```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

from sympy import *
import math
import sympy as sp
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

th, al, a, d, th0, th1, th2, th3, th4, th5 = symbols('theta alpha a d theta0 theta1
theta2 theta3 theta4 theta5', real=True)
#please give all the inputs in degrees in place of the zero for all the thetas
#please uncomment all below lines to get the simplified version of all matrices
#(teta not equal to zero)
# th0 = 90
# th1 = 0
# th2 = 0
# th3 = 0
# th4 = 0
# th5 = 0
# # # # converts the theta value into radians
# th0 = (sp.rad(th0))
# th1 = (sp.rad(th1))
# th2 = (sp.rad(th2))
# th3 = (sp.rad(th3))
# th4 = (sp.rad(th4))
# th5 = (sp.rad(th5))
a =[]
b=[]


"""To compute the DH parameters matrix"""
def Transformation_matrix(a, al, d, th):
      return Matrix([
      [sp.cos(th), -sp.sin(th) * sp.cos(al), sp.sin(th) * sp.sin(al), a *
      sp.cos(th)],
      [sp.sin(th), sp.cos(th) * sp.cos(al), -sp.cos(th) * sp.sin(al), a *
      sp.sin(th)],
      [0, sp.sin(al), sp.cos(al), d],
      [0, 0, 0, 1]
      ])

dy = 0
wx = 0
wy = 0
wz = 0
t = 0

# T1 = (Transformation_matrix(0, -sp.pi/2, 1.2, th0+sp.pi/2))
T1 = (Transformation_matrix(0, sp.pi/2, 1.2, th0))
# print("T01\n")
# pprint(T1, num_columns=10_000)
# print("\n")
#Transformation matrix for T12
# print("T12\n")
T2 = (Transformation_matrix(1, 0, 0, th1+sp.pi/4))
# pprint(T2, num_columns=10_000)
```

```
# print("\n")
#Transformation matrix for T23
# print("T23\n")
T3 = (Transformation_matrix(1, 0, 0, th2-sp.pi/4))
# pprint(T3, num_columns=10_000)
# print("\n")
#Transformation matrix for T34
# print("T34\n")
# T4 = (Transformation_matrix(0,0,0,0))
# # pprint(T4, num_columns=10_000)
# # print("\n")
# #Transformation matrix for T45
# # print("T45\n")
# T5 = (Transformation_matrix(0,0,0,0))
# # pprint(T5, num_columns=10_000)
# # print("\n")
# #Transformation matrix for T56
# # print("T56\n")
# T6 = (Transformation_matrix(0, 0,0,0))
# # pprint(T6, num_columns=10_000)
# # print("\n")
# #Transformation matrix for T06
# # print("T06")
# # print("\n")
T06 = ((T1 * T2 * T3))
# print("Final Transformation Matrix:")
# print("\n")
# pprint(T06)


T02 =(  T1 * T2)
# print("T02")
# print("\n")
# pprint(T02)


T03 = (T1 * T2 * T3)
# print("T03")
# print("\n")
# pprint(N(T03))

a = T03[:, 3]
# pprint(N(a))
x=a[0]
y=a[1]
z=a[2]


# j_matrix(rad(0),0,rad(0))da_dth1 = diff(a, th1)



# Robot parameters
L1 = 1.0  # Length of link 1
L2 = 1.0  # Length of link 2

# Joint limits
theta2_limits = np.linspace(-np.pi/3, np.pi/3, 100)
theta3_limits = np.linspace(-np.pi/3, np.pi/3, 100)
```

```
end_effector_positions = []
for theta2 in theta2_limits:
    for theta3 in theta3_limits:
        # Forward kinematics equations (adjust based on your robot's kinematics)
        x = cos(theta2) + cos(theta2 + theta3)
        z = sin(theta2) + sin(theta2 + theta3)+1.2
        # print(theta2)

        end_effector_positions.append([x, z])
        # print(x)

# Convert the list to a NumPy array for easier manipulation
end_effector_positions = np.array(end_effector_positions)

# Plot the workspace
plt.scatter(end_effector_positions[:, 0], end_effector_positions[:, 1], c='b',
marker='o')
plt.xlabel('X-axis')
plt.ylabel('Z-axis')
plt.title('Workspace of the Robot (2D)')
plt.grid(True)
plt.show()
```