```lua
#!/usr/bin/env lua
local b4={}; for k,_ in pairs(_ENV) do b4[k]=k end --used later (to find rogues)
local azzert,big,cli,fails,fmt,goalp,help,ignorep,klassp
local lessp,map,main,max,min,morep
local new,nump,o,oo,push,r,rows,slots,sort,sum,the,thing,things
local COLS, NUM, ROWS, SKIP, SOME, SYM = {},{},{},{},{},{}
------------------------------------------------------------------------------
function cli(want,x)
  for n,got in ipairs(arg) do if got==want then
    x = x==false and true or x==true and "false" or arg[n+1] end end
  if x=="false" then return false else return tonumber(x) or x end end

help = [[

./sl.lua [OPTIONS]

OPTIONS:
  -D       stack dump on assert fails    = false
  -d   F   data file                     = etc/data/auto93.csv
  -h       show help                     = false
  -k   P   max kept items                = 256
  -S   P   set seed                      = 10019
  -t   S   start up action (all= do all) = nothing

KEY: F=filename P=posint S=string
]]

the = {dump = cli("-D", false),
       data = cli("-d", "../etc/data/auto93.csv"),
       help = cli("-h", false),
       keep = cli("-k", 256  ),
       seed = cli("-S", 10019),
       todo = cli("-t", "nothing")}
------------------------------------------------------------------------------
-- ___ ____ ____ _    ____
--  |  |  | |  | |    [__
--  |  |__| |__| |___ ___]
--
--- strings
fmt = string.format

--- maths
big = math.huge
max = math.max
min = math.min
r   = math.random

--- column headers
function klassp(x)  return x:find"!$"     end
function lessp(x)   return x:find"-$"     end
function morep(x)   return x:find"+$"     end
function nump(x)    return x:find"^[A-Z]" end
function ignorep(x) return x:find":$"     end
function goalp(x)   return morep(x) or lessp(x) or klassp(x) end

--- tables
function push(t,x) table.insert(t,x); return x end
function sort(t,f) table.sort(t,f); return t end

--- meta
function new(k,t)  k.__index=k; k.__tostring=o; return setmetatable(t,k) end
function map(t,f, u) u={};for k,v in pairs(t) do push(u,f(v)) end; return u end
function sum(t,f, n) n=0; for _,v in pairs(t) do n=n+f(v)     end; return n end
function slots(t, u)
  u={}
  for k,v in pairs(t) do k=tostring(k);if k:sub(1,1)~="_" then push(u,k) end end
  return sort(u) end

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

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

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

--- errors
fails=0
function azzert(test, msg)
  print(test and "PASS: "or "FAIL: ",msg or "")
  if not test then
    fails=fails+1
    if the.dump then assert(test,msg) end end end
-- ____ ____ _  _ ____
-- [__  |  | |\/| |___
-- ___] |__| |  | |___
--
function SOME.new(k,keep) return new(k,{n=0,_all={}, keep=keep or the.keep}) end
function SOME.add(i,x)
  i.n = i.n+1
  if     #i._all < i.keep then push(i._all,x)          ; return i._all
  elseif r()     < i.keep/i.n then i._all[r(#i._all)]=x; return i._all end end
-- ____ _  _ _ ___
-- [__  |_/  | |__]
-- ___] | \_ | |
--
function SKIP.new(k,n,s) return new(k,{n=0,at=at or 0,txt=s or""}) end
function SKIP.add(i,x)   return x end
-- ____ _   _ _  _
-- [__   \_/  |\/|
-- ___]   |   |  |
--
function SYM.new(k,n,s) return new(k,{n=0,at=n or 0,txt=s or"",has={}}) end
function SYM.add(i,x,inc)
  if x ~= "?" then
    inc = inc or 1
    i.n = i.n + inc
    i.has[x] = inc + (i.has[x] or 0) end end
function SYM.dist(i,x,y)
  return  (x=="?" and y=="?" and 1) or (x==y and 0 or 1) end
-- _  _ _  _ _  _
-- |\ | |  | |\/|
-- | \| |__| |  |
--
function NUM.new(k,n,s)
  return new(k,{n=0,at=n or 0,txt=s or"",has=SOME:new(),
                w=lessp(s or "") and -1 or 1, lo=big, hi=-big}) end
function NUM.add(i,x)
  if x ~= "?" then
    i.n = i.n + 1
    i.has:add(x); i.lo,i.hi = min(x,i.lo), max(x,i.hi); end end
function NUM.norm(i,x)
  return math.abs(i.hi-i.lo)<1E-9 and 0 or (x-i.lo)/(i.hi - i.lo) end
function NUM.dist(i,x,y)
  if x=="?" and y=="?" then return 1
  elseif x=="?" then y=i.norm(y); x=y<0.5 and 1 or 0
  elseif y=="?" then x=i.norm(x); y=x<0.5 and 1 or 0
  else   x,y = i.norm(x), i.norm(y) end
  return math.abs(x-y) end
-- ____ ____ _    ____
-- |    |  | |    [__
-- |___ |__| |___ ___]
--
function COLS.new(k,row,   i)
  i= new(k,{all={},x={},y={}})
  for at,txt in ipairs(row) do  push(i.all, i:col(at,txt)) end
  return i end
function COLS.add(i,t)
  for _,col in pairs(i.all) do col:add( t[col.at] ) end
  return t end
function COLS.col(i,at,txt,     col)
  if ignorep(txt) then return SKIP:new(at,txt) end
  col = (nump(txt) and NUM or SYM):new(at,txt)
  push(goalp(txt) and i.y or i.x, col)
  if klassp(txt) then i.klass = col end
  return col end
-- ____ ____ _ _ _ ____
-- |__/ |  | | | | [__
-- |  \ |__| |_|_| ___]
--
function ROWS.new(k,inits,     i)
  i = new(k,{rows=SOME:new(), cols=nil})
  if type(inits)=="string" then for row in rows(inits) do i:add(row) end end
  if type(inits)=="table"  then for row in inits       do i:add(row) end end
  return i end
function ROWS.add(i,row)
  if   i.cols then i.rows:add( i.cols:add(row) )
  else i.cols = COLS:new(row) end end
function ROWS.dist(i,row1,row2,   d)
  function d(col) return col:dist(row1[col.at], row2[col.at])^the.p end
  return (sum(i.cols.x, d)/ #i.cols.x)^(1/the.p) end
------------------------------------------------------------------------------
-- ___  ____ _  _ ____ ____
-- |  \ |___ |\/| |  | [__
-- |__/ |___ |  | |__| ___]
--
local egs={}
function egs.nothing() return true end
function egs.the()     oo(the) end
function egs.rand()    print(r()) end
function egs.f1() print(1) end
function egs.f2() print(2) end

if the.help then print(help) else
  local b4={}; for k,v in pairs(the) do b4[k]=v end
  for _,todo in pairs(the.todo=="all" and slots(egs) or {the.todo}) do
    for k,v in pairs(b4) do the[k]=v end
    math.randomseed(the.seed)
    if type(egs[todo])=="function" then egs[todo]() end end end

for k,v in pairs(_ENV) do if not b4[k] then print("?",k,type(v)) end end
os.exit(fails)
```