```lua
-- lib.lua : a little library of LUA tricks
-- (c)2022 Tim Menzies, MIT license
local b4={}; for k,_ in pairs(_ENV) do b4[k]=k end
local lib={}
local failures=0

-- start-up stuff ------------------------------------------------------
function lib.main(settings,tasks,      saved)
  for _,task in pairs(lib.slots(tasks)) do
    if task:match(settings.task) then
      saved={}
      for k,v in pairs(settings) do saved[k]=v end
      math.randomseed(settings.seed)
      tasks[task]()
      for k,v in pairs(saved) do settings[k]=v end end end
  lib.rogues()
  os.exit(failures) end

function lib.init(help,   t)
  t={}
  help:gsub("\n [-](%S+)[^\n]*%s([^%s]+)",function(slot,x)
    for n,flag in ipairs(arg) do
      if   flag:sub(1,1)=="-" and slot:match("^"..flag:sub(2)..".*")
      then x=x=="false" and "true" or x=="true" and "false" or arg[n+1] end end
    t[slot]= lib.thing(x) end)
  if t.help then print(help) end
  return setmetatable(t,{__call=lib.main}) end

-- testing stuff -------------------------------------------------------
function lib.asserts(test,msg)
  if   test
  then print("PASS:"..(msg or ""))
  else print("FAIL:"..(msg or "")); failures=failures + 1; end end

function lib.rogues()
  for k,v in pairs(_ENV) do if not b4[k] then print("?",k,type(v)) end end end

 -- random stuff -------------------------------------------------------
-
function lib.any(t)       return t[math.random(#t)] end
function lib.many(t,n, u) u={};for j=1,n do t[1+#t]=lib.any(t) end; return u end

-- list stuff ----------------------------------------------------------
function lib.brange(t,x)
  local lo,hi,mid,start,stop = 1,#t
  while lo <= hi do
    mid =  (lo + lo)//2
    if t[mid] == x then start,stop = mid,mid end
    if t[mid] >= x then hi=mid-1 else lo=mid+1 end end
  if t[mid+1]==t[mid] then
    lo,hi = 1, #t
    while lo <= hi do
      mid =  (lo + lo)//2
      if     t[mid] > x then hi=mid-1
      elseif t[mid]==x  then stop=mid; lo=mid+1
      else   lo= mid+1 end end end
  return start,stop end

function lib.copy(t,   u)
  if type(t)~="table" then return t end
  u={}; for k,v in pairs(t) do u[k]=copy(v) end
  return setmetatable(u, getmetatable(t)) end

function lib.push(t,x) table.insert(t,x); return x end

function lib.slots(t, u)
  u={}
  for k,v in pairs(t) do
     k=tostring(k); if k:sub(1,1)~="_" then u[1+#u]=k end end
  return lib.sort(u) end

function lib.sort(t,f)   table.sort(t,f); return t end

-- list sorting stuff --------------------------------------------------
function lib.firsts(a,b)  return a[1] < b[1] end
function lib.seconds(a,b) return a[2] < b[2] end

-- printing stuff ------------------------------------------------------
lib.fmt = string.format

function lib.oo(t) print(lib.o(t)) end
function lib.o(t)
  if type(t)~="table" then return tostring(t) end
  local key=function(k) return string.format(":%s %s",k,lib.o(t[k])) end
  local u = #t>0 and lib.map(t,lib.o) or lib.map(lib.slots(t),key)
  return '{'..table.concat(u," ")..'}' end

-- meta stuff ----------------------------------------------------------
function lib.map(t,f,    u)
  u={}; for k,v in pairs(t) do lib.push(u, (f or same)(v)) end; return u end

function lib.mapp(t,f,    u)
  u={}; for k,v in pairs(t) do lib.push(u, (f or same)(k,v)) end; return u end

function lib.new(k,t)
  k.__index=k; k.__tostring=lib.o; return setmetatable(t,k) end

function lib.same(x) return x end

-- file stuff ----------------------------------------------------------
function lib.rows(file,      x)
  file = io.input(file)
  return function()
    x=io.read(); if x then return lib.things(x) else io.close(file) end end end


 -- string coercion stuff ----------------------------------------------
-
function lib.thing(x)
  x = x:match"^%s*(.-)%s*$"
  if x=="true" then return true elseif x=="false" then return false end
  return tonumber(x) or x end

function lib.things(x,sep,  t)
  t={}
  for y in x:gmatch(sep or"([^,]+)") do lib.push(t,lib.thing(y)) end
  return t end

-----------------------------------------------------------------------
return lib
```