5. Lists

2016-02-17

```
$Version
```

```
10.0 for Mac OS X x86 (64-bit) (December 4, 2014)
```

Any system doing computer algebra or symbolic manipulations is based on list processing. So it is very important to know how lists are structured and how they may be manipulated.

5.1 Some Simple List Operations

```
Clear[a, b, c, d, l0, l1, l2]
```

```
l0 = {3,5,1}
```

```
{3, 5, 1}
```

```
l1 = {a,b,c,d}
```

```
{a, b, c, d}
```

```
l2 = l1^2
```

$\{a^2, b^2, c^2, d^2\}$

An operation applied to a list leads to a new list obtained by applying the operation to each element of the old lists.

```
l1 + l2
```

$\{a + a^2, b + b^2, c + c^2, d + d^2\}$

```
l0 + l1
```

```
{3, 5, 1} + {a, b, c, d}
```

Thread::tdlen : "Objects of unequal length in {3, 5, 1} + {a, b, c, d} cannot be combined."

```
{3, 5, 1} + {a, b, c, d}
```

```
l1 * l2
```

$\{a^3, b^3, c^3, d^3\}$

```
l1 / l2
```

$\left\{\frac{1}{a}, \frac{1}{b}, \frac{1}{c}, \frac{1}{d}\right\}$

```
Exp[l0]
```

$\{e^3, e^5, e\}$

```
Exp[l0]//N
```

```
{20.0855, 148.413, 2.71828}
```

Output of operators as **Solve[], DSolve[], NDSolve[], FindRoot[]** is often in the form of lists.

```
p = x^4 - 1 + x
```

$-1 + x + x^4$

```
sp = Solve[p == 0., x]
```

```
{{x → -1.22074}, {x → 0.248126 - 1.03398 i},
 {x → 0.248126 + 1.03398 i}, {x → 0.724492}}
```

```
p /. sp
```

{8.88178 × 10^-16, 1.11022 × 10^-16 - 2.22045 × 10^-16 i,
1.11022 × 10^-16 + 2.22045 × 10^-16 i, -1.66533 × 10^-16}

```
Chop[%]
```

{0, 0, 0, 0}

Elements of lists may be again lists:

```
l4 = {a, b, c, d, {al, be, ga}, e}
```

{a, b, c, d, {al, be, ga}, e}

Length[*list*]	The length (= number of elements) of *list*

```
Length[l4]
```

6

5.2 Generating Lists

Lists are either generated by writing the elements within curly brackets; or by use of the command **Table**. Many commands, e.g. **Solve, NSolve, DSolve, NDSolve** render their results as lists.

```
l1 = {a,b,c}
```

{a, b, c}

```
l2 = {{a11, a12, a13}, {a21, a22, a23}, {a31, a32, a33}}
```

{{a11, a12, a13}, {a21, a22, a23}, {a31, a32, a33}}

```
TableForm[l2]
```

a11	a12	a13
a21	a22	a23
a31	a32	a33

```
MatrixForm[l2]
```

$$\begin{pmatrix} a11 & a12 & a13 \\ a21 & a22 & a23 \\ a31 & a32 & a33 \end{pmatrix}$$

```
Transpose[l2]
```

{{a11, a21, a31}, {a12, a22, a32}, {a13, a23, a33}}

```
TableForm[%]
```

a11	a21	a31
a12	a22	a32
a13	a23	a33

A representation of the ε-Tensor ε_{ijk}:

```
epst = {{{0,0,0},{0,0,1},{0,-1,0}},
        {{0,0,-1},{0,0,0},{1,0,0}},
        {{0,1,0},{-1,0,0},{0,0,0}}};
```

```
TableForm[%]
```

0 0 0	0 0 1	0 -1 0
0 0 -1	0 0 0	1 0 0
0 1 0	-1 0 0	0 0 0

```
Transpose[epst];
```

```
TableForm[%]
```

0	0	0
0	0	1
0	-1	0
0	0	-1
0	0	0
1	0	0
0	1	0
-1	0	0
0	0	0

The most important command to generate lists is the command **Table[f, {}]**

Table[f, {imax}]	give a list of **imax** values of **f**
Table[f, {i,imax}]	give a list of values of **f** as **i** runs from **1** to **imax**
Table[f, {i,imin,imax}]	give a list of values of with **i** running from **imin** to **imax**
Table[f, {i,imin,imax,di}]	use steps **di**
Table[f, {i,imin,imax},{j,jmin,jmax}]	generate a multidimensional table
TableForm[*list*]	display a list in tabular form
MatrixForm[*list*]	display a suitable list in matrix form

The running variables **i,j, …** need not be integers !

```
Table[i^2, {9}]
```

$\{i^2, i^2, i^2, i^2, i^2, i^2, i^2, i^2, i^2\}$

```
Table[i^2, {i,9}]
```

{1, 4, 9, 16, 25, 36, 49, 64, 81}

```
Table[ Exp[I x], {x,0,Pi,Pi/5} ]
```

$\left\{1, e^{\frac{i\pi}{5}}, e^{\frac{2i\pi}{5}}, e^{\frac{3i\pi}{5}}, e^{\frac{4i\pi}{5}}, -1\right\}$

```
Table[ Sin[x], {x,0,Pi,Pi/5} ]
```

$\left\{0, \sqrt{\frac{5}{8}-\frac{\sqrt{5}}{8}}, \sqrt{\frac{5}{8}+\frac{\sqrt{5}}{8}}, \sqrt{\frac{5}{8}+\frac{\sqrt{5}}{8}}, \sqrt{\frac{5}{8}-\frac{\sqrt{5}}{8}}, 0\right\}$

```
N[%]
```

{0., 0.587785, 0.951057, 0.951057, 0.587785, 0.}

```
Table[i j/k , {i,3}, {j,2}, {k,4}]
```

$\left\{\left\{\left\{1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}\right\}, \left\{2, 1, \frac{2}{3}, \frac{1}{2}\right\}\right\}, \left\{\left\{2, 1, \frac{2}{3}, \frac{1}{2}\right\}, \left\{4, 2, \frac{4}{3}, 1\right\}\right\}, \left\{\left\{3, \frac{3}{2}, 1, \frac{3}{4}\right\}, \left\{6, 3, 2, \frac{3}{2}\right\}\right\}\right\}$

Range[{nmax}]	give a list {1,2,3, ..., nmax}
Range[{nmin,nmax}]	give a list {nmin, nmin+1, ..., ≤ nmax}
Range[{nmin,nmax,di}]	give a list {nmin, nmin+di,nmin+2di, ..., ≤ nmax}

```
Range[5]
```

{1, 2, 3, 4, 5}

```
Range[-1, -5]
```

{}

```
-Range[1, 5] // Reverse
```

{-5, -4, -3, -2, -1}

```
Range[1 / 2, 9 / 2]
```

$\left\{\frac{1}{2}, \frac{3}{2}, \frac{5}{2}, \frac{7}{2}, \frac{9}{2}\right\}$

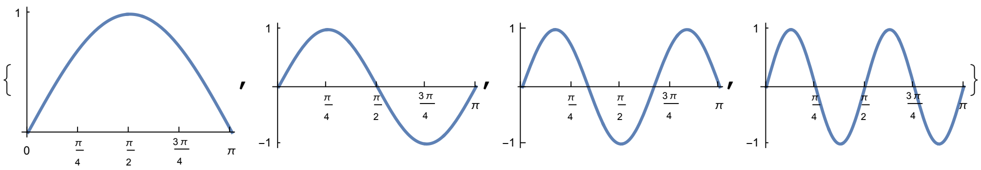
```
Range[1 / 2, 10]
```

$\left\{\frac{1}{2}, \frac{3}{2}, \frac{5}{2}, \frac{7}{2}, \frac{9}{2}, \frac{11}{2}, \frac{13}{2}, \frac{15}{2}, \frac{17}{2}, \frac{19}{2}\right\}$

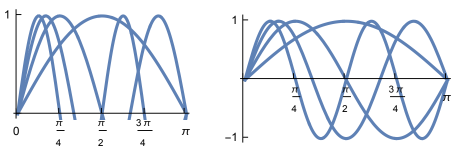
```
Range[1 / 2, 5, 1 / 3]
```

$\left\{\frac{1}{2}, \frac{5}{6}, \frac{7}{6}, \frac{3}{2}, \frac{11}{6}, \frac{13}{6}, \frac{5}{2}, \frac{17}{6}, \frac{19}{6}, \frac{7}{2}, \frac{23}{6}, \frac{25}{6}, \frac{9}{2}, \frac{29}{6}\right\}$

```
Table[Plot[Sin[n x], {x, 0, π}, ImageSize → 115,
  Ticks → {π Range[0, 1, 1 / 4] , {-1, 0, 1}}, BaseStyle → {FontSize → 6}], {n, 4}]
```



```
GraphicsRow[{Show[%], Show[Reverse[%]]}]
```



5.3 Getting List Elements

Part[*list*, *n*]	Get element ***n*** of *list*
list[[*n*]]	Get element ***n*** of *list*
Part[*list*, -*n*]	Get element ***n*** of *list* counted from the end
Part[*list*, {*n1*,*n2*,*n3*, ...}]	Get elements ***n1***,***n2***,***n3***,... of *list*
list[[{*n1*,*n2*,*n3*, ...}]]	Get elements ***n1***,***n2***,***n3***,... of *list*
First[*list*]	Get first element of *list*
Last[*list*]	Get last element of *list*
Part[*list*, **Range**[*n1*,*n2*]]	Get elements from ***n1*** to ***n2*** of *list*
list[[**Range**[*n1*,*n2*]]]	Get elements from ***n1*** to ***n2*** of *list*

```
li = Range[1, 8] // Reverse
```

{8, 7, 6, 5, 4, 3, 2, 1}

```
Part[li, 3]
```

6

```
Part[li, -3]
```

3

```
li[[1]]
```

8

```
First[li]
```

8

```
Last[li]
```

1

```
li[[{2, 4, 6}]]
```

{7, 5, 3}

```
li[[Range[2, 5]]]
```

{7, 6, 5, 4}

```
so = Solve[x + a == 0, x]
```

{{x → -a}}

```
Flatten[so]
```

{x → -a}

```
so[[1]]
```

{x → -a}

```
x /. so
```

{-a}

```
First[%]
```

-a

Take[*list*, n**]**	the first n elements of *list*
Take[*list*,-n**]**	the last n elements of *list*
Take[*list*, {k,n}**]**	*list* with elements k to n (inclusive)
Rest[*list***]**	*list* with its first element dropped
Drop[*list*,n**]**	*list* with its first n elements dropped
Drop[*list*,-n**]**	*list* with its last n elements dropped
Drop[*list*,{k,n}**]**	*list* with elements k to n dropped

```
Clear[a11, a12, a13, a21, a22, a23, a31, a32, a33]
l2 = {{a11, a12, a13}, {a21, a22, a23}, {a31, a32, a33}}
```

{{a11, a12, a13}, {a21, a22, a23}, {a31, a32, a33}}

```
Take[l2,2]
```

{{a11, a12, a13}, {a21, a22, a23}}

```
Take[l2,-3]
```

{{a11, a12, a13}, {a21, a22, a23}, {a31, a32, a33}}

```
Rest[l2]
```

{{a21, a22, a23}, {a31, a32, a33}}

```
Take[l2,{2,3}]
```

{{a21, a22, a23}, {a31, a32, a33}}

```
Drop[l2,-2]
```

{{a11, a12, a13}}

Extract[*expr*, *list*] extracts the part of *expr* at the position specified by *list*.
Extract[*expr*, {*list*$_1$, *list*$_2$, … }] extracts a list of parts of *expr*.
Extract[*expr*, … , *h*] extracts parts of *expr*, wrapping each of them with head *h* before evaluation.

Extract[expr, {i, j, … }] is equivalent to **Part[expr, i, j, …].**
The position specifications used by **Extract[]** have the same form as those returned by **Position[]**, and used in functions such as **MapAt[]** and **ReplacePart[]**.

Below the sublists of **li** containing the number 4 are extracted from **li.**

```
li = {{3, 5, 9}, 6, {2, 3, 4, 5, 9}, {1, 4, 9}, 7};
```

```
pos4 = Position[li, 4]
```

{{3, 3}, {4, 2}}

```
Map[Drop[#, -1] &, pos4]
```

{{3}, {4}}

```
Extract[li, %]
```

{{2, 3, 4, 5, 9}, {1, 4, 9}}

Below those terms of the integrated functions are extracted which contain a logarithm.

```
int = Integrate[x (x + 2)^2 / ((x - 1) (x + 3)), x]
```

$$2\,x + \frac{x^2}{2} + \frac{9}{4}\,\text{Log}[1 - x] + \frac{3}{4}\,\text{Log}[3 + x]$$

```
polog = Position[int, Log]
```

{{3, 2, 0}, {4, 2, 0}}

```
Map[Drop[#, {-2, -1}] &, polog]
```

{{3}, {4}}

```
Extract[int, %]
```

$$\left\{\frac{9}{4}\,\text{Log}[1 - x],\ \frac{3}{4}\,\text{Log}[3 + x]\right\}$$

```
poly = c^3 + a x^3 + 3 x y + b^4 y^3;
```

```
powerPositions = Position[poly, y_^n_]
```

{{1}, {2, 2}, {4, 1}, {4, 2}}

```
Extract[poly, powerPositions]
```

$$\{c^3, x^3, b^4, y^3\}$$

5.4 Adding, Removing, Searching for and Modifying List Elements

Prepend[*list,element*]	add *element* at the beginning of *list*
Append[*list,element*]	add *element* at the end of *list*
Insert[*list,element*,i]	insert *element* at position **i** in *list*
Insert[*list,element*,-i]	insert *element* at position **i** in *list* counting from the end of list
Delete[*list*, i]	delete the element at position **i** in *list*
DeleteDuplicates[*list*]	delete all duplicates from *list*
DeleteDuplicates[*list, test*]	applies *test* to pairs of elements to determine wether they should be considered duplicates

```
Clear[a, b, c, d, u, v, w, x, y, z]
```

```
Prepend[ {a,b,c},x]
{x, a, b, c}
```

```
Append[ {u,v,w}, x]
{u, v, w, x}
```

```
Insert[{a,b,c,d},x,2]
{a, x, b, c, d}
```

```
Insert[{a,b,c,d},x,-2]
{a, b, c, x, d}
```

```
Delete[%,-2]
{a, b, c, d}
```

Delete elements unless they are larger than ones that came before:

```
DeleteDuplicates[{1, 7, 8, 4, 3, 4, 1, 9, 9, 2}, Greater]
{1, 7, 8, 9, 9}
```

`Part[`*list*`,i] = `*value*	Give *value* to element at position `i` of *list*
list`[[i]] = `*value*	Give *value* to element at position `i` of *list*
`ReplacePart[`*listt*`,`*element*`,i]`	Replace the element at position `i` of *list* by *element*
`ReplacePart[`*list*`,`*element*`,-i]`	As above but counting from end
`Position[`*list*`, `*form*`]`	The position at which *form* occurs in *list*

The first two commands perform changes in list. The third and the fourth command leave list unchanged, but
generate a new list comprising the changes.

```
l1 = {a,b,c,d}
```

```
{a, b, c, d}
```

```
Part[l1,3] = gg
```

```
gg
```

```
l1
```

```
{a, b, gg, d}
```

```
l1[[3]] = c
```

```
c
```

```
l1
```

```
{a, b, c, d}
```

```
l2 = ReplacePart[l1,10,2]
```

```
{a, 10, c, d}
```

```
l1
```

```
{a, b, c, d}
```

```
l2
```

```
{a, 10, c, d}
```

```
Position[l2,10]
```

```
{{2}}
```

```
ReplacePart[l2, b, Position[l2,10]]
```

```
{a, b, c, d}
```

```
lf = (a^2 + 3 a b c)^2
```

$(a^2 + 3\, a\, b\, c)^2$

```
Position[lf, a]
```

```
{{1, 1, 1}, {1, 2, 2}}
```

```
Expand[lf]
```

$a^4 + 6\, a^3\, b\, c + 9\, a^2\, b^2\, c^2$

```
Position[%, a]
```

```
{{1, 1}, {2, 2, 1}, {3, 2, 1}}
```

5.5 Combining and Rearranging Lists

Join[list1, list2, ...]	concatenate lists

Union[*list1*, *list2*, ...]	combine lists, remove repeated elements, sort the result
Sort[*list*]	sort the elements of list into a standard order
Reverse[*list*]	reverse the order of elements in *list*
Flatten[*list*]	remove all braces except those of highest level
Partition[*list*,*n*]	split up *list* into sublists of length *n*
Split[*list*]	splits list into sublists consisting of runs of identical elements
Thread[*list*]	"threads" (distributes) operator

```
Clear[a, b, c, d, e, l1, l2, l3, l4]

l1 = {a,b,c}

{a, b, c}

l2 = {d,e,c}

{d, e, c}

l3 = Join[l1,l2]

{a, b, c, d, e, c}

l4 = Union[l1,l2]

{a, b, c, d, e}

Reverse[l4]

{e, d, c, b, a}
```

The command **Union[]** uses the command **SameTest[]** (cf.7.4.1) to check for duplicates. This test fails if a list contains numbers, which should be the same, but have somewhat different numerical values due to computational errors. At the end of the next section it is shown how this problem may be remedied.

```
l1 = {{1,2,3,4},{5,6},{7,8},{9,10},{11,12}}

{{1, 2, 3, 4}, {5, 6}, {7, 8}, {9, 10}, {11, 12}}

Flatten[l1]

{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}

Partition[%, 3]

{{1, 2, 3}, {4, 5, 6}, {7, 8, 9}, {10, 11, 12}}

Reverse[%]

{{10, 11, 12}, {7, 8, 9}, {4, 5, 6}, {1, 2, 3}}
```

Thread

```
lx = {x1, x2, x3, x4, x5};

FullForm[lx]

List[x1, x2, x3, x4, x5]

lt = lx == l4

{x1, x2, x3, x4, x5} == {a, b, c, d, e}
```

```
Thread[lt]
```

{x1 == a, x2 == b, x3 == c, x4 == d, x5 == e}

```
n = 5; Clear[lx]
lx = Thread[x_Range[n]]
```

$\{x_1, x_2, x_3, x_4, x_5\}$

```
Clear[lt]
```

```
lt = lx == l4
```

$\{x_1, x_2, x_3, x_4, x_5\}$ == {a, b, c, d, e}

```
eq1 = a * x ^ 2 + b * x == c
eq2 = Thread[eq1 - c, Equal]
```

$b x + a x^2$ == c

$-c + b x + a x^2$ == 0

```
eq3 = a * x ^ 2 + 0.5 * b * x == c
eq4 = Thread[eq3 * 2, Equal]
```

$0.5 b x + a x^2$ == c

$2 (0.5 b x + a x^2)$ == 2 c

Split

```
Split[{8, 8, 8, 1, 2, 1, 8, 8, 7, 7, 7, 1, 1, 2, 2, 3}]
```

{{8, 8, 8}, {1}, {2}, {1}, {8, 8}, {7, 7, 7}, {1, 1}, {2, 2}, {3}}

Level, TreeForm

Level[*expr*, *levelspec*]
gives a list of all subexpressions of *expr* on levels specified by *levelspec*.

Level[*expr*, *levelspec*, *f*]
applies *f* to the sequence of subexpressions.

Basic Examples (3)

Give all parts at level -1:

```
Level[a + f[x, y ^ n], {-1}]
```

{a, x, y, 5}

Give all parts down to level 2:

```
Level[a + f[x, y ^ n], 2]
```

$\{a, x, y^5, f[x, y^5]\}$

Give all parts at levels 0 through infinity:

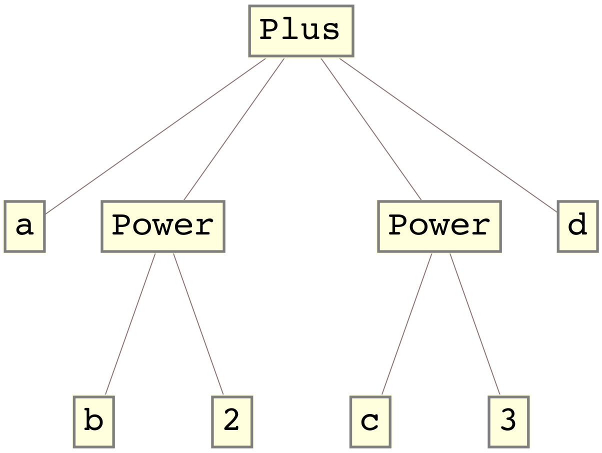
```
Level[a + f[x, y ^ n], {0, Infinity}]
```

$\{a, x, y, 5, y^5, f[x, y^5], a + f[x, y^5]\}$

TreeForm[*expr*]
displays *expr* as a tree with different levels at different depths.

`TreeForm[expr, n]`
displays *expr* as a tree only down to level n.

```
TreeForm[a + b^2 + c^3 + d]
```

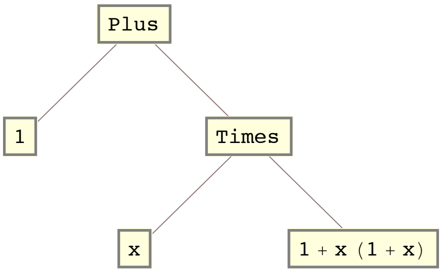


Show the tree form for the first two levels in the expression:

```
p = HornerForm[1 + x + x^2 + x^3, x]
```

```
1 + x (1 + x (1 + x))
```

```
TreeForm[p, 2]
```



Riffle

`Riffle[{`e_1`, `e_2`, …}, `x`]`	gives	`{`e_1`, `x`, `e_2`, `x`, …}`
`Riffle[{`e_1`, `e_2`, …}, {`x_1`, `x_2`, …}]`	gives	`{`e_1`, `x_1`, `e_2`, `x_2`, …}`
`Riffle[`*list*`, `x`, `n`]`	yields a list in which every n^{th} element is x.	

```
Riffle[{1, 2, 3, 4, 5, 6, 7}, {x, y}]
```

```
{1, x, 2, y, 3, x, 4, y, 5, x, 6, y, 7}
```

```
Clear[a, b, c, x, y, z]
Riffle[{a, b, c}, x]
```

```
{a, x, b, x, c}
```

```
Append[Riffle[{a, b, c}, x], x]
```

```
{a, x, b, x, c, x}
```

```
Riffle[{a, b, c}, {x, y, z}]
```

```
{a, x, b, y, c, z}
```

```
l1 = {x1, y1, z1};
l2 = {x2, y2, z2};
Riffle[l1, l2]
```

```
{x1, x2, y1, y2, z1, z2}
```

```
Riffle[Range[9], x, 3]
```

```
{1, 2, x, 3, 4, x, 5, 6, x, 7, 8, x, 9}
```

Sequence

`Sequence[expr1, expr2, ... ]`	represents a sequence of arguments to be spliced automatically into any function.

```
lst = Table[a[i], {i, 20}]
```

```
{a[1], a[2], a[3], a[4], a[5], a[6], a[7], a[8], a[9], a[10],
 a[11], a[12], a[13], a[14], a[15], a[16], a[17], a[18], a[19], a[20]}
```

```
le = lst[[Table[k, {k, 1, Length[lst], 2}]]]
```

```
{a[1], a[3], a[5], a[7], a[9], a[11], a[13], a[15], a[17], a[19]}
```

Partition

```
Partition[lst, Sequence[1, 2]]
```

```
{{a[1]}, {a[3]}, {a[5]}, {a[7]}, {a[9]},
 {a[11]}, {a[13]}, {a[15]}, {a[17]}, {a[19]}}
```

```
Flatten[%]
```

```
{a[1], a[3], a[5], a[7], a[9], a[11], a[13], a[15], a[17], a[19]}
```

```
le = lst[[Table[k, {k, 2, Length[lst], 2}]]]
```

```
{a[2], a[4], a[6], a[8], a[10], a[12], a[14], a[16], a[18], a[20]}
```

```
Partition[lst, Sequence[2, 2]]
```

```
{{a[1], a[2]}, {a[3], a[4]}, {a[5], a[6]}, {a[7], a[8]}, {a[9], a[10]},
 {a[11], a[12]}, {a[13], a[14]}, {a[15], a[16]}, {a[17], a[18]}, {a[19], a[20]}}
```

```
Partition[Drop[lst, 1], Sequence[1, 2]]
```

```
{{a[2]}, {a[4]}, {a[6]}, {a[8]}, {a[10]},
 {a[12]}, {a[14]}, {a[16]}, {a[18]}, {a[20]}}
```

```
le = lst[[Table[k, {k, 1, Length[lst], 3}]]]
```

```
{a[1], a[4], a[7], a[10], a[13], a[16], a[19]}
```

```
Partition[lst, Sequence[1, 3]]
```

```
{{a[1]}, {a[4]}, {a[7]}, {a[10]}, {a[13]}, {a[16]}, {a[19]}}
```

```
Partition[lst, Sequence[2, 3]]
```

```
{{a[1], a[2]}, {a[4], a[5]}, {a[7], a[8]},
 {a[10], a[11]}, {a[13], a[14]}, {a[16], a[17]}, {a[19], a[20]}}
```

```
lx  = {x1, x2, x3, x4, x5};
ly  = {y1, y2, y3, y4, y5};
```

```
lxy = Transpose[{lx, ly}]
```

```
{{x1, y1}, {x2, y2}, {x3, y3}, {x4, y4}, {x5, y5}}
```

FoldList

FoldList[*f*,x,*{a,b,c}*]	gives {x, f[x,a], f[f[x,a],b], ...}

Foldlist[] may be used to get the list **ls** of partial sums of a list **li** :

```
li = {r1, r2, r3, r4, r5}
```

```
{r1, r2, r3, r4, r5}
```

```
ls = FoldList[Plus, r1, Rest[li]]
```

```
{r1, r1 + r2, r1 + r2 + r3, r1 + r2 + r3 + r4, r1 + r2 + r3 + r4 + r5}
```

The summations by **FoldList[]** are (much) faster than those by combined commands:

```
lr = RandomReal[{0, 1}, 10 000];
```

```
{ts, lsf} = FoldList[Plus, lr[[1]], Rest[lr]] // Timing; ts
```

```
0.002096
```

```
{tss, lss} = Table[Sum[lr[[k]], {k, n}], {n, Length[lr]}] // Timing; tss
```

```
4.265837
```

```
{tst, lst} = Table[Apply[Plus, Take[lr, n]], {n, Length[lr]}] // Timing; tst
```

```
3.108871
```

```
lst[[-1]]
```

```
4989.39
```

```
lsf == lss
```

```
True
```

```
lss == lst
```

```
True
```

5.6 Comands for Sets

Union[list1, list2,...] $\hat{=}$ **list1∪list2 ∪ ...**	combine lists, remove repeated elements, sort the result
Intersection[list1, list2,...] $\hat{=}$ **list1∩list2 ∩ ...**	gives a sorted list of all elements common to all lists.
Complement[listall,list1, list2,...]	gives those elements in listall which are not in any of the **listi**

```
Clear[a, b, c, d, r, l1, l2, l3]
l1 = {a, b, c}; l2 = {a, 2, r};
l3 = {a, b, c, d};
```

```
la = Union[l1, l2, l3]
```

```
{2, a, b, c, d, r}
```

```
l1 ⋃ l2 ⋃ l3
```

```
{2, a, b, c, d, r}
```

The command **Union[]** uses the command **SameTest[]** (cf.5.6.1) to check for duplicates. This test is not reasonable due to computational errors. In the subsubsection below it is shown how this problem may be remedied.

```
li = l1⋂ l2⋂ l3
```

```
{a}
```

```
Complement[la, l1, l3]
```

```
{2, r}
```

5.6.1 Replacing SameTest[] by anonther test

The command **Union[]** uses the command **SameTest[]** to check for duplicates. This test is not appropriate if a list contains numbers, which should be the same, but have somewhat different numerical values due to computational errors. The test **SameTest[]** may be replaced by the test named **testdiff** given below to remedy this situation.

```
a = N[{π, π + 10^-5, π - 10^-5}, 6]
```

```
{3.14159, 3.14160, 3.14158}
```

```
testdiff = (If[NumericQ[#1 - #2], Abs[N[#1 - #2]] < 10^-4, #1 == #2] &);
Union[a, SameTest -> testdiff]
```

```
{3.14158}
```

5.7 Selecting Elements by Criteria

Cases[*{e1, e2, ... }, pattern*]	gives a list of the ***ei*** that match the ***pattern***.

```
Clear[a, b, c, d]
```

```
li = {1, 2, 3, -5, a, b, d, .33, -.73}
```

```
{1, 2, 3, -5, a, b, d, 0.33, -0.73}
```

```
Cases[li, _Integer]
```

```
{1, 2, 3, -5}
```

```
Cases[li, _Symbol]
```

```
{a, b, d}
```

```
Cases[li, _Real]
```

```
{0.33, -0.73}
```

```
Cases[li, _?EvenQ]
```

```
{2}
```

```
Cases[li, _?OddQ]
```

```
{1, 3, -5}
```

```
Cases[li, _?Positive]
```

```
{1, 2, 3, 0.33}
```

```
Cases[li, _?Negative]
```

```
{-5, -0.73}
```

```
Cases[li, _?NumberQ]
```

```
{1, 2, 3, -5, 0.33, -0.73}
```

`DeleteCases[expr, pattern]`	removes all elements of ***expr*** which match ***pattern***.

Some of the tests introduced below, which are closed by the ampercent (**&**) use pure functions (cf.4.1.4).

```
Select[li, IntegerQ]
```

```
{1, 2, 3, -5}
```

```
Select[li, (#[[0]] == Integer) &]
```

```
{1, 2, 3, -5}
```

```
Select[li, (#[[0]] == Real) &]
```

```
{0.33, -0.73}
```

```
li
```

```
{1, 2, 3, -5, a, b, d, 0.33, -0.73}
```

```
Select[li, (#[[0]] == Symbol) &]
```

```
{a, b, d}
```

```
Select[li, EvenQ]
```

```
{2}
```

```
Select[li, OddQ]
```

```
{1, 3, -5}
```

```
Select[li, # > 0.5 &]
```

```
{1, 2, 3}
```

```
Select[li, # < 0 &]
```

```
{-5, -0.73}
```

```
lin = {{2, 8, 0}, {3, 1, 0}, {3, 2, -6}, {3, 3, -5}, {3, 4, 0}, {3, 5, 0}};
```

We want all sublists of **lin** having a negative entry at the third place.

```
Select[lin, ( #[[3]] < 0 ) &]
```

```
{{3, 2, -6}, {3, 3, -5}}
```

```
li
```

```
{1, 2, 3, -5, a, b, d, 0.33, -0.73}
```

```
DeleteCases[li, _Integer]
```

```
{a, b, d, 0.33, -0.73}
```

```
DeleteCases[li, _Real]
```

```
{1, 2, 3, -5, a, b, d}
```

```
DeleteCases[li, _Symbol]
```

```
{1, 2, 3, -5, 0.33, -0.73}
```

```
DeleteCases[li, _?EvenQ]
```

```
{1, 3, -5, a, b, d, 0.33, -0.73}
```

```
DeleteCases[li, _?OddQ]
```

```
{2, a, b, d, 0.33, -0.73}
```

```
DeleteCases[li, _?Positive]
```

{-5, a, b, d, -0.73}

```
DeleteCases[li, _?Negative]
```

{1, 2, 3, a, b, d, 0.33}

```
DeleteCases[li, _?NumberQ]
```

{a, b, d}

```
ll = {a, b, c, 1, 2, 3, 1 / 2, 3 / 4, .5, .33, .2 + .3 I, -4, - 5, - 0.77}
```

$\{a, b, c, 1, 2, 3, \frac{1}{2}, \frac{3}{4}, 0.5, 0.33, 0.2 + 0.3\,i, -4, -5, -0.77\}$

```
Map[Head, ll]
```

{Symbol, Symbol, Symbol, Integer, Integer, Integer,
 Rational, Rational, Real, Real, Complex, Integer, Integer, Real}

```
li = {1, 2, 3, 4};
DeleteCases[li, _?(# > 2 &)] (* parentheses essential ! *)
```

{1, 2}

```
DeleteCases[li, x_ /; x > 2]
```

{1, 2}

```
Select[ll, NumberQ]
```

$\{1, 2, 3, \frac{1}{2}, \frac{3}{4}, 0.5, 0.33, 0.2 + 0.3\,i, -4, -5, -0.77\}$

```
Select[ll, (#[[0]] == Integer || #[[0]] == Real) &]
```

{1, 2, 3, 0.5, 0.33, -4, -5, -0.77}

```
Select[ll, (#[[0]] == Integer && # < 0) &]
```

{-4, -5}

Count[*list,pattern*]	gives the number of elemens in ***list*** that match ***pattern***
Count[*expr,pattern,levelspec*]	gives the total number of subexpressions matching ***pattern*** that appear at the levels in ***expr*** specified by ***levelspec***

```
Count[{1, 0, 0, 1, 2, 3, 7, 1}, 0]
```

2

```
Count[{3, -2, 7, -i, c, -6}, _?Negative]
```

2

```
Count[{7, 0.22, 0.31, 0.3 + i 0.1}, _Real]
```

2

ex = $c + e^{-x^2} + \mathrm{Exp}[z^2] + w^{5x} + s^2 + \frac{1}{y^2} + 1;$

This command returns the number of summands that are powers

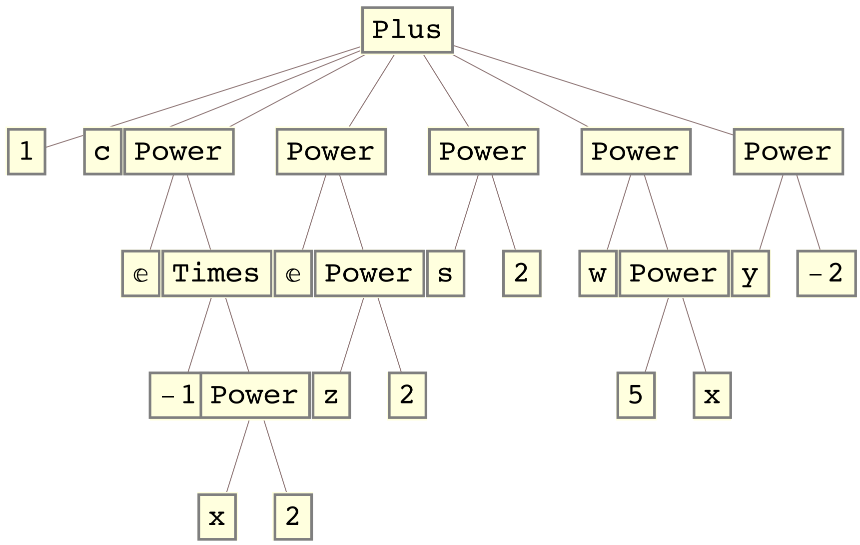
```
Count[ex, x_^y_]
```

5

```
Table[Count[ex, x_^y_, k], {k, 10}]
```

```
{5, 7, 8, 8, 8, 8, 8, 8, 8, 8}
```

```
TreeForm[ex]
```




```
Position[ex, Power]
```

```
{{3, 0}, {3, 2, 2, 0}, {4, 0}, {4, 2, 0}, {5, 0}, {6, 0}, {6, 2, 0}, {7, 0}}
```

{i, 0}, i = 3, 4, 5, 6, 7 correspond to level 1; these and {4,2,0}, {6,2,0} to level 2; these and {3,2,2,0} to level 3 and higher.

5.8 Exercises

5.1 Set up a list containing the roots of the polynomial
24 - 14 x - 13 x^2 + 2 x^3 + x^4.
Sort it according to the numeric values of the roots.
Set up a list containing only the positive roots.

5.2 Set up a list whose elements are lists containing all different pairs (cos(x), sin(x)) for x = k 2 π/8, k c IN. Get a sublist for those pairs where sin(x) >= 0.

5.3 Generate the list of natural numbers between 0 and 10.
Insert Pi at its appropiate place.

5.4 Generate a Table for Plots of Cos[n x] for n = 1,3,5,7 in the intervall [0 <= x <= π].

5.5 Generate a list of the natural numbers between 0 and 6.
Replace each even number *ne* by π ^*ne*. In the resulting list replace each π by - 3.

5.6 Write a function that works on a list such as {a1, a2, a3, a4, ...}
and returns a list {a2/a1, a3/a2, a4/a3, ...}.

5.7 Reduce the following list {a,b,c,e,b,e,f,f,a,d,b,c} such that each letter
occurs only once and order it in inverse alphabetical order.

5.8 Combine the follwing two lists into one list l3 such that the elements
of l1 come after all those of l2; l1 = {2,a,c,4}, l2 = {3,c,f,a,5}. Then
find the positions of the letter c in l3.

5.9 There are two lists, each containing 10 elements:
xl = { x[1], x[2], ..., x[10] }, yl = { y[1],...,y[10] }
Combine them into one list by one simple command:
xyl = { { x[1], y[1] }, ..., { x[10], y[10]} }

5.10 There are two lists l1, l2, each containing the coordinates of the

lx = { { x[1,1], x[1,2], x[1,3]}, ..., { x[n,1], x[n,2], x[n,3]} } ,
ly = { { y[1,1], y[1,2], y[1,3]}, ..., { y[n,1], y[n,2], y[n,3]} } .

Combine them into one interspersing list such that

lxy = { { x[1,1], x[1,2], x[1,3]}, { y[1,1], y[1,2], y[1,3]}, ... ,
{ x[n,1], x[n,2], x[n,3]}, { y[n,1], y[n,2], y[n,3]} } .

5.11 Generate a list of real and complex numbers by the statement:
lrc = {Random[Real,{-2,5}],Random[Real,{-2,5}],Random[Real,{-2,5}],Random[Complex,{-2,5I}],
Random[Real,{-2,5}],Random[Real,{-2,5}],Random[Complex,{-2,5I}],Random[Real,{-2,5}],
Random[Complex,{-2,5I}],Random[Real,{-2,5}],Random[Real,{-2,5}],Random[Complex,{-2,5I}],Rand
om[Real,{-2,5}]}
i) Generate the sublist containing only the complex numbers by a single command.
ii) Generate the complementary sublist comprising only the real numbers.