How to write good code
Part 1

BNA SIG EVENT: NEUROINFORMATICS TUTORIAL: WHAT MAKES GOOD CODE?, LONDON

BNA Event - 5th Apr 2018

Sanjay Manohar, Oxford

Content

Part 1:

- Naming Conventions
- Code Layout
- Commenting

Part 2:

- Conceptualisation
- Externalisation
- Functionising

Part 3:

- Debugging
- Testing

“Good code is a present for your future self”

Naming Conventions

- Depends on personal style – no set rule

- Long names
  - Self-explanatory
  - But take up more space
- Short names
  - Need comments
  - Easier to get mixed up
  - But more concise

Long Names

```
%locomotion/VR location
loco_ch = cell2mat(findFolders(exp_dir, '*ch_3.tif'));
%vis stim location
stim_ch = cell2mat(findFolders(exp_dir, '*ch_4.tif'));

%get image info (from vessel tif file)
info = imfinfo(vessel_ch);
width = info(1).Width;     %width of frame (pixels)
height = info(1).Height;   %height of frame (pixels)
num_frames = numel(info); %number of frames (frames)

%Initialize an empty array to store the tiff file in
rawIm = zeros(num_frames,width,height); %vessel channel
%only need to do for calc if taking raw tif file
if isempty(prefs.calc)
    calcium = zeros(1,num_frames); %calc channel
end
movement = zeros(1,num_frames); %locomotion channel
stim = zeros(1,num_frames); %vis stim channel
```

Short Names

```
% construct Ff and Fneu
Ff = [];
Fneu = [];
for j = 1:numel(Fcell)
    Ff   = cat(1, Ff, Fcell{j}');
    Fneu = cat(1,Fneu, FcellNeu{j}');
end
```

Code Layout

- Use white space
- Indent your code (*Python is different*)
- Meaningful paragraphs
- Prevent crowding (too many computations on one line)
- Character limit line (80 characters)

Formatting before

```
folder = cd;
placetiff = 'PC_comb.tif';
cd(folder);

    fn = fullfile(folder, placetiff);
    info = imfinfo(fn);
    num_images = numel(info);
    flims = [1,num_images];
    [mixedsig, mixedfilters, CovEvals, covtrace, movm, movtm] = CellsortPCA(fn, flims);

PCuse = [1:300];
CellsortPlotPCspectrum(fn, CovEvals, PCuse) ;
mu =0.2;
[ica_sig, ica_filters, ica_A, numiter] = CellsortICA(mixedsig, mixedfilters,CovEvals, PCuse, mu) ;


smwidth  =  3;    % std of Gaussian smoothing kernel (pixels)
thresh   =  3;    % threshold for spatial filters (standard deviations)
arealims =  [15]; % int vector, 1st number is the minimum ROI size, 2nd
                  % number (if present) the maximum ROI size (pixels)


[ica_segments, segmentlabel, segcentroid] = CellsortSegmentation(ica_filters, smwidth, thresh, arealims) ;
cell_sig = CellsortApplyFilter(fn, ica_segments, flims) ;
thresh = 0.5;
dt =1;
deconvtau = 0;
normalization = 1;
[spmat, spt, spc] = CellsortFindspikes(ica_sig, thresh, dt, deconvtau, normalization) ;
```

Formatting after

```
% Data variables
placetiff       =  'PC_comb.tif';
folder          =  cd;
fn              =  fullfile(folder, placetiff);
info            =  imfinfo(fn);
num_images      =  numel(info);

% CellSort inputs
smwidth         =  3;    % std of Gaussian smoothing kernel (pixels)
thresh          =  3;    % threshold for spatial filters (stds)
arealims        =  [15]; % int vector, 1st number is the minimum ROI size,
                         % 2nd number (if present) the maximum ROI size (pixels)
PCuse           =  [1:300];
mu              =  0.2;
flims           =  [1,num_images];
thresh          =  0.5;
dt              =  1;
deconvtau       =  0;
normalization   =  1;

cd(folder);

% CellSort steps
[mixedsig, mixedfilters, CovEvals, covtrace, movm, movtm] = ...
    CellsortPCA(fn, flims);
if false
    CellsortPlotPCspectrum(fn, CovEvals, PCuse) ;
end
[ica_sig, ica_filters, ica_A, numiter] = ...
    CellsortICA(mixedsig, mixedfilters,CovEvals, PCuse, mu) ;
[ica_segments, segmentlabel, segcentroid] = ...
    CellsortSegmentation(ica_filters, smwidth, thresh, arealims) ;
cell_sig = ...
    CellsortApplyFilter(fn, ica_segments, flims) ;
[spmat, spt, spc] = ...
    CellsortFindspikes(ica_sig, thresh, dt, deconvtau, normalization) ;
```

Commenting

- Need to inform reader of aim, and for functions the input and output
- Declaration of variable -> show what the variable is
- Comment for-loops to make them clearer
- Add author, version, date

- Don’t use comments to get rid of code you don’t want to run!
  - ‘If out’ instead!

Need to inform reader of aim, inputs and outputs

Header before

```
%Todo: Automatically make locomotion file - DONE
% Fix fluctuation detection locomotion
% Automatically select cells that are place cells  - DONE
% Overlay ROIs on image
% Visualize results
% Make masterfile a function with variables.

% folder = 'C:\Users\dorieke\Dropbox (Brain Energy Lab)\Everything\Dori\Data\Simon\2017_11_10\20171110_13_49_05_Env1_1526Hz';
% folder = 'C:\Users\dorieke\Dropbox (Brain Energy Lab)\Everything\Dori\Data\Maggie\2017_09_15\20170915_15_24_35_PC1_12Hz';
folder = cd;
% folder = 'C:\Users\User\Dropbox (Brain Energy Lab)\Everything\Dori\Data\Simon\2017_11_10\20171110_13_55_05_Env2_1526Hz';
% placetiff = 'placecellsCropSub.tif';
placetiff = 'PC_comb.tif';
cd(folder);
```

Header after

```
% Aim:      Extract ROIs from a preprocessed image and determine the
%           calcium traces per ROI
% Input:    Tiff-file containing preprocessed stack of calcium imaging
% Output:   MAT-file containing the deltaF/F calcium traces for each ROI


folder = cd;
placetiff = 'PC_comb.tif';
cd(folder);
```

Declaration of variable -> show what the variable is

Declaring variables before

```
smwidth = 3;
thresh = 3;
arealims =[15];
```

Declaring variables after

```
smwidth   =   3;     % std of Gaussian smoothing kernel (pixels)
thresh    =   3;     % threshold for spatial filters (standard deviations)
arealims  =   [15]; % int vector, 1st number is the minimum ROI size, 2nd
                     % number (if present) the maximum ROI size (pixels)
```

Don’t use comments to get rid of code you don’t want to run, use if false

```
%                       %plot original contour line in black,
%                       %with newpolyfit line in red
%                       figure; title('orig line, and poly line');
%                       plot(x,y,'k');
%                       hold on;
%                       plot(xl,yl,'r');
%                       legend('original','poly');
%
%                       %check polyfit line fused over the full skeleton
%                       %can see if it is in the right place before find
%                       %normal
%                       figure; title('skeleton with fitted line');
%                       imshow(imfuse(skeleton,squeeze(rawIm(i,:,:))));
%                       hold on;
%                       plot(xl,yl,'r')
```

```
%only plot when debugging, else too many figs!
if false

    %plot original contour line in black,
    %with newpolyfit line in red
    figure; title('orig line, and poly line');
    plot(x,y,'k');
    hold on;
    plot(xl,yl,'r');
    legend('original','poly');

    %check polyfit line fused over the full skeleton
    %can see if it is in the right place before find
    %normal
    figure; title('skeleton with fitted line');
    imshow(imfuse(skeleton,squeeze(rawIm(i,:,:))));
    hold on;
    plot(xl,yl,'r')

end
```

Comment for loops

Ending for-loops before

```
    for j = fliplr(1:w-1)
        X_ttt = squeeze(X_train(:, j, :));
        L1 = L1_values(:,:,j+1);
        L1_prev = L1_values(:,:,j);
        L1_error = L2_delta * W2';
        L1_delta = fut_L1_delta*WH/100;

        W1_update = W1_update + X_ttt*L1_delta;
        WH_update = WH_update + L1_prev'*L1_delta;

        fut_L1_delta = L1_delta;
    end

    W1 = W1+ theta*W1_update;
    W2 = W2+ theta*W2_update;
    WH = WH+ theta*WH_update;

    perf = corrcoef(L2, y_train);
    curr_err = perf(1,2);
    errdiff = curr_err - prev_err;
    prev_err = curr_err;

    i = i+1;
end
end
```

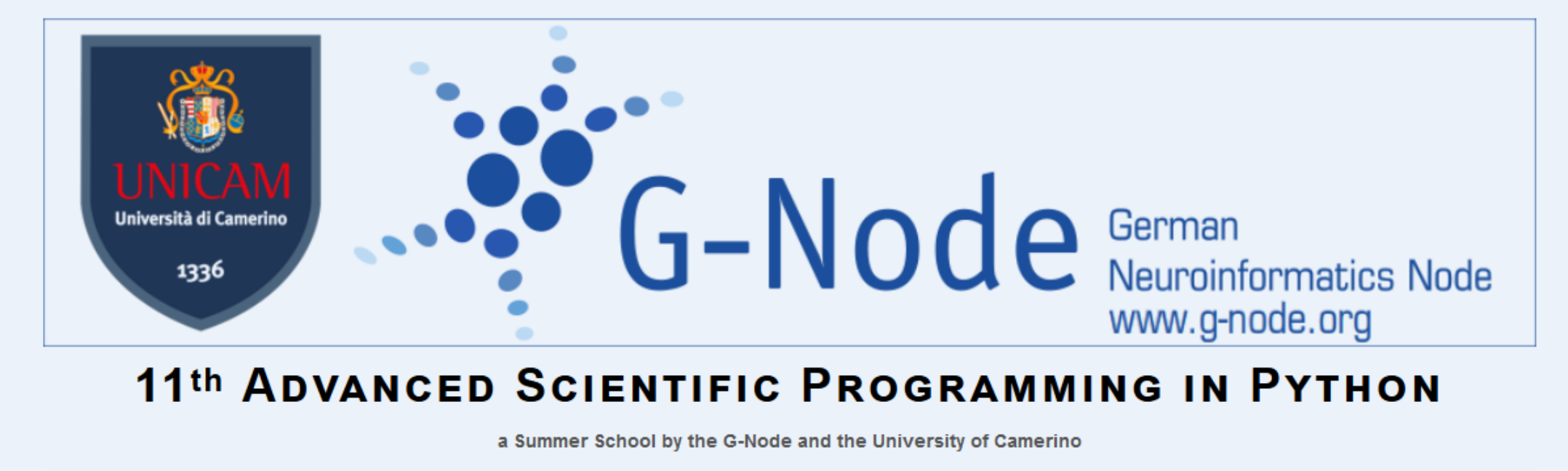
Ending for-loops after

```
    for j = fliplr(1:w-1)
        %loop through all time windows in reverse order
        X_ttt = squeeze(X_train(:, j, :));
        L1 = L1_values(:,:,j+1);
        L1_prev = L1_values(:,:,j);
        L1_error = L2_delta * W2';
        L1_delta = fut_L1_delta*WH/100;

        W1_update = W1_update + X_ttt*L1_delta;
        WH_update = WH_update + L1_prev'*L1_delta;

        fut_L1_delta = L1_delta;
    end % time windows end

    W1 = W1+ theta*W1_update;
    W2 = W2+ theta*W2_update;
    WH = WH+ theta*WH_update;

    perf = corrcoef(L2, y_train);
    curr_err = perf(1,2);
    errdiff = curr_err - prev_err;
    prev_err = curr_err;

    i = i+1;
end % iterations end
end % cross validation folds end
```

Add author, version, date

```
function xyFWHM(fname, vesselCh, VRflag, prefs, optionsFlag)

%function created by Kira, Dec 2017, updated April 2018
%
%loads in vessel.tif from inputted folder (will error if tif doesn't exist)
%finds diameter changes using FWHM along vessel (for every frame/branch)
%requires manual user input to draw mask around each of branches
%(i.e. draw mask around vessel, with little-no background signal within mask
%
%INPUTS-
%fname       : folder name with e.g. vessel.tif file(s) - can be a top dir
%vesselCh    : green (e.g.FITC) = 1, red (e.g.TR) = 2; default 2
%VRflag      : no VR = 0, VR environment on = 1; default 0
%prefs       : structure which contains calc and vessel names
%if user wants to edit, can send in own prefs
%default - prefs.vessel='vessel.tif',prefs.calc='cell_sig.mat'
%optionsFlag : 0 means uses automatic settings, 1 means need to send own
%options in via prefs - NB all options need to be covered, default = 0, see
%further down code for options to use
%
%OUTPUTS-
%no vars outputted from the function, but it will save some figures and
%mat files into the experimental directories with tif file in
%will save continuous time series with vessel diameter, loco trace,
%vis stim, etc., a video to show scan, and a fig to show cont traces
```

Future Events:



https://python.g-node.org/wiki/