Q1

The function then creates two matrices, `A` and `B`, which will be used to solve for the coefficients of the polynomial. The `A` matrix is a square matrix with dimensions `(n+1)x(n+1)`, and the `B` matrix is a column vector with dimensions `(n+1)x1`. The elements of the `A` and `B` matrices are calculated using the x and y values of the given points.

The function then uses `numpy.linalg.solve` to solve the matrix equation `Ax = B` and obtain the coefficients of the polynomial. The coefficients are stored in a `Polynomial` object, which is created using the `numpy` library.

Finally, the function displays the polynomial using the `Polynomial.show()` method, which takes the minimum and maximum x values of the given points as arguments. The function also plots the given points on a graph using the `matplotlib` library.

The function then returns the `Polynomial` object.

Q2

The first part of the function checks if the input is valid. The `n` input should be an integer, and should be nonnegative. If the input is invalid, the function raises an exception with a helpful error message.

Next, the function defines the interval [0, pi] and generates 100 equally spaced x values within the interval. The corresponding y values for these x values are obtained by evaluating the given function `fun` at each x value.

The function then creates a matrix `A` and a column vector `b`, which will be used to solve for the coefficients of the polynomial. The `A` matrix is a square matrix with dimensions `(n+1)x(n+1)`, and the elements of the matrix are calculated using integration via the `quad` function from the. The `b` vector is a column vector with dimensions `(n+1)x1`, and the elements of the vector are also calculated using integration via the `quad` function.The function then uses `numpy.linalg.solve` to solve the matrix equation `Sx = b` and obtain the coefficients of the polynomial.

Q3

It first constructs the polynomial num as a polynomial expression of the form (x**2 - 1)**n. This is done using the Polynomial class provided in the code.

It then takes the nth derivative of the polynomial using the derivative() method provided by the Polynomial class.

It then divides the resulting polynomial by the factor 2**n * math.factorial(n) to obtain the nth Legendre polynomial.

Finally, the function returns the nth Legendre polynomial as a polynomial object