```lua
--   __
--  / _|_   _ _ __
-- | |_| | | | '_ \
-- |  _| |_| | | | |
-- |_|  \__,_|_| |_|ctions. My fav LUA tricks. (c)2022 Tim Menzies, MIT license

--- ## Setting up
local b4={}; for k,_ in pairs(_ENV) do b4[k]=k end --used later (to find rogues)
local fun={}      -- code for this module
local failures=0  -- counter for failures (used by fun.asserts and fun.main).

--- ## Start-up
function fun.main(settings,tasks,      saved)
  saved={}
  for k,v in pairs(settings) do saved[k]=v end
  print("FILE "..tostring(arg[0]))
  for _,task in pairs(fun.slots(tasks)) do
    if task:match(settings.task) then
      math.randomseed(settings.seed)
      print("| TASK "..task)
      local ok,msg=pcall(tasks[task])
      if not ok then
        print("|| FAIL "..msg) failures=failures+1
        if settings.Debug then assert(false,msg) end end
      for k,v in pairs(saved) do settings[k]=v end end end
  fun.rogues()
  os.exit(failures) end

function fun.options(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]= fun.thing(x) end)
  if t.help then print(help) end
  return setmetatable(t,{__call=fun.main}) end

--- ## Testing
function fun.asserts(test,msg)
  if   test
  then print("|| PASS "..(msg or ""))
  else print("|| FAIL "..(msg or "")); failures=failures + 1; end end

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

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

--- ## Lists
function fun.bleft(t,x)
  local lo,hi,m,y = 1, #t
  while lo <= hi do
    m = (hi + lo) // 2
    if x<t[m] then hi=m-1 elseif x>t[m] then lo=m+1 else y=m; hi=m-1 end end
  return y or m end

function fun.bright(t,x)
  local lo,hi,m,y = 1, #t
  while lo <= hi do
    m = (hi + lo) // 2
    if x<t[m] then hi=m-1 elseif x>t[m] then lo=m+1 else y=m; lo=m+1 end end
  return y or m end

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

function fun.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 fun.sort(u) end


--- ## List Sorting
function fun.sort(t,f)   table.sort(t,f); return t end
function fun.firsts(a,b)  return a[1] < b[1] end
function fun.seconds(a,b) return a[2] < b[2] end

--- ## Printing
fun.fmt = string.format

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

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

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

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

function fun.same(x) return x end

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

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

--- ## Return
return fun
```