**Problem 1 – Altitude of a Satellite**

*Code*

import math

G = 6.67e-11 # m^3 / (kg \* s^2)

M\_earth = 5.97e24 # kg

R\_earth = 6371.0 # km

M\_mars = 0.107 \* M\_earth

R\_mars = 3390.0 # km

M\_jupiter = 317.8 \* M\_earth

R\_jupiter = 69911.0 # km

Radii = [R\_earth, R\_mars, R\_jupiter]

Masses = [M\_earth, M\_mars, M\_jupiter]

planets = ['Earth', 'Mars', 'Jupiter']

def height(R, M, T):

return (G \* M \* T\*\*2 / (4.0 \* math.pi\*\*2))\*\*(1.0/3.0) - (R \* 1000.0)

for i in range(len(planets)):

nextPlanet = 'n'

while (nextPlanet != 'y') & (nextPlanet != 'Y'):

print 'Planet:', planets[i]

T = float(input('Please enter the desired period in seconds: '))

print 'Altitude:', height(Radii[i], Masses[i], T), 'meters'

if planets[i] != 'Jupiter':

nextPlanet = raw\_input('Next planet? (y/n): ')

print '\n'

else:

nextPlanet = raw\_input('Quit? (y/n): ')

print '\n'

*Results*

Planet: Earth

Please enter the desired period in seconds: 3600\*24

Altitude: 35855910.1762 meters

Next planet? (y/n): n

Planet: Earth

Please enter the desired period in seconds: 60\*90

Altitude: 279321.625373 meters

Next planet? (y/n): n

Planet: Earth

Please enter the desired period in seconds: 60\*45

Altitude: -2181559.89781 meters

Next planet? (y/n): y

Planet: Mars

Please enter the desired period in seconds: 3600\*24

Altitude: 16657054.1586 meters

Next planet? (y/n): n

Planet: Mars

Please enter the desired period in seconds: 60\*90

Altitude: -232786.809642 meters

Next planet? (y/n): n

Planet: Mars

Please enter the desired period in seconds: 60\*45

Altitude: -1401080.32123 meters

Next planet? (y/n): y

Planet: Jupiter

Please enter the desired period in seconds: 3600\*24

Altitude: 218253582.64 meters

Quit? (y/n): n

Planet: Jupiter Please enter the desired period in seconds: 60\*90

Altitude: -24527922.0622 meters

Quit? (y/n): n

Planet: Jupiter

Please enter the desired period in seconds: 60\*45

Altitude: -41321452.3986 meters

Quit? (y/n): y

**Problem 2 – Special Relativity**

*Code*

import math

c = 299792458 # m/s

ly\_to\_m = c \* (3600.0 \* 24.0 \* 365.25)

x = float(raw\_input('Distance to destination, in light-years: '))

print x, 'light-years equals', x \* ly\_to\_m, 'meters.'

v = float(raw\_input('Velocity, as a fraction of the speed of light c: '))

t = x \* ly\_to\_m / (v \* c)

t\_prime = (t - (v \* c) \* x / (c\*\*2)) \* math.sqrt(1.0 - (v \* c)\*\*2 / c\*\*2)

print 'An observer on the Earth will see the ship take', t, 'seconds (or', t / (3600.0 \* 24.0 \* 365), 'years) to reach its destination.'

print 'An observer in the spaceship will see the ship take', t\_prime, 'seconds (or', t\_prime / (3600.0 \* 24.0 \* 365), 'years)to reach its destination.'

*Results*

Distance to destination, in light-years: 12

12.0 light-years equals 1.13528765671e+17 meters.

Velocity, as a fraction of the speed of light c: .99

An observer on the Earth will see the ship take 382516363.636 seconds (or 12.1295143213 years) to reach its destination.

An observer in the spaceship will see the ship take 53960573.4972 seconds (or 1.71107856092 years)to reach its destination.

**Problem 3 – Madelung Constant**

*Code*

import math

def V(i,j,k):

if (i + j + k) == 0.0:

return 0.0

else:

return (-1.0)\*\*(i + j + k) / (math.sqrt(i\*\*2.0 + j\*\*2.0 + k\*\*2.0))

length = range(-100,100)

M = 0.0

for i in length:

for j in length:

for k in length:

M += V(i,j,k)

print M

*Results*

-464.852266805

**Problem 4 – Prime Numbers**

*Code*

import math

import numpy as np

Nums = np.arange(3.0, 10000.0)

Primes = [2.0]

for i in Nums:

prime = False

notprime = False

while (prime == False) & (notprime == False):

for j in Primes:

if i%j == 0:

notprime = True

elif j >= math.sqrt(i):

prime = True

if (prime == True) & (notprime == False):

Primes.append(i)

print Primes[-5:]

*Results*

[9931.0, 9941.0, 9949.0, 9967.0, 9973.0]

**Problem 5 – The Semi-Empirical Mass Formula**

*Code*

import math

import numpy as np

a1 = 15.67 # MeV

a2 = 17.23 # MeV

a3 = 0.75 # MeV

a4 = 93.2 # MeV

def Even(num):

if (num % 2) == 0:

return True

else:

return False

def a5(A, Z):

if Even(A) == False:

return 0.0

elif Even(Z) == True:

return 12.0

else:

return -12.0

def BperA(A, Z):

num1 = A

num2 = A\*\*(2.0 / 3.0)

num3 = Z\*\*2.0 / (A\*\*(1.0/3.0))

num4 = (A - 2.0 \* Z)\*\*2.0 / A

num5 = A\*\*(-1.0 / 2.0)

B = (a1 \* num1 - a2 \* num2 - a3 \* num3 - a4 \* num4 + a5(A, Z) \* num5)

return B/A

def MostStable(Z):

A = np.arange(Z, 3.0 \* Z)

largestBperA= 0.0

mostStableA = 0.0

for a in A:

if BperA(a, Z) > largestBperA:

largestBperA = BperA(a, Z)

mostStableA = a

return [mostStableA, largestBperA]

Z = np.arange(1.0, 100.0)

largestBperA = 0.0

mostStableA = 0.0

mostStableZ = 0.0

for z in Z:

resultList = MostStable(z)

if resultList[1] > largestBperA:

largestBperA = resultList[1]

mostStableA = resultList[0]

mostStableZ = z

print 'Most stable Z:', mostStableZ, '\nMost stable A:', mostStableA, '\nLargest binding energy:', largestBperA

*Results*

Most stable Z: 24.0

Most stable A: 50.0

Largest binding energy: 8.53262275137