25 Key Runes

`=/  skin  hoon` `hoon` `=/  a  1  <rest-of-hoon>`
define a variable of value **`hoon`** with name **`skin`**

`|=  spec  hoon` `|=  a=@rs  (add:rs a .1.0)`
produce a gate (one-armed core with battery **`hoon`** and sample **`spec`**)

`|-  hoon`
produce a trap (one-armed core with battery **`hoon`** and arm `$`) and kick it

`|_  spec` `alas  (map term tome)  --`
produce a door (a generalized gate, a core with a sample but many arms) which accepts sample **`spec`**

`|%` `(unit term)  (map term tome)  --`
produce a generic core (cell of `[battery payload]`)

`%-  hoon  hoon` `(add 1 1)`
call a gate **`hoon`** (one-armed core) with sample **`hoon`**

`%~  wing` `hoon  hoon` `` `@t`~(x ne 0xf) ``
evaluate an arm **`wing`** in a door (resolve the wing as a gate and call it)

`%=  wing` `(list (pair wing hoon)` `$(count +(count))`
resolve a wing **`wing`** with changes; frequently used with `$` to iterate a trap forward as a loop

`++  term  hoon`
produce a normal arm with name **`term`** and content **`hoon`**

`+$  term  spec`
produce a structure arm (type definition) with name **`term`** and mold **`spec`**

`$=  skin  spec` `foo=baz`
assign a name **`skin`** to a **`hoon`** ("wrap a face around a **`hoon`**")

`$_  hoon` `_foo`
normalize structure to example

`^-  spec  hoon` `^-  @ud  a`
typecast explicitly

`^+  hoon  hoon` `^+  .1  a`
typecast by example

`?:  hoon  hoon  hoon` `?:((gth:rs a .0) a (sub:rs .0 a))`
branch conditionally on test; if **`hoon`** then **`hoon`** else **`hoon`**

`?.  hoon  hoon  hoon` `?.((gth:rs a .0) (sub:rs .0 a) a)`
reversed conditionality; branch conditionally on test; if **`hoon`** then **`hoon`** else **`hoon`**

`?=  spec  wing`
test pattern match, whether **`wing`** is type **`spec`**

`?>  hoon  hoon`
assert positively that **`hoon`** and **`hoon`** match

`.^  spec  hoon` `.^(arch %cy %)`
scry into vane namespace per instruction **`hoon`** and apply mold **`spec`** to the result

`:-  hoon  hoon` `:-  %say  foo`
construct a cell (2-tuple); see also *n*-tuple constructor **`:*`**

`;<  mold  hoon  hoon  hoon`
monadic bind, defer completion of **`hoon`** until after **`hoon`** has resolved; **`hoon`** is an adapter

`/+  path` `/+  generators`
imports a file from `lib/`**`path`** (`*` pinned with no face, `=` with specified face)

`~&  hoon` `~&  [foo <bar> <baz>]`
side effect: output value of **`hoon`** to `stderr`

`!>  hoon`
wrap a noun **`hoon`** in its type; frequently used as the "type spear" `-:!>`

`!!`
crash (no children); useful for stubbing out branches in development