Programming Updates in Maple 2023

**ArrayTools**

```maple
> with( ArrayTools ):

SortBy

• The new `ArrayTools:-SortBy` command allows one to easily sort a two-dimensional Array (like a Matrix) by a specific column or row. For example:

```maple
> A := Matrix( [[5,1,3,8,1], [20,18,12,0,22], [15,3,11,15,3], [4,10,19,3,22]] );

$A := \begin{bmatrix}
5 & 1 & 3 & 8 & 1 \\
20 & 18 & 12 & 0 & 22 \\
15 & 3 & 11 & 15 & 3 \\
4 & 10 & 19 & 3 & 22
\end{bmatrix}$

> B := SortBy( A, 'column', 3 );

$B := \begin{bmatrix}
5 & 1 & 3 & 8 & 1 \\
15 & 3 & 11 & 15 & 3 \\
20 & 18 & 12 & 0 & 22 \\
4 & 10 & 19 & 3 & 22
\end{bmatrix}$
```

• Custom sorting options and keywords are supported, as is in-place sorting:

```maple
> SortBy( A, 'row', 2, 'descending', 'inplace' );

> 'A' = A;

$A = \begin{bmatrix}
1 & 5 & 1 & 3 & 8 \\
22 & 20 & 18 & 12 & 0 \\
3 & 15 & 3 & 11 & 15 \\
22 & 4 & 10 & 19 & 3
\end{bmatrix}$
IsSubsequence

• The new ArrayTools:-IsSubsequence command checks if a one-dimensional container is a subsequence of another. More precisely, \( A \) is a subsequence of \( B \) when there is a list of strictly increasing indices \( K \) such that \( A=B[K] \). For example:

\[
\text{IsSubsequence( } <1,3,5>, <1,2,3,4,5> \text{ );}
\]
\[
\text{true}
\]

\[
\text{IsSubsequence( } <1,5,3>, <1,2,3,4,5> \text{ );}
\]
\[
\text{false}
\]

\[
\text{IsSubsequence( } <2,4,6>, <1,2,3,4,5> \text{ );}
\]
\[
\text{false}
\]

• The match option is used to customize how matches are determined. For example, to check a subsequence of floats, one can do the following:

\[
\text{B := Array( } [ \exp(1), \pi, \gamma ] \text{ );}
\]
\[
B := \begin{bmatrix} e & \pi & \gamma \end{bmatrix}
\]

\[
\text{A := evalf[5]( B[[2,3]] );}
\]
\[
A := \begin{bmatrix} 3.1416 & 0.57722 \end{bmatrix}
\]

\[
\text{IsSubsequence( A, B );}
\]
\[
\text{false}
\]

\[
\text{IsSubsequence( A, B, 'match' = 'float' );}
\]
\[
\text{false}
\]

\[
\text{IsSubsequence( A, B, 'match' = 'float', 'digits' = 5, 'ulp' = 1 ) ;}
\]
\[
\text{true}
\]

• The matching indices can also be returned:

\[
\text{A := [ "a", "e", "i", "o", "u" ] ;}
\]
\[
A := ["a", "e", "i", "o", "u"]
\]

\[
\text{B := [ seq( "a" .. "z" ) ] ;}
\]
\[
B := ["a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m", "n", "o", "p", "q", "r", "s", "t", "u", "v", "w", "x", "y", "z"]
\]

\[
\text{K := IsSubsequence( A, B, 'output' = 'indices' ) ;}
\]
\[
K := [1, 5, 9, 15, 21]
\]

\[
\text{EqualEntries( A, B[K] ) ;}
\]
\[
\text{true}
\]
ColorTools

- The ColorTools package now supports the CAM02 color spaces JCh (cylindrical) and Jab (rectangular). These color spaces are now the most modern perceptual color spaces in Maple. JCh is useful for making perceptually uniform changes to lightness (J), chromaticity/saturation (C), and hue (h).

```maple
> ColorTools:-Color("JCh", "#9f1");
JCh : 0.852 0.926 0.353
```

- Jab is useful as a space where Euclidean distance is roughly the same as perceptual distance. These different applications are the reason why the J channel is not the same in the two spaces. ColorTools:-Gradient now uses Jab for its best mode.

```maple
> ColorTools:-Color("Jab", "#9f1");
Jab : 0.907 -0.245 0.324
```

- The YUV rectangular color space is now supported, using exactly the same implementation as in ImageTools:-RGBtoYUV and ImageTools:-YUVtoRGB. The Y channel is lightness, while U and V are rectangular coordinates of chromacity.

```maple
> ColorTools:-Color("YUV", "#9f1");
YUV : 0.774 -0.348 -0.153
```

- The HSL cylindrical color space is now supported using a simple conversion from the existing HSV color space. The H is always the same between the two but the S saturation channel differs.

```maple
> ColorTools:-Color("HSL", "#9f1");
HSL : 0.238 1 0.533
```

```maple
> ColorTools:-Color("HSV", "#9f1");
HSV : 0.238 0.933 1
```

- A new grayscale color space is now supported called Gs. Colors are converted to grayscale by calculating their perceptual lightness.

```maple
> ColorTools:-Color("Gs", "Red");
Gs : 0.532
```

- Colors do not roundtrip convert due to loss of information about hue and chroma but all true shades of gray do.

```maple
> ColorTools:-Color("RGB", ColorTools:-Color("Gs", "#7e7e7e")) =
ColorTools:-Color("RGB", "#7e7e7e");
RGB : 0.494 0.494 0.494 = RGB : 0.494 0.494 0.494
```

- A new conversion color has been added to convert so that convert/color can be used to convert easily between supported color formats.
> convert(ColorTools:-Color("Jab", "#9f1"), 'color', "hex");

    "#99FF11"

and between color spaces

> convert(ColorTools:-Color("Jab", "#9f1"), 'color', "HSL");

    HSL : 0.238 1 0.533

> convert([0.6, 1.0, 0.066666667], 'color', "RGB", "YUV");

    [0.774000000038000, −0.348133333188000, −0.152666667000000]

> convert([0, 128, 0], 'color', "colorname");

    "Green"

- The `ColorTools:-Swatches` command now has two new options `mode` and `filter`. The mode option applies one of several preset styles. The filter option applies a procedure to each color before displaying it.

> ColorTools:-Swatches("Colorwheel", 'mode'='wheel', 'filter'=
    (c->ColorTools:-CVDSimulation(c,"tritanomaly")));

![Colorwheel with tritanomaly simulation]
ListTools

- Two new related commands, `ListTools:-InversePermutation` and `ListTools:-Unpermute`, determine, respectively, the inverse of a permutation given the forward permutation, and an unpermuted list given the permuted list and forward permutation. For example:

```maple
> with(ListTools):
> X := [seq("a".."j")]; # unpermuted list
   X := ["a", "b", "c", "d", "e", "f", "g", "h", "i", "j"]
> P := [4,6,3,9,8,10,1,7,2,5]; # forward permutation
   P := [4, 6, 3, 9, 8, 10, 1, 7, 2, 5]
> Y := X[P]; # permuted list
   Y := ["d", "f", "c", "i", "h", "j", "a", "g", "b", "c"]
> Q := InversePermutation(P); # inverse permutation
   Q := [7, 9, 3, 1, 10, 2, 8, 5, 4, 6]
> Y[Q]; # unpermuted list
   ["a", "b", "c", "d", "e", "f", "g", "h", "i", "j"]
> Unpermute(Y,P); # unpermuted list
   ["a", "b", "c", "d", "e", "f", "g", "h", "i", "j"]
```

membertype

- The `membertype(T,S)` function determines if there is an operand within the expression S of type T. In Maple 2023 `membertype` has been extended to support a wider variety of base expressions, S, to scan. It now supports products, sums, and rtables. For example:

```maple
> membertype( integer, 97*x+4 );
   true

> membertype( integer, Matrix([[1,2],[3,4]]));
   true
```
New inverse functions for Box objects

- For Maple 2023, `RealBox` and `ComplexBox` objects now support the inverse circular functions `arcsec`, `arccsc`, and `arccot`, as well as the inverse hyperbolic functions `arcsech`, `arccsch`, and `arccoth`.

```maple
> arcsec( RealBox( 4.7 ) );
(RealBox: 1.35639 ± 2.7043e-10)

> arcsech( RealBox( 0.23 ) );
(RealBox: 2.14933 ± 4.1295e-10)

> arccot( ComplexBox( 4.3*I - 0.33 ) );
(ComplexBox: [3.12285 +/- 4.38103e-10] + [-0.235372 +/- 2.71693e-10]*I)
```

- For more information, see `RealBox` and `ComplexBox`.

New Units prefixes

- In November, 2022, the 27th meeting of the General Conference on Weights and Measures approved the new prefixes quetta-, ronna-, ronto-, and quecto- for SI units, meaning factors of $10^{30}$, $10^{27}$, $10^{-30}$, and $10^{-27}$, respectively. These prefixes are fully integrated into Maple 2023. We can express the Earth's mass in ronnagrams and the Sun's in quettagrams.

```maple
restart;
> with(ScientificConstants):
> m__earth := Constant(M[Earth], units):
> m__earth := convert(evalf(m__earth), units, ronnagrams);

m
earth := 5.972190000 Rg

> m__sun := Constant(M[Sun], units):
> m__sun := convert(evalf(m__sun), units, Qg);

m
sun := 1990.000000 Qg
```

- The average (Newtonian) gravitational force between the Earth and the Sun is most compactly expressed in zettanewtons. (The prefix zetta-, meaning a factor of $10^{21}$, was introduced in 1991.)

```maple
> G := evalf(Constant(G, units));

G := 6.67408 #10K11
m
3
kg
s
2
```
The typical (Newtonian) gravitational force between two cups of tea, one on Earth and one on the moon, can be computed as follows.

- The gravitational force between two cups of tea, one on Earth and one on the moon, can be computed as follows.

```
> volume__tea := Unit(cup);
  volume\_tea := cup

> density__tea := ThermophysicalData:-Property("density", "water", T=95*Unit(Celsius), pressure=Unit(atm));
  density\_tea := 961.8879166 \( \text{kg m}^{-3} \)

> mass__tea := volume__tea * density__tea;
  mass\_tea := 961.8879166 \( \text{cup \ kg m}^{-3} \)

> distance__moon := 384400*Unit(km);
  distance\_moon := 384400 \( \text{km} \)

> force__tea_tea := convert(G * mass__tea^2 / distance__moon^2, units, qN);
  force\_tea\_tea := 23.39159750 \( \text{qN} \)
```

- We can express the Planck length in quectometers.

```
> planck := Constant(l[P], units):
> convert(evalf(planck), units, quectometer);
  0.00001616228373 \( \text{qm} \)
```