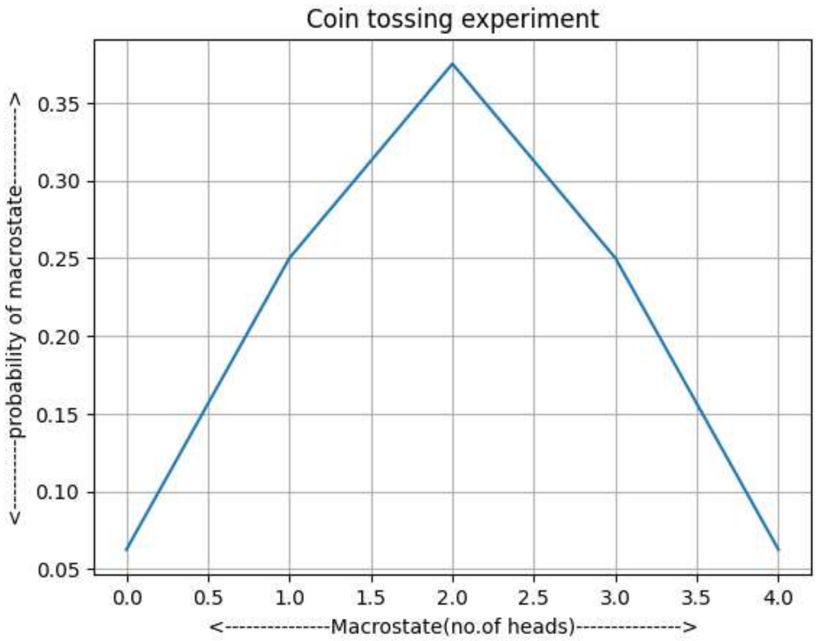
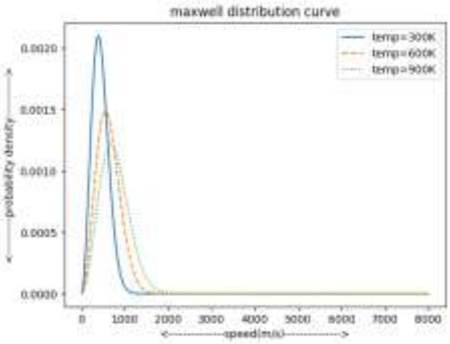
```python
import numpy as np
import matplotlib.pyplot as plt
from math import factorial as fact

n=int(input("                 no. of coins="))
T=2**n
r=[]
m=[]
Y=[]
for i in range(n+1):
  r.append(i)
  b=fact(n)/(fact(n-i)*fact(i))
  m.append(b)
  y=b/T
  Y.append(y)
print("(macrostates)no. of heads :",r)
print("        no. of microstates :",m)
print("probability of macrostate :",Y)

plt.plot(r,Y)
plt.xlabel("<---------------Macrostate(no.of heads)--------------->")
plt.ylabel("<----------probability of macrostate----------->")
plt.title("Coin tossing experiment")
plt.grid(True)
plt.show()
```

```
              no. of coins=4
(macrostates)no. of heads : [0, 1, 2, 3, 4]
       no. of microstates : [1.0, 4.0, 6.0, 4.0, 1.0]
probability of macrostate : [0.0625, 0.25, 0.375, 0.25, 0.0625]
```



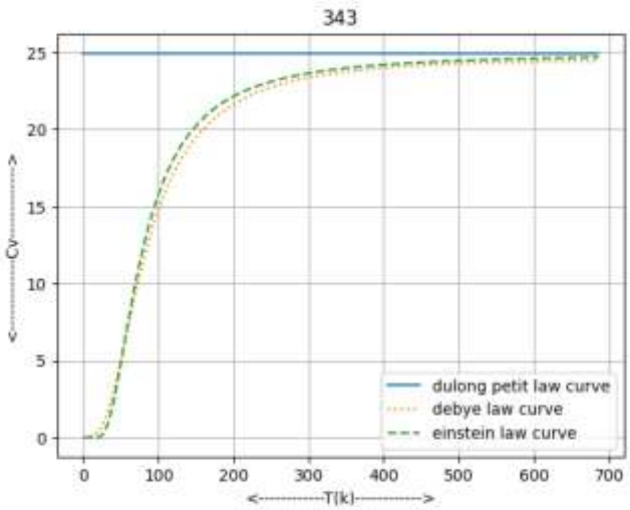
```python
import numpy as np
import matplotlib.pyplot as plt

M=int(input("Molar mass (gm/mol)="))
k=1.38e-23
Na=6.023e23
m=M/(Na*1000)
def MB(v,t):
  return 4*np.pi*v**2*(m/(2*np.pi*k*t))**(3/2)*np.exp(-m*v**2/(2*k*t))
v=np.linspace(10,8000,1000)
temp=[300,600,900]
for t in temp:
  vr=np.sqrt(3*k*t/m)
  va=np.sqrt(8*k*t/(np.pi*m))
  vm=np.sqrt(2*k*t/m)
  print("Temprature(K) =",t)
  print("root mean square speed (m/s) =",vr)
  print("         average speed (m/s) =",va)
  print("   most probable speed (m/s) =",vm)

plt.plot(v,MB(v,300),label="temp=300K")
plt.plot(v,MB(v,600),label="temp=600K",ls="--")
plt.plot(v,MB(v,900),label="temp=900K",ls=":")
plt.xlabel("<---------------speed(m/s)--------------->")
plt.ylabel("<----------probability density----------->")
plt.title("maxwell distribution curve")
plt.legend()
plt.show()
```

```
Molar mass (gm/mol)=32
Temprature(K) = 300
root mean square speed (m/s) = 483.4952817763581
         average speed (m/s) = 445.45277640193746
   most probable speed (m/s) = 394.77224446508393
Temprature(K) = 600
root mean square speed (m/s) = 683.7655848315268
         average speed (m/s) = 629.9653577843698
   most probable speed (m/s) = 558.2922621709887
Temprature(K) = 900
root mean square speed (m/s) = 837.4383932564831
         average speed (m/s) = 771.5468411007744
   most probable speed (m/s) = 683.7655848315269
```



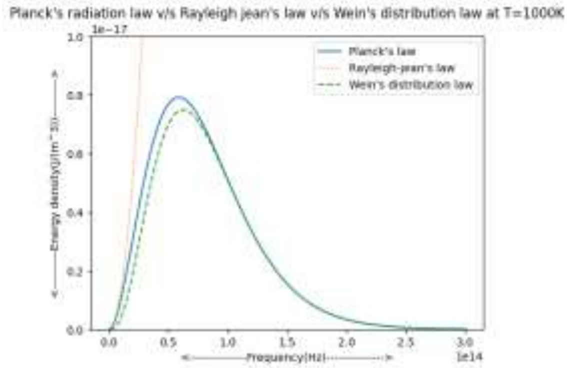
```
import matplotlib.pyplot as plt
import numpy as np
from scipy.integrate import quad
k=1.38e-23
Na=6.023e23
h=6.626e-34
R=Na*k
n=str(input   ("     Enter the name of solid:"))
od=float(input("   Enter Debye temprature(K):"))
oe=float(input("Enter Einstein temprature(K):"))
def ed(t):
  return 3*R*(oe/t)**2*np.exp(oe/t)/((np.exp(oe/t)-1)**2)
def db(t):
  e1=lambda x:(x**4*np.exp(x)/((np.exp(x)-1)**2))
  eq=9*R*((t/od)**3)*quad(e1,0.1,(od/t))[0]
  return eq
t=np.linspace(0,2*od,1000)
d=[];e=[];dp=[]
for i in range(len(t)):
  dp.append(3*R)
  d.append(db(t[i]))
  e.append(ed(t[i]))
plt.plot(t,dp,label="dulong petit law curve")
plt.plot(t,d,label="debye law curve",ls=':')
plt.plot(t,e,label="einstein law curve",ls='--')
plt.xlabel("<------------T(k)------------>")
plt.ylabel("<-------------Cv------------->")
plt.grid(True)
plt.legend()
plt.title(n)
plt.show()
```

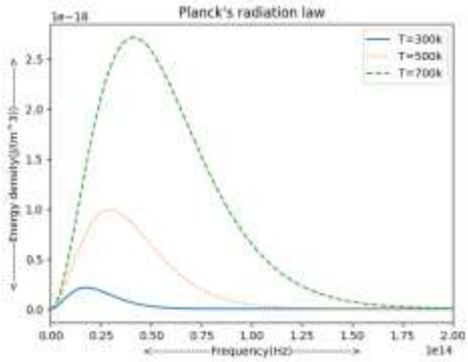
```
     Enter the name of solid:Copper
   Enter Debye temprature(K):343
Enter Einstein temprature(K):240
```

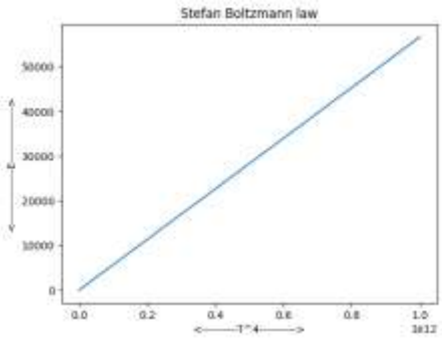


```python
import matplotlib.pyplot as plt
import numpy as np

h=6.626e-34
c=3.0e8
k=1.38e-23
a=5.67e-8
b=2.89e-3
f=np.linspace(1.0e10,3.0e14,1000)
t=np.linspace(1,1000,1000)

def p(f,t):
  return 8*np.pi*h*f**3/(c**3*(np.exp(h*f/(k*t))-1))
def r(f,t):
  return 8*np.pi*k*t*f**2/c**3
def w(f,t):
  return 8*np.pi*h*f**3/(c**3*np.exp(h*f/(k*t)))
def s(t):
  return a*t**4
def wd(t):
  return b/t

plt.plot(f,p(f,1000),label="Planck's law")
plt.plot(f,r(f,1000),':',label="Rayleigh-jean's law")
plt.plot(f,w(f,1000),'--',label="Wein's distribution law")
plt.xlabel("<----------------Frequency(Hz)---------------->")
plt.ylabel("<-----------Energy density(J/(m^3))----------->")
plt.ylim(0,10e-18)
plt.title("Planck's radiation law v/s Rayleigh jean's law v/s Wein's 
distribution law at T=1000K")
plt.legend()
plt.show()

plt.plot(f,p(f,300),label="T=300k")
plt.plot(f,p(f,500),':',label="T=500k")
plt.plot(f,p(f,700),'--',label="T=700k")
plt.xlabel("<----------------Frequency(Hz)---------------->")
plt.ylabel("<-----------Energy density(J/(m^3))----------->")
plt.title("Planck's radiation law")
plt.xlim(0,2e14)
plt.legend()
plt.show()

plt.plot(t**4,s(t))
plt.ylabel("<----------------E---------------->")
plt.xlabel("<-----------T^4----------->")
plt.title("Stefan Boltzmann law")
plt.show()
```

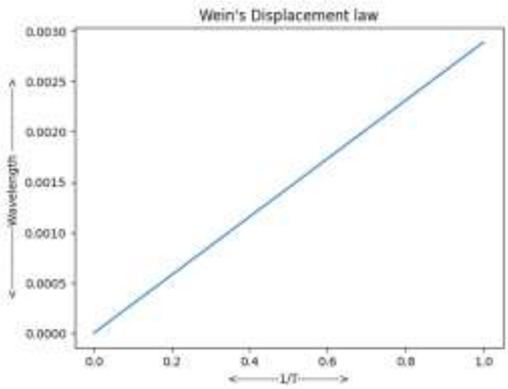
```
print("Slope = 5.67e-8")

plt.plot(1/t,wd(t))
plt.xlabel("<----------1/T---------->")
plt.ylabel("<---------------Wavelength --------------->")
plt.title("Wein's Displacement law")
plt.show()
print("Slope = 0.00289")
```







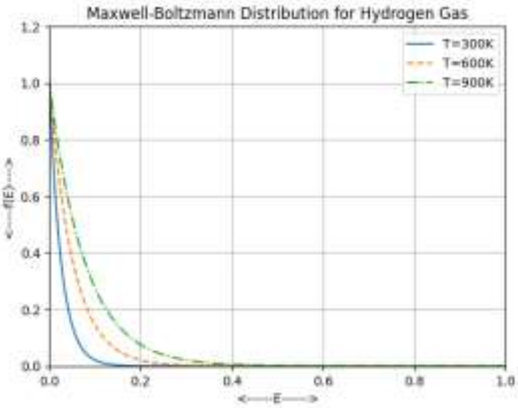

```
Slope = 5.67e-8
```

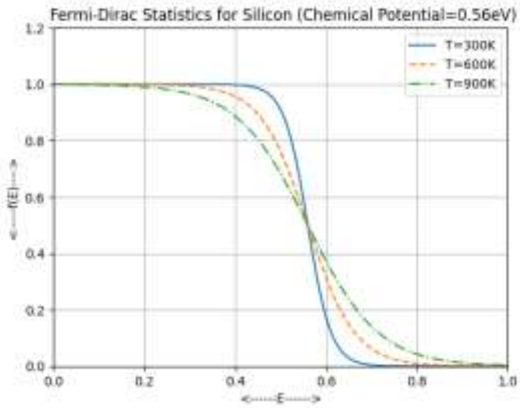


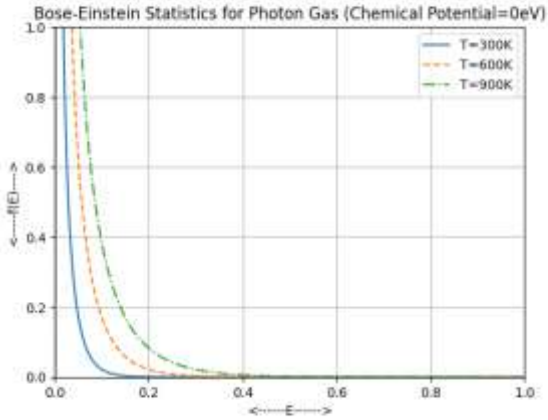

```
Slope = 0.00289
```

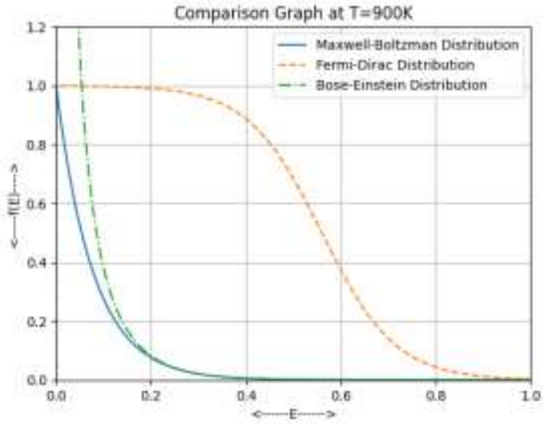
```python
import numpy as np
import matplotlib.pyplot as plt
def dis1(T):
  N=[]
  for i in range(len(E)):
    n=1/np.exp(((E[i])*e)/(k*T))
    N.append(n)
  return N
def dis2(T,a,nU):
  M=[]
  for i in range(len(E)):
    n=((E[i]-nU)*e)/(k*T)
    m=np.exp(n)+a
    fE=1/(m)
    M.append(fE)
  return M
e=1.6e-19
k=1.38e-23
E=np.linspace(-1,1,1000)
T=[300,600,900]
a1=1  #FD
a2=-1 #BE
nU1=0.56 # Chemical potential:(Photon gas)=0;(Silicon)=0.56
nU2=0

MB1=[];MB2=[];MB3=[]

FD1=[];FD2=[];FD3=[]

BE1=[];BE2=[];BE3=[]


MB1=dis1(T[0])
FD1=dis2(T[0],a1,nU1)
BE1=dis2(T[0],a2,nU2)

MB2=dis1(T[1])
FD2=dis2(T[1],a1,nU1)
BE2=dis2(T[1],a2,nU2)

MB3=dis1(T[2])
FD3=dis2(T[2],a1,nU1)
BE3=dis2(T[2],a2,nU2)

  #Graph for Maxwell-Boltzmann Distribution:
plt.plot(E, MB1, label="T=300K")
plt.plot(E, MB2, label="T=600K",ls="--")
plt.plot(E, MB3, label="T=900K",ls="dashdot")
plt.xlabel("<------E------>")
```

```
plt.ylabel("<-----f(E)---->")
plt.title("Maxwell-Boltzmann Distribution for Hydrogen Gas")
plt.xlim(0,1)
plt.ylim(0,1.2)
plt.grid()
plt.legend()
plt.show()


  #Graph for Fermi-Dirac Statistics:
plt.plot(E, FD1, label="T=300K")
plt.plot(E, FD2, label="T=600K",ls="--")
plt.plot(E, FD3, label="T=900K",ls="dashdot")
plt.xlabel("<------E------>")
plt.ylabel("<-----f(E)---->")
plt.title(" Fermi-Dirac Statistics for Silicon (Chemical
Potential=0.56eV)")
plt.xlim(0,1)
plt.ylim(0,1.2)
plt.grid()
plt.legend()
plt.show()

  #Graph for Bose-Einstein Statistics
plt.plot(E, BE1, label="T=300K")
plt.plot(E, BE2, label="T=600K",ls="--")
plt.plot(E, BE3, label="T=900K",ls="dashdot")
plt.xlabel("<------E------>")
plt.ylabel("<-----f(E)---->")
plt.title("Bose-Einstein Statistics for Photon Gas (Chemical
Potential=0eV)")
plt.grid()
plt.xlim(0,1)
plt.ylim(0,1)
plt.legend()
plt.show()

  #Comparison Graph
plt.plot(E, MB3, label="Maxwell-Boltzman Distribution")
plt.plot(E, FD3, label="Fermi-Dirac Distribution",ls="--")
plt.plot(E, BE3, label="Bose-Einstein Distribution",ls="dashdot")
plt.xlabel("<------E------>")
plt.ylabel("<-----f(E)---->")
plt.title("Comparison Graph at T=900K")
plt.grid()
plt.xlim(0,1)
plt.ylim(0,1.2)
plt.legend()
plt.show()
```


Maxwell-Boltzmann Distribution for Hydrogen Gas
T=300K
T=600K
T=900K
<----f(E)---->
<-----E----->


Fermi-Dirac Statistics for Silicon (Chemical Potential=0.56eV)
T=300K
T=600K
T=900K
<----f(E)---->
<-----E----->


Bose-Einstein Statistics for Photon Gas (Chemical Potential=0eV)
T=300K
T=600K
T=900K
<----f(E)---->
<-----E----->


Comparison Graph at T=900K
Maxwell-Boltzman Distribution
Fermi-Dirac Distribution
Bose-Einstein Distribution
<----f(E)---->
<-----E----->

```python
import numpy as np
import matplotlib.pyplot as plt
import sympy as sp

n=int(input("Enter number of Energy Levels: "))
N=[100,200,300]
E=float(input("Enter the difference between the said energy levels: "))
K=1.38e-23
e=1.6e-19
t=np.linspace(1,1000,100)
T=sp.Symbol('T')

def z(n,N,E):
  G=[]
  z=0          #Partition Function
  for i in range(0,n):
    g=sp.exp(-(i*E*e)/(K*T))
    G.append(g)
    z+=g
  Z=(z)**N

  #For Probability
  P=[]
  for i in range(3):
    p=G[i]/z
    P.append(p)

  #For Internal Energy
  l=sp.log(Z)
  u=K*(T**2)*(sp.diff(l,T))

  #For Specific Heat
  c=sp.diff(u,T)

  #For Helmholtz-free Energy
  f=-(K*T)*(sp.log(Z))

  #For Entropy
  s=-(sp.diff(f,T))
  return Z,u,c,f,s,P

Pr=[[], [], []]
Zs=[[], [], []]
UE=[[], [], []]
Cv=[[], [], []]
F= [[], [], []]
Ep=[[], [], []]
```

```
for i in range(len(N)):
  Z,u,c,f,s,P=z(n,N[i],E)
  Zsi=Zs[i]
  UEi=UE[i]
  Cvi=Cv[i]
  Fi=F[i]
  Epi=Ep[i]
  Pi=Pr[i]
  Pf=P[i]
  for j in range(len(t)):
    x=Z.subs(T,t[j])
    y=sp.log(x)
    Zsi.append(y)
    u=s.subs(T,t[j])  #Internal Energy
    UEi.append(u)
    d=c.subs(T,t[j])  #Specific Heat
    Cvi.append(d)
    g=f.subs(T,t[j])  #Helmholtz free Energy
    Fi.append(g)
    p=s.subs(T,t[j])  #Entropy
    Epi.append(p)
    g=Pf.subs(T,t[j])
    Pi.append(g)

#Graph for Internal Energy
plt.plot(t,UE[0], label="100 Particles")
plt.plot(t,UE[1], label="200 Particles")
plt.plot(t,UE[2], label="300 Particles")
plt.xlabel('---T(K)--->')
plt.ylabel('---UE(J)--->')
plt.title('Internal Energy vs Temperature')
plt.show()

#Graph for Specific Heat
plt.plot(t,Cv[0], label="100 Particles")
plt.plot(t,Cv[1], label="200 Particles")
plt.plot(t,Cv[2], label="300 Particles")
plt.xlabel('---T(K)--->')
plt.ylabel('---Cv--->')
plt.title('Specific Heat vs Temperature')
plt.show()

#Graph for Helmholtz Free Energy
plt.plot(t,F[0], label="100 Particles")
plt.plot(t,F[1], label="200 Particles")
plt.plot(t,F[2], label="300 Particles")
plt.xlabel('---T(K)--->')
plt.ylabel('---F--->')
```

```
plt.title('Helmholtz free Energy vs Temperature')
plt.show()

#Graph for Entropy
plt.plot(t,Ep[0], label="100 Particles")
plt.plot(t,Ep[1], label="200 Particles")
plt.plot(t,Ep[2], label="300 Particles")
plt.xlabel('---T(K)--->')
plt.ylabel('---S--->')
plt.title('Entropy vs Temperature')
plt.show()

#Graph for Partition Function
plt.plot(t,Zs[0], label="100 Particles")
plt.plot(t,Zs[1], label="200 Particles")
plt.plot(t,Zs[2], label="300 Particles")
plt.xlabel('---T(K)--->')
plt.ylabel('---Z--->')
plt.title('Partition Function vs Temperature')
plt.show()

#Graph for Probability Density
plt.plot(t,Pr[0], label="0th Energy Level")
plt.plot(t,Pr[1], label="1st Energy Level")
plt.plot(t,Pr[2], label="2nd Energy Level")
plt.xlabel('---T(K)--->')
plt.ylabel('---Probability--->')
plt.title('Probability Density vs Temperature')
plt.show()
```

```
Enter number of Energy Levels: 3
Enter the difference between the said energy levels: 0.01
```