```
    |\^/|     Maple 9 (IBM INTEL LINUX)
._|\|   |/|_. Copyright (c) Maplesoft, a division of Waterloo Maple Inc. 2003
 \  MAPLE  /  All rights reserved. Maple is a trademark of
 <____ ____>  Waterloo Maple Inc.
      |       Type ? for help.
>  interface(screenwidth=120);
>  with(linalg);
[BlockDiagonal, GramSchmidt, JordanBlock, LUdecomp, QRdecomp, Wronskian, addcol, addrow, adj, adjoint, angle, augment,

    backsub, band, basis, bezout, blockmatrix, charmat, charpoly, cholesky, col, coldim, colspace, colspan, companion,

    concat, cond, copyinto, crossprod, curl, definite, delcols, delrows, det, diag, diverge, dotprod, eigenvals,

    eigenvalues, eigenvectors, eigenvects, entermatrix, equal, exponential, extend, ffgausselim, fibonacci, forwardsub,

    frobenius, gausselim, gaussjord, geneqns, genmatrix, grad, hadamard, hermite, hessian, hilbert, htranspose,

    ihermite, indexfunc, innerprod, intbasis, inverse, ismith, issimilar, iszero, jacobian, jordan, kernel, laplacian,

    leastsqrs, linsolve, matadd, matrix, minor, minpoly, mulcol, mulrow, multiply, norm, normalize, nullspace, orthog,

    permanent, pivot, potential, randmatrix, randvector, rank, ratform, row, rowdim, rowspace, rowspan, rref, scalarmul,

    singularvals, smith, stackmatrix, submatrix, subvector, sumbasis, swapcol, swaprow, sylvester, toeplitz, trace,

    transpose, vandermonde, vecpotent, vectdim, vector, wronskian]

>  with(LinearAlgebra);
[&x, Add, Adjoint, BackwardSubstitute, BandMatrix, Basis, BezoutMatrix, BidiagonalForm, BilinearForm,

    CharacteristicMatrix, CharacteristicPolynomial, Column, ColumnDimension, ColumnOperation, ColumnSpace,

    CompanionMatrix, ConditionNumber, ConstantMatrix, ConstantVector, Copy, CreatePermutation, CrossProduct,

    DeleteColumn, DeleteRow, Determinant, Diagonal, DiagonalMatrix, Dimension, Dimensions, DotProduct,

    EigenConditionNumbers, Eigenvalues, Eigenvectors, Equal, ForwardSubstitute, FrobeniusForm, GaussianElimination,

    GenerateEquations, GenerateMatrix, GetResultDataType, GetResultShape, GivensRotationMatrix, GramSchmidt,

    HankelMatrix, HermiteForm, HermitianTranspose, HessenbergForm, HilbertMatrix, HouseholderMatrix, IdentityMatrix,

    IntersectionBasis, IsDefinite, IsOrthogonal, IsSimilar, IsUnitary, JordanBlockMatrix, JordanForm, LA_Main,

    LUDecomposition, LeastSquares, LinearSolve, Map, Map2, MatrixAdd, MatrixExponential, MatrixFunction, MatrixInverse,

    MatrixMatrixMultiply, MatrixNorm, MatrixPower, MatrixScalarMultiply, MatrixVectorMultiply, MinimalPolynomial, Minor,

    Modular, Multiply, NoUserValue, Norm, Normalize, NullSpace, OuterProductMatrix, Permanent, Pivot, PopovForm,

    QRDecomposition, RandomMatrix, RandomVector, Rank, RationalCanonicalForm, ReducedRowEchelonForm, Row, RowDimension,

    RowOperation, RowSpace, ScalarMatrix, ScalarMultiply, ScalarVector, SchurForm, SingularValues, SmithForm, SubMatrix,

    SubVector, SumBasis, SylvesterMatrix, ToeplitzMatrix, Trace, Transpose, TridiagonalForm, UnitVector,

    VandermondeMatrix, VectorAdd, VectorAngle, VectorMatrixMultiply, VectorNorm, VectorScalarMultiply, ZeroMatrix,

    ZeroVector, Zip]

>  n:=5:seq(i,i=1..n);seq(i^2,i=1..n);
                                                     1, 2, 3, 4, 5

                                                    1, 4, 9, 16, 25

>  DD:=diag(seq(i,i=1..n));
                                                   [1    0    0    0    0]
                                                   [                    ]
                                                   [0    2    0    0    0]
                                                   [                    ]
                                             DD := [0    0    3    0    0]
                                                   [                    ]
                                                   [0    0    0    4    0]
                                                   [                    ]
                                                   [0    0    0    0    5]

>  MDD:=DiagonalMatrix([seq(i,i=1..n)]);
                                                    [1    0    0    0    0]
                                                    [                    ]
                                                    [0    2    0    0    0]
                                                    [                    ]
                                             MDD := [0    0    3    0    0]
                                                    [                    ]
                                                    [0    0    0    4    0]
                                                    [                    ]
                                                    [0    0    0    0    5]

>  ?dim
Multiple matches found:
   LinearAlgebra,Dimension
   Student,LinearAlgebra,Dimension
>  Dimension(DD);# ne marche pas car DD est une matrix, il faudrait faire Dimension(Matrix(DD));
Error, (in LinearAlgebra:-Dimension) expects its 1st argument, A, to be of type {Matrix,Vector}, but received DD
>  Dimension(MDD);
                                                          5, 5

> with(linalg):with(LinearAlgebra):
>  A:=Matrix(3,3,rand(20)-10);
                                                      [-9     0     7]
                                                      [              ]
                                                 A := [-7     6     8]
```

```
                                                      [              ]
                                                      [-5    -2    -9]

>  x:=1;x:='x';x;
                                                         x := 1

                                                         x := x

                                                           x

> A-x;B:=evalm(A-x);
                                                      [-9     0     7]
                                                      [              ]
                                                 -x + [-7     6     8]
                                                      [              ]
                                                      [-5    -2    -9]

                                                 [-9 - x      0        7   ]
                                                 [                         ]
                                            B := [  -7      6 - x      8   ]
                                                 [                         ]
                                                 [  -5       -2      -9 - x]

> C:=Matrix(B);
                                                 [-9 - x      0        7   ]
                                                 [                         ]
                                            C := [  -7      6 - x      8   ]
                                                 [                         ]
                                                 [  -5       -2      -9 - x]

>  A:=Matrix([1]);B:=A;B[1,1]:=2;A;
                                                        A := [1]

                                                        B := [1]

                                                      B[1, 1] := 2

                                                          [2]

> A:=Matrix([1]);B:=A;B:=Matrix([2]);A;
                                                        A := [1]

                                                        B := [1]

                                                        B := [2]

                                                          [1]

# Cas1: A et B sont des pointeurs identiques, donc le contenu de A
# est modifie
#
#cas 2: B:=Matrix... cree un nouveau pointeur, donc le contenu de A
#n'est pas modifi{\'e}.
#############################################
#              RESUME:
#pour passer de matrix vers Matrix: evalm
#pour passer de Matrix vers matrix: Matrix
#############################################
>  f:=(i,j)->if (i=j) then 0 else if (i<j) then a[i,j] else -a[j,i] fi;fi;
f := proc(i, j) option operator, arrow; if i = j then 0 else if i < j then a[i, j] else -a[j, i] end if end if en

>  Matrix(4,4,f); d:=Determinant(Matrix(4,4,f)):
                                     [   0        a[1, 2]     a[1, 3]     a[1, 4]]
                                     [                                           ]
                                     [-a[1, 2]       0        a[2, 3]     a[2, 4]]
                                     [                                           ]
                                     [-a[1, 3]    -a[2, 3]       0        a[3, 4]]
                                     [                                           ]
                                     [-a[1, 4]    -a[2, 4]    -a[3, 4]       0   ]

>  factor(d);
                                                                                        2
                                 (a[1, 2] a[3, 4] - a[1, 3] a[2, 4] + a[2, 3] a[1, 4])

>  Matrix(8,8,a,shape=skewsymmetric);#aussi correct sous maple 7
             [   0        a(1, 2)     a(1, 3)     a(1, 4)     a(1, 5)     a(1, 6)     a(1, 7)     a(1, 8)]
             [                                                                                           ]
             [-a(1, 2)       0        a(2, 3)     a(2, 4)     a(2, 5)     a(2, 6)     a(2, 7)     a(2, 8)]
             [                                                                                           ]
             [-a(1, 3)    -a(2, 3)       0        a(3, 4)     a(3, 5)     a(3, 6)     a(3, 7)     a(3, 8)]
             [                                                                                           ]
             [-a(1, 4)    -a(2, 4)    -a(3, 4)       0        a(4, 5)     a(4, 6)     a(4, 7)     a(4, 8)]
             [                                                                                           ]
             [-a(1, 5)    -a(2, 5)    -a(3, 5)    -a(4, 5)       0        a(5, 6)     a(5, 7)     a(5, 8)]
             [                                                                                           ]
             [-a(1, 6)    -a(2, 6)    -a(3, 6)    -a(4, 6)    -a(5, 6)       0        a(6, 7)     a(6, 8)]
             [                                                                                           ]
             [-a(1, 7)    -a(2, 7)    -a(3, 7)    -a(4, 7)    -a(5, 7)    -a(6, 7)       0        a(7, 8)]
             [                                                                                           ]
             [-a(1, 8)    -a(2, 8)    -a(3, 8)    -a(4, 8)    -a(5, 8)    -a(6, 8)    -a(7, 8)       0   ]

#################################
> f:=(i,j)-> if (i=j-1) then 1 else 0 fi;
                   f := proc(i, j) option operator, arrow; if i = j - 1 then 1 else 0 end if end proc

> J:=n->Matrix(n,n,f);
                                               J := n -> Matrix(n, n, f)

>  diag(evalm(J(2)),evalm(J(3)));
                                                [0    1    0    0    0]
                                                [                    ]
                                               [0    0    0    0    0]
                                               [                    ]
                                               [0    0    0    1    0]
                                               [                    ]
                                               [0    0    0    0    1]
                                               [                    ]
                                               [0    0    0    0    0]

>  with(combinat);
[Chi, bell, binomial, cartprod, character, choose, composition, conjpart, decodepart, encodepart, fibonacci, firstpart,

    graycode, inttovec, lastpart, multinomial, nextpart, numbcomb, numbcomp, numbpart, numbperm, partition, permute,

    powerset, prevpart, randcomb, randpart, randperm, setpartition, stirling1, stirling2, subsets, vectoint]

>  l:=partition(4);
                                  l := [[1, 1, 1, 1], [1, 1, 2], [2, 2], [1, 3], [4]]

> liste:=proc(n)
> local l:
> l:=partition(n);
> [seq(diag(seq(evalm(J(i)),i=j)),j=l)];end proc;
      liste := proc(n) local l; l := combinat:-partition(n); [seq(diag(seq(evalm(J(i)), i = j)), j = l)] end proc

#
> liste(5);
 [0    0    0    0    0]  [0    0    0    0    0]  [0    0    0    0    0]  [0    0    0    0    0]
 [                    ]  [                    ]  [                    ]  [                    ]
 [0    0    0    0    0]  [0    0    0    0    0]  [0    0    1    0    0]  [0    0    0    0    0]
 [                    ]  [                    ]  [                    ]  [                    ]
[[0    0    0    0    0], [0    0    0    0    0], [0    0    0    0    0], [0    0    0    1    0],
 [                    ]  [                    ]  [                    ]  [                    ]
 [0    0    0    0    0]  [0    0    0    0    1]  [0    0    0    0    1]  [0    0    0    0    1]
 [                    ]  [                    ]  [                    ]  [                    ]
 [0    0    0    0    0]  [0    0    0    0    0]  [0    0    0    0    0]  [0    0    0    0    0]

    [0    1    0    0    0]  [0    0    0    0    0]  [0    1    0    0    0]
    [                    ]  [                    ]  [                    ]
    [0    0    0    0    0]  [0    0    1    0    0]  [0    0    1    0    0]
    [                    ]  [                    ]  [                    ]
    [0    0    0    1    0], [0    0    0    1    0], [0    0    0    1    0]]
    [                    ]  [                    ]  [                    ]
    [0    0    0    0    1]  [0    0    0    0    1]  [0    0    0    0    1]
    [                    ]  [                    ]  [                    ]
    [0    0    0    0    0]  [0    0    0    0    0]  [0    0    0    0    0]

>  suite:=n->seq([seq(rank(j^i),i=1..n)],j=liste(n));
                                                              i
                              suite := n -> seq([seq(rank(j ), i = 1 .. n)], j = liste(n))

>  #
> suite(5);
bytes used=4000408, alloc=3341724, time=0.14
 [0, 0, 0, 0, 0], [1, 0, 0, 0, 0], [2, 0, 0, 0, 0], [2, 1, 0, 0, 0], [3, 1, 0, 0, 0], [3, 2, 1, 0, 0], [4, 3, 2, 1, 0]

> quit
bytes used=6258432, alloc=3472772, time=0.18
```