Variable naming convention

For discussion but to be determined/frozen soon.

<name>: all lower case
<variableName>: lower case first letter; upper case subsequent words, no underscores; first word to be at least 2 letters long
For the variables below ?<Name> could be replaced by ?<VariableName>
b<Name>: boolean (ie flag); not to be used for other kinds of variables
i<Name>: variable is a loop counter over <Name> objects; <Name> is compulsory; not to be used for other kinds of variables
n<Name>: variable is a total count of <Name> objects; <Name> is compulsory; not to be used for other kinds of variables
<NAME>: 'variable' is a constant (value not to be changed!). [All global parameters are stored in a dictionary of parameters — this is for local use.]
No non-constant variable to consist of entirely upper case letters.

class_<Name>: a python class definition for <Name> objects
dict_<anyOfAbove>: a python dictionary
list_<anyOfAbove>: a python list
tuple_<anyOfAbove>: a python tuple
etc.

<function_name>: all lower case; words separated by underscores; at least 2 words. Not to start with one of the above python keywords corresponding to different types of python object. [Danger of this is that might add extra python objects to the above list at a later date, but probability of this causing a problem is very low.] f<Name> would be an alternative convention but that just looks odd.

Structure of L-Galaxies

main.c

```
read_parameter_file(argv[1])
check_options( )                                                      # Checks consistency of selected options
init( )                                                               # Loads in tables of physical parameters: cooling rates; feedback tables, etc
write_sfh_bins( )                                                     # Writes file of time bins used by star-formation history
Loop over files:
    load_tree_table(filenr)
    Loop over trees:
        Loop over halos:                                              # Done in a complicated way
            construct_galaxies(filenr, treenr, halonr)
        output_galaxy(treenr,0)                                       # Outputs “remaining” galaxies?
    free_galaxies_and_tree()
close_galaxy_files()

construct_galaxies(filenr, treenr, halonr)                            # Loops over galaxies in a complicated, recursive way
    ngal = join_galaxies_of_progenitors(fofhalo, ngal, centralgal)    # Collects together the progenitor galaxies; identifies mergers
    # Finds the central galaxy for that halo and the FOF halo
    evolve_galaxies(halonr, ngal, treenr)                             # This routine does the SAM

evolve_galaxies(halonr, ngal, treenr)
    sfh_update_bins(…)                                                # Update each galaxies SFH bins
    deal_with_satellites(ngal)                                        # Stripping of gas from satellites
    infallingGas = infall_recipe(ngal)                                # How much gas needs to infall
    Loop over mini-timesteps:
        # All of the following loop over galaxies, where appropriate
        add_infall_to_hot(…)                                          # Accrete onto central galaxy
        reincorporate_gas(…)                                          # From Ejected phase back onto Type 0 & 1
        compute_cooling(…)
        do_AGN_heating(…)
        cool_gas_onto_galaxy(…)                                       # From HotGas to ColdGas
        starformation(…)
        deal_with_galaxy_merger(…)                                    # If merger timescale drops below zero.
        Grow BHs
        update_yields_and_return_mass(…)                              # If not instantaneous
        disrupt(…)                                                    # If host density exceeds that of satellite
        output_galaxy(…)                                              # Outputs progenitor galaxies and frees storage (this is recursive/convoluted)
        update_type_two_coordinate_and_velocity(…)
```

Structure of py-lgal

L-Galaxies.ipynb

```
# Import modules
# Read in parameter file (into Dictionary)
# Define halo class (depends upon runtime parameters);
# Define structured array of galaxy properties (depends upon runtime parameters)
# Define internal functions (I/O, graph tracing, etc)
(Loop over files)                               # Let’s stick to a single file for now
Loop over graphs:
    Loop over snapshots:                        # Past to present
        Loop over halos:
            process_halo( )
    write_to_HDF5( )                            # Copy halo and galaxy output properties to HDF5 datasets.
# Tidy up and exit                              # Write remaining output buffers; close everything neatly.

processhalo:
    initialise_halo( )                          # Here, or all at once at the beginning? Reads properties; calculates quantities
    gather_progenitors_halo( )                  # Inherit components from progenitors
    Loop over galaxies:
        gather_progenitors_galaxy( )
    calculate_infall( )                         # New gas accreted to make up baryon deficit
    Loop over mini time steps:
        cool_onto_central( )                    # If there is a central
        Loop over galaxies:
            update_pos_and_merge_type2( )       # ***Type 2 galaxies are not associated with a sub halo***
            disrupt_and_strip( )                # Moves to start of loop
            reincorporate_gas( )                # Ejected back into Hot
            grow_BH( )                          # This should surely be here
            cool_gas( )                         # Hot to Cold; combining cooling + AGN heating
            form_stars( )                       # Includes feedback
    output_halos( )                             # Set halo properties for output later
    output_galaxies( )                          # Ditto for galaxies
```

Building up py-lgal step by step
Step 1

L-Galaxies.ipynb

```
# Import modules
# Read in parameter file (into Dictionary)
# Define halo class (depends upon runtime parameters);
# Define structured array of galaxy properties (depends upon runtime parameters)
# Define internal functions (I/O, graph tracing, etc)
Loop over graphs:
    Loop over snapshots:                        # Past to present
        Loop over halos:
            process_halo( )
    write_to_HDF5( )                            # Copy halo and galaxy output properties to HDF5 datasets.
# Tidy up and exit                              # Write remaining output buffers; close everything neatly.


processhalo:
    initialise_halo( )                          # Here, or all at once at the beginning? Reads properties; calculates quantities
    gather_progenitors_halo( )                  # Inherit components from progenitors
    Loop over galaxies:
        gather_progenitors_galaxy( )
    calculate_infall( )                         # New gas accreted to make up baryon deficit
    if central galaxy exists:
        star_formation_from_SHMR                # Toy model using star-halo mass ratio
    output_halo( )                              # Set halo properties for output later
    output_galaxies( )                          # Ditto for galaxies
```

Comments on individual routines

class haloProperties:

initialiseHalo:

```
# Construct an instance of haloClass
# Loop over galaxies to determine which, if any, is close to the dynamical centre of the halo.
```

gather_progenitors_halo:

```
Loop over halo progenitors:
    # Accumulate DM mass (perhaps not needed/interesting, but why not)
    # Accumulate baryons (whatever we have in halo class; in first instance simply Gas)
```

gather_progenitors_galaxy:

```
Loop over galaxy progenitors:
    # Accumulate DM mass (perhaps not needed/interesting, but why not)
    # Accumulate baryons (whatever properties we have in the structured array.
    # Note that, in the first instance at least, we only accumulate baryons if we are the first (main) descendent.
    # Add total baryon mass to that of the host halo.
    # Add total stellar mass to that of the host halo.
```

calculate_infall:

```
massBaryon_old=massBaryon
massBaryon=max(fBaryon*mass, massBaryon)
massBaryon_delta=massBaryon-massBaryon_old
```

star_formation_from_SHMR:

```
# Look up expected stellar mass from Behroozi etal 2013, http://arxiv.org/abs/1207.6105
massStars_old=massStars
massStars=max(massStars_Behroozi, massStars)
massStars_delta=massStars-massStars_old
SFR=massStars_delta/time_delta                  # time_delta is size of timestep (difference in time of snapshots)
# Add massStars_delta to central galaxy
# Update SFR of central galaxy
```

output_halo:

```
# Set values of any halo properties that we wish to output (in structured array haloOutput)
```

output_galaxies:

```
# Set values of any galaxy properties that we wish to output (in structured array galaxyOutput)
```

write_to_HDF5:

```
# Update structured array of halo properties that we want to output
if haloOutput buffer full:
    # write haloOutputBuffer to HDF5 halo dataset
do while galaxyOuput array not empty:
    # Add as much as galaxyOutput array as we can to HDF5 galaxyOutputBuffer
    if galaxyOutputBuffer full:
        # write galaxyOutputBuffer to HDF5 galaxy dataset
```

Tidy up and exit:

```
# Flush remaining HDF5 datasets
# Close HDF5 output files
# Write final diagnostics and close log files
```