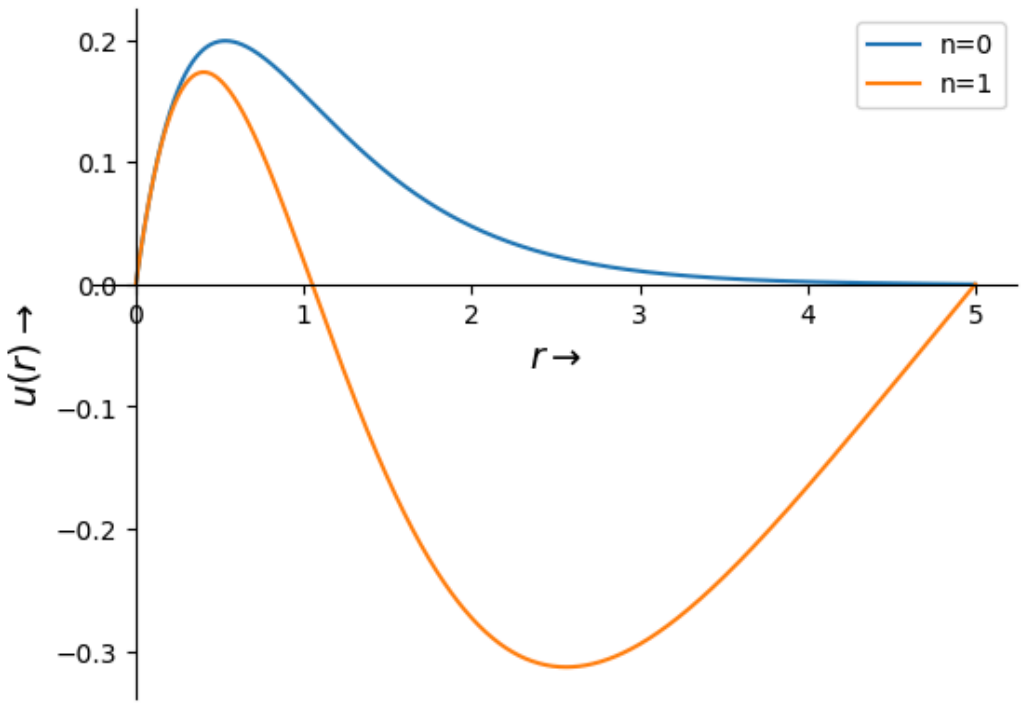
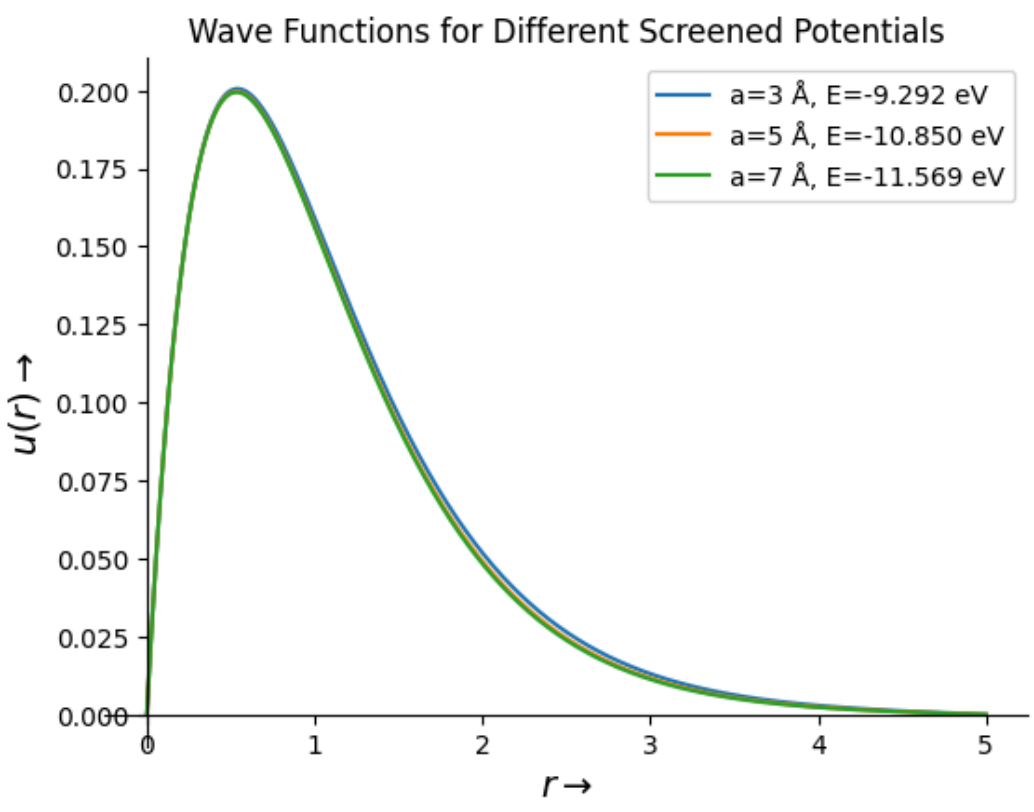
In [3]:
```
import numpy as np
from scipy.integrate import odeint
from scipy.optimize import bisect
import matplotlib.pyplot as plt
```

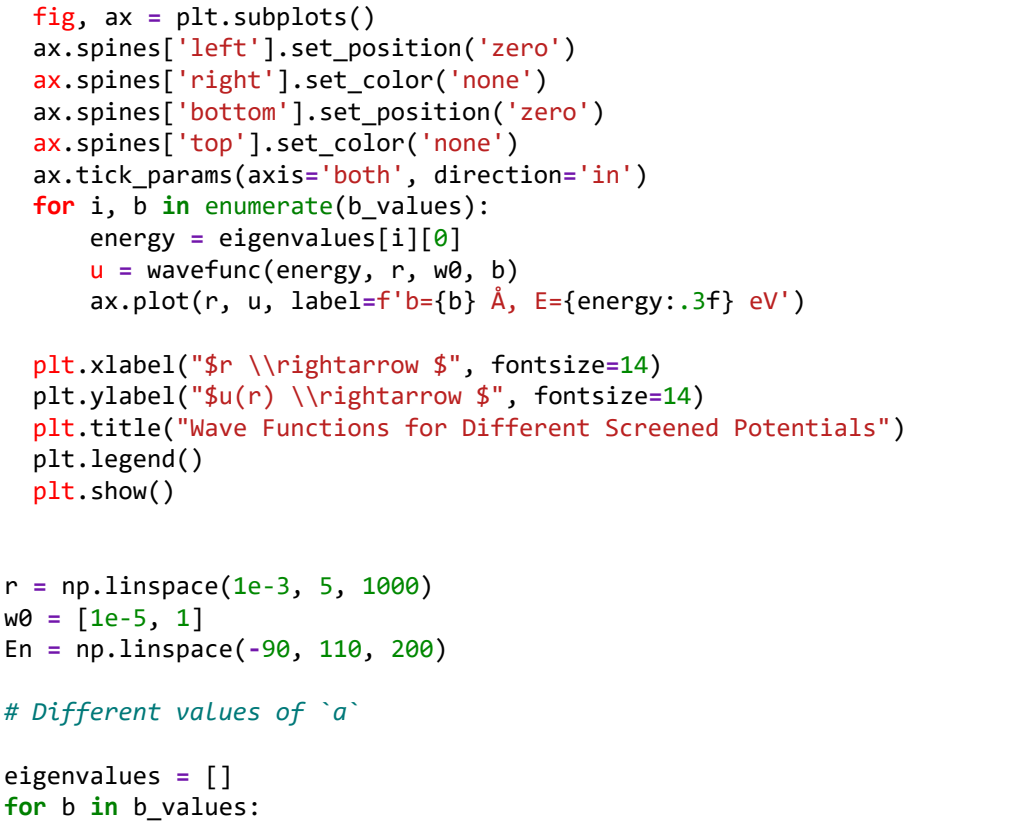
In [46]:
```
e = 3.795
h_barc = 1973
m = 0.511e6

def V(r):
    return -e**2 / r

def A(r, E):
    return (2 * m / h_barc**2) * (V(r) - E)

def model(w, r, E):
    u, v = w
    du_dr = v
    dv_dr = A(r, E) * u
    return [du_dr, dv_dr]

def wavefunc(energy, r, w0):
    sol = odeint(model, w0, r, args=(energy,))
    u = sol[:, 0]
    return u

def eigenval(energy_range, r, w0):
    eigen = []
    for i in range(len(energy_range) - 1):
        u1 = wavefunc(energy_range[i], r, w0)[-1]
        u2 = wavefunc(energy_range[i + 1], r, w0)[-1]
        if np.sign(u1) != np.sign(u2):
            eigen_value = bisect(lambda E: wavefunc(E, r, w0)[-1], energy_range[i], energy_range[i + 1])
            eigen.append(eigen_value)
    return eigen

def plot():
    fig, ax = plt.subplots()
    ax.spines['left'].set_position('zero')
    ax.spines['right'].set_color('none')
    ax.spines['bottom'].set_position('zero')
    ax.spines['top'].set_color('none')
    return fig, ax


def plot1(eigen, r, w0):
    fig, ax = plot()
    for i, en in enumerate(eigen):
        u = wavefunc(en, r, w0)
        ax.plot(r, u, label=f'n={i}')
    plt.xlabel("$r \\rightarrow $", fontsize=14)
    plt.ylabel("$u(r) \\rightarrow $", fontsize=14)
    plt.legend()
    plt.show()


r = np.linspace(1e-3, 5, 1000)
w0 = [1e-5, 1]
En = np.linspace(-20, 0, 200)

eigen = eigenval(En, r, w0)
plot1(eigen, r, w0)
```



In [64]:
```
e = 3.795
h_barc = 1973
m = 0.511e6
a_values = [3, 5, 7]

def V(r, a):
    return -e**2 / r * np.exp(-r / a)

def A(r, E, a):
    return (2 * m / h_barc**2) * (V(r, a) - E)

def model(w, r, E, a):
   u, v = w
   du_dr = v
   dv_dr = A(r, E, a) * u
   return [du_dr, dv_dr]

def wavefunc(energy, r, w0, a):
   sol = odeint(model, w0, r, args=(energy, a))
   u = sol[:, 0]
   return u

def eigenval(energy_range, r, w0, a):
   eigen = []
   for i in range(len(energy_range) - 1):
      u1 = wavefunc(energy_range[i], r, w0, a)[-1]
      u2 = wavefunc(energy_range[i + 1], r, w0, a)[-1]
      if np.sign(u1) != np.sign(u2):
         eigen_value = bisect(lambda E: wavefunc(E, r, w0, a)[-1], energy_range[i], energy_range[i + 1])
         eigen.append(eigen_value)
   return eigen

def plot_wavefunctions_all(eigenvalues, r, w0, a_values):
  fig, ax = plt.subplots()
  ax.spines['left'].set_position('zero')
  ax.spines['right'].set_color('none')
  ax.spines['bottom'].set_position('zero')
  ax.spines['top'].set_color('none')

  for i, a in enumerate(a_values):
      energy = eigenvalues[i][0]
      u = wavefunc(energy, r, w0, a)
      ax.plot(r, u, label=f'a={a} Å, E={energy:.3f} eV')

  plt.xlabel("$r \\rightarrow $", fontsize=14)
  plt.ylabel("$u(r) \\rightarrow $", fontsize=14)
  plt.title("Wave Functions for Different Screened Potentials")
  plt.legend()
  plt.show()


r = np.linspace(1e-3, 5, 1000)
w0 = [1e-5, 1]
En = np.linspace(-20, 0, 200)

# Different values of `a`

eigenvalues = []
for a in a_values:
    eigen = eigenval(En, r, w0, a)
    eigenvalues.append(eigen)


# Plot all wave functions
plot_wavefunctions_all(eigenvalues, r, w0, a_values)
```



In [10]:
```
h_barc = 197.3
m = 940
k = 100
b_values = [0,10, 30]


def V(r, b):
  return 0.5 * k * r**2 + (1 / 3) * b * r**3

def A(r, E, b):
  return (2 * m / h_barc**2) * (V(r, b) - E)

def model(w, r, E, b):
   u, v = w
   du_dr = v
   dv_dr = A(r, E, b) * u
   return [du_dr, dv_dr]

def wavefunc(energy, r, w0, b):
   sol = odeint(model, w0, r, args=(energy, b))
   u = sol[:, 0]
   return u

def eigenval(energy_range, r, w0, b):
   eigen = []
   for i in range(len(energy_range) - 1):
      u1 = wavefunc(energy_range[i], r, w0, b)[-1]
      u2 = wavefunc(energy_range[i + 1], r, w0, b)[-1]
      if np.sign(u1) != np.sign(u2):
         eigen_value = bisect(lambda E: wavefunc(E, r, w0, b)[-1], energy_range[i], energy_range[i + 1])
         eigen.append(eigen_value)
   return eigen

def plot_wavefunctions_all(eigenvalues, r, w0, b_values):
  fig, ax = plt.subplots()
  ax.spines['left'].set_position('zero')
  ax.spines['right'].set_color('none')
  ax.spines['bottom'].set_position('zero')
  ax.spines['top'].set_color('none')
  ax.tick_params(axis='both', direction='in')
  for i, b in enumerate(b_values):
      energy = eigenvalues[i][0]
      u = wavefunc(energy, r, w0, b)
      ax.plot(r, u, label=f'b={b} Å, E={energy:.3f} eV')

  plt.xlabel("$r \\rightarrow $", fontsize=14)
  plt.ylabel("$u(r) \\rightarrow $", fontsize=14)
  plt.title("Wave Functions for Different Screened Potentials")
  plt.legend()
  plt.show()


r = np.linspace(1e-3, 5, 1000)
w0 = [1e-5, 1]
En = np.linspace(-90, 110, 200)

# Different values of `a`

eigenvalues = []
for b in b_values:
    eigen = eigenval(En, r, w0, b)
    eigenvalues.append(eigen)


# Plot all wave functions
plot_wavefunctions_all(eigenvalues, r, w0, b_values)
```



In [8]:
```
hbar_c = 197.3
m = 940e6
D = 0.755501
alpha = 1.44
r0 = 0.131349
mu = m / 2

# Morse Potential Function
def V(r):
    r_prime = (r - r0) / r0
    return D * (np.exp(-2 * alpha * r_prime) - 2 * np.exp(-alpha * r_prime))


def A(r, E):
    return (2 * mu / hbar_c**2) * (V(r) - E)

def model(w, r, E):
    u, v = w
    du_dr = v
    dv_dr = A(r, E) * u
    return [du_dr, dv_dr]

def wavefunc(energy, r, w0):
    sol = odeint(model, w0, r, args=(energy,))
    u = sol[:, 0]
    return u

def eigenval(energy_range, r, w0):
    eigen = []
    for i in range(len(energy_range) - 1):
        u1 = wavefunc(energy_range[i], r, w0)[-1]
        u2 = wavefunc(energy_range[i + 1], r, w0)[-1]
        if np.sign(u1) != np.sign(u2):
            eigen_value = bisect(lambda E: wavefunc(E, r, w0)[-1], energy_range[i], energy_range[i + 1])
            eigen.append(eigen_value)
    return eigen

def plot_wavefunction(energy, r, w0):
    u = wavefunc(energy, r, w0)
    plt.figure()
    plt.plot(r, u, label=f"Energy = {energy:.5f} eV")
    plt.axhline(0, color='red', linewidth=0.8)
    plt.xlabel("$r$ (Å)")
    plt.ylabel("$u(r)$")
    plt.title("Wavefunction of the Lowest Vibrational State")
    plt.legend()
    plt.show()

# Main
r = np.linspace(0.1, 5, 1000)
w0 = [1e-5, 1]
En = np.linspace(0, 1, 200)

eigenvalues = eigenval(En, r, w0)
print(f"Lowest vibrational energy: {eigenvalues[0]:.5f} eV")
plot_wavefunction(eigenvalues[0], r, w0)
```

```
Lowest vibrational energy: 0.00359 eV
```