In[194]:= (*Instructor：Hiroyuki Akama

Basic Built–in Functions of Mathematica Linear Algebra– List Vector and Matrix (2)*)

```
t = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
t
? Table
Table[i, {i, 10}]
Table[i, {i, 4}, {j, 8}]
% // MatrixForm
? MatrixForm
```

Out[195]= {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

Table[*expr*, {i_{max}}] generates a list of i_{max} copies of *expr*.
Table[*expr*, {*i*, i_{max}}] generates a list of the values of *expr* when *i* runs from 1 to i_{max}.
Table[*expr*, {*i*, i_{min}, i_{max}}] starts with $i = i_{min}$.
Table[*expr*, {*i*, i_{min}, i_{max}, *di*}] uses steps *di*.
Table[*expr*, {*i*, {i_1, i_2, …}}] uses the successive values i_1, i_2, ….
Table[*expr*, {*i*, i_{min}, i_{max}}, {*j*, j_{min}, j_{max}}, …] gives a nested list. The list associated with *i* is outermost. ≫

Out[197]= {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

Out[198]= {{1, 1, 1, 1, 1, 1, 1, 1}, {2, 2, 2, 2, 2, 2, 2, 2},
{3, 3, 3, 3, 3, 3, 3, 3}, {4, 4, 4, 4, 4, 4, 4, 4}}

Out[199]//MatrixForm=

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 & 4 & 4 & 4 & 4 \end{pmatrix}$$

MatrixForm[*list*] prints with the elements of *list* arranged in a regular array. ≫

In[201]:=
```
matrix1 = {{1, 0, 0, 1}, {1, 1, 0, 1}, {0, 0, 1, 1}, {0, 1, 0, 1}};
matrix1 // MatrixForm
MatrixForm[matrix1]
matrix1 - IdentityMatrix[Length[matrix1]] // MatrixForm
? IdentityMatrix
```

Out[202]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix}$$

Out[203]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix}$$

Out[204]//MatrixForm=

$$\begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

IdentityMatrix[*n*] gives the $n \times n$ identity matrix. ≫

In[206]:=
```
F0[i_, j_] := 1;
Array[F0, {4, 4}]
? Array
Table[1, {i, 4}, {j, 4}]
matrix1 /. {0 -> 1, 1 -> 0} // MatrixForm
ReplaceAll[matrix1, {0 -> 1, 1 -> 0}] // MatrixForm
? ReplaceAll
```

Out[207]= {{1, 1, 1, 1}, {1, 1, 1, 1}, {1, 1, 1, 1}, {1, 1, 1, 1}}

Array[*f*, *n*] generates a list of length *n*, with elements *f*[*i*].
Array[*f*, {n_1, n_2, …}] generates an $n_1 \times n_2 \times \ldots$ array of nested lists, with elements *f*[i_1, i_2, …].
Array[*f*, {n_1, n_2, …}, {r_1, r_2, …}] generates a list using the index origins r_i (default 1).
Array[*f*, *dims*, *origin*, *h*] uses head *h*, rather than List, for each level of the array. ≫

Out[209]= {{1, 1, 1, 1}, {1, 1, 1, 1}, {1, 1, 1, 1}, {1, 1, 1, 1}}

Out[210]//MatrixForm=

$$\begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

Out[211]//MatrixForm=

$$\begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

expr /. *rules* applies a rule or list of rules in an attempt to transform each subpart of an expression *expr*. ≫

In[213]:=
```
F1[i_, j_] := If[i == j, 1, 0];
Array[F1, {7, 7}];
% // MatrixForm
IdentityMatrix[7] == Array[F1, {7, 7}]
```

Out[215]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Out[216]= True

In[217]:=
```
RandomInteger[{1, 4}]
? RandomInteger
samplelist = Table[RandomInteger[{1, 4}], {i, 8}]
```

Out[217]= 1

RandomInteger[{i_{min}, i_{max}}] gives a pseudorandom integer in the range {i_{min}, … , i_{max}}.
RandomInteger[i_{max}] gives a pseudorandom integer in the range {0, … , i_{max}}.
RandomInteger[] pseudorandomly gives 0 or 1.
RandomInteger[*range*, *n*] gives a list of *n* pseudorandom integers.
RandomInteger[*range*, {n_1, n_2, …}] gives an $n_1 \times n_2 \times \ldots$ array of pseudorandom integers.
RandomInteger[*dist*, …] samples from the symbolic discrete distribution *dist*. ≫

Out[219]= {1, 3, 2, 4, 1, 4, 4, 2}

In[220]:=
```
? Part
samplelist[[2]]
Part[samplelist, 2]
samplelist[[2]] == Part[samplelist, 2]
```

expr[[*i*]] or Part[*expr*, *i*] gives the ith part of *expr*.
expr[[−*i*]] counts from the end.
expr[[*i*, *j*, …]] or Part[*expr*, *i*, *j*, …] is equivalent to *expr*[[*i*]][[*j*]] ….
expr[[{i_1, i_2, …}]] gives a list of the parts i_1, i_2, … of *expr*.
expr[[*m* ;; *n*]] gives parts *m* through *n*.
expr[[*m* ;; *n* ;; *s*]] gives parts *m* through *n* in steps of *s*. ≫

Out[221]= 3

Out[222]= 3

Out[223]= True

In[224]:=
```
MyRandomFunction1[m_, n_] := Table[RandomInteger[{1, m}], {i, n}];
MyRandomFunction1[6, 8]
Table[MyRandomFunction1[6, 8], {10}]
? Table
```

Out[225]= {1, 1, 2, 1, 3, 3, 3, 1}

Out[226]=
```
{{3, 5, 3, 1, 5, 2, 2, 3}, {4, 5, 5, 5, 3, 5, 5, 6},
 {6, 2, 5, 1, 2, 5, 4, 6}, {6, 5, 5, 1, 6, 6, 6, 5},
 {5, 2, 2, 2, 3, 5, 2, 6}, {5, 6, 6, 6, 6, 6, 1, 6}, {3, 4, 5, 2, 6, 4, 5, 2},
 {4, 5, 4, 2, 3, 2, 1, 2}, {2, 3, 1, 5, 6, 4, 6, 6}, {1, 1, 5, 6, 2, 3, 6, 4}}
```

Table[*expr*, {i_{max}}] generates a list of i_{max} copies of *expr*.
Table[*expr*, {*i*, i_{max}}] generates a list of the values of *expr* when *i* runs from 1 to i_{max}.
Table[*expr*, {*i*, i_{min}, i_{max}}] starts with $i = i_{min}$.
Table[*expr*, {*i*, i_{min}, i_{max}, *di*}] uses steps *di*.
Table[*expr*, {*i*, {i_1, i_2, …}}] uses the successive values i_1, i_2, ….
Table[*expr*, {*i*, i_{min}, i_{max}}, {*j*, j_{min}, j_{max}}, …] gives a nested list. The list associated with *i* is outermost. ≫

In[228]:=
```
MyRandomFunction2[m_, n_, t_] := Table[Table[RandomInteger[{1, m}], {i, n}], {t}];
samplematrix = MyRandomFunction2[4, 8, 12]
MyRandomFunction2a[m_, n_, t_] := Table[RandomInteger[{1, m}], {i, n}, {j, t}];
samplematrix2a = MyRandomFunction2a[4, 12, 8]
```

Out[228]=
```
{{3, 4, 1, 2, 1, 3, 4, 4}, {3, 2, 1, 4, 4, 2, 1, 2}, {3, 2, 3, 4, 3, 2, 3, 2},
 {4, 4, 1, 1, 2, 4, 2, 2}, {2, 2, 1, 4, 3, 1, 3, 1}, {4, 2, 2, 2, 4, 4, 3, 2},
 {1, 3, 2, 4, 2, 4, 3, 3}, {3, 3, 4, 4, 4, 4, 1, 3}, {2, 1, 1, 1, 1, 4, 2, 2},
 {2, 1, 1, 1, 3, 1, 1, 2}, {1, 3, 4, 4, 2, 3, 1, 1}, {1, 3, 2, 2, 2, 3, 3, 1}}
```

General::spell1 :
New symbol name "MyRandomFunction2a" is similar to existing symbol "MyRandomFunction2" and may be misspelled. ≫

Out[230]=
```
{{4, 4, 2, 1, 2, 4, 4, 1}, {1, 3, 3, 3, 2, 3, 2, 2}, {4, 2, 4, 3, 1, 1, 4, 1},
 {3, 3, 2, 2, 3, 3, 2, 2}, {2, 3, 4, 2, 4, 3, 4, 2}, {3, 2, 2, 1, 2, 2, 2, 3},
 {3, 2, 3, 1, 3, 3, 4, 1}, {1, 3, 3, 2, 3, 2, 4, 2}, {4, 4, 1, 3, 2, 1, 3, 3},
 {4, 3, 3, 2, 4, 4, 4, 2}, {4, 4, 4, 4, 4, 2, 1, 1}, {2, 4, 1, 4, 3, 1, 3, 1}}
```

In[231]:=
```
samplematrix[[3]]
```

Out[231]= {3, 2, 3, 4, 3, 2, 3, 2}

In[232]:=
```
samplematrix[[3, 8]]
samplematrix[[3]][[8]]
samplematrix[[3, 8]] == samplematrix[[3]][[8]]
Part[samplematrix, 3]
Part[samplematrix, 3, 8]
```

Out[232]= 2

Out[233]= 2

Out[234]= True

Out[235]= {3, 2, 3, 4, 3, 2, 3, 2}

Out[236]= 2

In[237]:=
```
samplematrix1 = MyRandomFunction2[6, 8, 8]
```

Out[237]=
```
{{3, 2, 6, 6, 5, 1, 1, 1}, {1, 3, 1, 5, 2, 6, 4, 2},
 {4, 4, 2, 5, 6, 1, 6, 3}, {2, 4, 4, 1, 4, 4, 5, 3}, {1, 3, 6, 4, 1, 1, 3, 1},
 {2, 5, 1, 6, 2, 1, 6, 6}, {4, 2, 6, 3, 6, 2, 2, 2}, {5, 3, 3, 4, 3, 1, 5, 1}}
```

In[239]:=
```
MatrixPower[samplematrix1, 4] // MatrixForm
? MatrixPower
```

Out[239]//MatrixForm=

$$\begin{pmatrix} 45\,656 & 54\,521 & 64\,186 & 68\,318 & 63\,595 & 37\,830 & 66\,882 & 39\,148 \\ 44\,484 & 53\,024 & 62\,566 & 66\,587 & 61\,957 & 36\,810 & 64\,970 & 38\,120 \\ 54\,932 & 65\,350 & 77\,438 & 81\,821 & 76\,419 & 45\,773 & 80\,174 & 46\,730 \\ 49\,205 & 58\,774 & 69\,068 & 73\,833 & 68\,524 & 40\,622 & 72\,102 & 42\,242 \\ 38\,021 & 45\,398 & 53\,106 & 56\,731 & 53\,051 & 31\,475 & 55\,807 & 32\,608 \\ 52\,315 & 62\,303 & 73\,517 & 77\,774 & 72\,818 & 43\,557 & 76\,536 & 44\,519 \\ 48\,773 & 58\,337 & 68\,618 & 73\,268 & 67\,956 & 40\,302 & 71\,536 & 41\,923 \\ 45\,441 & 54\,049 & 63\,846 & 67\,638 & 63\,315 & 37\,863 & 66\,366 & 38\,699 \end{pmatrix}$$

MatrixPower[*m*, *n*] gives the nth matrix power of the matrix *m*.
MatrixPower[*m*, *n*, *v*] gives the nth matrix power of the matrix *m* applied to the vector *v*. ≫

In[241]:=
```
MyMatrixFunction[i_] := MatrixPower[samplematrix1, i];
MyMatrixFunction[3]
MyMatrixFunction[3][[4]]
MyMatrixFunction[3][[4]][[5]]
```

Out[242]=
```
{{1760, 2089, 2468, 2627, 2453, 1473, 2568, 1498},
 {1749, 2031, 2370, 2551, 2364, 1449, 2513, 1462},
 {2098, 2545, 2900, 3183, 2954, 1704, 3135, 1845},
 {1901, 2251, 2690, 2786, 2652, 1606, 2760, 1603},
 {1456, 1734, 2117, 2210, 2002, 1209, 2104, 1239},
 {1998, 2405, 2809, 3076, 2790, 1608, 2928, 1762},
 {1881, 2231, 2667, 2764, 2655, 1595, 2742, 1578},
 {1725, 2110, 2441, 2631, 2417, 1405, 2592, 1507}}
```

Out[243]=
```
{1901, 2251, 2690, 2786, 2652, 1606, 2760, 1603}
```

Out[244]=
```
2652
```

In[256]:=
```
samplelist
?Select
Select[samplelist, EvenQ]
Select[samplelist, OddQ]
Union[%]
```

Out[256]= {1, 3, 2, 4, 1, 4, 4, 2}

Select[*list*, *crit*] picks out all elements e_i of *list* for which *crit*[e_i] is True.
Select[*list*, *crit*, *n*] picks out the first *n* elements for which *crit*[e_i] is True. ≫

Out[258]= {2, 4, 4, 4, 2}

Out[259]= {1, 3, 1}

Out[260]= {1, 3}

Select[*list*, *crit*] *crit*[e_i]がTrueとなる *list* のすべての要素 e_i を取り出す.
Select[*list*, *crit*, *n*] *crit*[e_i]がTrueとなる最初から *n* 個の要素を取り出す. ≫

{4, 4, 4, 4, 2}

{1, 3, 3}

{1, 3}

In[250]:=
```
Cases[samplelist, x_?(Mod[#, 2] == 0 &)]
Cases[samplelist, x_?(Mod[#, 2] != 0 &)]
?Cases
?Mod
Select[samplelist, EvenQ] == Cases[samplelist, x_?(Mod[#, 2] == 0 &)]
Select[samplelist, OddQ] == Cases[samplelist, x_?(Mod[#, 2] != 0 &)]
```

Out[250]= {2, 4, 4, 4, 2}

Out[251]= {1, 3, 1}

Cases[{e_1, e_2, …}, *pattern*] gives a list of the e_i that match the pattern.
Cases[{e_1, …}, *pattern* –> *rhs*] gives a list of the values of *rhs* corresponding to the e_i that match the pattern.
Cases[*expr*, *pattern*, *levelspec*] gives a list of all parts of *expr* on levels specified by *levelspec* that match the pattern.
Cases[*expr*, *pattern* –> *rhs*, *levelspec*] gives the values of *rhs* that match the pattern.
Cases[*expr*, *pattern*, *levelspec*, *n*] gives the first *n* parts in *expr* that match the pattern. ≫

Mod[*m*, *n*] gives the remainder on division of *m* by *n*.
Mod[*m*, *n*, *d*] uses an offset *d*. ≫

Out[254]= True

Out[255]= True

In[261]:=
```
samplelist
Position[samplelist, x_?(# == 1 &)]
Position[samplelist, x_?(Mod[#, 2] != 0 &)]
Position[samplelist, x_?(# ≥ 3 &)]
Flatten[Position[samplelist, x_?(# == 3 &)]]
Part[samplelist, %]
```

Out[261]= {1, 3, 2, 4, 1, 4, 4, 2}

Out[262]= {{1}, {5}}

Out[263]= {{1}, {2}, {5}}

Out[264]= {{2}, {4}, {6}, {7}}

Out[265]= {2}

Out[266]= {3}

In[267]:=
```
Position[IdentityMatrix[4], x_?(# == 1 &), 2]
Position[IdentityMatrix[4], x_?(# ≠ 1 &), 2]
```

Out[267]= {{1, 1}, {2, 2}, {3, 3}, {4, 4}}

Out[268]= {{1, 2}, {1, 3}, {1, 4}, {2, 1}, {2, 3}, {2, 4}, {3, 1}, {3, 2}, {3, 4}, {4, 1}, {4, 2}, {4, 3}}

In[269]:=
```
samplematrix1
% // MatrixForm
Position[samplematrix1, x_?(# == 6 &), 2]
Position[samplematrix1, x_?(# < 3 &), 2]
?Position
```

Out[269]= {{3, 2, 6, 6, 5, 1, 1, 1}, {1, 3, 1, 5, 2, 6, 4, 2},
{4, 4, 2, 5, 6, 1, 6, 3}, {2, 4, 4, 1, 4, 4, 5, 3}, {1, 3, 6, 4, 1, 1, 3, 1},
{2, 5, 1, 6, 2, 1, 6, 6}, {4, 2, 6, 3, 6, 2, 2, 2}, {5, 3, 3, 4, 3, 1, 5, 1}}

Out[270]//MatrixForm=

$$\begin{pmatrix} 3 & 2 & 6 & 6 & 5 & 1 & 1 & 1 \\ 1 & 3 & 1 & 5 & 2 & 6 & 4 & 2 \\ 4 & 4 & 2 & 5 & 6 & 1 & 6 & 3 \\ 2 & 4 & 4 & 1 & 4 & 4 & 5 & 3 \\ 1 & 3 & 6 & 4 & 1 & 1 & 3 & 1 \\ 2 & 5 & 1 & 6 & 2 & 1 & 6 & 6 \\ 4 & 2 & 6 & 3 & 6 & 2 & 2 & 2 \\ 5 & 3 & 3 & 4 & 3 & 1 & 5 & 1 \end{pmatrix}$$

Out[271]= {{1, 3}, {1, 4}, {2, 6}, {3, 5}, {3, 7}, {5, 3}, {6, 4}, {6, 7}, {6, 8}, {7, 3}, {7, 5}}

Out[272]= {{1, 2}, {1, 6}, {1, 7}, {1, 8}, {2, 1}, {2, 3}, {2, 5}, {2, 8},
{3, 3}, {3, 6}, {4, 1}, {4, 4}, {5, 1}, {5, 5}, {5, 6}, {5, 8}, {6, 1},
{6, 3}, {6, 5}, {6, 6}, {7, 2}, {7, 6}, {7, 7}, {7, 8}, {8, 6}, {8, 8}}

Position[*expr*, *pattern*] gives a list of the positions at which objects matching *pattern* appear in *expr*.
Position[*expr*, *pattern*, *levelspec*] finds only objects that appear on levels specified by *levelspec*.
Position[*expr*, *pattern*, *levelspec*, *n*] gives the positions of the first *n* objects found. ≫

In[274]:=
```
samplematrix1
```

Out[274]= {{3, 2, 6, 6, 5, 1, 1, 1}, {1, 3, 1, 5, 2, 6, 4, 2},
{4, 4, 2, 5, 6, 1, 6, 3}, {2, 4, 4, 1, 4, 4, 5, 3}, {1, 3, 6, 4, 1, 1, 3, 1},
{2, 5, 1, 6, 2, 1, 6, 6}, {4, 2, 6, 3, 6, 2, 2, 2}, {5, 3, 3, 4, 3, 1, 5, 1}}

In[275]:=
```
Clear[samplematrix1]
?Clear
```

Clear[$symbol_1$, $symbol_2$, …] clears values and definitions for the $symbol_i$.
Clear["$form_1$", "$form_2$", …] clears values and
definitions for all symbols whose names match any of the string patterns $form_i$. ≫

In[277]:=
```
samplematrix1
```

Out[277]=
```
samplematrix1
```

In[278]:=
```
samplelist
```

Out[278]=
```
{1, 3, 2, 4, 1, 4, 4, 2}
```

In[279]:=
```
Range[Length[samplelist]]
?Range
```

Out[279]=
```
{1, 2, 3, 4, 5, 6, 7, 8}
```

Range[i_{max}] generates the list {1, 2, …, i_{max}}.
Range[i_{min}, i_{max}] generates the list {i_{min}, …, i_{max}}.
Range[i_{min}, i_{max}, di] uses step di. ≫

In[281]:=
```
Part[samplelist, 1]
Part[samplelist, 2]
Part[samplelist, 3]
Part[samplelist, 4]
Part[samplelist, 5]
Part[samplelist, 6]
Part[samplelist, 7]
Part[samplelist, 8]
{{1, Part[samplelist, 1]}, {2, Part[samplelist, 2]},
 {3, Part[samplelist, 3]}, {4, Part[samplelist, 4]}, {5, Part[samplelist, 5]},
 {6, Part[samplelist, 6]}, {7, Part[samplelist, 7]}, {8, Part[samplelist, 8]}}
Transpose[{Range[Length[samplelist]], samplelist}]
```

Out[281]= 1

Out[282]= 3

Out[283]= 2

Out[284]= 4

Out[285]= 1

Out[286]= 4

Out[287]= 4

Out[288]= 2

Out[289]= {{1, 1}, {2, 3}, {3, 2}, {4, 4}, {5, 1}, {6, 4}, {7, 4}, {8, 2}}

Out[290]= {{1, 1}, {2, 3}, {3, 2}, {4, 4}, {5, 1}, {6, 4}, {7, 4}, {8, 2}}

In[291]:=
```
Insert[{1, 2, 3, 4}, 2.5, 3]
? Insert
```

Out[291]= {1, 2, 2.5, 3, 4}

Insert[*list*, *elem*, *n*] inserts *elem* at position *n* in *list*. If *n* is negative, the position is counted from the end.
Insert[*expr*, *elem*, {*i*, *j*, …}] inserts *elem* at position {*i*, *j*, …} in *expr*.
Insert[*expr*, *elem*, {{i_1, j_1, …}, {i_2, j_2, …}, …}] inserts *elem* at several positions. ≫

In[293]:=
```
samplelist1 = {1, 3, 2, 1, 5, 6, 7, 3, 1, 9};
pos = {2, 7, 9};
samplelist1inserted = {1, {}, 3, 2, 1, 5, {}, 6, {}, 7, 3, 1, 9};
```

In[296]:=
```
Part[samplelist1inserted, pos]

Insert[samplelist1, {}, pos[[1]]]
Insert[Insert[samplelist1, {}, pos[[1]]], {}, pos[[2]]]
Insert[Insert[Insert[samplelist1, {}, pos[[1]]], {}, pos[[2]]], {}, pos[[3]]]
```

Out[296]= {{}, {}, {}}

Out[297]= {1, {}, 3, 2, 1, 5, 6, 7, 3, 1, 9}

Out[298]= {1, {}, 3, 2, 1, 5, {}, 6, 7, 3, 1, 9}

Out[299]= {1, {}, 3, 2, 1, 5, {}, 6, {}, 7, 3, 1, 9}

In[300]:=
```
? While
? Drop
i = 1;
tmp[1] = samplelist1;
While[i ≤ Length[pos], tmp[i + 1] = Insert[tmp[i], {}, pos[[i]]]; i++]
tmp[i]
j = 1;
poslist = {2, 4, 6, 8, 10, 12, 14, 16, 18};
mylist[1] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
While[j ≤ Length[poslist], mylist[j + 1] = Insert[mylist[j], NUL, poslist[[j]]]; j++]
mylist[j]
Drop[Flatten[{#, NUL} & /@ {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}], -1]
```

While[*test*, *body*] evaluates *test*, then *body*, repetitively, until *test* first fails to give True. ≫

Drop[*list*, *n*] gives *list* with its first *n* elements dropped.
Drop[*list*, −*n*] gives *list* with its last *n* elements dropped.
Drop[*list*, {*n*}] gives *list* with its nth element dropped.
Drop[*list*, {*m*, *n*}] gives *list* with elements *m* through *n* dropped.
Drop[*list*, {*m*, *n*, *s*}] gives *list* with elements *m* through *n* in steps of *s* dropped.
Drop[*list*, seq_1, seq_2, …] gives a nested list
in which elements specified by seq_i have been dropped at level *i* in *list*. ≫

Out[305]= {1, {}, 3, 2, 1, 5, {}, 6, {}, 7, 3, 1, 9}

Out[310]= {1, NUL, 2, NUL, 3, NUL, 4, NUL, 5, NUL, 6, NUL, 7, NUL, 8, NUL, 9, NUL, 10}

Out[311]= {1, NUL, 2, NUL, 3, NUL, 4, NUL, 5, NUL, 6, NUL, 7, NUL, 8, NUL, 9, NUL, 10}

In[312]:=
```
samplelist1 = {1, 3, 2, 1, 5, 6, 7, 3, 1, 9};
samplelist2 = samplelist1^2
Clear[samplelist1]
samplelist1 = samplelist2;
samplelist1
```

Out[313]= {1, 9, 4, 1, 25, 36, 49, 9, 1, 81}

Out[316]= {1, 9, 4, 1, 25, 36, 49, 9, 1, 81}

In[317]:=
```
? Prepend
? Join
Insert[samplelist1, 0, 1]
Prepend[samplelist1, 0]
Join[{0}, samplelist1]
Prepend[samplelist1, 0] == Insert[samplelist1, 0, 1]
Insert[samplelist1, 0, 1] == Join[{0}, samplelist1]
```

Prepend[*expr*, *elem*] gives *expr* with *elem* prepended. ≫

Join[$list_1$, $list_2$, …] concatenates lists or other expressions that share the same head.
Join[$list_1$, $list_2$, …, *n*] joins the objects at level *n* in each of the $list_i$. ≫

Out[319]= {0, 1, 9, 4, 1, 25, 36, 49, 9, 1, 81}

Out[320]= {0, 1, 9, 4, 1, 25, 36, 49, 9, 1, 81}

Out[321]= {0, 1, 9, 4, 1, 25, 36, 49, 9, 1, 81}

Out[322]= True

Out[323]= True

In[324]:=
```
Insert[samplelist1, 0, -1]
Append[samplelist1, 0]
Join[samplelist1, {0}]
Append[samplelist1, 0] == Insert[samplelist1, 0, -1] == Join[samplelist1, {0}]
```

Out[324]= {1, 9, 4, 1, 25, 36, 49, 9, 1, 81, 0}

Out[325]= {1, 9, 4, 1, 25, 36, 49, 9, 1, 81, 0}

Out[326]= {1, 9, 4, 1, 25, 36, 49, 9, 1, 81, 0}

Out[327]= True

In[328]:=
```
Union[{1, 2, 3}, {3, 4, 5}]
Intersection[{1, 2, 3}, {3, 4, 5}]
Complement[{1, 2, 3}, {3, 4, 5}]
Complement[{3, 4, 5}, {1, 2, 3}]
MyComplement[list1_, list2_] := {Complement[list1, list2], Complement[list2, list1]};
MyComplement[{1, 2, 3}, {3, 4, 5}]
```

Out[328]= {1, 2, 3, 4, 5}

Out[329]= {3}

Out[330]= {1, 2}

Out[331]= {4, 5}

Out[333]= {{1, 2}, {4, 5}}

In[334]:=
```
Table[RandomInteger[4], {i, 30}]

ex = Table[RandomInteger[4], {i, 30}] /. {0 → a, 1 → b, 2 → c, 3 → d, 4 → e}
ReplaceAll[Table[RandomInteger[4], {i, 30}], {0 → a, 1 → b, 2 → c, 3 → d, 4 → e} ]
? ReplaceAll
sortandsplitex = Split[Sort[ex]]
Split[Sort[ex]]
frequencyvalue = Length /@ sortandsplitex
Length[#] & /@ sortandsplitex
(Length[#1] &) /@ sortandsplitex
frequencylist = MapThread[{#1, #2} &, {{a, b, c, d, e}, frequencyvalue}]
? MapThread
Transpose[{{a, b, c, d, e}, frequencyvalue}]
```

Out[334]= {3, 2, 0, 1, 1, 1, 2, 2, 0, 4, 0, 3, 1, 3, 2, 1, 1, 4, 4, 2, 0, 0, 4, 4, 2, 0, 4, 0, 2, 1}

Out[335]= {d, d, e, b, c, c, b, b, c, c, e, e, e, c, e, d, c, e, b, d, d, e, c, d, a, b, a, e, b, c}

Out[336]= {e, a, b, e, a, d, c, d, b, c, e, c, b, b, b, e, c, e, c, d, d, e, e, e, c, d, a, e, a, c}

expr /. *rules* applies a rule or list of rules in an attempt to transform each subpart of an expression *expr*. ≫

Out[338]= {{a, a}, {b, b, b, b, b, b}, {c, c, c, c, c, c, c, c}, {d, d, d, d, d, d}, {e, e, e, e, e, e, e, e}}

Out[339]= {{a, a}, {b, b, b, b, b, b}, {c, c, c, c, c, c, c, c}, {d, d, d, d, d, d}, {e, e, e, e, e, e, e, e}}

Out[340]= {2, 6, 8, 6, 8}

Out[341]= {2, 6, 8, 6, 8}

Out[342]= {2, 6, 8, 6, 8}

Out[343]= {{a, 2}, {b, 6}, {c, 8}, {d, 6}, {e, 8}}

MapThread[f, {{a_1, a_2, …}, {b_1, b_2, …}, …}] gives {f[a_1, b_1, …], f[a_2, b_2, …], …}.
MapThread[f, {$expr_1$, $expr_2$, …}, n] applies f to the parts of the $expr_i$ at level n. ≫

Out[345]= {{a, 2}, {b, 6}, {c, 8}, {d, 6}, {e, 8}}

In[346]:=
```
Transpose[{Range[Length[samplelist1]], samplelist1}]
```

Out[346]= {{1, 1}, {2, 9}, {3, 4}, {4, 1}, {5, 25}, {6, 36}, {7, 49}, {8, 9}, {9, 1}, {10, 81}}

In[347]:=
```
MakeADummyCoefficientMatrix[size_] :=
  Module[{tmp1 = {}, tmp2 = {}, tmp3 = {}, tmp4 = {}, tmp5 = {}, i = 1, j = 1},
   While[i ≤ size, tmp2 = RandomReal[{-1.`, 1.`}, i]; tmp1 = Append[tmp1, tmp2]; i++];
   While[j ≤ size, tmp4 = Drop[tmp1, j - 1][[All, j]]; tmp3 = Append[tmp3, Rest[tmp4]]; j++];
   tmp5 = MapThread[Join[#1, #2] &, {tmp1, tmp3}];
   soukan = ReplacePart[tmp5, 1, Transpose[{Range[size], Range[size]}]]]];
MakeADummyCoefficientMatrix[5]
SymmetricalQ[matrix_] := If[Transpose[matrix] == matrix, True, False];
SymmetricalQ[soukan]
SymmetricalQ1[matrix_] := If[MatrixQ[matrix],
   If[Transpose[matrix] == matrix, True, False], Print["It is not a matrix."]];
SymmetricalQ1[soukan]
SymmetricalQ1[{1, 2, 3}]
```

Out[348]= {{1, -0.626117, 0.690885, 0.553006, 0.976262},
{-0.626117, 1, 0.577994, 0.483225, 0.801742}, {0.690885, 0.577994, 1, -0.644286, -0.0500596},
{0.553006, 0.483225, -0.644286, 1, -0.80577}, {0.976262, 0.801742, -0.0500596, -0.80577, 1}}

Out[350]= True

Out[352]= True

It is not a matrix.

In[354]:=
```
? FoldList
FoldList[Plus, 0, {a, b, c, d, e, f}]
```

FoldList[f, x, {a, b, …}] gives {x, f[x, a], f[f[x, a], b], …}. ≫

Out[355]= {0, a, a + b, a + b + c, a + b + c + d, a + b + c + d + e, a + b + c + d + e + f}

In[356]:=
```
FoldList[Times, 1, {a, b, c, d, e, f}]
```

Out[356]= {1, a, a b, a b c, a b c d, a b c d e, a b c d e f}

In[357]:=
```
lengths = RandomInteger[{1, 10}, 10]
flist0 = Drop[FoldList[Plus, 0, lengths], 1]
flist1 = Drop[FoldList[Plus, 1, lengths], -1]
possegment = Transpose[{flist1, flist0}]
```

Out[357]= {2, 7, 3, 7, 4, 10, 9, 9, 6, 3}

Out[358]= {2, 9, 12, 19, 23, 33, 42, 51, 57, 60}

General::spell1 : New symbol name "flist1" is similar to existing symbol "list1" and may be misspelled. ≫

Out[359]= {1, 3, 10, 13, 20, 24, 34, 43, 52, 58}

Out[360]= {{1, 2}, {3, 9}, {10, 12}, {13, 19}, {20, 23}, {24, 33}, {34, 42}, {43, 51}, {52, 57}, {58, 60}}