

「初めてさんのJ言語」

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

test=:3 :0	test 5	右側のボックスで“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左側のボックスで示される。
a=. i. y.	1 2 3 4 5	
b=:1+a	1+i.5	
)	1 2 3 4 5	
a	b	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。
value error: a	1 2 3 4 5	

【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 エクスプリシット(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	(mean=:+/%#)D]S=:+/D]N=:#	S % N
6	2	6	D	2
(sum1=:3 :'+/y.')	3 :'+/y.)%#y.'	3		
6	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。		
《フォーク(fgh)》			《フック(gh)》	
【単項】	【二項】	【単項】	【二項】	
g	g	g	g	
↙ ↘	↙ ↘	↙ ↘	↙ ↘	
f h	f h	y h	x h	
↓ ↓	↙ ↘ ↙ ↘	↓	↓	

<code> .^:1(.a),-.a</code>	<code> .^:0(.a),-.a</code>	「 」：絶対値をとる
<code>3 2 1 0 1 2 3</code>	<code>3 2 1 0 1 2 3</code>	「}.」：先頭の要素を落とす
<code>icoron 3</code>	<code>icoron _3</code>	「-」：符号の逆転 「-:」：数値を半分に
<code>3 2 1 0 1 2 3</code>	<code>3 2 1 0 1 2 3</code>	「 .」：全要素(アイテム)の反転
「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。		

<code>'abcdef' i. 'aec'</code>	(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)	
<code>6 0 4</code>		
<code>'abcae' i. 'aceg'</code>	左の要素にない” g” にはインデックス外の(5)が表示される。	
<code>0 2 4 5</code>		
<code>\$ a.</code>	<code>a. i. 'Aa'</code>	<code>65 97 { a.</code>
<code>256</code>	<code>65 97</code>	<code>Aa</code>
「a.」には256個の文字やキャラクターが入力されている。		
<code>'abcae' i:</code>	<code>a. i: 'A a'</code>	(i:)の両側形も (i.)の結果と ほぼ同じ
<code>'aceg'</code>	<code>65 32 97</code>	ただインデックスは 後ろから
<code>3 2 4 5</code>		

<pre>icolonc=:3 :0 d=. (+:r=. a)%b ['a b'=. +. y. while. 0<{:r do. r=. r, ({:r)-d end. if. 0={:r do. r=. (-r), .}:r else. r=. (-r), .r=.}:r end. ^:(-:1-*a)r)</pre>		
<code>i: 3j4</code>	<code>icolonc 3j4</code>	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。 「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、順序が反対の数値を出力する。
<code>_3 _1.5 0 1.5 3</code>	<code>_3 _1.5 0 1.5 3</code>	
<code>icolonc _3j4</code>	<code>icolonc _3j4</code>	
<code>3 1.5 0 1.5 3</code>	<code>3 1.5 0 1.5 3</code>	
<code>i: 3j5</code>	<code>icolonc 3j5</code>	
<code>_3 _1.8 _0.6 0.6 1.8</code>	<code>_3 _1.8 _0.6 0.6 1.8</code>	
<code>3</code>	<code>3</code>	
<code>i: _3j5</code>	<code>icolonc _3j5</code>	
<code>3 1.8 0.6 _0.6 _1.8</code>	<code>3 1.8 0.6 _0.6 _1.8</code>	
<code>3</code>	<code>3</code>	

【I. (Indices)】

]k=:i.@# a=:0 0 1 0 1 0 0 1 2 3 4 5	a # k 2 4	(indices=:#i.@#)a 2 4	I. a 2 4
]k=(i.@#)b=:0 0 1 0 2 0 0 1 2 3 4 5	b # k 2 4 4	indices b 2 4 4	I. b 2 4 4
]x=:?.10\$20 6 15 19 12 14 19 0 17 0 14	10 I.@:< x 1 2 3 4 5 7 9	10(<#i.@#)x 1 2 3 4 5 7 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	\$ M	\$ G]G=:i.2 3 4
# A	# L	# M	# G	0 1 2 3 4 5 6 7 8 9 10 11
1	2	2	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 +/"2 G 12 15 18 21 48 51 54 57	12 13 14 15 16 17 18 19 20 21 22 23

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。

【\$. (Sparse)】

+:&\$. i.5 0 2 4 6 8	*:&\$. i.5 0 1 4 9 16		
+:&\$. (^:_1)i.5 1 2	0\$.i.3 1 1	0\$.1+i.3 0 1	

2 4	2 2	1 2	
3 6		2 3	
4 8			

【\$: (Self Reference)】

1: `[*\$:@<:)]@.* 5 120	5*4*3*2*1 120	フレーズの生じた結果を代理して受け、左の連結詞や文の実行が完了したときに停止する。「5×4×3×2×1」や「5+4+3+2+1」
0: `([+\$:@<:)]@.* 5 15	5+4+3+2+1+0 15	
0: `([-\$:@<:)]@.* 5 3	5-4-3-2-1-0 3	

【##.#:】

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)
 シャープ・ピリ(#.) 片側形は 2進数 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

# 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)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
#. 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	
10 #. d=. 1+i. 4 1234 8 #. d 668	+/d*(10^3 2 1 0) 1234 +/d*(8^3 2 1 0)		10進数で「1 2 3 4」は「1234」 8進数で「1 2 3 4」は「668」 $1 \times 8^3 + 2 \times 8^2 + 3 \times 8^1 + 4 \times 8^0$ $= 512 + 128 + 24 + 4 = 668$	

	668	
t=:24 60 60]s=:t #.2 3 4 7384	t #: s 2 3 4	$2 \times (60 \times 60) + 3 \times 60 + 4$ $= 7200 + 180 + 4 = 7384$ 「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。
]tt=(*/¥.).t),1 3600 60 1	+ /2 3 4*tt 7384	
]b=#:3 5 7 0 1 1 1 0 1 1 1 1	#. b 3 5 7	「#: (片側形)」は「#. (片側形)」の逆変換

【**.*:】

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
 スター・コロンの(:)は 平方値 平方根なら パーセント・コロンの(%):

_3 0 2 1 0 1	(,]) 3j4 1 0.6j0.8	「3j4」という複素数を複素平面上に射影すると「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(%) 片側形なら 逆数で 両側形なら 割算を行う

行列の 割算行う パーセントピリ(%) 片側形なら 逆行列
 パーセントコロン(%) 片側形なら 平方根 両側形は 累乗根

% 2 4 5 0.5 0.25 0.2]d=% c=:3j4 0.12j 0.16	c * d 1	2 4 5 % 2 1 2 2.5
7j1 % 1j1 4j 3	7j1 % 3j4 1j 1	「%」は、実数だけでなく複素数どうしの割算も行う演算子である。	
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 1.41421 2 3	3 %: 8 27 2 3	3 %:_2j2 1j1	

【 “+ +. +:” 】

実数に プラス(+)の片側 そのままで 複素数には 共役複素数
 プラス(+)の両側 フツアの足し算 複素数でも イッツオーケー
 プラスピリ(+.) 複素数値に 適用すれば 実部と虚部の 数値を出力
 プラス・ピリ(+.)の 両側形は 最大公約数 複素数でも イッツオーケー
 プラスコロン(+:) 右引数を 倍にする 両側形は 機能無し

+ 0.4 0 3 0.4 0 3	+ 3j4 3j 4	1 2 3 + 4 5 6 5 7 9	1j2 + 3j4 4j6
+ . 3 3 0	+ . 3j4 3 4		
6 +. 8 2			
]Z1=:/:~j.^:(i.4)1j0 _1 0j_1 0j1 1]Z11=:/:~j.^:(i.4)1j1 _1j_1 _1j1 1j_1 1j1	1j2 +. _1j2 0j1	

<pre>]Z12=:/:~j.^:(i.4)1j2 _2j1 _1j_2 1j2 2j_1</pre>	<pre>]Z21=:/:~j.^:(i.4)2j1 _2j_1 _1j2 1j_2 2j1</pre>	<pre>1j2 % 0j1 2j_1</pre>	<pre>_1j2 % 0j1 2j1</pre>
<pre>]Z=:/:~Z1,Z11,Z12,Z21 _2j_1 2j1 _1j_2 _1j_1 1 _1j1 _1j2 0j_1 0j1 1j_2 1j_1 1 1j1 1j2 2j_1 2j1</pre>			
<pre>complex=:3:0 c=.a j. .a=.i.>.y. /:~.j.^:(i.4)"0 c)</pre>	<pre>less=:3:0 z=.complex r=.1 while. r<y. do.z=.z,complex r=.r+1 end.)</pre>	<pre>cdiv=:4:0 z=.less(+/ +x.)<+ +y. a=(0=([:+/-<.)"1+x.%a)#a=(0=z x.)#z b=(0=([:+/-<.)"1+y.%b)#b=(0=z y.)#z /:~(b e.a)#b)</pre>	
<pre>]a=:1j2 cdiv _1j2 _1 0j_1 0j1 1 1j2 % a _1j_2 _2j1 2j_1 1j2 _1j2 % a 1j_2 _2j_1 2j1 _1j2 1j2 +. _1j2 0j1</pre>	<pre>]b=:1j2 cdiv 2j4 _2j1 _1j_2 _1 0j_1 0j1 1 1j2 2j_1 1j2 % b 0j_1 _1 _1j_2 _2j1 2j_1 1j2 1 0j1 2j4 % b 0j_2 _2 _2j_4 _4j2 4j_2 2j4 2 0j2 1j2 +. 2j4 1j2</pre>		
<pre>+: 2 4</pre>	<pre>+: 3j4 6j8</pre>		

【-:~:]

マイナス(-)の片側形は 符号の反転 複素数でも 実・虚同時に
 マイナス(-)の両側フツの 引き算よ 複素数でも イッツオーケー
 マイナスピリ(-)の片側形は 足して1になる 「補数」を出力
 マイナスピリ(-)の両側形は 右引数に 無いモノを出す
 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)
 マイナスコロン(:)の両側形は 「形」まで含めて 一致(1)か否(0)か

-_5 0 2	- 3j4	4 5 6 - 1 3 2	3j4 - 1j2
5 0 2	3j 4	3 2 4	2j2
]b=:-. a=:4 0.3	a + b]d=:-. c=:1j2	c + d
1	1 1 1	0j_2	1
_3 0.7 0			
a=:1 2 [b=:1 3 5	a -: b]c=:-. a e. b	
	2	0 1	
]a=: -: 2 4 6	+: a]b=: -: 2j4	+: b
1 2 3	2 4 6	1j2	2j4
1 2 -: 1 2	1 2 = 1 2	z -: z=:1j2 3j4	z = z
1	1 1	1	1 1

動詞との 出会いひたすら 待つ「副詞」 右にこだわる 「接続詞」
 計算を マトメテ演算 したければ レベル(L:0)やイーチ(&>)を 使えばよい
 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

4!:0' a=:10'	4!:0' av=:/'	4!:0' c=:&'	4!:0' mean=:+/%#
0	1	2	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」			

<pre>]d=:1 2 3;4 5</pre> <table border="1" data-bbox="240 344 408 421"> <tr><td>1</td><td>2</td><td>4</td><td>5</td></tr> <tr><td>3</td><td></td><td></td><td></td></tr> </table>	1	2	4	5	3				<pre>+/L:0</pre> <table border="1" data-bbox="507 387 595 421"> <tr><td>6</td><td>9</td></tr> </table>	6	9	<pre>+/&> d</pre> <table border="1" data-bbox="818 387 906 421"> <tr><td>6</td><td>9</td></tr> </table>	6	9	<pre>+/&. ></pre> <table border="1" data-bbox="818 387 906 421"> <tr><td>6</td><td>9</td></tr> </table>	6	9	<pre>sum=:+/ 4!:0<'sum'</pre>
1	2	4	5															
3																		
6	9																	
6	9																	
6	9																	
<pre>(mean=:+/%#)L:0</pre> <table border="1" data-bbox="240 528 360 564"> <tr><td>2</td><td>4.5</td></tr> </table>	2	4.5	<pre>mean&> d</pre> <table border="1" data-bbox="539 495 627 528"> <tr><td>2</td><td>4.5</td></tr> </table>	2	4.5	<pre>mean&. > d</pre> <table border="1" data-bbox="754 483 882 528"> <tr><td>2</td><td>4.5</td></tr> </table>	2	4.5	<pre>each=:&> 4!:0<'each'</pre>									
2	4.5																	
2	4.5																	
2	4.5																	