Studying Celestial Mechanics with Python Programming Language

February 22nd, 2018

Matthew Hyeun and Dr. B. Mitrovic

Goals and Overview

Goals

- Study Kepler's Laws of Planetary Motion
- Application
- Integrate Python 3 Programming

Overview

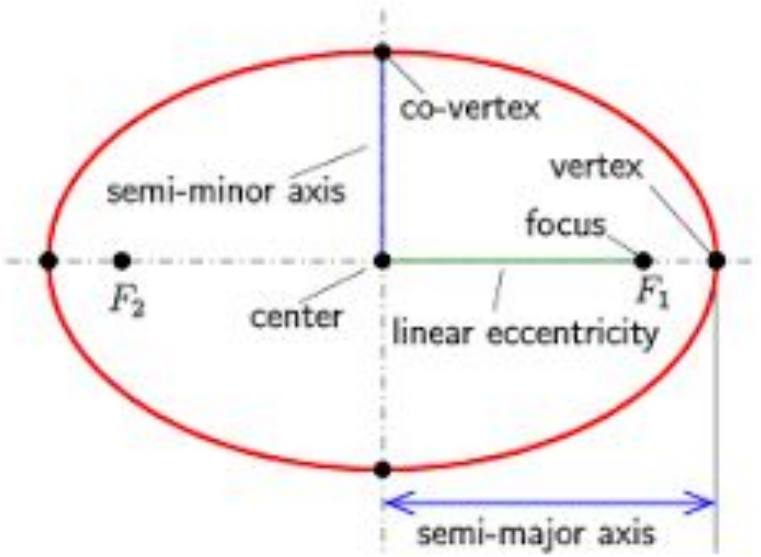
- Definitions
- Kepler's Laws/Newton's Laws
- Application of Laws
- Closing Remarks

Definitions

Important Definitions:

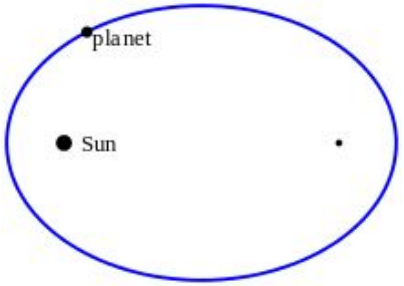
- Apoapsis/Aphelion
- Periapsis/Perihelion
- Eccentricity
  - e = 0
  - **0 < e < 1**
  - e = 1
  - e > 1
- Period
- Astronomical Unit
  - 1.496e+11 meters

Geometry of an Ellipse:



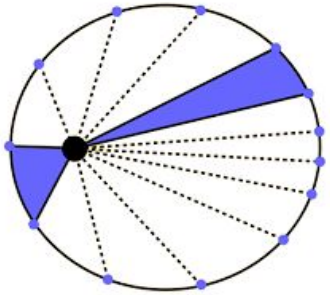
Kepler's Laws of Planetary Motion

1. *Law of Orbits:* All planets move in elliptical orbits, with the sun at one focus.



---

2. *Law of Areas*: A line that connects a planet to the sun sweeps out equal areas in equal times.



---

3. *Law of Periods*: The square of the period of any planet is proportional to the cube of the semimajor axis of its orbit.

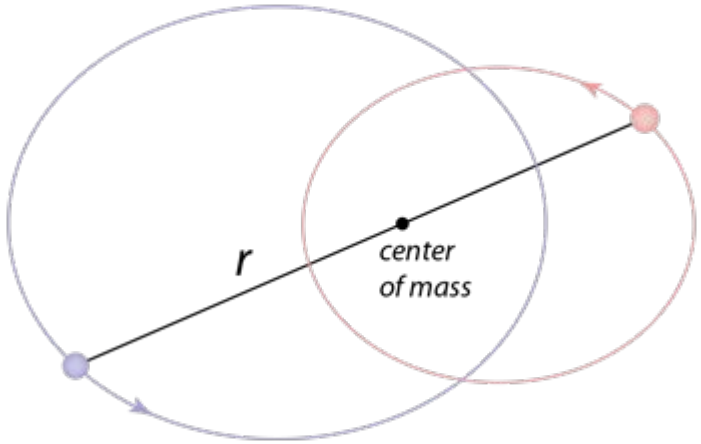
$$P^2 = a^3$$

Newtonian Versions of Kepler's Laws

However, there were issues with Kepler's Laws of Planetary Motion.

---

1. *Newton's Revised Version of the Law of Orbits* (generalized to binary): Two celestial masses orbiting in a bound binary orbit under the influence of gravity will follow elliptical orbits about the center of mass of the binary system.



Newtonian Versions of Kepler's Laws cont.

2. *Newton's Revised Version of the Law of Areas*:

The time rate of change of the area swept out by a line connecting a planet to the focus of an ellipse is a constant, one-half of the orbital angular momentum per unit mass.

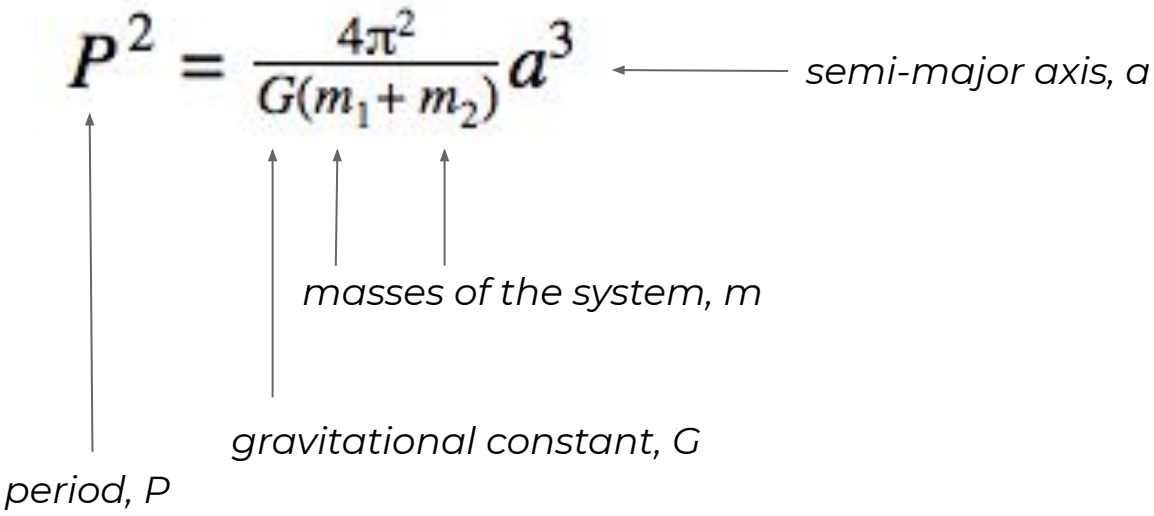
$$\frac{dA}{dt} = \frac{1}{2}\frac{L}{\mu}$$

$$L = \mu r v$$

Keywords: radial and angular components, angular momentum.

Newtonian Versions of Kepler's Laws cont.

3. Newton's Revised Version of the Law of Periods:



Keywords: area of an ellipse, geometry of an ellipse, Law of Orbits.

Application with Computer Science

Python 3 Programming Language

Applications with Computer Science

The user will be asked for orbital period, and orbital eccentricity.

Earth: P = 1yr, e = 0.0167. Halley's Comet: P = 75.32 yrs, e = 0.9673

---

1. What is the semimajor axis of any given orbit?

```
a = (Period ** 2) ** (1 / 3)
a = round(a, 4)

print("The semimajor axis of your orbit is", a, "AU (Astronimical Unit).\n")
```

Output for Earth: `The semimajor axis of your orbit is 1.0 AU (Astronimical Unit).`

Output for Halley's Comet: `The semimajor axis of your orbit is 17.8682 AU (Astronimical Unit).`

Applications with Computer Science cont.

2. Estimate the mass of the sun.

```
print("The semimajor axis must be converted into metres.")
ameters = a * 149597870700 #conversion factor for AU to m
print("Your semimajor axis is", "%.3e" % ameters, "meters. \n")


print("The period must be converted into seconds.")
periodseconds = Period * 31556926 #conversion factor from yrs to s
print("Your period is", "%.3e" % periodseconds, "seconds. \n")

print("The program will now estimate the mass of the sun, assuming that m2 is negligable. \n")

mSun = ( 4 * math.pi**2 * ameters**3 ) / ( G * periodseconds**2)

print("The estimated mass of the sun is", "%.3e" % mSun, "kilograms. \n")
```

Output for Earth: `The estimated mass of the sun is 1.990e+30 kilograms.`

Output for Halley's Comet: `The estimated mass of the sun is 1.990e+30 kilograms.`

Applications with Computer Science cont.

3. Calculate the distance from the Sun at perihelion and aphelion.

```
rp = a * ( 1 - Eccentricity ) #formula for perihelion
rp = round(rp, 4)

print("The distance from the Sun at perihelion is", rp, "AU. \n")


ra = a * ( 1 + Eccentricity ) #formula for aphelion
ra = round(ra, 4)

print("The distance from the Sun at aphelion is", ra, "AU. \n")
```

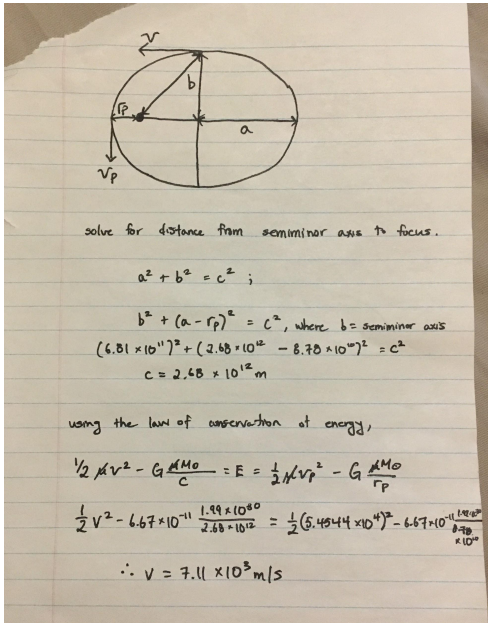
```
The distance from the Sun at perihelion is 0.9833 AU.

The distance from the Sun at aphelion is 1.0167 AU.
```

```
The distance from the Sun at perihelion is 0.5843 AU.

The distance from the Sun at aphelion is 35.1521 AU.
```

Applications with Computer Science cont.

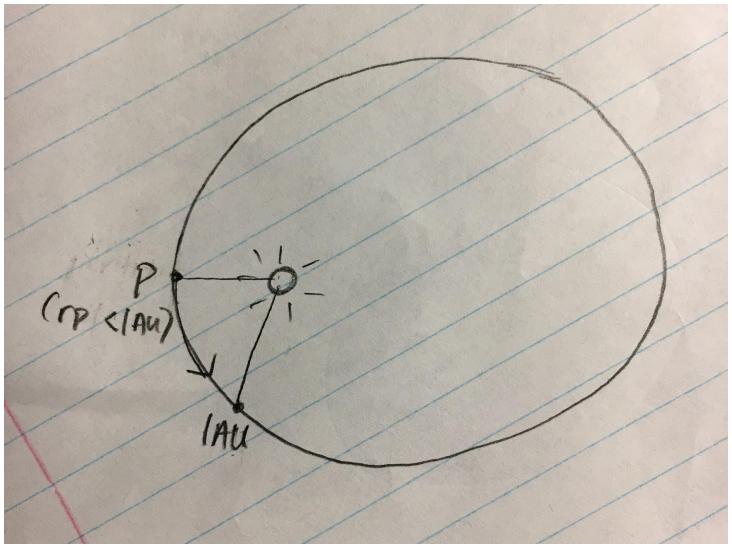
4. Calculate the orbital speed at perihelion, aphelion, and semi-minor axis.

```
vp = math.sqrt(((G * mSun) / ameters ) * ((1 + Eccentricity) / (1 - Eccentricity))) #velocity at perihelion
print("The orbital speed at perihelion is", "%.4e" % vp, "meters per second. \n")

va = math.sqrt(((G * mSun) / ameters ) * ((1 - Eccentricity) / (1 + Eccentricity))) #velocity at aphelion
print("The orbital speed at aphelion is", "%.4e" % va, "meters per second. \n")

  #semiminor axis

b = math.sqrt(( ameters ** 2) * ( 1 - (Eccentricity ** 2)))
print("The semiminor axis of the orbit is", "%.4e" % b, "meters.")

c =  math.sqrt(( b ** 2 ) + (( ameters - rp ) ** 2 ))
print("The distance from the semiminor axis to the sun is", "%.4e" % c, "meters.")

rpmeters = rp * 149597870700 #conversion factor from AU to m

vb = math.sqrt( 2 * ((( 1 / 2 ) * ( vp ** 2 )) - ( G * ( mSun / rpmeters )) + (G * ( mSun / c ))))

print("The orbital velocity at semiminor axis is", "%.4e" % vb, "meters per second. \n")
```



```
The orbital speed at perihelion is 3.0288e+04 meters per second.

The orbital speed at aphelion is 2.9293e+04 meters per second.

The semiminor axis of the orbit is 1.4958e+11 meters.
The distance from the semiminor axis to the sun is 2.1155e+11 meters.
The orbital velocity at semiminor axis is 1.9172e+04 meters per second.
```

```
The orbital speed at perihelion is 5.4655e+04 meters per second.

The orbital speed at aphelion is 9.0847e+02 meters per second.

The semiminor axis of the orbit is 6.7798e+11 meters.
The distance from the semiminor axis to the sun is 2.7577e+12 meters.
The orbital velocity at semiminor axis is 6.8305e+03 meters per second.
```



Main Question

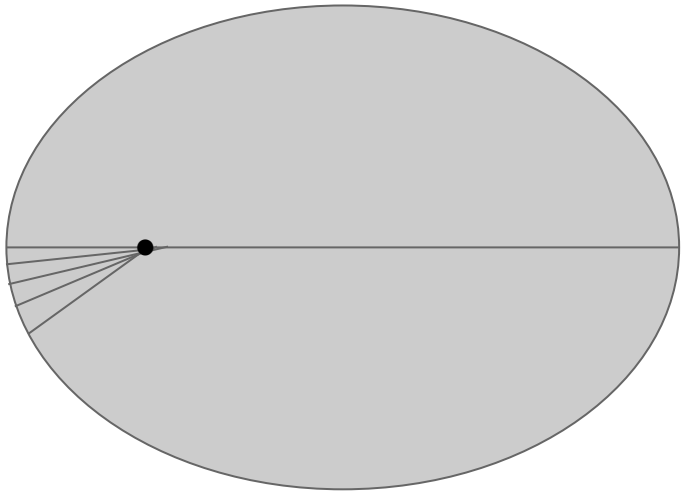
Estimate the amount of time required for a celestial object to move from perihelion to a distance of 1 AU away from the principal focus.

Approach

Calculate orbital period.

Divide orbital period into infinitesimally small parts i.e timesteps.

Calculate how many timesteps required to reach desired distance.



Code in Python Language

```
#Appendix G: A planetary orbit code
#Brock Mentorship
#by Matthew Hyeun

#Version 2.4 (no steps printed out)



import math
import matplotlib.pyplot as plt



#Constants

G = 6.67 * 10**-11 #Gravitational constant
Msun = 1.99 * 10**30 #Mass of sun
AU = 1.496 * 10**11 #AU conversion to meters
spyr = 3.154 * 10**7 #Seconds per year


#User inputs

Mstrsun = float(input("Enter mass of the parent star, in solar masses: "))
print( )

aau = float(input("Enter semimajor axis of the orbit, in AU: "))
print()

e = float(input("Enter the eccentricity of the orbit: "))
print()


#Orbital period calcuation

Pyears = math.sqrt(aau ** 3 / Mstrsun)
Pyears = round(Pyears, 2)
print("The orbital period is", Pyears, "years.")
print( )
```

Code in Python Language cont.

```
#User inputs for time steps and frequency of time steps

print("If too large of a time step is inputted, inaccurate results will be produced.")
print("If a high number of time steps is inputted, accurate results will be produced.")
print()

end = float(input("Enter the desired distance from the parent star for the calculation in AU: "))
print()

n = int(input("Enter the number of time steps for the calculation: "))
print()

kmax = 1


#Conversion to cgs units

period = Pyears * spyr
dt = period / (n - 1) #since first and last step is the same step
a = aau * AU
Mstar = Mstrsun * Msun


#State variables for loop

k = 0 #time step counter
theta = 0.00 #angle
time = 0.00 #self explanatory
```

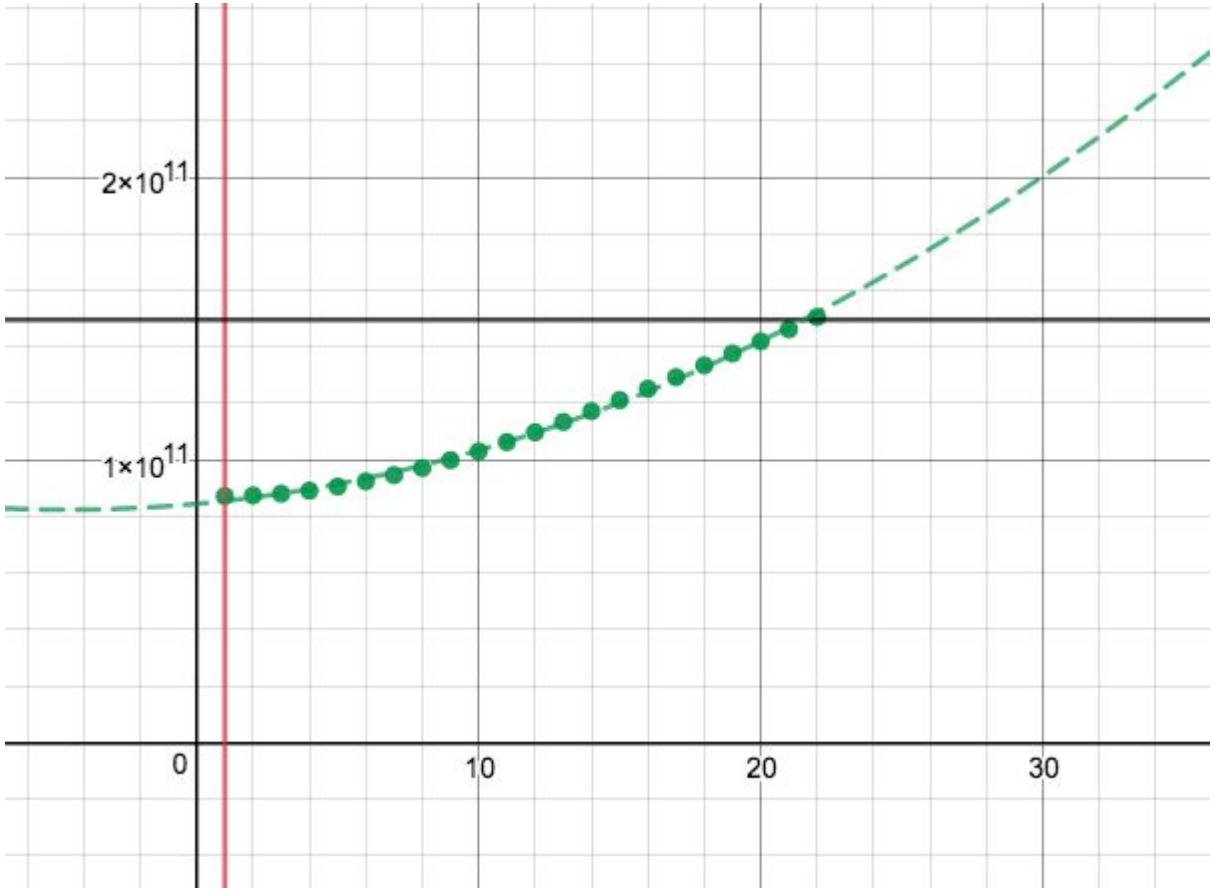
Code in Python Language cont.

```
#MAIN LOOP

data = []
data2 = []

for i in range(1, n):

    k = k + 1

    r = (a * (1 - e ** 2)) / (1 + e * math.cos(theta))

    if r >= (end * AU) or i == n:
        break

    #Calculate angluar momentum per unit mass L/m

    LoM = math.sqrt(G * Mstar * a * (1.00 - e ** 2))

    #Calculate next value of theta

    dtheta = LoM / r ** 2 * dt
    theta = theta + dtheta

    #Update elapsed time

    time = time + dt

    if k == kmax:
        k = 0

    x = r * math.cos(theta) / AU
    x = round(x, 2)


    y = r * math.sin(theta) / AU
    y = round(y, 2)

    data.append(x)
    data2.append(y)
```

Code in Python Language cont.

```
#CONCLUSIONS

print("The calculation is complete.")
print()

x = r * math.cos(theta) / AU
x = round(x, 2)


y = r * math.sin(theta) / AU
y = round(y, 2)


print("x is equal to", x, "AU.")
print("y is equal to", y, "AU.")
print()

tyears = time/spyr
tyears = round(tyears, 2)
print(tyears, "years have elapsed to reach a distance of", end, "AU from the Sun.")


#GRAPH
plt.xlabel('X-Value of Orbit (AU)')
plt.ylabel('Y-Value of Orbit (AU)')
plt.title('Orbit of Planet')


plt.plot(data, data2, "ro")
plt.show()
```

Visual Representation in Desmos



Perihelion is at x = 1.

$$y_1 \sim ax_1^2 + bx_1 + c$$

STATISTICS

$R^2 = 0.9983$

RESIDUALS

e_1 plot

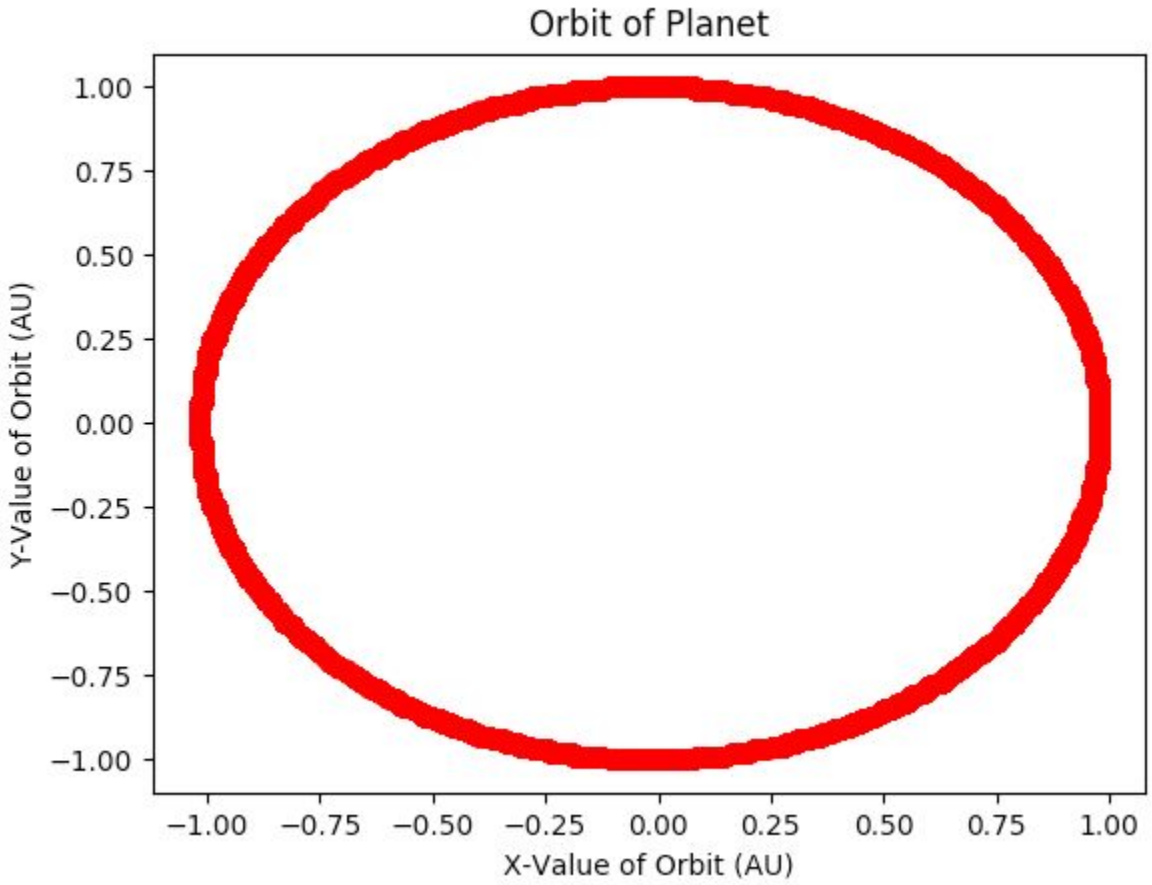
PARAMETERS

$a = 9.84683e + 7$ $b = 9.07976e + 8$

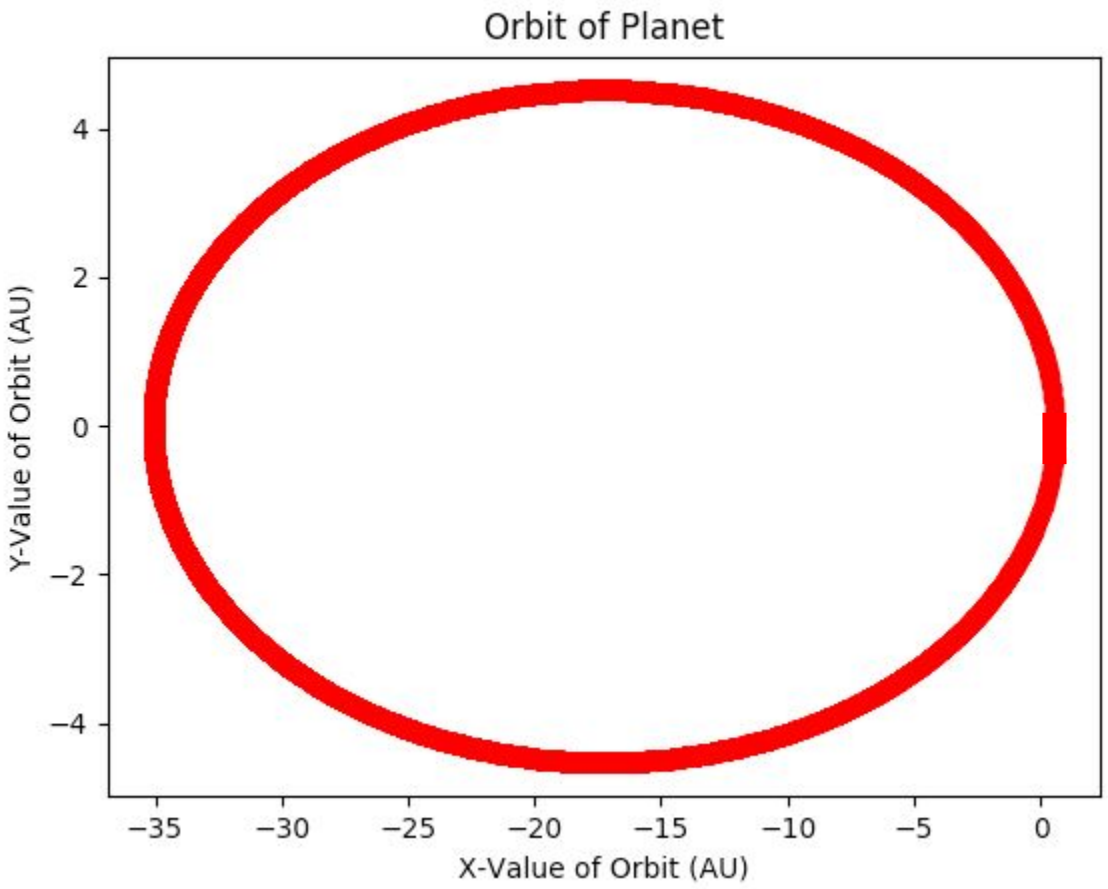
$c = 8.447 \times 10^{10}$

The line of best fit intersects with the line y = 1.496 x 10^11 when x = 21.518. Therefore, according to the line of best fit, Halley's Comet will be 1AU from the Sun after 39.27 days.

Visual Representation in Python Using matplotlib.pyplot



Visual Representation in Python Using matplotlib.pyplot



Works Cited

Bradly W. Carroll and Dale A. Ostlie, An Introduction to Modern Astrophysics, Addison-Wesley, Reading, Massachusetts (1996); Appendix G.

Closing Remarks

Thank you for listening to my presentation: ‘A Study of Celestial Mechanics with Python Programming’!

Thank you, **Mr. Peter Domarchuk**,
for being an outstanding co-op teacher.

Thank you, **Mrs. Heather Bellisario**,
for allowing many students to have an amazing opportunity.

Most importantly, thank you, **Dr. Bozidar Mitrovic**,
for sacrificing your valuable time and effort to be a mentor for my research.