

Box < _ 0 0 Less Than

<code><y</code> is an atomic encoding of <code>y</code> , as discussed in Section II A. The result has rank 0, and is decoded by <code>></code> .	<code>x<y</code> is 1 if <code>x</code> is tolerantly less than <code>y</code> . See Equal (<code>=</code>) for a definition of tolerance. <code><! .t</code> uses tolerance <code>t</code> .
---	---

Boxing is also effected by verbs such as Link (`;`) and Word Formation (`;`):

```
(<'abc'),(<5 7),(<i.2 3)
+---+---+---+
|abc|5 7|0 1 2|
|   |   |3 4 5|
+---+---+---+
;: 'Now is the time'
+---+---+---+
|Now|is|the|time|
+---+---+---+
] a=: 2;3 5;7 11 13
+---+---+---+
|2|3 5|7 11 13|
+---+---+---+
>a
2 0 0
3 5 0
7 11 13
```

Cut (`;`) with `<` has several uses (chosen by the right argument); the phrase `<@v` avoids the padding (and some domain errors) that may result from applying `v` alone:

```
<:_1 '/i sing/of olaf/'
+---+---+---+
|i sing|of olaf|
+---+---+---+
i."(0) 2 3 4
0 1 0 0
0 1 2 0
0 1 2 3
<@i."(0) 2 3 4
+---+---+---+
|0 1|0 1 2|0 1 2 3|
+---+---+---+
```

If `y` is a high-rank array, `<"_1 y` or `<"_2 y` often gives a more intelligible display than `y` itself. The display of a boxed array would normally be corrupted by control

characters (such as carriage returns and linefeeds) occurring therein; in the display such characters are replaced by spaces. For example, try `< 8 32 $ a.`

Floor < . 0 0 0 Lesser Of (Min)

<p><code><.y</code> gives the floor of <code>y</code>, that is, the largest integer less than or equal to <code>y</code>. Thus:</p> <pre><. 4.6 4 _4 _4.6 4 4 _4 _5</pre> <p>The implied comparison with integers is tolerant, as discussed under Equal (=), and is controlled by <code><.!.t.</code> See below for complex arguments.</p>	<p><code>x<.y</code> is the lesser of <code>x</code> and <code>y</code>. For example:</p> <pre>3 <. 4 _4 3 _4</pre> <pre><./7 8 5 9 2 2 <./\7 8 5 9 2 7 7 5 5 2</pre>
---	---

For a complex argument, the definition of `<.` is modelled by:

```
floor=: j./@(ip+(c2>c1),c1+:c2)
'`c1 c2 fp ip'=: (1:>+/@fp)`(>:/@fp)`(+.-ip)`(<.@+.)
```

As developed by McDonnell [10], this function has the following properties:

- Convexity: If $(\<.z1)=(\<.z2)$ and $z3$ lies on the line between $z1$ to $z2$, then $(\<z3)=(\<z1)$.
- Translatability: If $z4$ is a Gaussian integer, then $(z4+\<.z5)=(\<.z4+z5)$.
- Compatibility: $(\<.x j.0)=((\<.x)j.0)$ and $(\<.0 j.x)=(0 j.(\<.x))$

The function `<.` can be viewed as a tiling by rectangles of unit area, all arguments within a rectangle sharing the same floor. One rectangle has vertices at `1j0` and `0j1`, with the other side passing through the origin. Rectangles along successive diagonals are displaced by one-half the length.

The phrase `j./@ip` “floors” the individual parts of a complex argument. Moreover, the floor `<.y` is equivalent to `->.-y`. In other words, it is the dual of ceiling with respect to (that is, under) arithmetic negation: `<. ↔ >.&.-` and `>. ↔ <.&.-`. Thus:

```
(>.&.- ; <.) 4.6 4 _4 _4.6
+-----+-----+
|4 4 _4 _5|4 4 _4 _5|
+-----+-----+
```

Continued