```
import numpy as np
import matplotlib.pyplot as plt
import scipy.integrate as integrate


def wkb_class(energy, x, x2, D, m, omega, hbar):
    p_x = momentum(x, energy, m, omega)
    integrated_p_x = integrate_px(x, x2, energy, m, omega)
    fun = 2*D/np.sqrt(p_x)*np.sin(1/hbar*integrated_p_x + np.pi/4)
    return fun


def wkb_non_class_pos(energy, x, x2, D, m, omega, hbar):
    p_x = momentum(x, energy, m, omega)
    integrated_p_x = integrate_px(x2, x, energy, m, omega)
    fun = D / np.sqrt(p_x) * np.e ** (-1 / hbar * integrated_p_x)
    return fun


def wkb_non_class_neg(energy, x, x2, D, m, omega, hbar):
    p_x = momentum(x, energy, m, omega)
    integrated_p_x = integrate_px(x, x1, energy, m, omega)
    fun = D / np.sqrt(p_x) * np.e ** (-1 / hbar * integrated_p_x)
    return fun


def exact_solution_0(x, m , omega, hbar):
    alpha = m*omega/hbar
    y = np.sqrt(alpha)*x
    fun = (alpha/np.pi)**(1/4)*np.e**(-(y**2)/2)
    return np.abs(fun)**2


def exact_solution_10(x, m , omega, hbar):
    alpha = np.sqrt(m*omega/hbar)
    prod1 = 1/(2**10 * np.math.factorial(10))
    prod2 = np.sqrt(alpha)*np.sqrt(np.sqrt(np.pi))
    prod3 = np.exp(-alpha**2 * x**2/2)
    prod4 = physicist_herm_pol_10(alpha*x)
    fun = prod1*prod2*prod3*prod4
    return np.abs(fun)**2


def momentum(x, energy, m, omega):
    V = potential(x, m, omega)
    if E > V:
        return np.sqrt(2*m*(energy-V))
    else:
        return np.sqrt(2 * m * (V-energy))


def potential(x, m, omega):
    return (1/2)*m*(omega*x)**2


def integrate_px(x1, x2, energy, m, omega):
    return integrate.quad(momentum, x1, x2, args=(energy, m, omega))[0]


def physicist_herm_pol_10(x):
    return 1024*x**10 - 23040*x**8 + 161280*x**6 - 403200*x**4 + 302400*x**2 -
30240


```

```python
# Define normalization constant D
D = 0.37  # TODO normalize this
const10 = 1*1e9  # Constant for scaling exact_sol_10

# general constants
hbar = 1
omega = 1
m = 1
# Define constants in terms of E:
n = 11
E = (n-1/2)*omega*hbar
# E = V(x1) = V(x2)
x2 = np.sqrt(2*E/(m*omega**2))
x1 = -np.sqrt(2*E/(m*omega**2))

# Define x_range
x = np.linspace(-1, 1, 1000)*x2*2  # Have the interval be twice as wide as 0-x2

# Define regions
x_class = [s for s in x if np.abs(s) < x2]  # Classical region
x_non_class_1 = [s for s in x if s > x2]  # Positive non-classical region
x_non_class_2 = [s for s in x if s < x1]  # Negative non-classical region

# Generate data
wave_class = np.zeros((len(x_class), 1))
for i in range(len(x_class)):
    wave_class[i] = np.abs(wkb_class(E, x_class[i], x2, D, m, omega, hbar))**2

wave_non_class_1 = np.zeros((len(x_non_class_1), 1))
for i in range(len(x_non_class_1)):
    wave_non_class_1[i] = np.abs(wkb_non_class_pos(E, x_non_class_1[i], x2, D, m, omega, hbar))**2

wave_non_class_2 = np.zeros((len(x_non_class_2), 1))
for i in range(len(x_non_class_1)):
    wave_non_class_2[i] = np.abs(wkb_non_class_neg(E, x_non_class_2[i], x2, D, m, omega, hbar))**2


# Define plot
fig, ax = plt.subplots()
# Plot WKB
ax.plot(x_class, wave_class, 'b', label="WKB approximation")
ax.plot(x_non_class_1, wave_non_class_1, 'b')
ax.plot(x_non_class_2, wave_non_class_2, 'b')
# Plot exact solutions
#ax.plot(x, exact_solution_0(x, m, omega, hbar), 'r', label="Exact solution, n=0")
ax.plot(x, exact_solution_10(x, m, omega, hbar)*const10, 'r', label="Exact solution, n=10")
# Plot potential
ax.plot(x, potential(x, m, omega), 'b--', label="Potential", alpha=0.5)
# Plot energy
ax.plot(x, np.ones((len(x),1))*E, 'k--', label="Energy", alpha=0.5)
plt.legend(loc='upper right')
# Plot x and y axis thicker
# ax.axvline(0)
# ax.axhline(0)
plt.ylim([0, D*4])
ax.grid()
plt.title(f'WKB and exact solutions, in arbitrary units. n = {n-1}')
plt.ylabel("Psi^2, dimensionless")
plt.xlabel("x")

plt.savefig(f'wkb_n_{n-1}')
```

```
plt.show()
```