J Reference Card for version 6.02

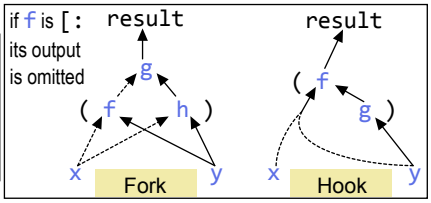
Arithmetic Dyads

2 + 8	Plus	10
2 - 8	Minus	_6
2 * 8	Times	16
2 % 8	Divide	0.25
2 | 8	Residue	0
2 ^ 8	Exponent	256
2 ^. 8	Log	3
3 %: 8	Root	2
2 >. 6	Greater	6
2 <. 6	Lesser	2
36 +. 24	GCD	12
36 *. 24	LCM	72
10 10 #. 8 3	Base	83
10 10 #: 83	Antibase	8 3
2 ! 8	CombOutOf	28
2 ? 8	Deal	4 2

Arithmetic Monads

<. 4.5	Floor	4
>. 4.5	Ceiling	5
* _4 0 3	Sign	_1 0 1
! 4	Factorial	24
? 20	Random in i.y	7
? 0	Random in (0,1)	0.452

x y	args to verbs
m n	nouns
u v	verbs
f g h	verbs
italics	optional



Comparisons (result 1 if TRUE)

=	Equal	>	Greater	>:	GreaterOrEqual
~:	NotEqual	<	Less	<:	LessOrEqual

-:	Match (rank _ _): equal in shape, boxing, and values; but if empty, type does not matter.
're' E.'reread'	WindowedMatch (ranks may be >1) 1 0 1 0 0 0

Comparisons are tolerant; follow by !.0 for exact comparison

Assignments

(n) =. v	AssignInd: value of n gives name(s) to assign
'n1 n2' =. v1;v2	AssignMult: one level of boxing is removed
'`add sub' =. +`-	AssignAR

A is
abcd
efgh
ijkl
mnop

Selections

1 0 2 # 'abc'	Copy	acc
1j1 0 2 # 'abc'	CopyFill	a cc
1j1 0 2 #!.'*' 'abc'	CopyCustom	a*cc
1 0 1&#^:_1 'ab'	Expand	a b
1 0 1&#^:_1!.'*' 'ab'	ExpandCustom	a*b
_1 1 { A	ItemsFrom	mnop / efgh
1 3 {"1 A	FromEachRow	bd / fh / jl / np
2 1 { A	From (All axes scalar)	j
1 3 { A	From (Omitted trailing axis)	efgh / mnop
1 2 { A	From (Axis 1 Complementary)	efh
1 3 1 0 2 { A	From (General axes)	feg / nmo
(<a:;2 0) { A	From (Omitted early axis)	ca / ge / ki / om
1 1 / 3 2 (<"1@[{]) A	FromUnboxed (Fast form)	fo

Monads i. & i:

i. 3	0 1 2
i. _3	2 1 0
i. 2 3	0 1 2 / 3 4 5
i: 2	_2 _1 0 1 2

Searches

'people' i. 'pow'	IndexOf[1]	0 2 6
'people' i: 'pow'	IndexOfLast[1]	3 2 6
'pow' e. 'people'	ElementOf[1]	1 1 0
I. 0 1 1 0 1	IndicesOfOnes	1 2 4
0 2 4 I. 2 3 _1 9	FindInsertionPoint[12]	1 2 0 3
(i. >./) 1 2 8 5	IndexOfLargest[34]	2
3 (= i. 1:) 1 3 3 0	FindFirstTrue[456]	1
3 ([: I. =) 1 3 3 0	IndicesWhereTrue[67]	1 2
m&i. e.&n m&i:	FastSearch (when used repeatedly)	

[1]rank searched for is rank of items of other operand [2]min index before which item can be inserted in order [3]or <. [4]or i: [5]or 0: [6]any comp. or e. [7]or +/ +./ *./

Operations on Ordered Sets

'rare'-. 'er'	RemoveItems	a
~. 'rare'	UniqueItems	rae
~: 'rare'	UniqueSieve	1 1 0 1
i.~'rare'	SelfClassify	0 1 0 3

Shorthands

<: 5	y-1	Decrement	4
>: 5	y+1	Increment	6
% 5	1%y	Reciprocal	0.2
-. 0.3	1-y	Complement	0.7
+: 9	y*2	Double	18
-: 9	y%2	Halve	4.5
*: 9	y^2	Square	81
%: 9	2%:y	SquareRoot	3
^ 1	e^y	Exp	2.718
^. 7.389	e^.y	NaturalLog	2
#. 1 0 1	2#.y	FromBase2	5
#: 5	2#:y	ToBase2*	1 0 1
m&#.^:_1	m#:y	ToBasem*	

*produce as many result items as needed to hold significance

Operations on Booleans

16b1a Base16 constant 26

Dyads:		Monad: -. NOT
+. OR	*. AND	~: XOR
+: NOR	*: NAND	= XNOR

> < >: <: are also meaningful.

m b.	(0≤m<16) Boolean function with truth table 2 2#:4 4#:m (1 b. is AND)
m b.	(16≤m<32) bitwise Boolean; applies (m-16) b. to each bit of integers

x 32 b. y	x 33 b. y	x 34 b. y
rotate y left x bits	unsigned shift y left (x>0) or right (x<0) x bits	signed

Join and Reshape

, ab / cd	Enfile	abcd
'ab' , 'cd'	Append	abcd
0 1 / 2 3 , 8 9	Append (unequal ranks)	0 1 / 2 3 / 8 9
0 1 / 2 3 , ,8	Append (short)	0 1 / 2 3 / 8 0
0 1 / 2 3 , 8	Append (atom)	0 1 / 2 3 / 8 8
,. 'ab'	EnfileItems	a / b
'ab' ,. 'cd'	AppendItems	ac / bd
$,: 'ab'	Itemize (adds leading axis)	1 2
'ab' ,: 'cd'	Laminate	ab / cd
3 $ 0 1 / 2 3	ReshapeItems	0 1 / 2 3 / 0 1
3 ($,) 0 1 / 2 3	Reshape	0 1 2
3 ; (4 ; 5)	Link	3 4 5
3 ,&< (4 ; 5)	JoinBoxed	3 4 5
; 0 1 / 2 3 ,4 6	Raze (expand items of opened boxes to size of largest, then append)	0 1 / 2 3 / 4 0 / 6 6
;: '2 wds'	JWords	2 wds
;:^:_1 w1 w2	RazeWords	w1 w2

Take and Drop

2 {. i. 6	Take	0 1
_2 {. i. 6	TakeLast	4 5
2 }. i. 6	Drop	2 3 4 5
_2 }. i. 6	DropLast	0 1 2 3
4 {. 2 3	Overtake	2 3 0 0
4 {.!.9 (2 3)	OvertakeCustom	2 3 9 9
2 _2 {. i. 4 4	TakeMultiAxis	2 3 / 6 7
{. 0 1 2	Head	0
{: 0 1 2	Tail	2
}. 0 1 2	Behead	1 2
}: 0 1 2	Curtail	0 1

Box Operations

B is 0 1 2 3 4 5 6 7 8

L. B	Level	2
$ L:0 B	AtLevel	,2 ,2 ,2 ,2
{. L:1 B	AtLevel	0 1 6 8
# S:0 B	Spread	2 2 2 2 1
|. &.> B	Each (fast)	4 5 2 3 0 1 7 6 8
1 {:: B	Fetch	6 7
0 1 {:: B	Fetch	2 3
0 2 0 {:: B	FetchList	4 5 0 1

Whole-Array Operations

|. 'abcde'	Reverse	edcba
2 |. 'abcde'	RotateLeft	cdeab
_2 |. 'abcde'	RotateRight	deabc
2 |.!.'*' 'abcde'	ShiftLeft	cde**
|.!.'*' 'abcde'	ShiftRightOne	*abcd
abcd / efgh / ijkl / mnop; 1 _1 |.	Rotate (multiaxis)	hefg / lijk / pmno / dabc
abcd / efgh / ijkl / mnop; |:	Transpose (reverse axes)	aeim / bfjn / cgko / dhlp
x |: y	ReorderAxes (moves axes x to end of axes)	
'c0 c1 c2' =. |: y	AssignIndividualColumns	
/: 3 1 4 1	GradeUp*	1 3 0 2
/:~ 3 1 4 1	SortUp*	1 1 3 4
'abcd' /: 3 1 4 1	SortUpUsing*	bdac
/:@/: 3 1 4 1	Ordinals*	2 0 3 1
abcd / efgh / ijkl / mnop; '*' (<1 2)}	Amend	abcd / ef*h / ijkl / mnop
ab / ef / ij / mn; '*+' [`(#@[)`] }	Amend (gerund form)	ab / ef / *+ / mn
y =. x m} y	AmendInPlace (fast form)	

*use \: for descending order