**PyFIREBALL Beta.**

***The DOS class****.*

Originally conceived as a once-off, the DOS class is uniquely powerful for “specialist” DOS and PDOS plots. The class has been altered considerably from how it is was first implemented, where a list was used for contributions. Older methods from the first implementation are currently in PyFIREBALL.py, but will be deprecated and finally removed in due course.

A DOS object is simple enough, it contains a HOMO variable, a dictionary of “contributions”, an energies and a totals column. The advantage is that with this strong-enough-but-not-terribly-strong data paradigm, we can hardcode many of the plotting functions we require and call them regularly without starting from the beginning.

A DOS object can be initialized in a number of ways, the initializer method requires the energies passed as a list, and the contributions passed as a dictionary object. There is the option to pass a HOMO, and to surpress the calculation of a “TOTALS” column. There is also a convenience method called “readDOS” which will read in all densities into a supercell object or atomList.

DOS objects can be added together with the “+” operator, which will sum the contributions so long as the energies and the contributions align perfectly and fail otherwise.

*immutable variables:*

NONE

*mutable variables:*

.energies -> A list of the energies this DOS object is calculated for.

.Totals -> Sum of all contributions in a list.

.contributions -> a dictionary object of each of the contributions to this DOS, the key to each of the entries is the title of that contribution. For example, if this is the DOS object of a supercell, the contributions may be each species in the cell, or if this DOS object is a single atom, or group of similar atoms, the contributions will be the contributions by shell. In this way, plotting the PDOS of this DOS object allows for individual contributions in the plot with names in the legend for each one.

*DOS internal methods:*

.offset(offsetVal) -> Returns a new DOS object with the energies offset by - the value of offsetVal

. plotme(name = "DOSplot", printToFile = True, legend = True, plotContributions = True, plotHOMO = True, plotOffset = False, xlim = [], plotTOT = True, multiHOMO = {}, contriblw = 1, totlw = 2, legendLoc = 'best', returnAxes = False) -> Renders a plot of the calling DOS object to the screen and to file. The optional arguments are as follows, with default values:

name = "DOSplot" -> Filename for outputted plot in .png format

printToFile = True -> Flag to print to a file

legend = True -> Display legend toggle

plotContributions = True -> Toggle to plot the contributions in the DOS object.

plotHOMO = True -> Toggle to plot the vertical HOMO line

plotOffset = False -> Toggle to offset the plot so as HOMO = 0.0eV

xlim = [] -> Toggle to limit the x-axis plot

plotTOT = True -> Toggle to plot the Totals

multiHOMO = {} -> Dictionary object if each contribution contains a different HOMO

contriblw = 1 -> Line width of the contributions plot

totlw = 2 -> Line width of the Totals plot

legendLoc = 'best' -> location of the legend, defaults to “best” other options are as per matplotlib.pyplot

returnAxes = False -> returns an axes object as part of a multiplot set up, most users can ignore this option.

.plotmeListVersion() -> Same as above, but for a previous version of DOS objects where the contributions are in a list. Marked for deprecation.

. subPlot3() -> Part of plot3DOS, basically returns a pyplot.subplot(31Position) (3 rows, 1 column, plot#) for this DOS object, otherwise this is very similar to the DOS.plotme() method. SHOULD NOT BE REQUIRED BY USERS.

. writeToFile(filename = 'dens\_out.dat', header = True, columns = True, asList = False) -> Outputs the density to a file in the FIREBALL format (when defaults \*\*kwargs are used).

.writeToFileListVersion() -> previous version of writeToFile()

.makeContributionsList() -> converts contributions directory to a list, returns that list. This was for compatibility and has been flagged for deprecation.

. sumtots(other) -> Returns a sum of the Totals column of this and another DOS object. Marked for deprecation as the “+” operator now carries out this fuction.

.cat(other) -> Concatenates the contributions of other to this DOS object, recalculates the Totals column.

*global methods that apply to DOS():*

(N.B. Refer to the “supercell” documentation for some other DOS-related methods)

readDOS(atomList, directory = '.') -> Appends the DOS contributions to the atoms in atomList as read in from a directory.

atom2DOS(atom, HOMO = 999) -> Returns the DOS object for that particular atom.

atom2DOS\_list(atom, HOMO = 999) -> Deprecated. Returns a DOS object with a list, superceded by atom2DOS().

plotDOSList(densList, name = "multiDOSplot", legend = True, plotContributions = True, plotHOMO = True, plotOffset = True, xlim = [], saveFile = True) -> Marked for deprecation as plot3DOS() does a similar job in a more pythonic fashion with better functionality (such as multiHOMO), but this takes a list of 3 DOS objects and creates a stacked plot. \*\*kwargs are the same as for the .plotme method.

plot3DOS(densList, name = "multiDOSplot", legend = True, plotContributions = True, plotHOMO = True, xlim = [], saveFile = True, subLegend = False, plotOffset = False, plotTOT = False, multiHOMO = [], contriblw = 1, totlw = 2, legendLoc = 'best', subLegendLoc = 'best') -> Generates a stacked 3-DOS plot based on the list of DOS objects in densList. \*\*kwargs allow for tailoring of the plot. The multiHOMO[] list MUST contain a list of disctionaries with the same keys as the contributions in the densList (in the same order).

catDOSTots(DOSdict, passHOMO = True) -> Accepts a dictionary of DOS objects and returns a DOS object with the Totals of each dictionary object as a contribution.