penlight

Lua libraries for use in LuaLaTeX

Kale Ewasiuk (kalekje@gmail.com)

2022–10–24

Documentation for the Lua library this includes can be found here:
`https://lunarmodules.github.io/Penlight`
This package uses version `1.13.1`

Required Package Option

The first option sent to this package MUST be one of:
`[penlight]` or `[pl]`.
All Penlight sub-modules are then available under this global variable by either
`penlight.XYZ` or `pl.XYZ`

texlua usage

If you want to use Penlight (and extras) with the `texlua` intrepreter (no document made, only for Lua files, useful for testing), you can access it by setting `__SKIP_TEX__ = true` and adding the package to path. For example:

```
package.path = package.path .. ';'..'path/to/texmf/tex/latex/penlight/?.lua'
penlight = require'penlight'
-- below is optional
__SKIP_TEX__ = true  --only required if you want to use
                     --penlightextras without a LaTeX run
__PL_GLOBALS__ = true -- optional, include global definitions
__SKIP_LUAKEYS__ = true -- luakeys is laoded in penlightextras be default, you may skip
require'penlightextras'
```

Additional Package Options

`stringx`	will import additional string functions into the string meta table. this will be ran in pre-amble: `require('pl.stringx').import()`
`format`	allows % operator for Python-style string formating this will be ran in pre-amble: `require('pl.stringx').format_operator()` `https://lunarmodules.github.io/Penlight/libraries/pl.stringx.html#format_oper`
`func`	allows placehold expressions eg. `_1+1` to be used this will be ran in pre-amble: `penlight.utils.import('pl.func')` `https://lunarmodules.github.io/Penlight/libraries/pl.func`
`extras`	does the above three (`func,stringx,format`); adds some additional functions to `penlight` module; and adds the `pl.tex` sub-module.
`extrasglobals`	does the above `extras` but makes many of the functions global variables as well.

Extras

If `extras` is used, the following Lua globals will be defined:

Misc stuff

`__SKIP_TEX__` If using package with `texlua`, set this global before loading `penlight`
The gloals flags below are taken care of in the package options:
`__PL_GLOBALS__` If using package with `texlua` and you don't want to set some globals (described in next sections), set this global before to `true` loading `penlight`
`__SKIP_LUAKEYS__`
`__PL_NO_HYPERREF__`
`__PL_EXTRAS__` false, 1 or 2

`hasval(x)` Python-like boolean testing
`COMP'xyz'()` Python-like comprehensions:
`https://lunarmodules.github.io/Penlight/libraries/pl.comprehension.html`
`math.mod(n,d)`, `math.mod2(n)` math modulous
`string.totable(s)` string a table of characters
`string.delspace(s)` clear spaces from string
`pl.char(n)` return letter corresponding to 1=a, 2=b, etc.
`pl.Char(n)` return letter corresponding to 1=A, 2=B, etc.
`kpairs(t), npairs(t)` iterate over keys only, or include nil value from table ipairs

`pl.utils.filterfiles(dir,filt,rec)` Get files from dir and apply glob-like filters. Set rec to `true` to include sub directories

`pl.tex.` module is added

`add_bkt_cnt(n), close_bkt_cnt(n), reset_bkt_cnt` functions to keep track of adding curly brackets as strings. `add` will return `n` (default 1) {'s and increment a counter. `close` will return `n` }'s (default will close all brackets) and decrement.

`_NumBkts` internal integer for tracking the number of brackets

`opencmd(cs)` prints `\cs` { and adds to the bracket counters.

`_xNoValue,_xTrue,_xFalse`: `xparse` equivalents for commands

`prt(x),prtn(x)` print without or with a newline at end. Tries to help with special characters or numbers printing.

`prtl(l),prtt(t)` print a literal string, or table

`wrt(x), wrtn(x)` write to log

`help_wrt(s1, s2)` pretty-print something to console. S2 is a flag to help you find.

`prt_array2d(tt)` pretty print a 2d array

`pkgwarn(pkg, msg1, msg2)` throw a package warning

`pkgerror(pkg, msg1, msg2, stop)` throw a package error. If stop is true, immediately ceases compile.

`defcmd(cs, val)` like `\gdef`

`newcmd(cs, val)` like `\newcommand`

`renewcmd(cs, val)` like `\renewcommand`

`prvcmd(cs, val)` like `\providecommand`

`deccmd(cs, dft, overwrite)` declare a command. If `dft` (default) is `nil`, `cs` is set to a package warning saying `'cs' was declared and used in document, but never set`. If `overwrite` is true, it will overwrite an existing command (using `defcmd`), otherwise, it will throw error like `newcmd`.

`get_ref_info(l)` accesses the `\r @label` and returns a table

Macro helpers

`\MakeluastringCommands [def]{spec}` will let `\plluastring (A|B|C..)` be `\luastring (N|O|T|F)` based on the letters that `spec` is set to (or `def` if nothing is provided) This is useful if you want to write a command with flexibility on argument expansion. The

user can specify `n`, `o`, `t`, and `f` (case insensitve) if they want no, once, twice, or full expansion.

Split stuff

Splitting text (or a cmd) into oxford comma format via: `\splitToComma [expansion level]{text}{text to split on}`:

```
-\splitToComma{  j doe  }{\and}-\\
-\splitToComma{  j doe \and s else  }{\and}-\\
-\splitToComma{  j doe \and s else \and a per }{\and}-\\
-\splitToComma{  j doe \and s else \and a per \and f guy}{\and}-

\def\authors{j doe \and s else \and a per \and f guy}
\splitToComma[o]{\authors}{\and}
```

-j doe-
-j doe and s else-
-j doe, s else, and a per-
-j doe, s else, a per, and f guy-
j doe, s else, a per, and f guy

The expansion level is up to two characters, `n|o|t|f`, to control teh expasion of each argument.

```
spliToItems:
\begin{itemize}
  \splitToItems{kale\and john}{\and}
  \splitToItems{kale -john -someone else}{-}
\end{itemize}
```

spliToItems:

- kale
- john
- kale
- john
- someone else

global extras

If `extrasglobals` is used and NOT `extras`, many globals are set.
All `pl.tex` modules are made global.
`hasval`, `COMP`, `kpairs`, `npairs` are globals.

Disclaimer: I am not the author of the Lua Penlight library. Penlight is Copyright ©2009-2016 Steve Donovan, David Manura. The distribution of Penlight used for this library is: `https://github.com/lunarmodules/penlight`
The author of this library has merged all Lua sub-modules into one file for this package.