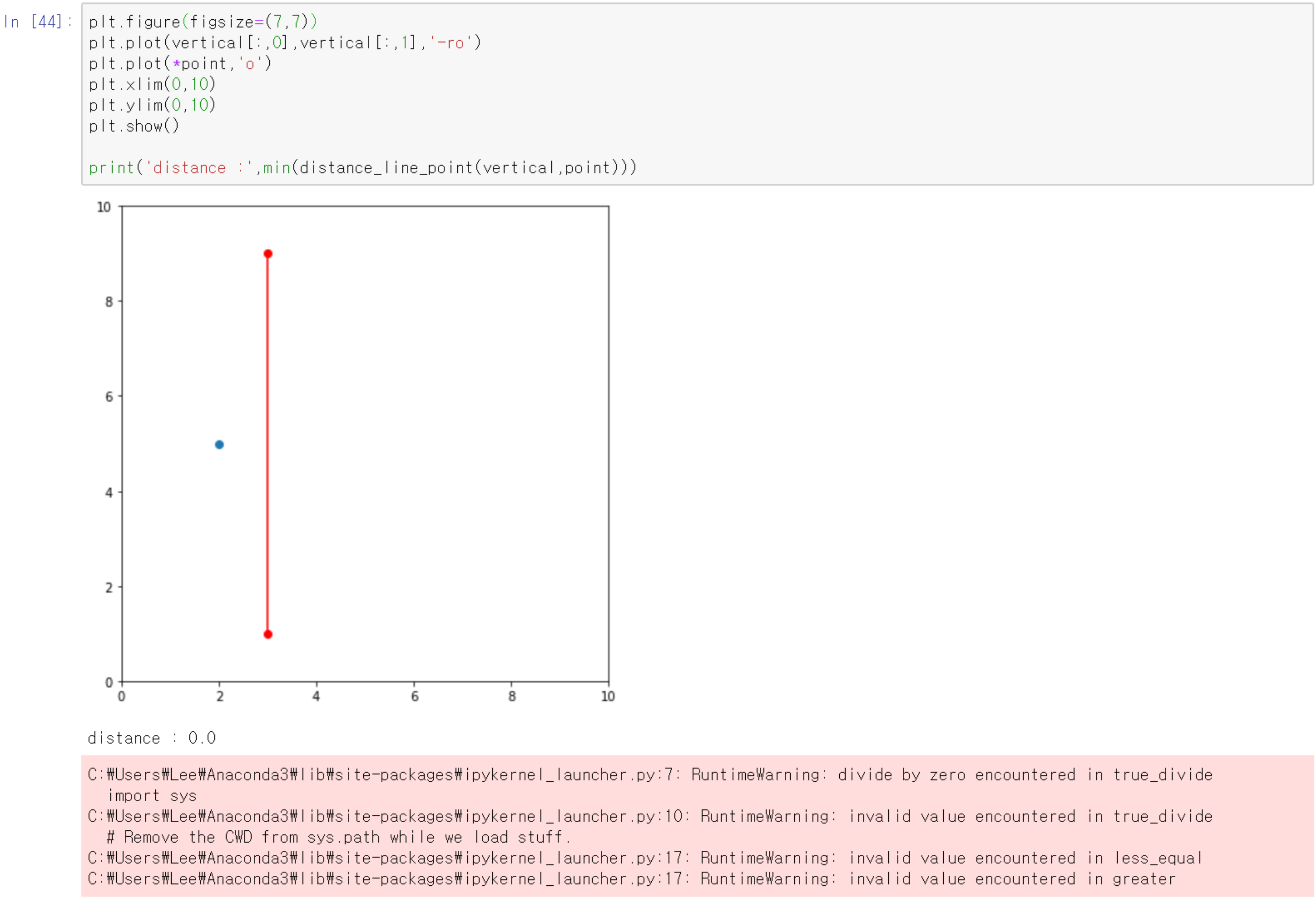
```
def distance_line_point(line,point):
    lines=line[:-1]-line[1:]
    a=-lines[:,1]
    b=lines[:,0]
    c=-a*line[:,0][:-1]-b*line[:,1][:-1]
    shortest=np.abs(a*point[0]+b*point[1]+c)/np.sqrt(a*a+b*b)
    m1=-a/b
    m2=-1/m1
    #print(m2)
    x=(m1*line[:,0][:-1]-m2*point[0]-line[:,1][:-1]+point[1])/(m1-m2)
    y=m2*(x-point[0])+point[1]
    #print(x,y)
    yesorno=(line[:,0][:-1]-x)*(line[:,0][1:]-x)+(line[:,1][:-1]-y)*(line[:,1][1:]-y)
    #print(yesorno<0)
    len1=np.sqrt((line[:,0][:-1]-point[0])**2+(line[:,1][:-1]-point[1])**2)
    len2=np.sqrt((line[:,0][1:]-point[0])**2+(line[:,1][1:]-point[1])**2)
    short=shortest*(yesorno<=0)+np.minimum(len1,len2)*(yesorno>0)

    return short
```



```
In [43]: plt.figure(figsize=(7,7))
         plt.plot(line[:,0],line[:,1],'-ro')
         plt.plot(*point,'o')
         plt.xlim(0,10)
         plt.ylim(0,10)
         plt.show()

         print('distance :',min(distance_line_point(line,point)))
```

```
distance : 1.7888543819998317
```

Error in vertical line



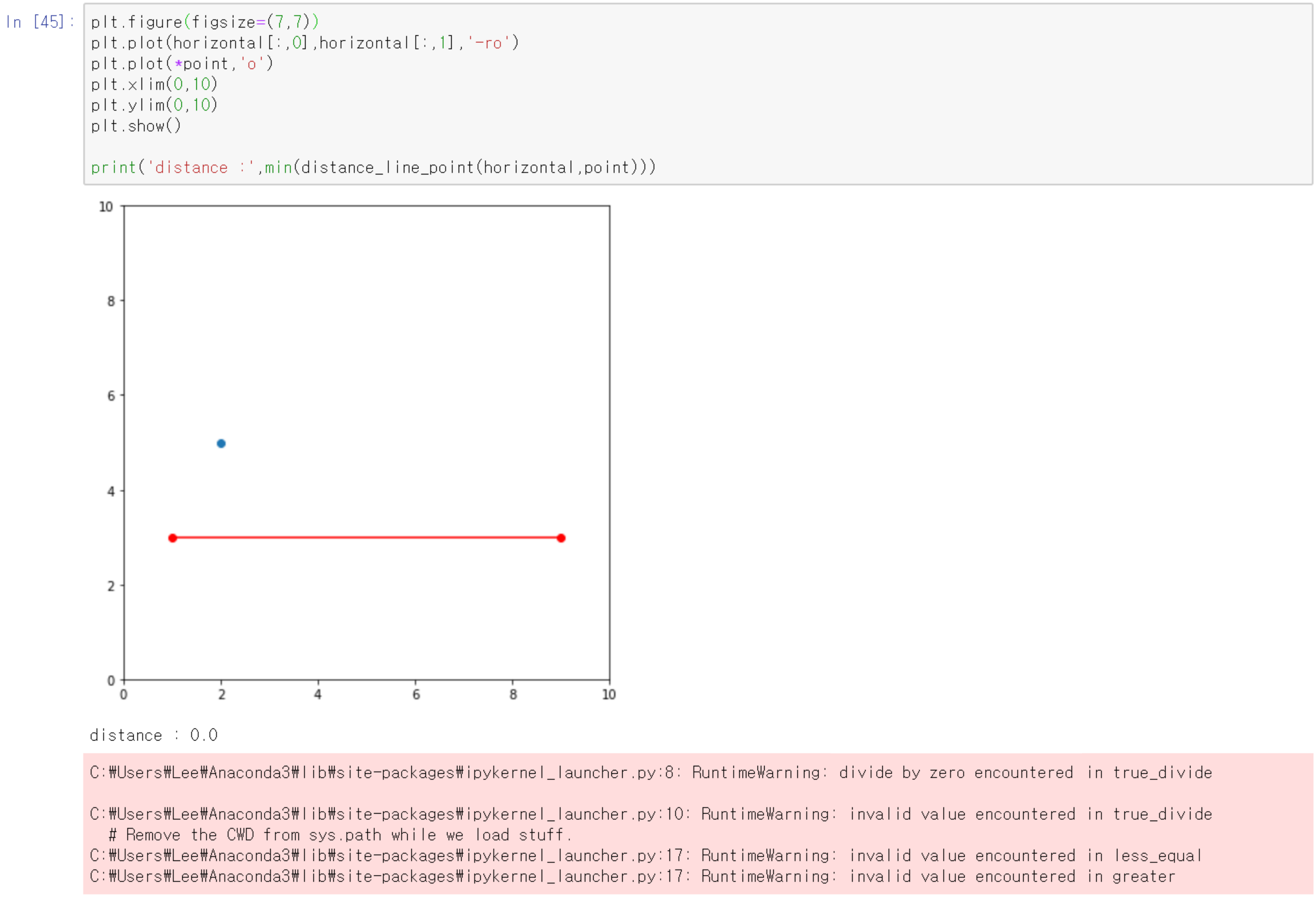
In [44]:

```
plt.figure(figsize=(7,7))
plt.plot(vertical[:,0],vertical[:,1],'-ro')
plt.plot(*point,'o')
plt.xlim(0,10)
plt.ylim(0,10)
plt.show()

print('distance :',min(distance_line_point(vertical,point)))
```

```
distance : 0.0
```

```
C:₩Users₩Lee₩Anaconda3₩lib₩site-packages₩ipykernel_launcher.py:7: RuntimeWarning: divide by zero encountered in true_divide
  import sys
C:₩Users₩Lee₩Anaconda3₩lib₩site-packages₩ipykernel_launcher.py:10: RuntimeWarning: invalid value encountered in true_divide
  # Remove the CWD from sys.path while we load stuff.
C:₩Users₩Lee₩Anaconda3₩lib₩site-packages₩ipykernel_launcher.py:17: RuntimeWarning: invalid value encountered in less_equal
C:₩Users₩Lee₩Anaconda3₩lib₩site-packages₩ipykernel_launcher.py:17: RuntimeWarning: invalid value encountered in greater
```

Error in horizontal line

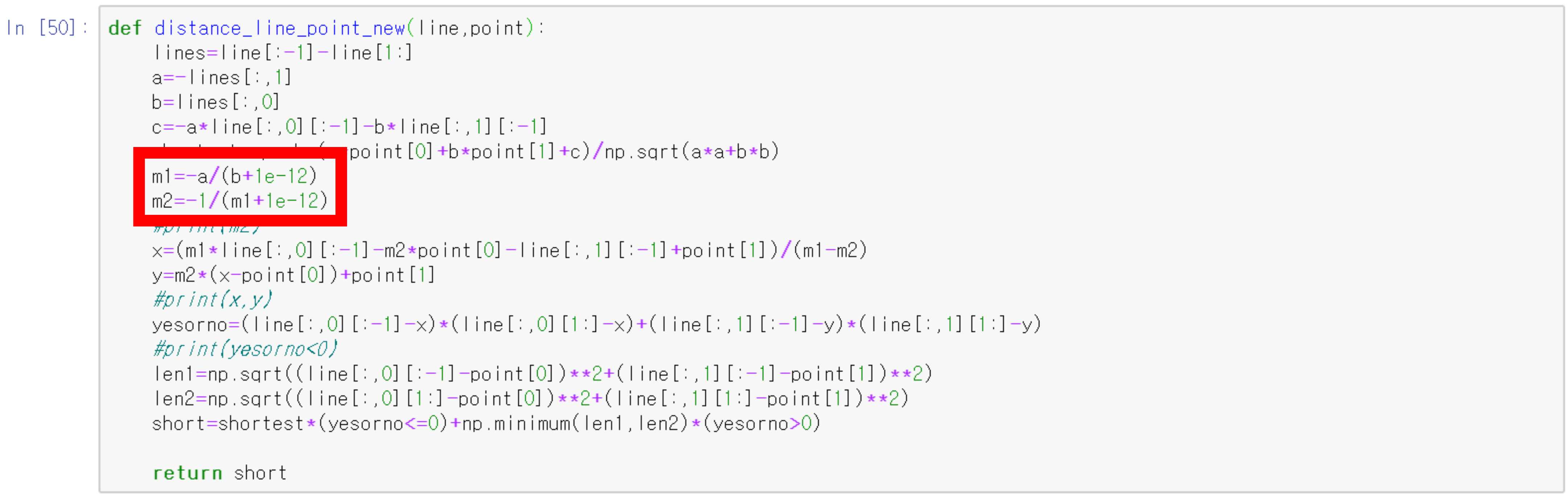


```
In [45]: plt.figure(figsize=(7,7))
         plt.plot(horizontal[:,0],horizontal[:,1],'-ro')
         plt.plot(*point,'o')
         plt.xlim(0,10)
         plt.ylim(0,10)
         plt.show()

         print('distance :',min(distance_line_point(horizontal,point)))
```

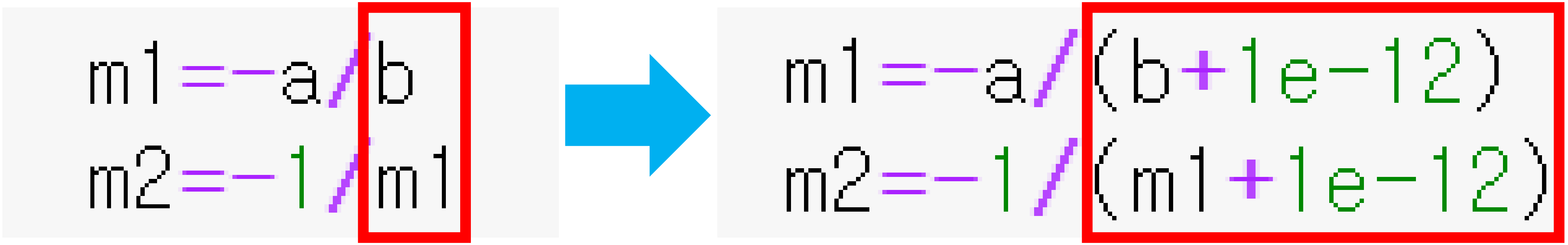
```
distance : 0.0
```

```
C:₩Users₩Lee₩Anaconda3₩lib₩site-packages₩ipykernel_launcher.py:8: RuntimeWarning: divide by zero encountered in true_divide

C:₩Users₩Lee₩Anaconda3₩lib₩site-packages₩ipykernel_launcher.py:10: RuntimeWarning: invalid value encountered in true_divide
  # Remove the CWD from sys.path while we load stuff.
C:₩Users₩Lee₩Anaconda3₩lib₩site-packages₩ipykernel_launcher.py:17: RuntimeWarning: invalid value encountered in less_equal
C:₩Users₩Lee₩Anaconda3₩lib₩site-packages₩ipykernel_launcher.py:17: RuntimeWarning: invalid value encountered in greater
```

Problem solve

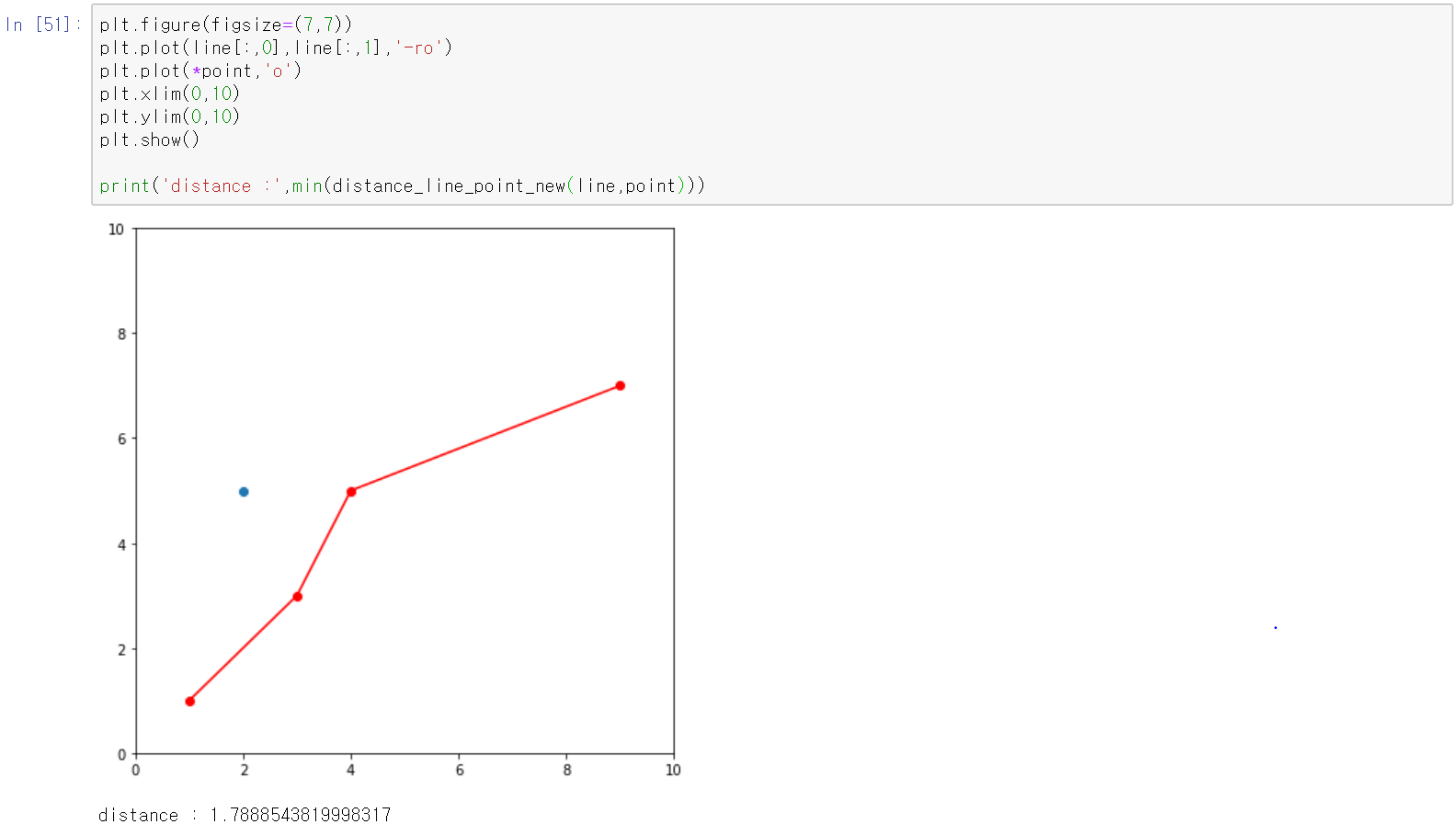


```
In [50]: def distance_line_point_new(line,point):
             lines=line[:-1]-line[1:]
             a=-lines[:,1]
             b=lines[:,0]
             c=-a*line[:,0][:-1]-b*line[:,1][:-1]
             shortest=np.abs(a*point[0]+b*point[1]+c)/np.sqrt(a*a+b*b)
             m1=-a/(b+1e-12)
             m2=-1/(m1+1e-12)
             #print(m2)
             x=(m1*line[:,0][:-1]-m2*point[0]-line[:,1][:-1]+point[1])/(m1-m2)
             y=m2*(x-point[0])+point[1]
             #print(x,y)
             yesorno=(line[:,0][:-1]-x)*(line[:,0][1:]-x)+(line[:,1][:-1]-y)*(line[:,1][1:]-y)
             #print(yesorno<0)
             len1=np.sqrt((line[:,0][:-1]-point[0])**2+(line[:,1][:-1]-point[1])**2)
             len2=np.sqrt((line[:,0][1:]-point[0])**2+(line[:,1][1:]-point[1])**2)
             short=shortest*(yesorno<=0)+np.minimum(len1,len2)*(yesorno>0)

             return short
```



```
m1=-a/b
m2=-1/m1
```

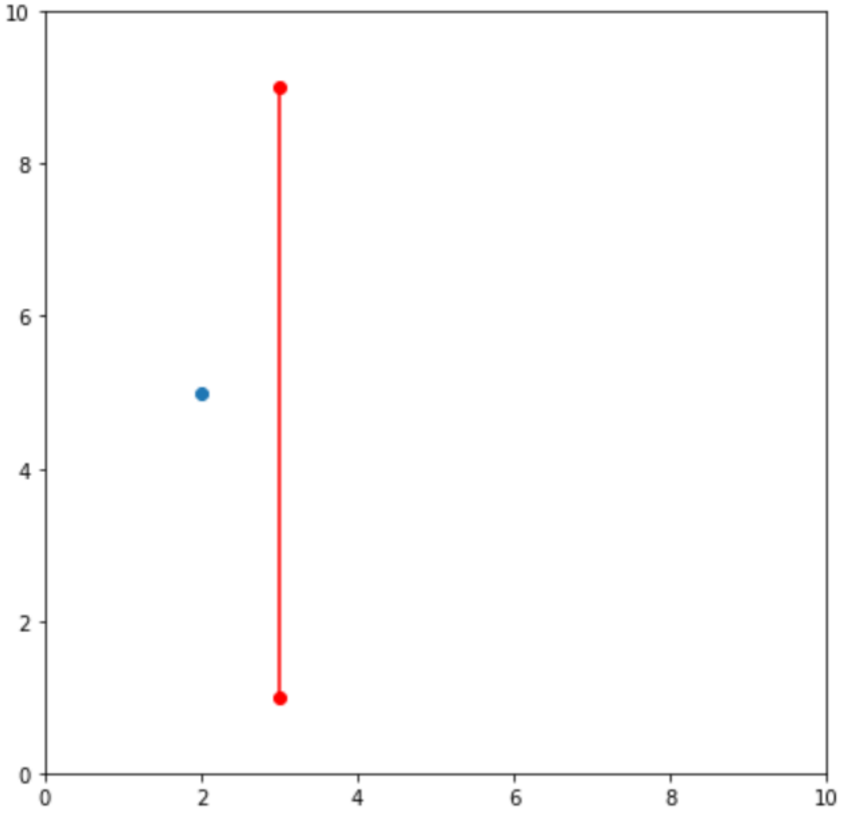
```
m1=-a/(b+1e-12)
m2=-1/(m1+1e-12)
```



```
In [51]: plt.figure(figsize=(7,7))
         plt.plot(line[:,0],line[:,1],'-ro')
         plt.plot(*point,'o')
         plt.xlim(0,10)
         plt.ylim(0,10)
         plt.show()

         print('distance :',min(distance_line_point_new(line,point)))
```

```
distance : 1.7888543819998317
```

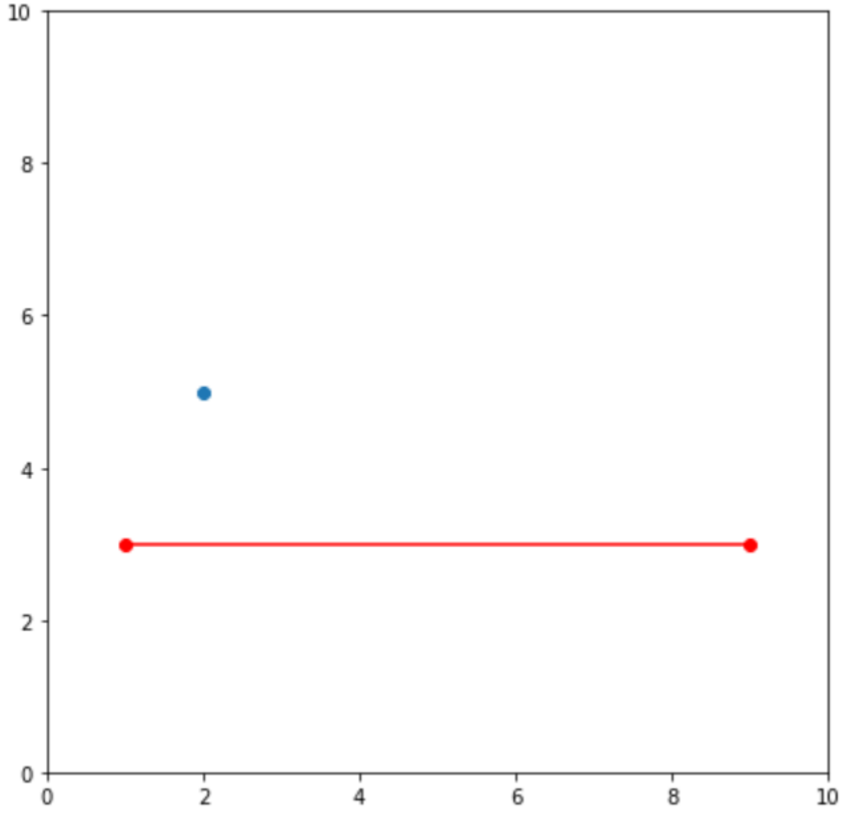
```
In [52]: plt.figure(figsize=(7,7))
         plt.plot(vertical[:,0],vertical[:,1],'-ro')
         plt.plot(*point,'o')
         plt.xlim(0,10)
         plt.ylim(0,10)
         plt.show()

         print('distance :',min(distance_line_point_new(vertical,point)))
```



```
distance : 1.0
```

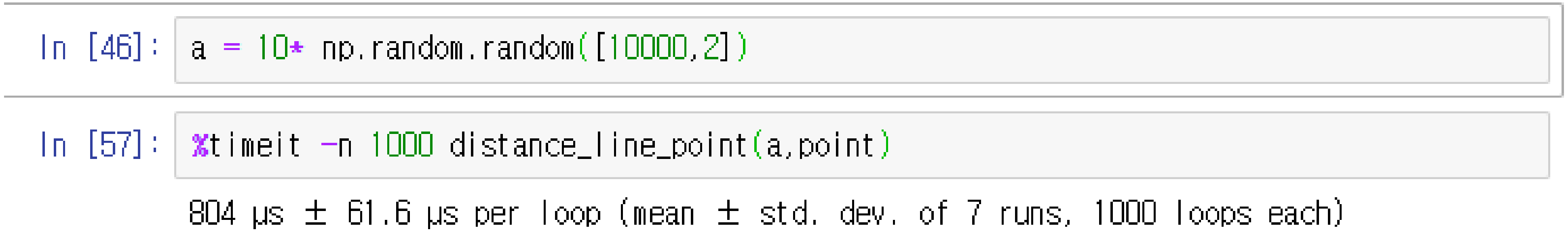
In [53]:

```
plt.figure(figsize=(7,7))
plt.plot(horizontal[:,0],horizontal[:,1],'-ro')
plt.plot(*point,'o')
plt.xlim(0,10)
plt.ylim(0,10)
plt.show()

print('distance :',min(distance_line_point_new(horizontal,point)))
```



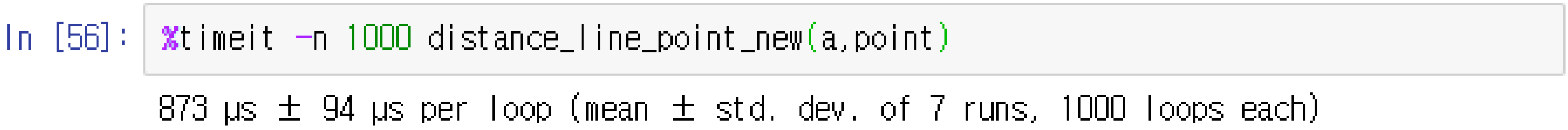
```
distance : 2.0
```

Time before



```
In [46]: a = 10* np.random.random([10000,2])

In [57]: %timeit -n 1000 distance_line_point(a,point)

         804 µs ± 61.6 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)
```

Time after

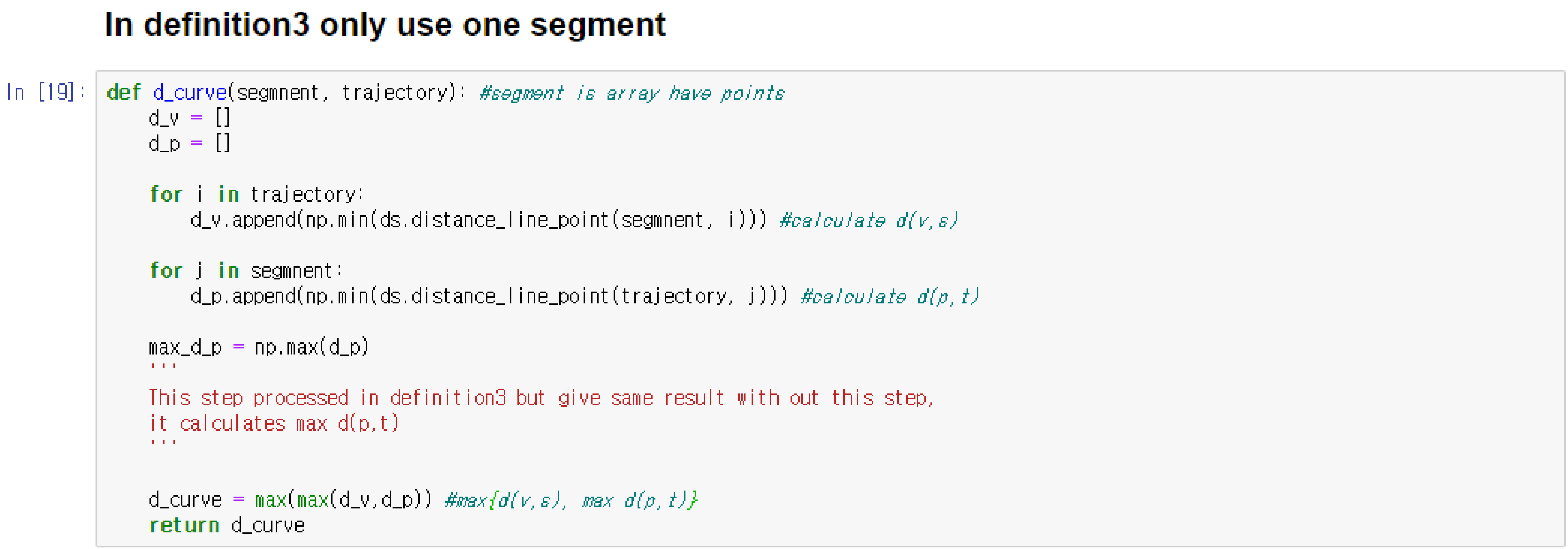


```
In [56]: %timeit -n 1000 distance_line_point_new(a,point)

         873 µs ± 94 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)
```

DEFINITION 3. *The distance between a segment s and a trajectory sample point $\mathbf{v}_{t,i}$ is*

$$d_{\text{curve}}(s, \mathbf{v}_{t,i}) = \max\left\{ d(\mathbf{v}_{t,i}, s), \max_{\mathbf{p} \in s} d(\mathbf{p}, t) \right\},$$

where $d(\mathbf{p}, c)$ denotes the Euclidean distance between point $\mathbf{p}$ and its closest point in piecewise linear curve c.

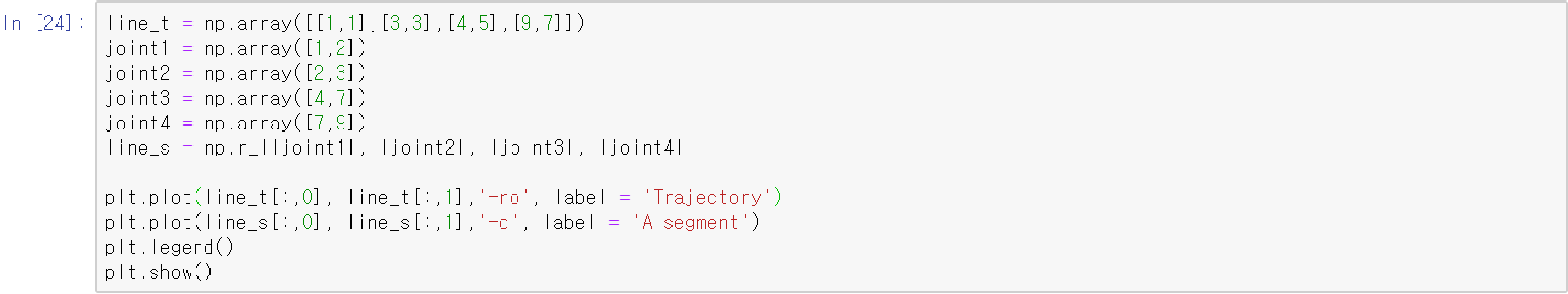


In definition3 only use one segment

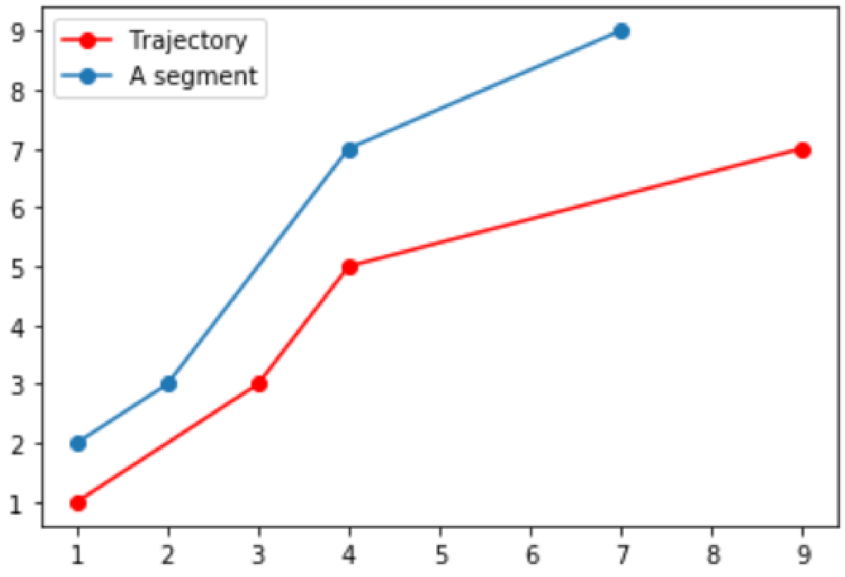
In [19]:

```
def d_curve(segmnent, trajectory): #segment is array have points
    d_v = []
    d_p = []

    for i in trajectory:
        d_v.append(np.min(ds.distance_line_point(segmnent, i))) #calculate d(v,s)

    for j in segmnent:
        d_p.append(np.min(ds.distance_line_point(trajectory, j))) #calculate d(p,t)

    max_d_p = np.max(d_p)
    '''
    This step processed in definition3 but give same result with out this step,
    it calculates max d(p,t)
    '''


    d_curve = max(max(d_v,d_p)) #max{d(v,s), max d(p,t)}
    return d_curve
```

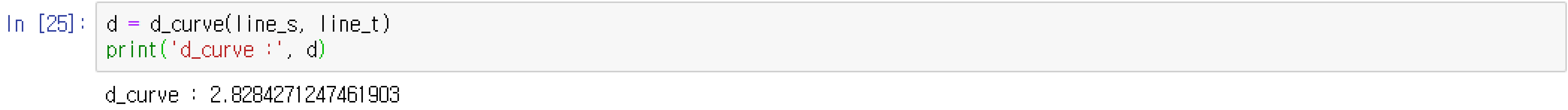
d_{curve}



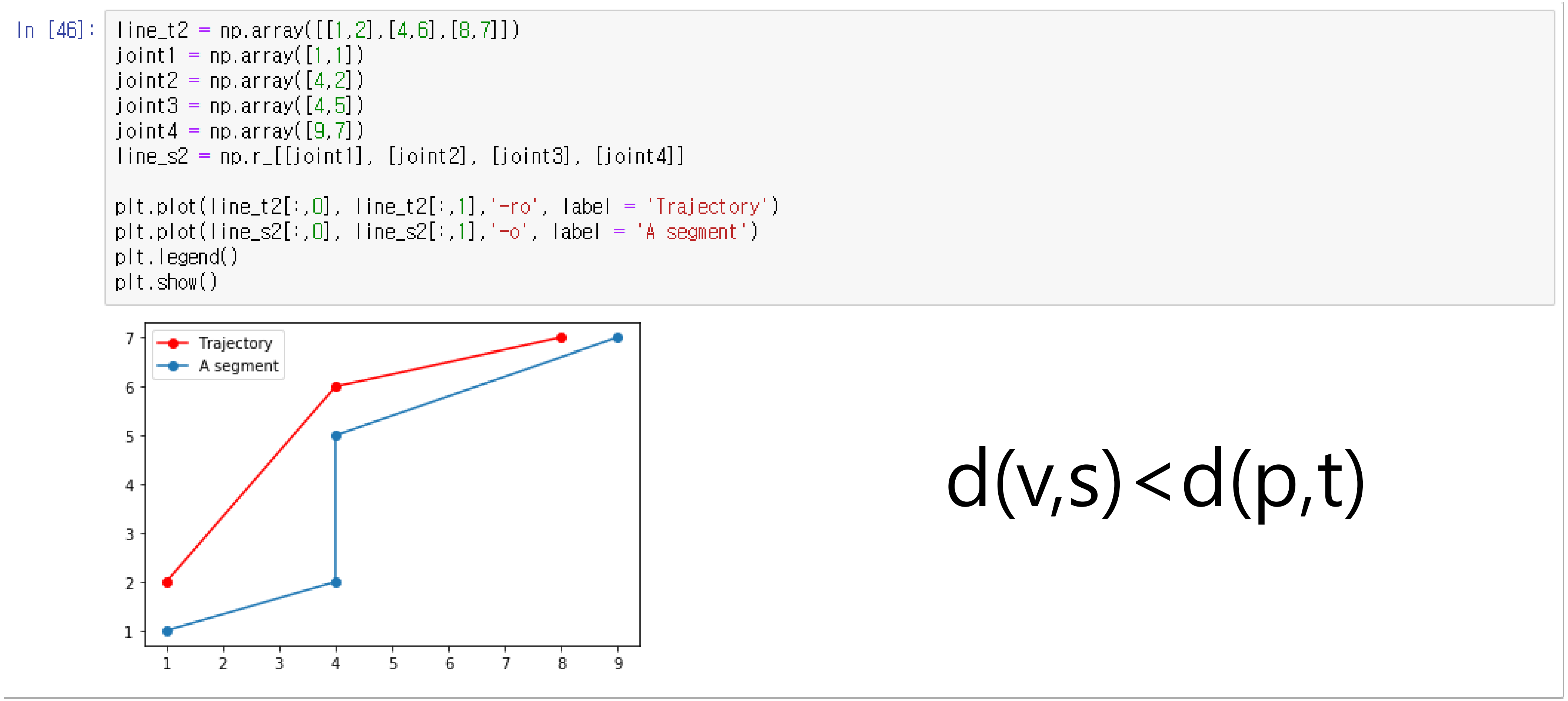
```
In [24]: line_t = np.array([[1,1],[3,3],[4,5],[9,7]])
         joint1 = np.array([1,2])
         joint2 = np.array([2,3])
         joint3 = np.array([4,7])
         joint4 = np.array([7,9])
         line_s = np.r_[[joint1], [joint2], [joint3], [joint4]]

         plt.plot(line_t[:,0], line_t[:,1],'-ro', label = 'Trajectory')
         plt.plot(line_s[:,0], line_s[:,1],'-o', label = 'A segment')
         plt.legend()
         plt.show()
```



$$d(v,s) > d(p,t)$$

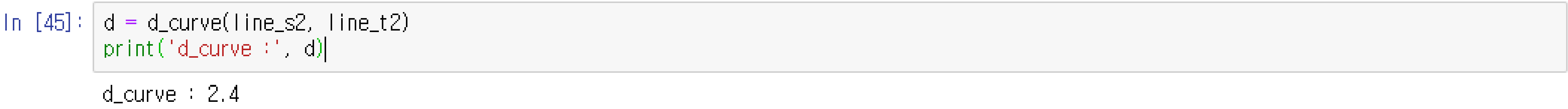


```
In [25]: d = d_curve(line_s, line_t)
         print('d_curve :', d)

         d_curve : 2.8284271247461903
```



```
In [46]: line_t2 = np.array([[1,2],[4,6],[8,7]])
         joint1 = np.array([1,1])
         joint2 = np.array([4,2])
         joint3 = np.array([4,5])
         joint4 = np.array([9,7])
         line_s2 = np.r_[[joint1], [joint2], [joint3], [joint4]]

         plt.plot(line_t2[:,0], line_t2[:,1],'-ro', label = 'Trajectory')
         plt.plot(line_s2[:,0], line_s2[:,1],'-o', label = 'A segment')
         plt.legend()
         plt.show()
```

$$d(v,s) < d(p,t)$$



```
In [45]: d = d_curve(line_s2, line_t2)
         print('d_curve :', d)

         d_curve : 2.4
```