|  |  |
| --- | --- |
| TEA PRIMITIVE | SEMANTICS |
| O: | |  |  | | --- | --- | | NAME | Order | | PURPOSE | Order, or rather, Sort things | | SYNTAX  & SEMANTICS | |  | | --- | | o: | | Return the AI sorted alphabetically by words it contains. | | o:VALUE | | Same as o:, but using VALUE instead of AI | | o!: | | Return the AI sorted alphabetically by characters it contains. | | o!:VALUE | | Same as o!:, but using VALUE instead of AI | | o\*:vNAME  o\*!:vNAME | | Same as o:VALUE and o!:VALUE, but using the string stored in the vault with the name vNAME as the VALUE | |  | | NOTES | Being able to order things at will is such a formidable power, it can’t be underestimated that it is built into the TEA language as a primitive. Ordering makes design possible, prevents or limits chaos and randomness, and allows structure to be created or imposed on things. For TEA, the O: command space offers several utilities for performing ordering operations on strings directly in the program, user-provided or those stored in memory.  A basic illustration of this power can be demonstrated by the following example TEA program that returns the alphabetically sorted initials of a person’s name:  # following URL is hypothetical, but expected to  # return the full, undecorated name of someone  W!: <https://mit.edu/vc/name.txt>  # (=”Terrance L. Epstein Von Zalta”)  D!:^.: [a-zA-Z]  # (=”T L E V Z”)  G:  # (=”TLEVZ”)  O!:  # (=”ELTVZ”)  Another example:  i!:{mice ice best acts zap}|o:  #should sort input by lexical order of words and then return…  #=acts best ice mice zap  i!:{mice ice best acts zap}|o!:  #should sort input by lexical order of letters, and thus return:  #= aabccceeeiimpssttz |  | |