Language & Programming Updates in Maple 2022

ArrayTools

```
> with( ArrayTools ):
```

Lookup

- The new Lookup command searches for a specified value in a 1-D container or row/column of a 2-D container, and for all the matches, looks up and returns the corresponding values in another 1-D container or row/column of the same 2-D container. Various options provide flexibility as to what registers as a match.
- The default type of match is exact, and for a Matrix or 2-D Array, the default is to search for matches in column 2, and look up the corresponding values for matches in column 1:

```
> A := Matrix( [["a",1,2,3],["b",3,1,2],["c",2,3,1]] );
```

$$A := \begin{bmatrix} \text{"a"} & 1 & 2 & 3 \\ \text{"b"} & 3 & 1 & 2 \\ \text{"c"} & 2 & 3 & 1 \end{bmatrix}$$

```
> Lookup( 2, A );
```

$$\text{"c"}$$

```
> Lookup( 2, A, 'column', 4, 1 );
```

$$\text{"b"}$$

- When floats are involved, floating-point comparisons can be used (and tolerance can be modified with the digits, ulps, and relativeerror options):

```
> A := [ sqrt(2), evalf(sqrt(2)), sqrt(3), evalf(sqrt(3)) ];
```

$$A := \left[\sqrt{2}, 1.414213562, \sqrt{3}, 1.732050808\right]$$

```
> B := [ 10, 20, 30, 40 ];
```

$$B := [10, 20, 30, 40]$$

```
> Lookup( sqrt(2), A, B, 'match' = 'exact' );
```

$$10$$

```
> Lookup( sqrt(2), A, B, 'match' = 'float' );
```

$$10, 20$$

- Regular expressions and wildcards can be used to match strings:

```
> A := [ "ab", "abc", "ad", "abbc", "a2c", "aBc" ];
```

$$A := [\text{"ab"}, \text{"abc"}, \text{"ad"}, \text{"abbc"}, \text{"a2c"}, \text{"aBc"}]$$

```
> B := [ 10, 20, 30, 40, 50, 60 ];
```

$$B := [10, 20, 30, 40, 50, 60]$$

```
> Lookup( "(^)(a)([a-z]+)(c)($)", A, B, 'match' = 'regexp' );
```

20, 40

```
> Lookup( "a*c", A, B, 'match' = 'wildcard' );
```

20, 40, 50, 60

- Searches, by default, are performed in the forward direction, but can also be performed in reverse:

```
> Lookup( "a*c", A, B, 'match' = 'wildcard', 'direction' =
  'reverse' );
```

60, 50, 40, 20

- Furthermore, we can request that results be restricted to a certain quantity or even a range:

```
> Lookup( "a*c", A, B, 'match' = 'wildcard', 'numresults' = 2 );
```

20, 40

```
> Lookup( "a*c", A, B, 'match' = 'wildcard', 'numresults' = 0 .. 2
  );
```

20, 40

- DataFrames are also supported with the default for search being column 1 and lookup being in the row labels. Consider this DataFrame with batting statistics for the 2021 Toronto Blue Jay hitters with 100 or more at bats:

```
> Data := Matrix([[0.242,165,19,40,8,0,8,24,19,0.328,0.436],[0.298,
  640,121,191,30,1,29,102,40,0.343,0.484],[0.224,250,27,56,10,1,7,
  27,37,0.322,0.356],[0.282,131,16,37,6,2,4,15,9,0.329,0.45],
  [0.223,184,32,41,13,0,11,28,17,0.299,0.473],[0.264,299,59,79,19,
  1,22,50,37,0.352,0.555],[0.246,114,9,28,6,0,2,11,8,0.293,0.351],
  [0.276,500,62,138,28,2,21,84,32,0.319,0.466],[0.265,652,115,173,
  39,2,45,102,66,0.334,0.538],[0.241,511,59,123,25,1,22,81,27,
  0.281,0.423],[20.53,198,22,50,15,0,1,10,15,0.31,0.343],[0.209,
  139,12,29,4,1,4,8,9,0.272,0.338],[0.311,222,32,69,13,1,2,17,22,
  0.376,0.405],[0.296,550,92,163,29,0,32,116,36,0.346,0.524],
  [0.311,604,123,188,29,1,48,111,86,0.401,0.601]]):
> Players := ["Alejandro Kirk","Bo Bichette","Cavan Biggio","Corey
  Dickerson","Danny Jansen","George Springer","Joe Panik","Lourdes
  Gurriel Jr","Marcus Semien","Randal Grichuk","Reese McGuire",
  "Rowdy Tellez","Santiago Espinal","Teoscar Hernandez","Vladimir
  Guerrero Jr"]:
> Categories := ["AVG","AB","R","H","2B","3B","HR","RBI","BB",
  "OBP","SLG"]:
```

```
> DF := DataFrame( Data, 'rows' = Players, 'columns' = Categories )
  ;
```

DF :=

	"AVG"	"AB"	"R"	"H"	"2B"	"3B"	"HR"	"RBI"	"BB"	"OBP"	…
"Alejandro Kirk"	0.242	165	19	40	8	0	8	24	19	0.328	…
"Bo Bichette"	0.298	640	121	191	30	1	29	102	40	0.343	…
"Cavan Biggio"	0.224	250	27	56	10	1	7	27	37	0.322	…
"Corey Dickerson"	0.282	131	16	37	6	2	4	15	9	0.329	…
"Danny Jansen"	0.223	184	32	41	13	0	11	28	17	0.299	…
"George Springer"	0.264	299	59	79	19	1	22	50	37	0.352	…
"Joe Panik"	0.246	114	9	28	6	0	2	11	8	0.293	…
"Lourdes Gurriel Jr"	0.276	500	62	138	28	2	21	84	32	0.319	…
"Marcus Semien"	0.265	652	115	173	39	2	45	102	66	0.334	…
"Randal Grichuk"	0.241	511	59	123	25	1	22	81	27	0.281	…
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	

15 x 11 DataFrame

- To determine the players with 30 or more home runs or 100 or more RBIs, we can pass a custom matching procedure:

```
> Lookup( 30, DF, "HR", 'match' = `>=` );
```

"Marcus Semien", "Teoscar Hernandez", "Vladimir Guerrero Jr"

```
> Lookup( 100, DF, "RBI", 'match' = `>=` );
```

"Bo Bichette", "Marcus Semien", "Teoscar Hernandez", "Vladimir Guerrero Jr"

Reverse

- The Reverse command has been optimized to run using compiled code, and now executes much more quickly. For more information, see the performance update page.

```
> A := LinearAlgebra:-RandomVector( 10^5 );
```

$$A := \begin{bmatrix} -98 \\ 41 \\ 46 \\ 30 \\ -6 \\ 68 \\ -29 \\ -59 \\ 25 \\ -94 \\ \vdots \end{bmatrix}$$

100000 element Vector[column]

```
> CodeTools:-Usage( ArrayTools:-Reverse( A ), 'iterations' = 25 );
```

```
memory used=0.76MiB, alloc change=17.61MiB, cpu time=640.00us, real time=640.00us, gc time=0ns
```

$$\begin{bmatrix} 67 \\ -31 \\ 92 \\ 44 \\ 29 \\ 99 \\ 69 \\ 8 \\ 27 \\ -4 \\ \vdots \end{bmatrix}$$

100000 element Vector[column]

The MultivariatePowerSeries Package

- The MultivariatePowerSeries package was added for Maple 2021. It received a number of updates for 2022.

```
> with(MultivariatePowerSeries):
```

Creating PowerSeries Objects from Arbitrary Expressions

- PowerSeries objects can now be created from arbitrary expressions, not just polynomials--as long as the expression has a power series representation at the origin in all of the variables, and that representation can be found by the series command.

```
> ps_exp := PowerSeries(exp(x));
```

$$ps\_exp := \left[\text{PowerSeries of } e^{x} : 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24} + \frac{x^5}{120} + \ldots\right]$$

```
> Truncate(ps_exp, 3);
```

$$1 + x + \frac{1}{2}x^2 + \frac{1}{6}x^3$$

```
> ps_sin := PowerSeries(sin(x + y - 2*y*z));
```

$$ps\_sin := \left[\text{PowerSeries of } -2\sin\ldots\ldots\cos(y) + \ldots - \cos(x)\sin(y) : x + y - 2yz - \frac{xy^2}{2} - \frac{x^3}{6} - \frac{y^3}{6} - \frac{x^2y}{2} + x^2yz + 2xzy^2 + zy^3 + \frac{xy^4}{24} - 2xy^2z^2 + \frac{x^3y^2}{12} + \frac{x^5}{120} + \frac{y^5}{120} - 2z^2y^3 + \frac{x^2y^3}{12} + \frac{x^4y}{24} + \ldots\right]$$

```
> Truncate(ps_sin, 3);
```

$$x + y - 2yz - \frac{1}{2}xy^2 - \frac{1}{6}x^3 - \frac{1}{6}y^3 - \frac{1}{2}x^2y$$

Substituting into a PowerSeries Object

- The Substitute command can substitute a power series (or a polynomial) into a PowerSeries object. This works only if one or both of the following conditions are satisfied: the substituted power series is *not* invertible, or the power series that we substitute into knows its analytic expression.

```
> ps_exp_2 := Substitute(x = PowerSeries(2*x/(1-y)), ps_exp);
```

$$ps\_exp\_2 := \left[\text{PowerSeries of } e^{-\frac{2x}{-1+y}} : 1 + \ldots\right]$$

```
> Truncate(ps_exp_2, 3);
```

$$1 + 2x + 2x^2 + 2xy + \frac{4}{3}x^3 + 4x^2y + 2xy^2$$

```
> ps_exp_3 := Substitute(x = ps_exp, ps_exp);
```

$$ps\_exp\_3 := \left[\text{PowerSeries of } e^{e^x} : e + \ldots\right]$$

```
> Truncate(ps_exp_3, 3);
```

$$e + ex + ex^2 + \frac{5ex^3}{6}$$

- A Taylor shift is a substitution of $x + c$ for x, where x is a variable and c is a constant. There is a separate command for such substitutions:

```
> ps_exp_4 := TaylorShift(ps_exp, x = Pi*I/2);
```

$$ps\_exp\_4 := \left[\text{PowerSeries of } e^{x + \frac{I\pi}{2}} : I + Ix + \frac{I}{2}x^2 + \frac{I}{6}x^3 + \frac{I}{24}x^4 + \frac{I}{120}x^5 + \ldots\right]$$

```
> Truncate(ps_exp_4, 3);
```

$$I + Ix + \frac{Ix^2}{2} + \frac{Ix^3}{6}$$

Additional Updates

Type foreign

- The data type foreign was added, for so-called "foreign DAGs". These are used to encapsulate some data structures that do not originate in Maple itself, but in other programs or libraries. For example, data structures coming from Python that are not automatically translated to their Maple equivalents. This also applies to some internals of the RealBox and ComplexBox objects, which are also new in 2022.

```
> dictionary := Python:-EvalString("{}");
```

$$dictionary := \text{"<Python object: \{\}>"}$$

```
> type(dictionary, foreign);
```

$$true$$

The LinearAlgebra Package

- The CompressedSparseForm command in the LinearAlgebra package received some new options to fine-tune the result, called length0, includediagonal, and structuralsymmetry. They force the command to include certain entries in the output, even if the corresponding matrix entry is 0.

```
> with(LinearAlgebra):
> A := Matrix([[1, 2, 0], [2, 0, 3], [0, 0, -1]], datatype=integer
  [4]):
```

- By default, only the five nonzero entries in A are included in the output.

```
> CompressedSparseForm(A);
```

$$\left[\begin{bmatrix}1\\3\\4\\6\end{bmatrix},\begin{bmatrix}1\\2\\1\\2\\3\end{bmatrix},\begin{bmatrix}1\\2\\2\\3\\-1\end{bmatrix}\right]$$

- When we specify the includediagonal option, any zero entries on the diagonal are also included; in this case, $A_{2,2}$. The structuralsymmetry option ensures that whenever $A_{i,j}$ is included, so is $A_{j,i}$. For this matrix, that means that $A_{3,2}$ is included, even though it is zero.

```
> CompressedSparseForm(A, includediagonal, structuralsymmetry);
```

$$\left[\begin{bmatrix}1\\3\\6\\8\end{bmatrix},\begin{bmatrix}1\\2\\1\\2\\3\\2\\3\end{bmatrix},\begin{bmatrix}1\\2\\2\\0\\0\\3\\-1\end{bmatrix}\right]$$

Initializing Paired Local Declarations

- When declaring a local variable inside a procedure, you have long been able to also assign each variable a value if you choose. New in Maple 2022 is the ability to assign a sequence of local variables to the output of a single function call.

```
> split := proc( i )
      local (p, np) := selectremove(isprime,i);
      [p,np];
  end proc:
> split([1,2,3,4,5,6,7]);
```

$$[[2, 3, 5, 7], [1, 4, 6]]$$