（ＪＡＰＬＡ 2006/08/5-7）

「チュートリアル セミナー」

帝京平成大学　鈴木義一郎

§１　整数列の生成

integer=:>:@i. NB. 整数列を生成する(片側形の)関数

i.5	1+i.5	>: i.5	(>:i.)5
0 1 2 3 4	1 2 3 4 5	1 2 3 4 5	1 1 1 1 1

Ｊ言語では、指標原点が常に0に設定されるので、「i.」は0から始まる連続した整数を、右引数で与えた個数だけ生成する。そこで、「>:」は「1 を加える」という演算子で、前の結果に１を加えている。ところが、カッコをつけてしまうと「フック」となり、「5>:i.5」のように演算する。「>:」の両側形は「≧」で、5 が右側の要素のいずれよりも大きいから、論理演算は真で１になる。

>:i.4 5 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20	「i.4 5」は、0 から始まる連続した 20(=4×5)個の整数を「4×5」の行列で生成する。さらに「>:」で、１から始まるようにしてある。
>:i._5 5 4 3 2 1	引数に負の値を入力すると、逆順の整数列が得られる
i:5 _5 _4 _3 _2 _1 0 1 2 3 4 5 i:_5 5 4 3 2 1 0 _1 _2 _3 _4 _5	「i:」という演算子の片側形は、右引数で与えた数の負の値から正の値までの整数列を生成する。また負の値を入力した場合には、逆順の結果を出力する。

……「Ｊ」言語メモ ……

【合成関数ｇ・ｆが両側形として機能する場合】

次のように、３通りの結合のし方が考えられる。そこでＪ言語では

ｘ（ｇ＠ｆ）ｙ＝ ｇ（ｘ ｆ ｙ）	ｆが両側形でｇが片側形の場合
ｘ（ｇ＆ｆ）ｙ＝（ｆ ｘ）ｇ（ｆ ｙ） ｘ（ｇ ｆ）ｙ ＝ ｘ ｇ（ｆ ｙ）	ｆが片側形でｇが両側形の場合 両側形の場合の「フック(hook)」

【合成関数ｇ・ｆが片側形として機能する場合】

（ｇ＠ｆ）ｙ＝（ｇ＆ｆ）ｙ＝ｇ（ｆ ｙ）	ｆもｇも片側形の場合

（ｇｆ）ｙ　＝　ｙｇ（ｆｙ）	ｆが片側形でｇが両側形の場合

§２　総和

sum=:+/ NB. 総和を求める(片側形の)関数

]D1=:>:i.5 1 2 3 4 5		D1 というベクトル（Ｊ言語では「リスト」という）を定義し、「] 」によりその内容を表示している。
+/ D1 15	sum D1 15	「/」は副詞で、左側の動詞を引数のアイテム間に挿入する。結局、D1 というリストの総和を与えている。
]D2=:>:i.4 5 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20		D2 という行列（Ｊ言語では「テーブル」という）を定義し、その内容を表示している。
sum D2 34 38 42 46 50		D2 のアイテムについての総和で、したがって０軸(縦)方向の和を与えている。
sum"1 D2 15 40 65 90		D2 のランク１のセル、したがって１軸(横)方向の和を与えている

……「Ｊ」言語メモ ……

Ｊ言語では、動詞、名詞のように、通常の言語と同様に「品詞」という概念が対応している例えば、変数として定義された「D1」は名詞であり、また「sum」という関数の定義に用いた「+/」，「] 」は、「全ての要素を足す」，「右引数を取り出す」といった動詞である。

特に名詞に関しては、「ランク」という概念が対応している。単独の数値「1.1」や文字「a」などは「アトム」と呼ばれ、数学の場合のスカラーに対応し、ランクは０である。ランク１の名詞は「リスト」と呼ばれ、数学の(横)ベクトルに対応する。また行列に対応するのが「テーブル」で、ランクは２である。さらにランクが３以上のものも考えられ、一般に「アレイ」と呼ばれる。アレイの形は

```
   $ D2
4 5
```

といったように、「$」という演算子で与えられる。この形を示す数値の位置を左から順に、0 軸、１軸、……　というように呼称する。（中村先生は「ケシカラン」と喚いている！）

i.３１	$ i.３１	なおＪ言語には、「縦ベクト

0 1 2	3 1	ル」に対応するものはなく、「3 1」という形のテーブルである。

§３　平均

mean=:+/%#　NB. 算術平均を求める(片側形の)関数

D1	# D1	+/ D1
1 2 3 4 5	5 【D1 というリストの個数】	15 【D1 というリストの総和】

(+/D1)%(#D1)	mean D1
3 【「15÷5」の値を求めている】	3 【D1 というリストの(算術)平均の値】

D2	mean D2	D2 のアイテムについて、つまり０軸(縦)方向の平均を求めている。
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20	8.5 9.5 10.5 11.5 12.5 mean"1 D2 3 8 13 18	D2 というテーブルの 1-セルであるリストを引数に、リストのアイテムであるアトムに対して「mean」という動詞が作動し、１軸(横)方向の平均を求めている。

……「Ｊ」言語メモ ……

あるアレイの軸の右の方からｋ番目まで取り出した要素を「k-セル」とか「ランクｋのセル」という。例えばD2 というアレイの「1 2 3 4 5」などは、１つの「1-セル」である。またD2 の「2-セル」はD2 自身で、１つしかない。「0-セル」は必ずアトムで、「1-セル」はリストである。特に、アレイのランクより１つだけ小さいランクのセルを「アイテム」という。一般に、動詞の作用する対象はアイテムであるが、「" 1」という副詞をつけると、「1-セル」を引数とみなして演算を行なうことになる。

またＪ言語では、３種類の動詞が連なったものを「フォーク」と呼んでいる。一般に

(fgh) y

のように右側だけに「引数」のある場合を「片側形」といい

x (fgh) y

のように両側に「引数」のある場合を「両側形」と呼んでいる。

そして演算の順序は

x（と y）にｈという動詞をオペレートした結果を右引数に

x（と y）にｆという動詞をオペレートした結果を左引数に

といった手順で実行される（中央の動詞「ｇ」は両側形でなければならない！）。

§４　幾何平均と調和平均

```
meang=:#%:*/  NB. 幾何平均を求める(片側形の)関数
meanh=:[:%[:mean%  NB. 調和平均を求める(片側形の)関数
```

```
   (mean D1)>(meang D1)>meanh D1
1
```

D1 1 2 3 4 5	# D1 5	*/ D1 120	「#」という演算子は右引数のアイテム数を与える。「*/」は引数の各要素の積を与える。
G=:(#D1)%:(*/D1) 2.60517 meang D1 2.60517	G ^ 5 120 (meang D1)^5 120		120(*/D1)の 5(# D1)乗根を G に挿入し表示している。この G を 5 乗すると 120 の戻る D1 というリストの幾何平均の値を出力している。

([:mean%) D1 0.456667 ([:%[:mean%) D1 2.18978 meanh D1 2.18978	「%」という演算子で D1 の各要素の逆数を求め、その平均を求めている。[[:」は“何もしない”という動詞である。 直上の結果の逆数を与えている。つまり、D1 の調和平均を求めている。 「([:gh)」とう形のフォークでは「g」という中の動詞は片側形である。
(mean D1)>(meang D1)>meanh D1 1	(算術平均)＞(幾何平均)＞(調和平均) であるから、論理式の出力は 1(正)である。

……「Ｊ」言語メモ ……

「〇月〇日、晴れ。山へ登った。頂上でお弁当を食べて遊んだ。山から降りた。」

この文章では

ｆ＝「山へ登る」

ｇ＝「頂上で弁当を食べて遊ぶ」

ｈ＝「山から降りる」

という３つの動詞が、上から順に実行されている。つまり「ｈ＠ｇ＆ｆ」という合成関数がオペレートされたことになる。

ところで、山へ登りっぱなしという人はいなくて、必ず山を降りるはずである。「山から降りる」という動詞は、「山へ登る」前の状態に戻ることだから、ｈはｆという動詞の「逆演算」で、Ｊ言語では、「f:_1」で与えられる。そこで、関数「ｇ&. ｆ」が「(f^:_1)@g&f」と同じ

働きをするように、アンダーと呼ばれる接続詞「&.」を用意しておけば、h という動詞は特記する必要がなくなることになる。（残念なことに、新バージョンになってから、この「&.」の“動き”がオカシクなってしまった！）

§５　最大値と最小値

```
max=:>./  NB. 右引数の最大値を与える(片側形の)関数
min=:<./  NB. 右引数の最小値を与える(片側形の)関数
```

D1=:>:i.5 NB. 0から4までの5つ数列に0 1 2 3 4に1を加えてD1に挿入する。		
4 >. 5 5	4 <. 5 4	「>.(<.)」という演算子の両側形は、左右の引数の大きい(小さい)方の値を取り出す働きをする。従って、4と5では5の方が大きい(4の方が小さい)から5(4)を出力する。
3 >. 4 >. 5 5	3 <. 4 <. 5 3	右引数のアイテムの後方から順次「>.(<.)」を演算して、結局、最大値(最小値)を出力。
1>.2>.3>.4>.5 5	1<.2<.3<.4<.5 1	
>./ D1 5	<./ D1 1	D1の中では、5(1)が最も大きい(小さい)から、5(1)を出力する。
max D1 5	min D1 1	ユーザーが定義した「代動詞」を用いても同じ結果が得られる。
max 3 5 1 2 4 5	min 3 5 1 2 4 1	データが特に大小順になっていなくても、結果は同じである。
D2=:>:i.4 5 NB. 1から20(=4×5)までの数を、4×5のテーブルに並べてD2に挿入。		
max D2 16 17 18 19 20	min D2 1 2 3 4 5	D2のアイテム間に「<.(>)を挿入するから、縦方向に関しての最大値(最小値)を出力。
max"1 D2 5 10 15 20	min"1 D2 1 6 11 16	「1-セル」のアイテム(アトム)間に挿入するから、横方向の最大値(最小値)を出力。

……「Ｊ」言語メモ ……

Ｊ言語に用意されている「>. 」などは、「原始動詞(atomic verb)」と呼ばれている。これに対して、ユーザーが勝手に定義した動詞を「代動詞(proverb)」と名づけて区別する。つまり、同じ内容のものを「max」ではなく「saidaiti」と表わしても、最大値を求めるという働きは同じである。各原始動詞の働きは固有のものであるのに対して、代動詞の働きの方は如何様にも変化し得るということになる。

「/」はインサート(insert)と呼ばれる「副詞」で、「左側の動詞を引数のアイテム間に挿入する」働きをする。先に定義した「sum=:+/」や「meang=:#%:*/」のような関数にも、この「/」という副詞が使われていた。一般の「副詞」も、その左にある動詞を修飾して、結果も「動詞」となる。

§６　範囲

range=:>./-<./　NB. 範囲を与える(片側形の)関数

]D1=:>:i.5 1 2 3 4 5	>./D1 5	<./D1 1	「>.」と「<. 」とは対をなす演算子で、D1 の各要素の「最大値」と「最小値」を出力。
(>./D1)-(<./D1) 4			range D1 4

]D2＝ : >:i.4 5 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20	]M=:4 7 $ MIN2 2.1 1.5 2.9 3 3.3 3.7 4.7 4.2 4.8 4.9 3 2.9 1.1 1.8 2 3.7 5 3.3 0.5 0.9 0.5 _1.4 _0.4 2.3 3.9 8.4 7.2 6.7	東京地区の平成 9 年 2 月の最低気温のデータ MIN2 を、4×7 のテーブルに成形して M に挿入している。

range D2 15 15 15 15 15 range"1 D2 4 4 4 4	D2 のアイテムの中での最大のものと最小のものとの差(範囲)を与えている。 D2 のランク 1 のセル(リスト)の要素(横方向)についての範囲を求めている。
range M 5.6 5.2 2.7 0.9 7.9 6.3 6.2 range"1 M 3.2 3.8 4.5 9.8	M のアイテムの中での最大値と最小値との差で、縦方向の範囲は実質的意味はない。 最後の 7 日間で、気温の変動が最も激しかったことなどが分かる。

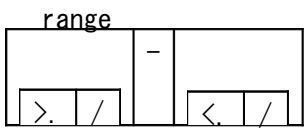
]M4=:2 15 $ MIN4

8 11.1 10.5 10.7 13.3 12.6 10.9 10.3 12.9 12.6 10.5 8.8 9.5 12 10.6
8.9 9.9 11.2 11 12.1 13.5 14.5 9.9 9.6 11.1 11.8 11.3 14.1 13.2 16.1

>./"1 M4 13.3 16.1	<./"1 M4 8 8.9	range"1 M4 5.3 7.2	MIN4 (4 月の最低気温)を、2×15 のテーブルに成形して M4 に挿入

……「J」言語メモ ……

上で定義した「range」という関数に、引数を入力せず「リターン・キー」を押すと



といったように、関数のカラクリがボックスで囲まれた形で表示される。この表示から、「range」という関数は３つの動詞の連なった「フォーク」であることが分かる。

§７　平均偏差

```
mdev=:[:mean[:|(-mean)　NB. 平均偏差(絶対偏差の平均)を求める関数
```

]D1=:>:i.5 1 2 3 4 5	$ MIN2 28	$ MIN3 31	$ MIN4 30	東京地区の平成 9 年 2,3,4 月の最低気温のデータ
(mean D1);(-mean)D1 3 | _2 _1 0 1 2 ([:|(-mean))D1 2 1 0 1 2 ([:mean[:|(-mean))D1 1.2 mdev D1 1.2				D1 というデータの偏差を求めている。「(-mean)y」は「y-mean y」のように演算を行なう「フック」である。 「|」は「絶対値をとる」という演算子で、偏差の絶対値(絶対偏差)を求めている。 絶対偏差の平均で、結局「平均偏差」を求めている。 ユーザーの定義関数で平均偏差を求めている。
mdev MIN2 1.68852	mdev MIN3 1.95088	mdev MIN4 1.44667		2,3,4 月の最低気温のデータの平均偏差を算出している。3 月の気温の変動が激しい。

……「Ｊ」言語メモ ……

「|」という演算子の片側形は「絶対値をとる」という機能をもつが、両側形で用いると

```
   2 | D1=:1 2 3 4 5
1 0 1 0 1
```

といったように、左引数で割算したときの「剰余」を与える。

また、割算の「商(割った結果の整数部）」を求めるには

```
   2 <.@%~ D1
0 1 1 2 2
```

のようにすればよい。さらに

```
   2([*<. @%~)D1
0 2 2 4 4
```

であるから

```
   2(|+[*<.@%~)D1
1 2 3 4 5
```

のように、元の D1 に戻ってしまう。つまり

(A mod B)＋A×[B/A] = B

という関係を示している。

§８ 分散と標準偏差

```
var=:[:mean[:*:(-mean)  NB. 分散を求める関数
sdev=:[:%:var  NB. 標準偏差(分散の平方根)を求める関数
```

```
   T ; Z
┌──────────────┬──────────┐
│80 70 60 90 100│50 60 70 80 90│
└──────────────┴──────────┘
```

太郎と次郎の５回のテストの成績のデータで、「;」という演算子を用いてボックスで囲んで表示している。

```
   > T;Z
80 70 60 90 100
50 60 70 80 90
```

```
   ;T;Z
80 70 60 90 100 50 60 70 80 90
```

「>」は「<」の逆演算で、「ボックスを開く」という働きをする。「；」で開くと結果はリストになってしまう。

```
   (-mean) T
0 _10 _20 10 20
   ([:*:(-mean) T
0 100 400 100 400
   ([:mean[:*:(-mean) T
200
```

Ｔから平均を引いて、平均からの「偏差」を求めている。

偏差を平方している。「*:」の片側形は「平方値」を与える。

直上の結果(偏差平方値)の平均を求めている。

```
   var T
200
```

```
   var Z
200
```

上と同じ結果で、つまりＴというデータの(Ｚというデータも)分散を求めている。

```
   var&> T;Z
200 200
   sdev&> T;Z
14.1421 14.1421
```

```
   >var L:0 T;Z
200 200
   >sdev L:0 T;Z
14.1421 14.1421
```

２組のデータの分散を同時に求めている。

「ｆ L:0」はボックス内でｆの演算を行う。

その後「>」で開いて表示している。

さらに標準偏差の値も同時に求めている。

```
   var L:0 MIN2;MIN3;MIN4
┌─────┬─────┬─────┐
│4.81667│6.04741│3.23006│
└─────┴─────┴─────┘
```

```
   sdev L:0 MIN2;MIN3;MIN4
┌─────┬─────┬─────┐
│2.19469│2.45915│1.79724│
└─────┴─────┴─────┘
```

2,3,4 月の最低気温の分散と標準偏差を算出している。

……「Ｊ」言語メモ ……

「ｆ&>」という関数は、ボックスで囲まれたデータを右引数で与えると、「ｆ」という演算を各ボックスごとに行う(「earch」と呼ばれている)。

```
   var&.> T;Z
┌─────┬─────┐
│200 │200 │
└─────┴─────┘
```

```
   var L:0 T;Z
┌─────┬─────┐
│200 │200 │
└─────┴─────┘
```

演算結果をボックスで囲んだ形にしておきたければ、「ｆ&.>」のように「&.」という接続詞に替えてやればよい。

var&> T ; i.10 200 8.25	データの個数が違っていても、一向に構わない。

§９　分類されたデータの平均と標準偏差

```
meanc =:+/@:*/%+/@:{:   NB. ０行と１行で与えた分類データの平均を求める関数
sdevc=:[:%:[:+/{:*[:*:{.-meanc   NB. ０行と１行で与えた分類データの標準偏差
```

TEST 40 42 44 46 48 50 52 54 56 58 60 1 3 7 12 17 20 17 12 7 3 1		最初の行が級代表値で、最後の行が度数を示す分類された合計 100 個の分類データ。
*/ TEST 40 126 308 552 816 1000 884 648 392 174 60		アイテム間の積を与えている。
+/@(*/) TEST 5000	+/@:*/ TEST 5000	上の結果の総和で、分類されたデータの総和を求めている。
(+/@:*/%+/@:{:) TEST 50 meanc TEST 50 ([:mean{:#{.)TEST 50		上の総和をデータの総数「+/@:{:」で割った値で、分類されたデータの平均値 上と同じ結果で、「meanc」は分類されたデータの平均を求める関数である。 「{:#{.」によって、(1 行)の個数だけ(0 行)の値をコピーするので、結局、100 個の分類されてないデータの平均をとっても同じ結果が得られる。

({.-meanc)TEST _10 _8 _6 _4 _2 0 2 4 6 8 10	分類されたデータの級代表値の平均からの偏差を与えている。
([:*:{.-meanc)TEST 100 64 36 16 4 0 4 16 36 64 100	平均からの偏差の平方値を与えている。
({:*[:*:{.-meanc)TEST 100 192 252 192 68 0 68 192 252 192 100	偏差の平方値に、度数を掛けているフォーク 上の結果の合計値で、分類されたデータの偏差平方和を与えている。
([:+/{:*[:*:{.-meanc)TEST	

1608 　((\[:+/{:*\[:*:{.-meanc)%+/@:{:)TEST 16.08 　sdevc TEST 4.00999 　(\[:sdev{:#{.)TEST 4.00999	偏差平方和をデータの総数「＋／@{：」で割った値で、分散の値を与えている。 直上の結果の平方根で、「sdevc」は分類されたデータの標準偏差を与える関数。 「{:#{.」によって、100 個の分類されてないデータの標準偏差をとっても同じ結果

§１０ 累積度数法による平均と標準偏差

```
sub=:+/¥&.|.  NB. rsk1 rsk2 を求めるための補助関数
rsk1=:[:+/sub  NB. 第１累積度数の和を求める関数
rsk2=:[:+/[:}.sub^:2 NB. 第２累積度数の和を求める関数
meanr=:{.@[+{:@[*([:<:rsk1%+/)@] NB. 等間隔分類されたデータの平均を求める関数
sdevr=:[:%:*:@[*((+:@rsk2+rsk1([-*:@[%+/@])])%+/)@] NB. 分類データの標準偏差
```

F=:1 3 7 12 17 20 17 12 7 3 1	級代表値の小さいほうからの度数分布
(sub=:+/¥&.|.)F 100 99 96 89 77 60 40 23 11 4 1	後方からの度数の逐次和で、第１累積度数が与えられる。
]S=:rsk1 F 600	上の第１累積度数の合計をＳに挿入し表示している。
40 2({:@[*([:<:rsk1%+/)@])F 10	「(S÷N)-1」に左引数の末尾の要素(級間隔)を掛けた値である。
40 2 meanr F 50	上の結果に左引数の先頭の要素(最小の級代表値)を足せば平均値を与える。
sub^:2 F 600 500 401 305 216 139 79 39 16 5 1	第１累積度数の逐次和で、第２累積度数を与えている。
]T=:rsk2 F 1701	上の第２累積度数で先頭の要素を取り除いた合計値をＴに挿入し表示している。
(+:T)+S([-*:@:[%])N=.+/F 402	「$2T+S-S^2/N=+/F$」という値を計算している。
%:(*:2)*((+:T)+S([-*:@:[%])N)%N 4.00999	「$(2T+S-S^2/N)\div N$」に級間隔の２乗を掛けてから平方根をとった値である。
2 sdevr F 4.00999	上と同じ結果で、等間隔で分類されたデータの標準偏差を与えている。

等間隔に分類されたデータの度数 $\{f_1, f_1, \cdots\cdots, f_k\}$ が与えられたとき、平均(M)や分散(V)は第１累積度数 $\{g_j\}$ と第２累積度数 $\{h_j\}$ を用いて次式で与えられる：

$$S = \sum_{i=1}^{k} g_i \ (g_j = \sum_{i=j}^{k} f_i \ ; 1 \le j \le k) \quad ; \quad T = \sum_{i=2}^{k} h_i \ (h_j = \sum_{i=j}^{k} g_i \ ; 2 \le j \le k)$$

$$M = x_1 + w\{S/n - 1\} \quad ; \quad V = w^2\left\{2T + S - S^2/n\right\}/n$$

§１１ 偏差値

```
sdev=:[:%:[:mean[:*:(-mean=:+/%#)  NB. 標準偏差を求める関数
hnst=:50”_+10”_*(-mean)%sdev  NB. 偏差値を求める関数
meanc =:+/@:*/%+/@:{:  NB. ０行と１行で与えた分類データの平均を求める関数
sdevc=:[:%:[:+/{:*[:*:{.-meanc  NB. ０行と１行で与えた分類データの標準偏差
hnstc=:50”_+10”_*({.-meanc)%sdevc  NB. 分類データの偏差値を求める関数
```

```
   ]T=:45+2*i.11
45 47 49 51 53 55 57 59 61 63 65
   (mean,sdev)T
55 6.32456
   (-mean)T
_10 _8 _6 _4 _2 0 2 4 6 8 10
```

11 人の生徒の試験の成績のデータ

T の平均と標準偏差を求めている。

T の各要素から平均の 55 を引いた値で、平均からの偏差を与えている。

```
   6{. Z=: ((-mean)%sdev)T
_1.58114 _1.26491 _0.948683 _0.632456 _0.316228
0
```

上の偏差の値を標準偏差で割った結果を Z に挿入 6 個の値を表示

```
   50+10*Z
34.1886 37.3509 40.5132 43.6754 46.8377 50 53.1623 56.3246 59.4868 62.6491
65.8114
 NB. Z の 10 倍した値に 50 を加えた数値
   hnst T
34.1886 37.3509 40.5132 43.6754 46.8377 50 53.1623 56.3246 59.4868 62.6491
65.8114
   0.1": hnst T
34.2 37.4 40.5 43.7 46.8 50.0 53.2 56.3 59.5 62.6 65.8
```

NB.直上の値を、書式関数「“:」を用い１桁で”四捨五入“した値(数値ではなく文字)。

```
   ]TEST=:T,:1 3 7 12 17 20 17 12 7 3 1
45 47 49 51 53 55 57 59 61 63 65
 1  3  7 12 17 20 17 12  7  3  1
```

```
   (meanc,sdevc)TEST
55 40.0999
  ({.-meanc)TEST
_10 _8 _6 _4 _2 0 2 4 6 8 10
```

```
   50+10*Z=:(({.-meanc)%sdevc)TEST
```

```
47.5062  48.005  48.5037  49.0025  49.5012  50  50.4988  50.9975  51.4963  51.995
52.4938
   hnstc TEST
47.5062  48.005  48.5037  49.0025  49.5012  50  50.4988  50.9975  51.4963  51.995
52.4938
   0.1″:hnstc TEST
47.5 48.0 48.5 49.0 49.5 50.0 50.5 51.0 51.5 52.0 52.5
```

§１２ 変動係数

```
sdev=:[:%:[:mean[:*:(-mean=:+/%#)  NB. 標準偏差を求める関数
coefv=:100"_*sdev%mean  NB. 変動係数を求める関数(関数型定義)
coef=:3 :' 100*(%:mean*:y.-m)%m=.mean y.' NB. 明示的定義による変動係数
```

SM 9820 13836 11506 8330 8761 11744 12769 15356 12316 13540 SW 6491 8641 8637 6114 6967 8231 7773 8027 7772 6976	男子の年間自殺者数 (5 年間隔の年次データ) 女子の年間自殺者数
mean L:0 SM;SW 11830.9 |7562.9	sdev L:0 SM;SW 2334.55 837.462 女子の自殺者のほうが大分少ない。 男子の自殺者数のほうの変動が激しい。
100*(sdev%mean)SM 19.7327	標準偏差を平均で割って 100 倍した値で、変動係数(%表示)と呼ばれるもの。
coefv SM 19.7327 coef SM 19.7327	coefv SW 11.0733 coef SW 11.0733 ユーザーの定義関数を用いても同じ結果が得られる。 明示的定義による関数を用いても同じ結果が得られる。
coefv L:0 MIN2;MIN3;MIN4 71.042 36.18 15.7422	2,3,4 月の最低気温のデータの変動係数を算出している。 2 月の値が異常に大きくなっているが、これは 2 月の気温の平均が小さいことに起因している(変動係数の欠陥！)。

……「Ｊ」言語メモ ……

これまでは、原子動詞や副詞等を直接連結する形で、いろいろな関数を定義してきた。このような「関数型定義」の方法は、Tacit Definition と呼ばれている。ところがＪ言語には、Explicit Definition と呼ばれる、もう一つ別の方法で関数を定義することもできる。これを「明示的定義」と名づける。

まず、「coef」といった動詞を定義するときには、

```
coef=:3 :' ＊＊＊＊＊……'
```

のように「3 :」の後に「シングルクォーテーション(')」で囲むことと、右引数の変数として「y. 」を用いることである。(変数の文字の後に「ピリオド(.)」をつけないとエラーになる。) さらに、「MEAN=:13 :'(+/y.)%#y.'」は明示的定義であるが

13 :'(+/y.)%#y.'	「13 :」の後に(')で囲んだ明示的定義を入力して「リターンキー」を押すと関数型定義の内容が表示される。

+/ % #	

§１３ メディアン

```
median=:[:-:@+/([:(<:,-)[:>.-:@#){/:~  NB. メディアンを求める関数
```

/:~D=:2 4 6 8 10 12 14 16 2 4 6 8 10 12 14 16	Dというリストを「/:~」により昇順形に並べ変えている。
]M=:([:>.-:@#)D 4	Dのアイテム数(個数)を半分にしてから、小数点以下を切捨ててMに挿入する。
]I=:(<:,-)M 3 _4	Mというインデックスの１を引いた値と符号を変えた値とを連結している。
I{/:~D 8 10	Dの昇順形の先頭から４番目と末尾から４番目の要素を取り出している。
-:@+/ 8 10 9	右引数の要素を足してから「-:」で半分にしている。結局、平均を求めている。
median D 9	上と同じ結果で、「median」はデータの昇順形の４番目と５番目の要素の平均。
median D,18 10	奇数個のデータの場合には中央の(この場合は５番目の)要素を与えることになる。
median >:i.10 5.5	５番目と６番目の要素の平均を与えている。

……「Ｊ」言語メモ ……

「/:」という演算子は、片側形で用いると

```
   /:D
4 1 6 2 0 7 3 5
```

のように昇順の「インデックス」を与えている。つまりこの結果の左端の４は、Dの要素の最小値１は０から数えて４番目（通常の５番目）の位置にあることを示している。

「{」の両側形は、「選択(from)」と呼ばれている演算子で、

2{1 2 3 4 5 6 3	先頭から３番目の要素が選択される。
_2{1 2 3 4 5 6 5	末尾から２番目の要素が選択される。

といったように、左引数で指定した数値の要素を右引数のリスト(一般にはアレイ)から選択する。ここで左引数がプラスの場合には、2は0から数えているので、先頭から３番目の

要素が選択される。(「選択(from)」の左引数に、正数と負数を与えた場合には、順番に“整合性を欠いている”点に注意を要する。)

§１４ 四分位数と四分位偏差

```
qtl1=:-:@+/@(>.@(<:,-&0.5)@(#%4:){/:~)　NB. 「第１四分位数」を与える関数
qtl3=:-:@+/@(-@>.@(],+&0.5)@(#%4:){/:~)　NB. 「第３四分位数」を与える関数
qtl=:qtl3-qtl1)%2:　NB. 右引数で与えたデータの四分位偏差を与える関数
```

]A=:(#%4:)D=:5 2 4 7 1 8 3 6 2	「4:」は左右の引数に関係なく、常に「4 を取り出す」という動詞で、「#%4:」はフォークで、「(#D)%4＝8%4＝2」のように演算。
(<:,-&0.5)A 1 1.5	A=2 から 1 を引いた値と 0.5 引いた値とを取り出している。
]B=:>.@(<:,-&0.5)A 1 2	上の値を「>.」という演算子で切り上げて整数にしている。
]C=:B{/:~D 4 6	D の昇順形の 1、2 番目(通常の 2、3 番目)の要素を取り出して C に挿入している。
-:@+/ C 5	C の要素を足してから半分にしている。
qtl1 D 5	上と同じ結果で、D の昇順形の 2、3 番目の要素の平均で「第１四分位数」を与える。
qtl1 >:i.10 3	データ数が 10 個のときには、小さいほうから 3 番目の要素が「第１四分位数」になる。
]K=:-@>.@(],+&0.5)2 _2 _3	末尾からの「インデックス」を出力して K に挿入している。
K {/:~ D 14 12	D の昇順形の末尾から 2、3 番目(_2 _3)の要素を取り出している。
qtl3 D 13	「第 3 四分位数」を求めるプログラムも同様に定義できる。
qtl D 4	第 3 四分位数(13)と第 1 四分位数(5)の差の半分が「四分位偏差」である。

……「Ｊ」言語メモ ……

左右の引数に関係なく常に定数を取り出す関数は、0 から 9 までと_1 から_9 までの整数に「コロン(:)」をつけて定義できる。しかし「0.5」を引く場合には「-&0.5」のように記述しなければならない。また、任意の数字を“動詞化”するには「10 “_」や「(o.2)” _」といった

ようにすればよい。

§１５ 平均差

```
meandif=:([:+/[:|[:,-/~)%2:*2:!#  NB. 右引数で与えたデータの平均差を与える
```

```
   D1=:1 2 3 4 5
   ]A=:(2:*2:!#)D1
20
   ]B=:-/~ D1
0 _1 _2 _3 _4
1  0 _1 _2 _3
2  1  0 _1 _2
3  2  1  0 _1
4  3  2  1  0
   ]C=:+/|,B
40
```

「2！5」」は 2 項係数「${}_5C_2$」を与える演算子で、「2:*」で結果を 2 倍にしている。

「-/~D1」は「D1-/D1」と同じで、D1 の各要素の引き算に関するクロス表を B に挿入している。

上の行列 B を「, 」でベクトル化し、「|」で絶対値をとってから加えている。

```
   C % A
2
```

```
   meandif D1
2
```

D1 というデータの「平均差(係数）」を与えている。

```
   meandif 2+D1
2
   sdev D1
1.41421
   meandif 2.8 2.9 3 3.1 3.2
0.2
   sdev 2.8 2.9 3 3.1 3.2
0.141421
```

データ全体をずらしても、平均差の値は変わらない（“係数”という表記は変だ！）。

平均差は標準偏差と同じような値になる。

変動幅が 10 分の 1 になれば、平均差の値も 10 分の 1 になる。

標準偏差の値も 10 分の 1 になる。

```
   meandif L:0 MIN2;MIN3;MIN4
```

2.53466	2.80903	2.07931

東京地区の平成 9 年 2,3,4 月の最低気温のデータの「平均差」を算出している。

……「統計学」メモ ……

ｎ個のデータ $\{x_i, x_i, \cdots\cdots, x_n\}$ が与えられたとき

$$d = \frac{1}{n(n-1)} \sum_{i=1}^{n} \sum_{j=1}^{n} |x_i - x_j|$$

によって定義されるものを「平均差（係数）」と呼んでいる。標準偏差は“平均からのずれ”の平均であるが、この平均差のほうは“各データ間のずれ”の平均である。

§１６ 級間分散(BV)と級内分散(WV)

```
var=:[:mean*:@(-mean=:+/%#)  NB. 分散を求める関数
weight=:[:(%+/)#&>  NB. ウエイトを与える関数
varb=:[:+/weight*[:*:mean&>-mean@;  NB. 級間分散を求める関数
varw=:[:+/weight*var&>  NB. 級内分散を求める関数
```

KT;KZ 80 70 60 90 100 | 50 60 70 80 90 mean&> KT;KZ 80 70		太郎と次郎の国語の成績(架空のデータ) 2組のデータの平均を出力している。
#&>KT;KZ 5 5	(%+/)#&> KT;KZ 0.5 0.5	2組のデータの個数と標本比率のウェイトを与えている。「%+/」はフックである。
;KT;KZ 80 70 60 90 100 50 60 70 80 90		２組のデータを合わせてリストにしている。「;」は「,@:>」と同じである。
mean@; KT;KZ 75	(mean&>-mean@;)KT;KZ 5 _5	上記の複合データの平均を出力している。各組の平均から、複合データの平均を引いている。これもフォークである。
(weight*[:*:mean&>-mean@;)KT;KZ 12.5 12.5 varb KT;KZ 25		上の結果の平方値にデータ数のウェイトを掛けている。 直上の結果の合計で、「varb」は複数組のデータの級間分散を求める関数である。
var&>KT;KZ 200 200	(weight*var&>)KT;KZ 100 100	2組のデータの分散と、それにウエイトを掛けた値である。
varw KT;KZ 200		2組のデータの級内分散を求めている。

……「統計学」メモ ……

m 個のデータの平均を $M(x)$，分散を $V(x)$、n 個のデータの平均を $M(y)$，分散を $V(y)$ とするとき２組のデータを合わせた複合データの平均 $M(z)$ と分散 $V(z)$ は

$$M(z) = \alpha M(x) + (1-\alpha)M(y)$$
$$V(z) = \alpha V(x) + (1-\alpha)V(y) + \alpha\{M(x) - M(z)\}^2 + (1-\alpha)\{M(y) - M(z)\}^2$$

のように表される。ここで、$\alpha = m/(m+n)$ は標本比率である。

§１７ 相関比

```
varb=:[:+/weight*[:*:mean&>-mean@;  NB. 級間分散を求める関数
varw=:[:+/weight*var&>  NB. 級内分散を求める関数
cratio=:varb([%+]varw  NB. 複数組のデータの相関比を求める関数
```

```
   KT;KZ
┌──────────┬──────────┐
│80 70 60 90 │50 60 70 80 │
│100         │90          │
└──────────┴──────────┘
ST;SZ
┌──────────┬──────────┐
│78 79 82 81 80 │70 69 71 68 │
│               │72          │
└──────────┴──────────┘
(mean,var)&.> KT;KZ;ST;SZ
┌──────┬──────┬──────┬──────┐
│80 200 │70 200 │80 2  │70 2  │
└──────┴──────┴──────┴──────┘
```

太郎と次郎の国語の成績

太郎と次郎の数学の成績

4組のデータの平均と分散を求めている。

```
   varw KT;KZ
200
   varb KT;KZ
25
   var@; KT;KZ
225
```

級内分散と級間分散の和は全体の分散と一致する。

```
   25([%+)200
0.111111
```

25%(25+200)の値を求めている。

```
   cratio KT ; KZ
0.111111
   cratio ST;SZ
0.925926
```

国語よりも数学の成績のほうが顕著な違いがある。

……「統計学」メモ ……

m 個のデータの平均を $M(x)$, 分散を $V(x)$、n 個のデータの平均を $M(y)$, 分散を $V(y)$ さらに２組のデータを合わせた複合データの平均を $M(z)$,分散を $V(z)$ とするとき

級間分散(BV)と級内分散(WV)は

$$BV = \alpha\{M(x) - M(z)\}^2 + (1-\alpha)\{M(y) - M(z)\}^2$$
$$WV = \alpha V(x) + (1-\alpha)V(y) \quad \alpha = m/(m+n)$$

のように与えられる。さらに、複合データの分散 $V(z)$ は次のように表される。

$V(z) = BV + WV$ （全分散の分解）

「相関比」とは

$$H = \frac{BV\text{（級間分散）}}{WV\text{（級内分散）} + BV\text{（級間分散）}}$$

のように定義され、平均値間の差の相対的な尺度を与えるものである。

§１８ 順序尺度データの散布度

```
devdif=:[:+/}:*[:}.(+/¥^:2)&.|.   NB. u-関数を求める補助関数
ordv=:devdif%2:!+/  NB. 順序尺度データの散布度(ｄ)を求める関数
```

F=:7 1 2 1

力士の連敗のパターン分布で、勝ちが７、単敗、２連敗、３連敗が１，２，１度ある。

```
   ]H=:+/¥^:2&.|.F
19 8 4 1
   ]U=:+/(}.F)*}.H
62
```

```
   (}.F)*}.H
56 4 2
   devdif F
62
```

度数分布Ｆの第２累積度数を求めている。

Ｆの先頭の要素を落したリストとＨの末尾の要素を落したリストの積。

上のリストの総和で、「ｕ関数」を求めてＵに挿入し表示している。

```
   ]C=:(2:!+/)F
55
```

Ｕの値を異なる組合せの総数「C=n×(n-1)÷2」の値で割っている。

```
   U % C
1.12727
```

```
   ordv F
1.12727
```

「ordv」は順序尺度データの散布度(ｄ)を求める関数である。

```
   ]A=:|-/~i.4
0 1 2 3
1 0 1 2
2 1 0 1
3 2 1 0
```

```
   ]B=:*/~F
49 7 14 7
 7 1  2 1
14 2  4 2
 7 1  2 1
```

```
   A*B
 0 7 28 21
 7 0  2  2
28 2  0  2
21 2  2  0
```

```
   (-:+/,A*B)
62
   (-:+/,A*B)%C
1.12727
```

……「統計学」メモ ……

順序尺度のカテゴリカルデータが与えられている場合に、ｉ番目のカテゴリーの観測度数を $\{f_i\}$ とする(i＝0,1, …… ,m)。隣あったカテゴリー間の距離を１と考えると、「平均差」に対応するものは

$$d = \frac{1}{n(n-1)} \sum_{i \neq j}^{n} |i-j| f_i f_j$$

によって定義される。

そこで、第１、第２累積度数を

$$g_j = \sum_{i=j}^{k} f_i \,(1 \le j \le k) \quad ; \quad h_j = \sum_{i=j}^{k} g_i \,(2 \le j \le k)$$

のように定義すると、

$$d = \frac{u}{n(n-1)} \quad ; u = \sum_{j=0}^{m} \sum_{j+1}^{m} (i-j) f f_j = \sum_{k=1}^{m} f_{k-1} h_k \quad (n = \sum_{i-1}^{k} f_i)$$

で与えられるものが、順序尺度データの散布度を表わすことになる。

§１９ カテゴリカルデータの散布度

```
catu=:1:-+/@:*:%[:*:+/ NB. カテゴリカルデータの散布度(U係数)
catv=:%:@(#%+/)*catu NB. カテゴリカルデータの散布度(V)を求める関数
```

TAKA=:7 3 2 1 1		貴乃花の平成８年初場所の決まり手の分布（カテゴリー間に順序なし！）
]A=:([:*:+/)TAKA 196]B=:([:+/*:)TAKA 64		TAKAの要素の総和(n)の平方値を求めてAに挿入している。 TAKAの各要素の平方値の総和を求めてBに挿入している。
A % B 0.326531	(+/@:*:%[:*:+/)TAKA 0.326531	TAKAの要素の平方和(A)を要素の総和の平方値(B)で割る。
1-0.326531 0.673469	]U=:catu TAKA 0.673469	TAKAというカテゴリカルデータの散布度(U係数)を与えている。
(#%+/)TAKA 0.357143	]C=:%:@(#%+/)TAKA 0.597614	TAKAのアイテム数を要素の総和で割った値とその平方根をCに挿入している。
C * catu TAKA 0.402475	catv TAKA 0.402475	「catu」を演算した結果に$\sqrt{K/N}$を掛けた値で、「U係数」を修正した「V係数」を与えている。

……「統計学」メモ ……

順序関係のないｉカテゴリーの観測度数数を$\{f_i\}$とする(i＝01,2, …… ,k)とき、

$$U = 1 - \frac{1}{n^2}\sum_{i-1}^{k} f_i^{\,2} \qquad (n = \sum_{i-1}^{k} f_i)$$

といったものが、平均差に対応するカテゴリカルデータの散布度と解釈できる。

しかし、観測数が多いときには、次のように修正した係数を用いるほうがよい。

$$V = \frac{\sqrt{k}}{\sqrt{n}}\left\{1 - \frac{1}{n^2}\sum_{i-1}^{k} f_i^{\,2}\right\}$$

【関数の script】

```
integer=:>:@i.   NB.整数列を生成する(片側形の)関数

sum=:+/  NB. 総和を求める(片側形の)関数
mean=:+/%#  NB. 算術平均を求める(片側形の)関数
meang=:#%:*/  NB. 幾何平均を求める(片側形の)関数
meanh=:[:%[:mean%  NB. 調和平均を求める(片側形の)関数
max=:>./  NB. 右引数の最大値を与える(片側形の)関数
range=:>./-<./  NB. 範囲を与える(片側形の)関数
mdev=:[:mean[:|(-mean)  NB. 平均偏差(絶対偏差の平均)を求める関数
var=:[:mean[:*:(-mean)  NB. 分散を求める関数
sdev=:[:%:var  NB. 標準偏差(分散の平方根)を求める関数
meanc =:+/@:*/%+/@:{:  NB. 0行と1行で与えた分類データの平均を求める関数
sdevc=:[:%:[:+/{:*[:*:{.-meanc  NB. 0行と1行で与えた分類データの標準偏差
sub=:+/¥&.|.  NB. rsk1 rsk2 を求めるための補助関数
rsk1=:[:+/sub  NB. 第1累積度数の和を求める関数
rsk2=:[:+/[:}.sub^:2 NB. 第2累積度数の和を求める関数
meanr=:{.@[+{:@[*([:<:rsk1%+/)@] NB. 等間隔分類されたデータの平均を求める関数
sdevr=:[:%:*:@[*((+:@rsk2+rsk1([-*:@[%+/@])])%+/)@] NB. 分類データの標準偏差
hnst=:50”_+10”_*(-mean)%sdev  NB. 偏差値を求める関数
hnstc=:50”_+10”_*({.-meanc)%sdevc  NB. 分類データの偏差値を求める関数
coefv=:[:100”_+ sdevc%mean  NB. 変動係数を求める関数
median=:[:-:@+/([:(<:,-)[:>.-:@#){/:~  NB. メディアンを求める関数
qtl1=:-:@+/@(>.@(<:,-&0.5)@(#%4:){/:~)  NB. 「第1四分位数」を与える関数
qtl3=:-:@+/@(-@>.@(],+&0.5)@(#%4:){/:~)  NB. 「第3四分位数」を与える関数
qtl=:qtl3-qtl1)%2;  NB. 右引数で与えたデータの四分位偏差を与える関数
meandif=:([:+/[:|[:,-/~)%2:*2:!#  NB. 右引数で与えたデータの平均差を与える
weight=:[:(%+/)#&>  NB. ウエイトを与える関数
varb=:[:+/weight*[:*:mean&>-mean@;  NB. 級間分散を求める関数
varw=:[:+/weight*var&>  NB. 級内分散を求める関数
cratio=:varb([%+]varw  NB. 複数組のデータの相関比を求める関数
devdif=:[:+/}:*[:}.(+/¥^:2)&.|.  NB. u-関数を求める補助関数
ordv=:devdif%2:!+/  NB. 順序尺度データの散布度(d)を求める関数
catu=:1:-+/@:*:%[:*:+/  NB. U係数を求める補助関数
catv=:[:%:(#%+/)*catu  NB. カテゴリカルデータの散布度(V)を求める関数
```

【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

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

【J言語の特徴】

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)

```
   (sum=:+/)D=:3 1 2
6
   (sum1=:3 :'+/y.')D
6
```

```
   (mean=:+/%#)D
2
   3 :'(+/y.)%#y.'D
2
```

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)

```
   +/D=:3 1 2
6
```

```
   # D
3
```

```
   (+/%#)D
2
```

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)」

```
   (-mean=:+/%#)D
1 _1 0
```

```
   (]-mean)D
1 _1 0
```

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)

```
   (]-+/%#)D
1 _1 0
```

```
   (-+/%#)D
1 _1 0
```

動詞との　出会いひたすら　待つ「副詞」

右にこだわる　「接続詞」

```
   av=:/
4!:0<'av'
1 NB. 副詞
```

```
   c=:&
4!:0<'c'
2 NB. 接続詞
```

```
   mean=:+/%#
4!:0<'mean'
3 NB. 動詞
```

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

```
   ]d=:1 2 3;4 5
┌─────┬───┐
│1 2 3│4 5│
└─────┴───┘
   +/L:0 d
┌─┬─┐
│6│9│
└─┴─┘
```

```
   +/&> d
6 9
   +/&.> d
┌─┬─┐
│6│9│
└─┴─┘
```

(+/ .*)は　行列同士の掛算を　行う便利な

プリミティブ

```
   C +/ .* |:C=:i.2 3
 5 14
14 50
```

(-/ .*)は　片側動詞で　正方行列の

行列式の値を　出力す

```
   -/ .* A=:1+I.2
2
2
```

```
   (1*4)-(1*2)
2
```

([%1:,.])は　回帰係数を　出力する

```
   v=:3 5 6 [ u=:1 2 4
```

チョー便利な　定義関数	v ([%1:,.]) u 2.5 0.928571

【 “+ +. +:” 】

実数に　プラス(+)の片側　そのままで 　複素数には　共役複素数	+ 0.4 0 3 0.4 0 3	+ 3j4 3j_4
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	1 2 3 + 4 5 6 5 7 9	1j2 + 3j4 4j6
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	+. 3 3 0	+. 3j4 3 4
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)	6 +.8 2	1j2 +. _1j2 0j1
	1j2 % 0j1 2j_1	_1j2 % 0j1 2j1
プラスコロン(+:)　右引数を　倍にする 　両側形は　機能無し	+: 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	- _5 0 2 5 0 _2	- 3j4 _3j_4
マイナス(-)の両側　フツーの引き算 複素数でも　イッツオーケー	4 5 6 - 1 3 2 3 2 4	3j4 - 1j2 2j2
マイナスピリ(-.)の　片側形は 　足して１になる　補数を出力	-. 4 0.3 1 _3 0.7 0	-. 1j2 0j_2
マイナスピリ(-.)の　両側形は 　右引数に　無いモノを出す	(a=:1 2)-.b=:1 3 5 2	(-.a e.b)#a 2
マイナスコロン(-:)の　片側形は 　右引数を　半分にする	-: 2 4 6 1 2 3	-: 2j4 1j2
マイナスコロン(-:)の　両側形は 　形まで含めて　一致か否か	1 2 -: 1 2 1	1 2 = 1 2 1 1

ダブルクォート(“)　ピリ(.)で数値化 コロン(:)で文字化　書式も与える　スグレモノ	1+″:2 |domain error |　1　+″:2	1+″.″:2 3
	a=:’1+2+3’ 　　″. a	″. a 6
	]b=:’1 2 3’,’4 5’,:’’	8 ″. b

	1 2 3 4 5	1 2 3 4 5 8 8 8 8

【“* *. *:”】

シグナム(*)は　符号与える　片側形 複素数には　単位円に射影	* _3 0 2 _1 0 1	* 3j4 0.6j0.8
スターピリ(*.)　複素数には　大きさと 偏角与える　片側関数	*. _3 2 3 3.14159 2 0	*._3j0 3 3.14159 *. 3j4 5 0.927295
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)	4 *. 6 12	1j1 *. 3j4 7j1
スターコロン(*:)は　平方値 平方根なら　パーセントコロン(%:)	*: 4 16	%: 4 2

【“% %. %:”】

パーセント(%)　片側形なら　逆数で 両側形なら　割算を行う	% 2 4 5 0.5 0.25 0.2	2 4 5 % 2 1 2 2.5
行列の　割算行う　パーセントピリ(%.) 片側形なら　逆行列	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
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 9 1.41421 2 3	3 %: 8 27 2 3

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！)	i. 3 0 1 2	i._3 2 1 0
マイナスの　整数値まで　出力す iにコロン(i:)は　重宝動詞	i: 3 _3 _2 _1 0 1 2 3	i: _3 3 2 1 0 _1 _2 _3
(i.)の両側形は　左指定の　数のインデックスを　右引数のリストに　与える	7 8 9 i.8 7 9 7 1 0 2 0	7 8 9 i.8 7 9 6 1 0 2 3

コンマ(,)という　動詞の　片側形は 右引数を　リストに変換	]t=:i.2 3 0 1 2 3 4 5	,t 0 1 2 3 4 5
コンマにピリ(,.)の　片側形は 右引数を　テーブル化する	]a=:i.2 0 1	,. a 0 1
コンマにコロン(,:)の　片側形は ランクを１つ　上げたアレイに	,: a 0 1	$,: a 1 2

コンマ(,)という　動詞の　両側形は ランクを変えずに　左右を接続	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
コンマにピリ(,.)の　両側形は 左右の引数を　横に接続	1 2 3 ,. 4 5 6 1 4 2 5 3 6	(|:A),.b 0 3 6 1 4 7 2 5 8
コンマにコロン(,:)の　両側形は ランクを上げた　アレイになる	1 2 3 ,: 4 5 6 1 2 3 4 5 6	A,:4 5 0 1 2 3 4 5 4 5 0 0 0 0

ビックリ(!)の　片側形は　階乗よ 両側形は　２項係数	! 3 4 5 6 24 120	! 0.5 1.5 0.886227 1.32934
	2 ! 5 10	(bic=:i.@>:!])5 1 5 10 10 5 1
	^|. i.3 7.38906 2.71828 1	1x1^|.i.3 7.38906 2.71828 1
	(^!._1)~c=:2 3 4	*/@(]+_1:*i.)"0 c 2 6 24

	2 6 24	

【 “| |. |:” , “+/ .*” , “-/ .*” , “[% 1:,.]” 】

割算の　余り求める　棒(|)一本 片側形なら　絶対値	3	i.6 0 1 2 0 1 2	| 1 _2 3 _4 1 2 3 4
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	|.d=:1 2 3 4 5 5 4 3 2 1	_1 |. d 5 1 2 3 4
棒ピリ(|.)の　両側形は　左の数だけ 右に回転(rotate)　負なら左へ	1 |.d 2 3 4 5 1	2 |.d 3 4 5 1 2
棒にコロン(|:) 片側形なら アレイの軸の 順序をソックリ　入れ替える	m;|:m=:2 3$'abcdef' abc Ad def be cf		
棒にコロン(|:)　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	(1 0|:m);(1|:m);(<0 1)|:m ad abc Ae be def cf		

	]n=:i.2 3 0 1 2 3 4 5	2 3 [4 5 2 3 2 3] 4 5 4 5

	f=:*: :. %: f i.5 0 1 4 9 16 f^:_1 f i.5 0 1 2 3 4	g=:*: :. +: g i.5 0 1 4 9 16 g^:_1 g i.5 0 2 8 18 32

【アレイの形と変形】

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	$ 2	# 2 1	$ 2 1 2	# 2 1 2
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	]M=:i.2 3 0 1 2 3 4 5	$ M 2 3	# M 2	
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)	0 1 1 # 1 3 5 3 5	0 1 # M 3 4 5		
シャープ・ピリ(#.)　片側形は　２進数 10進数の　数値に変換	#.1 0 1 5	+/(1 0 1)*2^2 1 0 5		
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	10　#. d=.1+i.4 1234 8 #. d 668	+/d*(10^3 2 1 0) 1234 +/d*(8^3 2 1 0) 668		
シャープ・コロン(#:)　片側形は 10進数を　２進の数値に　変換す	]b=:#:3 5 7 0 1 1 1 0 1 1 1 1	#.b 3 5 7		
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	t=:24 60 60]s=:t #.2 3 4 7384 t #: s 2 3 4	]tt=:(*/¥.}.t),1 3600 60 1 +/2 3 4*tt 7384		
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	2 {. K 1 2	2 }. K 3		

(i.)の両側形は　左指定の　数のインデックスを　右引数のリストに　与える	; i.2 3 0 1 2 3 4 5	;/ i.2 3 0 1 2 3 4 5

【接続詞】

<table>
<tr><td>片側の　動詞を順に　結ぶのが

　アンド(&)やアット(@ @:)の　接続詞</td><td>*:&+: 2

16</td><td>*:@+: 2

16</td><td>*:@:+: 2

16</td></tr>
<tr><td>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</td><td colspan="2">*:&.+: 2

8</td><td>-:&*:&+: 2

8</td></tr>
<tr><td>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</td><td>([:*:+:)2

16</td><td>([:%:+:)2

2</td><td>%:&+: 2

2</td></tr>
<tr><td>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)</td><td>*&2 a=:2
3

4 6</td><td>2&* a

4 6</td><td>+: a

4 6</td></tr>
<tr><td>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</td><td colspan="2">+`*/ i.6

29
+`%/ 3 1 4

3.25</td><td>0+1*2+3*4+5

29
3 + 1 % 4

3.25</td></tr>
<tr><td>(u`v`:0)は　全ての動詞を　演算し
(u`v`:3)は (u`v/)と　同じに機能<br>(u`v`:6)は　(uv)というフック<br>(u`v`w`:6)は　(uvw) というフォーク
と同じ演算</td><td colspan="2">+:`-:`:0(3 2 4)

6 4 8

1.5 1 2
+`*`:3 a=:1 2 3

7
+`*`:6(3 2 5)

4 3 6
+`*`-`:6(3 2 5)<br><br>_9 _4 _25</td><td>(+:,:-:)3 2 4<br><br>6 4 8<br><br>1.5 1 2<br>+`*/ a

7
(+*) 3 2 5

4 3 6
(+*-)3 2 5

_9 _4 _25</td><td>1+2*3

7</td></tr>
<tr><td>Even(..)は　２つの動詞を　接続し

別の動詞を　作り出す</td><td colspan="2">*: .. +: 4

40
+: .. *: 4

20</td><td>-:@(*:+*:&+:)4

40
-:@(+:++:&*:)4

20</td></tr>
<tr><td>Odd(.:)は　２つの動詞を　接続し

別の動詞を　作り出す</td><td colspan="2">*: .: +: 4

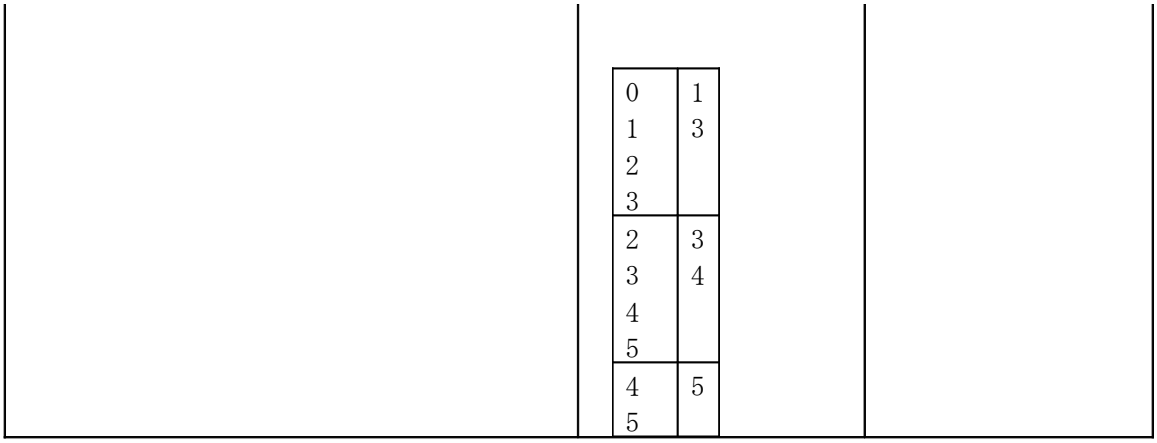
_24
+: .: *: 4

_12</td><td>-:@(*:-*:&+:)4

_24
-:@(+:-+:&*:)4

_12</td></tr>
</table>

(];.0)は　全ての軸を　逆順に	];.0 i.2 2 3 2 1 0	|.″1|. i.2 2 3 2 1 0
	2 2];.0 i.3 2 0 1 2 3	2 1];.0 i.3 2 0 2
(<;.1)は先頭　(<;.2)は末尾 フレット(0 1)の表れた　位置で切る	<;.1 (3 2$i.4) 0 | 0 1 1 2 3	<;.2 (3 2$i.4) 0 | 2 3 1 | 0 1
(<;.1)や(<;.2)の　両側形は 先頭や末尾から　1が現れると　区切る	1 0 1 <;.1 i.3 2 0 | 4 5 1 2 3	0 1 0 <;.2 i.3 2 0 1 2 3
(<;._1)は先頭　(<;._2)は末尾 フレットを除いて　区切りを入れる	<;._1 (3 2$i.4) 2 | 3 |	<;._2 (3 2$i.4) | 2 3
(<;._1)や(<;._2)の　両側形は (<;.1)や(<;.2)の　先頭や末尾を 削除する	1 0 1 <;._1 i.3 2 2 | 3 |	0 1 0 <;._2 i.3 2 0 1
();.3)の　片側形は	<;.3 i.3 0 1 | 1 2 | 2 2	3 <;.3 i.3 0 1 | 1 2 | 2 2
	2 2 <;.3 i.3 2	1 2 <;.3 i.3 2 0 1 | 1 2 3 | 2 4 5 | 5


0
1
2
3
1
3
2
3
4
5
3
4
4
5
5

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 (アトムに対しては“ボックス(<)”と同じ)	0 1 ; 2 3 ┌──┬──┐ │0 1│2 3│ └──┴──┘ ┌──┐ │3 │ └──┘ {3	{ 0 1 ; 2 3 ┌──┬──┐ │0 2│0 3│ ├──┼──┤ │1 2│1 3│ └──┴──┘
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	1 { 1 2 3 2	0 { i.2 3 0 1 2
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	1 1 0 } i.2 3 3 4 2	
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)	3 4(0 1)}i.3 3 4 2	
ヘッド({.}で先頭　テール({:)で末尾 　要素取り出す　片側動詞	{. K=:1+i.3 1	{: K 3

【副詞は後から　役割果す　動詞の活躍　拡大する】

両側動詞に　ウエーブ(~)つけりゃ 右引数を　左にも	*/~ a=:1+i.3 1 2 3 2 4 6 3 6 9		a */ a 1 2 3 2 4 6 3 6 9
左右に数値が　ある場合には 　左右の引数を　交換する	5 %~ i.5 0 0.2 0.4 0.6 0.8		(i.5) % 5 0 0.2 0.4 0.6 0.8
ウエーブ･ピリ(~.nub)の　片側形は 　重複要素を　排除する ウエーブ･コロン(~:)の　片側形は 　ダブリの位置に　“0”を与える	a =: 1 2 1 3 3 2 1 ~. a 1 2 3	~:a 1 1 0 1 0 0 0	(~:a) # a 1 2 3
ウエーブ･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)	(1 2)~:2 1 1 1 (1 2)~:1 2 0 0 (1 2)~:2 1 3 length error		(1 2)-:2 1 0 (1 2)-:1 2 1 (1 2)-:2 1 3 0

データに　プラス・スラッシュ(+/) 合計算	+/ 3 1 2 6
スラッシュ・ピリ(/.) 右引数の テーブルを 逐一斜めの　対角要素(oblique)	</.i.3 4 0 | 1 4 | 2 5 8 | 3 6 9 | 7 10 | 11
イコール(=)記号の　片側形は　分類・集計に　チョー便利	a=:1 2 3 1 3 2 1

]b=:= a 1 0 0 1 0 0 1 0 1 0 0 0 1 0 0 0 1 0 1 0 0	(~.a)=/a 1 0 0 1 0 0 1 0 1 0 0 0 1 0 0 0 1 0 1 0 0	+/"1 b 3 2 2	(~.a),:+/"1 b 1 2 3 3 2 2

b <@# A=:'abcdefg' adg | bf | ce	a </. A adg | bf | ce

データを 昇順にする グレードアップ(/:~) グレードダウン(¥:~)は　降順に	/:~ 3 1 2 1 2 3	¥:~ 3 1 2 3 2 1

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 1 2 3]A=:1 2 ; 3 4 1 2 | 3 4	> B 1 2 3 ; A 1 2 3 4

小にチョン(<.)　両側形は　最小値 大にチョン(>.)なら　最大値	3 <. 3.14 3	3 >. 3.14 3.14
小にピリ(<.)　片側形は　切り捨てる 大にピリ(>.)なら　切り上げる	<. 3.14 3	>. 3.14 4
数値から　1をマイナス　小にコロン(<:) 大にコロン(>:)は　1を加える	<: 3 3.14 2 2.14	>: 3 3.14 4 4.14

*: d.1 +:	^. d.1 %	(^.**:)d.1 (% * *:) + ^. * +:	(1:+_3&*+*:)d.1 _3 2x&p.