

「初めてさんのJ言語(改定版)」

帝京平成大学 鈴木義一郎

【局所定義と大局定義】

局所定義は イコールピリ(=.) イコールコロンの(=:)で 大局定義

<pre>test=:3 :0 a=.i.y. a:b=:1+a)</pre>	<pre>test 5 0 1 2 3 4 1 2 3 4 5 (i.5);1+i.5 0 1 2 3 4 1 2 3 4 5</pre>	<p>左側のボックスで“test”という片側形の関数を定義して、引数に「5」を挿入して実行した結果が右側のボックスで示される。</p> <p>「;(セミコロン)」は左右の引数をボックスで囲んで接続する動詞である。</p>
<pre>a value error: a</pre>	<pre>b 1 2 3 4 5</pre>	<p>関数の実行後、イコールピリで定義した“a”は消え、イコールコロンの(=:)で定義した“b”は残っている。</p>
<p>関数内で大局定義を用いると、関数の実行後にいろんな変数が残ってしまい煩雑になる恐れがある場合には、局所定義を用いたほうがベターである。</p>		
<pre>i. 5 0 1 2 3 4</pre>	<pre>1+i. 5 1 2 3 4 5</pre>	<p>「i.»は0から始まる自然数列を右引数の個数分生成する。「1」から始めたければ“1+”とすればよい。</p>
<pre>i. _5 4 3 2 1 0</pre>	<pre>>:i. _5 5 4 3 2 1</pre>	<p>引数にマイナスの数値を入力すると“逆順”の数列を出力する。「>:」は右引数(の実数部)に1を加算</p>
<p>「始点」を「0」ではなく「1」にすべきであるとシツコク主張するユーザーもいる。 (J言語の前身である「APL」では、「0」と「1」のいずれにするかが選択できた！)</p>		

<pre>stat_reg=:3 :0 regb=[% 1:.,] NB.regression coefficient regp=(1:.,)+/ .*regb NB.predicted value regq=[:+/[:*[:-regp NB.sum of residueale regcd=:100"_*1:-regq%[:+/[:*[:(-+/%#)@[mat=[:%.([:+/ .*)]@(1:.,) NB.inverse of data matrix resvar=:regq%[::/[:\$1:.,] NB.residual variance regt=:regb%[:%:resvar*[:(<1 0)& :mat@] mll=:>:@^.@((o.2)"_*regq%#@)*#@[%_2: NB.Mll regaic=:+:@(1:+#@(1:,:)))-2:*mll 'set of regression model'</pre>	<p>左のように、回帰分析に必要な関数群を「大局定義」で行えば、</p> <pre>stat_reg" set of regression model</pre> <p>のように「stat_reg」という関数を実行することにより、必要に応じて複数個の関数をまとめて定義することができる。</p>
--	---

)	
---	--

【J言語には、「名詞(0)」、「副詞(1)」、「接続詞(2)」、「動詞(3)」といった品詞がある】

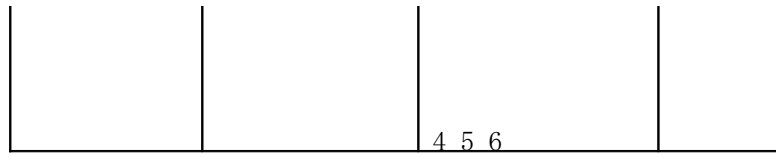
動詞との 出会いひたすら 待つ「副詞」 右にこだわる 「接続詞」
 オープン(>)は 動詞でアンド(&)は 接続詞 イーチ(& &.>)にすれば 副詞に変身
 計算を マトメテ演算 したければ レベル(L:0)やイチ(&.)を 使えばよい

a=:10 4!:0<' a' 0	av=:/ 4!:0<' av' 1	c=:& 4!:0<' c' 2	mean=:+/%# 4!:0<' mean' 3											
「4!:0<' def'」は定義内容の品詞を出力する：「0：名詞、1：副詞、2：接続詞、3：動詞」														
]d=:1 2 3;4 5 <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>1</td><td>2</td><td>4 5</td></tr><tr><td>3</td><td></td><td></td></tr></table>	1	2	4 5	3			+/L:0 d <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>6</td><td>9</td></tr></table>	6	9	+/&> d 6 9	+/&.> d <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>6</td><td>9</td></tr></table>	6	9	sum=:+/ 4!:0<' sum' 3
1	2	4 5												
3														
6	9													
6	9													
mean L:0 d <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>2</td><td>4.5</td></tr></table>	2	4.5	mean&.> d <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>2</td><td>4.5</td></tr></table>	2	4.5	mean&> d 2 4.5	mean each d 2 4.5	each=:&> 4!:0<' each' 1						
2	4.5													
2	4.5													
「;(セミコロン)」は左右の引数をボックスで囲みながら接続する動詞である。														

ほとんどの動詞には、(右引数だけの)単項(monadic)と、(左右に引数をとる)二項(dyadic)の2種類がある。			
]nl=:^.100 4.60517	^ nl 100]ol=:10 ^. 100	「^」の片側形は自然対数関数 「^」の片側形は指数関数(^.の逆関数)
1x1 2.71828	1x1 ^. 100 4.60517	10 ^ ol 100	「^」の両側形は左引数を底とする対数関数 「1x1」はオイラーの定数

【J言語と数学用語との対応表】

J言語	数学用語	Jでの表示例	ランク
アトム	スカラー	2	0
リスト	ベクトル	2 4 6(2*1+i.3)	1
テーブル	マトリクス	1 2 3 4 5 6(1+i.2 3)	2
レポート	多次元配列	1+i.2 1 3 1 2 3	3



【基本的な数学演算】

用法	結果と用例(単項)	用法	結果と用例(二項)	
+ Conjugate	実数の範囲内では「そのままの値」を出力する： + 0.4_5 0 0.4_5 0 + 3j4 3j 4 (共役複素数)	Plus 加算	左右の引数の和を出力する。	
			5 + 1 2 3 1 2 3 + 4 5 6	
			6 7 8 5 7 9	
			3j4 + 1j1 3j4 + 1j_1 4j5 4j3	
- Negate 逆符号	反対の符号にする： - 0.4_5 0 _0.4 5 0 - 3j4 _3j_4 実数部と虚数部の値を共に反対の符号にする。	Minus 減算	左引数から右引数の値を引算した結果を出力する。	
			5 - 1 2 3 1 2 3 - 4 5 6	
			4 3 2 3 3 3	
			3j4 - 1j1 3j4 - 1j_1 2j3 2j5	
* Signum 符号	実数に対しては正には1、0には0、負には-1を付与： *_5 0 2 _1 0 1 複素数に対しては、複素平面上の単位円周上に射影する： *_ 3j4 0.6j0.8 「*。」の片側形は、複素数の極座標(絶対値と偏角)を出力 *_ 3j4 5 0.927295	Timus 乗算	左右の引数の積を出力する。	
			2 * 1 2 3 1 2 * 4 5	
			2 4 6 4 10	
			3j4 * 1j1 3j4 * 1j_1 1j7 7j1	
			3j4 * 0j1 3j4 * 0j_1 4j3 4j 3	
			4j_3 * 0j1 4j_3 * 0j_1 3j4 3j 4	
			「0j1」を掛けると虚部の符号を変えて実部へ「0j_1」を掛けると実部の符号を変えて虚部へ	

%	% 2_0.5	1 % 2_0.5	Devide	左引数を右引数で割算した結果を出力する。	
Reciprocal	0.5 2	0.5 2	除算	2 4 6 % 2	2 8 % 4 0.4
逆数]r=% 3j4	r * 3j4		1 2 3	0.5 20
	0.12j 0.16	1]z=:3j4%1j1	z * 1j1
	*. 3j4	*.r		3.5j0.5	3j4
	5 0.927295	0.2 0.927295		3j4 % 0j_1	3j4 * 0j1
	極座標表示では、逆数の絶対値は元の複素数の絶対値の逆数で、偏角は符号が反対になる。		_4j3	_4j3	「z % 0j_1」と「z * 0j1」は同じ！

3j4 * 0j1	3j4 % 0j_1	j. 3j4	「j.」の片側形は右引数の複素数を複素平面上で90度回転した複素数を出力する。
_4j3	_4j3	_4j3	
]Z1=:j./ 3 4]a=:1+i.2 2]Z2=:j./ a	
3j4	1 2 3 4	1j3 2j4	
1 2 j. 3 4]Z3=:1 j. a	j./"1 a	
1j3 2j4	1j1 1j2 1j3 1j4	1j2 3j4	
(\$;#) Z1	(\$;#) Z2	(\$;#) Z3	
1	2 2	2 2 2	

icoron=:3 :'.^:(-:1-*.s) (-d), .d=.b-(i.b+1)*c=. (b=. {a})*%/a=. s=.+.y.'			
]a=. i. 1+ 3]a=. i. 1b+ _3	.a	-.a
0 1 2 3	0 1 2 3	1 2 3	1 2 3
).^:1(.a),-.a).^:0(.a),-.a	「 」 : 絶対値をとる	
3 2 1 0 1 2 3	3 2 1 0 1 2 3	「}.」 : 先頭の要素を落とす	
icoron 3	icoron _3	「-」 : 符号の逆転 「-:」 : 数値を半分に	
3 2 1 0 1 2 3	3 2 1 0 1 2 3	「 .」 : 全要素(アイテム)の反転	
i: 2 3	i: 2 3	「i:」はランク0の引数に対してのみ作動する。さらに「i.」や「i:」は整数値でない引数に対しては” domain error”となる。	
_2_1 0 1 2 0 0	0 1 2		
3 2 1 0 1 2 3	3 4 5		

i: 2j5	icoron 2j5	(-d), .d=:2-(4%5)*i.>:2
--------	------------	-------------------------

2 1.2 0.4 0.4 1.2 2	2 1.2 0.4 0.4 1.2 2	2 1.2 0.4 0.4 1.2 2
i: _2j5	icoron _2j5	(.d), -d=:2-(4%5)*i. ->:2
2 1.2 0.4 0.4 1.2 2	2 1.2 0.4 0.4 1.2 2	2 1.2 0.4 0.4 1.2 2
<p>演算子「i:」の片側形は、虚数部が正整数の「複素数」に対して演算する。</p> <p>「a+ib」という整数値(b>0)の複素数に「i:」を適用すれば、a>0の場合には、_a から a まで、中間は(2a/b)の間隔の数値を出力する。a<0の場合には、順序が反対の数列を出力。</p>		

【J 言語の特徴：Fork と Hook】

タシット(Tacit)で 定義するのが 醍醐味さ J特有の 面白さ
 演算を 右から順に 作動さす ことも可能さ エクスプリシト(Explicit)
 動詞が3つ 並んだときは 左右が先よ 中の動詞は 3番手(フォーク Fork)
 片側動詞に 両側動詞が 連結すれば カッコでくくり これ「フック(Hook)」
 並んだ動詞は 右からフォーク 残った動詞で またフォーク(フック)

(sum=:+/)D=:3 1 2 6 (sum1=:3 :'+/y. ')D 6	(mean=:+/%#)D 2 3 :'+(+/y.)%#y. 'D 2]S=:+/D 6]N=:# D 3	S % N 2
		中の両側動詞を挟んで3つの連結動詞「fgh」をフォーク(Fork)という。 2連動詞「gh」はフック(Hook)		
《フォーク(fgh)》 【単項】 【二項】 <div style="display: flex; justify-content: space-around;"> <div style="text-align: center;"> g ↙ ↘ f h ↓ ↓ y y </div> <div style="text-align: center;"> g ↙ ↘ f h ↙ ↘ ↙ ↘ ↓ ↓ ↓ ↓ X y x y </div> </div>		《フック(gh)》 【単項】 【二項】 <div style="display: flex; justify-content: space-around;"> <div style="text-align: center;"> g ↙ ↘ y h ↓ y </div> <div style="text-align: center;"> g ↙ ↘ x h ↓ y </div> </div>		
(]-+/%#)D=:1+i.5 2 1 0 1 2	(]-mean)D 2 1 0 1 2	([:*:] -mean)D 4 1 0 1 4	「[:(cap)]」は「何もしない」という動詞	
]d=: (+/%#)D 2 1 0 1 2	(dev=: -mean)D 2 1 0 1 2	([:mean*:])d 2	([:mean])d 1.2	
「5連動詞」は、右端の3つの動詞でフォークを作って1つの動詞となり、これと左端の2つの動詞と連結してフォークとなる。また「4連動詞」は、右端の3つの動詞でフォークを作って1つの動詞となり、これと左端の1つの動詞と連結してフックとなる。				
([:mean[:*:] -mean)D 2	(var=: [:mean[:*:] dev)D 2	(sdev=: [:%: var)D 1.41421		
([:mean[:] -mean)D 1.2	(mdev=: [:mean[:] dev)D 1.2	「sdev」は標準偏差 「mdev」は平均偏差		

13 :'+(+/y.)%#y. '	13 :'+mean *[:dev y. '	13 :'+mean dev y. '
---------------------	------------------------	---------------------

+ / % #	[: mean [: * : dev	[: mean [: dev
エクスプリシトで記述したプログラム(program)を、「13 :' program' 」によってタシットに変換できる。		

【 “+ +. +:” 】

実数に プラス(+)の片側 そのままで 両側形は フツアの足し算
 プラス・ピリ(+.)の 両側形は 最大公約数 複素数でも イツオーケー
 プラスコロン(+:) 右引数を 倍にする 両側形は 論理演算

+ 0.4 0 3	+ 3j4	1 2 3 + 4 5 6	複素数に対して「+」の片側形は共役複素数
0.4 0 3	3j_4	5 7 9	
6 +. 8	6 cdiv 8	6 +. 5	6 cdiv 5
2	2	1	1
cdiv=:4 :' {:/:~(a e. div y.)#a=.div x.' [div=:3 :' (0=t y.)#t=:1+i. y.' 「最小公倍数(LCD)」を出力する関数を定義している。			
div 6	div 8	div 5	「div」は右引数の全ての約数を出力する関数
1 2 3 6	1 2 4 8	1 5	
]b=:+: a=:1 2 3	2 * a	-: b	-:b._1
2 4 6	2 4 6	1 2 3	+:
			「-:」は「+:」の逆演算

【(Negative/Minus -)・(Not -)・(Halve -:)]

マイナス(-)の 片側形は 符号の反転 両側フツアの 引き算よ
 マイナスピリ(-.)の 片側形は 足して1になる 「補数」を出力
 マイナスコロン(-:)の 片側形は 右引数を 半分にする (倍增演算子「+:」の逆演算)

- _5 0 2	4 5 6 - 1 3 2	- 3j4	3j4 - 1j1
5 0 _2	3 2 4	3j_4	2j3
]b=:-. a=:4 0.3	a + b]d=:-. c=:3j4	c + d
1	1 1 1	_2j_4	1
3 0.7 0	2 4 6 % 2	(halve=:%2:)2 4	+: a
]a=: -: 2 4 6		6	

1 2 3	1 2 3	1 2 3	2 4 6
-:b._1	+:b._1	「-:」は「+:」の逆演算である。	
+:	-:		

【* * . * :】

シングナム(*)は 符号与える 片側形 複素数には 単位円に射影
 スター・ピリ(*) 複素数には 大きさと 偏角与える 片側関数
 スター・ピリ(*)の 両側形は 最小公倍数 複素数でも イッツオーケー

* _3 0 2 _1 0 1	([,]) * 3j4 1 0.6j0.8	「3j4」という複素数を複素平面の単に円上に射影すると「0.6j0.8」で、絶対値は1である。	
*. 2 2 0	*. _3 3 3.14159	実数の「正数」には偏角は0で、「負数」の偏角は「 $\pi = 3.14159$ 」である。	
*. 3j4 5 0.927295	*. _3j0 3 3.14159	一般の複素数に対しては、絶対値と偏角の値を出力する。	
4 * . 6 12 1j1 * . 3j4 7j1	4 lcm 6 12 1j1 lcm 3j4 7j1	lcm=:4 :0 a=. x. *m=. 1+i. >. (x.) <. y. /*~{(b e. a)#b=. y. *m)	
*: 4 16	?: 4 2	*:1j1 0j2	?:3j4 2j1

【% % . % :】

パーセント(%) 片側形は 逆数で 両側形なら フツアの割り算
 行列の 割算行う パーセントピリ(%) 片側形なら 逆行列
 パーセントコロソ(%) 片側形なら 平方根 両側形は 累乗根

% 2 4 5 0.5 0.25 0.2	2 4 5 % 2 1 2 2.5	4 9 % 2 3 2 3	「%」の片側形は逆数で、両側形は割算。
14 40 % . A=:2 2\$1 1 2 4 8 6			「鶴と亀の頭が14個で、足が40本である。鶴と亀はそれぞれ何匹づついるか？」
]B=:%. A 2 _0.5 1 _0.5	A +/ .* B 1 0 0 1	A +/ .* 8 6 14 40	「+/ .*」は行列と一般アレイ(リストやテーブル等)の掛算
?: 2 4 9	3 %: 8 27	0.5 %: 2 3	*: 2 3

1.41421 2 3	2 3	4 9	4 9
-------------	-----	-----	-----

【(\$ Shape), (# Talley) : アレイの形とアイテム】

形なき たったひとつは 「アトム」なり アトムが並んで 「リスト」を作る
 テーブルの 「形」を示す ドル(\$)マーク アイテム数は シャープ(# talley)さん
 引数の 低次のランクの 全てのものを 「1セル」 「2セル」 などと呼ぶ
 演算は 1つ低次の セル相手 これを名づけて 「アイテム」と呼ぶ

]A=:2 2]L=:2 1 2 1]M=:i.2 3 0 1 2 3 4 5	「行列論」でスカラーに相当するのが「アトム」、横ベクトルが「リスト」で、いわゆる行列が「テーブル」である。	
\$ A	\$ L 2	\$ M 2 3	\$ G=:i.2 3 4 2 3 4	G 0 1 2 3 4 5 6 7 8 9 10 11
# A 1	# L 2(アトムが2個)	# M 2(リストが2個)	# G 2	
+/A 2	+/L 3 +/"1 L 3	+/M 3 5 7 +/"1 M 3 12	+/G 12 14 16 18 20 22 24 26 28 30 32 34 +/"1 G 6 22 38 54 70 86	12 13 14 15 16 17 18 19 20 21 22 23 (3×4のテーブルが2枚!)

Aのようなスカラーに相当する「アトム」の形は無(次数は0)で、アイテム数は1である。

「]」は右で定義した変数を表示する。「" 1」は次数(rank)1の引数に作用させる「副詞」

「+/'」はアイテム間に" +”を挿入させることで、「/'」は「副詞」である。

A, L, M, Gの次数はそれぞれ0, 1, 2, 3で、そのアイテムの次数はそれぞれ0, 0, 1, 2である。

【# (tally : 片側形)・# (copy : 両側形)】

シャープ (#)の 片側形は タリー(tally)と呼んで アイテム数を 表示する

右で与えた データから 左指定の 個数取り出す 両側コピー(# copy)

# 3 1	# 3 4 2	# i.3 4 3	「# y.」はアイテム(引数y. より1つランクの低いセル)の数を出力する。	
0 1 1 # 1 3 5 3 5	0 2 1 # 1 3 5 3 3 5]M=:i.2 3 0 1 2 3 4 5	0 1 # M 3 4 5	

【(#.Base2)・(#:Antibase2)】

シャープ・ピリ(#) 片側形は 2進数を 10進数の 数値に変換
 左で与えた 進数で 右の数値を変換す シャープ・ピリ(#.)の 両側形
 シャープ・コロンの(#:) 片側形は 10進数を 2進の数値に 変換す
 シャープ・コロンの(#:) 両側形は シャープ・ピリ(#.)の 両側形の逆変換

#.1 0 1 5	+/(1 0 1)*2^2 1 0 5	2進数で「1 0 1」は10進数では「5」である。 「#.」の片側形は2進数を10進数に変換	
bi_10 1 0 1 5	biten 1 0 1 5	bi_10=:3 :'+/y.*2^i.-#y.' biten=:[:+/*2^[[:i.-@#	
#:b._1 #	#.b._1 #:	「#: (片側形)」は「#. (片側形)」の逆変換 「v b._1」は関数「v」の逆関数を出力する。	
]c=:#:3 5 7 0 1 1 1 0 1 1 1 1	#. c 3 5 7 (#:^:_1)c 3 5 7	+:b._1 -: *:b._1 %:	-:b._1 +: %:b._1 *:
「+:」は右引数の値を2倍にし、「-:」は右引数の値を半分にする演算子である。 「*:」は右引数の値の平方値、「%:」は右引数の値の平方根を出力する演算子である。			

10 #. d=.1+i.4 1234	+/d*(10^3 2 1 0)	10進数で「1 2 3 4」は「1234」 8進数で「1 2 3 4」は「668」
------------------------	---------------------	--

8 #. d 668	1234 +/d*(8^3 2 1 0) 668	$1 \times 8^3 + 2 \times 8^2 + 3 \times 8^1 + 4 \times 8^0$ $= 512 + 128 + 24 + 4 = 668$	
eit_10 d 668	eiten d 668	<pre>eit_10=:3 :'+/y.*8^i.-#y.'</pre> <pre>eiten=:[:+/*8:^[:i.-@#</pre>	
<pre>]c=:4 4 #:</pre> 5+i.3 1 1 1 2 1 3	4 #. c 5 6 7 c +/ .* 4^i._2 5 6 7	<pre>]b=:4 4 4 #:</pre> 31+i.3 1 3 3 2 0 0 2 0 1	4 #. b 31 32 33 b +/ .* 4^i._3 31 32 33
a=:24 60 60 b=:3600 60 1 b #: s 2 3 4]s=:a #.2 3 4 7384 +/2 3 4 * b 7384	$(2 \times 60 \times 60) + (3 \times 60) + 4$ $= 7200 + 180 + 4 = 7384$ <p>「2時間3分4秒」は「7384秒」である。 逆に「7384秒」は「2時間3分4秒」である。</p>	

【アレイの変形 : (, Ravel), (, Ravel Items), (: Itemize)】

コンマ(,)という 動詞の 片側形は 右引数を リストに変換
 コンマにピリ(,)の 片側形は 右引数を テーブル化
 コンマにコロンの(:)の 片側形は ランクを1つ 上げたアレイに

$]t=:i.2\ 3$ 0 1 2 3 4 5	$,t$ 0 1 2 3 4 5	$, i.2\ 2\ 2$ 0 1 2 3 4 5 6 7	「,(Ravel)」という動詞の片側形は、アトムやリストやテーブル等全てが「リスト化」される。
$]a=:i.2$ 0 1	$,. a$ 0 1	$,. i.2\ 2\ 2$ 0 1 2 3 4 5 6 7	「,(Ravel Items)」という動詞の片側形は、右引数の全てのアレイが「テーブル化」される。
$,: a$ 0 1	$\$, : a$ 1 2	$\$, : i.2\ 2$ 2 1 2 2 2	「:(Itemize)」という動詞の片側形は、ランクを1つ上げる。
「,: a」は見かけは「リスト」のようだが、ランクを1つ上げた「テーブル」である。			

【アレイの接続 : (, Append), (, Stitch), (: Laminate)】

コンマ(,)という 動詞の 両側形は ランクを増やさず 左右を接続
 コンマにピリ(,)の 両側形は 左右の引数を 横に接続
 コンマにコロンの(:)の 両側形は ランクを上げた アレイになる

$1\ 2\ 3, 4\ 5$ 1 2 3 4 5	$(A=:i.2\ 3), b=:6\ 7\ 8$ 0 1 2 3 4 5 6 7 8	「,(Append)」という動詞の両側形は、左右の引数の最大ランクの方に接続する。		
$1\ 2\ 3, .$ b	$2, . b$ 2 6	$(:A), . b$ 0 3 6	$(:A), (, . b)$ 0 3 6	「,(Stitch)」という動詞の両側形は「層連結」と呼ばれ、「x,.y」は「x,(.y)」と同じ結果である。
1 6 2 7 3 8	2 7 2 8	1 4 7 2 5 8	1 4 7 2 5 8	
「,.」による接続は、ランクが増すことも、変わらないこともある				
$1, :2$ 1	$A, :4\ 5$ 0 1 2	「:(Laminate)」による接続は、ランクが必ず1つ上がる。「形」が不揃いの箇所には「0」が付加される。		

2	3 4 5
1 2 3 , : 4 5 6	
1 2 3	4 5 0
4 5 6	0 0 0

【接続詞 右にこだわる 接着剤】

片側の 動詞を順に 結ぶのが アンド(&)やアット(@ @:)の 接続詞
 アンダー(&.)で 2つの動詞を 連結すれば 逆演算が 付加される
 動詞と名詞を アンド(&)で結べば 新たな動詞を 作り出す(“@”は不可)
 複数の 動詞を交互に 連結するのは タイ(tie)と呼ばれる 接続詞
 Even(...)やOdd(...:)の 接続詞 2つの動詞を接続 別の動詞を 作り出す

*:&+: 2 3 4 16 36 64	*:@+: 2 3 4 16 36 64	*:@+: 2 3 4 16 36 64	
:&.+: 2 3 4 8 18 32	-:&:&+: 2 3 4 8 18 32	「u&.v」は「u&v」を演算した後で、さらに「uの逆演算」が実行される。	
2&* 2 3 4 4 6 8	(*&2) 2 3 4 4 6 8	*: 2 3 4 4 9 16	
+`*/ i.6 29	0+1*2+3*4+5 29	+`%/ 3 1 4 3.25	3 + 1 % 4 3.25
(+`-`:`:0) 3 2 6 6 4 12 1.5 1 3	(+`:, -`:`:0) 3 2 6 6 4 12 1.5 1 3	(u`v`:`:0)は 全ての動詞を 演算する。	
(+`*`:`:3) 3 2 6 15	(+`*/) 3 2 6 15	3 + 2 * 6 15	(u`v`:`:3)は (u`v`/) と 同じに機能
(+`*`:`:6) 3 2 6 4 3 7	(+*) 3 2 6 4 3 7	a+*a=:3 2 6 4 3 7	(u`v`:`:6)は (uv)と いうフック
(+`*`-`:`:6) 3 2 6 9 4 36	(+*-) 3 2 6 9 4 36	(u`v`w`:`:6)は (uvw) というフオーク と同じ演算	

【ボックス(<)とオープン(>)・セミコロン(;)]

ボックス(<)で 囲めば全てが アトムに変身 オープン(>)使って 蘇生する

ボックスで 囲み連結 セミコロン(;) 片側形なら リストに変身!

$]B=<1\ 2\ 3$ <table border="1"> <tr><td>1</td><td>2</td></tr> <tr><td>3</td><td></td></tr> </table>	1	2	3		$> B$ 1 2 3	$(\$;\#) B$ <table border="1"> <tr><td>1</td><td>1</td></tr> </table>	1	1	「ボックス」で囲んだものは「アトム」							
1	2															
3																
1	1															
$]A=:1\ 2\ ;\ 3\ 4$ <table border="1"> <tr><td>1</td><td>2</td><td>3</td><td>4</td></tr> </table>	1	2	3	4	$> A$ 1 2 3 4	$; A$ 1 2 3 4	$4\ 5\ ;\ i.\ 2\ 2$ <table border="1"> <tr><td>4</td><td>5</td><td>0</td><td>1</td></tr> <tr><td></td><td></td><td>2</td><td>3</td></tr> </table>	4	5	0	1			2	3	$> 1\ 2\ ;\ 3\ 4\ 5$ 1 2 0 3 4 5
1	2	3	4													
4	5	0	1													
		2	3													

【副詞は後から 役割果す 動詞の活躍 拡大す】

両側動詞に ウェーブ(~)つけりゃ 右引数を 左にも

左右に数値が ある場合には 左右の引数を 交換す

ウェーブ・ピリ(~.nub)の 片側形は 重複要素を 排除する

ウェーブ・コロン(~:)の 片側形は ダブリの位置に “0” を与える

ウェーブ・コロン(~:)の 両側形は 各要素毎の 不一致に “1”

マイナス・コロン(-:)の 両側形は 引数マトメテ 一致に “1” (match)

$+/\sim a=:1+i.3$	$a +/ a$		$*/\sim a$	$a */ a$
2 3 4 3 4 5 4 5 6	2 3 4 3 4 5 4 5 6		1 2 3 2 4 6 3 6 9	1 2 3 2 4 6 3 6 9
$1\ 2\ 3\ -\sim 5$	$5 - 1\ 2\ 3$		$5 \% \sim i.5$	$(i.5) \% 5$
4 3 2	4 3 2		0 0.2 0.4 0.6 0.8	0 0.2 0.4 0.6 0.8
$a =: 1\ 2\ 1\ 3\ 3\ 2\ 1$	$\sim. a$	$\sim:a$	$(\sim:a) \#$	(~.nub)の片側形は 重複要素の排除
	1 2 3	1 1 0 1 0 0 0	a 1 2 3	
$(1\ 2)\sim:2\ 1$	$(1\ 2)\sim:1\ 2$		$(1\ 2)\sim:2\ 1\ 3$	(~:)の両側形は各要素毎の 不一致に “1”
1 1	0 0		length error	
$(1\ 2)-:2\ 1$	$(1\ 2)-:1\ 2$		$(1\ 2)-:2\ 1\ 3$	(-:)の両側形は引数 マトメテ一致に “1”
0	1		0	

【切捨て(<.)と切上げ(>.)・1減(<:)1増(>:)・(<:)と(>:)の両側形は論理演算】

小にピリ(<.) 片側形は 切り捨てる 大にピリ(>.)なら 切り上げる
 数値から 1をマイナス 小にコロシ(<:) 大にコロシ(>:)は 1を加える
 小にピリ(<.) 両側形は 小さいほう 大にピリ(>.)なら 大きいほう
 小にコロシ(<:) 大にコロシ(>:)の 両側形は 論理演算

<. 3.14 3	>. 3.14 4	Sgn 3.54 4	sgn=:<.@+&0.5 「四捨五入」関数
<: 3 3.14 2 2.14	>: 3 3.14 4 4.14	(-&1)3 3.14 2 2.14	(&1)3 3.14 4 4.14
3 <. 3.14 3	3 >. 3.14 3.14	<./ 3 3.14 4 3	>./ 3 3.14 4 4
3 <: 3.14 1	3 >: 3.14 0	3>:3 1	3>3 0

【“i. i:”】

被負整数を 瞬時に作る アイにピリ(i.) 但し始点は 0にご注意(1ではない!)
 マイナスの 整数値まで 出力す iにコロシ(i:)は 重宝動詞(但しランク0に作動)

i.3 0 1 2	i._3 2 1 0	“i.”は0から始る自然数列を生成する原始動詞。引数が負の場合には逆順になる。	
i:3 3 2 1 0 1 2 3	i:_3 3 2 1 0 1 2 3	「i:y.」は0を挟んでy.>0なら(-y.)から(y.)までの整数列を生成(y.<0なら逆順)。	
icolon=:3 :' .^:(-:1+*y.) (.,-@.)i.1+ y.'		“i:” を使わず同じ働きの関数。	
]a=.i.1+ 3 0 1 2 3]a=.i.1+ _3 0 1 2 3	. a 3 2 1 0	-@ . a 1 2 3
(.,-@.) a 3 2 1 0 1 2 3	(* 3) 1	(*_3) 1	(-:1+*3) 1
.^:1(.,-@.)a 3 2 1 0 1 2 3	.^:0(.,-@.)a 3 2 1 0 1 2 3	「 」:絶対値をとる 「 .」:先頭の要素を落とす 「-」:符号の逆転 「-:」:数値を半分に 「 .」:全要素(アイテム)の反転	
icolon 3 3 2 1 0 1 2 3	icolon _3 3 2 1 0 1 2 3	「i.」や「i:」は整数値でない引数に対しては” domain error”となる。また「i:」はランク0の引数に対してのみ作動する。	

'abcdef' i. 'aec'			(i.)の両側形は 左の要素のインデックスを 右引数のリストに与える(空欄には6)
6 0 4 2			
'abcae' i. 'aceg'			左の要素にない” g” にはインデックス外の数値(5)が表示される。
0 2 4 5			
\$ a.	a. i. 'Aa'	65 97 { a.	「a.」には256個の文字やキャラクターが入力されている。
256	65 97	Aa	
'abcae' i:	a. i: 'A a'		(i:)の両側形も (i.)の結果と ほぼ同じ ただインデックスは 後ろから
'aceg'	65 32 97		
3 2 4 5			

【指数関数と対数関数/オイラーの定数(^.^.^:x)】

ハット(^)という 動詞の片側形は 指数関数を 出力す
 ハット(^)という 動詞の両側形は 左の数だけ 累乗す
 ハットピリ(^.) 片側形は 自然対数 両側形は 左を底の対数値
 ハットコロンの(:) 反復演算の 接続詞 マイナス1なら 逆演算

$\hat{\cdot}$ i.3	1x0 1x1 1x2	*:123456789	*:123456789x
1 2.71828 7.38906	1 2.71828 7.38906	1.52416e16	15241578750190521
(1x1) $\hat{\cdot}$ i.3	2 3 $\hat{\cdot}$ 3 4	$\hat{\cdot}$ &(2 3)	*: 2 3
1 2.71828 7.38906	8 81	4 9	4 9
$\hat{\cdot}$. 1 2	(1x1) $\hat{\cdot}$. 1 2	5 10 $\hat{\cdot}$. 125 100	5 10 $\hat{\cdot}$ 3 2
0 0.693147	0 0.693147	3 2	125 100
>: $\hat{\cdot}$:2 i.3	>:>: i.3	>: $\hat{\cdot}$:_1 (2 3 4)	<: 2 3 4
2 3 4	2 3 4	1 2 3	1 2 3

【乱数の生成 (? : Roll)と(? . : Deal)】

? 10\$10	? .8\$10	? .8\$10	「?」の片側形は重複を許さぬ乱数の生成。「?.」はシードを固定
8 3 8 1 2 8 0 0 2 1	6 5 9 2 4 9 0 7	6 5 9 2 4 9 0 7	
10 ? 10	8 ? .10	8 ? .10	「?」の両側形は重複を許した乱数の生成。「?.」はシードを固定
5 9 6 0 7 3 4 8 2 1	6 9 1 4 0 2 3 8	6 9 1 4 0 2 3 8	
? 2 8 \$ 10	? .2 8 \$ 10	? .2 8 \$ 10	0 から 9 までの重複乱数を 2×8 の形で生成している。「?.」を使えばシードを固定できる。
5 7 8 8 2 1 9 7	6 5 9 2 4 9 0 7	6 5 9 2 4 9 0 7	
9 9 1 4 3 9 7 2	0 4 6 8 3 8 1 2	0 4 6 8 3 8 1 2	

【[,], [:]】

セイム(same [,])は 左右のいずれかを 出力させる 便利な動詞
 キャップ[:]はなんとも 不思議な動詞 何もしないで フォークを作る

2 3 [4 5	2 3] 4 5		
2 3	4 5		
([:*+:)2 3 4	([:>+:)2 3 4	*:&+: 2 3 4	>:&+: 2 3 4
16 36 64	5 7 9	16 36 64	5 7 9
2(+*-)1]a:=(2+1)*(2-1)	2([:>+*-)1	>:a

3	3	4	4
(abs=: :[:])_1 2 3 1 2 3	2(res=:[: :)1 2 3 1 0 1	キャップ([: cap])は 演算結果に 関係せず	

【 { } { : }】

ボックスで 与えた要素の 組合せ 片側動詞の カタログ({ catalogue)なり
 中カッコ({} 左で与えた インデクスの アイテムを取る 両側関数
 カッコ閉じ()) 左で与えた インデクスの アイテム修正 両側関数
 修正値と インデクスを左に 入力すれば 右引数の値を 修正す() amend)
 ヘッド({.}で先頭 テール(:)で末尾 要素取り出す 片側動詞
 左で与えた個数分 take({.}は取りで drop().)は除く 便利な両側 動詞なり

0 1 ; 2 3 <table border="1"><tr><td>0</td><td>2 3</td></tr><tr><td>1</td><td></td></tr></table>	0	2 3	1		{3 <table border="1"><tr><td>3</td></tr></table> アトムに対して は“ボックス(<)”と 同じ	3	{ 0 1 ; 2 3 <table border="1"><tr><td>0</td><td>0 3</td></tr><tr><td>2</td><td></td></tr><tr><td>1</td><td>1 3</td></tr><tr><td>2</td><td></td></tr></table>	0	0 3	2		1	1 3	2		
0	2 3															
1																
3																
0	0 3															
2																
1	1 3															
2																
1 { 1 2 3 2	0 { i.2 3 0 1 2]M=:i.2 3 0 1 2 3 4 5	1 1 0 } M 3 4 2													
3 4(0 1)}i.3 3 4 2	'BD' (1 3)}A=: 'abcde' aBcDe	A abcde	修正したものを定義し直 さないと変更されない。													
]K=:1+i.3 1 2 3	{. K 1	{: K 3	_1 {. K 3													
2 {. K 1 2	2 }. K 3	「:」の両側形は機能無し。														

【マイナス、無限大と不定形(_ . :)】

- 3.14 3.14	- _3.14 3.14		アンダー・バー(_)は 負数を示す 名詞 無限大(∞)も示す 両刀使い
2 % _ 0	2 % __ 0	_ + _	
_ - _ .	3 + _. .	2 * _. 0	“_.”は 不定形(indeterminate)の名詞
: ''	(]%:)2 0		「_:」は無限大(∞)を出力する動詞
(]+1"_)i.3	(]+1:)i.3		「" _」は数値につけて “動詞化” する。



【(| |. |:)]

割算の 余り求める 棒(|)一本 片側形なら 絶対値
 棒にピリ(|.) 片側形なら アイテムの 順序をそっくり 逆にする
 棒ピリ(|.)の 両側形は 左の数だけ 右に回転(rotate) 負なら左へ
 棒にコロソ(|:) 片側形なら アレイの軸の 順序をソックリ 入れ替える
 棒にコロソ(|:) 両側形は 左指定の 軸を0軸に 転置(transpose)する
 左にボックスの データを入力すれば 対角要素を 出力す

3 i.6 0 1 2 0 1 2	1 _2 3 _4 1 2 3 4	3j4 5	
.d=:1 2 3 4 5 5 4 3 2 1	. i.2 3 3 4 5 0 1 2	「 .」の片側形はアイテムの順序の逆転	
1 .d 2 3 4 5 1	_1 . d 5 1 2 3 4	2 .d 3 4 5 1 2	_2 . d 4 5 1 2 3
:i.3 3 0 3 6 1 4 7 2 5 8]m=:2 3\$'abcdef' abc def	:m ad be cf	1 0 :m ad be cf
1 : m abc def	0 1 :m abc def	(<0 1) :m ae	(<0 1) : i.3 3 0 4 8