Array processing with J.

!: - FOREIGN

```
2/ Data-types : let be a=.'ananas' and f=.+&7@*: then 4!:0 is
      'a';'f';'gaga';'__haga_'
       equals   0 3 _1 _2
       meaning: 2 not valid
               _1 unused
                0 noun
                1 adverb
                2 conjunction
                3 verb
                6 locale
```

```
 3/ Let be f=.*-+ then 5!:1 <'f' results in a 3, meaning we have fork..
    the listing ..
    0 : noun , 2 : hook , 3 : fork , 4 : bonded conj , 5 : bident  , 6 : trident
    7 : defined operator(pro-adverb,pro-conj)

 4/ 5!:4 <'f'  ... shows the TREE-form of f
 5/ 6!:2 'a=.f m'…shows the CPU-Time in seconds
 6/ 6!:3 y …creates a delay von y seconds
 7/ 7!:0 …used space
 8/ 7!:2 'a=.f m'…space of an execution
 9/ 9!:14 '' … current J-Version – result is a string...
              Ex: 3.05/1997-10-20/00:03
10/ 9!:12 '' … underlying OS: result is a number
              6 → Windows 32 , 4 → Unix , 5 → Os/2 , 3 → MacIntosh
11/ 9!:11 printing precision: pps =. 9!:11  → pps 12
12/ 4!:55 "erase" (ex: 4!:55 <'x')
13/ 4!:56 "erase all names and locales"
14/ 1!:1  "read"  (Ex: 1!:1 filename)
15/ 4!:3 '' … scripts listing … better overview by ,.4!:3 ''
16/ 7!:4 '' release space
17/ 1!:40'' J application path .. eg: 1!:40'' gives 'd:\j402a\'
18/ 1!:21 <file creates a file – → LOCK
        Result is the file-handle ... ex: 269012640
    1!:22 <FN …UNLOCK
```

1!:20`` creates a 2 column matrix of file handles and names…
Sample:

269012640	h:\j402a\zh1.txt
269012672	h:\j402a\zh2.txt

Column 0 is a number

.
19/ **3!:1 y** Convert to Binary
20/ **3!:2 y** Convert from Binary (inverse to 19/)
21/ [x] **18!:1 y** Name List , example **'sk'** 18!:1 [0 lists all *locales* which begin with letters s or k.

{. *Take*
}. *Drop*
n{ *n-th element or row*

special example: (2 7,1 9,:5 1) { 8 5 6 9 8 7 1 4 5 4 4 returns:

```
6 4
5 4
7 5
```

<"1 *enclose each vector element or matrix row*

```
                    4 2
Example:       m =.3 3   → <"1 m gives [4 2]⊃[3 3]⊃[8 3]
                    8 3
```

{m does the same and is faster than <"1 m !

Scattered indexing:

```
                  3 5 6 7
                  4 0 1 1                              3 2
example: m =.      6 5 4 3    p=.(coordinates) =.  4 1
                  1 3 4 5                              0 2
                  9 8 1 3
```

(<"1 p) { m returns : ***4 8 6*** *or tacit:* ***sc =. (<"1) @:]{***

sample: 77 (<"1 (x,2)$2#i.[x=.#m) }m writes the value 77 into the main-diagonal of the 2-dim matrix m

a) <"0 v *enclosing each element of vector v*
b) </. v *also… version a) is approx 10 times faster*

|: ***Transpose***

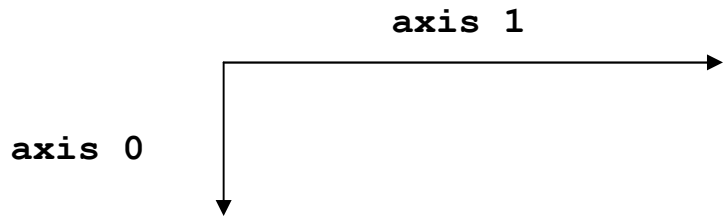
Reversing vectors... `|.v …`

a|: **translation of a to the end** ⇔ `1|:matrix(2d)` returns the matrix itself.
(index origin is 0)

So `0|:matrix(2d)` is the transpose

In case the left argument is boxed then we get an intersection
of the transforms..

Consequence: in case of a 2-dim matrix we obtain by **(<0 1)|:matrix(2d)**
the invariants of original and transpose → main diagonal



example: Is y =. 6 4$'marsrockfirebluepeerfast'

then: (1|:y);(0|:y);y;(<0 1)|:y

mars	mrfbpf	mars	more
rock	aoilea	rock	
fire	rcrues	fire	
blue	skeert	blue	
peer		peer	
fast		fast	

(see also **COMPOSE (&)**)

|. vector reverse
n|. roll
count **resp** number of rows (2-dim)
"repeat each" with left argument: 5#1 2 3 returns 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3

#. ***„base"***. *2 #. 1 1 1 returns 7 (or #. 1 1 1)*
10 #. 2 4 0 returns 240
10 #. i.3 2 returns 1 23 45
#. 3 2 4 returns 20 → basis 2 ($3x2^2+2x2^1+4x2^0$)

base allows to define polynomials...
f =. #.&2 _3 4 7 _3"0

here: f(x) = $2x^4-3x^3+4x^2+7x-3$.
f D1 seems 5% faster than via p. But without derivation, p. is about 10% faster

#: **„antibase"**. #: i.8 returns:

```
0 0 0
0 0 1
0 1 0
0 1 1
1 0 0
1 0 1
1 1 0
1 1 1
```

…→ #: is the i**nverse** to #. (in case of basis 2) ⇔ encode/decode
5 #: 12 19 22 returns 2 4 2 ie. (12 19 22) mod 5

$ *shape.(Result is a vector)*
***** *multiplication or „sign",ie *4 5 _2 3 0 4*
returns 1 1 _1 1 0 1 …also see the norm of complex numbers

_ ***infinity*** *… 10000 < _ returns 1*

^x e^x
α^x α^x
^.z $ln\ z$
b^.z $log_b\ z$
%:z $z^{1/2}$

n%:z $z^{1/n}$
***:z** z^2
+:z $2z$
-:z $z/2$
α-:β $\alpha \equiv \beta$ … *identity*
" **rank** …

Ex 1:
nice application : **^&2 3"0 (1 2 3 4 5)** returns :

```
1   1
4   8
9  27
16  64
25 125
```

or

Ex 2:

Let's look at the following matrix m:

```
0 5 3 3 1 0 4
4 6 2 3 5 0 0
3 4 0 2 0 2 4
4 6 5 3 0 4 2
```

Simultaneous evaluation of the occurrence of the number 4 in column 0, the 3 in the column 3 and the 0 in column 6

So, instead of: (4 = 0{"1 m),(3 = 3{"1 m),.(0 = 6{"1 m) it is more elegant to say: **4 3 0 = "1]0 3 6{"1 m** … and we get:

```
0 1 0
1 1 1
0 0 0
1 1 0
```

Ex 3:

Let be v =. 1 6 4 4 1 0 6 6 8 3 4 7 0 0 4

What we want is: (p1=.0 4 8 12){v , (p2=.1 5 9 13){v ,(p3=.2 6 10 14){v

Can be obtained in one go: **,(p1,p2,:p3){"1 h**

!. *„fit-customize"*
%. *matrix-inversion/division*
<: *≤ …dyad x→x-1 …monad Ex: <:101 returns 100*
>: *≥ …dyad x→x+1 …monad >:99 returns 100*

<. ***„floor"*** *<. 4 4.8 9.2 returns 4 4 9 monad.*
„min" 5 <. 4 6.3 5 2 returns 0 1 0 0 dyad.

>. ***„ceiling"*** *>. 4 4.8 9.2 returns 4 5 10 monad.*
„max" 8 >. 1 6.3 9 8.1 returns 0 0 1 1 dyad.

```
     Ex: „integertest" v =<. v
         The corresponding „hook" would be     h=.=>.
         The corresponding „fork" would be     f=.]=>.

~.   „nub".. ex: ~.1 2 4 2 2 3 returns 1 2 4 3
             ex: (!) give matrix m:

                 17 17 17        17  0  0
                 18 18 19   →    18  19 0
                 9  11 12         9 11 12

                    m               ~."1 m


    Hint: the dimension can change:

                 17 17 17        17  0
                 18 18 19   →    18  19
                 9  11 11         9  11

                    m            ~."1 m

~:   (like ≠ in APL) .. ex :  ~:4 6 5 6 returns 1 1 1 0

~    „evoke" :kind of „execute".

             Ex1: m=.1 2 3 → 'm'~  gives 1 2 3
             Ex2: m=.+/    → 'm'~ 1 2 3 returns 6
```

Ex3: m=./ → *+ 'm'~ 1 2 3 returns 6*

~ *„reflexive"* x f~ y ⇔ y f x
resp. f~ y ⇔ y f y

Ex 1: 2%~i.8 returns 0 0.5 1 1.5 2 2.5 3 3.5

Ex 2: +/~ 1 3 5 returns

```
2 4  6                    1 + 1   1 + 3   1 + 5
4 6  8                    3 + 1   3 + 3   3 + 5
6 8 10 … 1 3 5 +/ 1 3 5 … 5 + 1   5 + 3   5 + 5
```

Ex 3: +/~ 0 0.1 0.2 0.3 0.4 returns

```
0.0 0.1 0.2 0.3 0.4
0.1 0.2 0.3 0.4 0.5
0.2 0.3 0.4 0.5 0.6
0.3 0.4 0.5 0.6 0.7
0.4 0.5 0.6 0.7 0.8 …
```

Interesting for function evaluations .. f(x),f(x+Δx),f(x+2Δx),…,f(x+nΔx)

Ex 4: **Modulo tabel:** modtab =. |/~
then modtab >:i.5 returns:

```
0 0 0 0 0
1 0 1 0 1
1 2 0 1 2
1 2 3 0 1
1 2 3 4 0
```

.** *Ex1:45 *. 60 returns 120* ***(..least common multiple)
*Ex2:(1 0 0 1) *. (0 0 0 1) returns 0 0 0 1 ..* ***log."and"***
*Ex3: *.0j1 returns 1,1.57 ..thus (x,y)* → *(r,ϕ)*
cartesian→polar

***** **norm of a complex number : x+iy ➔ $(x+iy)/(x^2+y^2)^{1/2}$**

+. *Ex1: 45+.60 returns 15* ***(…largest common divisor)***
Ex2: (1 0 0 1)+.(0 0 0 1) returns 1 0 0 1 .. ***log."or"***
Ex3: +. 4j8 returns 4 8

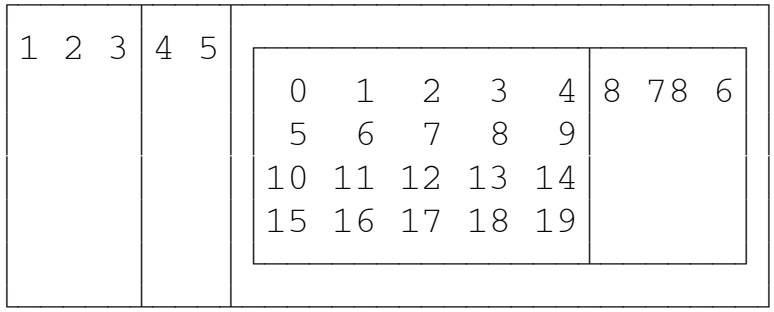
+ *+z returns conj(z) also : + 4j9 returns 4j_9*
➔ *a complex number is real if z=conj(z)*
➔ *iscomplex =.] = +*

-. ***x→1-x***
But also „exclude" : (2 6 8 2 7) -. 6 returns 2 8 2 7
(2 6 8 2 7) -. 2 returns 6 8 7

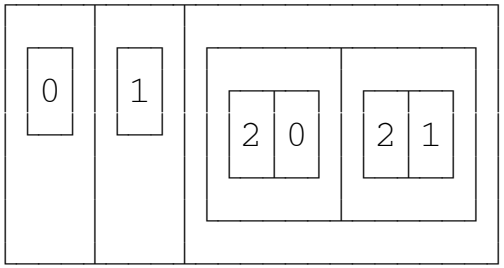
%. ***matrix inversion and division***

{:: ***„Map"***. *Show structure*

Let be d:



Then {::d



; **„link"** a;b;c …enclosing a vector

```
                              2 3 4 4
      In special: Let be  m =. 1 3 2 9…
```

then ;/m equals (2 3 4 4) ⊃ (1 3 2 9) … enclosing rows

Ex: f =. +/ % # g =. +/ and v =. 1 3 8 ➔ **(f;g) v** returns 4 ⊃ 12

„raze" ;(2 1)⊃7⊃(77 78 79 80 81)⊃(55 56) returns : 2 1 7 77 78 79 80 81 55 56
… like a disclose

„multi-assign" : (`a1`;`a2`;`a3`;`a4`) =. 33 21 6 1000

/ *insert* a b c +/ d e

gives: a + d a + e
b + d b + e
c + d c + e

Ex1) special: Δ//a_{ij} (i≤2 and j≤2) - **tensor multiplication**

a_{11} Δ a_{21} a_{11} Δ a_{22}
a_{12} Δ a_{21} a_{12} Δ a_{22} ie… a 2x2 matrix

Ex2) like above, but (i≤3 and j≤2) returns:

$a_{11} \Delta a_{21} \Delta a_{31}$ $a_{11} \Delta a_{21} \Delta a_{32}$
$a_{11} \Delta a_{22} \Delta a_{31}$ $a_{11} \Delta a_{22} \Delta a_{32}$

$a_{12} \Delta a_{21} \Delta a_{31}$ $a_{12} \Delta a_{21} \Delta a_{32}$
$a_{12} \Delta a_{22} \Delta a_{31}$ $a_{12} \Delta a_{22} \Delta a_{32}$

eine 2 x 2 x 2 – Matrix

```
                16 5                    24  30
So if: m =      1 16   then      +//m   39  45
                7 13
                                        13  19
                                        14  28
```

```
16  5  16  5  16  5  16  5  16  5  16  5  16  5  16  5
 1 16   1 16   1 16   1 16   1  6   1 16   1 16   1 16
 7 13   7 13   7 13   7 17   7 13   7 17   7 13   7 13

000    001    010    011    100    101    110    111
```

Ex2) like above but (i≤4 and j≤2):

0000	$a_{11} \Delta a_{21} \Delta a_{31} \Delta a_{41}$	*0001*	$a_{11} \Delta a_{21} \Delta a_{31} \Delta a_{42}$
0010	$a_{11} \Delta a_{21} \Delta a_{32} \Delta a_{41}$	*0011*	$a_{11} \Delta a_{21} \Delta a_{32} \Delta a_{42}$
0100	$a_{11} \Delta a_{22} \Delta a_{31} \Delta a_{41}$	*0101*	$a_{11} \Delta a_{22} \Delta a_{31} \Delta a_{42}$
0110	$a_{11} \Delta a_{22} \Delta a_{32} \Delta a_{41}$	*0111*	$a_{11} \Delta a_{22} \Delta a_{32} \Delta a_{42}$
1000	$a_{12} \Delta a_{21} \Delta a_{31} \Delta a_{41}$	*1001*	$a_{12} \Delta a_{21} \Delta a_{31} \Delta a_{42}$
1010	$a_{12} \Delta a_{21} \Delta a_{32} \Delta a_{41}$	*1011*	$a_{12} \Delta a_{21} \Delta a_{32} \Delta a_{42}$
1100	$a_{12} \Delta a_{22} \Delta a_{31} \Delta a_{41}$	*1101*	$a_{12} \Delta a_{22} \Delta a_{31} \Delta a_{42}$
1110	$a_{12} \Delta a_{22} \Delta a_{32} \Delta a_{41}$	*1111*	$a_{12} \Delta a_{22} \Delta a_{32} \Delta a_{42}$

…thus a 2 x 2 x 2 x 2 matrix (→ 4 dimensions)

```
Ex:                                          27  26
                                             33  32


16   5
 1  16  …=m     then          +//m          42  41
 7  13                                      48  47
 3   2
                                             16  15
                                             22  21

                                             31  30
                                             27  26
```

[,] ***identity..****a[b returns a and a]b returns b*
(i.2 3) +"1(1 2 3) can be expressed as (i.2 3) a +"1 [1 2 3

: ***explicit monad/dyad*** $f_3 = f_{monad} : f_{dyad}$
Ex: g =.^:< … f(y)=.e^y resp f(x,y) =. x < y
g 1 returns 2.71828… and 2 g 4 returns 1.

```
::  „adverse"    u :: v  executes v in case u fails

                 Ex: f =. *: :: +
                      f 3 returns 9
                      f 5 returns 25
                    3 f 5 returns 8 because 3 *: 5 terminates with a valence error
```

<;.2 *part.incl.Delimiter.(Delimiter at the end)*
<;._2 *„ without Delimiter*
<;.1 *part.incl.Delimiter.(Delimiter is at the beginning)*
<;._1 *„ ohne Delimiter*
,: ***„itemize"..****ie. v→1 x v and m x n→1 x m x n*

```
1 4 5 6,3 2 1 5,1 6 7 8,2 3 2 1,:0 0 3 2 returns1 4 5 6
                                                3 2 1 5
                                                1 6 7 8
                                                2 3 2 1
                                                0 0 3 2
```

*Ex: area=.****clean*** *6* ***linsert*** *_3 3 → area is _3 _2 _1 0 1 2 3*
(clean is useful to suppress rounding errors)

```
f1 =. >:@*:           NB. f(x)= x^2+1
f2 =. %:@(5: + ])     NB. f(x)=(x+5)^1/2
f3 =. *&4             NB. f(x)=4x
```

```
    f =. f1,f2,:f3 . then 6.2 ": f area returns the matrix:

                    10.00  5.00  2.00  1.00  2.00  5.00 10.00
                     1.41  1.73  2.00  2.24  2.45  2.65  2.83
                   _12.00 _8.00 _4.00  0.00  4.00  8.00 12.00

;:  „box"..dh ;:'mescal jonathan major' returns 'mescal'⊃'jonathan'⊃'major'
                blank is automatically the delimiter

      al d
    ;:ab t  ..returns
      xy d
```

al	d
ab	t
xy	d

```
,.  v→v x 1 and m x n→ m x n x 1
```

/. „oblique"

1.Ex </. i.3 3 anti-diagonal direction

```
                                                    0 1 2
[0}[1 3][2 4 6][5 7][8] ...       i.3 3 returns     3 4 5
                                                    6 7 8
```

2.Ex

```
        1 2 3 4
For m=. 5 6 7 8 we obtain from </.m the result [1][2 5][3 6][4 7][8]
```

3a) Ex v=.'icbierd'. Then (1 1 3 3 1 3 3) </. v …[ice][bird]

3b) (1 4 3 3 4 3 3) </. v ist [i][ce][bird]

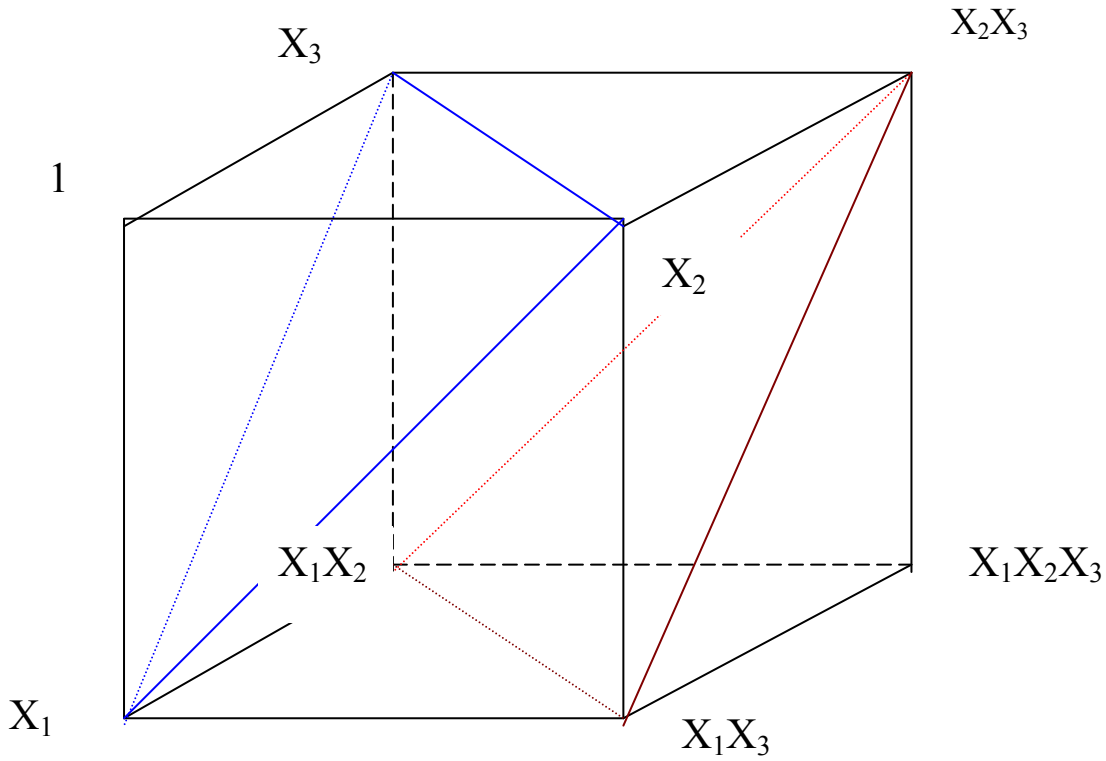
4. ***polynomial multiplication***

Vieta: $(x-x_1)(x-x_2)(x-x_3)(x-x_4)$

$=x^4-x^3(x_1+x_2+x_3+x_4)+x^2(x_1x_2+x_1x_3+x_1x_4+x_2x_3+x_2x_4+x_3x_4)-$
$x(x_1x_2x_3+x_1x_3x_4+x_1x_2x_4+x_2x_3x_4)+x_1x_2x_3x_4$

or

$(x-x_1)(x-x_2)(x-x_3)=x^3-x^2(x_1+x_2+x_3)+x(x_1x_2+x_1x_3+x_2x_3)-x_1x_2x_3$



Special application: **ppr =. +//.@(*/)**

Ex1: p1 ppr p2 … ***multiplication of two polynomials p1 and p2***

Ex2: cfr=: [: ppr/ - ,. 1: … creates the polynomial from the roots

Ex1 : p(x)=2x^2+1 and q(x)=x^2-4x-3 → p(x)q(x)=2x^4-8x^3-5x^2-4x-3

So (2 0 1) ppr 1 _4 _3 returns 2 _8 _5 _4 _3
Resp
(1 0 2) ppr _3 _4 1 returns _3 _4 _5 _8 2

Ex2 : Given the roots 5 _2 3
Then cfr 5 _2 3 returns 30 _1 _6 1
⇔
(x-5)(x+2)(x-3)=30-x-6x^2+x^3

\. *„suffix-outfix":*

*Ex: *\. 1 0 _5 4 returns*

```
 1  0 _1  1
 0 _1  1  0
_1  1  0  0
 1  0  0  0
```

D. derivative:

Ex 1: Let be f(x) = x^3 … thus f =. ^&3 and df =. f D. 1
then df i.3 returns the matrix:

```
  0 0  0 resp

  df "0 i.3 returns 0 3 12
                    0 3  0
                    0 0 12
```

$3x^2$ = f'(x)

Ex 2: $f(x) = x^5-x^3+1$
$f'(x) = 5x^4-3x^2$
$f''(x) = 20x^3-6x$
$f'''(x) = 60x^2-6$

We can define f in 2 ways ..

```
p1 =. 1 0 0 _1 0 1"_ p. [
p2 =. >:@((^&3) * <:@*: ) NB. p1 is 9 times faster than p2

better: 1 0 0 _1 0 1 & p.  (can be used with D.2)
```

```
D1 =. ("0) (D.1)
D2 =. ("0) (D.2)
D3 =. ("0) (D.3)

p11D =. p1 D1  ➔ p11D 3 returns 378
```

There are also other ways to define higher derivations…

```
c =. 1 0 0 _1 0 1

c&p. D1 3 returns 378
c&p. D2 3 returns 522
c&p. D3 3 returns 534

p21D =. p2 D1  ➔ p21D 3 returns 378
p22D =. p2 D2  ➔ p22D 3 returns 522
p23D =. p2 D3  ➔ p23D 3 returns 534

analytic way

p(x) =α0+α1x+α2x^2+α3x^3+α4x^4+…αnx^n
➔
p'(x)= αkkx^k-1 .

pD =. 1: }. ] * i.@#  (polynomial derivative)
or
pD =. }.@(* i.@#)  … playful
pD =. [: }. ] * [: i. # … playful

➔ derivation of matrices: |: pD |:M

∫ p(x) dx = αkx^k+1 1/k+1
```

*pI =. 0: ,] % 1: + i.@# (**polynomial integral**)*

➔ *integration of matrices:* **|: pI |:M**

A. *k A. v defines the **k-th permutation** of vector v*

Ex: 0 A. 1 2 3 returns 1 2 3
1 A. 1 2 3 returns 1 3 2
2 A. 1 2 3 returns 2 1 3
3 A. 1 2 3 returns 2 3 1
etc.
Ie (i.6) A. 1 2 3 gives all permutations as a 6 x 3 matrix

E. *works only for vectors*
Ex1: 3 5 E. 4 21 1 2 3 5 6 3 2 3 6 3 5 returns 0 0 0 0 1 0 0 0 0 0 0 0 1 0
Ex2: 'la' E. 'trallala' returns 0 0 0 0 1 0 1 0

x: ***„extended precision" floating numbers ⇔ rational numbers***

Ex1: x: 5.24 6.125 8.333333333333333 8.0625 returns 131r25 49r8 25r3 129r16
Ex2: separation of nominator from denomitor

2 x: 131r25 49r8 25r3 129r16 returns a 4 x 2 matrix:

131 25

```
 49  8
 25  3
129 16
```

Ex3: %: x: 2.25 1.69 0.25 returns 3r2 13r10 1r2
But: *%: x: 2.25 1.69 0.251 returns 1.5 1.3 0.500999*

x fractions … ***Ex:*** *(3 7 4 3)%2 3 9 2x returns 3r2 7r3 4r9 3r2*

e *„exponential" 2e3 returns 2000. 4e3p5 is* $4000\pi^5$

u. and v. explicit functional arguments

```
ff =: 2 : 0
 u. i.v. )
```

u. is a verb *and v. is a noun.*
Ex: +/ ff 9 returns 6

T. ***taylor approximation:***

Ex: Let be $f(x) = e^{-x^2}$

And so $f(x) \approx 1 - x^2 + \frac{x^4}{2} - \frac{x^6}{6} + \frac{x^8}{24} - \ldots$

(these are the first 9 elements, the odd ones are 0)

f =. ^@-@: and x =. 2%~i.8 … x is 0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5*
f T. 9 x *returns*
1 0.778809 0.375000 0.450684 5.0000000 37.16940000 184.37500000 695.685
Hint:
f x *returns*
1 0.778801 0.367879 0.105399 0.0183156 0.00193045 0.00012341 4.78512e_6
Reasonable convergence from 0.00 bis …1.00

t. ***Taylor-Polynomial:***
The Taylor polynomial we obtain by:

f t. i.n (approx around x = 0)

Ex: f =. ^@-@: Then f t. i.9 equals: 1 0 _1 0 0.5 0 _0.166667 0 0.0416667*

e. *If b =. 2 3$'catbag'. Then b e. 'bag' gives us 0 1 0*
1 1 1

bool<;. 1 *part.acc.boolVector*

(1 1 0 0 1) <;. (2 6 9 9 2) returns [2][6 9 9][2]
bool <;. 1 matrix …enclosing rows!
bool <;. 1 "1 matrix …enclosing columns!

bool<;._1 just also dropping the 1st elements

```
".    Char→Number
":    Number→Char … sample 6.2 „: matrix (of floats) … „thorn" in APL
>./   MAXscan
<./   MINscan
/:    sort..ex1:  (/:matrix){matrix
                      → „fork" f1=./: { ]


                  Ex2:  one other possibility to sort a vector would be by:
                       vector /: vector → ie f2 =. /:~

                   both solutions are equally fast

                  Ex3: monotony:  upmon =. -: /: { ]  or upmon =. -: /:~

                        upmon 3 6 7 111 201 returns 1
                        upmon 3 7 6 111 201 returns 0

\:    sort

<\    partitions.. Ex: <\1 2 3 returns 1/1 2/1 2 3
                 or    <\'efgh' returns ,e'/'ef'/'efg'/'efgh'
<\.                 Ex: <\.1 2 3 returns 1 2 3/2 3/3

NB.       Comment

"_        for const functions: vect=.1 2 4 3"_
```

Then vect 66 gives 1 2 4 3.

sample: We want to define f(n)=2kπi where k∈{0,1,…,n-1}

2 solutions we see immediately:

```
f =. 0j2p1&  * i.
g =. 0j2p1"_ * i.
```

g 3 returns correctly 0 0j6.28319 0j12.5664

Ex2: The ***n-th root of 1***

$1^{1/n} = e^{2k\pi i/n}$, *where* $k \in \{0,1,2,\ldots,n-1\}$.

In J:

```
s =.  [: ^ 0j2p1"_ * i. % ]
```

a. *The ASCII set*

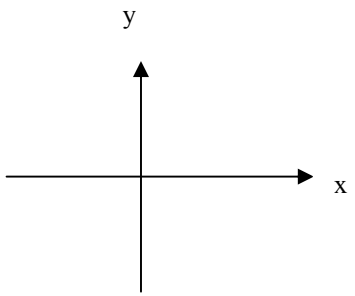
p: *p: n th prime number*
Ex1: p: 0 ist 2
Ex2: p: 1 ist 3
Ex3: p: 0 1 ist 2 3
Ex4: p: 100 ist 547

i. *(2 3 4 0 7 0) i. 0 returns 3*
(0 0 0 0 0 1) i. 1 returns 5
→ *the diadic i. is an* ***„iota"***

Ex:

```
      i. 2 3 is      0 1 2      i. 2 _3 is    2 1 0
                     3 4 5                    5 4 3

      i. _2 3 is     3 4 5      i. _2 _3 is 5 4 3
                     0 1 2                    2 1 0
```



i: dyadic: **"last occurrence"**
Ex: (1 5 5 7 6 3 3) i: 3 returns 6

o. multiple π.
Ex1: o.7 returns 7π
Ex2: toarc =. %&180@o.

j. 5j7 complex numbers : 5+7i

Example 1 : (5 6 7) j. 6 9 8 returns 3j6 4j9 5j87

Example 2 : (2 2$8 2 1 9) j. 2 2$9 8 1 4 returns 8j9 2j8
1j1 9j4

Example 3 : without a left argument: **j. x+iy ➔ −y+ix**

Example 4 : **α+iβ j. x+iy ➔ (−y+α)+i(x+β)**

p xpy ⇔ $(\pi x)^y$

Ex: 4p1 is 4π
Ex: 2p2 is $4\pi^2$
Ex: 2p5 is $(2\pi)^5=32\pi^5$

Ex: 0j2p1 is $2\pi i$
Ex: f(x) = $2\pi i/x$ would be f =. 0j2p1&%

Consequence: $2k\pi i/x$ mit $k\in\{0,1,2,\ldots,n-1\}$: h =. 0j2p1&% * i.

roots =. [: ^ h corresponds to : $e^{\frac{2k\pi i}{n}}$

means the way via xpy is sometimes easier than via o.

a: ***'ace'*** *… boxed empty list : equivalent to <$0*

v1 fn/ v2 *binomial coefficients : binc1 =. i. !/ i.*
or binc2 =. !/ & i.~

v1,"0 v2 *v2 vectors into an N x 2 matrix (or by simply v1,.v2)*

&.> **each-operator.**

Ex: **v =. 2 4 5 ⊃3 3⊃1 9 0 1⊃1 7 6** then **+/&.>v 11⊃6⊃11⊃14**
Ex: 2 4 {. &.> 1 returns 1 0 1 0 0 0

^: _1 ***inverse***

1.Ex: cos ^: _1 (0 0.5 _1 2) returns 1.5708,1.0472,3.1416,0

2.Ex f=.(+&32)@(&1.8) ie* $C^0 \rightarrow F^0$
f 15 25 returns 59 77
g=.f ^:_ 1 is the inverse! …g 59 77 returns 15 25

3.Ex inver =. ^:_1 invf =. f inver

b. **„obverse".**

Ex: f(x)=1+x^3 *f =. >:@^&3 then* ***f b. _1*** *equals* ***3&%:@<:***

]y=. f ^: _ ***limit***

Ex 1: ***newton interpolation:***

Let be: $f=2x^4-3x^3+4x^2+7x-3$ and df =. f D1

$f(x_n)= x_{n-1}-f(x_{n-1})/f'(x_{n-1})$. Because f/f' is a function of x_{n-1}, we can define the fork $x_{n-1}-f/f'(x_{n-1})$: newton =.] - f % df

```
limit =. ^:_
startingpoints =. _2 _1 0 1 2 3
```

newton limit starting-points returns

-1.05… -1.05… 0.367… 0.367… 0.367… 0.367…

p. *Polynomial evaluation*

Ex: $y=4-3x^2+5x^4+x^5$. y(2) is (4 0 _3 0 5 1) p. 2 and returns 104.

```
6 6 8 3
4 7 0 0
4 6 0 3  p. 3 returns 177 25 103 198 65
0 3 6 5
8 7 4 0
```

= *„self-classify"*

```
= 1 3 4 4 3 5 3 1 returns : 1 0 0 0 0 0 0 1 (pos vom 1)
                            0 1 0 0 1 0 1 0 (pos von 3)
                            0 0 1 1 0 0 0 0 (pos von 4)
                            0 0 0 0 0 1 0 0 (pos von 5)
```

=i.5 generates the 5x5 identity matrix

. ***„dot product"***

*a) matrix multiplication: +/ . **
*b) determinant : -/ . **

func ***„train"***

Ex1 : Let be f=.(+4&)@: ie f(x)=4+x²*

```
Then f\1 2 3 4 ius the matrix:
5 0  0  0
5 8  0  0
5 8 13  0
5 8 13 20
```

The function (+/)@(f\) defines the sum: 20 24 26 20

difference between the use of „train" and „suffix":

```
+:\1 2 3 4          +:\.1 2 3 4

2 0 0 0             0 2 4 6
2 4 0 0             2 4 6 0
2 4 6 0             4 6 0 0
2 4 6 8             6 0 0 0
```

Ex2: ***moving average***

```
v =. 5 9 8 7, then as we know …  <\v  [5][5 9][5 9 8][5 9 8 7]
avg =. +/ % #

mavg =. avg \ moving average
mavg v gives 5 7 7.33333 7.25

Hint: 2 mavg v would be the average of sequences of the length 2
(2 <\v is [5 9][9 8][8 7] )
```

special functions with ^: (Power)

a) ***π(n) ie*** *all primes < n* ***p:^:_1 n***
Ex 1: p:i.(p:^:_1) 30 returns 2 3 5 7 11 13 17 19 23 29

b) ***Expand*** *…* ***#^:_1***
Ex 2: f =. #^:_1 … (1 0 0 1 0) f 4 9 gives 4 0 0 9 0

Control-structures:

```
if. T do. B end.
if. T do. B else.   B1 end.
if. T do. B elseif. T1 do. B1 elseif. T2 do. B2 end.

try. B catch. B1 end.
while.  T do. B end.
whilst. T do. B end.

for. B do. T1 end.
```

Example:

```
test15 =: 3 : 0
index  =. >: bag   =. 0
for. i.y. do. bag   =. bag + *:index
                index =. >:index
end.
bag)
```

„test15 20“ runs from index 0 to 19 (i.y.)

```
for_xyz B do. T1 end.
```

```
Example:     test25 =: 3 : 0
                 s =. 0
                 for_xyz. i.y. do.  if. 2|xyz do. continue. end.
                                    s =. s + xyz
                 end.
                 )

                 xyz runs through the range given by i.y.

  select. B
  case. B1 do. T1
  case. B2 do. T2
  case. B3 do. T3
  case.    do. T4 NB. T4 default
end.

Ex1:         test35 =: 3 : 0
                  select. y.
                     case. 'a' do. 'case a'
                     case. 'b' do. 'case b'
                     case.     do. 'case c'
                  end.
                  )

Ex2:         test45 =: 3 : 0
                   tx =. ,:defaultvalue =. 'defaultvalue'
                  select. y.
```

```
   case.  'a'  do. tx=.tx,'case a'
   fcase. 'b'  do. tx=.tx,'case b'
   case.  `c`  do. tx=.tx,'case e'
   case.  'd'  do. tx=.tx,'case d'
   case.       do. tx=.tx,'case c'
end.
)
```

fcase *stands for „further continued case",means in case of test45`b` also the next line is executed*

and so....

```
defaultvalue
case b
case e
```

break.	go to end of while. or whilst. control block
continue.	go to top of while. or whilst. control block
goto_name.	go to label of same name
label_name.	target of goto_name
return.	exit the function

(f g h)α ⇔ (fα)g(hα) ***FORK***
α(f g h)β ⇔ (α f β) g (α h β)

speziell: ([: g h)α ⇔ g h α …!!!≠(g @ h)α

„Cap"... [:

```
                              0  1  4
Ex1:     f=.[: +/ %:  and m=. 9 16 25  then f works like f m ➔ +/%:m ie 3 5 7.
         g=.[: +/ [: +/ %:  …is also a FORK and works like +/+/%:m ie 15

Ex2:      m =. 4 6$'38290' ..so        38290 3
                                       82903 8
                                       29038 2
                                       90382 9

         g =. [: ". (2 3"_) f ] converts columns 2 and 3 in a vector.
                               : 29 90 3 38
                               See also ATOP or AT.

Ex3: g =. +/ > >./ fork checking if the sum is larger than the max
     g 9 8 7 55 _22 1 returns 1, but g  9 8 7 55 _29 1 returns 0.

Ex4: one other fork is the harmonic mean hm.
```

$$h_m := \frac{n}{\frac{1}{a_1}+\frac{1}{a_2}+\frac{1}{a_3}+\frac{1}{a_4}+\ldots+\frac{1}{a_n}}$$

```
hm =. # % [: +/ %
```

Ex5: ***q_m quadratic mean:***

$$q_m := \sqrt{\frac{a_1^2 + a_2^2 + a_3^2 + a_4^2 + ... + a_n^2}{n}}$$

```
qn =.[: %: [: +/ *: % #  or qm =.[: %: ([: +/ *:) % #

or qm =. [: %: # %~ [: +/ *:   NB. it is clear that qm is faster than qn
```

Ex6: f =. ~:&0 %] + =&0 (f is 1/x and 0 is a also a fix-point)
And so f 1 5 0 0 10 →1 0.2 0 0 0.1

([:f [) g ([:h]) ⇔ f(x) g h(y)
(f g)α ⇔ α f g α ***HOOK***
α (f g) β ⇔ α f g β
(f @ g) α ⇔ f(g(α)) ***ATOP conjuction***

Dyadic: ***x (f @ g) y ⇔ f(g(x,y))***

(f @ g)a f is applied on each element of g(a)

Ex1: *v=.1 4 9 16 and* ***f=.(+/)@%:***
f works like ***(+/%:1),(+/%:4),(+/%:9),(+/%:16)*** *and not like*
+/%:1 4 9 16 (…see ***FORK*** *or* ***AT****)*

Ex2: *Let m =. 4 6$'38290' ie*

*38**29**03*
*82**90**38*
*29**03**82*
*90**38**29*

f =. ". @ (2 3&{"1) gives us (29 90 3 38)
But +/ @ ". @ (2 3&{"1) would not bring anything
Better +/ @: ". @: (2 3&{"1) ... returns 160.

(f @: g) α ⇔ f(g(α)) ***AT***

```
Ex:     Let m =. 2 2$2 7 5 9.
        Let f =. +/ @: *:  Then f m returns 29 130
        Let g =. +/ @: +/ @: *:  …g m returns 159

Hint:   [: +/ [: +/ *: faster than g!
```

Linearity check... f is linear if **f@:+ -: f@[+ f@]** holds

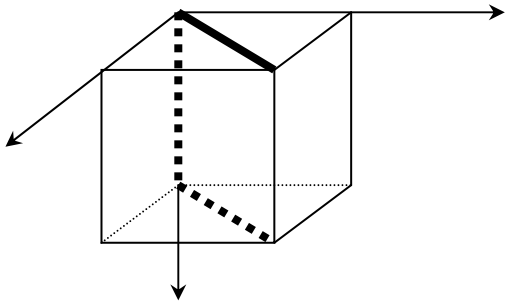
x (f & g) y ⇔ g(x) f g(y) ***COMPOSE (&)***

resp

u & n1 y ⇔ y u n1 … so ^&3 (1 2 3 4 5) is equivalent to (1 2 3 4 5)^3
n2 & v y ⇔ n2 v y … so 10&^. (1 100 59) returns 0 2 1.77085 …(decadic log)

Ex: f=.-&:* *x f y : (x,y)→x^2-y^2*
Ex: f=(:)&(+:)* *x →$4x^2$ ...same as (*:)@(+:)*
Ex: in special…diag=.(< 0 1)&|: *main diagonal of a 2-dim matrix*

Ex: (<1 2)&|: m …main diagonal plane of a 3-dim matrix.



```
Ex:      0  1  2
         3  4  5
         6  7  8

         9 10 11
        12 13 14
        15 16 17

        18 19 20
        21 22 23
        24 25 26   …  (<1 2)&|: m returns:    0  4  8
                                              9 13 17
                                             18 22 26
```

x (f &. g) y ⇔ g^{-1} (g(x) f g(y)) *UNDER (&.)*

Ex: f =. +&.%: (x,y) → $(x^{1/2}+y^{1/2})^2$
Ex: f =. -&.: (x,y) → $(x^2-y^2)^{1/2}$ NB: %:@(- * +) is faster*
Ex: f =. +&.^ (x,y) → $ln(e^x+e^y)$

Hint: commutativity of f and g: if (f &. g) ≡ f

x (f &: g) y ⇔ g(x) f g(y) *APPOSE (&:)*

Exceptions: Ex: a =. 'abcd';'efgh' and b =. 'ABCD';'EFGH'

a ,&> b returns a 2x8 matrix:

abcdefgh
ABCDEFGH *…**compose***

a ,&:> b returns the 4x4 matrix:

abcd
efgh
ABCD
EFGH

$f_0`f_1`f_2`\ldots`f_{n-1}$ @. g x *GERUND/AGENDA*

⇔

= f_k(x) if g(x) = k. If k∉{0,1,…,n-1}➔index error!

Ex: (+&8)`(&11) @.]0 returns 8*
Ex: (+&8)`(&11) @.]1 returns 11*
Ex: (&4)`(+&5)`(*&_9) @.]2 returns _18*
Ex: 44:`] @. ]0 returns 44…resp (]&44)`] @.]0 returns 44
Ex: 44:`] @.]1 returns 1

Ex: f(x) = x^2 for x>0 else f(x) = -x → f =. -`: @.(>&0)*

13 : ***tacit definition*** *… Ex: ft =. 13 : `(+/y.) % (#y.)`*
3 : ***explicit definition*** *… Ex: fx =. 3 : '(+/y.) % (#y.)'*

f=.'(:x.)+(*:y.)'* ***explicit to tacit*** *ft =.13 : fx*

1: … **9:** **constant functions ...** *f=1,…,f=9*

```
Ex:  f=.-&3                f:x→x-3
Ex:  f=.4&-@%:             f:x→4-x^1/2  resp. (x,y)→4-y^1/x
Ex:  f=.^&2                f:x→x^2
Ex:  f=.10&^.              f:x→log10 x
Ex:  f=.3&^                f:x→3^x
Ex:  f=.([: ][)^([:(+&3@-)])      f:(x,y)→x^3-y
Ex:  f=.(_3&%:)@(+&1)      f:x→(1+x)^-1/3
Ex:  a]b returns b and a[b returns a
Ex:  f=.(%:)@+             f: (x,y)→(x+y)^1/2
Ex:  f=. *:@+              f: (x,y)→(x+y)^2
                              x→x^2
Ex:  f=.(%:)@*             f: (x,y)→(xy)^1/2
Ex:  (>:&2)*.(<:&9)        f(x)=1 if in {2,..,9},f=0 else.”interval-test”
Ex:  f=.([:*:[)+([:*:])    f:(x,y)→x^2+y^2
     better: +&*:
Ex:  f=.(%:)@-&*:          f:(x,y)→(x^2-y^2)^1/2 …better : -&.*:
Ex:  f=.+*-                f:(x,y)→(x-y)(x+y)
Ex:  f=.[*<:               f:x→x(x-1)
Ex:  f=. ] * &(o.%180)     f:x→πx/180 (deg → arc)
Ex:  f=.[+*:               f:x→x+x^2
Ex:  f=.+*:                f:(x,y)→x+y^2
Ex:  f=.[ * %:@]           f:(x,y)→xy^1/2
```

```
Ex:  f=.{:@/:              …position of the last max
Ex:  theoretical example: ⊗/x1,x2,x3,x4 equals x1⊗(x2⊗(x3⊗x4))
```

Amending lists ..

i) **Write/Read ⇔ }/{**

```
    Ex: v =. 2 5 7 8

    v =. 13 (1)} v          ➔ 2 13 7 8
    v =. 14 15 (0 3)} v     ➔ 14 5 7 15 ..(0 3){v ist 14 15

          2 3 5 5
    Sei m = 1 5 0 6
          1 9 9 9
                          2 3 5 5
    m =. 7 (<1 3)} m ➔    1 5 0 7 .. (<1 3){m is 7
                          1 9 9 9
                                          2 3 5 5
    m =. (2 2$3 7 2 3) (<1 2;0 1)} m  ➔   3 7 0 6
                                          2 3 9 9
    but

                                          2 8 5 5
    m =. (2 8) (1 2;0 1)} m  ➔            1 5 2 6
                                          1 9 9 9
```

```
m =. 7 (2)} m                    ➔
                                   2 3 5 5
                                   1 5 0 6
                                   7 7 7 7

m =. 7 (2)}"1 m                  ➔
                                   2 3 7 5
                                   1 5 7 6
                                   1 9 7 9

m =. (4 5 1 3) 1} m              ➔
                                   2 3 5 5
                                   4 5 1 3
                                   1 9 9 9

: (3 4 5) 2}"1 m  …              ➔  Rank Error!

(3 4 5) (<(i.3);2) } m           ➔
                                   2 3 3 5
                                   1 5 4 6
                                   1 9 5 9
```

Read/Write work the same

ii)

```
last row ..          _1{.m or {:m
last column ..       _1{."1 m or {:"1 m
2 (<:#v)} v          last element becomes 2
```

LOCALES

```
Child locales:  base   j    laserjet  …

Parent locale:  Z locale

Ex: 'n' namelist_z_ 3 lists all functions of the Z-locale,
    starting with an "n"

default is the base locale

listing locales: namelist 6
listing variables: namelist 0 3 resp namelist__ 0 3
direct definition: x_temp_=.1 7 9 (variable x in locale temp)
searching for names is a locale:
'an'  nl_temp_ '' NB. Search for elements in temp, starting with 'an'
'*an' nl_temp_ '' NB. Search for elements containing 'an'
or we can search selectively
nl=selective namelist

erasing a locale „temp": clear 'temp'
loading into a locale … scriptread_temp_ <'main\someutils.js'
```

```
special example:

calc_def_ =. 3 : 0 NB. Definition of „calc“ in the locale „temp“
len =. 15          NB. defining the variable len within the locale „temp“,

MAX =: 100         NB. Defining the global variable MAX in the locale „temp“
NDX_util_=: 11     NB.
)                  NB. end

A variable in the base locale can be defined by abc__ =. 34  or abc =. 34
```

SCRIPTS

<dates,convert,misc,files,format,numeric,strings,validate,jfiles,debug>

a) load‘dates‘

calendar,todate,todayno,tsdiff,tsrep,tstamp,valdate

```
calendar year,month(s)    Ex: calendar 1998 4 5  (displaying may and april)

[opt] todate daynumbers   number of day relative to 1.1.1800.

                          Ex: todate 44249 returns 1921 2 25
                          Ex: 0 todate 49666 returns 19351226
```

[opt] ***todayno*** *dates*	inverse to *„todate"* Ex: **todayno 1898 4 30** returns 35913 Ex: **0 todayno 2 3$1833 10 30 1899 2 2** returns: 12355 36191 Ex: **1 todayno 19590804** returns 58288
tsdiff	days between dates Ex: **1959 04 08 tsdiff 1833 10 30,:1899 2 2** returns 125.441 60.1831
tsrep	converts the timestamp(6!:0) into a number. Ex: **":!.13 tsrep 1998 3 24 15 6 393.347** returns : 6255414753347 inverse: **1 tsrep 6255414753347**
tstamp	converts the timestamp(also 6!:0) into a formatted form. Ex:**tstamp 6!:0 ''** returns 24 Mar 98 15:57:05
valdate	validates a date,and returns 1 for a valid date. Ex: **valdate 1996 2 29,:1996 2 30** gives 1 0

„todayno" and „todate" take also leap years into consideration

b) load‘convert‘

av,detab,dfh,hfd,hex,mfh,tolower,toupper,vfm

av converts to ASCII and the other way round
Ex: **av'abcABC+'** returns the vector
97 98 99 65 66 67 43
Ex: **av 97 98 99 65 66 67 43** returns 'abcABC+'

detab deletes the TAB
Ex: **detab'aga',TAB,'memmnon'** returns 'aga memmnon'
..no left argument

dfh,hfd HEX → DEZ → HEX
dfh ***hfd***

00f
Ex: **dfh 010** returns 15 16 333
14d

00f
Ex: **hfd 15 16 17** returns 010
14d

hex operations on HEX numbers
Ex: **'71' - hex'2'** returns 6f
Ex: **'7' * hex'222'** returns 'eee'
Ex: **'85' + hex'222'** returns '2a7'

mfv cut.. where blank is the default
Ex: **'/' mfh 'this/is/a/test'**

```
returns:     this
             is
             a
             test
```

Ex: **mfh 'Easter Egg'**

```
returns :    Easter
             Egg
```

Ex: **mfh 'an Easter Egg'**

```
returns:     an
             Easter
             Egg
```

tolower,toupper a → A → a
toupper tolower

Ex: **tolower'uSsR'** returns 'ussr'

vfm transforms a CHAR-matrix into a vector with a LF delimiter

```
        dogs
Let a =. star a 3x4 Matrix.
```

```
                    Carl
                                                    dogs
Then vfm a is a string of the form                  star
                                                    Carl
and has the length 14
```

c) load‘misc‘

boxcols,bx,(? assert),chop,default,diff,join,nubcount,pathname,show

boxcols

```
enclosed packaging
synthax: partition boxcols matrix
Ex1 : 3 boxcols i.3 7 returns

 0   1   2 *     3  4  5 *     6
 7   8   9 *    10 11 12 *    13
14  15  16 *    17 18 19 *    20

1 0 1 0 0 0 1 boxcols i.3 7 returns

 0   1 *  2    3   4   5  *  6
 7   8 *  9   10  11  12  * 13
14  15 * 16   17  18  19  * 20
```

bx occurrence of the 1’s Ex: bx 0 0 1 1 0 1 returns 2 3 5

chop enclosing according to delimiter (default is blank)

Ex1: **chop 'tick trick track'** returns
'tick' ⊃ 'trick' ⊃'track'

Ex2: **'*' chop 'tick trick*track'** returns
'tick trick' ⊃ 'track'

Ex3:

'x' chop 'abxcdxefx' returns 'ab'⊃'cd'⊃'ef'
'x' chop 'abxcdxef' returns the same

default defines a global default name

diff delta of order 1
diff 3 6 10 11 1 returns 3 4 1 _10

works also for matrices

```
     0 0 4 6          0 3 _4 3
diff 0 3 0 3 returns: 6 2  8 4
     6 5 8 7
```

join Ex1: **join 'cssr'⊃'sssr'⊃'zssr'**

```
          csz
returns:  sss
          sss
          rrr
```

nubcount absolute distribution

```
                                    l a m b d
Ex1: |:nubcount'lambada' returns    1 3 1 1 1

          sas
          uhu          sas 3
Ex2: nubcount uhu  returns  uhu 2
          sas
          sas
```

pathname split path + file
Ex: **pathname 'c:\j305\profile.js'**
(c:\j305\)⊃(profile.js)

show a namelist

Ex: **show 3** …show all functions in jx

```
f:1 2 0 1 "_ p. ] NB. f(x)=1-2x+x^3
g:+&4@*           NB. f(x)=4+x^2
h:(>:&6)*.(<:&16) NB. Test 6 ≤ x ≤ 16
s:f&h
```

Ex: **show 2** …show all variables and functions

```
a:i.9
b:i.99
f:1 2 0 1 "_ p. ] NB. f(x)=1-2x+x^3
g:+&4@*           NB. f(x)=4+x^2
h:(>:&6)*.(<:&16) NB. Test 6 ≤ x ≤ 16
s:f&h
v:6 3 2 1 3…alphabetically ordered
```

d) load‘files‘

fread,fappend,fappends,fdir,ferase,fexist,freadr,freads,freplace,fsize,fss,fselect,fview,fwrite,fwrites

fread

Ex1:**file=.fread ‘c:\j305\glsdata55‘ or**
fread <‘c:\j305\glsdata55‘

Ex2: sequential read:
fread <‘c:\j402a\mega.txt‘;startposition,length

Ex3: **‘b‘ fread ‘c:\j305\test.txt‘**
|1st row|2nd row|3rd row| …

Ex4:**‘m‘ fread ‘c:\j305\test.txt‘**

```
{resp :'m' fread 'c:\j305\test.txt';a,len)
```

Each CRLF triggers a new line

Ex4:**'s' fread 'c:\j305\test.txt'**

Ex5:**'m' fread 'c:\j305\test.txt';3 11**

freadr Ex1:**freadr 'c:\j305\snoopy.txt'**

output as matrix ..

Ex2:**freadr 'c:\j305\snoopy.txt';2 5**

Reading 5 rows starting with row number 2

freads …is equivalent to 's' fread …

fappend Ex: **(2 4$'abcddcba') fappend 'c:\j305\x.txt'**
appends here 8 characters to the file. File will be created if not existent. Result is 8 here

(7 33 12) fappend … returns the errormessage _1

fappends Ex: **(2 4$'ggggfrfr') fappends 'c:\j305\y.txt'**

like fappend, but adds a CRLF

fdir Ex: **fdir 'c:\j305\uh*.txt'**
wildcard search
Result:

```
uhu.txt |1998 4 18 22  9 12| 55| rw- | -----a
uhsc.txt|1998 4 18 22 17  8| 23| rw- | -----a

name    |changed |last |time|bytes|permission|
```

ferase Ex: **ferase 'c:\j305\uhu.txt'**

freplace Ex1:**'abc' freplace 'c:\j305\bise.txt';4**
substring replacement on file-contents

fsize Ex: ***fsize 'c:\j305\new.txt'***

fss Ex: **'ak' fss 'c:\j305\file.txt'**
file string search

fselect Ex: **fselect 'c:\j305*.txt'**
listing

fview Ex: **fview 'c:\j305\marcy.txt'**

fwrite Ex1:**('1984',CRLF,'jump') fwrite 'c:\j305\marcy.txt'**
replacement of the contents (old data erased)

fwrites Ex1: **('1984',CR,'jump') fwrites 'c:\j305\marcy.txt'**
similar, just with CRLF

fwrite+fwrites echo the number of bytes

e) load'format'

center,clipfmt,colhdr,expand,expandn,flatten,fmt,fold,hexdump,nfmt,ruler,sqzint,sqzrun,table

center Ex: **13 center >'san';'antonio'**

xxxxx**san**xxxxx
xxx**antonio**xxx … x=blank

clipfmt Ex: **clipfmt i.2 3**

colhdr Ex: c =.'name,firstname;tel,age,id;car,company'

(12 9 22;3) colhdr c

width…	justify…	0=center header,then rightjustify
		1=center header in width
		2=right justify
		3=left justify

```
name        tel     car
firstname   age     company
            id
```

expand

```
Ex1:1 1 0 0 1 0 expand 13 4 1
 13 4 0 0 1 0   …analog „\" in APL
```

expandn

```
Ex1:4 expandn 'starmarsduke'
    'star mars duke '

Ex2:2 expandn matrix

matrix is 6 x 3
                lan
                cap
                lan
                tom
                cat
                sgb


    Resultat:   lan … 9x3
                cap
                    … row of blanks
                lan
                tom
```

```
    … row of blanks
cat
sgb
    … row of blanks
```

flatten Ex: **flatten matrix**

```
optically:      lan
                cap
                lan
                tom
                cat
                sgb

in reality a vector
```

fmt m a 2x4 float

```
0 1       2       3
4 5 _3.5432 19222.2

(c = commafill-format)
'c8' fmt m … 0       1       2       3
             4       5      -4  19,222  … 2x32
(z = zerofill-format)
'z8' fmt m … 00000000000000010000000200000003
             00000004000000005-000000400019222… 2x32
```

```
(b = blankfill-format)  …zeroes become blanks, too
'b8' fmt m …          1       2       3
              4       5      -4   19222 … 2x32

(xn = filling n blanks,between the blocks)

'z8,x2' fmt m … 00000000  00000001  00000002  00000003
                00000004  00000005  -0000004  00019222
                ... 2x38 no blanks at the end

'1r11.4' fmt m … 0.0000     1.0000     2.0000     3.0000
                 4.0000     5.0000    -3.5432 19222.2000

 → 2x44 matrix

'z3,x1,z4,x1,2r11.4' fmt m …
000 0001      2.0000     3.0000
004 0005     -3.5432 19222.2000

column 0 → z3 formatted
column 1 → z4 formatted
columns 2 & 3 → 2r11.4 formatted
```

fold

```
Ex1: 7 fold 'mid life crisis'
    mid
    life
    crisis  … a 3x7 Matrix
```

```
Ex2: 8 fold 'mid life crisis'
     mid life
     crisis … a 2x8 matrix

Ex3: 5 fold 'mid life crisis'
     mid
     life
     s
     crisi  … a 4x5 matrix.
```

Hexdump **hexdump fread 'c:\j305\marcy.txt'**

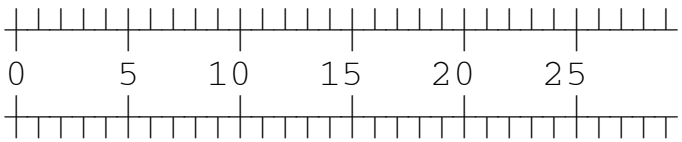
```
0  43 68 61 72-6C 79 0D 0A  42 72 6F 77-6E 0D 0A     |Charly..Brown.. |
…as character
```

nfmt

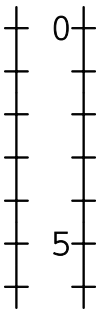
```
Ex1:6 nfmt matrix
    matrix is :  _9.5 59.5 26.5
                   35    0  _19
    returns…
    [0]  _9.5   59.5   26.5
    [1]  35.0    0.0  _19.0 … a 2x45 Matrix.
```

ruler

ruler 30 …horizontal



1 ruler 7 …vertical



sqzint

Ex: **sqzint 13,14,15,(44+i.109),51555+i.8**
Returns: 13-15,44-152,51555-51562…a summary (as string)

sqzrun

Ex: **sqzrun 13 13 13 15 15 44 9 9 9 9 9 9**
Returns: 3#13,2#15,44,6#9…absolute distribution
(as string)

table

Ex1:**fn table vector**… "vector fn vector"

| table 1+i.4 … modulo-table: x mod y

	1	2	3	4
1	0	0	0	0
2	1	0	1	0
3	1	2	0	1
4	1	2	3	0

Ex2: **(+&*:) table 1++:i.5** …f(x,y)=x^2+y^2

	1	3	5	7	9
1	2	10	26	50	82
3	10	18	34	58	90
5	26	34	50	74	106
7	50	58	74	98	130
9	82	90	106	130	162

f) load‘numeric‘

baserep,clean,colsum,fraction,groupndx,linsert,randomize,range,recur,round,rounddist

baserep

```
Ex1:16 baserep 258
    returns 1 0 2  … also 1x16^2+0x16^1+12x16^0

Ex2:100 baserep 44512345
    returns 44 51 23 45
```

clean

```
Ex1:clean 1e_11 _1e_7 1e_14 … returns 0 _1e_7  0
    (assigning 0 for all x < 1e_10)
```

colsum

```
Ex1:     m : 1 7 4 5 0 5 1 0 1
             2 0 6 6 0 4 3 1 1
             9 3 5 8 0 3 4 1 0
             0 0 5 6 0 9 4 0 0
             0 3 0 4 0 9 3 1 1
             1 3 2 5 6 5 2 0 1

      then 7 colsum m  :   2 10 11 16 6 19  7 0 2 …
                          11  6 11 18 0 16 10 1 2

      column 7 is the partitioning key
```

Ex2: **7 8 colsum m** (columns 7 and 8 are the partitioning key)

```
0  0 5  6 0  9 4 0 0
9  3 5  8 0  3 4 1 0
2 10 6 10 6 10 3 0 1
2  3 6 10 0 13 6 1 1
```

fraction Ex: **fraction 0.125 0.75 4.8 0.33 _8.625**
(numeric values to fractions)

```
1 3 24  33 _69
8 4  5 100   8 … a 2x5 matrix
```

groupndx Ex: **5 7 9 groupndx i.14 (allg.:** *groups* **groupndx** *data***)**
returns … _1 _1 _1 _1 _1 0 0 1 1 2 2 2 2 2

```
    _1         1
 ------→      -→
(0 1 2 3 4 5 6 7 8 9 10 11 12 13)
           -->     ----------→
            0           2
```

linsert Ex1:**3 linsert 2 5 77** … returns **2** 3 4 **5** 29 53 **77**
linear interpolation

works also for matrices ...

Ex2:**3 linsert "1 matrix** … when

```
matrix:   1 10 91
         17 47 59

result: 1  4  7 10 37 64 91
       17 27 37 47 51 55 59
```

Ex3: Linear interpolation on interest curves

$T_0 = 3.5\%$ $T_{29} = 3.8\%$ $T_{59} = 4.1\%$ $T_{179} = 4.5\%$ $T_{359} = 4.97\%$

(**diff** 0 29 59 179 259) **linsert** 3.5 3.8 4.1 4.5 4.97

Returns a vector of length 360

or as a fork: f =. diff linsert [

randomize Ex: **randomize 0**

random seed (default is 16807)

range Ex1:**range 3 19 4** … last element defines the metrics (jumps)

result: 3 7 11 15 19

Ex2:**range 3 19 3**

result: 3 6 9 12 15 18

recur **m recur a.** solving : $\mathbf{r_i = a_i + r_{i-1}\ m_{i-1}}$

Ex: **0.05 0.06 0.03 recur 100 120 140 155**
returns 100 125 147.5 159.425

$a_0 = 100$, $a_1 = 120$, $a_2 = 140$, $a_3 = 155$
$m_0 = 0.05$, $m_1 = 0.06$, $m_2 = 0.03$.

$r_0 = 100 + 0 \quad \times 0 \quad = \mathbf{100}$
$r_1 = 120 + \mathbf{100} \quad \times 0.05 \quad = \mathbf{125}$
$r_2 = 140 + \mathbf{125} \quad \times 0.06 \quad = \mathbf{147.5}$
$r_3 = 155 + \mathbf{147.5} \quad \times 0.03 \quad = \mathbf{159.425}$

round Ex: **0.05 round 11.43 97.18 7.33**
returns 11.45 97.2 7.3 (0.05 is the tolerance)

Ex: **0.1 round 11.43 97.18 7.33** returns 11.4 97.2 7.3

rounddist Ex: **0.1 rounddist 7.44 7.44 7.44 7.46**

returns: 7.5 7.4 7.4 7.5 ... distributed rounding logic

g) load'strings'

charsub,cuts ,deb,delstring,dropafter,dropto,ljust,rjust,rplc,ss,takeafter,taketo

charsub Ex: **'aooidv' charsub 'jahn solder'**
replace a by o, o by i, d by v.

result: *john silver*

cuts Ex1:**'gus'< cuts 1 'ragulin gusarov jakushev guseev'**
result:

ragulin gus

Ex2: **'gus'#cuts 1 'ragulin gusarov jakushev guseev'**
result: 11

deb Ex: **deb ' tasley morgan israel '**
delete excess blanks

result: '*tasley morgan israel*'

delstring Ex: **'tom' delstring 'tommorgan tomrichardon'**

```
                result: `morgan richardon`

dropto/         Ex: 'long' dropto 'john long silver' …`long silver`
dropafter       Ex: 'long' dropafter 'john long silver'
                    …`john long`

taketo/         Ex: 'long' taketo 'john long silver'…'john'
takeafter       Ex: 'long' takeafter 'john long silver` …'silver'


ljust/rjust     …left or right justifying text…

rplc            string rplc old;new

                Ex: 'he is a columnist' rplc 'lum';'mmu'
                'he is a communist'

ss              Ex: 'hu' ss 'joshua flint hunter'
                string search … `hu` … 3 and 13
```

h) load‘validate‘

inrange,isboolean,isboxed,ischaracter,isdate,isinteger,isnumeric

```
inrange         Ex: 3 17 inrange 4 7 11 13
                elements in range 3 - 17
```

```
isboolean
isboxed
ischaracter
isinteger
isnumeric       isnumeric 3 1 2r9 1.1 0 4j10 returns 1

isdate          …a date-check: isdate 1996 2 30 returns 0.
                               isdate 1996 2 29 returns 1
```

i) load‘jfiles‘

jcreate,jread,jappend,jsize,jreplace,jdup,jerase

```
jcreate       jcreate‘hawkins‘   … creates the J-file ‘hawkins‘
                                   result is 1

jappend       ('israel';'hands') jappend 'hawkins'
              0 1

              ('john';'long';'silver') jappend 'hawkins'
              3 4

              'tom morgan' jappend 'hawkins'
              5

jsize          jsize 'hawkins'
```

0 5 1184 0

jread *jread 'hawkins';1 3 5 NB. selective read*

hands	long	tom morgan

jread 'hawkins';i.2 2 *NB. incl formatting*

israel	hands
john	long

jdup ***'f2' jdup 'f1'***

NB. Copies file f1 to f2, f2 will be created if not yet
NB. existing. Jdup echoes a result (number of components)

jreplace ***('tasley';'james branden';'hunter') jreplace 'hawkins';1 2 3***

```
jread 'hawkins';i.6
```

israel	tasley	james branden	hunter	silver	tom morgan

jerase

```
jerase 'hawkins' NB.

>jread 'jim';i.6 NB. 6 x 10 matrix

israel
hands
john
long
silver
tom morgan
```

We can also work with file-handles

```
handle =. jopen_files_ `jim`
… jread h;1 2 3
jclose_files_ handle NB. result is 1 if successful
```

j) load‘debug‘

```
testfu =: 3 : 0
```

```
   t0 =. y.
   t1 =. 1 + w
   t2 =. 5*t1
   t3 =. t2 - 10 ← row 4 ...program should "crash" here
   t4 =. *:t3
   t5 =. t4 + 1959
)
```

a) 13!:0]1 … activating the debugger
(we can check via 13!:17 `` if debugger is set ... if yes 1 is the result)

b) 13!:3 'testfu 4:*' (4 = monadic, line 4 /* = dyadic means at each line)

c) testfu 5 NB.executing

debugger stops with the message:

```
stop: testfu
       t3=.t2-10
testfu[4]
```

Local variables t0,t1,t2 (t0=5,t1=6,t2=30) visible now.

13!:4 `` jump to location counter
13!:5 `` jump to location counter + 1

We can trace a whole function by: **13!:3 'testfu *:*'**

13!:4 `` jumps to the next line

13!:18 `` :Returns the message as matrix!

example:

```
stop
      t2=.5*t1
testfu[3]
      testfu 5
```

13!:12 `` :last error-message as a vector

Ex 13!:12 `` returns:

```
stop
      t2=.5*t1
testfu[3]
      tes
```

13!:11 `` returns the last error number