In Class Activity

This will calculate the electrostatic force between two charges using Coulomb's Law:

$$F_E = \frac{kq_1q_2}{r^2}$$

Where $q_1 = 1.3 * 10^{-8}C$ and $q_2 = 4.3 * 10^{-8}C$ and $k = 8.99 * 10^9 \frac{Nm^2}{C^2}$ and r = 2 cm

In [11]:
```
k = 8.99e9           #N m^2 / C^2
q1 = 1.3e-8
q2 = 4.3e-8
r = 2e-2

#Calculate force between
F = k*q1*q2/r/r
print(f'The force is {F:.3e} Newtons')
```

```
The force is 1.256e-02 Newtons
```

Conclusion

This is a very straight forward program that calculates Coulomb Force. There were no difficulties.

Exercise 2.4 In Class Activity

Exercise 2.4: A spaceship travels from Earth in a straight line at relativistic speed v to another planet x light years away. Write a program to ask the user for the value of x and the speed v as a fraction of the speed of light c, then print out the time in years that the spaceship takes to reach its destination

(a) in the rest frame of an observer on Earth

(b) as perceived by a passenger on board the ship. Use your program to calculate the answers for a planet 10 light years away with v = 0.99c.

In [22]:
```
c = 3e8
d = 10*c*365*24*60*60
v = 0.99*c
γ = 1/np.sqrt(1-v**2/c**2)
#Observed on Earth
te = d/v
print(f'The observer on earth sees it take {te:.0f} seconds or {te/60/60/24/365:.3f}
#Observed on Ship
ts = te/γ
print(f'The observer on the ship sees it take {ts:.0f} seconds or {ts/60/60/24/365:.3
```

```
The observer on earth sees it take 318545455 seconds or 10.101 years
The observer on the ship sees it take 44936366 seconds or 1.425 years
```

Conclusion

I made the mistake of having the time on the ship be time on Earth times gamma to start. That made it so the time on the ship was longer than the time on Earth. I knew that was wrong, so changed it and found this to be the solution.

Exercises 2.1 & 2.2

Exercise 2.1: Another ball dropped from a tower

A ball is dropped from a tower of height h with initial velocity zero. Write a program that asks the user to enter the height in meters of the tower and then calculates and prints the time the ball takes until it hits the ground, ignoring air resistance. Use your program to calculate the time for a ball dropped from a 100 m high tower.

Introduction

This code will calculate the time it will take to have the ball hit the ground. It has the user input the height. The parent equation is $x_f = x_0 + v_0 t + \frac{1}{2} g t^2$. With the initial velocity being 0 m/s, the equation becomes:

$$t = \sqrt{\frac{2x_0}{g}}$$

Where x_0 is the height.

In [5]:
```
import numpy as np
h = float(input('What is the height of the tower (in meters)? '))
v0 = 0
g = 9.8
t = np.sqrt(2*h/g)
print(f'It takes {t:.3f} seconds for the ball to hit the ground.')
```

```
What is the height of the tower (in meters)? 100
It takes 4.518 seconds for the ball to hit the ground.
```

Conclusion

This problem was simple. I forgot to make the input a float at first. But this was something I had done many times before in PH 121.

Exercise 2.2: Altitude of a satellite

Introduction

The code below will be able to do these different calculations. I will make a function to calculate the altitude.

Instructions

A satellite is to be launched into a circular orbit around the Earth so that it orbits the planet once every T seconds. a) Show that the altitude h above the Earth's surface that the satellite must have is

$$h = (\frac{GMT^2}{4\pi^2})^{\frac{1}{3}} - R$$

where G = $6.67 * 10^{-11} m^3 kg^{-1} s^{-2}$ is Newton's gravitational constant, M = $5.97 * 10^2 4$ kg is the mass of the Earth, and R = 6371 km is its radius.

b) Write a program that asks the user to enter the desired value of T and then calculates and prints out the correct altitude in meters.

c) Use your program to calculate the altitudes of satellites that orbit the Earth once a day (so-called "geosynchronous" orbit), once every 90 minutes, and once every 45 minutes. What do you conclude from the last of these calculations?

d) Technically a geosynchronous satellite is one that orbits the Earth once per sidereal day, which is 23.93 hours, not 24 hours. Why is this? And how much difference will it make to the altitude of the satellite?

In [10]:
```
import numpy as np
#Function
def altitude(T):
    G = 6.67e-11
    M = 5.97e24
    R = 6371e3
    return (G*M*T**2 / 4/np.pi**2)**(1/3)-R

#Part B
T = float(input('What is the period of the orbit (in seconds)? '))
print(f'The needed altitude is {altitude(T):.3f} meters.')

#Part C
geosync = 24*60*60
print(f'The altitude of a geosynchronous orbit is {altitude(geosync):.3f} meters')
min90 = 90*60
print(f'The altitude of a 90 minute orbit is {altitude(min90):.3f} meters')
min45 = 45*60
print(f'The altitude of a 45 minute orbit is {altitude(min45):.3f} meters')
# We cannot have something orbit at that rate

#Part D
newgeosync = 23.93*60*60
```

```
What is the period of the orbit (in seconds)? 2800
The needed altitude is -2078745.180 meters.
The altitude of a geosynchronous orbit is 35855910.176 meters
The altitude of a 90 minute orbit is 279321.625 meters
The altitude of a 45 minute orbit is -2181559.898 meters
```

Exercises 2.5 & 2.6

Exercise 2.5: Quantum potential step

A well-known quantum mechanics problem involves a particle of mass m that encounters a one-dimensional potential step.

The particle with initial kinetic energy E and wavevector $k_1 = \sqrt{\frac{2mE}{\hbar^2}}$ enters from the left and encounters a

sudden jump in potential energy of height V at position $x = 0$. By solving the Schrodinger equation, one can show that when E > V the particle may either (a) pass the step,in which case it has a lower kinetic energy of

E − V on the other side and a correspondingly smaller wavevector of $k_2 = \sqrt{\frac{2m(E-V)}{\hbar^2}}$, or (b) it may be reflected, keeping all of its kinetic energy and an unchanged wavevector but moving in the opposite direction. The probabilities T and R for transmission and reflection are given by

$$T = \frac{4k_1k_2}{(k_1+k_2)^2}, R = (\frac{k_1-k_2}{k_1+k_2})^2$$

Suppose we have a particle with mass equal to the electron mass $m = 9.11x10^{-31}$ kg and energy 10 eV encountering a potential step of height 9 eV. Write a Python program to compute and print out the transmission and reflection probabilities using the formulas above.

In [1]:
```
import numpy as np
def k1(E):
    ħc=197.3 #eV nm
    mc2 = .511e6 #eV nm^2
    return np.sqrt(2*mc2*E/ħc**2)
def k2(E,V):
    ħc=197.3 #eV nm
    mc2 = .511e6 #eV nm^2
    return np.sqrt(2*mc2*(E-V)/ħc**2)
def T(E,V):
    k_1 =k1(E)
    k_2 = k2(E,V)
    return 4*k_1*k_2/(k_1+k_2)**2
def R(E,V):
    k_1 =k1(E)
    k_2 = k2(E,V)
    return ((k_1-k_2)/(k_1+k_2))**2

E = 10 #Energy in eV
V = 9  #Height in eV

print(f'Transmission: {T(E,V):.3f} \nReflection: {R(E,V):.3f}')
```

```
Transmission: 0.730
Reflection: 0.270
```

Conlusion

I used functions to calculate it so I could change the numbers in the future and it would still work. You can make sure the formulas are correct by adding R&T. If it equals 1, it is correct.

Exercise 2.6: Planetary orbits

The orbit in space of one body around another, such as a planet around the Sun, need not be circular. In general it takes the form of an ellipse, with the body sometimes closer in and sometimes further out. If you are given the distance ℓ_1 of closest approach that a planet makes to the Sun, also called its perihelion, and its linear velocity v_1 at perihelion, then any other property of the orbit can be calculated from these two as follows.

a) Kepler's second law tells us that the distance ℓ_2 and velocity v_2 of the planet at its most distant point, or aphelion, satisfy $\ell_2 v_2 = \ell_1 v_1$. At the same time the total energy, kinetic plus gravitational, of a planet with velocity v and distance r from the Sun is given by

$$E = \frac{1}{2}mv^2 - G\frac{mM}{r}$$

where m is the planet's mass, $M = 1.9891 \times 10^{30}$ kg is the mass of the Sun, and $G = 6.6738 \times 10^{-11} m^3 kg^{-1} s^{-2}$ is Newton's gravitational constant. Given that energy must be conserved, show that v_2 is the smaller root of the quadratic equation

$$v_2^2 - \frac{2GM}{v_1 \ell_1} v_2 - (v_1^2 - \frac{2GM}{\ell_1}) = 0$$

Once we have v_2 we can calculate ℓ_2 using the relation $\ell_2 = \ell_1 v_1 / v_2$.

b) Given the values of v_1, ℓ_1, and ℓ_2, other parameters of the orbit are given by simple formulas can that be derived from Kepler's laws and the fact that the orbit is an ellipse:

Semi-major axis:$a = \frac{1}{2}(\ell_1 + \ell_2)$ Semi-minor axis: $b = \sqrt{\ell_1 \ell_2}$ Orbital period: $T = \frac{2\pi ab}{\ell_1 v_1}$ Orbital eccentricity: $e = \frac{\ell_2 - \ell_1}{\ell_2 + \ell_1}$

Write a program that asks the user to enter the distance to the Sun and velocity at perihelion, then calculates and prints the quantities ℓ2, v2, T, and e.

c) Test your program by having it calculate the properties of the orbits of the Earth (for which $\ell_1 = 1.4710 \times 10^{11}$ m and $v_1 = 3.0287 \times 10^4 ms^{-1}$) and Halley's comet ($\ell_1 = 8.7830 \times 10^{10} m$ and $v_1 = 5.4529 \times 10^4 ms^{-1}$). Among other things, you should find that the orbital period of the Earth is one year

In [11]:

```
import numpy as np
#Part B
M = 1.9891e30
G = 6.6738e-11

def v_2(v1,l1):
    a = 1
    b = -2*G*M/v1/l1
    c = -(v1**2-2*G*M/l1)
    plus= (-b+np.sqrt(b**2 - 4*a*c))/2/a
    minus =  (-b-np.sqrt(b**2 - 4*a*c))/2/a
    if minus>0 and minus < plus:
        return minus
    else:
        return plus
def l_2(v1,l1):
    v2 = v_2(v1,l1)
    return l1*v1/v2
def semi_major(v1,l1):
    l2=l_2(v1,l1)
    return 1/2*(l1+l2)
def semi_minor(v1,l1):
    l2=l_2(v1,l1)
    return np.sqrt(l1*l2)
def orbital_period(v1,l1):
    a = semi_major(v1,l1)
    b = semi_minor(v1,l1)
    return 2*np.pi*a*b/l1/v1
def orbital_eccen(v1,l1):
    l2=l_2(v1,l1)
    return (l2-l1)/(l2+l1)

distance = float(input("What is your distance to the sun? "))
velocity = float(input("What is your velocity at perihelion? "))

print(f'l2: {l_2(velocity,distance):3e}\nv2: {v_2(velocity,distance):.3e}\nT: {orbita

#Part C
l1_earth = 1.471e11
v1_earth = 3.0287e4
l1_Halley = 8.7830e10
v1_Halley = 5.4529e4
print(f'Earth\nl2: {l_2(v1_earth,l1_earth):3e}\nv2: {v_2(v1_earth,l1_earth):.3e}\nT:
print(f'Halley\nl2: {l_2(v1_Halley,l1_Halley):3e}\nv2: {v_2(v1_Halley,l1_Halley):.3e}
```

```
What is your distance to the sun? 1
What is your velocity at perihelion? 1
l2: 3.766519e-21
v2: 2.655e+20
T: 1.928e-10
e:-1.000e+00
Earth
l2: 1.520272e+11
v2: 2.931e+04
T: 1.000 years
e:1.647e-02
Halley
l2: 5.282215e+12
v2: 9.067e+02
T: 76.082years
e:9.673e-01
```

Conclusion

I wrote a lot of functions. The hardest part was calculating v2. I hadn't added the if statement at first and it was giving me the wrong numbers.