Lua 5.3 for FlyingSticks

A Short Reference by Graham Henstridge (Updated 20/6/2019)

Lua Core Language

Reserved Words

and	**break**	**do**	**else**	**elseif**	**end**	**false**	**for**
function		**goto**	**if**	**in**	**local**	**nil**	**not**
or	**repeat**	**return**	**then**	**true**	**until**	**while**	

_A... A system variable, where **A** any uppercase letter.

Other Reserved Strings

+	**-**	*****	**/**	**%**	**^**	**#**	**&**	**~**	**|**	**<<**	**>>**	**//**
==	**~=**	**<=**	**>=**	**<**	**>**	**=**	**(**	**)**	**{**	**}**		
[	**]**	**::**	**;**	**:**	**,**	**.**	**..**	**...**				

Identifiers

Any string of letters, digits and underscores not starting with a digit and not a reserved word. Identifiers starting with underscore and uppercase letter are reserved.

Comments

--	Comment to end of line.
--[[... **]]**	Multi-line comment (commonly --[[to --]])
#!	At start of first line for Linux executable.

Strings and Escape Sequences

' ' " " [[ ]] [=[ ]=]
string delimiters; **[[ ]]** can be multi-line, escape sequences ignored. If **[=[ ]=]** number of **=**'s must balance.

\a - bell	**\b** - backspace	**\f** - form feed
\n - newline	**\r** - return	**\t** - tab
\v - vert. tab	\\ - backslash	**\\"** - double quote
\\' - single quote	**\\[** - square bracket	**\\]** - square bracket

* - skips following white space span
\ddd - character represented 3 digit decimal number ddd
\xFF - character represented by two hexadecimal digits **FF**
\u{FF} - a UTF-8 character represented by 2 to 8 hex digits **FF**
\0 - zero

Types

Type belongs to the value, NOT the variable:

boolean	**nil** and **false** count as **false**, all other **true i**ncluding **0** and null string. Use **type(*x*)** to discover type of ***x.***
number	64 bit IEEE floating point
string	Can include zero, internally hashed.
table	Index by numbers, strings
function	Can return multiple values
thread	A cooperative coroutine.
userdata	C pointer to a C object. Can be assigned a metatable to allow use like a table or function
nil	A special value meaning "nothing".

Numerical Constants

Valid formats are:

3 3.0 3.1416 314.16e-2 0.31416E1
0xff 0x0.1E 0xA23p-4 0X1.921FB54442D18P+1

Operators in Precedence Order

^	(right-associative, math lib required)				
not	**#** (length)		**–** (unary negative)		~
*****	**/**	**//**	**%**		
+	**–**				
..	(string concatenation, right-associative)				
<<	**>>**				
&					
~					
|					
<	**>**	**<=**	**>=**	**~=**	**==**
and	(stops on **false** or **nil,** returns last evaluated value)				
or	(stops on **true** (not **false** or **nil**), returns last evaluated value)				

Assignment and Coercion Examples

a = 5	Simple assignment.
a = "hi"	Variables are not typed, they can hold different types.
a, b, c = 1, 2, 3	Multiple assignment.
a, b = b, a	Swap values, because right side values evaluated before assignment.
a, b = 4, 5, 6	Too many values, **6** is discarded.
a, b = "there"	Too few values, **nil** is assigned to **b**.
a = nil	**a**'s prior value will be garbage collected if unreferenced elsewhere.
a = #b	Size of **b.** If table, first index followed by **nil**.
a = z	If z is not defined **a** = **nil**.
a = "3" + "2"	Strings converted to numbers: **a = 5**.
a = 3 .. 2	Numbers are converted to strings: **a = "32"**.

Conditional Expression Results

False: **false** and **nil** values only
True: anything not false, including 0 and empty strings

Relational and Boolean Examples

"abc" < "abe"	True: based first different character
"ab" < "abc"	True: missing character is less than any

Statements, Scope, Blocks and Chunks

By default all variables have global scope from first use.

local	Reduces scope from point of definition to end of block. **local *var_name*** initialized to **nil**. Locals significantly faster to access

block	Is the body of a control structure, body of a function or a chunk.
chunk	A file or string of executable script.
;	statement separator
;;	empty statement

Control Structures

In following exp, var and name have local scope
if exp **then** block {**elseif** exp **then** block} [**else** block] **end**
do block **end** (simply a means of forcing local scope)
while exp **do** block **end**
repeat block **until** exp
for var **=** from_exp, to_exp [, step_exp] **do** block **end**
for var(s) **in** iterator **do** block **end** (var(s) local to loop)
break exits loop
return exits loop, but must be last statement in block

goto name	transfers control to label name.
::name**::**	label visible in its entire block but must not be in scope of a local variable.

Table Constructors

t = {}	New empty table assigned to **t**.
t = {"yes", "no"}	A array, **t[1]** = **yes**, **t[2]** = **no**.
t = {[2] = "no", [1] = "yes"}	Same as line above.
t = {[-900] = 3, [900] = 4}	Sparse array, two elements.
t = {x=5, y=10}	Hash table **t["x"]**, **t["y"]**, **t.x**, **t.y**
t = {x=5, y=10; "yes", "no"}	Mixed fields: **t.x**, **t.y**, **t[1]**, **t[2]**.
t = {"choice", {"yes", "no"}}	Nested table.

See **table.insert()** etc. below for additional info.
Table length **#t** only returns sensible results for tables with sequence {1..n} of non-nil values.

Function Definition

Functions can return multiple results.
function name **(** ***args*** **)** body [**return** values] **end**
Global function.
local function name **(** ***args*** **)** body [**return** values] **end**
Function local to chunk.
f = function (***args*** **)** body [**return** values] **end**
Anonymous function assigned to variable **f**
function (***...*** **)** body [**return** values] **end**
(***...***) indicates variable args and {***...***} places them in a table accessed as … .
function t.name **(** ***args*** **)** body [**return** values] **end**
Shortcut for t.name = **function** [...]
function obj:name **(** ***args*** **)** body [**return** values] **end**
Object function getting extra ***arg*** **self**.

Function Call

f (***args*** **)**	Simple call, returning zero or more values.
f ***arg***	Calling with a single string or table argument
t.f (*args*)	Calling function stored in field **f** of table **t**.
t:f (***args***)	Short for **t.f (t, *args*).**
***arg*:f**	Short for **f (** ***arg*** **).**

Metatable Operations

Base library required. Metatable operations allow redefining and adding of new table behaviours.
setmetatable (***t, mt*** **)**
Sets mt as metatable for ***t***, unless ***t***'s metatable has a **__metatable** field. Returns ***t***
getmetatable (t)
Returns **__metatable** field of ***t***'s metatable, or ***t***'s metatable, or **nil**.
rawget (***t, i*** **)**
Gets ***t***[***i***] of a table without invoking metamethods.
rawset (***t, i, v*** **)**
Sets ***t***[***i***] = ***v*** on a table without invoking metamethods.
rawequal (***t1, t2*** **)**
Returns boolean (***t1*** == ***t2***) without invoking metamethods.

Metatable fields for tables and userdata

__add	Sets handler **h (*a*, *b*)** for '**+**'.
__sub	Sets handler **h (*a*, *b*)** for binary '**-**'.
__mul	Sets handler **h (*a*, *b*)** for '*****'.
__div	Sets handler **h (*a*, *b*)** for '**/**'.
__mod	Sets handler **h (*a*, *b*)** for '**%**'.
__pow	Sets handler **h (*a*, *b*)** for '**^**'.
__unm	Sets handler **h (*a*)** for unary '**-**'.
__concat	Sets handler **h (*a*, *b*)** for '**..**'.
__eq	Sets handler **h (*a*, *b*)** for '**==**', '**~=**'.
__lt	Sets handler **h (*a*, *b*)** for '**<**', '**>**' and '**<=**', '**>=**' if no **__le**
__le	Sets handler **h (*a*, *b*)** for '**<=**', '**>=**'.
__len	Sets handler **h (*a*, *b*)** for '**#**'
__index	Sets handler **h (*t*, *k*)** for non-existing field access.
__	**Index** iteration function.
__newindex	Sets handler **h (*t*, *k*)** for assignment to non-existing field.
__call	Sets handler **h (*f*, *...*)** for function call,using the object as a function.
__pairs	**Key** iteration function.
__tostring	Sets handler **h (*a*)** to convert to string, e.g. for **print ( )**.
__gc	Set finalizer **h (*ud*)** for userdata (can be set from the C side only).
__mode	Table mode: '**k**' = weak keys, '**v**' = weak values, '**kv**' = both.
__metatable	Set value returned by **getmetatable ( )**.

The Basic Library

The Basic Library provides many standard functions and does not require a prefix as with add-on libraries.

Environment and Global Variables

_G	Variable whose value = global environment.
_VERSION	Variable with interpreter's version.

Loading and Executing

dofile ([***filename***] **)**
Loads and executes the contents of ***filename*** [default: standard input]. Returns file's returned values.
load (***ld*** [, ***source*** [, ***mode***]] **)**
Loads a chunk using ***ld*** to get its pieces. If ***ld*** is a function it is called repeatedly to return strings (last = **nil**) that are concatenated to make a chunk. If ***ld*** a string that is the chunk. Returns compiled chunk as a global function or **nil** and error message. To call functions in the chunk, the chunk must be run. Optional chunk ***source*** for error messages and debugging (default is **"=(load)"**). Optional ***mode***: **"t"** for a text chunk, **"b"** for a precompiled chunk or **"bt"** for either (default)
loadin (***env, ...*** **)**
Similar to load () but sets ***env*** as the environment of the created function. ???
loadfile (***filename*** **)**
Loads contents of ***filename***, without executing. Returns compiled chunk as function, or **nil** and error message.
loadstring (***string*** [, ***name***] **)** depreciated for **load()**
Returns compiled ***string*** chunk as function, or **nil** and error message. Sets chunk ***name*** for debugging.
loadlib (***library***, ***func*** **)**
Links to dynamic ***library*** (.***so*** or .***dll***). . Returns function named ***func***, or **nil** and error message.
pcall (***function*** [, ***arg1***, ...] **)**
Calls ***function*** with arguments in protected mode; returns **true** and results or **false** and error message.
xpcall (***function***, ***handler*** [, ***arg1***, ...] **)**
As **pcall ()** but passes error ***handler***. Returns as **pcall ()** but with the result of ***handler ()*** as error message, (use **debug.traceback ()** for extended error info).

Simple Output and Error Feedback

print (***args*** **)**
Prints each of passed ***args*** to **stdout** using **tostring**.

error (***msg*** [, ***n***] **)**
Terminates the program or the last protected call (e.g. **pcall ()**) with error message ***msg*** quoting level ***n*** [default: 1, current function].

assert (***v*** [, ***msg***] **)**
Calls error (***msg***) if ***v*** is **nil** or false [default ***msg***: "**assertion failed!**"].

Information and Conversion

select (***i***, ***...*** **)**
For numeric index ***i***, returns the ***i***'th argument from the ***...*** argument list (for negative ***i***, from end of list). For ***i*** = string **"#"** (including quotes) returns total number of arguments including **nil**'s.

type (***x*** **)**
Returns type of ***x*** as string e.g. "**nil**", "**string**", "**number**".

tostring (***x*** **)**
Converts ***x*** to a string, using table's metatable's **__tostring** if available.

tonumber (***x*** [, ***b***] **)**
Converts string ***x*** representing a number in base ***b*** [2..36, default: 10] to a number, or **nil** if invalid; for base 10 accepts full format (e.g. "1.5e6").

Loop Iterators

Yieldable functions that assist in scanning a table's content

(***t*** **)**
Returns an iterator getting index, value pairs of array ***t*** in numeric order up to **#** length of ***t***.

pairs (***t*** **)**
Returns an iterator getting key, value pairs of table ***t*** in no particular order.

next (***t*** [, ***index***] **)**
Returns next index-value pair (**nil** when finished) from ***index*** (default **nil**, i.e. beginning) of table ***t***.

Garbage Collection

collectgarbage (***option*** [, ***v***] **)**
where ***option*** can be:

"**stop**"	Stops garbage collection.
"**restart**"	Restart garbage collection.
"**collect**"	Initiates a full garbage collection.
"**count**"	Returns total memory used.
"**step**"	Perform garbage collection step size ***v***, returns true if it finished a cycle.
"**setpause**"	Set **pause** (default 2) to ***v***/100. Larger values is less aggressive.
"**setstepmul**"	Sets **multiplier** (default 2) to ***v***/100. Controls speed of collection relative to memory allocation.
"**isrunning**"	Returns **true** if collector running

Coroutines

coroutine.create (***function*** **)**
Creates a new coroutine with ***function***, and returns it.

coroutine.resume (***coroutine***, ***args*** **)**
Starts or continues running ***coroutine***, passing ***args*** to it. Returns **true** (and possibly values) if ***coroutine*** calls **coroutine.yield ( )** or terminates, or returns **false** and error message.

coroutines.running ( )
Returns current running coroutine or **nil** if main thread.

coroutine.yield (***args*** **)**
Suspends execution of the calling coroutine (not from within C functions, metamethods or iterators), any ***args*** become extra return values of **coroutine.resume ( )**.

coroutine.status (***co*** **)**
Returns the status of coroutine ***co*** as a string: either "**running**", "**suspended**" or "**dead**".

coroutine.wrap (***function*** **)**
Creates coroutine with ***function*** as body and returns a function that acts as **coroutine.resume ( )** without first arg and first return value, propagating errors.

Modules and the Package Library

A package is a collection of modules. A module is library that defines a global name containing a table that contains everything the module makes available after being **require()**'d

require (***module*** **)**
Loads ***module*** and returns final value of **package.loaded[*module*]** or raises error. In order, checks if already loaded, for Lua module, for C library.

package.config
A sequence of lines describing some compile-time configurations for packages:
1. directory separator string. "\" for Windows, otherwise "/"
2. character that separates templates in a path. Default ';'.
3. string that marks substitution points in template. Default "?".
4. string in Windows path, replaced by the executable's directory. Default "!".
5. mark to ignore all before it when building the luaopen_ function name. Default "-".

package.path, package.cpath
Variable used by **require ( )** for a Lua or C loader. Set at startup to environment variables LUA_PATH or LUA_CPATH. (see Path Formats below).

package.loaded
Table of packages already loaded. Used by **require ( )**

package.loadlib (***library***, ***function*** **)**
Dynamically links to ***library***, which must include path. Looks for ***function*** and returns it,or **0** and error message.

package.preload
A table to store loaders for specific modules (see **require**).

package.searchpath (***name***, ***path*** **)**
Searches for ***name*** in the ***path*** and returns name of first readable file or **nil** and error message. The ***path*** is a string of semicolon separated templates, each of which a path that can contain "**?**"s that are replaced by name.

package.seeall (***module*** **)**
Sets a metatable for ***module*** with **_index** field referring to global environment.

Path Formats

A path is a sequence of path templates separated by semicolons. For each template, **require (** ***filename*** **)** will substitute each "?" by ***filename***, in which each dot replaced by a "directory separator" ("/" in Linux); then it will try to load the resulting file name. Example:
require (***dog.cat*** **)** with path **/usr/share/lua/?.lua;lua/?.lua** will attempt to load ***cat*.lua** from **/usr/share/lua/*dog*/** or **lua/*dog*/**

The Table Library

Tables as arrays (lists)

table.insert (***table***, [***i***,] ***v*** **)**
Inserts ***v*** at numerical index ***i*** [default: after the end] in ***table***, increments table size.

table.remove (***table*** [, ***i***] **)**
Removes element at numerical index ***i*** [default: last element] from ***table***, decrements table size, returns removed element.

table.sort (***table*** [, ***cf***] **)**
Sorts (in-place) elements from ***table***[1] to ***table***[#***t***], using compare function ***cf*** (***e1***, ***e2***) [default: '<']. May swap equals.

table.concat (***table*** [, ***string*** [, ***i*** [, ***j***]]] **)**
Returns a single string made by concatenating table elements ***table***[***i***] to ***table***[***j***] (default: ***i*** =1, ***j*** = table length)separated by ***string*** (default = ***nil***). Returns empty string if no given elements or ***i*** > ***j***

table.unpack (***t*** **)**
Returns ***t*** **[1]..*t* [n]** as separate values, where **n** = **#*t***.

Iterating on table contents

Use the **pairs** or iterators in a **for** loop. Example:
for *k*, *v* in pairs(*table*) do print (*k*, *v*) end
will print the key (***k***) and value (***v***) of all the ***table***'s content.

The Math Library

Basic operations

math.abs (***x*** **)**	Returns the absolute value of ***x***.
math.fmod (***x***, ***y*** **)**	Returns the remainder of ***x*** **/** ***y*** as a rounded-down integer, for ***y*** **~= 0**.
math.floor (***x*** **)**	Returns ***x*** rounded down to integer.
math.ceil (***x*** **)**	Returns ***x*** rounded up to the nearest integer.
math.min(***args*** **)**	Returns minimum value from ***args***.
math.max(***args*** **)**	Returns maximum value from ***args***.
math.huge	Returns largest represented number
math.modf (***x*** **)**	Returns integer AND fractional parts of ***x***

Exponential and logarithmic

math.sqrt (***x*** **)**	Returns square root of ***x***, for ***x*** >= 0.
math.exp (***x*** **)**	Returns **e** to the power of ***x***, i.e. **e^*x***.
math.log (***x*** [, ***b***] **)**	Returns logarithm base ***b*** (“**e**” (default) or **10**) of x, for x >= 0.
math.frexp (***x*** **)**	If ***x*** = m2^{e}, returns **m** (0, 0.5-1) and integer **e** .

Trigonometrical

math.deg (***a*** **)**	Converts angle ***a*** from radians to degrees.
math.rad (***a*** **)**	Converts angle ***a*** from degrees to radians.
math.pi	Constant containing the value of **Pi**.
math.sin (***a*** **)**	Sine of angle ***a*** in radians.
math.cos (***a*** **)**	Cosine of angle ***a*** in radians.
math.tan (***a*** **)**	Tangent of angle ***a*** in radians.
math.asin (***x*** **)**	Arc sine of ***x*** in radians, for ***x*** in [-1, 1].
math.acos (***x*** **)**	Arc cosine of ***x*** in radians, for ***x*** in [-1, 1].
math.atan (***x*** **)**	Arc tangent of ***x*** in radians.

Pseudo-random numbers

math.random ([***n*** [, ***m***] **)**
Pseudo-random number in range [0, 1], [1, ***n***] or [***n***, ***m***].

math.randomseed (***n*** **)**
Sets a seed ***n*** for random sequence. Same seed, same sequence.

The String Library

Basic operations

String indices start from 1. Negative indices from end of string so -1 is last element of string. String element values 0-255.

string.len (***string*** **)**
Returns length of ***string***, including embedded zeros.

string.sub (***string***, ***i*** [, ***j***] **)**
Returns substring of ***string*** from position ***i*** to ***j*** [default: -1 which is to end].

string.rep (***string***, ***n*** **)**
Returns a string of ***n*** concatenated copies of ***string***.

string.upper (***string*** **)**
Returns a copy of ***string*** converted to uppercase.

string.lower (***string*** **)**
Returns a copy of ***string*** converted to lowercase.

string.reverse (***string*** **)**
Returns a string that is the reverse of ***string***.

Character codes

string.byte (***string*** [, ***i***] **)**
Numeric ascii code of character at position ***i*** [default: **1**] in ***string***, or **nil** if invalid ***i***.

string.char (***args*** **)**
Returns a string from ascii codes passed as ***args***.

Formatting

string.format (***string*** [, ***args***] **)**
Returns a copy of ***string*** where formatting directives beginning with '**%**' are replaced by the value of [, ***args***]:
% [flags] [field_width] [.precision] type

Types

%d		Decimal integer.
%o		Octal integer.
%x	**%X**	Hexadecimal integer lowercase, uppercase.
%f		Floating-point in the form [-]nnnn.nnnn.
%e	**%E**	Floating-point in exp. form [-]n.nnnn e [+|-]nnn, uppercase if **%E**.
%g	**%G**	Floating-point as **%e** if exp. < -4 or >= precision, else as **%f**; uppercase if **%G**.
%c		Character having the code passed as integer.
%s		String with no embedded zeros.
%q		String between double quotes, with special characters (including control) escaped.
%%		The '**%**' character (escaped)

Flags

-	Left-justifies, default is right-justify.
+	Prepends sign (applies to numbers).
(space)	Prepends sign if negative, else space.
#	Adds "**0x**" before **%x**, force decimal point; for **%e**, **%f**, leaves trailing zeros for **%g**.

Field width and precision

n	Puts at least **n** characters, pad with blanks.
0n	Puts at least **n** characters, left-pad with zeros
.n	Use at least **n** digits for integers, rounds to **n** decimals for floating-point or no more than **n** chars. for strings.

Formatting examples

string.format ("dog: %d, %d",7,27)	dog: 7, 27
string.format ("<%5d>", 13)	`<   13>`
string.format ("<%-5d>", 13)	`<13   >`
string.format ("<%05d>", 13)	<00013>
string.format ("<%06.3d>", 13)	`<   013>`
string.format ("<%f>", math.pi)	<3.141593>
string.format ("<%e>", math.pi)	<3.141593e+00>
string.format ("<%.4f>", math.pi)	<3.1416>
string.format ("<%9.4f>", math.pi)	`<   3.1416>`
string.format ("<%c>", 64)	<@>
string.format ("<%6.4s>", "goodbye")	`<  good>`
string.format("%q",[[she said "hi"]])	"she said "hi""

Finding, replacing, iterating

string.find (***string***, ***pattern*** [, ***i*** [, ***d***]] **)**
Returns first and last position of ***pattern*** in ***string***, or **nil** if not found, starting search at position ***i*** [default: 1]; returns parenthesized 'captures' as extra results. If ***d*** is true, treat pattern as plain string. (see Patterns below)

string.gmatch (***string***, ***pattern*** **)**
Returns an iterator getting next occurrence of ***pattern*** (or its captures) in ***string*** as substring(s) matching the ***pattern***. (see Patterns below)

string.match (***string***, ***pattern*** **)**
Returns the first capture matching ***pattern*** (see Patterns below) or **nil** if not found.

string.gsub (***string***, ***pattern***, ***r*** [, ***n***] **)**
Returns copy of ***string*** with up to ***n*** [default: **1**] occurrences of ***pattern*** (or its captures) replaced by ***r*.** If ***r*** is a string (***r*** can include references to captures of form **%n**). If ***r*** is table, first capture is key. If ***r*** is function, it is passed all captured substrings, and should return replacement string, alternatively with a **nil** or **false** return, original match is retained. Returns second result number of substitutions (see Patterns below).

Patterns and pattern items

General pattern format: pattern_item [pattern_items]

cc	Matches a single character in the class **cc** (see Pattern character classes below).
cc*	Matches zero or more characters in the class **cc**; matches longest sequence.
cc-	Matches zero or more characters in the class **cc**; matches shortest sequence.
cc+	Matches one or more characters in the class **cc**; matches longest sequence.
cc?	Matches zero or one character in the class cc.
%n	(n = 1..9) Matches **n**-th captured string.
%bxy	Matches balanced string from character **x** to character **y** (e.g. nested parenthesis).
%f[set]	Frontier pattern matches an empty string at position where next character belongs to **set** and precious does not.

^	Anchor pattern to string start, must be first in pattern.
$	Anchor pattern to string end, must be last in pattern.
\0	Zero.

Pattern captures

(sub_pattern)	Stores substring matching sub_pattern as capture **%1**..**%9**, in order.
()	Stores current string position as capture **%1**..**%9**, in order.

Pattern character classes (cc's)

.	Any character.
%symbol	The symbol itself.
x	If **x** not **^$()%.[]*+-** or **?** the character itself.
[set **]**	Any character in any of the given classes, can also be a range [c1-c2].
[^set **]**	Any character not in set.

For all classes represented by single letters (%a, %c, etc.), the corresponding uppercase letter represents the complement of the class. For instance, %S represents all non-space characters.

%a	Any letter character
%c	Any control character.
%d	Any digit.
%l	Any lowercase letter.
%p	Any punctuation character
%s	Any whitespace character.
%u	Any uppercase letter.
%w	Any alphanumeric character.
%x	Any hexadecimal digit.

examples

string.find("Lua is great!", "is")
> 5 6

string.find("Lua is great!", "%s")
> 4 4

string.gsub("Lua is great!", "%s", "-")
> Lua-is-great! 2

string.gsub("Lua is great!", "[%s%l]", "*")
> L***********! 11

string.gsub("Lua is great!", "%a+", "*")
> * * *! 3

string.gsub("Lua is great!", "(.)", "%1%1")
> LLuuaa iiss ggrreeaatt!! 13

string.gsub("Lua is great!", "%but", "")
> L! 1

string.gsub("Lua is great!", "^.-a", "LUA")
> LUA is great! 1

string.gsub("Lua is great!", "^.-a", function (s)
return string.upper(s) end)
> LUA is great! 1

Function storage

string.dump (***function*** **)**
Returns binary representation of Lua ***function*** with no upvalues. Use with **loadstring ( )**.

Note: String indexes go from 1 to **string.len (** s **)**, from end of string if negative (index -1 refers to the last character).

The UTF8 Library

Basic support for the extended character set provided by UTF-8 standard.

utf8.char(…)
Return string of utf-8 characters based on passed integers.

utf8.charpattern
Returns a pattern string

utf8.codes(s)
Construction for iteration loops such as in
for p, c in utf8.codes do *body* end

utf8.codepoint(s [, i[, j]])
Returns code points from all characters in s

utf8.len(s [, i[, j]])
Returns number of characters in passed string s between i and j.

utf8..offset(s, n [, i])
Returns position in bytes of the n th character.

The I/O Library

The I/O functions return **nil** and a message on failure unless otherwise stated; passing a closed file handle raises an error.

File Input / Output

Two types of file support - default files for input and output, or implicit file descriptors.

io.open (***filename*** [, ***mode***] **)**
Opens ***filename fn*** in ***mode***:
"**r**" read [default], "**w**" write,
"**a**" append, "**r+**" update-preserve,
"**w+**" update-erase, "**a+**" update-append
Some systems (e.g. Windows) require a trailing "**b**" for binary mode. Returns file handle.

file:close ( )
Closes **file**. Garbage collector may also close.

file:read (***formats*** **)**
Returns a value from **file** for each of the passed ***formats***:
"**n**" reads a number,
"**a**" reads whole **file** as a string from current position,
"**l**" reads a line (**nil** at end of **file**) [default],
n = reads a string of up to **n** characters (**nil** at end of **file**).

file:lines (***format*** **)**
Returns an iterator function reading line-by-line from **file** as per ***format*** (default *l); the **file** not closed when finished.

file:write (***values*** **)**
Write each of ***values*** (strings or numbers) to **file**, with no added separators. Numbers are written as text, strings can contain binary data (may need binary mode read). Returns ***file*** or **nil** and error message.

file:seek ([***p***] [, ***offset***] **)**
Sets current position in **file** relative to ***p*** ("**set**" start of **file** [default], "**cur**" current, "**end**" end of file) adding ***offset*** [default: zero]. Returns new position in **file**.

file:setvbuf (***mode*** [, ***size***] **)**
Sets buffering mode for an output file. ***mode***: **"no"**, **"full"** and **"line"**, ***size*** in bytes

file:flush ( )
Writes to **file** any data still held in buffers.

Simple I/O

io.input ([***file***] **)**
Sets ***file*** as default input file; ***file*** can be either an open file object or a file name; in the latter case the ***file*** is opened for reading in text mode. Returns a file object, the current one if no file given; raises error on failure.

io.output ([***file***] **)**
Sets ***file*** as default output file (current output file is not closed); ***file*** can be either an open file object or a file name; in the latter case ***file*** is opened for writing in text mode. Returns a file object, the current one if no file given. Raises error on failure.

io.popen (***file*** [, ***mode***] **)**
Starts program in ***file*** in separate process. ***mode***: **"r"** (default) for read only, **"w"** for writing data to program. Returns a file handle for data access to program.

io.close ([***file***] **)**
Closes file object ***file***. Default: closes default output file. If ***file*** created by **io.popen**, a success returns exit status of process.

io.read (***formats*** **)**
Reads from default input file, same as **file:read ( )**.

io.lines ([***fn***] **)**
Opens file name ***fn*** for reading. Returns an iterator function reading from it line-by-line. Iterator closes file when finished. If no ***fn***, returns iterator reading lines from default input file.

io.write (***values*** **)**
Writes to the default output file, same as **file:write ( )**.

io.flush ( )
Writes to default output file any data in buffers.

Standard files and utility functions

io.stdin	Predefined input file object.
io.stdout	Predefined output file object.
io.stderr	Predefined error output file object.

io.type (***x*** **)**
Returns string "**file**" if ***x*** is an open file, "**closed file**" if ***x*** is a closed file, **nil** if ***x*** is not a file object.

io.tmpfile ( )
Returns file object for temporary file (deleted when program ends).

The OS Library

Many characteristics of this library are determined by operating system support. Unix and Unix like systems are assumed.

Date/time

Time and date accessed via time-table ***tt*** = {**year** = 1970-2135 , **month** = 1-12, **day** = 1-31, [**hour** = 0-23,] [**min** = 0-59,] [**sec** = 0-59,] [**isdst** = **true**-**false**,] }

os.time ([***tt***] **)**
Returns date/time, in seconds since epoch, described by table ***tt*** [default: current]. **Hour**, **min**, **sec**, **isdst** fields optional.

os.difftime (***t2***, ***t1*** **)**
Returns difference ***t2*** - ***t1*** between two **os.time ( )** values.

os.date ([***fmt*** [, ***t***]] **)**
Returns a table or string describing date/time ***t*** (that should be a value returned by **os.time**), according to the format string ***fmt***:

!	A leading “**!**” requests UTC time
***t**	Returns a table similar to time-table

while the following format a string representation:

%a	**%A**	Abbreviated, full weekday name.
%b	**%B**	Abbreviated, full month name.
%c		Date/time (default)
%d		Day of month (01..31).
%H	**%I**	Hour (00..23), (01..12).
%M		Minute (00..59).
%m		Month (01..12).
%p		Either “am” or “pm”.
%S		Second (00..61).
%w		Weekday (0..6), 0 is Sunday.
%x	**%X**	Date only, time only.
%y	**%Y**	Year (nn), (nnnn).
%Z		Time zone name if any

os.clock ( )
Returns the approx. CPU seconds used by program.

System interaction

os.execute (***string*** **)**
Calls system shell to execute ***string***, returns **true** on success or **nil** and error message.

os.exit ([***code***] **)**
Terminates script, returning ***code*** [default: success]. May close **state**.

os.getenv (***variable*** **)**
Returns a string with the value of the environment ***variable***, or **nil** if no ***variable*** exists.

os.setlocale (***string*** [, ***category***] **)**
Sets the locale described by ***string*** for ***category***: "**all**" (default), "**collate**", "**ctype**", "**monetary**", "**numeric**" or "**time**". Returns name of new locale, or **nil** if not set.

os.remove (***file*** **)**
Deletes ***file***, or returns **nil** and error description.

os.rename (***file1***, ***file2*** **)**
Renames ***file1*** to ***file2***, or returns **nil** and error message.

os.tmpname ( )
Returns a string usable as name for a temporary file. Subject to name conflicts - use **io.tmpfile()** instead.

FlyingSticks Lua Library

The FlyingSticks scripting library provides access to some of the inner workings of the ballistic calculator. Most displayed fields are accessible via their unique parameter name. These names become predefined global Lua variables.

FlyingSticks to Lua Bridging Library

The FlyingSticks Library provides a linkage between Lua and the inner workings of FlyingSticks. This linkage is very similar to the graphical user interface (GUI).

Parametric Variable Access

FlyingSticks parametric variables (parVars) are accessed either directly via the global " **pv** " table or by function call.

Direct parVar Access

Unlike standard Lua tables, the " **pv** " table only allows access to FlyingSticks predefined variables (e.g. pv.BowDrawLength = 0.780). Attempts to create a new entry in the table or accessing a non-existing parVar will cause a run-time error. Assigning a value outside the parVar's valid range it will set the parVar to the appropriate limit. ParVars accessed via the " **pv** " table are always in SI units.

Function parVar Access

An alternate method for accessing parametric variables is via two function calls get() and set(), with the advantage of providing additional options. The parVars are passed as an index obtained from the global " **vi** " table, so the direct access "**pv.BowMass**" becomes "**vi.BowMass**" (i.e. ***param*** below).

fly.get (***param*** [, ***modifier***] **)**

Gets the ***param*** from the FlyingSticks ballistics engine. ***param*** may be a name string or index. The optional ***modifier*** can be:

"text"	returns the string as seen in calculator's field
"SI"	returns a value in SI units
"CU"	returns a value in the current units

Returns **value** on success or **nil** and an error message.

fly.set (***param, value*** [, ***modifier***] **)**

Sets the ***param*** (name or index) to the ***value*** passed. The ***value*** may be a string or number. If a string it may contain units text and allow the current units to be changed. The optional ***modifier*** can be:

"text"	assumes ***value*** a string as in calculator's field
"SI"	assumes ***value*** in SI units (default)
"CU"	assumes ***value*** in the current units

Returns **true** on success or **nil** and an error message.

Other Control Functions

fly.button (***command*** [, ***arg1*** [, ***arg2*** …]] **)**

Notionally does same as a button click in FlyingSticks. Returns **true** on success or **nil** and error message. The following commands are available:

"resight"	Calculates the launch elevation and azimuth for current range and wind.
"launch"	This function launches an arrow with the current setup, running the full ballistics model.
"shooting" [,"enable"l"disable"]	Enables or disables the simulated group shooting and associated scoring.
"refreshScreen"	Refreshes the screen based on the current setup. This can be relatively slow compared to other commands, so should be used sparingly.
"clearResults"	Clears the results panel, ready to receive more script output. Not functional in current version.
"applyLocation"	Calculates gravity and various atmospheric parameters such as pressure, air density, viscosity, temperature and humidity. These parameters may subsequently be changed if desired.
"tuneAll"	Does a first estimate of spine tuning by adjusting the current arrow's static spine, shaft length and point mass. Sometimes may fail to find a sensible solution.
"tuneLength"	Leaving static spine and point mass unchanged, adjusts the shaft length within sensible limits to find a solution.
"tuneStaticSpine"	Leaving shaft length and point mass unchanged, adjusts the static spine within sensible limits to find a solution.
"tunePoint"	Leaving static spine and shaft length unchanged, adjusts the shaft length within sensible limits to find a solution.
"sightlineHorizontal"	Sets the target height to the peep or eye height.
"setOtherDiameters"	One an arrow's shaft diameter(s) have been loaded, can be used to set the point, insert and nock diameters.
"ethicalRange"	Calculates the ethical range for the current kit and game type without loading it to the current range.
"scoreRound"	Calculates the current round score for the archer's current form. This generally happens automatically.
"updateWind"	Needs to be called after changing wind speed, direction, inclination, gusting, or land surface.
"autoRunOn"	Ensures the script is run each time the calculator does a full recalculation. Use with caution as it can consume excessive CPU time.
"autoRunOff"	Turns autoRun mode off.

fly.printCSV (***param1*** [, ***param2*** [, ***param3*** …]] **)**

Writes a line of coma separated ***param*** values to the results screen. The ***param*** may be either a FlyingSticks parameter string or a Lua variable.

fly.printU (***param1*** [, ***param2*** [, ***param3*** …]] **)**

Similar to printCSV except it includes the units-text and thousands separators.

fly.pPlot (***plot, xAxis*****,** ***yAxis*** [, ***cAxis***] **)**

A parametric plot function that generates its own data by running the currently configured ballistic calculator and plotting the results. The function inputs are an independent variable (x-axis), a dependent variable (y-axis) and an optional control variable. The control variable will generate a trace for each control value. The variables may be any of FlyingStick's valid parameters, however care needs to be taken in ensuring sensible selections.

This function provides a similar plot to that in FlyingSticks' Plots>Parametric_Plots panel, except it has greater flexibility.

A pPlot is defined by three or four arguments:

fly.pPlot (***plot, xAxis*****,** ***yAxis*** [, ***cAxis***] **)**

Where:

The required plot argument defines the general plot style or simply the plot's title:

plot = "title" | {***title***="title", ***autoSight***=f, ***autoUpdate***=f,
precision=f, ***multiPlot***=f, ***grid***=f }

The required independent variable argument:

xAxis = "var" | { ***var***="var", ***label***="varName",
min=f, ***max***=f, ***step***=f | {f, f, f…} }

The required dependent variable argument:

yAxis = "var" | { ***var***="var", ***label***="varName",
min=f, ***max***=f, ***step***=f | {f, f, f…} }

The optional control variable argument:

cPara = "var" | { ***var***="var", ***label***="varName",
min=f, ***max***=f, ***step***=f | {f, f, f…} }

Where:

title	title text placed at top of plot area.
autoSight	=1 re-sights target for each point generated, =0 no re-sighting (default).
autoUpdate	=1 re-runs the Lua script each time FlyingSticks recalculates. Defaults to 0.
precision	=0 standard precision (default), =1 approx. 10x improved statistical calculations.
multiPlot	=1 allows multiple plots in window =0 window per plot (default).
grid	=0 no grid, =1 vertical grid, =2 horizontal grid, =3 full grid (default).
label	the axis label. If not provided will be created from parameter name.
min	minimum axis value. If not provided, is set to 1/2 of current value.
max	maximum axis value. If not provided, is set to 2x current value.
step	step size in incrementing independent or control variable, or alternatively a table of specific steps. If not provided, set for 5 equal steps.

Various parts may be omitted and best endeavours are made to calculated sensible values.

The function opens a Script Plot window into which the plot is placed. If the script generates more than one plot (i.e. multiple pPlot() calls, then by default a new window will be opened for each plot. If ***multiPlot***=1 has been set, then all plots will be placed as a vertical stack in a single window.

A maximum of 8 plots can be generated.

Known Issues

1. The FlyingSticks scripts Results view does not handle UTF8's extended characters as expected. Thus for example 1.200 kg/m³ will appear as 1.200 kg/mÂ³. All other fields in the calculator function correctly. This has proven an intractable problem to date.

FlyingSticks Parameter List

Parameter Name	Description	Index	Typical Value
AirApparentTemp	Apparent temperature	0000	15.6°C
AirCloud	Cloud reflectance	0001	0.0 %
AirDensity	Air density	0002	1.200 kg/m³
AirDewPoint	Dew point	0003	9.2°C
AirDynamicViscosity	Air dynamic viscosity	0004	18.57 μPa·s
AirMeasurementHeight	Wind measurement height	0005	2.00 m
AirPressure	Air pressure	0006	1013.2 hPa
AirRadiation	Radiant heat flux	0007	0 W/m²
AirRelativeHumidity	Relative humidity	0008	50.0 %
AirSpeedSound	Speed of sound in air	0009	344.0 m/s
AirTemperature	Air temperature	0010	20.0°C
AirTimeOfDay	Time of solar day	0011	12.0 hr
AirWindDirection	Wind direction	0012	90.0°
AirWindGradThreshold	Wind gradient threshold	0013	5.000 m
AirWindSpeed	Wind speed	0014	10.0 m/s
AirWindVariability	Wind variability	0015	50.0 %
AirWindVertDirection	Wind vertical direction	0016	0.0°
AshbyMechanicalAdvantage	Ashby's mechanical advantage	0017	-16 %
AshbyTipDesign	Ashby's tip design	0018	1 %
AshbyBevel	Ashby's bevel contribution	0019	-14 %
AshbySlickness	Ashby's slickness contribution	0020	0 %
AshbyFoC	Ashby's FoC contribution	0021	0 %
AshbySharpness	Ashby's sharpness contribution	0022	0 %
AshbyTaper	Ashby's taper contribution	0023	0 %
AshbyMassThreshold	Ashby's mass threshold contrib…	0024	0 %
AshbyFeruleRatio	Ashby's ferule ratio contribution	0025	-20 %
AshbyArrowIntegrity	Ashby's arrow integrity	0026	-0 %
AshbyNet	Ashby's net contribution	0027	-41 %
AshbyNetFactor	Ashby's factor	0028	0.590
ArrowAeroLength	Arrow aerodynamic length	0029	771.0 mm
ArrowBoundaryLayer	Boundary layer thickness	0030	13.2 mm
ArrowCalcResonantDamp	Resonant dampening, calculated	0031	1.341 s
ArrowCalcResonantFreq	Resonant frequency, calculated	0032	68.2 Hz
ArrowCenterOfGavity	Arrow center of gravity	0033	107.5 mm
ArrowCenterOfPressure	Arrow center of pressure	0034	-16.8 mm
ArrowDragCoef	Drag coef.	0035	2.217
ArrowDragCoefFromBallistics	Drag coef. from ballistics	0036	2.217
ArrowDragCoefFromTheory	Drag coef. from theory	0037	2.217
ArrowDragCoefManual	Drag coef. manual	0038	2.000
ArrowDragContFletch	Drag contribution of fletches	0039	10.4 %
ArrowDragContNock	Drag contribution of nock	0040	8.8 %
ArrowDragContPoint	Drag contribution of point	0041	24.8 %
ArrowDragContShaft	Drag contribution of shaft	0042	56.0 %
ArrowDragTrim	Drag trim	0043	0.0 %
ArrowDynamicSpine	Arrow dynamic spine	0044	37.79 kgf
ArrowFeruleRatio	Ferule to shaft diameter ratio	0045	98.4 %
ArrowFrontOfCenter	Front of center (FoC)	0046	14.4 %
ArrowLiftCoefSlope	Lift coef. slope	0047	0.00560 /°
ArrowManResonantAmp	Resonant amplitude, manual	0048	15.00 mm
ArrowManResonantDamp	Resonant dampening, manual	0049	1.000 s
ArrowManResonantFreq	Resonant frequency, manual	0050	80.0 Hz
ArrowManResonantOrient	Resonant orientation, manual	0051	90.0°
ArrowManResonantPhase	Resonant phase, manual	0052	0.0°
ArrowMass	Arrow mass	0053	412.4 gr
ArrowMassCalculated	Arrow mass calculated	0054	412.4 gr
ArrowMassMeasured	Arrow mass measured	0055	379.9 gr
ArrowMomentOfInertia	Moment of inertia about CoG	0056	0.002 kg·m²
ArrowNetSectionalArea	Net sectional area	0057	0.6 cm²
ArrowReynoldsNumberDia	Reynolds Number ref. diameter	0058	38,360
ArrowReynoldsNumberLen	Reynolds Number ref. length	0059	3,765,000
ArrowSectionalDensity	Sectional density	0060	586.0 kg/m²
ArrowSelectedResonantDamp	Resonant damping, selected	0061	1.341 s
ArrowSelectedResonantFreq	Resonant frequency, selected	0062	68.2 Hz
ArrowSpinSpeedConstant	Spin speed constant	0063	36.46 rpm/m/s
ArrowSpinTimeConstant	Spin time constant	0064	0.087 s
ArrowStabilityFactor	Arrow stability factor	0065	16.6 %
ArrowStdLength	Arrow standard length	0066	748.0 mm
ArrowSteadyStateSpinRate	Spin speed, steady-state	0067	2840 rpm

ArrowTerminalVelocity	Terminal velocity	0068	65.7 m/s
ArrowTransitionalRe	Transitional Reynolds Number	0069	0.000
ArrowYawDamp	Yaw dampening	0070	0.000 s
ArrowYawFreq	Yaw frequency	0071	0.0 Hz
ArrowIntegrity	Arrow integrity	0072	0.000
BallisticDataArrowMass	Ballistic data arrow mass	0073	401.2 gr
BallisticDataDrop	Ballistic data drop	0074	3.727 m
BallisticDataEstLaunchSpeed	Ballistic data est. launch speed	0075	59.5 m/s
BallisticDataGroupSize	Ballistic data group size	0076	260 mm
BallisticDataRange	Ballistic data range	0077	50.0 m
BallisticDataScaleValue	Ballistic data scale value	0078	2 mm
BallisticDataSightPointHeight	Ballistic data sight point height	0079	71 mm
BallisticDataVertTargetError	Ballistic data vertical target error	0080	0 mm
BowArrowMassForMeasuredSpeed1	Bow arrow mass for measured speed1	0081	300.0 gr
BowArrowMassForMeasuredSpeed2	Bow arrow mass for measured speed2	0082	463.0 gr
BowAvailableEnergy	Available energy	0083	99.28 J
BowBraceHeight	Brace height	0084	171.4 mm
BowClaimedSpeed	Claimed speed	0085	101.2 m/s
BowCogX	Bow CoG x	0086	-25 mm
BowCogY	Bow CoG y	0087	-100 mm
BowCogZ	Bow CoG z	0088	0 mm
BowDesignTrim	Design trim	0089	2.5 %
BowDrawCurveDistance	Draw curve distance	0090	0 mm
BowDrawCurveEfficiency	Draw curve efficiency	0091	87.0 %
BowDrawCurveEnergy	Draw curve energy	0092	109.1 J
BowDrawCurveEstimateEfficiency	Draw curve estimate efficiency	0093	71.0 %
BowDrawCurveForce	Draw curve force	0094	0.000 kgf
BowDrawNetEfficiency	Bow draw net efficiency	0095	71.0 %
BowDropOff	Bow drop off	0096	75.0 %
BowEffectDraw	Effect draw weight	0097	24.7 kgf
BowEfficiency	Bow efficiency	0098	87.0 %
BowEnergyTrim	Bow energy trim	0099	2.5 %
BowLength	Bow length	0100	762 mm
BowManufDrawLength	Draw length, manufacturer's	0101	0 mm
BowManufDrawWeight	Draw weight, manufacturer's	0102	0.000 kgf
BowMass	Bow mass	0103	1633 g
BowMeasuredSpeed1	Bow measured speed1	0104	98.5 m/s
BowMeasuredSpeed2	Bow measured speed2	0105	85.0 m/s
BowMinArrowMass	Min. safe arrow mass per draw	0106	73 mg/N
BowMyDrawForce	Draw weight	0107	24.95 kgf
BowMyDrawLength	Draw length	0108	690.0 mm
BowMyDrawLengthTolerance	Draw length tolerance	0109	2.0 mm
BowNockAboveSquare	Nock above square	0110	6.5 mm
BowNockAboveSquareInitial	Nock above square, initial	0111	18.1 mm
BowOptDynamicSpine	Bow dynamic spine	0112	52.35 kgf
BowPlungerMaxLoad	Plunger maximum load force	0113	0.816 kgf
BowPlungerPosition	Plunger position	0114	-3.5 mm
BowPlungerPreLoad	Plunger pre load force	0115	0.357 kgf
BowPlungerTravel	Plunger travel	0116	10.0 mm
BowPowerStroke	Power stroke	0117	518.6 mm
BowRestAboveCenter	Rest above center	0118	45.0 mm
BowStringStrands	Number of string strands	0119	18
BowSystCogX	Bow system CoG x	0120	-25 mm
BowSystCogY	Bow system CoG y	0121	-100 mm
BowSystCogZ	Bow system CoG z	0122	0 mm
BowTemperature	Limb temperature	0123	20.0°C
BowTillerDifference	Tiller difference	0124	0.0 mm
BowTillerSpace	Tiller measurement spacing	0125	0 mm
BowVirtualMass	Virtual mass	0126	92.6 gr
BowVirtualMassAdjust	Virtual mass adjust	0127	0.0 gr
FletchLength	Fletch length	0128	80.0 mm
FletchNetMass	Fletch net mass	0129	10.2 gr
FletchNumOfFletches	Number of fletches	0130	3
FletchOffsetAngle	Fletch offset angle	0131	2.0°
FletchOffsetDistance	Fletch offset distance	0132	2.8 mm
FletchPosition	Fletch position	0133	20.0 mm
FletchSpinPitch	Fletch spin travel pitch	0134	1.6 m
FletchThickness	Fletch thickness	0135	0.40 mm
FletchUnitMass	Fletch unit mass	0136	3.4 gr
FletchWidth	Fletch width	0137	13.0 mm
InsertCogFromShaft	Insert CoG from shaft	0138	-15.0 mm
InsertDiameter	Insert diameter	0139	7.00 mm
InsertExtendLength	Insert extend length	0140	1.0 mm

InsertMass	Insert mass	0141	11.0 gr
LaunchAcceleration	Average launch acceleration	0142	596.6 G
LaunchAverageSpeed	Average speed	0143	71.6 m/s
LaunchAzimuth	Azimuth	0144	0.00°
LaunchCalcRotateOscDamp	Rotate osc. dampening, calculated	0145	0.486 s
LaunchCalcRotateOscFreq	Rotate osc. frequency, calculated	0146	4.32 Hz
LaunchCanter	Canter angle	0147	0.0°
LaunchCanterAimingOffsetY	Canter vertical aiming offset	0148	0 mm
LaunchCanterAimingOffsetZ	Canter horizontal aiming offset	0149	0 mm
LaunchCanterTolerance	Canter angle hold tolerance	0150	0.3°
LaunchCloutFlightRatio	Clout range to flight range ratio	0151	0.0 %
LaunchCurrentCanterAzimuthOffset	Canter azimuth offset, current	0152	0.0°
LaunchDriftY	Launch drift y	0153	0 mm
LaunchDriftZ	Launch drift z	0154	0 mm
LaunchElevation	Elevation	0155	3.4°
LaunchElevationMaxRange	Elevation for maximum range	0156	40.0°
LaunchElevationMaxRangeCurent	Elevation for maximum range, current	0157	40.0°
LaunchFlightTime	Flight time	0158	8.374 s
LaunchFlightTimeCurrent	Flight time, current	0159	8.374 s
LaunchHoriOffsetAngle	Launch horizontal offset angle	0160	0.0°
LaunchImpactPrecision	Target impact precision	0161	0.3 mm
LaunchInitBowRotateSpeed	Initial bow rotate speed	0162	-inf rpm
LaunchInitialDragDeceleration	Initial deceleration by drag	0163	1.603 G
LaunchInitRotateAmplitude	Initial rotate amplitude	0164	3.0°
LaunchInitRotateAmpNock	Initial rotate amplitude at nock	0165	25.21 mm
LaunchInitRotateOrientation	Initial rotate orientation	0166	90.0°
LaunchInitRotatePhase	Initial rotate phase	0167	0.0°
LaunchInitRotatePhaseTolerance	Initial rotate phase tolerance	0168	5.0°
LaunchInitRotateSpeed	Initial rotate speed	0169	16 rpm
LaunchLaunchSpeed	Launch speed	0170	77.9 m/s
LaunchManualRotateOscDamp	Rotate osc. dampening, manual	0171	0.250 s
LaunchManualRotateOscFreq	Rotate osc. frequency, manual	0172	5.00 Hz
LaunchMaxFlightRange	Flight range	0173	320.0 m
LaunchMaxFlightRangeCurrent	Flight range, current	0174	320.0 m
LaunchMaxHeight	Maximum height	0175	2.588 m
LaunchMaxHeightAtMaxRange	Maximum height for flight range	0176	88.46 m
LaunchMaxHeightAtMaxRangeCurrent	Maximum height for flight range, current	0177	88.46 m
LaunchMaxHeightRange	Range of maximum height	0178	34.2 m
LaunchMaxHeightSightline	Maximum height above sightline	0179	1.111 m
LaunchPlotHeightOffset	Plot height offset	0180	0.000 m
LaunchResonanceWavelength	Resonance ground wavelength	0181	1.1 m
LaunchReversalHeight	Flight reversal height	0182	#####
LaunchReversalRange	Flight reversal range	0183	#####
LaunchRotateAngleAtImpactY	Rotate vertical angle at impact	0184	0.0°
LaunchRotateAngleAtImpactZ	Rotate horizontal angle at impact	0185	0.0°
LaunchRotateDeviation	Rotate induced deviation	0186	0 mm
LaunchRotateExcursion	Rotate excursion	0187	0 mm
LaunchRotateOffsetAtImpactY	Rotate vertical offset at impact	0188	0 mm
LaunchRotateOffsetAtImpactZ	Rotate horizontal offset at impact	0189	0 mm
LaunchRotateWavelength	Rotation ground wavelength	0190	15.6 m
LaunchScore	Score	0191	519
LaunchSpineMatch	Spine match	0192	-0.360
LaunchSweetRangeMax	Sweet range maximum	0193	20.7 m
LaunchSweetRangeMin	Sweet range minimum	0194	5.5 m
LaunchSweetRangeSet	Sweet range set range	0195	18.5 m
LaunchTimeIncrement	Time increment for ballistic engine	0196	1.0000 ms
LaunchVertOffsetAngle	Vertical offset angle	0197	0.0°
LaunchWindCanterCalib	Wind - canter calibration factor	0198	0.97°/m/s
LaunchWindCanterRequired	Canter required for wind	0199	9.7°
LaunchXSightlineNegRange	Launch x sightline negative range	0200	70.0 m
LocationAltitude	Location altitude	0201	200.0 m
LocationGravity	Location gravity	0202	1.0000 G
LocationLatitude	Location latitude	0203	35.0°
LocationLongitude	Location longitude	0204	0.0°
NockCogFromShaft	Nock CoG from shaft	0205	3.0 mm
NockDiameter	Nock diameter	0206	7.50 mm
NockLength	Nock length	0207	15.0 mm
NockMass	Nock mass	0208	9.0 gr
NockNock2ShaftLength	Nock to shaft length	0209	8.0 mm
NockWidthAtString	Nock width at string	0210	5.0 mm
PerformCloutGroupHeightChange	Clout group height change	0211	#####
PerformCurrentGroupHoriAngSD	Current group horizontal angle SD	0212	18.0 MOA
PerformCurrentGroupSkew	Current group skew	0213	1.00

PerformCurrentGroupVertAngSD	Current group vertical angle SD	0214	17.9 MOA
PerformRoundGroupSolutionScore	Round group solution score	0215	608.1
PerformRoundGroupSolutionXs	Round group solution X's	0216	4.0
PerformRoundScoreEst2SD	Round score estimated 2SD	0217	31.5
PerformRoundScoreEstimate	Round score estimate	0218	519.3
PerformRoundScoreEstimateX	Round score estimate X's	0219	1.9
PerformRoundXsScoreEst2SD	Round Xs score estimated 2SD	0220	2.6
PerformGroupFaceDia	Face diameter	0221	1220 mm
PerformGroupRoundScore	Round score	0222	608
PerformGroupRoundXs	Round X's score	0223	4
PerformGroupHeight	Group height	0224	370 mm
PerformGroupHeightNoIncludes	Group height, no includes	0225	366 mm
PerformGroupHoriDiaDirect	Group width, direct	0226	367 mm
PerformGroupRoundScoreMax	Maximum possible round score	0227	720
PerformGroupRoundXCountMax	Maximum possible round X count	0228	72
PerformGroupVertDiaDirect	Group height, direct	0229	363 mm
PerformGroupWidth	Group width	0230	765 mm
PerformGroupWidthNoIncludes	Group width, no includes	0231	367 mm
PerformGustGroupHeight	Gust group height	0232	4 mm
PerformGustGroupWidth	Gust group width	0233	398 mm
PerformHuntGroupHeight	Hunter group height	0234	482 mm
PerformHuntGroupHeightNoIncludes	Hunter group height, no includes	0235	476 mm
PerformHuntGroupWidth	Hunter group width	0236	1002 mm
PerformHuntGroupWidthNoIncludes	Hunter group width, no includes	0237	480 mm
PerformRangeDirect	Range for group, direct	0238	70.0 m
PerformRangeEstError	Range estimation error	0239	± 10.0 %
PerformRangeEstErrorGroupHeight	Range estimation error on group height	0240	0 mm
PerformSplashScore	Splash score	0241	608.9
PerformSplashScore2SD	Splash score 2SD	0242	20
PerformSplashXs	Splash X's count	0243	3.9
PerformSplashXs2SD	Splash X's 2SD	0244	4
PerformUkRating	Archer's Form Rating	0245	0.0
PerformUkRatingTemp	Archer's Rating in Handicap Panel	0246	25.9
PerformOzRating	Archer's Form Rating	0247	0.0
PerformOzRatingTemp	Archer's Rating in Handicap Panel	0248	84.3
PerformHandicap	Archer's Round Handicap	0249	112
PersonAnchorHeight	Archer's anchor height	0250	1530 mm
PersonArmSpan	Archer's arm span	0251	1795 mm
PersonBirthYear	Archer's birth year	0252	1900
PersonGroupShotPeriod	Time between simulated releases	0253	4.0 s
PersonGroupShots	Number of shots in a group end	0254	6
PersonHeight	Archer's height	0255	1770 mm
PersonMass	Archer's weight	0256	70.0 kg
PersonMaxDraw	Archer's max comfortable draw	0257	24.9 kgf
PersonMaxIterationLoops	Iteration loops allowed	0258	500
PersonNumPlotVarSteps	Plots: number of x-axis steps	0259	0
PersonPenetrationLinear	Target penetration linear term	0260	1.0000
PersonPenetrationPower	Target penetration power term	0261	1.3500
PersonPlotSpanAngle	Plot span angle	0262	0.0°
PersonPlotSpanDrift	Plot span drift	0263	0 mm
PersonPlotSpanOffset	Plot span offset	0264	0 mm
PersonPlotSpanTrajectory	Plot span trajectory	0265	0.0 m
PersonResightFaceMultiple	Resight trigger, face multiple	0266	4.0
PersonSpineCalibForm	Archer's spine tuning trim	0267	0.0 %
PointBladeUnitArea	Point blade unit area	0268	0.0 cm²
PointBladeCutDiameter	Point blade cut diameter	0269	7.5 mm
PointBladeLength	Point blade length	0270	0.0 mm
PointBladeThickness	Point blade thickness	0271	0.00 mm
PointCogDistance	Point CoG distance from shaft	0272	10.0 mm
PointFerruleDiameter	Point ferrule diameter	0273	7.50 mm
PointLength	Point length	0274	15.0 mm
PointMass	Point mass	0275	120.0 gr
PointMechanicalAdvantage	Point mechanical advantage	0276	0.50
PointNumOfBlades	Point number of blades	0277	0
PointSweepAngle	Point sweep angle	0278	0.0°
PointTipWidth	Point blade tip width	0279	5.0 mm
ShaftDiameterMax	Shaft diameter	0280	7.62 mm
ShaftDiameterFront	Shaft diameter front	0281	7.62 mm
ShaftDiameterRear	Shaft diameter rear	0282	7.62 mm
ShaftFootLength	Shaft foot length	0283	0.0 mm
ShaftFootMassPerLen	Shaft foot mass per length	0284	12.00 gr/in
ShaftInternalDiameter	Shaft internal diameter	0285	6.62 mm
ShaftLength	Shaft length	0286	740.0 mm

ShaftMass	Shaft mass	0287	262.2 gr
ShaftMassPerLength	Shaft mass per length	0288	9.00 gr/in
ShaftSpineAMO	Shaft spine AMO	0289	35.93 kgf
ShaftStaticSpine	Shaft static spine	0290	8.34 mm(AMO)
SightArmLength	Sight arm length	0291	69.8 mm
SightArrowMass1	Sight arrow mass 1	0292	401.2 gr
SightArrowMass2	Sight arrow mass 2	0293	#####
SightBallisticSpeed1	Sight ballistic speed 1	0294	59.8 m/s
SightBallisticSpeed2	Sight ballistic speed 2	0295	#####
SightBarrelIntDia	Sight barrel internal diameter	0296	32.0 mm
SightCloutAngle	Sight clout angle	0297	5.5°
SightCloutHeight	Sight clout height	0298	0.0 mm
SightCurrentPinHeight	Sight current pin height	0299	67.6 mm
SightDragCoef1	Sight drag coef. 1	0300	2.217
SightDragCoef2	Sight drag coef. 2	0301	#####
SightFontSize	Tape font size	0302	10.0 pt
SightLastPrintedLength	Tape last printed length	0303	#####
SightMajorTick	Tape major tick interval	0304	10.0 m
SightMass	Sight mass	0305	200.0 g
SightMaxRangePinHeight	Sight maximum range pin height	0306	65.5 mm
SightMaxSafeSightRange	Sight maximum safe sight range	0307	0.0 m
SightMaxTapeRange	Sight maximum tape range	0308	90.0 m
SightMinorTicksPerMajor	Tape minor ticks per major	0309	10
SightMinRangePinHeight	Sight minimum range pin height	0310	101.7 mm
SightMinSafePinHeight	Sight minimum safe pin height	0311	40.0 mm
SightMinTapeRange	Tape minimum range	0312	5.0 m
SightNumberToPrint	Number of tapes to print	0313	1
SightNumOfPins	Number of sight pins	0314	1
SightPeep2NockArrow	Peep (eye) to nock along arrow	0315	159.5 mm
SightPeep2Sight	Peep (eye) to sight pin	0316	632.0 mm
SightPeep2Arrow	Peep (eye) height above arrow	0317	120.0 mm
SightPendulumHeight	Sight pendulum height	0318	50.0 mm
SightPinHoriOffset	Sight pin horizontal offset	0319	-0.0 mm
SightPinSpacingRange	Multi-pin range spacing	0320	10.0 m
SightPinWindage	Sight pin windage offset	0321	-0.0 mm
SightPivotHeight	Sight pivot height	0322	88.0 mm
SightPivotLength	Sight pivot length	0323	44.4 mm
SightPrintScaleTrim	Tape print scale trim	0324	0.0 %
SightRefBallisticsArrowMass	Sight ref. ballistics arrow mass	0325	0.0 gr
SightRefBallisticsMeanVelocity	Sight ref. ballistics mean velocity	0326	0.0 m/s
SightRefBallisticsRange	Sight ref. ballistics range	0327	0.0 m
SightRefBallisticsSightPointHeight	Sight ref. ballistics sight point height	0328	0.0 mm
SightScaleMultiplier	Tape scale multiplier	0329	1.000
SightScaleTurnoverRange	Tape scale turnover range	0330	12.4 m
SightStandardDeviation	Sight standard deviation	0331	± 0.3 mm
SightTapeCheck	Tape check value	0332	40.0 mm
SightTapeCropMargin	Tape crop margin	0333	5.0 mm
SightTapeDatumOffset	Tape reference offset	0334	0.0 mm
SightTapeLength	Tape length	0335	60.0 mm
SightTapeWidth	Tape width	0336	6.0 mm
SightTurnoverPinHeight	Sight turnover pin height	0337	107.2 mm
SightVirtualMass	Sight virtual mass	0338	#####
SightWorkingPinHeight	Sight working pin height	0339	78.7 mm
StabilizerEditRodNum	Stabilizer edit rod number	0340	1
StabilizerMiShoulderFull	MoI, shoulder full	0341	1.073 kg·m²
StabilizerMiShoulderNoStab	MoI of shoulder, no stabilizer	0342	1.073 kg·m²
StabilizerRodAttachX	Stabilizer rod attach x	0343	50 mm
StabilizerRodAttachY	Stabilizer rod attach y	0344	-120 mm
StabilizerRodAattachZ	Stabilizer rod attach z	0345	0 mm
StabilizerRodAzimuth	Stabilizer rod azimuth	0346	5.0°
StabilizerRodContribution	Stabilizer rod contribution	0347	0.0 %
StabilizerRodElevation	Stabilizer rod elevation	0348	5.0°
StabilizerRodLength	Stabilizer rod length	0349	950 mm
StabilizerRodMass	Stabilizer rod mass	0350	250.0 g
StabilizerRodWeight	Stabilizer rod weight	0351	500.0 g
StabilizerTotalMass	Stabilizer total mass	0352	0.0 g
StabilizerWobbleFrequency	Stabilizer wobble frequency	0353	0.47 Hz
StabilizerWobbleReduction	Stabilizer wobble reduction	0354	0.0 %
StabilizerWobbleToGroupSizeProp	Stabilizer wobble to group size prop	0355	45.0 %
TargetAimingOffsetDistanceY	Aiming vertical offset distance	0356	0 mm
TargetAimingOffsetDistanceZ	Aiming horizontal offset distance	0357	0 mm
TargetAlarmTime	Target alarm time	0358	0.780 s
TargetAngle	Target angle	0359	-4.3°

TargetAngleAzmuth	Target angle azimuth	0360	-0.0°
TargetArcDistance	Arc travel distance	0361	70.1 m
TargetDropRate	Drop rate at target	0362	-75 mm/m
TargetEthicality	Target ethicality	0363	0 %
TargetFlightTime	Flight time	0364	0.984 s
TargetGameKillDiam	Game kill zone diameter	0365	244 mm
TargetGameOrientation	Game orientation relative to archer	0366	90°
TargetGameMaxRib	Game maximum rib thickness	0367	3.0 mm
TargetGameRibCoverage	Game rib coverage	0368	100 %
TargetGamePenetration	Game penetration required at orientation	0369	10 mm
TargetHeightFromGnd	Target height relative to archer's ground	0370	1.300 m
TargetImpactSpeed	Impact speed	0371	66.3 m/s
TargetInclination	Target inclination	0372	12.5°
TargetKineticEnergy	Impact kinetic energy	0373	58.83 J
TargetLeadDistance	Lead distance	0374	0.0 m
TargetMarginTime	Alert time margin	0375	0.000 s
TargetMassThreshold	Ashby's threshold arrow mass	0376	24.1 gr
TargetMaximumEthicalRange	Maximum ethical range	0377	#####
TargetMaxRange	Maximum range	0378	70.4 m
TargetMinRange	Minimum range	0379	69.6 m
TargetMomentum	Impact momentum	0380	1.773 kg·m/s
TargetMovementSpeed	Target movement speed	0381	0.5 m/s
TargetPeneForce	Average penetration force	0382	1470.9 N
TargetPenetationFactor	Penetration factor	0383	0.9613 MJ/m²
TargetPenetration	Penetration	0384	482 mm
TargetPenetrationNoRib	Penetration without bone strike	0385	0 mm
TargetPenetrationDelta	Penetration change	0386	0 mm
TargetRange	Range	0387	70.0 m
TargetRangeTol	Range tolerance	0388	± 0.4 m
TargetReactionTime	Game reaction time	0389	#####
TargetReturnTime	Sound return time	0390	1.188 s
TargetSightlineDistance	Sightline distance	0391	70.0 m
TargetSpeedAccuracy	Speed accuracy	0392	± 25 %
TargetSpinSpeed	Impact spin speed	0393	2418 rpm
TargetTolerance	Target zone tolerance	0394	± 25 mm
TargetToleranceGame	Target tolerance zone for game	0395	± 0 mm
TargetXOffset	Target x offset	0396	-20,000 mm
TargetYOffset	Target y offset	0397	-0 mm
TargetZOffset	Target z offset	0398	0 mm
BowPerformFromBallisticsAvailableEnergy	Available energy from ballistics	0399	57.22 J
BowPerformFromBallisticsLaunchSpeed	Launch speed from ballistics	0400	59.1 m/s
BowPerformFromBallisticsVirtualMass	Virtual mass from ballistics	0401	92.6 gr
BowPerformFromDesignAvailableEnergy	Available energy from design	0402	99.28 J
BowPerformFromDesignLaunchSpeed	Launch speed from design	0403	77.9 m/s
BowPerformFromDesignVirtualMass	Virtual mass from design	0404	92.6 gr
BowPerformFromManufLaunchSpeed	Launch speed from manufacturer	0405	81.2 m/s
BowPerformSpeedMeasVirtualMass	Virtual mass from speed measurement	0406	175.3 gr
PerformRoundSetup0Range	Round setup 0 range	0407	90.0 m
PerformRoundSetup0Face	Round setup 0 face	0408	1220 mm
PerformRoundSetup0Arrows	Round setup 0 arrows	0409	36
PerformRoundScores0Score	Round scores 0 score	0410	261
PerformRoundScores0XCount	Round scores 0 X-count	0411	1
PerformRoundScores0Error	Round scores 0 error	0412	#####
PerformRoundScores0Cert	Round scores 0 cert	0413	#####
PerformRoundSetup1Range	Round setup 1 range	0414	70.0 m
PerformRoundSetup1Face	Round setup 1 face	0415	1220 mm
PerformRoundSetup1Arrows	Round setup 1 arrows	0416	36
PerformRoundScores1Score	Round scores 1 score	0417	297
PerformRoundScores1XCount	Round scores 1 X-count	0418	2
PerformRoundScores1Error	Round scores 1 error	0419	#####
PerformRoundScores1Cert	Round scores 1 cert	0420	#####
PerformRoundSetup2Range	Round setup 2 range	0421	50.0 m
PerformRoundSetup2Face	Round setup 2 face	0422	800 mm
PerformRoundSetup2Arrows	Round setup 2 arrows	0423	36
PerformRoundScores2Score	Round scores 2 score	0424	302
PerformRoundScores2XCount	Round scores 2 X-count	0425	2
PerformRoundScores2Error	Round scores 2 error	0426	#####
PerformRoundScores2Cert	Round scores 2 cert	0427	#####
PerformRoundSetup3Range	Round setup 3 range	0428	30.0 m
PerformRoundSetup3Face	Round setup 3 face	0429	800 mm
PerformRoundSetup3Arrows	Round setup 3 arrows	0430	36
PerformRoundScores3Score	Round scores 3 score	0431	340
PerformRoundScores3XCount	Round scores 3 X-count	0432	6

PerformRoundScores3Error	Round scores 3 error	0433	#####
PerformRoundScores3Cert	Round scores 3 cert	0434	#####