

```

1  --
2  --
3  --
4  --
5  -- [LUA]ctions. My fav LUA tricks. (c)2022 Tim Menzies, MIT license
6
7  local b4={}; for k,_ in pairs(_ENV) do b4[k]=k end
8  local lib={}
9  local failures=0
10
11  -- start-up stuff -----
12  function lib.main(settings,tasks, saved)
13  saved={}
14  for k,v in pairs(settings) do saved[k]=v end
15  for _,task in pairs(lib.slots(tasks)) do
16  if task:match(settings.task) then
17  math.randomseed(settings.seed)
18  local ok,msg=pcall(tasks[task])
19  if not ok then
20  print("FAIL:".msg) failures=failures+1
21  if settings.Debug then assert(false,msg) end end
22  for k,v in pairs(saved) do settings[k]=v end end end
23  lib.rogues()
24  os.exit(failures) end
25
26  function lib.options(help, t)
27  t={}
28  help:gsub("\n [^(%s+)]^%s(%s+)",function(slot,x)
29  for n,flag in ipairs(arg) do
30  if flag:sub(1,1)=="-" and slot:match("^"..flag:sub(2)..".*")
31  then x=x.."false" and "true" or x=="true" and "false" or arg[n+1] end end
32  t[slot]= lib.thing(x) end)
33  if t.help then print(help) end
34  return setmetatable(t,{__call=lib.main}) end
35
36  -- testing stuff -----
37  function lib.asserts(test,msg)
38  if test
39  then print("PASS:".msg or "")
40  else print("FAIL:".msg or ""); failures=failures + 1; end end
41
42  function lib.rogues()
43  for k,v in pairs(_ENV) do if not b4[k] then print("?",k,type(v)) end end end
44
45  -- random stuff -----
46  function lib.any(t) return t[math.random(#t)] end
47  function lib.many(t,n, u) u={};for j=1,n do t[1+#t]=lib.any(t) end; return u end
48
49  -- list stuff -----
50  function lib.brange(t,x)
51  local lo,hi,mid,start,stop = 1,#t
52  while lo <= hi do
53  mid = (lo + hi)//2
54  if t[mid] == x then start,stop = mid,mid end
55  if t[mid] >= x then hi=mid-1 else lo=mid+1 end end
56  if t[start+1]==t[start] then
57  lo,hi = stop, #t
58  while lo <= hi do
59  mid = (lo + hi)//2
60  if t[mid] > x then hi=mid-1 else stop=mid; lo=mid+1 end end end
61  return start,stop end
62
63  function lib.support(t,x,y)
64  if x < t[1] then x0,x1 = 1,1 else x0,x1 = lib.brange(t,x) end
65  if y > t[#t] then y0,y1 = #t,#t else y0,y1 = lib.brange(t,y) end
66  return (1 + y1-x0) end
67
68  function lib.copy(t, u)
69  if type(t)~="table" then return t end
70  u={}; for k,v in pairs(t) do u[k]=copy(v) end
71  return setmetatable(u, getmetatable(t)) end
72
73  function lib.push(t,x) table.insert(t,x); return x end
74
75  function lib.slots(t, u)
76  u={}
77  for k,v in pairs(t) do
78  k=tostring(k); if k:sub(1,1)~="_" then u[1+#u]=k end end
79  return lib.sort(u) end
80
81  function lib.sort(t,f) table.sort(t,f); return t end
82
83  -- list sorting stuff -----
84  function lib.firsts(a,b) return a[1] < b[1] end
85  function lib.seconds(a,b) return a[2] < b[2] end
86
87  -- printing stuff -----
88  lib.fmt = string.format
89
90  function lib.o(t) print(lib.o(t)) end
91  function lib.o(t)
92  if type(t)~="table" then return tostring(t) end
93  local key=function(k) return string.format("%s%s",k,lib.o(t[k])) end
94  local u = #t>0 and lib.map(t,lib.o) or lib.map(lib.slots(t),key)
95  return '{ '..table.concat(u," ")}' end
96
97  -- meta stuff -----
98  function lib.map(t,f, u)
99  u={}; for k,v in pairs(t) do lib.push(u, (f or same)(v)) end; return u end
100
101  function lib.mapp(t,f, u)
102  u={}; for k,v in pairs(t) do lib.push(u, (f or same)(k,v)) end; return u end
103
104  function lib.new(k,t)
105  k.__index=k; k.__tostring=lib.o; return setmetatable(t,k) end
106
107  function lib.same(x) return x end
108
109  -- file stuff -----
110  function lib.rows(file, x)
111  file = io.input(file)
112  return function()
113  x=io.read(); if x then return lib.things(x) else io.close(file) end end end
114
115  -- string coercion stuff -----
116  function lib.thing(x)
117  x = x:match("^%s*(-)%s*$")
118  if x=="true" then return true elseif x=="false" then return false end
119  return tonumber(x) or x end
120
121  function lib.things(x,sep, t)
122  t={}
123  for y in x:gmatch(sep or "[^,]+") do lib.push(t,lib.thing(y)) end
124  return t end
125
126  -----
127  return lib

```