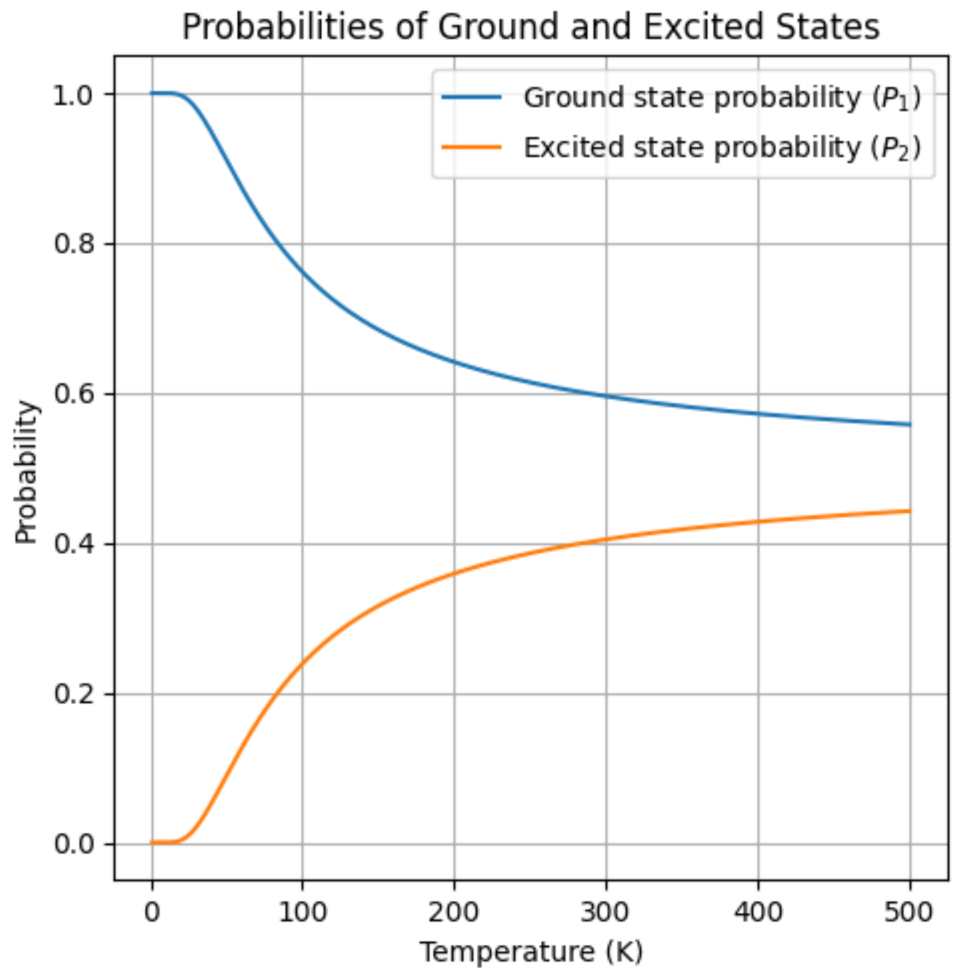
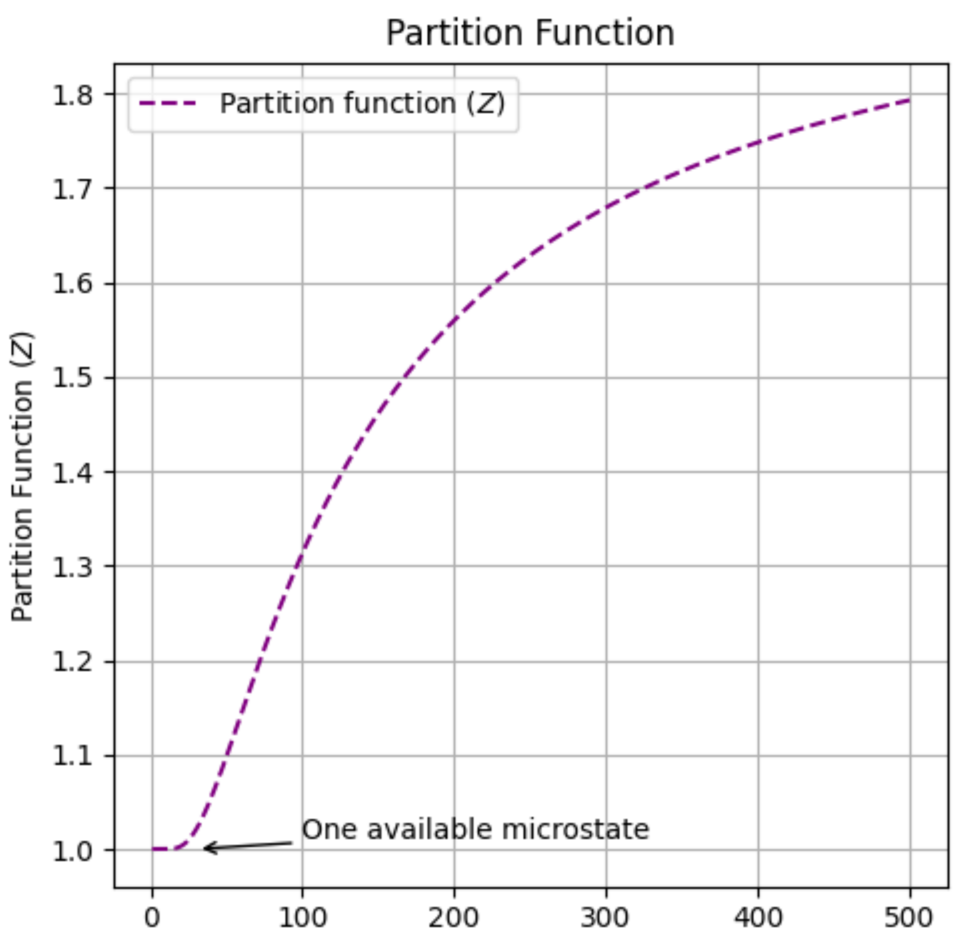
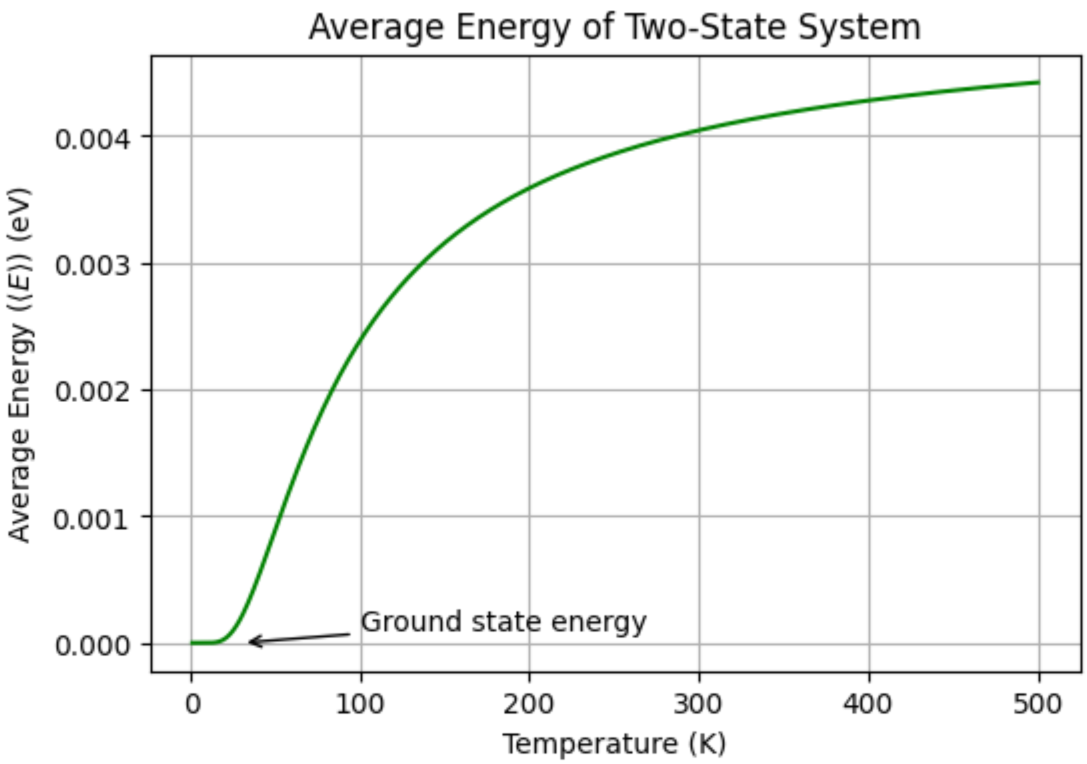
In [3]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.constants import k, eV

# Constants
k_B = k / eV  # Boltzmann constant in eV/K
epsilon = 0.01 # Energy difference in eV

# Temperature range (in Kelvin)
T = np.linspace(1, 500, 500)

# Partition function Z
Z = 1 + np.exp(-epsilon / (k_B * T))

# Probabilities of ground state (P1) and excited state (P2)
P1 = 1 / Z
P2 = np.exp(-epsilon / (k_B * T)) / Z

# Plotting
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(5, 10))

# Plotting probabilities
ax1.plot(T, P1, label='Ground state probability ($P_1$)')
ax1.plot(T, P2, label='Excited state probability ($P_2$)')
ax1.set_xlabel('Temperature (K)')
ax1.set_ylabel('Probability')
ax1.set_title('Probabilities of Ground and Excited States')
ax1.legend()
ax1.grid(True)

# Plotting partition function
ax2.plot(T, Z, label='Partition function ($Z$)', color='purple', linestyle='
ax2.annotate('One available microstate', xy=(30, 1), xytext=(100, 1.01),
             arrowprops=dict(facecolor='black', arrowstyle='->'))
ax2.set_xlabel('Temperature (K)')
ax2.set_ylabel('Partition Function ($Z$)')
ax2.set_title('Partition Function')
ax2.legend()
ax2.grid(True)

plt.tight_layout()
plt.show()
```


Probabilities of Ground and Excited States
Ground state probability (P_1)
Excited state probability (P_2)
Probability
1.0
0.8
0.6
0.4
0.2
0.0
0
100
200
300
400
500
Temperature (K)

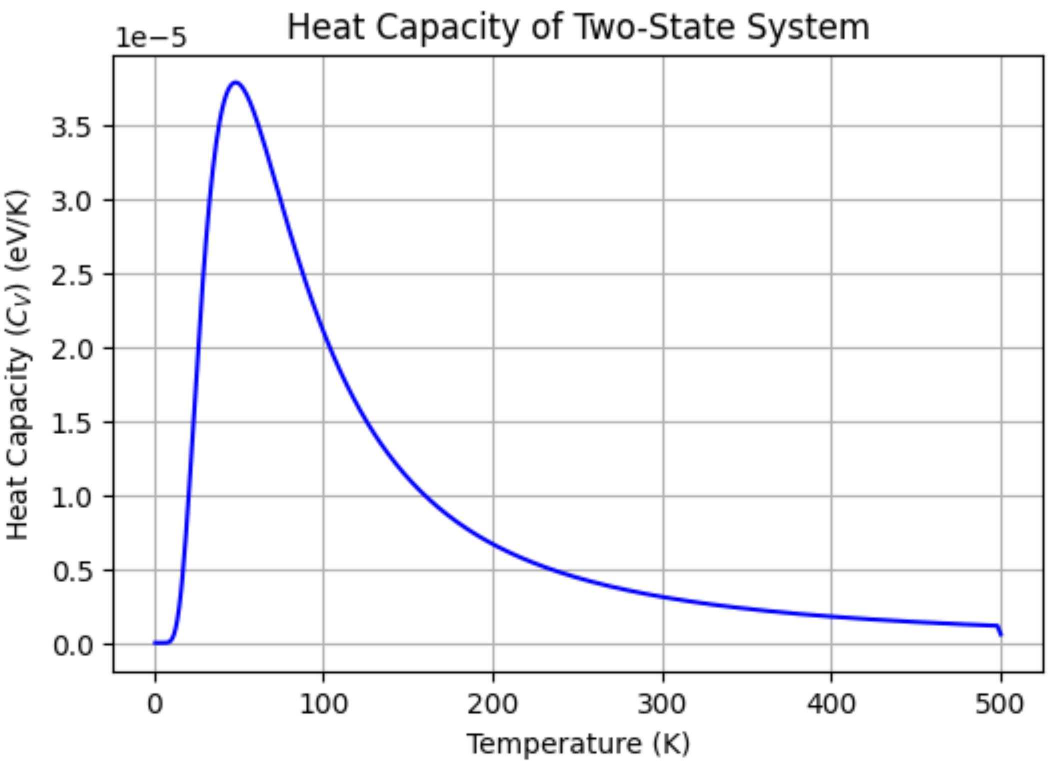

Partition Function
Partition function (Z)
Partition Function (Z)
1.8
1.7
1.6
1.5
1.4
1.3
1.2
1.1
1.0
One available microstate
0
100
200
300
400
500

Temperature (K)

In [4]:
```
# Average energy
E_avg = -np.gradient(np.log(Z), 1 / (k_B * T))

# Plotting average energy
plt.figure(figsize=(6, 4))
plt.plot(T, E_avg, color='green')
plt.annotate('Ground state energy', xy=(30, 0), xytext=(100, 0.0001),
             arrowprops=dict(facecolor='black', arrowstyle='->'))
plt.xlabel('Temperature (K)')
plt.ylabel('Average Energy ($\\langle E \\rangle$) (eV)')
plt.title('Average Energy of Two-State System')
plt.grid(True)
plt.show()
```

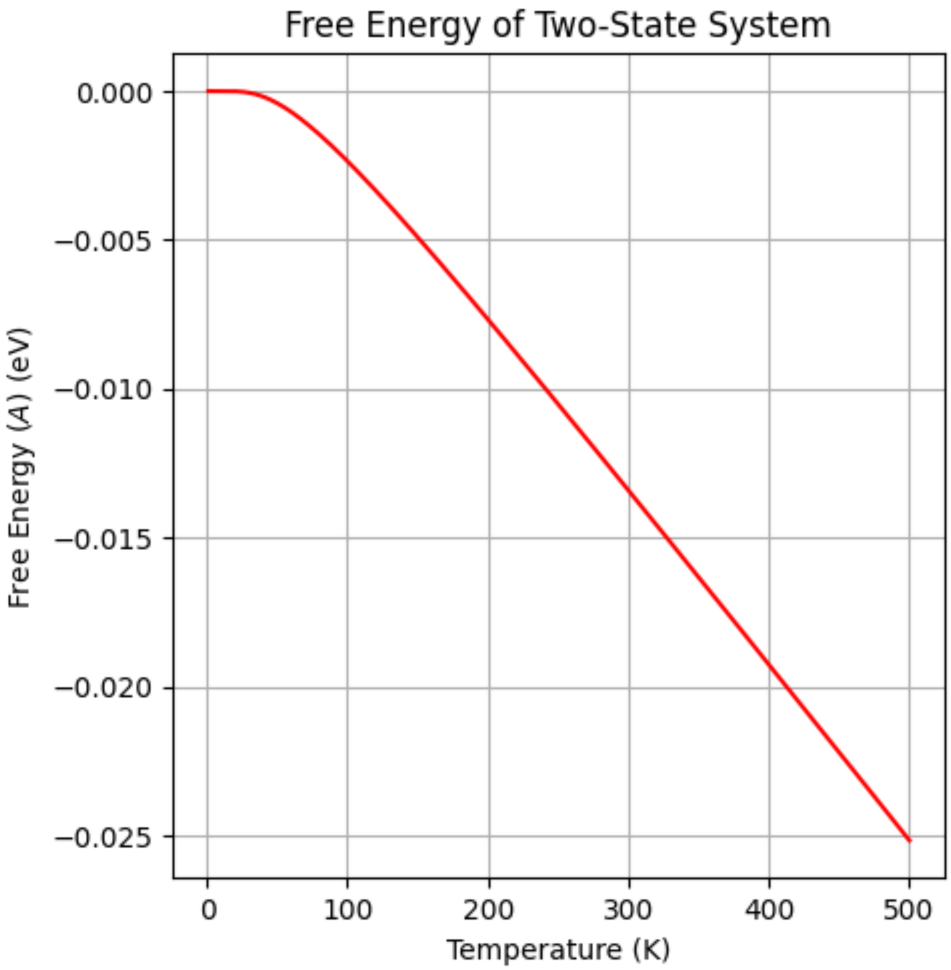


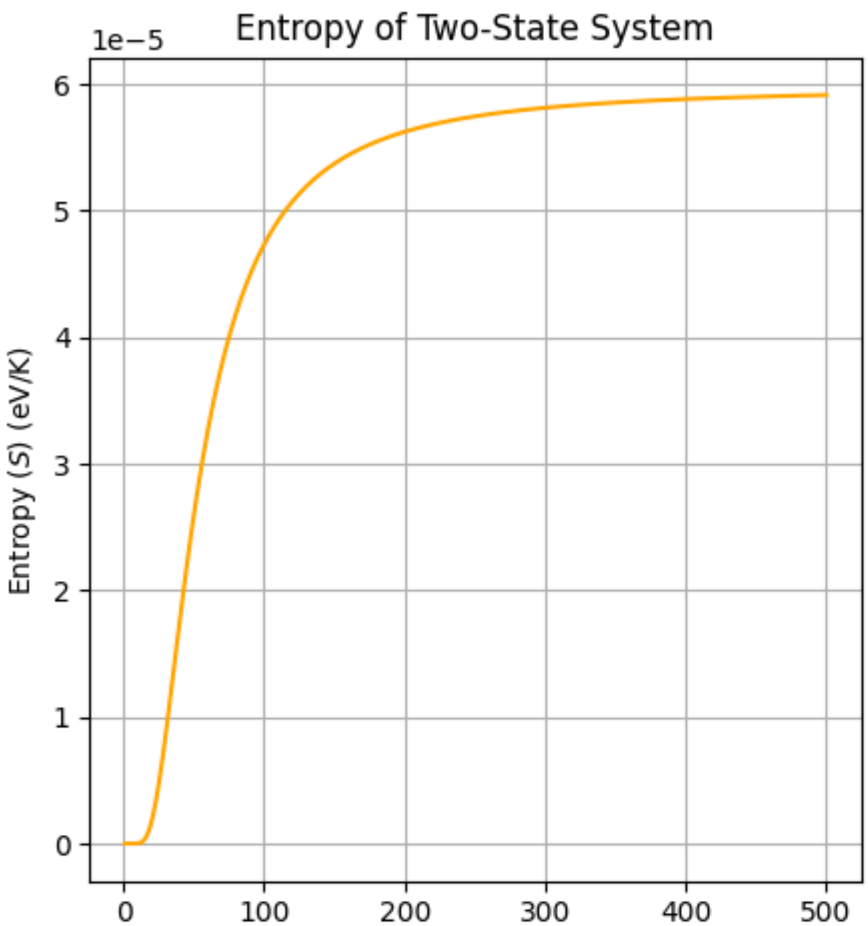
In [5]:
```
# Heat capacity
C_V = np.gradient(E_avg, T)

# Plotting heat capacity
plt.figure(figsize=(6, 4))
plt.plot(T, C_V, color='blue')
plt.xlabel('Temperature (K)')
plt.ylabel('Heat Capacity ($C_V$) (eV/K)')
plt.title('Heat Capacity of Two-State System')
plt.grid(True)
plt.show()
```



In [7]:

```
# Free energy
A = -k_B * T * np.log(Z)

# Entropy
S = -np.gradient(A, T)

# Plotting free energy and entropy
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(5, 10))

# Plotting free energy
ax1.plot(T, A, color='red')
ax1.set_xlabel('Temperature (K)')
ax1.set_ylabel('Free Energy ($A$) (eV)')
ax1.set_title('Free Energy of Two-State System')
ax1.grid(True)

# Plotting entropy
ax2.plot(T, S, color='orange')
ax2.set_xlabel('Temperature (K)')
ax2.set_ylabel(r'Entropy ($S$) (eV/K)')
ax2.set_title('Entropy of Two-State System')
ax2.grid(True)

plt.tight_layout()
plt.show()
```


Free Energy of Two-State System
Free Energy (A) (eV)
0.000
−0.005
−0.010
−0.015
−0.020
−0.025
0
100
200
300
400
500
Temperature (K)


Entropy of Two-State System
1e−5
Entropy (S) (eV/K)
6
5
4
3
2
1
0
0
100
200
300
400
500

Temperature (K)

```
In [ ]:
```