```
################################################################################
### PROMPT: CUBIC SPLINE INTERPOLATION
################################################################################

import numpy as np
from scipy.interpolate import CubicSpline
import matplotlib.pyplot as plt

################################################################################
# Swaption Data
################################################################################

term = [1, 2, 4, 5]             # units are years
volatility = [30, 20, 35, 40]     # units are percentages

################################################################################
# Analysis
################################################################################

# Defining the function for natural cubic spline interpolation
def cubic_spline_interp(x, y):
    """
    Returns an object such that isinstance(object,CubicSpline)
    is true. The cubic spline has the second derivative
    at the curve ends equal to zero.

    x: is an array-like object containing the x elements
    y: is an array-like object containing the y elements
    """
    return CubicSpline(x, y, axis=0, bc_type='natural')

# Calculating and outputting the interpolating functions
interpolatingFunctions = cubic_spline_interp(term, volatility)

# Evaluating the relevant interpolating function at 3-years
result = interpolatingFunctions(3)

# Outputting the response
print(result)

################################################################################
# PLOT
################################################################################

plt.figure(figsize=(14,5))
t = np.linspace(1,5)
v = interpolatingFunctions(t)
plt.plot(t,v,'r')
plt.plot(term, volatility, '.b')
```

```
plt.title('5-Year Payer Swaption Volatility - 25% OTM')
plt.xlabel('Expiration (years)'), plt.ylabel('Implied Volatility')
plt.grid(True)
plt.show()
```