1) Function Name: get_mass_rover
    a. Calling Syntax: get_mass_rover(edl_system)
    b. Description: This function computes the mass of the rover defined in the rover field of the edl system structure. The mass is calculated as follows: 6 * motor mass + 6 * speed reducer mass + 6 * wheel mass + chassis mass, science payload mass + power subsystem mass. The input, edl_subsystem, must be a dictionary.
    c. Input Arguments:
        i. edl_subsystem
            1. Type: Dictionary
            2. Definition: A data structure describing characteristics of the entire EDL system
    d. Output Arguments:
        i. m
            1. Type: Scalar
            2. Definition: A positive scalar stating the total mass of the rover
            3. Units: Kilograms (Kg)
2) Function Name: get_mass_rockets
    a. Calling Syntax: get_mass_rockets(edl_system)
    b. Description: This function computes and returns the mass of all the rockets on the edl system. The mass is calculated as follows: number of rockets * structural mass of rockets + the rocket's fuel mass. The input, edl_subsystem, must be a dictionary.
    c. Input Arguments:
        i. edl_subsystem
            1. Type: Dictionary
            2. Definition: A data structure describing characteristics of the entire EDL system
    d. Output Arguments:
        i. m
            1. Type: Scalar
            2. Definition: A positive scalar stating the total mass of the rockets on the EDL System
            3. Units: Kilograms (Kg)
3) Function Name: get_mass_edl
    a. Calling Syntax: get_mass_edl(edl_system)
    b. Description: This function computes and returns the total current mass of the EDL system. To do this, it checks if the parachute and/or the heat shield is still attached. If not, the mass of these components is removed from the returned value. In addition to the mass of the parachute and heat shield, the function sums the mass of the rockets (using get_mass_rockets), the mass of the rover (using get_mass_rover), and the mass of the sky crane system. The input, edl_subsystem, must be a dictionary
    c. Input Arguments:
        i. edl_subsystem

                1. Type: Dictionary
                2. Definition: A data structure describing characteristics of the entire EDL system
    d. Output Arguments:
        i. m
            1. Type: Scalar
            2. Definition: A positive scalar stating the total current mass of the entire EDL System
            3. Units: Kilograms (Kg)

4) get_local_atm_properties
    a. Calling Syntax: get_local_atm_properties(planet, altitude)
    b. Description: Returns the local atmospheric properties at a given altitude. This function returns a temperature and a pressure of the atmosphere depending on if the current altitude of the EDL module is above or below a set threshold. The specific altitude is inputted into a function which then returns a dictionary of temperature and pressure. Once these pressure and temperature values are determined, an air density value is also calculated and returned. The input planet must be a dictionary. The input altitude must be a scalar and cannot be inputted as a vector. The function returns the pressure, temperature, and density as a tuple, in that order. It is important to note, however, that the pressure, temperature, and density can be stored independently of the other 2
    c. Input Arguments:
        i. planet
            1. Type: Dictionary
            2. Definition: A data structure describing characteristics of the planet that the EDL is entering, descending, or landing on.
        ii. altitude
            1. Type: Scalar
            2. Definition: The number of meters the EDL Module is away from the planet's surface
            3. Units: meters (m)
    d. Output Arguments:
        i. density
            1. Type: Scalar
            2. Definition: A scalar describing the air density of the planet's atmosphere at a pressure and temperature derived from the altitude of the EDL module.
            3. Units: Kilograms (Kg)
        ii. temperature
            1. Type: Scalar
            2. Definition: A scalar describing the temperature based on the planet's atmospheric data and the distance above the planet's surface.
            3. Units: Celsius (°C)

        iii. pressure
            1. Type: Scalar
            2. Definition: A scalar describing the atmospheric pressure based on the planet's atmospheric characteristics and a specified altitude
            3. Units: Kilopascals (KPa)

5) F_buoyancy_descent
    a. Calling Syntax: F_buoyancy_descent(edl_system, planet, altitude)
    b. Describe: This function utilizes the density from get_local_atm_properties, the accelertation due to gravity from the planet's dictionary, and the volume of the EDL System from its data structure. These values are utilized to calculate the buoyancy force which is pointing strictly upwards. The formula to calculate this force is density * acceleration due to gravity * volume of the system. This function is not vectorized as the helper function, get_local_atm_properties, does not output vectors.
    c. Input Arguments:
        i. edl_system
            1. Type: Dictionary
            2. Definition: A data structure describing characteristics of the entire EDL system
        ii. planet
            1. Type: Dictionary
            2. Definition: A data structure describing characteristics of the planet
        iii. altitude
            1. Type: Scalar
            2. Definition: Distance from planet's surface to EDL Module at that time
            3. Units: meters (m)
    d. Output Arguments
        i. F
            1. Type: Scalar
            2. Definition: The buoyancy force exerted on the EDL system in the upward direction and calculated by density * acceleration due to gravity * volume of system
            3. Units: Newtons (N)

6) F_drag_descent
    a. Calling Syntax: F_drag_descent(edl_system, planet, altitude, velocity)
    b. Describe: This function calculates and returns the drag force on the EDL Module. The drag force is defined as 0.5 * density * EDL velocity^2 * area * drag coefficient. The drag coefficient and area depend on whether the heat shield has been ejected or not. Furthermore, this function also calculates the drag force based on the status of the parachute and if it has been deployed or ejected. The function considers the drag force for all these possibilities. It is important to note

that this function is not vectorized because the helper function, get_local_atm_properties cannot return a vector.

c. Input Arguments
    i. edl_system
        1. Type: Dictionary
        2. Definition: A data structure describing characteristics of the entire EDL system
    ii. planet
        1. Type: Dictionary
        2. Definition: A data structure describing characteristics of the planet
    iii. altitude
        1. Type: Scalar
        2. Definition: Distance from planet's surface to EDL Module at that time
        3. Units: meters (m)
    iv. velocity
        1. Type: Scalar
        2. Definition: The velocity of the EDL System relative to the air that it is flying through.
        3. Units: meters per second (m/s)

d. Output Arguments
    i. F
        1. Type: Scalar
        2. Definition: The ultimate drag force created from the sum of the air drag and either the parachute, heat shield, or sky crane module drag.
        3. Units: Newtons (N)

7) F_gravity_descent
    a. Calling Syntax: F_gravity_descent(edl_system, planet)
    b. Describe: Computes and returns the gravitational force acting on the EDL System. It is governed by EDL Mass * acceleration due to gravity.
    c. Input Arguments
        i. edl_system
            1. Type: Dictionary
            2. Definition: A data structure describing characteristics of the entire EDL system
        ii. planet
            1. Type: Dictionary
            2. Definition: A data structure describing characteristics of the planet
    d. Output Arguments
        i. F
            1. Type: Scalar

            2. Definition: Gravitational force acting on the EDL system
            3. Units: Newtons (N)

8) v2M_Mars
    a. Calling Syntax: v2M_Mars(v, a)
    b. Describe: This function converts a velocity, inputted in m/s, to a Mach number on Mars as a function of altitude. It utilizes the data points for the speed of sound vs. altitude and creates a fit around these points. It is important to note that the function will return the absolute value Mach number.
    c. Input Arguments
        i. v
            1. Type: Scalar or Vector
            2. Definition: The velocity of the EDL as a scalar or a list
            3. Units: meters per second (m/s)
        ii. a
            1. Type: Scalar or Vector
            2. Definition: The current altitude of the EDL with respects to the planet's surface as a scalar or a vector
            3. Units: meters (m)
    d. Output Arguments
        i. M
            1. Type: Scalar or Vector
            2. Definition: The Mach number on Mars
            3. Units: dimensionless

9) thurst_controller
    a. Calling Syntax: thrust_controller(edl_system, planet)
    b. Describe: This function implements a PID controller for the EDL System. It uses edl_system, a data structure characterizing the EDL System, and planet, a data structure to characterize the planet, to create a modified edl_system data structure. It modifies fields in rocket and telemetry sub data structures depending on
    c. Input Arguments:
        i. edl_system
            1. Type: Dictionary
            2. Definition: A data structure describing characteristics of the entire EDL system.
        ii. planet
            1. Type: Dictionary
            2. Definition: A data structure describing characteristics of the planet.
    d. Output Arguments
        i. edl_system
            1. Type: Dictionary
            2. Definition: Edited edl_system data structure with edited rocket and telemetry sub data structures.

10) edl_events
    a. Calling Syntax: edl_events(edl_system, mission_events)
    b. Describe: This function describes events that occur in the EDL System Simulation. It will eject the heat shield at the appropriate altitude. The function will also eject the parachute at the appropriate altitude. Furthermore, this function notates when rockets should turn on, when the crane and altitude control system should turn on, when the rockets are out of fuel, if the EDL System crashes, when the speed controlled decent is required, when altitude control is required, and when the rover has touched down on the surface. A list of these events is returned
    c. Input Arguments
        i. edl_system
            1. Type: Dictionary
            2. Definition: A data structure describing characteristics of the entire EDL system.
        ii. mission_events
            1. Type: Dictionary
            2. Definition: A data structure describing the altitudes of when certain mission events should occur.
    d. Output Arguments
        i. events
            1. Type: List
            2. Definition: A list of event outcomes including altitude to eject heat shield, altitude to eject parachute, altitude to turn on rockets, altitude to turn on crane and altitude control, when the rockets are out of fuel, if the EDL System crashed at zero altitude, the speed at which the speed controlled descent and altitude control system is required, and if the rover touches down on the surface.

11) edl_dynamics
    a. Calling Syntax: edl_dynamics(t, y, edl_system, planet)
    b. Describe: This function describes the dynamics of the EDL as it decends and lowers the rover onto the surface of Mars. Because this is a variable mass system, the forces are described as momentum changes with time. In addition to the forces calculated in previous functions, this functions takes into account the various modes in which the EDL can run in. The dynamics of the EDL will differ as the system operates its rockets, position controller, and speed controller. It can differentiate its dynamic properties if the rockets are on or off. In addition, the function will calculate the error in the velocity and acceleration based on its current and excepted status.
    c. Input Arguments
        i. t
            1. Type: Scalar or Vector

            2. Definition: The individual or array of times at which the dynamic properties of the EDL want to be found
            3. Units: Seconds (s)
        ii. y
            1. Type: Scalar or Vector
            2. Definition: The individual or array of altitudes at which the dynamic properties of the EDL want to be found
            3. Units: Meters (m)
        iii. edl_system
            1. Type: Dictionary
            2. Definition: A data structure describing characteristics of the entire EDL system.
        iv. planet
            1. Type: Dictionary
            2. Definition: A data structure describing characteristics of the planet.
    d. Output Arguments
        i. dydt
            1. Type: numpy array
            2. Definition: A numpy array of values describing acceleration of the EDL, velocity of the EDL, change of mass of the rockets due to fuel burn, error signal of the velocity, error signal of the position, the rover's accelerations relative to the sky crane, and the rover's velocity relative to the sky crane.
            3. Units: [m/s^2, m/s, kg/s, m/s, m, m/s^2, m/s]

12) update_edl_state
    a. Calling Syntax: update_edl_state(edl_system, TE, YE, INTER_INFO)
    b. Describe: This function updates the characteristics of the edl_system based on the simulation events that occur. It also decides if the simulation should be terminated due to the rover landing or a terminal failure on the EDL system. The rocket's mass is also updated as the fuel is expelled from the system
    c. Input Arguments
        i. edl_system
            1. Type: Dictionary
            2. Definition: A data structure describing characteristics of the entire EDL system.
        ii. TE
            1. Type: numpy array
            2. Definition: List of times that events occur during the EDL simulation.
            3. Units: Seconds (s)
        iii. YE
            1. Type: 2-D numpy array

        2. Definition: List of state vectors describing the EDL altitude and velocity during the EDL Simulations.
        3. Units: [m/s, m]
    iv. INTER_INFO
        1. Type: Boolean
        2. Definition: If True, the function will display iteration information at each stage. Otherwise, no information will be displayed.
  d. Output Arguments
    i. edl_system
        1. Type: Dictionary
        2. Definition: A data structure describing characteristics of the entire EDL system.
    ii. y0
        1. Type: numpy array
        2. Definition: The state of the system once the EDL system is updated. These are initial conditions to solve ODE IVPs in the next function. It includes the velocity, altitude, rocket fuel mass,
        3. Units: [m/s, m/s^2, kg]

13) simulate_edl
  a. Calling Syntax: simulate_edl(edl_system, planet, mission_events, tmax, INTER_INFO)
  b. Describe: This function simulates the EDL system. It uses IVP solvers to solve the function created in edl_dynamics. It uses initial conditions and updated initial conditions from update_edl_state to do so. The function continues to solve the updated EDL system until the simulation reaches the maximum time or once the update_edl_state function returns a signal to terminate the simulation.
  c. Input Arguments
    i. edl_system
        1. Type: Dictionary
        2. Definition: A data structure describing characteristics of the entire EDL system.
    ii. planet
        1. Type: Dictionary
        2. Definition: A data structure describing characteristics of the planet.
    iii. mission_events
        1. Type: Dictionary
        2. Definition: A data structure describing the altitudes of when certain mission events should occur.
    iv. tmax
        1. Type: Scalar
        2. Definition: Maximum simulation time
        3. Units: Seconds (s)
    v. INTER_INFO

        1. Type: Boolean
        2. Definition: If True, the function will display iteration information at each stage. Otherwise, no information will be displayed.

d. Output Arguments
    i. T
        1. Type: numpy array
        2. Definition: numpy array of the t solution from the ODE IVP solution.
        3. Units: Seconds (s)
    ii. Y
        1. Type: 2-D numpy array
        2. Definition: 2-D numpy array of the altitude solutions from the ODE IVP solution.
        3. Units: Meters (m)
    iii. edl_system
        1. Type: Dictionary
        2. Definition: An updated data structure describing characteristics of the entire EDL system.