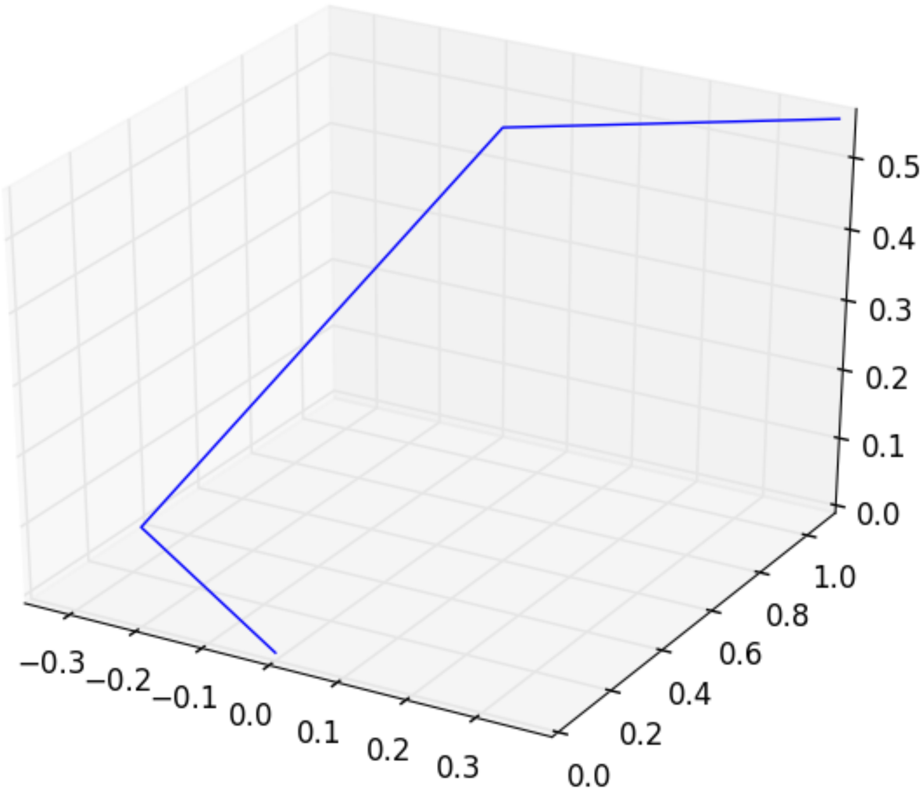
```python
#!/usr/bin/env python
import numpy as np, sympy as sy
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

def end_effector( list_of_links):
  number_of_links = len(list_of_links)
  whatami = 0
  for val in list_of_links:
    for thing in val:
      if str(type(thing)) == "<class 'sympy.core.symbol.Symbol'>":
        whatami = 1
  if whatami == 0:
    A_list = np.array([ A_matrix(val) for val in list_of_links])
  else:
    A_list = np.array([ A_sym_matrix(val) for val in list_of_links])
    print len(A_list)
  val = np.matrix(np.identity(4))
  EOL_A = []
  i = 0
  # pack a list
  for i in A_list:
    val = val*i
    EOL_A.append(val[:,3][:3])
    i = i + 1
  return val, EOL_A

def A_matrix(link_list):
  [a, al, d, th] = link_list
  A_mat = np.matrix((
    (np.cos(th), -np.sin(th)*np.cos(al), np.sin(th)*np.sin(al), a*np.cos(th)),
    (np.sin(th),  np.cos(th)*np.cos(al),-np.cos(th)*np.sin(al), a*np.sin(th)),
    (0         ,  np.sin(al)           , np.cos(al)           , d           ),
    (0         ,  0                    , 0                    , 1           ),
    ))
  return A_mat

def A_sym_matrix(link_list):
  [a, al, d, th] = link_list
  A_mat = np.matrix((
    (sy.cos(th), -sy.sin(th)*sy.cos(al), sy.sin(th)*sy.sin(al), a*sy.cos(th)),
    (sy.sin(th),  sy.cos(th)*sy.cos(al),-sy.cos(th)*sy.sin(al), a*sy.sin(th)),
    (0         ,  sy.sin(al)           , sy.cos(al)           , d           ),
    (0         ,  0                    , 0                    , 1           ),
    ))
  return A_mat

if __name__ == "__main__":
  [th1,th2,th3,th5,th6,d3] = [np.pi/4, np.pi/4, 0, np.pi/4, np.pi/4, np.pi/4]
  dh_mat = [
      [0,-np.pi/2, 0, th1],
      [0, np.pi/2,.5, th2],
      [0, 0, d3, 0],
      [0,-np.pi/2, 0, th3],
      [0, np.pi/2, 0, th5],
      [0, 0, .5, th6]
      ]
  stuff = end_effector(dh_mat)
  A06 = stuff[0]
  end_of_links = stuff[1]
  fig = plt.figure()
  ax = fig.add_subplot(111, projection='3d')
  X = np.array([ np.array(val[0])[0][0] for val in end_of_links])
  Y = np.array([ np.array(val[1])[0][0] for val in end_of_links])
  Z = np.array([ np.array(val[2])[0][0] for val in end_of_links])
  print "The position of the End Effector is"
  print A06[:,3][:3]
  print "The orientation of the End Effector is"
  print A06[:3,:3]
  # make graph
  ax.plot(X,Y,Z)
  plt.show()
  Axes3D.plot()
  ax.savefig('plot1.png')


  '''
  Answer
  ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
  The position of the End Effector is
  [[ 0.39269908]
  [ 1.09980586]
  [ 0.55536037]]
  The orientation of the End Effector is
  [[ -5.00000000e-01  -5.00000000e-01   7.07106781e-01]
  [  5.00000000e-01   5.00000000e-01   7.07106781e-01]
  [ -7.07106781e-01   7.07106781e-01   2.22044605e-16]]
  +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'''
```


−0.3
−0.2
−0.1
0.0
0.1
0.2
0.3
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.1
0.2
0.3
0.4
0.5