Matlab Code Listing – Color

```
function russell_demo()
% Do the example in ch 17 (p501) of Russell and Norvig
% (1,1) is top left corner.
r = 3; c = 4; p = 0.8; action_cost = -1/25;
obstacle = zeros(r,c); obstacle(2,2)=1;
terminal = zeros(r,c); terminal(1,4)=1; terminal(2,4)=1;
absorb = 1;
wrap_around = 0;
noop = 0;
T = mk_grid_world(r, c, p, obstacle, terminal, absorb, wrap_around, noop);
% Add rewards for terminal states
nstates = r*c + 1;
if noop
  nact = 5;
else
  nact = 4;
end
R = action_cost*ones(nstates, nact);
R(10,:)  = 1;
R(11,:)  = -1;
R(nstates,:)  = 0;
discount_factor = 1;

V = value_iteration(T, R, discount_factor);
%reshape(V(1:end-1),[r c])
%    0.8116    0.8678    0.9178    1.0000
%    0.7616    0.7964    0.6603   -1.0000
%    0.7053    0.6553    0.6114    0.3878
% Same as the book p501

Q = Q_from_V(V, T, R, discount_factor);
[V, p] = max(Q, [], 2);


use_val_iter = 1;
% (I-gT) is singular since g=1 and there is an absorbing state (i.e., T(i,i)=1)
% Hence we cannot use value determination.
[p,V] = policy_iteration(T, R, discount_factor, use_val_iter);

%reshape(V(1:end-1),[r c])
%    0.8115    0.8678    0.9178    1.0000
%    0.7615    0.7964    0.6603   -1.0000
%    0.7048    0.6539    0.6085    0.3824
```

Lua Code Listing – Black and White

```
-- version   :  1.0.0 - 07/2005
-- author    :  Hans Hagen - PRAGMA ADE - www.pragma-ade.com
-- copyright :  public domain or whatever suits
-- remark    :  part of the context distribution

-- TODO:  name space for local functions

-- loading:  scite-ctx.properties

-- generic functions

local crlf = "\n"

function traceln(str)
    trace(str ..  crlf)
    io.flush()
end

table.len  = table.getn
table.join = table.concat

function table.found(tab, str)
    local l, r, p
    if string.len(str) == 0 then
        return false
    else
        l, r = 1, table.len(tab)
        while l <= r do
            p = math.floor((l+r)/2)
            if str < tab[p] then
                r = p - 1
            elseif str > tab[p] then
                l = p + 1
            else
                return true
            end
        end
        return false
    end
end

function string.grab(str, delimiter)
    local list = {}
    for snippet in string.gfind(str,delimiter) do
        table.insert(list, snippet)
    end
```

```
    return list
end

function string.join(list, delimiter)
    local size, str = table.len(list), ''
    if size > 0 then
        str = list[1]
        for i = 2, size, 1 do
            str = str ..  delimiter ..  list[i]
        end
    end
    return str
end

function string.spacy(str)
    if string.find(str,"^%s*$") then
        return true
    else
        return false
    end
end

function string.alphacmp(a,b,i) -- slow but ok
    if i and i > 0 then
        return string.lower(string.gsub(string.sub(a,i),'0',' ')) < string.lower(string.gsub(string.s
'))
    else
        return string.lower(a) < string.lower(b)
    end
end

function table.alphasort(list,i)
    table.sort(list, function(a,b) return string.alphacmp(a,b,i) end)
end

function io.exists(filename)
    local ok, result, message = pcall(io.open,filename)
    if result then
        io.close(result)
        return true
    else
        return false
    end
end

function os.envvar(str)
    if os.getenv(str) ~= '' then
        return os.getenv(str)
    elseif os.getenv(string.upper(str)) ~= '' then
```

```
        return os.getenv(string.upper(str))
    elseif os.getenv(string.lower(str)) ~= '' then
        return os.getenv(string.lower(str))
    else
        return ''
    end
end

function string.expand(str)
    return string.gsub(str, "ENV%((%w+)%)", os.envvar)
end

function string.strip(str)
    return string.gsub(string.gsub(str,"^%s+",''),"%s+$",'')
end

function string.replace(original,pattern,replacement)
    local str = string.gsub(original,pattern,replacement)
--     print(str) -- indirect, since else str + nofsubs
    return str -- indirect, since else str + nofsubs
end
```