```
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import numpy as np
import math

#=========================================
# comFunction　：　共通関数クラス
#=========================================

class comFunction:
    #
    #  A : 動的エネルギー計算(je)
    #  <Param>
    #    m ：質量(kg)
    #    v ：速度(km/h)
    #
    def A(self,m,v):
        return m*v**2                   # A=mv^2

    #
    #  v　：　動的エネルギー>速度　換算(km/h)
    #  <Param>
    #    A ：動的エネルギー(je)
    #    m ：質量(kg)
    #
    def v(self,A,m):
        return math.sqrt(A/m)           # v=sqrt(A/m)

    #
    #  S　：　静的エネルギー計算(je)
    #  <Param>
    #    M ：中心天体質量(kg)
    #    m ：周回天体質量(kg)
    #    r ：２天体間距離(km)
    #    n　:1:通常静的エネルギー　2:脱出静的エネルギー
    #
    def S(self,M,m,r,n):
        μ = self.μ(M,m)                 # Em = mp x c^2
        Sr = n * μ / r            # Sr = Em x ac / r
        return Sr

    #
    #  μ　：　重力定数
```

```
#  <Param>
#    M ：中心天体質量(kg)
#    m ：周回天体質量(kg)
#
def μ(self,M,m):
    Em = self.Em(m)                # Em = mp x c^2
    ac = self.a(M,m)               # ac = U x (m + mp)
    μ =  Em * ac                   # μ = Em x ac
    return μ

#
#  Em ：　質量エネルギー(je)
#  <Param>
#    M ：質量(kg)
#
def Em(self,m):
    cc = comConst()
    Em = m * cc.c**2               # Em = mp x c^2
    return Em



#
#  a ：　光速時基準軌道半径(km)
#  <Param>
#    M ：質量(kg)
#    m ：周回天体質量(kg)
#
def a(self,M,m):
    cc = comConst()
    ac = cc.U * (M + m)            # ac = U x (m + mp)
    return ac                      # Em = mp x c^2

#
#  hToS ：　時間 を　秒に換算
#  <Param>
#    h ：時間(h)
#
def hToS(self,h):
    return h/3600

#
#  radToDeg ：　radian を　度に換算
#  <Param>
```

```
    #    rad ：時間(h)
    #
    def radToDeg(self,rad):
        return rad * 180 /math.pi

#=========================================
# moon   :   月クラス
#=========================================

class moon:

    M  = 5.97219E+24          # 地球質量 (kg)
    m  = 7.34581e22           # 月質量 (kg)

    a0 = 356400               # 原始基準軌道 (km)

    f1 = 4200                 # 第１衝突後単振動振幅
    a1 = a0 + f1              # 第１衝突後基準軌道半径

    i2 = 2800                 # 第２衝突位置（第１衝突後基準軌道半径からの位置）
    f2 = 21000                # 第２衝突後単振動振幅
    a2 = a1 + i2 + f2         # 第２衝突後基準軌道半径

    f_max =40000
    rto = 3/(a2+f2)

    T0 = 584.68               #原始　公転時間(h)
    T1 = 595.04               #第１衝突後　公転時間(h)
    T2 = 654.91               #第２衝突後　公転時間(h)



    def listData(self):
        print("")
        print("-----------------------------------")
        print("＜月データ＞")
        print("-----------------------------------")
        print("")

        print("地球質量                M  = %.5e kg"%(self.M))
        print("月質量              m  = %.5e kg"%(self.m))
        print("")
        print("##軌道半径")
```

```
        print("原始軌道半径          a0  = %.5e km"%(self.a0))
        print("第１衝突後軌道半径    a1  = %.5e km"%(self.a1))
        print("第２衝突後軌道半径    a2  = %.5e km"%(self.a2))
        print("")
        print("")

    def viewText(self,ax,line):
        ax.text(0.5,line,'Mass of Earth M = '+str(self.M)+" kg")
        ax.text(0.5,line-0.2,'Mass of moon  m = '+str(self.m)+" kg")

        ax.text(0.5,line-0.6,'Origin orbit')
        ax.text(0.5,line-0.8,'Semi Axis     a0 = '+str(self.a0)+" km")

        ax.text(0.5,line-1.2,'1.st Impact orbit')
        ax.text(0.5,line-1.4,'Semi Axis     a1 = '+str(self.a1)+" km")
        ax.text(0.5,line-1.6,'Amplitude     f1 = '+str(self.f1)+" km")

        ax.text(0.5,line-2.0,'2.nd Impact orbit')
        ax.text(0.5,line-2.2,'Semi Axis     a2 = '+str(self.a2)+" km")
        ax.text(0.5,line-2.4,'Amplitude     f2 = '+str(self.f2)+" km")
        ax.text(0.5,line-2.6,'Impact Point  i2 = '+str(self.i2)+" km")




#=========================================
# moonOrbit    :    月軌道描画クラス
#=========================================

class orbitView:

    #
    # setPlot：描画プロット
    #
    def setPlot(self):
        fig = plt.figure(figsize=(10,10),
                 dpi=200,
                facecolor='lightgrey',
                edgecolor='black',
                linewidth=2,
                frameon=True,
```

```
            tight_layout=False)

    ax = fig.add_subplot(111)
    ax.set_xlim([0,10])
    ax.set_ylim([0,10])
    ax.set_xticks([0,1,2,3,4,5,6,7,8,9,10])
    ax.set_yticks([0,1,2,3,4,5,6,7,8,9,10])
    return ax

#
# centralBody：中心天体　描画
#
def centralBody(self):
    c1 = mpatches.Circle(xy=(5,5),
                radius=0.1,
                edgecolor='Yellow',
                facecolor='green',
                fill=True)
    return c1



#
# virtualOrbit：仮想軌道(楕円)　描画
#
#  <Param>
#    pex ：x軸
#    pey ：y軸
#    f　：単振動振幅（焦点距離）
#
def virtualOrbit(self,pex,pey,f,color):

    e1 = mpatches.Ellipse(xy=(5+f,5),
                width     = pex*2,
                height    = pey*2,
                angle     = 0,
                edgecolor = color,
                facecolor = 'white',
                linewidth = 0.5,
                fill      = False)
    return e1

#
# baseOrbit：基準軌道　描画
```

```
    #
    #  <Param>
    #    pax ：x軸
    #    pay ：y軸
    #
    def baseOrbit(self,pax,pay,color):

        bo = mpatches.Circle(xy=(5,5),
                    radius=pax,
                    edgecolor=color,
                    linestyle='dashed',
                    label='Base Orbit',
                    linewidth=0.5,
                    fill=False)
        return bo

    #
    # view：軌道　描画
    #
    #  <Param>
    #    ax ：サブプロット
    #    e1 ：楕円軌道　描画
    #    bo ：基準軌道　描画
    #
    def view(self,ax,e1,bo):
        ax.grid(linestyle='-')
        ax.add_patch(e1)
        ax.add_patch(bo)

    def viewMoonOrbit(self):
        mn = moon()
        mo = moonOrbit()
        #月軌道　表示
        mn.listData()
        ax = self.setPlot()
        mn.viewText(ax,9.8)
        c1 =ob.centralBody()
        ax.add_patch(c1)
        mo.convPlotData(ax,mn.a1,-mn.f1) # 1.st Impact
        mo.convPlotData(ax,mn.a2,mn.f2) # 2.nd Impact
        plt.show()

#=========================================
```

```
# moonOrbit　：　月軌道クラス
#=========================================

class moonOrbit:

    cf = comFunction()
    ob = orbitView()
    mn = moon()

    def b(self,a,f):
        return math.sqrt(a**2 - f**2)

    def convPlotData(self,ax,a,f):
        rto = mn.rto
        b   = self.b(a,f)
        pax = rto * a
        pay = rto * a
        pex = rto * a
        pey = rto * b
        f   = rto * f

        e1  = ob.virtualOrbit(pex,pey,f,'blue')
        bo  = ob.baseOrbit(pax,pay,'red')
        ob.view(ax,e1,bo)

#=========================================
# geocentricCoordinatesView　：　地心座標系クラス
#=========================================

class geocentricCoordinatesView:

    #
    # setPlot：描画プロット
    #
    def setPlot(self):
        fig = plt.figure(figsize=(10,10),
                 dpi=200,
                facecolor='lightgrey',
                edgecolor='black',
                linewidth=2,
                frameon=True,
                tight_layout=False)
```

```
    ax = fig.add_subplot(111)
    ax.set_xlim([0,10])
    ax.set_ylim([0,10])
    ax.set_xticks([0,1,2,3,4,5,6,7,8,9,10])
    ax.set_yticks([0,1,2,3,4,5,6,7,8,9,10])

    return ax

#
# impact1Orbit：第１衝突　地心距離　描画
#

def impact1Orbit(self,ax):
    mn = moon()
    x1 = np.arange(0,100,0.01)
    rto = 3/mn.f_max

    pf = rto * mn.f1
    pa = rto * mn.a1

    y1 = pf*np.cos(x1*10)+5

    ax.plot(x1,y1)

#
# impact2Orbit：第２衝突　地心距離　描画
#

def impact2Orbit(self,ax):
    mn = moon()
    x1 = np.arange(0,100,0.01)
    rto = 3/mn.f_max

    pa0 = rto * mn.a0

    pf1 = rto * mn.f1
    pa1 = rto * mn.a1

    pf2 = rto * mn.f2
    pa2 = rto * mn.a2
    pi2 = rto * mn.i2

    β = (mn.T2/mn.T0-1)
```

```
        y1 = (pf1*np.cos(x1*10)-pi2)*np.cos(β*x1*10)+pf2*np.cos(x1*10)+pa2-pa0+5

        ax.plot(x1,y1)


    #
    # view：地心距離　描画
    #
    #
    def view(self):
        print("")
        print("----------------------")
        print("  地心距離")
        print("----------------------")
        print("")

        mn=moon()
        ax=self.setPlot()
        ax.grid(linestyle='-') #グリッド表示
        x1 = np.arange(0,100,0.01)
        mn.viewText(ax,4.5)
        self.impact1Orbit(ax)  #1回目衝突
        self.impact2Orbit(ax)  #2回目衝突

        plt.show()

#==================================
# Main Routine
#==================================

mn = moon()
mo = moonOrbit()
ob = orbitView()
gc = geocentricCoordinatesView()

#月軌道　表示
ob.viewMoonOrbit()


#地心距離　表示
```

gc.view()