```python
#                   TITLE BLOCK
#************************************************
#Author:    Brandon White
#Date:      08/26/2019
#Desc:      Creates a MAV object with mass, moment
#           of inertia, and gravity properties
#************************************************

from rotations import *

#Calling the class with an aircraft name below creates an MAV object
class MAV:
    def __init__(self, aircraft = "None"):
       #All units listed in English units as denoted
       self.name = aircraft
       self.mass = 10  # Mass (lbf)
       #Inert = [Ixz, Ix, Iy, Iz]
       self.inert = [20, 10, 10, 10]  # Moment of Inertia (lbf*ft^2)
       self.gravity_needed = False
       #State = [p_n, p_e, p_d, u, v, w, e0, e1, e2, e3, p, q, r]
       self.state0 = [0, 0, -500, 50, 0, 0, 1, 0, 0, 0, 0, 0, 0]
            #Level flight at 500 ft at 50 ft/s
       #FM = [Fx, Fy, Fz, Ell, M, N]
       self.FM = [0, 0, 0, 0, 0, 0]
            #Gravity ONLY in base model
       self.FMeq = [0, 0, (lambda t: 32.2*self.mass), 0, 0, 0]


       if aircraft != "None":
           try:
               method_to_call = getattr(self, aircraft.lower())
               method_to_call()
           except:
               print("No preconfig by given name: " +
               aircraft.lower())

    def update_mass(self, new_mass):
        #NOTE: Automatically updates gravity force in FM
        self.mass = new_mass

    def update_inert(self, new_inert):
        self.inert = new_inert

    def update_state0(self, new_state):
        if len(new_state) != 13:
```

```
44          if len(new_state) != 13:
45              print("Error - Not 13 items! \n You might need to convert
•               angular\
46                      values to quaternions...")
47          else:
48              self.state0 = new_state
49
50      def update_FM(self, t):
51          from math import sin, cos
52          from white_brandon_HW1 import EP2Euler321
53
54          #Angularize Gravity
55          angles = EP2Euler321(self.state0[6:10])
56          Fg = f2b(angles, [0, 0, 32.2*self.mass])
57
58          #All Other Forcing Functions
59          for i in range(6):
60              try:
61                  self.FM[i] = self.FMeq[i](t)
62              except:
63                  self.FM[i] = self.FMeq[i]
64
65          #Add in Gravity
66          self.FM[0] += Fg[0]
67          self.FM[1] += Fg[1]
68          self.FM[2] += Fg[2]
69
70          return self.FM
71
72      #Add templated aircraft below this line to pregenerate aircraft
73      def hw1_1(self):
74          self.state0 = [100, 200, -500, 50, 0, 0,
75                         0.70643, 0.03084, 0.21263, 0.67438, 0, 0, 0]
76          self.FMeq = [0, 0, 0, 0, 0, 0]
77
78      def hw1_2(self):
79          from math import sin, cos
80          self.state0 = [100, 200, -500, 50, 0, 0,
81                         0.70643, 0.03084, 0.21263, 0.67438, 0, 0, 0]
82          self.FMeq = [(lambda t: sin(t)), 0, 0,
83                         0, 1e-4, 0]
84
```