```
#Assignment 1- Alexandra Mulholland 17336557
#Professor Aline Vidotto- SS Planetary and Space science

import numpy as np
import matplotlib.pyplot as plt

#---------------------------------------------------------------------------
#Calculating values of the Jeans flux over the number density of various
#molecules and elements at the exobase.
#---------------------------------------------------------------------------

#We are going to use molecular weights ranging from that of Hydrogen to CO2
#i.e. where mu ranges from 1 to 44 where mass=mu*(proton mass)
#Firstly defining the parameters present in the expression for flux over number
#density. We require the most probable velocity of each particle in the MB
#distribution, for which we need their masses and the temperature.
#Assuming the temperature at the exobase is 1000K

T=1000 #[K]
k=1.38e-23 #Boltmann constant [m^2 kg s^-2 K^-1]
mu=np.array([1,2,4,16,17,18,28,40,44])

#Some species present in Earth's atmosphere, of varying abundance
#Species          mu value
#H                    1
#H2                   2
#He                   4
#O                   16
#CH4(methane)    16
#NH3(ammonium)17
#H2O                 18
#N2                  28
#Ar                  40
#CO2                 44
#mass of each species for mu ranging from 1 to 44 is m_p * mu
m_p=1.67e-27 #proton mass [kg]
mass=mu*m_p
#Most likely velocity of each species according to MB distribution:
v_o=(2*k*T/mass)**(1.0/2.0) #[m/s]

#Escape velocity of Earth
v_esc=10800 #[m/s]

#defining lambda to simplify expression
lam=(v_esc/v_o)**2 #unitless

#Naming the flux over the number density as 'effusion'
def effusion(v_o,lam):
    return (v_o*(1+lam)*np.exp(-lam))/(2*np.sqrt(np.pi))
#---------------------------------------------------------------------------
#BEST FIT LINES THROUGH POINTS FOR EARTH
#---------------------------------------------------------------------------

mu_bestfit=np.linspace(1,44,num=1000)
v_ofit=(2*k*T/(m_p*mu_bestfit))**(1.0/2.0)
lam_bestfit=(v_esc/v_ofit)**2
def eff_bestfit(v_ofit,lam_bestfit):
    return (v_ofit*(1+lam_bestfit)*np.exp(-lam_bestfit))/(2*np.sqrt(np.pi))
#----------------------------------------------------------------------------
```

```
fig=plt.subplots(figsize=(10,10))
plt.scatter(mu, effusion(v_o,lam),marker='o',zorder=10, label='Effusion rate on Earth')
plt.plot(mu_bestfit,eff_bestfit(v_ofit,lam_bestfit))
plt.xlabel('Mean molecular mass of species, \u03BC',fontsize=(12))
plt.ylabel('Effusion rate [m/s]',fontsize=(12))
plt.annotate(" Earth H: \n Effusion rate \n is ~8m/s", (1, 7.95487891e+000),xytext=(-1, 17))
plt.annotate("   Earth H$_2$ \n ~0.009m/s",(2,9.08374671e-003),xytext=(-1,-22))
plt.annotate(' Earth He~10$^{-8}$m/s',(2,9.20550357e-009),xytext=(4, -12))
plt.annotate('Earth and \n moon O',(16,2.97200313e-045),xytext=(13, -22))
plt.annotate(' Moon and \n Earth NH$_3$',(17,2.63589311e-048),xytext=(15,7))
plt.annotate(' Earth and \n moon H$_2$O',(18,2.33387975e-051),xytext=(18, -22))
plt.annotate('Earth and \n moon N$_2$',(28,6.48921008e-082),xytext=(27,-22))
plt.annotate('Earth and \n moon Ar',(40,1.28305942e-118),xytext=(39,-22))
plt.annotate('Earth CO$_2$',(44,7.38851584e-131),xytext=(42, 7))

#Doing this also for the moon, whose atmospheric temperature we must estimate
#See details of temperature estimate for the moon's surface in notes
T_moon=270 #[K]
#The species' most likely velocity would be different in a different temperature
v_o_moon=np.sqrt(2*k*T_moon/mass)

#escape velocity of the moon- see notes
v_esc_moon=2375.8 #[m/s]

lam_moon=(v_esc_moon/v_o_moon)**2
#---------------------------------------------------------------------------
#BEST FIT LINE FOR LUNAR POINTS
#---------------------------------------------------------------------------
v_ofit_moon=(2*k*T_moon/(m_p*mu_bestfit))**(1.0/2.0)
lam_bestfit_moon=(v_esc_moon/v_ofit_moon)**2
def eff_bestfit_moon(v_ofit_moon,lam_bestfit_moon):
    return (v_ofit_moon*(1+lam_bestfit_moon)*np.exp(-lam_bestfit_moon))/(2*np.sqrt(np.pi))
#---------------------------------------------------------------------------

def effusion_moon(v_o_moon,lam_moon):
    return (v_o_moon*(1+lam_moon)*np.exp(-lam_moon))/(2*np.sqrt(np.pi))
plt.plot(mu_bestfit,eff_bestfit_moon(v_ofit_moon,lam_bestfit_moon))
plt.scatter(mu,effusion_moon(v_o_moon,lam_moon), marker='D',s=60,label='Effusion rate on the moon')
plt.legend(prop={'size': 15})
plt.title('Effusion rate for species on Earth and on the moon',fontsize=20)
plt.annotate('Moon H: Effusion rate is ~380m/s',(1,3.80958896e+02),xytext=(2,380))
plt.annotate(' Moon H$_2$ ~120m/s',(2,1.18499671e+02),xytext=(2,122))
plt.annotate('Moon He~11m/s',(2,1.14604188e+01),xytext=(4.5,11))
plt.ylim(-40,400)
plt.savefig('Effusionrate.pdf')
plt.show()

print('Effusion rate on Earth for H = ',effusion(v_o,lam)[0],'m/s')
print('Effusion rate on moon for H = ',effusion_moon(v_o_moon,lam_moon)[0],'m/s')
print('\n')
print('Effusion rate on moon for H\u2082 = ',effusion_moon(v_o_moon,lam_moon)[1],'m/s')
print('Effusion rate on Earth for H\u2082 = ',effusion(v_o,lam)[1],'m/s')
print('\n')
print('Effusion rate on moon for He = ',effusion_moon(v_o_moon,lam_moon)[2],'m/s')
print('Effusion rate on Earth for He = ',effusion(v_o,lam)[2],'m/s')
print('\n')
print('Effusion rate on moon for O = ',effusion_moon(v_o_moon,lam_moon)[3],'m/s')
print('Effusion rate on Earth for O = ',effusion(v_o,lam)[3],'m/s')
print('\n')
```

```
print('Effusion rate on moon for NH\u2083 = ',effusion_moon(v_o_moon,lam_moon)[4],'m/s')
print('Effusion rate on Earth for NH\u2083 = ',effusion(v_o,lam)[4],'m/s')
print('\n')
print('Effusion rate on moon for H\u2082O = ',effusion_moon(v_o_moon,lam_moon)[5],'m/s')
print('Effusion rate on Earth for H\u2082O = ',effusion(v_o,lam)[5],'m/s')
print('\n')
print('Effusion rate on moon for N\u2082 = ',effusion_moon(v_o_moon,lam_moon)[6],'m/s')
print('Effusion rate on Earth for N\u2082 = ',effusion(v_o,lam)[6],'m/s')
print('\n')
print('Effusion rate on moon for Ar = ',effusion_moon(v_o_moon,lam_moon)[7],'m/s')
print('Effusion rate on Earth for Ar = ',effusion(v_o,lam)[7],'m/s')
print('\n')
print('Effusion rate on moon for CO\u2082 = ',effusion_moon(v_o_moon,lam_moon)[8],'m/s')
print('Effusion rate on Earth for CO\u2082 = ',effusion(v_o,lam)[8],'m/s')
```