```
r""" This module is designed to test the functions contained in the
module RV2COE

Author: Thomas J Susi
"""

import numpy as np

from astro import RV2COE

r""" Tests for Level 0
"""
def test_unit_vector():
    r""" Test for the Unit Vector Function
    """

    vector = [1, 1, 1]
    actual_out = RV2COE.unit_vector(vector)
    expected_out = [1 / np.sqrt(3), 1 / np.sqrt(3), 1 / np.sqrt(3)]

    np.testing.assert_allclose(actual_out, expected_out)


r""" Tests for Level 1
"""
def test_ang_momentum():
    r""" Test for Angular Momentum Function
    """

    r_vector = [1, 2, 3]
    v_vector = [4, 5, 6]
    actual_out = RV2COE.ang_momentum(r_vector, v_vector)
    expected_out = [-3, 6, -3]

    np.testing.assert_allclose(actual_out, expected_out)

def test_eccentricity():
    r""" Test for Eccentricity Function
    """

    mu = 10
    r_vector = [1, 2, 3]
    v_vector = [4, 5, 6]
    h_vector = [-3, 6, -3]
    actual_out = RV2COE.eccentricity(mu, r_vector, v_vector, h_vector)
    expected_out = [(-51/10)-(1/np.sqrt(14)), (-3/5)-np.sqrt(2/7),
(39/10)-(3/np.sqrt(14))]

    np.testing.assert_allclose(actual_out, expected_out)
```

```
def test_spec_mech_energy():
    r""" Test for Angular Momentum Function
    """

    mu = 10
    r_vector = [1, 2, 3]
    v_vector = [4, 5, 6]
    actual_out = RV2COE.spec_mech_energy(mu, r_vector, v_vector)
    expected_out = (77/2)-(10/np.sqrt(14))

    np.testing.assert_allclose(actual_out, expected_out)


r""" Tests for Level 2
"""
def test_line_of_nodes():
    r""" Test for Line of Nodes Function
    """

    h_hat = [1,2,3]
    actual_out = RV2COE.line_of_nodes(h_hat)
    expected_out = [-2, 1, 0]

    np.testing.assert_allclose(actual_out, expected_out)

def test_semi_latus_rectum():
    r""" Test for Semi Latus Rectum Function
    """

    mu = 10
    h_vector = [-3,6,-3]
    actual_out = RV2COE.semi_latus_rectum(mu, h_vector)
    expected_out = 5.4

    np.testing.assert_allclose(actual_out, expected_out)

def test_true_anom():
    r""" Test for True Anomaly Function
    """

    r_vector = [1, 2, 3]
    h_vector = [-3, 6, -3]
    e_vector = [2, 3, 4]
    actual_out = RV2COE.true_anom(r_vector, h_vector, e_vector)
    expected_out = -0.12186756768575521

    np.testing.assert_allclose(actual_out, expected_out)

def test_inclination():
    r""" Test for Inclination Function
```

```
    """

    h_hat = [.25, .75, .5]
    actual_out = RV2COE.inclination(h_hat)
    expected_out = 1.0471975511965976

    np.testing.assert_allclose(actual_out, expected_out)

def test_semi_major_axis():
    r""" Test for Semi Major Axis Function
    """

    p = .75
    e = .5
    actual_out = RV2COE.semi_major_axis(p, e)
    expected_out = 1

    np.testing.assert_allclose(actual_out, expected_out)

def test_R_A_A_N():
    r""" Test for RAAN Function
    """

    n_vector = [.25, .5, .75]
    actual_out = RV2COE.R_A_A_N(n_vector)
    expected_out = 1.1071487177940904

    np.testing.assert_allclose(actual_out, expected_out)

def test_arg_of_periapsis():
    r""" Test for Argument of Periapsis Function
    """

    r_vector = [1, 2, 3]
    h_vector = [-3, 6, -3]
    e_vector = [2, 3, 4]
    actual_out = RV2COE.arg_of_periapsis(r_vector, e_vector, h_vector)
    expected_out = 0.12186756768575521

    np.testing.assert_allclose(actual_out, expected_out)

def test_rad_peri():
    r""" Test for Radius of Periapsis Function
    """

    p = 1
    e = .5
    actual_out = RV2COE.rad_peri(p, e)
    expected_out = .6666666666666
```

```
    np.testing.assert_allclose(actual_out, expected_out)

def test_rad_apo():
    r""" Test for Radius of Apoapsis Function
    """

    p = 1
    e = .5
    actual_out = RV2COE.rad_apo(p, e)
    expected_out = 2

    np.testing.assert_allclose(actual_out, expected_out)

def test_flight_ang():
    r""" Test for Radius of Apoapsis Function
    """

    e = .5
    theta = 1.2
    actual_out = RV2COE.flight_ang(e, theta)
    expected_out = 0.37578860669992659

    np.testing.assert_allclose(actual_out, expected_out)

def test_Orbit_Period():
    r""" Test for Orbital Period Function
    """

    mu = 2
    a = 2
    actual_out = RV2COE.Orbit_Period(mu, a)
    expected_out = 12.5663706144

    np.testing.assert_allclose(actual_out, expected_out)
```