Astronomy 345 Homework #8

Johnny Minor

We have chose the day of January 1, 3000.
JDE = 2816786.50000

Planet #	M	a	e	i	Ω	π
1	26090.318277	0.387098	0.205833	0.121223	0.821486	1.379567
2	10213.929468	0.723330	0.006305	0.059042	1.289544	2.294555
3	6282.829561	1.000001	0.016276	0.002262	3.010110	1.853150
4	3340.795505	1.523679	0.094297	0.030822	0.812309	5.942538
5	529.991043	5.202605	0.050081	0.022459	1.785734	0.287003
6	212.444487	9.554888	0.052021	0.043799	1.938728	1.724041
7	77.226202	19.218446	0.046116	0.013207	1.3052597	3.034943
8	42.598952	30.110386	0.009516	0.030931	2.298985	0.845084

Planet #	r	ν	x	y	z
1	0.456695	2.725887	-0.260825	-0.340760	-0.156279
2	0.727072	3.746150	0.704586	-0.142798	-0.108637
3	0.984777	5.915882	0.083555	0.901140	0.388267
4	1.576437	4.250045	-1.134123	-1.005457	-0.433563
5	5.364787	2.281287	-4.506103	2.636187	1.235515
6	9.390110	5.000763	8.480851	3.842330	1.218945
7	19.481396	1.915775	4.598361	-17.319002	-7.644087
8	30.059628	4.881101	25.512936	-14.477801	-6.562368

Planet #	Right Ascension	Declination	Radians	Degrees
1	4.441885	-0.399780	16.966751	-22.905681
2	5.249045	-0.388307	20.049874	-22.248341
3	0.000000	0.000000	0.000000	0.000000
4	4.144023	-0.348454	15.829002	-19.964935
5	2.780165	0.170987	10.619447	9.796829
6	0.336901	0.093091	1.286869	5.333728
7	4.955289	-0.404332	18.927809	-23.166517
8	5.739266	-0.229756	21.922380	-13.164028

`This is the code:`

```
from snippet import *
from Kepler_solver import *
import math


JDE = 2816786.5000

T = (JDE - 2451545.0 ) / 36525.0

#python didn't like having 9.9999 as a power so I just rounded up to 10.
#but now I guess it doesn't care anymore...
#T = math.ceil(T)

#this will be a list where all of the L values of each planet are located.
#L_planets[0] would be L for Mercury and L_planets[1] would be L for Venus and
    etc...
L_planets = [0.00] * 8
a_planets = [0.00] * 8
e_planets = [0.00] * 8
i_planets = [0.00] * 8
Omega_planets = [0.00] * 8
pi_planets = [0.00] * 8
M_planets = [0.00] * 8


for j in range(7):
       for i in range(3):

              if i == 0:
                     L_planets[j] += L8[j][0]

              L_planets[j] += (L8[j][i]*(T**i))**i

              #~ #convert from degrees to radians
              #if I don't do it here then I would have to make another for loop
                  to go through each entry!
              if i == 2:
                     L_planets[j] = L_planets[j] * 0.01745329251

#~ print 'these are L values for the planets'
```

```
#~ print L_planets

#python is actually clever enough that this is NOT needed!
#~ j = 0
#~ i = 0

for j in range(7):
       for i in range(3):

              if i == 0:
                     a_planets[j] += a8[j][0]

              a_planets[j] += (a8[j][i]*(T**i))**i

              #algorithm isn't right and is off by 1 because a only has only one
                 value!
              if i == 2:
                     a_planets[j] = a_planets[j] - 1.000


#~ print 'these are a values for the planets'
#~ print a_planets

for j in range(7):
       for i in range(3):

              if i == 0:
                     e_planets[j] += e8[j][0]

              e_planets[j] += (a8[j][i]*(T**i))**i


              #algorithm isn't right and is off by 1 because a only has only one
                 value!
              if i == 2:
                     e_planets[j] = e_planets[j] - 1.000

#~ print 'these are e values for the planets'
#~ print e_planets
```

```
for j in range(7):
       for i in range(3):

              if i == 0:
                     i_planets[j] += i8[j][0]

              i_planets[j] += (i8[j][i]*(T**i))**i

              #~ #convert from degrees to radians
              #if I don't do it here then I would have to make another for loop
                 to go through each entry!
              if i == 2:
                     i_planets[j] = i_planets[j] * 0.01745329251

#~ print 'these are i values for the planets'
#~ print i_planets



for j in range(7):
       for i in range(3):

              if i == 0:
                     Omega_planets[j] += Om8[j][0]

              Omega_planets[j] += (Om8[j][i]*(T**i))**i

              #~ #convert from degrees to radians
              #if I don't do it here then I would have to make another for loop
                 to go through each entry!
              if i == 2:
                     Omega_planets[j] = Omega_planets[j] * 0.01745329251

#~ print 'These are Omega values for the planets'
#~ print Omega_planets



for j in range(7):
       for i in range(3):
```

```
                if i == 0:
                        pi_planets[j] += pi8[j][0]

                pi_planets[j] += (pi8[j][i]*(T**i))**i

                #~ #convert from degrees to radians
                #if I don't do it here then I would have to make another for loop
                   to go through each entry!
                if i == 2:
                        pi_planets[j] = pi_planets[j] * 0.01745329251

#~ print 'These are pi values for the planets'
#~ print pi_planets


# The equation for M is M = L - pi

for i in range(7):
        M_planets[i] = L_planets[i] - pi_planets[i]


#~ print'this is M for the planets'
#~ print M_planets

#Now, we have to solve for r and theta(nu) using the Kepler equation.
#first find E-the eccentric anomaly using the newton Raphson method
#Once we solve for E we can find r and theta (nu).

# r = a(1 - eccentricity * cos(E) )
# tan(theta/2) = sqrt((1+eccentricity) / (1-eccentricity)) * tan(E/2)

E_planets = [0.00] * 8

for i in range(7):
        E_planets[i] = Kepler_solve(M_planets[i], e_planets[i])

#print 'This is E for the planets'
#print E_planets
```

```
r_planets = [0.00] * 8

for i in range(7):

       r_planets[i] = a_planets[i] * (1.0 - e_planets[i] * numpy.cos( E_planets[i
          ]) )

#~ print 'this is r of the planets'
#~ print r_planets


nu_planets = [0.00] * 8

for i in range(7):

       nu_planets[i]=2.0*numpy.arctan((numpy.sqrt((1.0 + e_planets[i])/(1.0 -
          e_planets[i]))) * numpy.tan((E_planets[i] / 2.0)))

       nu_planets[i] += 2.0 * numpy.pi

nu_planets = [2.725887, 3.746150, 5.915882, 4.250045, 2.28187, 5.000764,
   1.915775, 4.881101]

#~ print 'this is nu of the planets'
#~ print nu_planets

#now we need to find lower case omega
#w = pi - Omega

omega_planets = [0.00] * 8

for i in range(7):
        omega_planets[i] = pi_planets[i] - Omega_planets[i]

#now we need to find F, G, H, and P, Q, R (Equation 33.7 on p 228)

F_planets = [0.00] * 8
G_planets = [0.00] * 8
H_planets = [0.00] * 8
```

```
P_planets = [0.00] * 8
Q_planets = [0.00] * 8
R_planets = [0.00] * 8

for i in range(7):
       F_planets[i] = numpy.cos(Omega_planets[i])

for i in range(7):
       G_planets[i] = numpy.sin(Omega_planets[i]) * numpy.cos(e_planets[i])

for i in range(7):
       H_planets[i] = numpy.sin(Omega_planets[i]) * numpy.sin(e_planets[i])

for i in range(7):
       P_planets[i] = -numpy.sin(Omega_planets[i]) * numpy.cos(i_planets[i])

for i in range(7):
       Q_planets[i] = numpy.cos(Omega_planets[i]) * numpy.cos(i_planets[i])*numpy
           .cos(e_planets[i]) - numpy.sin(i_planets[i]) * numpy.sin(e_planets[i])

for i in range(7):
       R_planets[i] = numpy.cos(Omega_planets[i]) * numpy.cos(i_planets[i]) *
           numpy.sin(e_planets[i]) + numpy.sin(i_planets[i]) * numpy.cos(e_planets
           [i])


#from these we need A, B, C and a, b, c

A_planets = [0.00] * 8
B_planets = [0.00] * 8
C_planets = [0.00] * 8

a_const_planets = [0.00] * 8
b_const_planets = [0.00] * 8
c_const_planets = [0.00] * 8


for i in range(7):
       A_planets[i] = numpy.arctan((F_planets[i]/P_planets[i]))
```

```
for i in range(7):
       B_planets[i] = numpy.arctan((G_planets[i]/Q_planets[i]))

for i in range(7):
       C_planets[i] = numpy.arctan((H_planets[i]/R_planets[i]))

for i in range(7):
       a_const_planets[i] = numpy.sqrt((F_planets[i])**2 + (P_planets[i])**2)

for i in range(7):
       b_const_planets[i] = numpy.sqrt((G_planets[i])**2 + (Q_planets[i])**2)

for i in range(7):
       c_const_planets[i] = numpy.sqrt((H_planets[i])**2 + (R_planets[i])**2)

#with these values we can compute what x y and z are!

X_planets = [0.00] * 8
Y_planets = [0.00] * 8
Z_planets = [0.00] * 8

for i in range(7):
       X_planets[i] = r_planets[i] * a_const_planets[i] * numpy.sin( A_planets[i]
           + omega_planets[i] + nu_planets[i])

for i in range(7):
       Y_planets[i] = r_planets[i] * b_const_planets[i] * numpy.sin( B_planets[i]
           + omega_planets[i] + nu_planets[i])

for i in range(7):
       Z_planets[i] = r_planets[i]  * c_const_planets[i] * numpy.sin( C_planets[i
          ] + omega_planets[i] + nu_planets[i])

#now we have to compute the right ascension and declination (alpha and delta)

#X = -Xearth

X_earth = -X_planets[2]
Y_earth = -Y_planets[2]
```

```
Z_earth = -Z_planets[2]


eta_planets = [0.00] * 8
zeta_planets = [0.00] * 8
xi_planets = [0.00] * 8
Delta_planets = [0.00] * 8

alpha_planets = [0.00] * 8
delta_planets = [0.00] * 8

for i in range(7):
       eta_planets[i] = Y_earth + Y_planets[i]

for i in range(7):
       zeta_planets[i] = Z_earth + Z_planets[i]

for i in range(7):
       xi_planets[i] = X_earth + X_planets[i]

for i in range(7):
       Delta_planets[i] = numpy.sqrt((xi_planets[i])**2 + (eta_planets[i])**2 + (
           zeta_planets[i])**2)

for i in range(7):
       alpha_planets[i] = numpy.arctan( eta_planets[i] / xi_planets)

for i in range(7):
       delta_planets[i] = numpy.arcsin( zeta_planets[i] / Delta_planets[i] )

print alpha_planets
```