|  |  |
| --- | --- |
| TEA PRIMITIVE | SEMANTICS |
| I: | |  |  | | --- | --- | | NAME | Interact | | PURPOSE | Explicitly set the AI | | SYNTAX  & SEMANTICS | |  | | --- | | i: | | Using the current AI as the prompt, prompt for and set whatever is the user-provided input---at runtime, as AI | | i:VALUE | | Only if AI is currently empty or unset, then set it to VALUE | | i!: | | Unconditionally set the AI to the EMPTY STRING | | i!:VALUE | | Unconditionally set the AI to the provided VALUE | | i\*: | | Using the current AI as the visual prompt (to be rendered in a separate PLAIN TEXT/HTML rendering window with the input prompt displayed at the bottom), prompt for input from the user at runtime. Input returned set as the AI | | i\*:vPROMPT | | With vPROMPT, same as i\*:, but using text sourced from the vault vPROMPT or just the literal text vPROMPT if the vault doesn’t exist. | | i\*!:  i\*!:vPROMPT | | Without parameter, as i\*:, but with no prompt message except just the visual prompt window. User input returned as AI. With vPROMPT, same as i\*:vPROMPT | |  | | NOTES | It is important to note that under standard TEA environments, it is possible for a TEA program to be invoked with a user or externally provided Active Input. In those cases, if the canonical form of **i:** instruction is used, with or without value, it doesn’t make any changes to AI unless the command is used at a moment in the program where AI is essentially either empty or unset.  For example, using the standard TTTT TEA operating environment on the command line [3], the following program if invoked thus  tttt -i "ABC" -c "i:{XYZ} | q:XYZ | x!":-OK  Shall return “ABC-OK” instead of “XYZ”. Otherwise  tttt -i "ABC" -c "i!"":{XYZ} | q:XYZ | x!"":-OK"  Which, despite the queer command line invocation syntax---this example was adapted from a test via the Linux/GNU Bash Shell [6]. The essential TEA program itself is actually:  i!:{XYZ} | q:XYZ | x!:-OK  So, that, or even  bash -H -c 'set +H; tttt -i TEST -c "i!:{XYZ} | q:XYZ | x!:-OK"'  Should return “XYZ” because of the forced setting of the AI using **i!:.** Because of its power to set AI unconditionally, i: is among the most important primitive instructions in TEA.  Also, and importantly so, the unparameterized form of the Input command is the only way in TEA, to prompt for and store runtime user-input. This also becomes the only way in TEA, to display arbitrary values/strings, to the user, at runtime, before the program terminates using the idiom: i:{PROMPT }|i: or i!:{PROMPT }|i:  Thus for example, to write the **Minimum Basic Output**  **Program** (LOCMBOP)[7] in TEA, one just writes:  i:Hello World  Which just prints “Hello World” and returns. This basic program demonstrates the simplest way to write a TEA program that just displays a value and does nothing else.  However, in cases where one needs to display some value that depends on user-provided input, especially at runtime; for example, a modified version of the above Hello World program, that instead greets the user with their provided name, would be attained using the user-prompting version of the I-command thus:  i:{What is your name please? }|i:|x:{Hello }  Shall display the prompt “What is your name please?”, block and wait for user-input, and assuming user enters “Joseph”, shall then print the greeting “Hello Joseph” as shown in screenshot below  **EXPERIMENTS|< 13:20:05 $>\*** tttt -c "i:{What is your name please? }|i:|x:{Hello }"  What is your name please? Joseph  Hello Joseph  **EXPERIMENTS|< 13:20:29 $>\***  For TEA, then, the **Minimum Basic Input Program** (LOCMBIP) [7] becomes the following program:  i:{What is your name please? }|i:  Such as we see in the screenshot below:  **EXPERIMENTS|< 13:27:52 $>\*** tttt -c "i:{What is your name please? }|i:"  What is your name please? John Doe  John Doe  **EXPERIMENTS|< 13:28:05 $>\***  This feature then allows us to write arbitrarily complex programs involving displaying things to the user, sometimes prompting for, then displaying dynamic outputs based on user-provided values, at runtime.  Finally, note that I: is not only the ONLY way to block a program and prompt for input from the user, but is also the only means to make a TEA process pause, while it optionally displays some [useful] message (whatever is current AI). It makes TEA programs INTERACTIVE. A great example of I: in action is the ZHA[9] q-AGI personal assistant that ships with standard TEA (tttt) package on Linux/Unix. |  | |