# **Feature #10173: Relative linewidth and markersize**

**Feature:** [**feature request: relative linewidth and markersize #10173**](https://github.com/matplotlib/matplotlib/issues/10173)

**Estimated Hours:**

**- Explore and create a solution (8 h)**

**- Implement solution (7 h)**

**- Testing/validation (3 h)**

**- Code Review (3 h)**

**- Documentation (5 h)**

**- *Total: 26 h***

**Description:**

This feature is the option to choose from a set of strings (like "x-small", "medium", "large", etc.) when determining line widths and marker sizes. Additionally, new rcParams (lines.base.linewidth and lines.base.markersize) that affect the calculation of these relative size values should be created so that the values represented are configurable by the user.

Users should be able to set these new relative size options both by passing parameters into plotting functions and by using the existing lines.linewidth/lines.markersize properties of rcParams. Currently, the only type of parameter accepted as line widths and marker sizes are floats. The implementation of this feature would allow users to pick customizable relative sizes in additional to floats, as is the case for the existing fontsize property.

**Solution:**

[**Updated Forked Repo**](https://github.com/gravitybear/matplotlib)

In preparation for our solution, we read through the responses on the feature request, and made a quick plan for what we wanted to accomplish: adding the ability to specify the linewidth and/or markersize for their plots using quantitative string values, relative to a base value in rcParams, by converting it to a float when it’s needed.

Following that, we looked for what files would be affected; along with what methods are relevant. We found that most of the work would be done in ***lines.py***, which is responsible for creating a line, specifying its linewidth, and also markersize. It contains the methods ***set\_linewidth*** and ***set\_markersize***, which we see as a good place to convert a string argument given as the width or size to a float. We also noticed through a search that ***patches.py*** and ***collections.py*** contain similar methods named ***set\_linewidth***, and planned on adding support for this feature to those methods.

We began our solution by adding a couple new rcParam values to ***rcsetup.py*** to represent the base value, and also adding support to the existing linewidth and markersize rcParams by creating type verification functions for them.

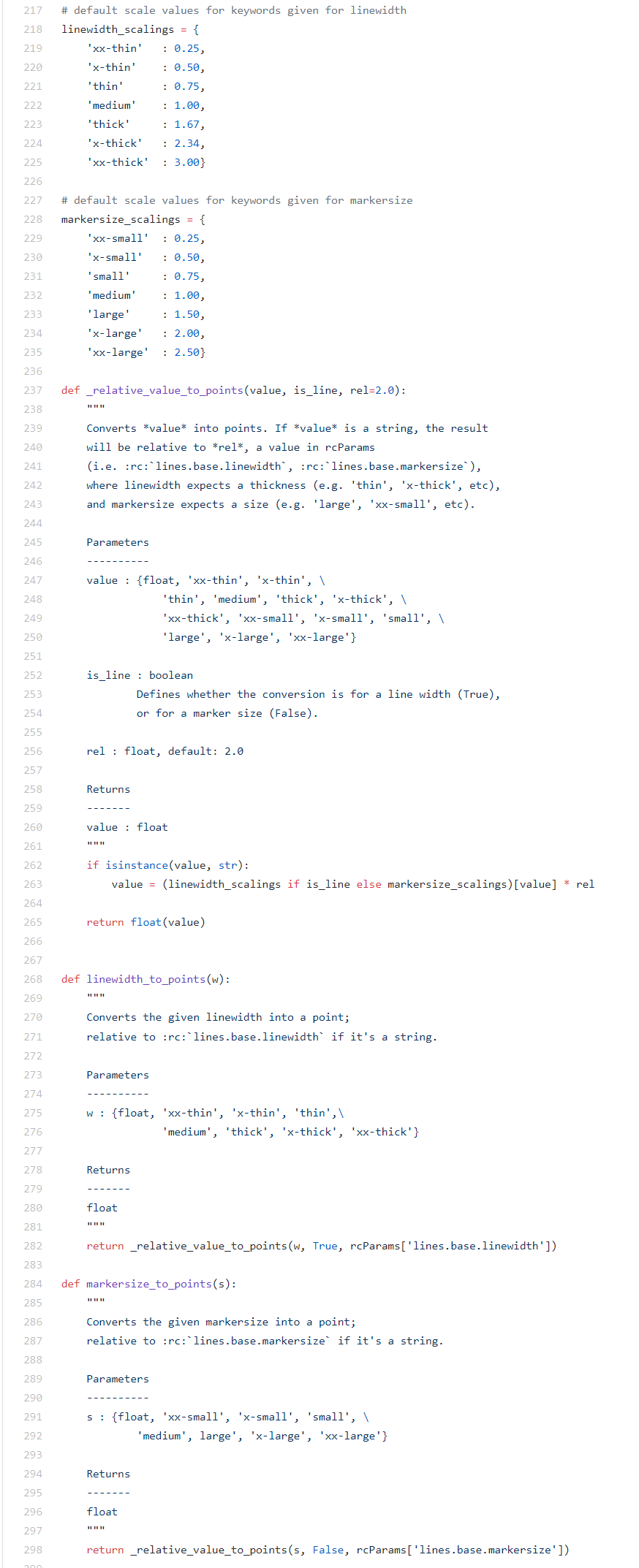
After that, we discussed what string values we should support, and went with a format similar to the names we found ***font\_manager.py*** in effort to be consistent with the existing code base.

For linewidth: ’xx-thin', 'x-thin', 'thin', 'medium', 'thick', 'x-thick', 'xx-thick'.

For markersize: 'xx-small', 'x-small', 'small', ‘medium’, 'large', 'x-large', 'xx-large'.

So, in ***lines.py,*** we added 2 dictionaries with those values as the key, and a multiplier as the value which would be multiplied with the base value rcParam we created. Following that, we created: ***\_relative\_value\_to\_points,*** which if the value is a string, it uses that to get the multiplier from the dictionary and multiplies it over the base rcParam value; returning a float. Otherwise, it just returns the value it was given. This method gets called by one of 2 other methods we created: ***linewidth\_to\_points, markersize\_to\_points***, which takes the linewidth/markersize, and returns its result. We created these methods so we wouldn’t have to repeat code that would otherwise have to specify which type of value we’re dealing with (linewidth or markersize) which are treated a little differently.

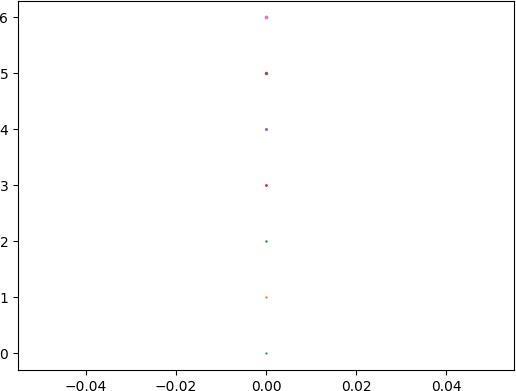
For most setter methods, we simply used ***linewidth\_to\_points*** or ***markersize\_to\_points,*** on the corresponding value to convert it into a float. The given value is only affected if it was originally a string. The method that needed another implementation was ***set\_linewidth*** in ***collections.py,*** which can be given a sequence of linewidths. So we used list comprehensions to call ***linewidth\_to\_points*** on each item in the sequence, to convert them into floats.



**Testing:**

Image comparison was used to test aspects of this new feature. Specifically, we tested that all the relative sizes for the markers as well as relative widths for the lines appeared as expected. We also tested our integration of relative widths and LineCollection by creating a LineCollection and passing in relative line widths as parameters and comparing the resulting figure to a baseline image.

The new tests are located in the fork and also in the group repository under ***GoonSquad/D5/relative\_linewidth\_markersize\_values/test/***. The tests were added to matplotlib's existing ***tests/baseline\_images/test\_lines.py*** because that's where their line and marker tests are.



*marker\_sizes.png*

|  |  |
| --- | --- |
|  |  |
| *line\_widths.py* | *line\_collection\_widths.py* |

**Integration:**

These changes fit well into the existing code base. Since the involved classes (**Line2D, Collection...**) call ***set\_linewidth*** and/or ***set\_markersize,*** rather than explicitly setting the value, there is no chance for a string value to cause an issue because it’s converted into a useable float before anything else in the code base could have the chance to use it in an invalid state. In addition, they follow an existing naming convention (similar to font sizes in ***font\_manager.py***).