Programming Updates in Maple 2025

Assignment to a Vector

- You can now make an assignment with a Vector on the left-hand side and a Vector of the same size on the right-hand side. This is element-wise assignment.

```
> <x,y,z>:=<2,5,3>:
```

```
> x;
```

$$2 \tag{1}$$

```
> y;
```

$$5 \tag{2}$$

```
> z;
```

$$3 \tag{3}$$

Sort

- The sort command was overhauled in Maple 2025 adding a new general sorting algorithm and introducing several new options. For full examples, see sort. Some of the new features include the following:
- Support indexed names using lexorder and lexorder[n] sorting methods. Previously these would lead to errors.

```
> sort([b,c,c[1],a,a[1],a[a],a[2],a[1,2]],'lexorder');
```

$$\left[a, a_{1}, a_{1,2}, a_{2}, a_{a}, b, c, c_{1}\right] \tag{4}$$

```
> sort([ [ 1, b[2], 1], [2,b,2], [3,b[1],3] ], 'lexorder'[2]);
```

$$\left[\left[2, b, 2\right], \left[3, b_{1}, 3\right], \left[1, b_{2}, 1\right]\right] \tag{5}$$

- New ascending and descending options have been added:

```
> sort([3,1,2],'ascending'=true);
```

$$\left[1, 2, 3\right] \tag{6}$$

```
> sort([3,1,2],'ascending'=false);
```

$$\left[3, 2, 1\right] \tag{7}$$

```
> sort([3,1,2],'descending'=true);
```

(8)

$$[3, 2, 1] \quad \textbf{(8)}$$

```
> sort([3,1,2],'descending'=false);
```

$$[1, 2, 3] \quad \textbf{(9)}$$

- The new option, pass, allows pass-through of additional arguments to comparison function.

This example uses the pass option to allow a third argument to the comparison procedure. Here, the {0,undefined} set will be passed as the third argument in each call to ecmp.

```
> ecmp := proc(a,b,exclude)
    if a in exclude then
        false;
    elif b in exclude then
        true;
    else
        a <= b;
    end if;
  end proc:
> sort( [3,undefined,1,0,2], ecmp, 'pass'={0,undefined});
```

$$[1, 2, 3, 0, undefined] \quad \textbf{(10)}$$

- Support was added to handle datatype-specific complex[8] and complex(sfloat) rtables when using key sort.

Here the key function is applied first, computing the modulus of each element, then the list is sorted by those values.

```
> A := Array( [ 1+I, .5+2*I, I-.9], datatype=complex[8] );
```

$$A := \begin{bmatrix} 1. + I & 0.500000000000000 + 2.\,I & -0.900000000000000 + I \end{bmatrix} \quad \textbf{(11)}$$

```
> sort( A, 'key'=(x->abs(x)) );
```

$$\begin{bmatrix} -0.900000000000000 + I & 1. + I & 0.500000000000000 + 2.\,I \end{bmatrix} \quad \textbf{(12)}$$

```
> map(abs,A);
```

$$\begin{bmatrix} 1.41421356237310 & 2.06155281280883 & 1.34536240470737 \end{bmatrix} \quad \textbf{(13)}$$

- A new comparison algorithm, general, has been added that is especially useful when comparing mixed types. The elements are sorted in the following way:

- numeric values are sorted in ascending numerical order
- symbols and strings are sorted in lexorder (ASCII order)
- indexed names and unevaluated functions are sorted first by the name part, then, if both objects being compared have an index, then the index or argument sequence
- lists, sets, and rtables are compared element-wise as long as there are still elements in the container. The shorter container is considered less-than if it entirely matches the first n-elements of the other container. Multi-dimension rtables are scanned in memory order, which is column-major by default.
- specialized types are sorted by their id-classification, or objectid

```
> sort( [ f(x,-3.1), f(x,2) ] );
```

$$[f(2,2),f(2,-3.1)] \tag{14}$$

```
> sort( [ f(x,-3.1), f(x,2) ], 'general' );
```

$$[f(2,-3.1),f(2,2)] \tag{15}$$

- Both key=keyFunc and a comparison function are now allowed simultaneously in the same call to sort.

Here the key function is applied first, computing the degree of each element, then the list is sorted by those degrees.

```
> sort([x^2,x,x^5],'numeric',key=(x->degree(x)));
```

$$[4,2,32] \tag{16}$$

```
> sort([x^2,x,x^5],'numeric',key=degree,descending);
```

$$[4,2,32] \tag{17}$$

- Basic support for sorting of quantities with mixed units was added.

```
> sort( [ 1*Unit(m),3*Unit(ft),30*Unit(cm)] );
```

$$[30\ \mathrm{cm}, 3\ \mathrm{ft}, \mathrm{m}] \tag{18}$$

- We noticed that sort is often accidentally called with a set as an argument. Given that sets are already in a fixed order that can not be changed, this could lead to surprises in user code. As a result, when a non-default sort algorithm is specified, we now return a sorted list instead of an unsorted set in this case, and introduced a new option, kernelopts(sortsetoutput), to allow enabling of the previous behavior. For details see Calling sort on a Set.

```
> myset := {b[1],a[1],a};
```

$$myset := \{a, a_1, b_1\} \tag{19}$$

```
> sort(myset);
```

$$\{a, a_1, b_1\} \quad (20)$$

```
> sort(myset,'lexorder');
```

$$[a, a_1, b_1] \quad (21)$$

- The efficiency of sort has been improved when using custom comparison functions. For example, the following call is now twice as fast as Maple 2024.

```
> N := 10^4:
> data := LinearAlgebra:-RandomVector(N):
> CodeTools:-Usage(sort(data,(a,b)->a>b)):
memory used=6.02MiB, alloc change=0 bytes, cpu time=63.00ms, real time=62.00ms, gc time=0ns
```

- In order to make the various options and capabilities of sort more clear, the documentation has been updated and reorganized. Additionally the description of sorting of algebraic objects like polynomials has been separated onto its own page, independent of the topic of sorting of data inside lists and arrays.

LaTeX to MathML Converter

- The new MathML:-FromLatex command converts a LaTeX expression to MathML, and optionally further processes it into a Maple expression.

```
> L := "\\cos (2\\theta) = \\cos^2 \\theta - \\sin^2 \\theta":
> mml := MathML:-FromLatex(L);
```

$mml :=$ (22)

```
"<math xmlns="http://www.w3.org/1998/Math/MathML" display=
"inline"><mrow><mi>cos</mi><mo stretchy="false">&#x00028;
</mo><mn>2</mn><mi>&#x003B8;</mi><mo stretchy="false">&#x00029;
</mo><mo>&#x0003D;
</mo><msup><mi>cos</mi><mn>2</mn></msup><mi>&#x003B8;
</mi><mo>&#x02212;
</mo><msup><mi>sin</mi><mn>2</mn></msup><mi>&#x003B8;
</mi></mrow></math>"
```

```
> mml := MathML:-FromLatex(L,output='maple');
```

$$mml := \cos(2\,\theta) = \cos(\theta)^2 - \sin(\theta)^2 \quad (23)$$

Deleting Entries in an Array

- The ArrayTools:-Remove command has been updated to understand empty range endpoints.
- In Maple, you can use shorthand Array indexing syntax. For example, A[5..] means all the entries starting with 5 and going to the end of the data. The ArrayTools:-Remove command now supports this syntax.

```
> A := Array(1..10,i->i);
```

$$A := \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \end{bmatrix} \quad (24)$$

```
> ArrayTools:-Remove(A,5..);
```

$$\begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix} \quad (25)$$

Evaluation Rules for Sum

- As of Maple 2025, the sum command has special evaluation rules. Thus, unevaluation quotes are no longer necessary when summing over certain indexed expressions. For example, previous versions of Maple required single quotes around v[k] in this example. This is no longer necessary.

```
> v := Vector([1,2,3,4,5]);
```

$$v := \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{bmatrix} \quad (26)$$

```
> sum(v[k],k=1..5);
```

$$15 \quad (27)$$

Special Character Entity Names

- The StringTools:-DecodeEntities command now understands a wider list of entity names.

```
> StringTools:-DecodeEntities("&varepsilon;");
```

"ε" **(28)**