```python
# this file is intended to process and extract meaningful data from the inputs
import numpy as np
from math import *
from Inputs import Aircraft
import matplotlib.pyplot as plt
#______________________________________________________________________#



# Declare varaible
# Typically, intended to have correct sizes regardless of content

# Item 1
CL  = np.linspace  (0,1,Aircraft.steps, endpoint = True)
CD  = np.linspace  (0,1,Aircraft.steps, endpoint = True)

# Item 2
M_at_CL_max  = np.linspace  (0,1,Aircraft.steps, endpoint = True)
CD_at_CL_max =   np.linspace  (0,1,Aircraft.steps, endpoint = True)

# Item 3
Speed_of_Sound = np.linspace (0,1,Aircraft.steps, endpoint = True)
V_M48 = np.linspace (0,1,Aircraft.steps, endpoint = True)

# Item 4
V_stall = np.linspace (0,1,Aircraft.steps, endpoint = True)

# Item 5
V_maxR = np.linspace (0,1,Aircraft.steps, endpoint = True)
a = 1
#______________________________________________________________________#



# Drag Profile
# Item 1 - [C_D vs C_L]
# Note: highlight points of min drag & min power & max range
# Include Drag variation when C_L is flexible vs fixed
CL_min_drag  = sqrt(Aircraft.C_D0/Aircraft.k);     CD_min_drag = 2*Aircraft.C_D0
CL_min_power = sqrt(3*Aircraft.C_D0/Aircraft.k);   CD_min_power = 4*Aircraft.C_D0
CL_max_range = sqrt(Aircraft.C_D0/(3*Aircraft.k)); CD_max_range = (4/3)*Aircraft.C_D0

for i in range (0, Aircraft.steps):
    # Assume at zero pressure altitude
    x = i/(Aircraft.steps*0.5) # such that CL range [0-2]
    CL [i] = x
    CD [i] = Aircraft.Zero_Lift_Drag(Aircraft.M_from_CL(CL[i])) + Aircraft.k * ((CL [i])**2)


# Item 2 Drag variation at CL_max  [C_D vs M]

# Note: should be constant until the drag divergence kicks in
# Assume varibale height, implying pressure/density variability
# in order to maintain fixed Cl at Clmax
for i in range (0, Aircraft.steps):
    M = i/Aircraft.steps
    M_at_CL_max [i] = M   # assume constant all through
    CD_at_CL_max [i] = Aircraft.Zero_Lift_Drag(M) + Aircraft.k * Aircraft.C_Lmax**2


# Velocity Profile
# Item 3
#                                Speed of Sound
for i in range (0, len(Aircraft.h)):
    Speed_of_Sound[i] = Aircraft.conditions.speed_of_sound[i]
    V_M48[i] = (Aircraft.conditions.speed_of_sound[i])*0.48

# Item 4
#                                V_stall
for i in range (0, len(Aircraft.h)):
    V_stall[i] = sqrt(2*Aircraft.W/((Aircraft.conditions.density[i])*Aircraft.C_Lmax*Aircraft.S))

# Item 5
#                                V_min
for i in range (0, len(Aircraft.h)):
    a = sqrt (Aircraft.k/Aircraft.C_D0) # reciprocal of CL
    V_maxR[i] = sqrt(2*a*Aircraft.W/((Aircraft.conditions.density[i])*Aircraft.S))
#________________________________________________________#


# Thrust Profile
# from V stall to variation, we input (m, alt, to get power)
# True speed vs Thrust variation such that out
# True Speed vs Altitude
# Required Thrust at 30,000 ft

# Item 6   Overly Glitching and plot unlike anticipated
if False:
    # Required Thrust at 15,000 ft with variation in velocity
    # Required Thrust at 30,000 ft with variation in velocity
    Required_Thrust1 = np.linspace (0,1,2000)
    Required_Thrust2 = np.linspace (0,1,2000)
    Required_Thrust_Speed1 = np.linspace (0,1,2000)
    Required_Thrust_Speed2 = np.linspace (0,1,2000)
    density1 = 0.770488088287476 # air density in kg/m^3
    density2 = 0.459729898832652 # air density in kg/m^3
    a1 = 322.2 # speed of sound in m/s
    a2 = 303.1 # speed of sound in m/s
    for i in range (1, 2000+1): # 100 points will be start from one to avoid dividing by zero.
        Required_Thrust_Speed1  [i-1]= i
        Required_Thrust1        [i-1]= 0.5*density1*Aircraft.S*Aircraft.C_D0*(pow(i,2)) + (pow (i,-2))*Aircraft.k*(Aircraft.W**2)/ (0.5*density1*Aircraft.S)
        Required_Thrust_Speed2  [i-1]= i
        Required_Thrust2        [i-1]= 0.5*density2*Aircraft.S*Aircraft.C_D0*(pow(i,2)) + (pow (i,-2))*Aircraft.k*(Aircraft.W**2)/ (0.5*density2*Aircraft.S)
```

```python
        #print (0.5*density2*Aircraft.S*Aircraft.C_D0, Aircraft.k*(Aircraft.W**2)/ (0.5*density2*Aircraft.S))

    fig, (tax, tax0) = plt.subplots(nrows=2)
    tax.plot(Required_Thrust_Speed1, Required_Thrust1*(0.000001), color='blue')
    tax.set_xlabel('True Velocity (m/s)')
    tax.set_ylabel('Thrust (10e6 N)')
    tax.set_title ('at 15,000 ft elevation')
    tax0.plot(Required_Thrust_Speed2, Required_Thrust2*(0.000001), color='blue')
    tax0.set_xlabel('True Velocity')
    tax0.set_ylabel('Thrust (10e6)')
    tax0.set_title ('at 30,000 ft elevation')
    plt.show()

# Item 7 approx ceiling using anderson's equation
if False:
    Approx_ceiling_AEO_JT9D = Aircraft.ceiling(P_max_JT9D*2)
    Approx_ceiling_AEO_4056 = Aircraft.ceiling(P_max_4056*2)

    Approx_ceiling_OEI_JT9D = Aircraft.ceiling(P_max_JT9D)
    Approx_ceiling_OEI_4056 = Aircraft.ceiling(P_max_4056)

    print (Approx_ceiling_AEO_JT9D, Approx_ceiling_AEO_4056, Approx_ceiling_OEI_JT9D, Approx_ceiling_OEI_4056)


# Item 8 thrust Available at 0 ft, 15 kft, 30 kft, 45 kft for all 4 configuration
if True:
    thrust_AEO_JT9D_00, mach_AEO_JT9D_00, mmax_AEO_JT9D_00, hvmax_AEO_JT9D_00 = Aircraft.Thrust(00000,'Profile1', 'AEO')
    thrust_AEO_JT9D_15, mach_AEO_JT9D_15, mmax_AEO_JT9D_15, hvmax_AEO_JT9D_15 = Aircraft.Thrust(15000,'Profile1', 'AEO')
    thrust_AEO_JT9D_30, mach_AEO_JT9D_30, mmax_AEO_JT9D_30, hvmax_AEO_JT9D_30 = Aircraft.Thrust(30000,'Profile1', 'AEO')
    thrust_AEO_JT9D_45, mach_AEO_JT9D_45, mmax_AEO_JT9D_45, hvmax_AEO_JT9D_45 = Aircraft.Thrust(45000,'Profile1', 'AEO')

    thrust_AEO_4056_00, mach_AEO_4056_00, mmax_AEO_4056_00, hvmax_AEO_4056_00 = Aircraft.Thrust(00000,'Profile2', 'AEO')
    thrust_AEO_4056_15, mach_AEO_4056_15, mmax_AEO_4056_15, hvmax_AEO_4056_15 = Aircraft.Thrust(15000,'Profile2', 'AEO')
    thrust_AEO_4056_30, mach_AEO_4056_30, mmax_AEO_4056_30, hvmax_AEO_4056_30 = Aircraft.Thrust(30000,'Profile2', 'AEO')
    thrust_AEO_4056_45, mach_AEO_4056_45, mmax_AEO_4056_45, hvmax_AEO_4056_45 = Aircraft.Thrust(45000,'Profile2', 'AEO')

    thrust_OEI_JT9D_00, mach_OEI_JT9D_00, mmax_OEI_JT9D_00, hvmax_OEI_JT9D_00 = Aircraft.Thrust(00000,'Profile1', 'OEI')
    thrust_OEI_JT9D_15, mach_OEI_JT9D_15, mmax_OEI_JT9D_15, hvmax_OEI_JT9D_15 = Aircraft.Thrust(15000,'Profile1', 'OEI')
    thrust_OEI_JT9D_30, mach_OEI_JT9D_30, mmax_OEI_JT9D_30, hvmax_OEI_JT9D_30 = Aircraft.Thrust(30000,'Profile1', 'OEI')
    thrust_OEI_JT9D_45, mach_OEI_JT9D_45, mmax_OEI_JT9D_45, hvmax_OEI_JT9D_45 = Aircraft.Thrust(45000,'Profile1', 'OEI')

    thrust_OEI_4056_00, mach_OEI_4056_00, mmax_OEI_4056_00, hvmax_OEI_4056_00 = Aircraft.Thrust(00000,'Profile2', 'OEI')
    thrust_OEI_4056_15, mach_OEI_4056_15, mmax_OEI_4056_15, hvmax_OEI_4056_15 = Aircraft.Thrust(15000,'Profile2', 'OEI')
    thrust_OEI_4056_30, mach_OEI_4056_30, mmax_OEI_4056_30, hvmax_OEI_4056_30 = Aircraft.Thrust(30000,'Profile2', 'OEI')
    thrust_OEI_4056_45, mach_OEI_4056_45, mmax_OEI_4056_45, hvmax_OEI_4056_45 = Aircraft.Thrust(45000,'Profile2', 'OEI')
print ('---------------------')
print ('JT9D')
print (mmax_AEO_JT9D_00, hvmax_AEO_JT9D_00)
print (mmax_AEO_JT9D_15, hvmax_AEO_JT9D_15)
print (mmax_AEO_JT9D_30, hvmax_AEO_JT9D_30)
print (mmax_AEO_JT9D_45, hvmax_AEO_JT9D_45)
print ('---------------------')
print (mmax_OEI_JT9D_00, hvmax_OEI_JT9D_00)
print (mmax_OEI_JT9D_15, hvmax_OEI_JT9D_15)
print (mmax_OEI_JT9D_30, hvmax_OEI_JT9D_30)
print (mmax_OEI_JT9D_45, hvmax_OEI_JT9D_45)
print ('---------------------')
print ('4056')
print (mmax_AEO_4056_00, hvmax_AEO_4056_00)
print (mmax_AEO_4056_15, hvmax_AEO_4056_15)
print (mmax_AEO_4056_30, hvmax_AEO_4056_30)
print (mmax_AEO_4056_45, hvmax_AEO_4056_45)
print ('---------------------')
print (mmax_OEI_4056_00, hvmax_OEI_4056_00)
print (mmax_OEI_4056_15, hvmax_OEI_4056_15)
print (mmax_OEI_4056_30, hvmax_OEI_4056_30)
print (mmax_OEI_4056_45, hvmax_OEI_4056_45)
print ('---------------------')

fig, (lax,lax0,lax1,lax2) = plt.subplots(nrows=4)
lax.plot(mach_AEO_JT9D_00, thrust_AEO_JT9D_00*Aircraft.lb_to_N*2, color='green')
lax.plot(mach_AEO_JT9D_15, thrust_AEO_JT9D_15*Aircraft.lb_to_N*2, color='green')
lax.plot(mach_AEO_JT9D_30, thrust_AEO_JT9D_30*Aircraft.lb_to_N*2, color='green')
lax.plot(mach_AEO_JT9D_45, thrust_AEO_JT9D_45*Aircraft.lb_to_N*2, color='green')
lax.set_ylabel ('JT9D Thrust (N)')
lax.set_title ('AEO')

lax0.plot(mach_OEI_JT9D_00, thrust_OEI_JT9D_00*Aircraft.lb_to_N, color='green')
lax0.plot(mach_OEI_JT9D_15, thrust_OEI_JT9D_15*Aircraft.lb_to_N, color='green')
lax0.plot(mach_OEI_JT9D_30, thrust_OEI_JT9D_30*Aircraft.lb_to_N, color='green')
lax0.plot(mach_OEI_JT9D_45, thrust_OEI_JT9D_45*Aircraft.lb_to_N, color='green')
lax0.set_title ('OEI')

lax1.plot(mach_AEO_4056_00,thrust_AEO_4056_00*Aircraft.lb_to_N*2, color='green')
lax1.plot(mach_AEO_4056_00,thrust_AEO_4056_00*Aircraft.lb_to_N*2, color='green')
lax1.plot(mach_AEO_4056_00,thrust_AEO_4056_00*Aircraft.lb_to_N*2, color='green')
lax1.plot(mach_AEO_4056_00,thrust_AEO_4056_00*Aircraft.lb_to_N*2, color='green')
lax1.set_ylabel ('4056 Thrust (N)')
lax1.set_title ('AEO')

lax2.plot(mach_OEI_4056_00,thrust_OEI_4056_00*Aircraft.lb_to_N, color='green')
lax2.plot(mach_OEI_4056_00,thrust_OEI_4056_00*Aircraft.lb_to_N, color='green')
lax2.plot(mach_OEI_4056_00,thrust_OEI_4056_00*Aircraft.lb_to_N, color='green')
lax2.plot(mach_OEI_4056_00,thrust_OEI_4056_00*Aircraft.lb_to_N, color='green')
lax2.set_xlabel ('Mach Number')
lax2.set_title ('OEI')
plt.show()

#____________________________________________________#


if True:
    # Plotting
    # Flight Envelope (Fully Operative)
```

```python
fig, (ax, ax0, ax1) = plt.subplots(nrows=3)
# Speed of Sound (broken into subset till mach 1)
ax.plot(Speed_of_Sound*1.0, Aircraft.h, label='0.1-1 Speed of Sound', color='gray')
ax.plot(Speed_of_Sound*0.9, Aircraft.h, color='gray')
ax.plot(Speed_of_Sound*0.8, Aircraft.h, color='gray')
ax.plot(Speed_of_Sound*0.7, Aircraft.h, color='gray')
ax.plot(Speed_of_Sound*0.6, Aircraft.h, color='gray')
ax.plot(Speed_of_Sound*0.5, Aircraft.h, color='gray')
ax.plot(Speed_of_Sound*0.4, Aircraft.h, color='gray')
ax.plot(Speed_of_Sound*0.3, Aircraft.h, color='gray')
ax.plot(Speed_of_Sound*0.2, Aircraft.h, color='gray')
ax.plot(Speed_of_Sound*0.1, Aircraft.h, color='gray')

# V M0.48
ax.plot(V_M48, Aircraft.h, label='0.48 Mach Velocity')
# V stall
ax.plot(V_stall, Aircraft.h, label='Stall Velocity')
# V min
ax.plot(V_maxR, Aircraft.h, label='Maximum Range Velocity')
ax.plot(V_maxR, Aircraft.h, label='Maximum Range Velocity')

ax.set_xlabel('Airspeed (m/s)')  # Add an x-label to the axes.
ax.set_ylabel('Elevation (m)')  # Add a y-label to the axes.
#ax.set_title("Flight Envelope")  # Add a title to the axes.
ax.legend()  # Add a legend
# ax.grid(linewidth=1)
ax.axis([0, 400, 0, Aircraft.h_max])

# Flight Envelope (Partially Operative)
# Speed of Sound (broken into subset till mach 1)
ax0.plot(Speed_of_Sound*1.0, Aircraft.h, label='0.1-1 Speed of Sound', color='gray')
ax0.plot(Speed_of_Sound*0.9, Aircraft.h, color='gray')
ax0.plot(Speed_of_Sound*0.8, Aircraft.h, color='gray')
ax0.plot(Speed_of_Sound*0.7, Aircraft.h, color='gray')
ax0.plot(Speed_of_Sound*0.6, Aircraft.h, color='gray')
ax0.plot(Speed_of_Sound*0.5, Aircraft.h, color='gray')
ax0.plot(Speed_of_Sound*0.4, Aircraft.h, color='gray')
ax0.plot(Speed_of_Sound*0.3, Aircraft.h, color='gray')
ax0.plot(Speed_of_Sound*0.2, Aircraft.h, color='gray')
ax0.plot(Speed_of_Sound*0.1, Aircraft.h, color='gray')

# V M0.48
ax0.plot(V_M48, Aircraft.h, label='0.48 Mach Velocity')
# V stall
ax0.plot(V_stall, Aircraft.h, label='Stall Velocity') # independent of thrust
# V maxR
ax0.plot(V_maxR, Aircraft.h, label='Maximum Range Velocity')
ax0.axis([0, 400, 0, Aircraft.h_max])


ax0.set_xlabel('Airspeed (m/s)')  # Add an x-label to the axes.
ax0.set_ylabel('Elevation (m)')  # Add a y-label to the axes.
#ax0.set_title("Flight Envelope")  # Add a title to the axes.
ax0.legend()  # Add a legend
# ax.grid(linewidth=1)
ax0.axis([0, 400, 0, Aircraft.h_max])

# Plotting of Drag Profile and stats
ax1.plot(CD, CL)
ax1.plot(CD_min_power, CL_min_power, 'ro')
ax1.plot(CD_min_drag, CL_min_drag,   'ro')
ax1.plot(CD_max_range,  CL_max_range,  'ro')
ax1.set_xlabel('Cd')  # Add an x-label to the axes.
ax1.set_ylabel('Cl')  # Add a y-label to the axes.
ax1.axis([0, 0.2, 0, 2])
plt.show()
```