HW 2

William Gonzalez-Guzman
9-27-22

1. $K = \frac{1}{\sqrt{3}}(1,1,1)^T$, $\theta = 90°$. Find $R_{k,\theta}$

$$K = \frac{1}{\sqrt{3}}\begin{bmatrix}1\\1\\1\end{bmatrix} = \begin{bmatrix}1/\sqrt{3}\\1/\sqrt{3}\\1/\sqrt{3}\end{bmatrix}$$

$$R_{K,\theta} = \left[\begin{array}{c|c|c} k_x^2 V_\theta + C_\theta & k_x k_y V_\theta - k_z S_\theta & k_x k_z V_\theta + k_y S_\theta \\ k_x k_y V_\theta + k_z S_\theta & k_y^2 V_\theta + C_\theta & k_y k_z V_\theta - k_x S_\theta \\ k_x k_z V_\theta - k_y S_\theta & k_y k_z V_\theta + k_x S_\theta & k_z^2 V_\theta + C_\theta \end{array}\right]$$

$$V_\theta = 1 - C_\theta \qquad C_\theta = \cos\theta \quad S_\theta = \sin\theta$$

$$R_{k,90} = \left[\begin{array}{c|c|c} \frac{1}{3}\cdot(1-0)+0 & \frac{1}{3}(1-0) - \frac{1}{\sqrt{3}}\cdot 1 & \frac{1}{3}(1-0) + \frac{1}{\sqrt{3}}\cdot 1 \\ \frac{1}{3}\cdot(1-0) + \frac{1}{\sqrt{3}}\cdot 1 & \frac{1}{3}(1-0)+0 & \frac{1}{3}(1-0) - \frac{1}{\sqrt{3}}\cdot 1 \\ \frac{1}{3}\cdot(1-0) - \frac{1}{\sqrt{3}}\cdot 1 & \frac{1}{3}(1-0) + \frac{1}{\sqrt{3}}\cdot 1 & \frac{1}{3}(1-0)+0 \end{array}\right]$$

$$\boxed{R_{K,90} = \left[\begin{array}{c|c|c} 1/3 & 1/3 - 1/\sqrt{3} & 1/3 + 1/\sqrt{3} \\ 1/3 + 1/\sqrt{3} & 1/3 & 1/3 - 1/\sqrt{3} \\ 1/3 - 1/\sqrt{3} & 1/3 + 1/\sqrt{3} & 1/3 \end{array}\right]}$$

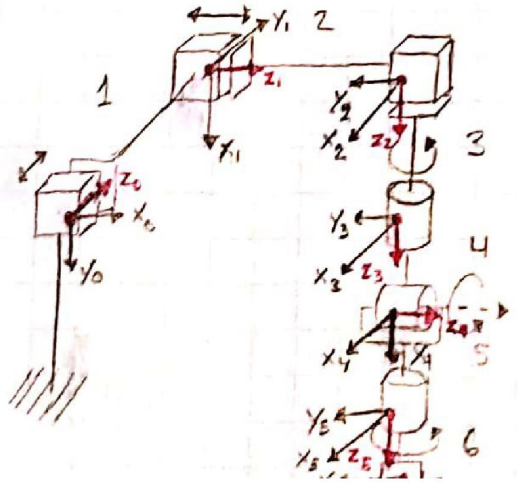
2. $$\begin{bmatrix}0 & 0 & 1\\ \sqrt{2}/2 & \sqrt{2}/2 & 0\\ -\sqrt{2}/2 & \sqrt{2}/2 & 0\end{bmatrix} = R_{K,\theta}$$

$$\theta = \cos^{-1}\left(\frac{Tr(R_{k,\theta})-1}{2}\right)$$

$$\theta = \cos^{-1}\left(\frac{\sqrt{2}/2 - 1}{2}\right) = 98,421$$

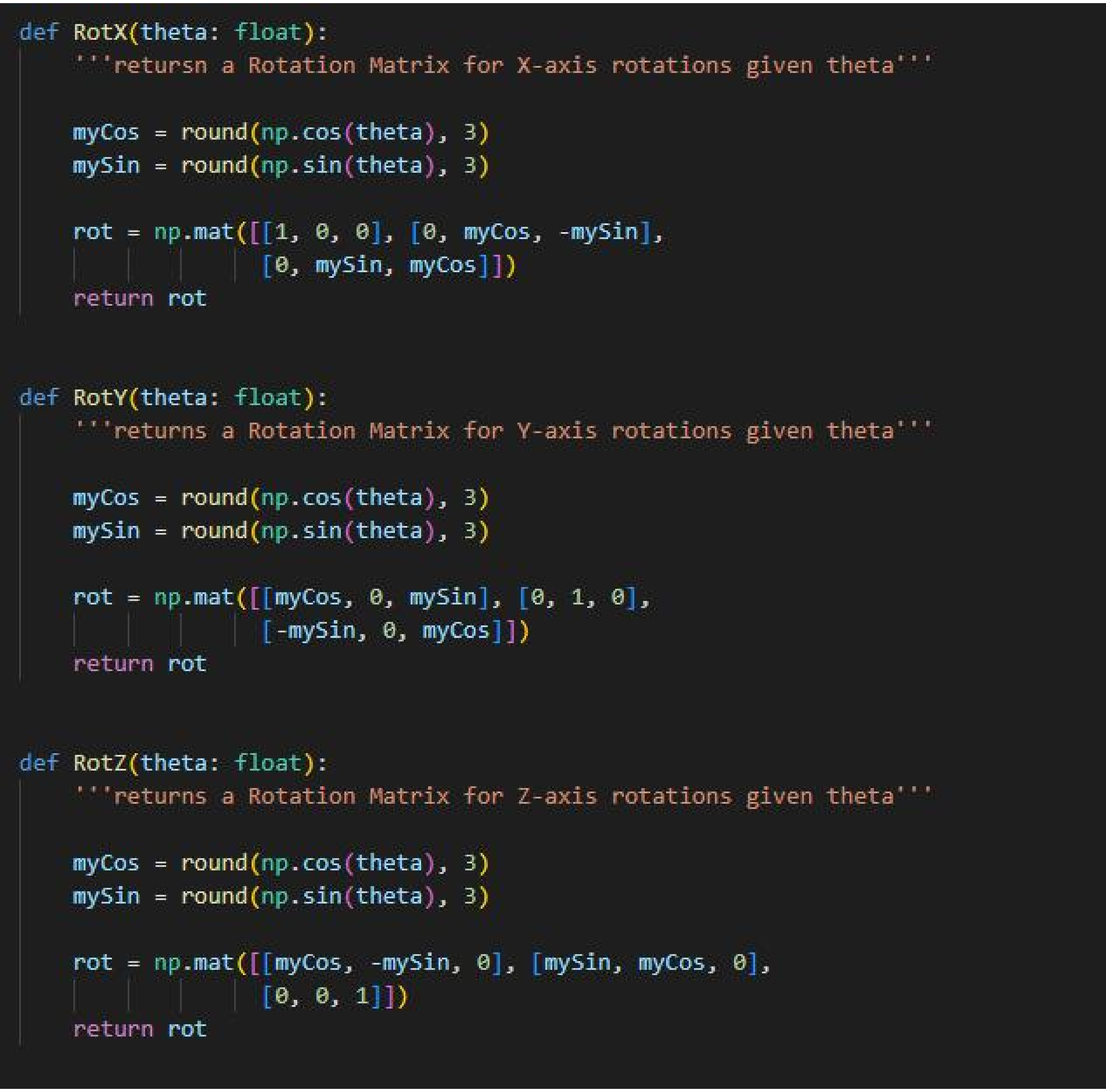
$$K = \frac{1}{2\sin\theta}\begin{bmatrix}r_{32}-r_{23}\\ r_{13}-r_{31}\\ r_{21}-r_{12}\end{bmatrix} = \frac{1}{1.978}\begin{bmatrix}\sqrt{2}/2 - 0\\ 1 + \frac{\sqrt{2}}{2}\\ \sqrt{2}/2 - 0\end{bmatrix} = \begin{bmatrix}0,357\\0,863\\0,357\end{bmatrix}$$

3. a)

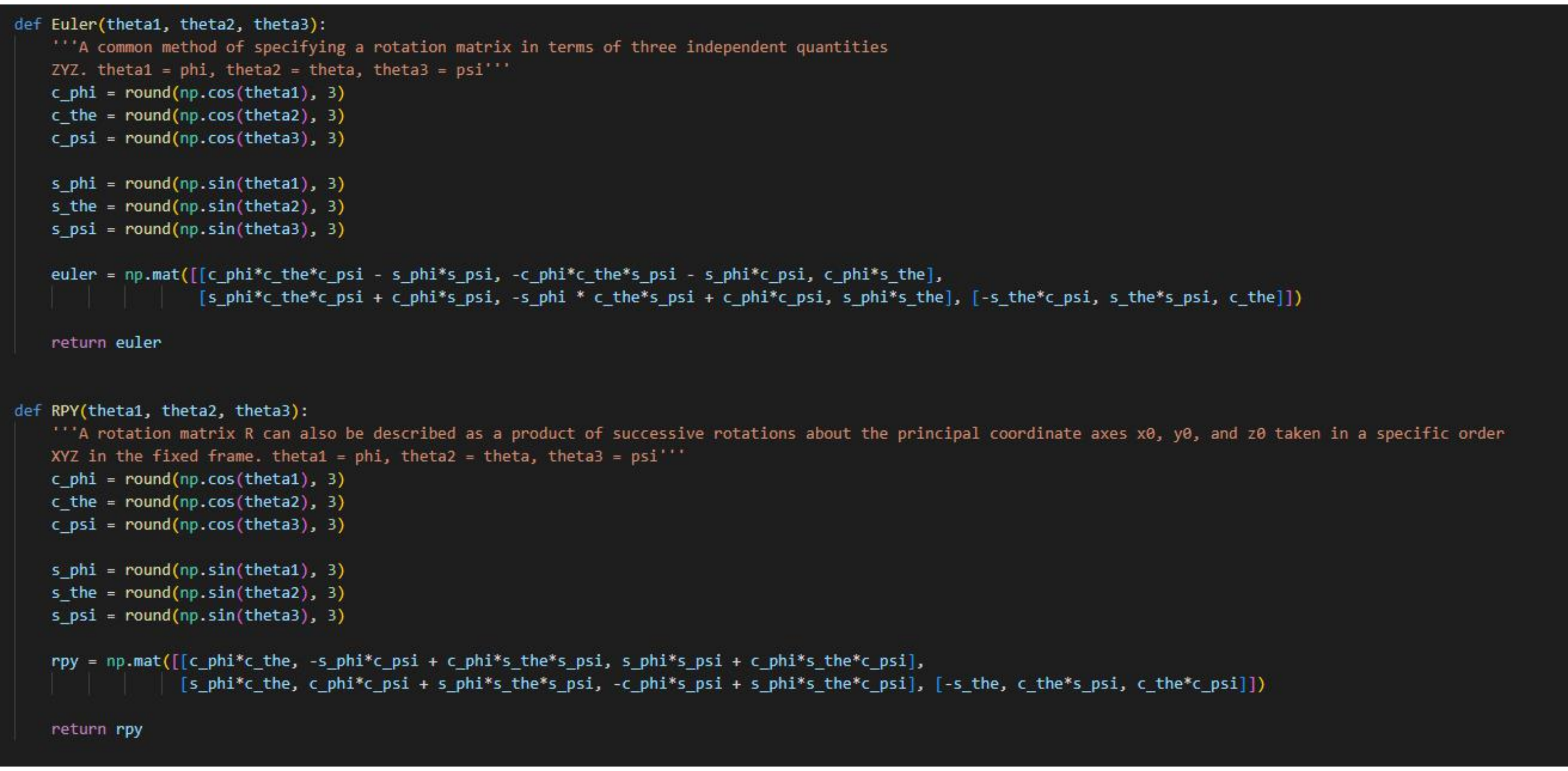


b)

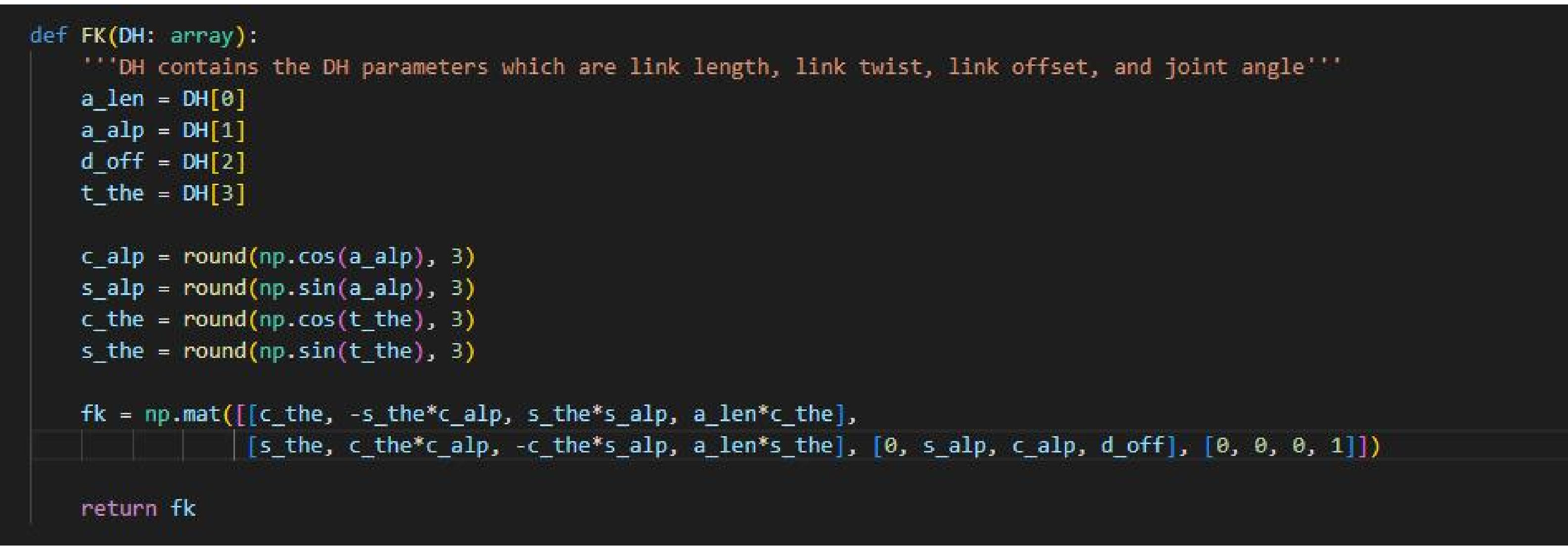
link	a_i	α_i	d_i	θ_i
1	0	+90	d_1^*	90
2	0	-90	d_2^*	-90
3	0	0	d_3^*	0
4	0	90	0	0
5	0	-90	0	0
6	0	0	d_6	0



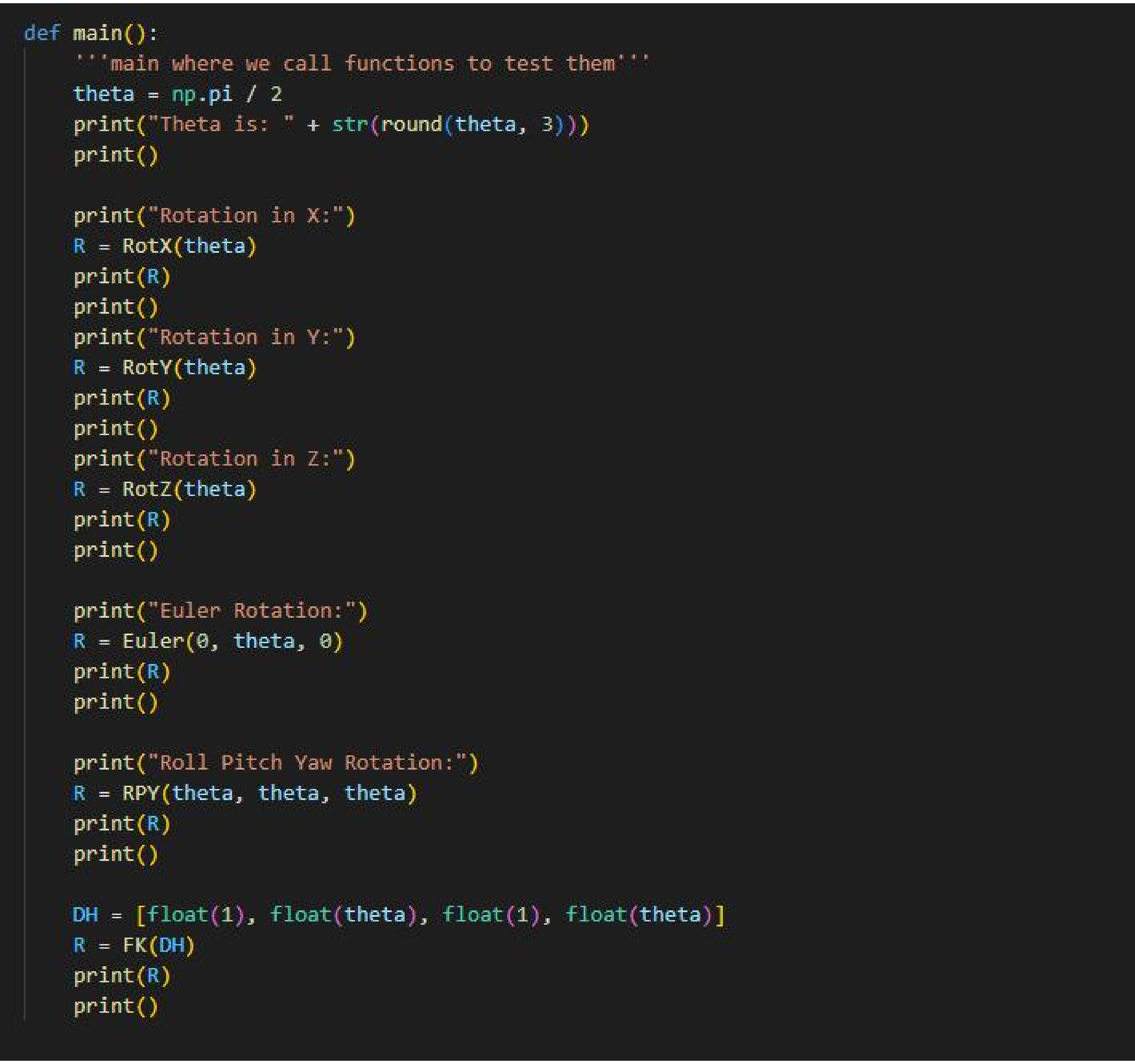
```
def RotX(theta: float):
    '''retursn a Rotation Matrix for X-axis rotations given theta'''

    myCos = round(np.cos(theta), 3)
    mySin = round(np.sin(theta), 3)

    rot = np.mat([[1, 0, 0], [0, myCos, -mySin],
                  [0, mySin, myCos]])
    return rot


def RotY(theta: float):
    '''returns a Rotation Matrix for Y-axis rotations given theta'''

    myCos = round(np.cos(theta), 3)
    mySin = round(np.sin(theta), 3)

    rot = np.mat([[myCos, 0, mySin], [0, 1, 0],
                  [-mySin, 0, myCos]])
    return rot


def RotZ(theta: float):
    '''returns a Rotation Matrix for Z-axis rotations given theta'''

    myCos = round(np.cos(theta), 3)
    mySin = round(np.sin(theta), 3)

    rot = np.mat([[myCos, -mySin, 0], [mySin, myCos, 0],
                  [0, 0, 1]])
    return rot
```



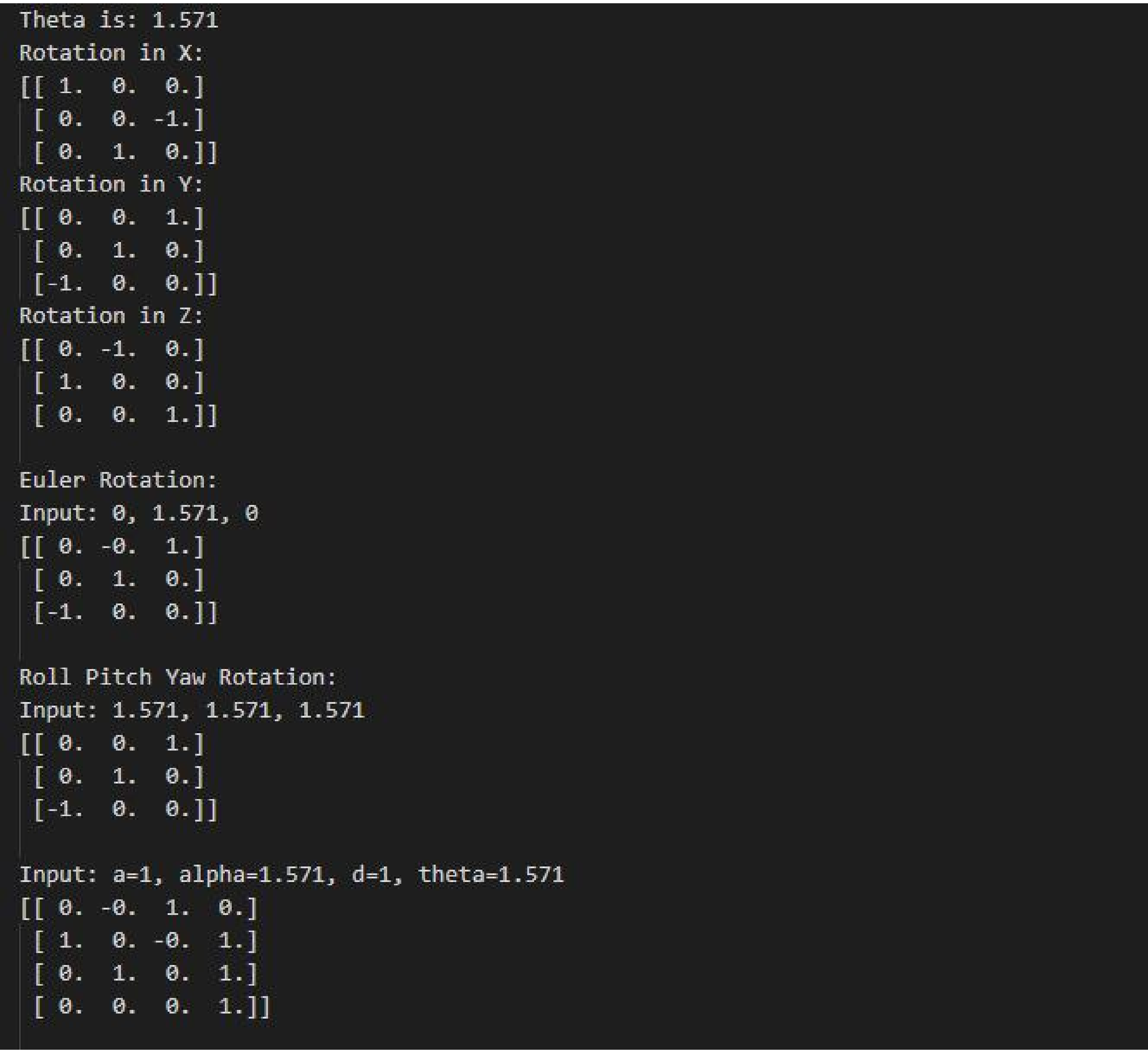
```python
def Euler(theta1, theta2, theta3):
    '''A common method of specifying a rotation matrix in terms of three independent quantities
    ZYZ. theta1 = phi, theta2 = theta, theta3 = psi'''
    c_phi = round(np.cos(theta1), 3)
    c_the = round(np.cos(theta2), 3)
    c_psi = round(np.cos(theta3), 3)

    s_phi = round(np.sin(theta1), 3)
    s_the = round(np.sin(theta2), 3)
    s_psi = round(np.sin(theta3), 3)

    euler = np.mat([[c_phi*c_the*c_psi - s_phi*s_psi, -c_phi*c_the*s_psi - s_phi*c_psi, c_phi*s_the],
                    [s_phi*c_the*c_psi + c_phi*s_psi, -s_phi * c_the*s_psi + c_phi*c_psi, s_phi*s_the], [-s_the*c_psi, s_the*s_psi, c_the]])

    return euler


def RPY(theta1, theta2, theta3):
    '''A rotation matrix R can also be described as a product of successive rotations about the principal coordinate axes x0, y0, and z0 taken in a specific order
    XYZ in the fixed frame. theta1 = phi, theta2 = theta, theta3 = psi'''
    c_phi = round(np.cos(theta1), 3)
    c_the = round(np.cos(theta2), 3)
    c_psi = round(np.cos(theta3), 3)

    s_phi = round(np.sin(theta1), 3)
    s_the = round(np.sin(theta2), 3)
    s_psi = round(np.sin(theta3), 3)

    rpy = np.mat([[c_phi*c_the, -s_phi*c_psi + c_phi*s_the*s_psi, s_phi*s_psi + c_phi*s_the*c_psi],
                  [s_phi*c_the, c_phi*c_psi + s_phi*s_the*s_psi, -c_phi*s_psi + s_phi*s_the*c_psi], [-s_the, c_the*s_psi, c_the*c_psi]])

    return rpy
```



```python
def FK(DH: array):
    '''DH contains the DH parameters which are link length, link twist, link offset, and joint angle'''
    a_len = DH[0]
    a_alp = DH[1]
    d_off = DH[2]
    t_the = DH[3]

    c_alp = round(np.cos(a_alp), 3)
    s_alp = round(np.sin(a_alp), 3)
    c_the = round(np.cos(t_the), 3)
    s_the = round(np.sin(t_the), 3)

    fk = np.mat([[c_the, -s_the*c_alp, s_the*s_alp, a_len*c_the],
                 [s_the, c_the*c_alp, -c_the*s_alp, a_len*s_the], [0, s_alp, c_alp, d_off], [0, 0, 0, 1]])

    return fk
```



```
def main():
    '''main where we call functions to test them'''
    theta = np.pi / 2
    print("Theta is: " + str(round(theta, 3)))
    print()

    print("Rotation in X:")
    R = RotX(theta)
    print(R)
    print()
    print("Rotation in Y:")
    R = RotY(theta)
    print(R)
    print()
    print("Rotation in Z:")
    R = RotZ(theta)
    print(R)
    print()

    print("Euler Rotation:")
    R = Euler(0, theta, 0)
    print(R)
    print()

    print("Roll Pitch Yaw Rotation:")
    R = RPY(theta, theta, theta)
    print(R)
    print()

    DH = [float(1), float(theta), float(1), float(theta)]
    R = FK(DH)
    print(R)
    print()
```



```
Theta is: 1.571
Rotation in X:
[[ 1.  0.  0.]
 [ 0.  0. -1.]
 [ 0.  1.  0.]]
Rotation in Y:
[[ 0.  0.  1.]
 [ 0.  1.  0.]
 [-1.  0.  0.]]
Rotation in Z:
[[ 0. -1.  0.]
 [ 1.  0.  0.]
 [ 0.  0.  1.]]

Euler Rotation:
Input: 0, 1.571, 0
[[ 0. -0.  1.]
 [ 0.  1.  0.]
 [-1.  0.  0.]]

Roll Pitch Yaw Rotation:
Input: 1.571, 1.571, 1.571
[[ 0.  0.  1.]
 [ 0.  1.  0.]
 [-1.  0.  0.]]

Input: a=1, alpha=1.571, d=1, theta=1.571
[[ 0. -0.  1.  0.]
 [ 1.  0. -0.  1.]
 [ 0.  1.  0.  1.]
 [ 0.  0.  0.  1.]]
```