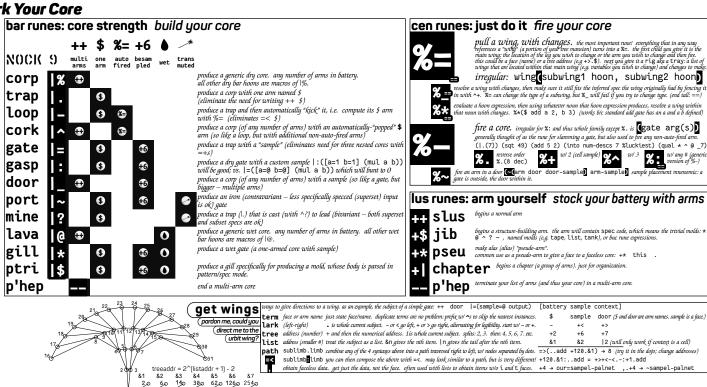
Work Your Core



Catch Your Typos you gotta check yoself before you wreck yoself

buc runes: shape sifters

hoon's type system defines sets. the point of it is for you, the programmer, to declare your intentions of what you're trying to and specifically what kind of data you're trying to generate and pass around. this helps you because then if you make some mistal or type in your program, that will many times alter the upp of the data and the type system will calc that and fall to compile these but runes are the mod huiders, what is a mold, you ask? a mod is just a specific kind of gate, and a very simple one, takes in as a sumple on you must all configure to know the system of the programmer when the system is a which it is specific, it output is that some houre (people of S and S-1) passing spec generally means the noun is the right shape of tree topologically, though sometimes specific atoms can be required at opening modes, triant of a shape-sorting toy with holes of different shapes and blocks to try to pass through them—a modd is like that, a few basic modds funce been shortened to a single character:

* noun ^ cell @ atom ? flag (an atom that's 0 or 1) ~ null (atom that's 0). also, ! represents not-a



%spam totally discards the sample it's given and just produces the noun produced by the \$'s child hoon no matter what.

[a=@ b= \star c= $^$ d= \star 27] create a cell mold (to pass through, the noun mu be a cell). the required shape of the cell further defined by specs within the \star :

?(%good %evil) the noun may fit any of the specs in the list (an "or" union use only for a union of atoms. more complex nouns, use \$@, $\* , and/or \$%.

\$\(\epsilon\) a pass-through, allowing all atoms and cells through the mold. \$\(\epsilon\) (null spec) allows only a particular kind of atom (a null) and cell (a spec).

\$%([%number @] [%fizz tape] [%buzz tape]) a whole list of cell molds, with the head a constant (cold atom), to be selected based on the head.

\$>(? [**]) \rightarrow the head must be a flag (0 or 1) require the head of a cell mold to the a mold. e.g. could sect one mold from a % fits. using above examples, \$>(Xnumber bccn-exmpl) \rightarrow [Xnumber e] \rightarrow \$\(\frac{\pi}{\pi}\) Xnumber [e] \rightarrow \$\(\frac{\pi}{\pi}\) Xnumber [e]

takes a spec and further restricts it.
\$ (@ (curr lth 10)) creates a mold for atoms less than 10. \$|

[a-\$%-mold-combining-old-and-new-versions gate-normalizing-to-new] upgrades a structure from an old version to a new one. $\$8(\$\%([\$v1 \land] [\$v2 \land]) = (x=[a=0 b=^] [\$v2 b.x]))$

take any spec and define a custom default value for it. \$~(%88 \$=(face @)) is face=@ but with a default of 88

input a noun fitting one mold to get the bunt of another as output. \$-(@ ^) takes an atom, outputs a cell (it will just bunt)
\$-([@ @] ^) will take a cell of two atoms and output a cell bunt. \$-(@ %0k) returns %0k when fed an atom

ket runes: cast away generate generalities

when you cast, you are always throwing away type info, never adding, so you can only cast from more specific to more general. the most specific info is just exactly what the particular noun is, which the compiler aready knows, obviously; so this should make sense, so the terminology "cast to x" means "cast away all info except x".



cast to a spec by simply explicitly writing the spec to which you wish to cast the noun. assign a face to a noun.

Fence ast to "whatever this is." cast by writing a noun and letting the system infer its spec. $+(>?< \%fnz) \rightarrow 'fnz' \rightarrow (\sim fyn `@`255) \rightarrow \sim fes$ $\$mul \rightarrow 1 \quad \$mul : rs \rightarrow 0 \ (a poorly-chosen bunt)$ [gate hoon] cast based on the noun produced by passing 4 to p — the hoon (any expression) to the gate (e.g. Limo which makes it a list).

transmute to zinc transmute to iron transmute to lead constant

NOCIC

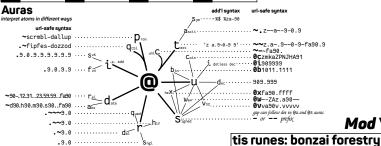
• IS INITIAL SUBJECT, A NOUN (A TREE OF NUMBERS)

P, Q, AND R ARE NOUNS

[] IS A CELL (A NON-SINGLE-NODE THEE)

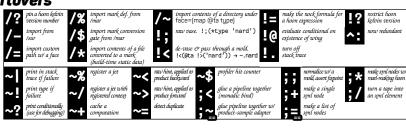
P%(Q) IS KNOCK THE NOCK FORMULA Q WITH THE SUB.

Nodock P and noder o, then ender the α -product bith the p-product as subject 3 is P a Cell? $^{9}(3|r) \rightarrow x$ or x 4 and One to P (Terrenewit'), $^{9}(4|r) \rightarrow r$ plus one 5 is P insertical to Q2: $^{9}(5|r) = x$ or x 6 if Thex-Elbest $^{9}(6|r) = \alpha$ if $r \rightarrow r$ produces x, r if x 7 P Marcs a Net Subject to Q (Gain a) $r \rightarrow r$ or r 8 if $r \rightarrow r$ produces x, r if x 9 P Marcs a Net Subject to Q (Gain a) $r \rightarrow r$ (Gibest r P Rest node r , then with that as the nee subject indick θ P Ado (as Head) to Subject for $0 \rightarrow (9/6) P (\theta) P (\theta)$ 9 Boild a Code & Fine an And $r \ll 10 P (\theta) P (\theta) P (\theta)$ 9 Boild a Code & Fine an And $r \ll 10 P (\theta) P (\theta) P (\theta)$ p a tree anomers, α is a time function of α 10 Replace Part of Subject $r \ll 10 P (\theta) P (\theta)$



Mod Your Tree

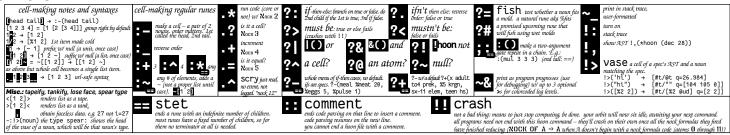




prune off, graft on, and swap out. a couple of oft-used fas runes, too 7 replaces the subject, 8 prepends to subject, 10 changes the subject. NOCK 7 NOCK 8 NOCK 10 compose 2 hoons. erases old subject! add a new noun to the subject compose 2 hoon: in reverse order =+ in reverse =< (sub +2 1) [(mul 7 4) .] → 27 =/ & = . pin a noun & change leg

pin add be-faced noun to subject =/ in reverse order =? change leg if a statement is true [cet:yo .] [(sub - 124) .] [(div - 100) .] [(div - 52) .] [(mul - 4) .] [(sub - 1) .] add a typed bunt to the subject alas define alias define a bridge i /+ import from /lib directory егроѕе а п /*

Do Tricks



.9.0

not a bad thing: means to just stop computing, be done, your urbit will now sit idle, awaiting your next command.
all programs need not end with this hoon command – they'll crash on their own once all the nock formulas they fired have finished reducing (NOCK OF A → A when A doesn't begin with a nock formula code (atoms 0 through 11))