（ＪＡＰＬＡ 2007/5/26）

「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(=:)で　大局定義

test=:3 :0 a=.i.y. b=:1+a)	test 5 1 2 3 4 5 1+i.5 1 2 3 4 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 |value error: a	b 1 2 3 4 5	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(=:)で定義した”b”は残っている。

【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】

```
g
↙ ↘
f   h
↓   ↓
```

【二項】

```
  g
↙   ↘
f     h
↙ ↘ ↙ ↘
```

《フック(gh)》

【単項】

```
g
↙ ↘
y   h
    ↓
```

【二項】

```
  g
↙   ↘
x     h
      ↓
```

y　y　↓　↓↓　↓ x　y x　y	y　y

(]-+/%#)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
「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。			
(-+/%#)D _2 _1 0 1 2	(dev=:-mean)D _2 _1 0 1 2	([:mean[:*:dev)D 2	
「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。			
([:(+/%#)[:*:]-+/%#)D 2	([:mean[:*:]-mean)D 2	(var=:[:mean[:*:dev)D 2	
([:(+/%#)[:|]-+/%#)D 1.2	([:mean[:|]-mean)D 1.2	(mdev=:[:mean[:|dev)D 1.2	
	([:%:[:mean[:*:dev)D 1.41421	(sdev=:[:%:var)D 1.41421	
	７連動詞	５(３)連動詞	

【“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にコロン(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)。	
icoron=:3　:’|.^(-:1+*y.)(|.a),-}.a=.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 1 2 3	-}.a _1 _2 _3
(|.a),-}.a 3 2 1 0 _1 _2 _3	(* 3) 1	(*_3) _1	(-:1+*3) 1	(-:1+*_3) 0

|.^:1(|.a),-}.a _3 _2 _1 0 1 2 3	|.^:0(|.a),-}.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
icoron 3 _3 _2 _1 0 1 2 3	icoron _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' 6 0 4		(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)
'abcae' i. 'aceg' 0 2 4 5		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。
$ a. 256	a. i. 'Aa' 65 97	65 97 { a. Aa	「a.」には 256 個の文字やキャラクターが入力されている。
'abcae' i: 'aceg' 3 2 4 5	a. i: 'A a' 65 32 97	(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

```
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
)
```

i: 3j4 _3 _1.5 0 1.5 3 icolonc _3j4 3 1.5 0 _1.5 _3	icolonc 3j4 _3 _1.5 0 1.5 3 icolonc _3j4 3 1.5 0 _1.5 _3	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、順序が反対の数列を出力する。
i: 3j5 _3 _1.8 _0.6 0.6 1.8 3 i: _3j5 3 1.8 0.6 _0.6 _1.8 _3	icolonc 3j5 _3 _1.8 _0.6 0.6 1.8 3 icolonc _3j5 3 1.8 0.6 _0.6 _1.8 _3	

【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 2	$ M 2 3	$ G 2 3 4	]G=:i.2 3 4 0 1 2 3 4 5 6 7 8 9 10 11
# A 1	# L 2	# M 2	# 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 +/"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。				

【$.(Sparce)】

+:&.$.i.5 0 2 4 6 8	*:&.$.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 3 | 6 4 | 8	2 | 2	1 | 2 2 | 3	

【$:(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(#:)　片側形は　10 進数を　２進の数値に　変換す

シャープ・コロン(#:)　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。
]tt=:(*/¥.}.t),1 3600 60 1	+/2 3 4*tt 7384	逆に「7384秒」は「２時間３分４秒」である。
]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(*:)は　平方値　平方根なら　パーセント・コロン(%:)

* _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「π＝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　フツーの足し算　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

```
   ]Z12=:/:~j.^:(i.4)1j2
_2j1 _1j_2 1j2 2j_1
```

```
   ]Z21=:/:~j.^:(i.4)2j1
_2j_1 _1j2 1j_2 2j1
```

```
      1j2  %
0j1

2j_1
```

```
   _1j2 % 0j1
2j1
```

```
   ]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
```

```
   complex=:3 :0
c=.a j.|.a=.i.>:y.
/:~~.,j.^:(i.4)"0 c
)
```

```
   less=:3 :0
z=.complex r=.1
while. r<y.
   do.z=.z,complex r=.r+1
end.
)
```

```
   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
)
```

```
   ]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
```

```
   ]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
```

```
   +: 2
4
```

```
   +: 3j4
6j8
```

【- -. -:】

マイナス(-)の 片側形は 符号の反転　複素数でも　実・虚同時に

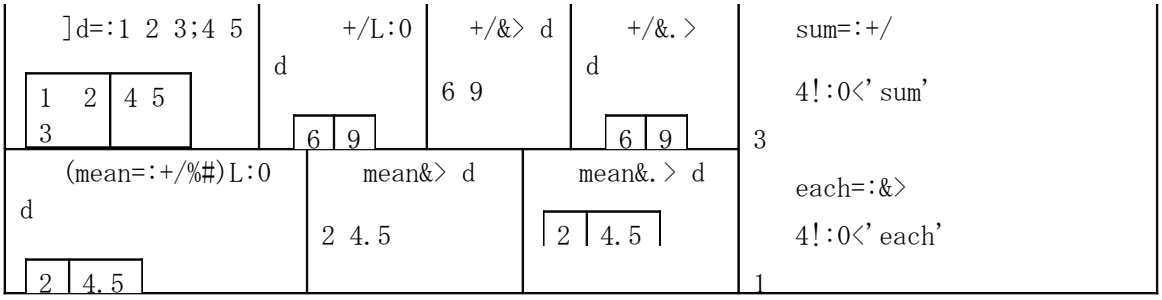
マイナス(-)の　両側フツーの　引き算よ　複素数でも　イッツオーケー
マイナスピリ(-.)の　片側形は　足して１になる　「補数」を出力
マイナスピリ(-.)の　両側形は　右引数に　無いモノを出す
マイナスコロン(-:)の　片側形は　右引数を　半分にする（倍増演算子「+:」の逆演算）
マイナスコロン(-:)の　両側形は　「形」まで含めて　一致(1)か否(0)か

- _5 0 2 5 0 _2	- 3j4 _3j_4	4 5 6 - 1 3 2 3 2 4	3j4 - 1j2 2j2
]b=:-.a=:4 0.3 1 _3 0.7 0	a + b 1 1 1	]d=:-.c=:1j2 0j_2	c + d 1
a=:1 2 [b=:1 3 5	a -: b 2	]c=:-.a e.b 0 1	
]a=: -: 2 4 6 1 2 3	+: a 2 4 6	]b=: -: 2j4 1j2	+: b 2j4
1 2 -: 1 2 1	1 2 = 1 2 1 1	z -: z=:1j2 3j4 1	z = z 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' 0	4!:0<'av=:/' 1	4!:0<'c=:&' 2	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
1 2 3
4 5
+/L:0
d
6 9
+/&> d
6 9
+/&.>
d
6 9
sum=:+/
4!:0<'sum'
3
(mean=:+/%#)L:0
d
2 4.5
mean&> d
2 4.5
mean&.> d
2 4.5
each=:&>
4!:0<'each'
1