Command list

|  |  |  |
| --- | --- | --- |
| Command | Name | Usage/Description |
|  | Space | Separates commands from each other. [. > .] |
| $ | Output | [$ …] will print 50 if the variable […] is set to 50 |
| . | Variable | Represents an instance of a variable. [..] or […] are different variables |
| @. | Constant | Represents an instance of a constant. [@..] or [@…] are different constants |
| : | Assignment | Setting a variable equals to something. [. : 50] |
| :: | Comparison |  |
| > | Addition | [. > 5] equals 6 if [.] was equal to 1 |
| < | Subtraction | [. < 5] equals -4 if [.] was equal to 1 |
| >> | Multiplication | [. >> 5] equals 5 if [.] was equal to 1 |
| << | Division | [. << 5] equals 0.2 if [.] was equal to 1 |
| {} . | Repetition | {[instruction to repeat]} [..] would repeat the instruction 3 times if variable [..] was equal to 3 |
| & | Until | Used in conjunction with repetition. [$ … ! **{**… : … > … . : . > ….**} & . \:: ..** | . : 1 .. : 5 … : 1 …. : 1] will print 16 (See example below) |
| () | Previous value of variable | [. : 120 . : 50 $ (.)] would print 120. No space between command and variable. |
| | | ~~Assuming~~ Where/Given | [$ . | . : 1] print variable assuming the future value of variable is 1 |
| ! | After | [$ . > .. ! | . : 1 .. : 2] would print 3. |
| ~ | If - condition checking | [$ … ! … : . ~ . :: .. | … : 0 . : 1 .. : 2] would print 0 |
| , | Else Condition | [$ … ! … : . , … : .. ~ . :: .. | … : 0 . : 1 .. : 2] would print 2 |
| [] | Containment | Functional start and finish |
| \ | Greater than | [$ … ! … : . ~ . \ .. - … : .. | . : 1 .. : 2] would print 2 |
| / | Smaller than | [$ … ! … : . ~ . / .. - … : .. | . : 1 .. : 2] would print 1 |
| \:: | Greater than or equals to | [$ … ! … : . ~ . \:: .. - … : .. | . : 1 .. : 2] would print 2 |
| /:: | Smaller than or equals to | [$ … ! … : . ~ . /:: .. - … : .. | . : 1 .. : 2] would print 1 |
| ` | Remainder | [$ … ! … : . ` .. | . : 4 .. : 2] would print 0,  [$ … ! … : . ` .. | . : 5 .. : 2] would print 5 |
| \_ | Nothing |  |
| .\* \* | Embedded function | [.] Represents the function unique name. Function text between \* and \*. Call function by using /.\*  Functions can have multiple embedded functions, but each embedded function is only accessible from the parent container. Function text is only executed when the function is called by the parent container. |
| || | Parameter list | Call function by using /.\*[..] or /.\*[.. …] to include parameters. Parent variables cannot be accessed from inside an embedded function and as such, parameter name can be the same in the parent and embedded function, but they would have different values.  [$. ! . : .. > /.\*[..] | . : 0 .. : 2 .\*[$ … ! … : … >> … | … : ..||..]\*] would return 6 |

Example

[$ … ! **{**… : … > … . : . > ….**} & . \:: ..** | . : 1 .. : 5 … : 1 …. : 1]

Start with assignment [. : 1 .. : 5 … : 1 …. : 1]

A = 1

B = 5

C = 1

D = 1

Next perform function {{… : … > … . : . > ….} & . \:: .. }

Check how many times it needs to be performed [& . \:: ..]

Until A >= B -> until 1 >= 5

Execute loop content [… : … > … . : . > ….]

C = C +C and A = A + D

1. 1+1 = 2 and 1+1=2 -> C = 2, A = 2
2. 2+2=4 and 2+1=3 -> C = 4, A = 3
3. 4+4=8 and 3+1=4 -> C = 8, A = 4
4. 8+8=16 and 4+1 = 5 -> C = 16, A = 5

Next execute [$ …]

Print C

Result -> 16

Questions: What would the output of each of the following programs be?

Question 1

[$ . ! | . : 1]

Question 2

[$ . ! . : . < .. | . : 2 .. : 1]

Question 3

[$ . ! { . : . > . } .. | . : 1 .. : 3]

Question 4

[$ . ! { . : (.) > . } .. | . : 0 . : 1 .. : 10]

Question 5

[$ . ! { . : (.) >> . } .. | . : 1 . : 2 .. : 4]

Question 6

[$ …. ! { …. : …. > .. .. : .. > . } & .. \:: … | . : 1 .. : 0 … : 5 …. : 0 ]

Question 7

[$ . ! [. : ... , [. : . , . : .. ~ .. << .. \:: . ] ~ .. << .. :: .] | . : 1 .. : 2 ... : 0 ]

Question 8

[$ . ! . : @. {[[. : @.. … : .. ] , … : … > @. ] ~ .. ` … :: @.. } & … /:: .. | . : \_ .. : 5 … : 2 @. : 1 @.. : 0]

Question 9

[$. ! . : /.\*[. .. …] | . : 0 .. : 1 ... : 12 \*[$.... ! | .... : . > .. /.\*[.. .... ...] ~ .... /... |.... : \_||. .. … ]\*]

Question 10

Build a program using the syntax above to calculate the sum of all even numbers from 1 to 50

Question 11

Extend the program created in question 6 to half every second even number while counting