CSCD01 Deliverable 5

Code Design:

The design of our solution was similar to our initial design because our initial design was correct, but the implementation was drastically different, as we found a simpler solution to our issue. In our initial solution we were going to make a shallow copy of the color map object and a deep copy of the _lut attribute so that when the color map was modified the _lut attribute of the original would not be modified. What we actually did for our solution is we imported the python copy library and used the built in deepcopy method.

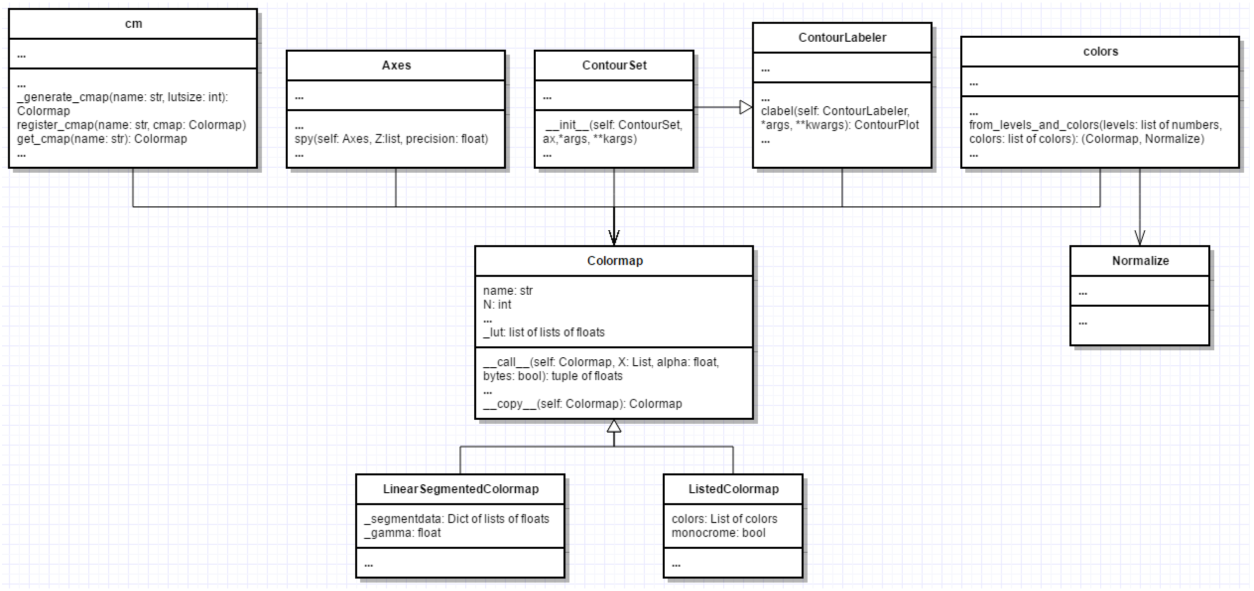
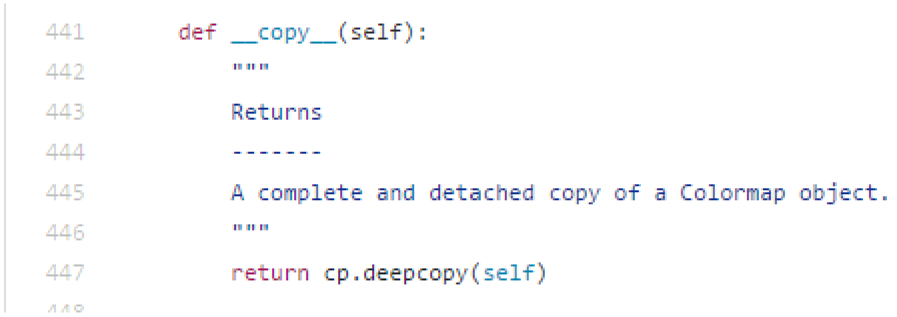

cm
...
...
_generate_cmap(name: str, lutsize: int): Colormap
register_cmap(name: str, cmap: Colormap)
get_cmap(name: str): Colormap
...
Axes
...
...
spy(self: Axes, Z:list, precision: float)
...
ContourSet
...
__init__(self: ContourSet, ax,*args, **kargs)
...
ContourLabeler
...
...
clabel(self: ContourLabeler, *args, **kwargs): ContourPlot
...
colors
...
...
from_levels_and_colors(levels: list of numbers, colors: list of colors): (Colormap, Normalize)
...
Colormap
name: str
N: int
...
_lut: list of lists of floats
__call__(self: Colormap, X: List, alpha: float, bytes: bool): tuple of floats
...
__copy__(self: Colormap): Colormap
Normalize
...
...
LinearSegmentedColormap
_segmentdata: Dict of lists of floats
_gamma: float
...
ListedColormap
colors: List of colors
monocrome: bool
...

Figure 1: The implemented design



```python
    def __copy__(self):
        """
        Returns
        -------
        A complete and detached copy of a Colormap object.
        """
        return cp.deepcopy(self)
```

Figure 2: The changed copy function (https://github.com/CSCD01-Winter2017/team06-Project/blob/master/matplotlib/lib/matplotlib/colors.py)

Acceptance Tests:

Acceptance tests can be found at https://github.com/CSCD01-Winter2017/team06-Project/tree/master/Deliverable5/acceptence_tests

Since there were no significant changes to our design, the acceptance tests we initially had already covered any potential usage of our feature.

Test Name	Test Description	Expected Output
test_no_copy_save.py	Tests that implementing copy for the colormap object did not affect how images are saved.	The image file test_no_copy_save.png should look identical to original.png. The top left square should be white, while the other squares should be varying shades of red with the darkest being in the bottom right corner.
test_no_copy_modify_save.py	Tests that implementing copy for the colormap object did not affect how a colormap is modified.	The image file test_no_copy_modify_save-unmodified.png should look identical to original.png. The top left square should be white, while the other squares should be varying shades of red with the darkest being in the bottom right corner. The image file test_no_copy_modify_save-modified.png should look identical to modified.png. The top left square should be blue, while the other squares should be varying shades of red with the darkest being in the bottom right corner.
test_copy_save_old.py	Tests that using the new copy method does not affect the original colormap.	The image file test_copy_save_old-original.png should look identical to original.png. The top left square should be white, while the other squares should be varying shades of red with the darkest being in the bottom right corner. The image file test_copy_save_old-notcopy.png should look identical to original.png. The top left square should be white, while the other squares should be varying shades of red with the darkest being in the bottom right corner.

test_copy_save_new.py	Tests that using the new copy method does not modified the copied colormap.	The image file test_copy_save_new-original.png should look identical to original.png. The top left square should be white, while the other squares should be varying shades of red with the darkest being in the bottom right corner. The image file test_copy_save_new-copy.png should look identical to original.png. The top left square should be white, while the other squares should be varying shades of red with the darkest being in the bottom right corner.
test_copy_modify_save_new.py	Tests that using the copy method and then modifying the copy results in the original remaining the same and the copy being modified.	The image file test_copy_modify_save_new-original.png should look identical to original.png. The top left square should be white, while the other squares should be varying shades of red with the darkest being in the bottom right corner. The image file test_copy_modify_save_new-new_modified_copy.png should look identical to modified.png. The top left square should be blue, while the other squares should be varying shades of red with the darkest being in the bottom right corner.
test_copy_modify_save_multiple.py	Tests that copying the colormap, then modifying the colormap and saving the original doesn't modify the original. Also that saving the new copy and then the old copy will result in the copy being changed and not the original.	The image file test_copy_modify_save_multiple-original.png should look identical to original.png. The top left square should be white, while the other squares should be varying shades of red with the darkest being in the bottom right corner. The image file test_copy_modify_save_multiple-unmodified.png should look identical to original.png. The top left square should be white, while the other squares should be varying shades of red with the darkest being in the bottom right corner.

		The image file test_copy_modify_save_multiple-modified.png should look identical to modified.png. The top left square should be blue, while the other squares should be varying shades of red with the darkest being in the bottom right corner. The image file test_copy_modify_save_multiple-unmodified2.png should look identical to original.png. The top left square should be white, while the other squares should be varying shades of red with the darkest being in the bottom right corner.

User Guide:

Copying a colormap gives the user an easier alternative to fully creating a new colormap and modifying it in the same way as the desired one. Copying also ensures that two colormaps are completely detached from one another.

For example, this is how you would create and make a copy of a colormap

```
import matplotlib.pyplot as plt
import copy
import numpy as np

data = np.ma.masked_array([[1,2,3],[2,3,4]], mask=[[1,0,0],[0,0,0]])
cm1 = plt.cm.Reds

cm2 = copy.copy(cm1)
plt.imshow(data,cmap=cm2)
plt.savefig("image.png")
```

Unit Tests:

Unit test can be found at https://github.com/CSCD01-Winter2017/team06-Project/blob/master/Deliverable5/unit_tests/colormap_unittests.py

Since our original design was correct, our initial unit tests already covered all potential cases, so there was no need to add any more.