1. Suppose that $x_0 < x_1 < \cdots < x_n$ and $y_0, y_1, \ldots, y_n$ are given real numbers. Then there exists a unique polynomial $p \in \mathcal{P}_n$ such that $p(x_i) = y_i$ for all $i \in \{0, \ldots, n\}$.

2. Suppose that $x_0 < x_1 < \cdots < x_n$ and $y_0, y_1, \ldots, y_n$ are given real numbers. Then there exists a unique polynomial $p \in \mathcal{P}_n$ such that $p(x_i) = y_i$ for all $i \in \{0, \ldots, n\}$.

3. Given $x_0 < x_1 < \ldots < x_n \in \mathbb{R}$ and $y_0, y_1, \ldots, y_n \in \mathbb{R}$, there exist multiple polynomials $p \in \mathcal{P}_{n+1}$ such that $p(x_i) = y_i$ for all $i \in \{0, 1, \ldots, n\}$.

4. In interpolation, the choice of interpolating points can affect the accuracy of the interpolation.

5. Suppose that $f\colon [-1,1] \to \mathbb{R}$ is the function which to $x$ associates $e^x$, and for any $n \in \mathbb{N}$, let $f_n \in \mathcal{P}_n$ denote the polynomial interpolating $f$ at $n+1$ equidistant points $-1 = x_0 < x_1 < \ldots < x_n = 1$. Then
$$\lim_{n\to\infty} \left( \max_{-1 \le x \le 1} \bigl| f(x) - f_n(x) \bigr| \right) = 0.$$

6. There exists a polynomial $p$ such that
$$\forall n \in \mathbb{N}, \qquad p(n) = 2^n\,.$$

7. Given $x_0 < x_1 < \ldots < x_n \in \mathbb{R}$ and $y_0, y_1, \ldots, y_n \in \mathbb{R}$ with $n = 10$, there exists a unique affine polynomial $p(x) = ax + b$ that minimizes
$$J(a,b) = \frac{1}{2} \sum_{i=1}^{n} \bigl| y_i - p(x_i) \bigr|^2.$$

8. In Julia, if `A` is a matrix, then `A[:, 1:2]` gives the first two rows of `A`.

9. In Julia, if `A` is a matrix, then `A[mod.(1:end, 2) .== 0, mod.(1:end, 2) .== 0]` gives the matrix obtained by removing the second column and the second row.

10. In Julia, typing `;` in a REPL (command line) enables to access package mode, from which new packages can be installed.

11. In the following code, `p` is the interpolating polynomial through the data in `x` and `y`.

```
using Plots
x = [0, 1, 2, 3]
y = [1, 2, 1, 2]

function p(x)
    return (y[1]
            + diff(y)[1] * x
            + 1/2 * diff(diff(y))[1] * x * (x-1)
            + 1/6 * diff(diff(diff(y)))[1] * x * (x-1) * (x-2))
end

plot(p, xlims=(0, 5))
scatter!(x, y)
```