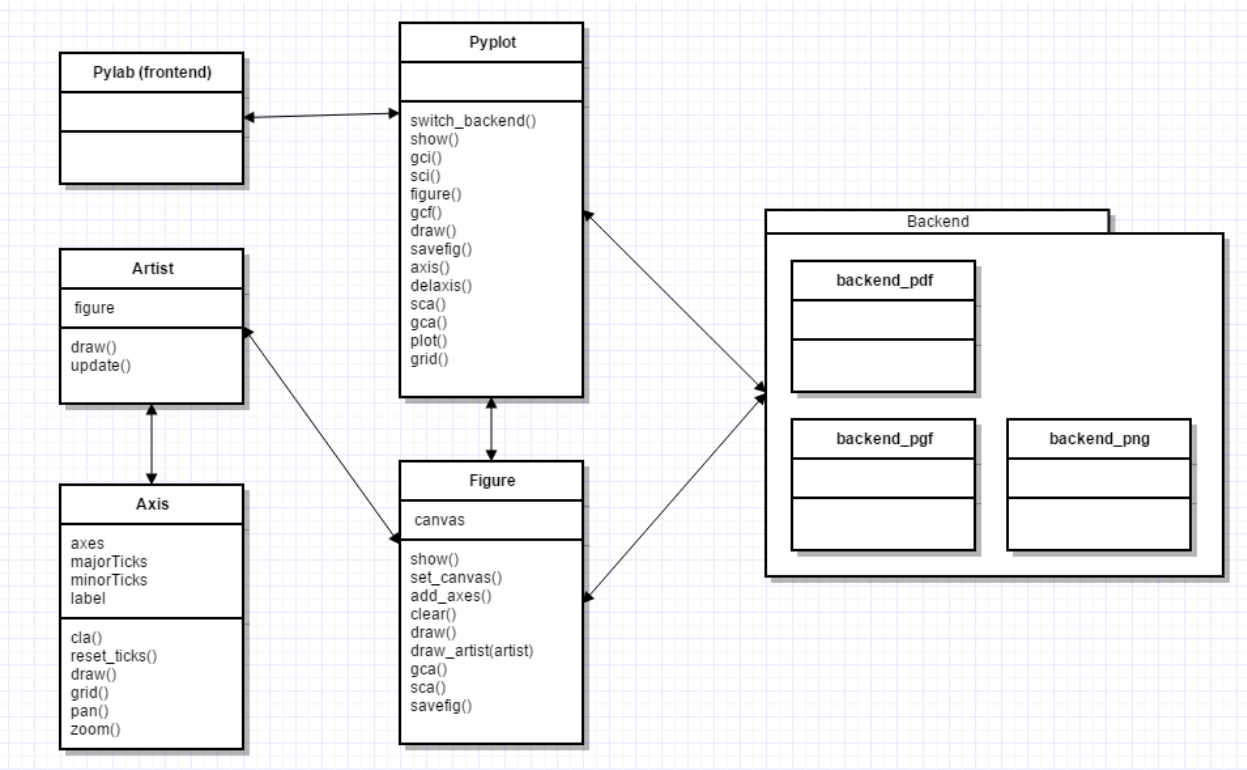
CSCD01 Deliverable 2 – Team6

Overall Architecture



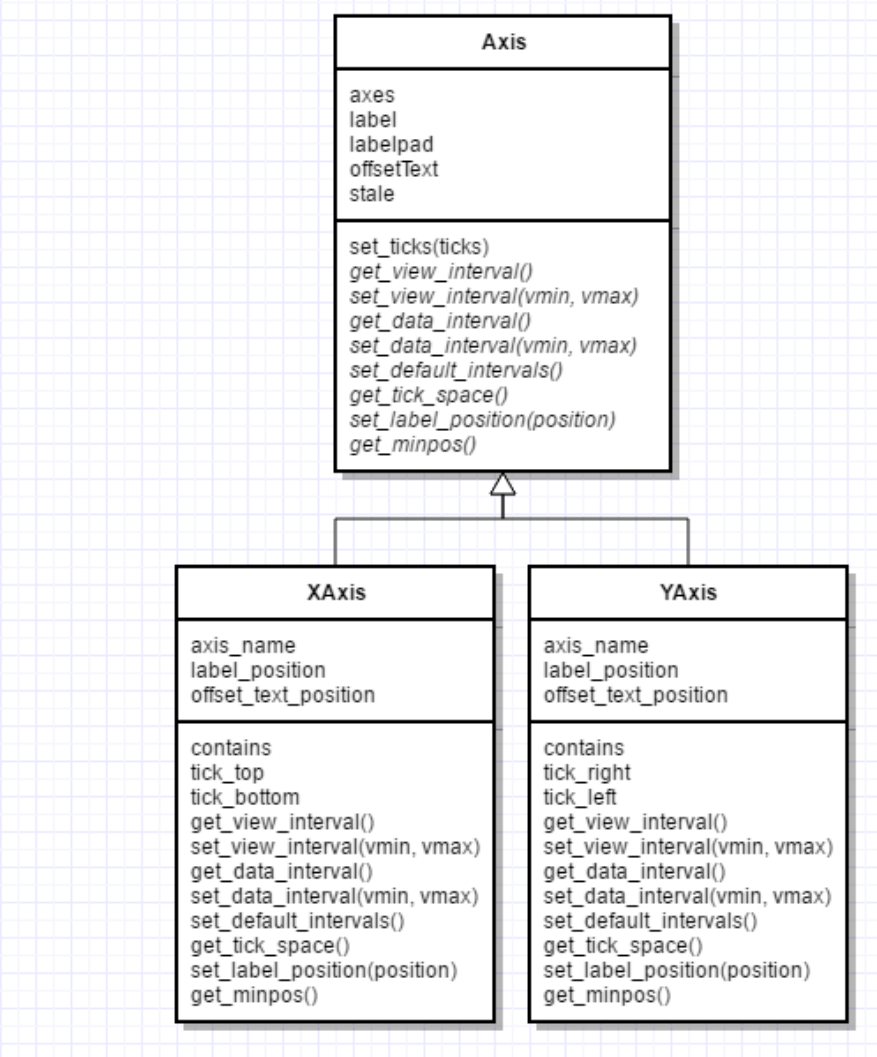
Matplotlib is split into three sections: interface, frontend, and backend. The main part of the interface is pylab (https://github.com/matplotlib/matplotlib/blob/master/lib/matplotlib/pylab.py)

which doesn't contain any important functions itself, it mainly imports functions from the frontend. The frontend contains four major classes: axis, artist, figure and pyplot. Pyplot (https://github.com/matplotlib/matplotlib/blob/master/lib/matplotlib/pyplot.py) is the main class in the frontend. It acts as the interaction layer between the interface and the backend. The job of artist (https://github.com/matplotlib/matplotlib/blob/master/lib/matplotlib/artist.py) is to draw everything which the user can see visually, including the graph, axis and frame. The main function in artist is draw, which draws all the aforementioned objects. The draw method in artist is achieved through calling the draw method of the object which is calling it. For example if an Axis object were to call draw, artist would run the draw method in axis. Axis (https://github.com/matplotlib/matplotlib/blob/master/lib/matplotlib/axis.py) is the main class for XAxis and YAxis which sets the intervals and range of the axis on which the graph is to be displayed, while also creating the labels for each axis (x and y) and the grid. Figure (https://github.com/matplotlib/matplotlib/blob/master/lib/matplotlib/figure.py) is the main screen on which the axis is drawn. The method add_axes adds an axis object which was set in sca which sets the current axis which is then displayed by show. The backend contains files such as backend_ps

(https://github.com/matplotlib/matplotlib/blob/master/lib/matplotlib/backends/backend_ps.py) and backend_pdf, (https://github.com/matplotlib/matplotlib/blob/master/lib/matplotlib/backends/backend_pdf.py) each one of these files allows matplotlib to draw the plot dependent on what the device it is being run on is. For example, pyplot contains a function called switch_backend, if the backend was set to pdf, it would allow the user to generate a pdf file from that plot.

One of the things we found interesting was how the backend was able to generate a plot for so many different file types and devices, and the efficient way in which was done. The backend would be managed by a backend manager, and then pyplot would use a single function to switch between the various types of backends.

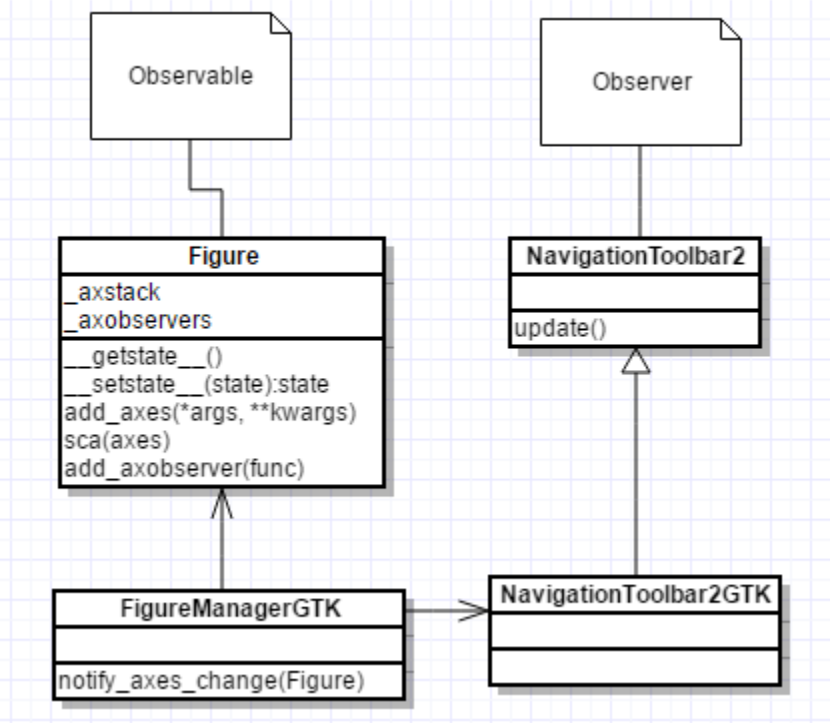
One of issues we found that needed to be improved was the amount of bidirectional dependencies, there are too many of them. The backend depends on the frontend which depends on the interface, and then the interface is also dependent on the frontend.

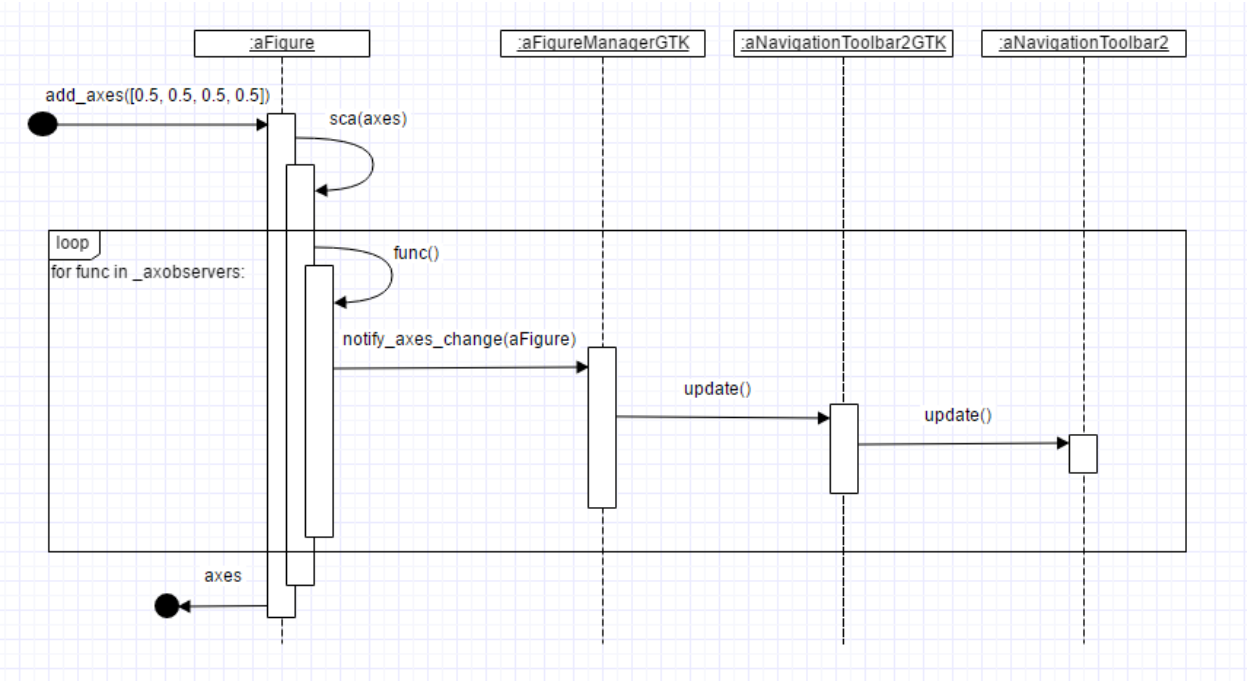


The axis object should have been abstract so that no instance of it could have been created, only XAxis and YAxis objects should be created. Another improvement would be having one class which interacts with the backend instead of multiple classes.

Design Patterns

Observer





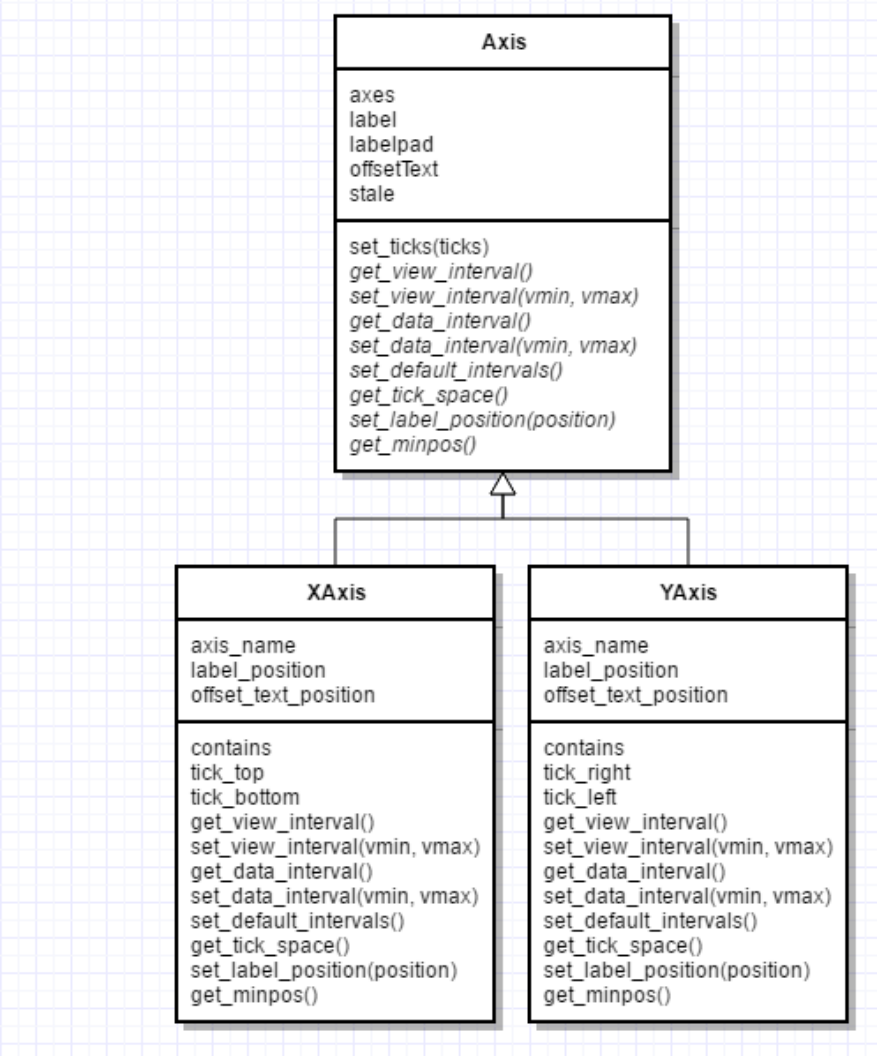
Observer Design Pattern

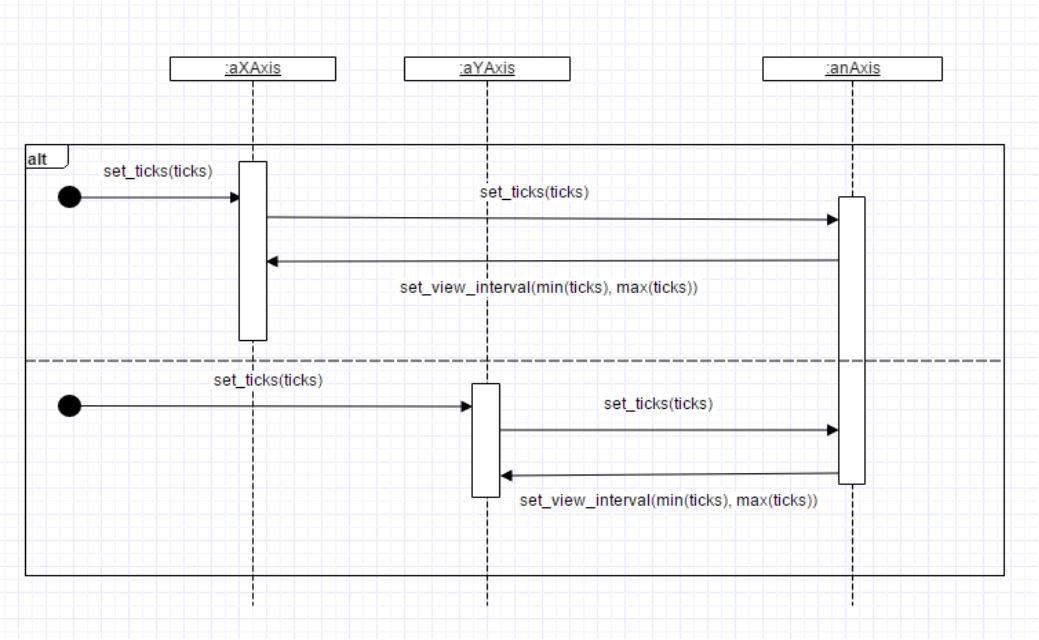
Found in:

> https://github.com/matplotlib/matplotlib/blob/master/lib/matplotlib/figure.py#L244
>
> https://github.com/matplotlib/matplotlib/blob/master/lib/matplotlib/backends/backend_gtk.py#L542
>
> https://github.com/matplotlib/matplotlib/blob/master/lib/matplotlib/backends/backend_gtk.py#L667
>
> https://github.com/matplotlib/matplotlib/blob/master/lib/matplotlib/backend_bases.py#L2712

When a FigureManager object is created, it contains a function notify_axes_change which notifies all observers anytime the axis is changed. The figure has a list of observers, and every time the axis changes, the figure notifies all its observers by running a function which is in _axobservers which tells the navigation toolbars to update. This function is within a list of functions, which are all run. An observer is added by running add_axobservers which takes in a function as an argument which is then added to the list of functions in _axobservers which will be run each time figures are notified from now on. The function designates which observer needs to be updated. The function sca takes an axis object, and sets that axis as the current axis in use.

Strategy


Axis
axes
label
labelpad
offsetText
stale
set_ticks(ticks)
get_view_interval()
set_view_interval(vmin, vmax)
get_data_interval()
set_data_interval(vmin, vmax)
set_default_intervals()
get_tick_space()
set_label_position(position)
get_minpos()
XAxis
axis_name
label_position
offset_text_position
contains
tick_top
tick_bottom
get_view_interval()
set_view_interval(vmin, vmax)
get_data_interval()
set_data_interval(vmin, vmax)
set_default_intervals()
get_tick_space()
set_label_position(position)
get_minpos()
YAxis
axis_name
label_position
offset_text_position
contains
tick_right
tick_left
get_view_interval()
set_view_interval(vmin, vmax)
get_data_interval()
set_data_interval(vmin, vmax)
set_default_intervals()
get_tick_space()
set_label_position(position)
get_minpos()



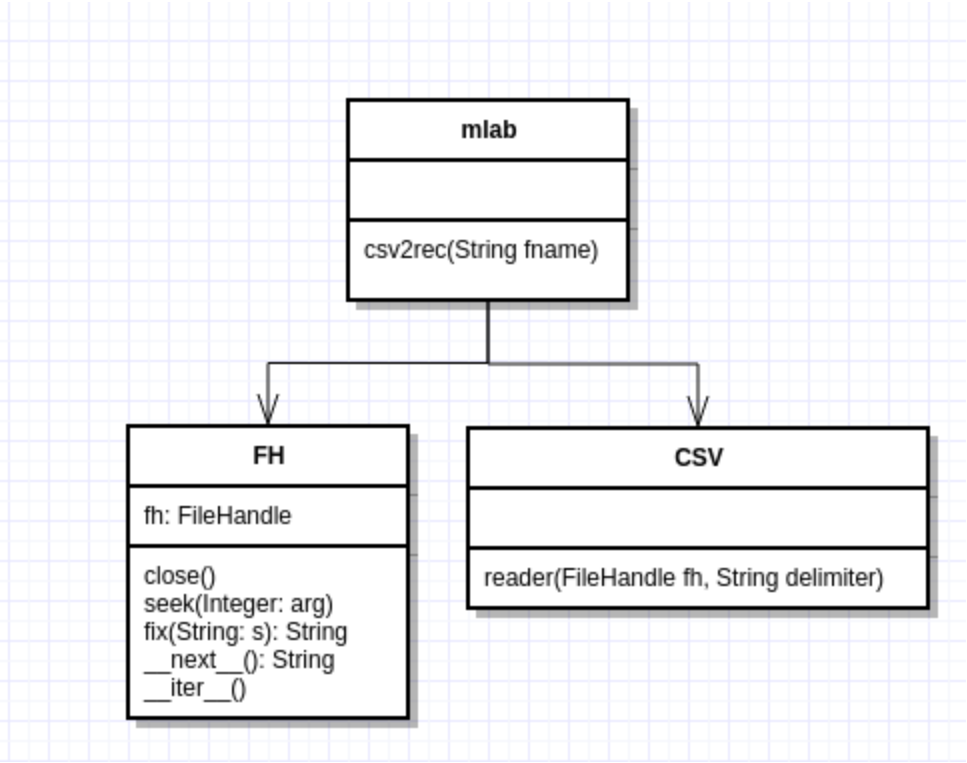
Strategy Design Pattern

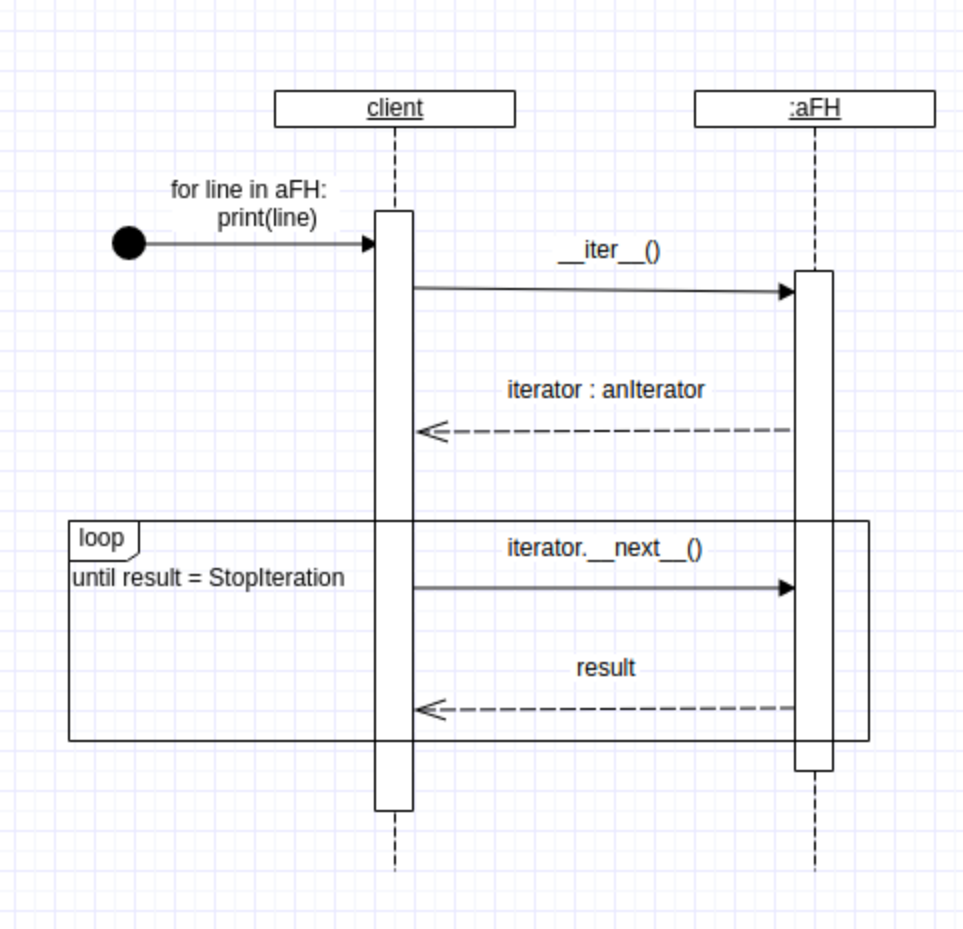
Found in:

> https://github.com/matplotlib/matplotlib/blob/master/lib/matplotlib/axis.py#L608
>
> https://github.com/matplotlib/matplotlib/blob/master/lib/matplotlib/axis.py#L1700
>
> https://github.com/matplotlib/matplotlib/blob/master/lib/matplotlib/axis.py#L2029

All of the italicized functions in Axis are unimplemented and raise errors when called, but are implemented in both XAxis and YAxis. If the instance of Axis is a XAxis object then the top sequence is run, if the instance is a YAxis then the bottom sequence is run.

Iterator


mlab
csv2rec(String fname)
FH
fh: FileHandle
close()
seek(Integer: arg)
fix(String: s): String
__next__(): String
__iter__()
CSV
reader(FileHandle fh, String delimiter)



Iterator Design Pattern

Found in:

https://github.com/matplotlib/matplotlib/blob/master/lib/matplotlib/mlab.py#L2729

mlab is a file which contains a function csv2rec which define the FH class and creates an instance of FH and calls the method in CSV, reader which takes the new instance of FH. Reader then iterates over the iterable in FH. Reader returns an iterator which is evident from the fact that later on, there is a for loop which reads each row in Reader. CSV is an external library.